WebSphere® IBM WebSphere Process Server for Multiplatforms
Versión 7.0.0

Desarrollo y despliegue de módulos

IBM®

WebSphere® IBM WebSphere Process Server for Multiplatforms
Versión 7.0.0

Desarrollo y despliegue de módulos

IBM®

Abril de 2010

Esta edición se aplica a la versión 7, release 0, modificación 0 de WebSphere Process Server for Multiplatforms (número de producto 5724-L01) y a todos los releases y las modificaciones subsiguientes hasta que se indique lo contrario en nuevas ediciones.

Para enviar comentarios sobre este documento, envíe un mensaje de correo electrónico a doc-comments@us.ibm.com. Esperamos sus comentarios.

Contenido

Tablas

Parte 1. Desarrollo de aplicaciones

Capítulo 1. Desarrollo de soluciones de gestión de procesos de negocio

Esta sección describe los fundamentos del modelo de programación de la gestión de procesos de negocio (BPM). Presenta el estándar SCA (Service Component Architecture) y describe los patrones relacionados con la integración empresarial.

BPM es la disciplina que permite a las empresas identificar, consolidar y optimizar procesos de negocio. El objetivo es mejorar la productividad y maximizar la eficacia organizativa. El interés en BPM se ha intensificado a medida que las empresas se fusionan y consolidan, y a medida que acumulan una patrimonio de activos de información muy dispares. Dichos activos, a menudo, no disponen de coherencia ni coordinación, hecho que genera “islas de información”.

BPM tiene sólidos lazos con la arquitectura orientada a servicios (SOA). En función de la naturaleza de la empresa y el alcance de las necesidades de integración, BPM plantea distintos requisitos para los departamentos TI. Algunos proyectos se ocupan sólo de unos pocos aspectos, mientras que algunos proyectos mayores pueden englobar muchos de estos requisitos: He aquí algunos de los aspectos más comunes de los proyectos BPM:

- **Integración de aplicaciones**: es un requisito común. La complejidad de proyectos de integración de aplicaciones varía desde casos sencillos, en los que necesita asegurarse de que pocas aplicaciones puedan compartir información, hasta situaciones más complejas, en las que las transacciones y los intercambios de datos se deben reflejar de forma simultánea en varias aplicaciones de programa de fondo. La integración de aplicaciones compleja requiere, a menudo, la gestión de unidades de trabajo complejas, así como realizar tareas de transformación y correlación.
- **Automatización de procesos**: es otro aspecto clave que garantiza que las actividades llevadas a cabo por un individuo o una organización desencadenen, de forma sistemática, actividades importantes en cualquier otro lugar. Esto garantiza la finalización satisfactoria del proceso empresarial global. Por ejemplo, cuando una empresa contrata a un empleado, debe actualizarse la información de gestión de nóminas, el departamento de seguridad debe llevar a cabo las acciones necesarias apropiadas, y se le deben proporcionar al empleado las herramientas necesarias, etc. Ciertas actividades de un proceso pueden capturar entradas e interacciones efectuadas por los usuarios, mientras que otras pueden invocar scripts en sistemas "de fondo" y demás servicios del entorno.
- **Conectividad**: es un aspecto abstracto, aunque muy importante, tanto a nivel de empresa como en términos de los business partner. La conectividad hace referencia al flujo de información entre organizaciones o empresas y la capacidad de acceder a servicios TI distribuidos.

Algunos de los desafíos técnicos de las implementaciones de la integración empresarial se pueden resumir tal como se indica a continuación:

- Tratar con diferentes formatos de datos y, por tanto, no poder llevar a cabo una transformación de datos eficiente.
- Tratar con distintos protocolos y mecanismos para acceder a los servicios de TI que se han desarrollado utilizando tecnologías diferentes.
- Orquestar diferentes servicios de TI que se pueden distribuir geográficamente, o que diferentes organizaciones pueden ofrecer.

- Proporcionar normas y mecanismos para clasificar y gestionar los servicios que están disponibles (gobierno).

Como tal, BPM engloba muchos de los temas y elementos que también son comunes a SOA. La visión de IBM® de BPM se basa en muchos de los mismos conceptos fundacionales que se encuentran en SOA. Una de las consecuencias inmediatas de esta visión es que las soluciones BPM requieren varios productos para su realización. IBM proporciona un portafolio de herramientas y plataformas de tiempo de ejecución que dan soporte a los diferentes tipos de fases y aspectos operacionales.

Para parafrasear la visión de IBM de BPM, permite a las empresas definir, crear, fusionar, consolidar y agilizar procesos de negocio que utilizan las aplicaciones que se ejecutan en una infraestructura de TI SOA. El trabajo de BPM se basa realmente en los roles. A gran nivel, implica modelar, desarrollar, dirigir, gestionar y supervisar las aplicaciones de proceso de negocio. Con la ayuda de las herramientas y los procedimientos adecuados, le permite automatizar los procesos de negocio en los que intervienen personas y sistemas heterogéneos, tanto dentro como fuera de la empresa. Uno de los aspectos clave de BPM es la capacidad de optimizar las operaciones de negocio de forma que sean lo suficientemente eficientes, escalables, fiables y flexibles para manejar los cambios.

BPM requiere herramientas de desarrollo, servidores de tiempo de ejecución, herramientas de supervisión, un repositorio de servicios, kits de herramientas y plantillas de proceso. Puesto que existen tantos aspectos relacionados con BPM, encontrará que debe utilizar más de una herramienta de desarrollo para desarrollar una solución. Estas herramientas permiten a los desarrolladores de integración ensamblar soluciones empresariales complejas. Un servidor es un motor empresarial de alto rendimiento o un contenedor de servicios que ejecuta aplicaciones complejas. El departamento de gestión siempre quiere saber quién está haciendo qué en la organización, y aquí es donde entran en juego las herramientas de supervisión. A medida que las empresas crean estos procesos o servicios empresariales, la gestión, la clasificación y el almacenamiento de dichos servicios resulta crucial. El encargado de desempeñar esta función es un repositorio de servicios. A menudo se necesitan kits de herramientas específicos para crear las partes especializadas de la solución como, por ejemplo, los conectores o adaptadores para los sistemas existentes.

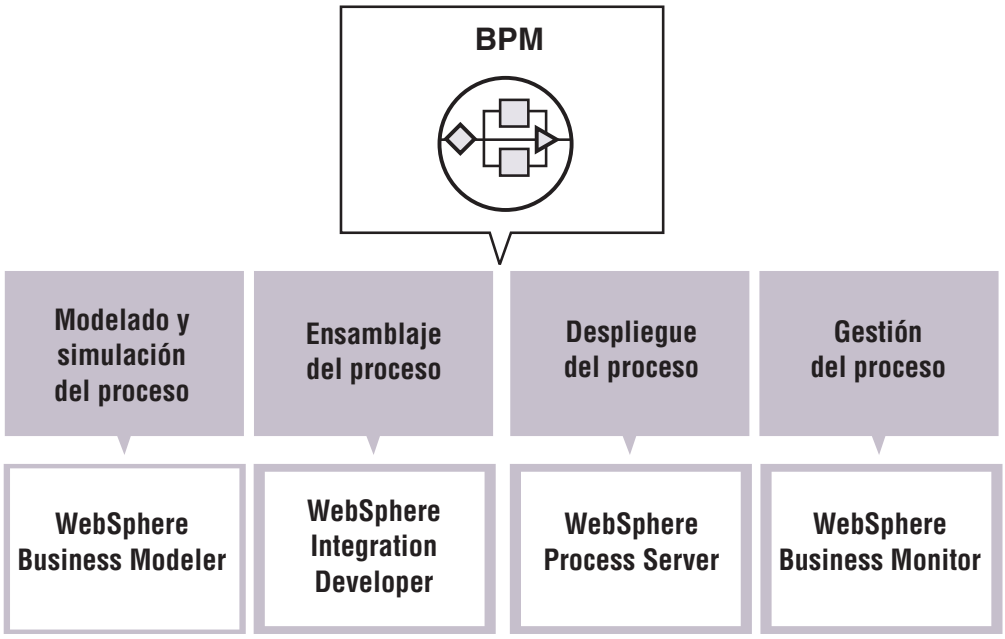


Figura 1. Las herramientas de IBM abarcan el ciclo de vida completo de BPM, lo que le permite modelar, ensamblar, desplegar y gestionar procesos.

BPM no se basa en un único producto. Implica a casi todas las personas y todos los aspectos empresariales involucrados en una organización y en varias organizaciones. BPM engloba muchos de los servicios y elementos de la arquitectura de referencia SOA.

Para obtener más detalles sobre estos conceptos, junto con ejemplos de programación, consulte:

- *WebSphere Business Integration Primer: Process Server, BPEL, SCA, and SOA*, IBM Press, 2008.
- *Getting Started with IBM WebSphere Process Server and IBM WebSphere Enterprise Service Bus Part 1: Development*, IBM Redbooks, SG24-7608-00, June 2008.
- *IBM Business Process Management Reviewer's Guide*, IBM Redpapers, REDP-4433-01, abril 2009.

Arquitectura y patrones de la integración empresarial

Un proyecto de gestión empresarial típico implica coordinar varios activos de TI diferentes que, potencialmente, se ejecutan en distintas plataformas y que se han desarrollado en momentos distintos utilizando tecnologías diferentes. Un desafío técnico muy importante es ser capaz de manipular e intercambiar información, fácilmente, con conjunto variado de componentes. Esta tarea se puede llevar a cabo mejor mediante el modelo de programación utilizado para desarrollar soluciones de integración empresarial.

En esta sección se presenta el estándar SCA (Service Component Architecture) y describe los patrones relacionados con la integración empresarial. Da la sensación de los patrones invaden nuestras vidas. Existen patrones para coser, patrones de aprendizaje para niños, patrones de construcción de casas, patrones para tallar madera, patrones de vuelo, patrones de vientos, patrones de prácticas en medicina, patrones de compras de los clientes, patrones de flujos de trabajo, patrones de diseño en informática y muchos más.

Se ha demostrado, satisfactoriamente, que los patrones realmente sirven de ayuda a los diseñadores y desarrolladores de soluciones. Por tanto, no es de extrañar que ahora tengamos patrones de integración empresarial y patrones de integración comercial. Existe una amplia gama de patrones que son aplicables a la integración empresarial, incluidos los patrones para el direccionamiento de petición y respuesta, los patrones de canal (por ejemplo, publicación/suscripción), y muchos más. Los patrones abstractos proporcionan una plantilla para resolver una determinada categoría de problemas, mientras que los patrones concretos proporcionan indicaciones más especificas sobre cómo implementar una solución específica. Esta sección se concentra en los patrones relacionados con la invocación de datos y servicios, que se encuentran en la base del modelo de programación de la estrategia del software de IBM para WebSphere Business Integration.

Escenarios de integración empresarial

Las empresas tienen varios sistemas de software distintos que utilizan para dirigir su negocio. Además, tienen sus propias maneras de integrar dichos componentes de empresa.

Los dos escenarios de Business Integration que más prevalecen son los siguientes:

- **Intermediario de integración**: en este escenario, la solución Business Integration actúa como un intermediario localizado entre varias aplicaciones de programa de fondo. Por ejemplo, es posible que tenga que asegurarse de que cuando un cliente efectúe un pedido mediante la aplicación de gestión de pedidos en línea, la transacción actualice la información relevante en la aplicación "de fondo" de gestión de relaciones con los clientes (CRM). En este escenario, la solución de integración necesita poder capturar y, posiblemente, transformar la información necesaria procedente de la aplicación de gestión de pedidos, e invocar los servicios adecuados en la aplicación de CRM.
- **Automatización de procesos**: en este escenario, la solución de integración actúa como el nexo de unión entre diferentes servicios de TI que, de otra forma, no estarían relacionados entre sí. Por ejemplo, cuando una empresa contrata un empleado, es necesario que se lleve a cabo la secuencia de acciones siguientes:
  - La información del empleado se añade al sistema de gestión de nóminas.
  - Se debe garantizar al empleado el acceso físico a las instalaciones de la empresa, y se le debe proporcionar un identificador.
  - Es posible que la empresa necesite asignar un conjunto de activos físicos al empleado (espacio en la oficina, un equipo informático, etc).
  - El departamento de TI necesita crear un perfil de usuario para el empleado, y otorgarle acceso a una serie de aplicaciones.

    La automatización de este proceso también es un caso de uso común en un escenario de integración empresarial. En este escenario, la solución implementa un flujo automatizado que se desencadena cuando se añade el empleado al sistema de gestión de nóminas. Después, el flujo desencadena los otros pasos creando los elementos de trabajo para las personas que son responsables de efectuar las acciones o llamando a los servicios apropiados.

En ambos escenarios, la solución de integración debe:

1. Trabajar con fuentes de información dispares y formatos de datos distintos, y poder convertir la información entre formatos diferentes.
2. Poder invocar distintos servicios utilizando de forma potencial diferentes mecanismos y protocolos de invocación.

Roles, productos, y desafíos técnicos

Los proyectos de integración empresarial que terminan en éxito dependen de la mezcla de roles de desarrollo especializados, técnicas de programación y suites de herramientas.

Los proyectos de integración empresarial requieren algunos ingredientes básicos:

- Una clara separación de los roles en la organización del desarrollo para promover la especialización que, normalmente, mejora la calidad de los componentes individuales que se desarrollan.
- Un modelo de objeto empresarial (OE) común que permite representar la información empresarial en un modelo lógico común.
- Un modelo de programación que separa, de forma muy estricta, las interfaces de las implementaciones, y que da soporte a un mecanismo de invocación de servicios genérico que es totalmente independiente de la implementación, y que sólo implica interactuar con las interfaces.
- Un conjunto integrado de herramientas y productos que da soporte a los roles de desarrollo y que conserva la separación que existe entre los mismos.

En las secciones siguientes se explican cada uno de estos ingredientes.

Una clara separación de roles

Un proyecto de integración empresarial requiere personas que se agrupen en cuatro roles colaborativos, pero separados claramente:

- **Analista empresarial**: los analistas empresariales son expertos de dominio responsables de capturar los aspectos empresariales de un proceso y de crear un modelo de proceso que represente, adecuadamente, el proceso en sí mismo. Su objetivo es optimizar el rendimiento financiero de un proceso. Los analistas empresariales no se dedican a los aspectos técnicos de la implementación de procesos.
- **Desarrollador de componentes**: los desarrolladores de componentes son responsables de implementar servicios y componentes individuales. Su objetivo es la tecnología específica utilizada para la implementación. Este rol requiere unos profundos conocimientos sobre programación.
- **Especialista de integración**: este rol, relativamente nuevo, describe la persona que es responsable de ensamblar un conjunto de componentes existentes en una solución de integración empresarial mayor. Los desarrolladores de integración no necesitan conocer los detalles técnicos de cada uno de los componentes y servicios que reutilizan y conectan juntos. Lo ideal es que los desarrolladores de integración sólo se preocupen de comprender las interfaces de los servicios que están ensamblando. Los desarrolladores de integración deben basarse en las herramientas de integración para el proceso de ensamblaje.
- **Desplegador de soluciones**: los desplegadores de soluciones y los administradores se encargan de que las soluciones de integración empresarial estén disponibles para los usuarios finales. Lo ideal es que el objetivo principal de un desplegador de soluciones sea el enlace de una solución con los recursos físicos que ya están preparados para que pueda funcionar (bases de datos, gestores de colas, etc.) y que no sea tener que comprender profundamente los detalles internos de una solución. El centro de atención del desplegador de soluciones es la calidad de servicio (QoS).

Un modelo de objeto de negocio común

Tal como se ha explicado anteriormente, entre los aspectos clave de un proyecto de integración empresarial se incluyen la capacidad de coordinar la invocación de varios componentes y la capacidad de manejar el intercambio de datos entre los mismos. En particular, distintos componentes pueden utilizar técnicas diferentes para representar elementos de negocio como, por ejemplo, los datos de un pedido, información de un cliente, etc. Por ejemplo, es posible que tenga que integrar una aplicación Java™ que utiliza la entidad Enterprise Java Beans (EJB) para representar los elementos empresariales y una aplicación de herencia que organice la información en formato COBOL Copybook. Por tanto, una plataforma cuyo objetivo sea simplificar la creación de soluciones de integración también debe proporcionar una manera genérica de representar los elementos empresariales, aparte de las técnicas utilizadas por los sistemas "de fondo" de manejo de datos. Este objetivo se logra en WebSphere Process Server y en WebSphere Enterprise Service Bus gracias a la *infraestructura de objeto empresarial*.

La infraestructura de objeto empresarial permite a los desarrolladores utilizar los esquemas XML para definir la estructura de datos empresariales, y acceder y manipular las instancias de dichas estructuras de datos (objetos empresariales), a través de XPath o del código Java. La infraestructura de objeto empresarial se basa en el estándar SDO (Service Data Object).

El modelo de programación SCA (Service Component Architecture)

El modelo de programación SCA representa la base de cualquier solución que deba desarrollarse en WebSphere Process Server y en WebSphere Enterprise Service Bus. SCA proporciona a los desarrolladores un modo de encapsular las implementaciones de servicios en componentes reutilizables. Permite definir interfaces, implementaciones y referencias independientemente de la tecnología que se utilice. Este enfoque ofrece la oportunidad de enlazar los elementos a cualquier tecnología que elija. También existe un modelo de programación de cliente SCA que permite invocar dichos componentes. En particular, permite que las infraestructuras de tiempo de ejecución basadas en Java interactúen con tiempos de ejecución que no sean Java. SCA utiliza objetos empresariales como los elementos de datos para la invocación de servicio.

Herramientas y productos

IBM WebSphere Integration Developer es el entorno de desarrollo integrado que tiene todas las herramientas necesarias para crear y componer soluciones de integración empresarial basadas en las tecnologías que se acaban de mencionar. Estas soluciones se despliegan, normalmente, en WebSphere Process Server o, en algunos casos, en WebSphere Enterprise Service Bus.

Capítulo 2. Enlaces

En la base de la arquitectura orientada a servicios se encuentra el concepto del *servicio*, una unidad de funcionalidad que se ejecuta mediante una interacción entre dispositivos de cálculo. Una *exportación* define la interfaz externa (o punto de acceso) de un módulo, de forma que los componentes SCA (Service Component Architecture) del módulo pueden proporcionar sus servicios a los clientes externos. Una *importación* define una interfaz para los servicios fuera de un módulo, de forma que los servicios puedan invocarse desde dentro del módulo. Los *enlaces* específicos del protocolo se utilizan con las importaciones y exportaciones para especificar el medio de transporte de los datos dentro o fuera del módulo.

Exportaciones

Los clientes externos pueden invocar componentes SCA en un módulo de integración a través de una amplia variedad de protocolos (por ejemplo, HTTP, JMS, MQ y RMI/IIOP), con datos en distintos formatos (por ejemplo, XML, CSV, COBOL y JavaBean). Las exportaciones son componentes que reciben estas solicitudes de orígenes externos y, a continuación, invocan los componentes de WebSphere Process Server utilizando el modelo de programación SCA.

Por ejemplo, en la siguiente figura, una exportación recibe una petición a través del protocolo HTTP de una aplicación de cliente. Los datos se transforman en un objeto empresarial, con el formato utilizado por el componente SCA. A continuación, el componente se invoca con dicho objeto de datos.

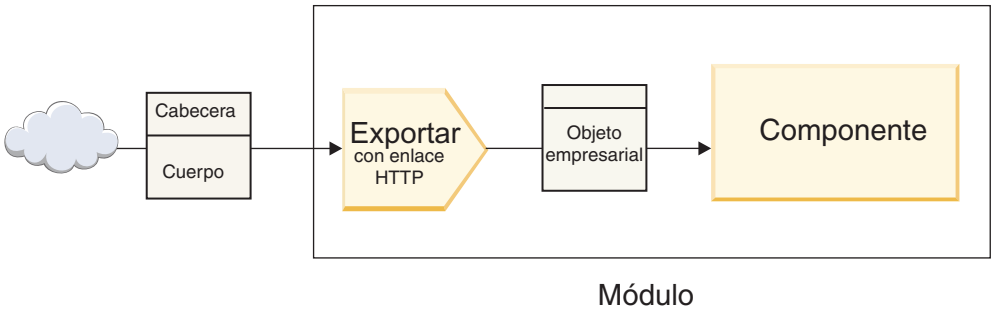


Figura 2. Una exportación con el enlace HTTP

Importaciones

Un componente SCA puede desear invocar un servicio externo no SCA que espere datos en otro formato. El componente SCA utiliza una importación para invocar el servicio externo utilizando el modelo de programación SCA. A continuación, la importación invoca el servicio de destino de la forma que espera el servicio.

Por ejemplo, en la siguiente figura, la importación envía una petición desde un componente SCA a un servicio externo. El objeto empresarial, que tiene el formato utilizado por el componente SCA, se transforma en el formato esperado por el servicio y se invoca el servicio.

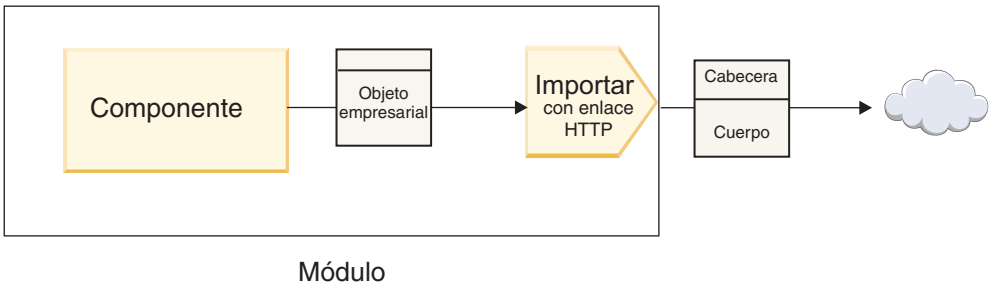


Figura 3. Una importación con el enlace HTTP

Lista de enlaces

Utilice WebSphere Integration Developer para generar un enlace para una importación o una exportación, y para configurar el enlace. Los tipos de enlaces que hay disponibles se describen en la lista siguiente.

- SCA

  El enlace SCA, que es el valor por omisión, permite al servicio comunicarse con servicios en otros módulos SCA. Utilice una importación con un enlace SCA para acceder a un servicio en otro módulo SCA. Utilice una exportación con un enlace SCA para ofrecer un servicio a otros módulos SCA.

- Servicio Web

  Un enlace de servicio Web permite acceder a un servicio externo mediante mensajes SOAP interoperables y calidades de servicio. También puede utilizar los enlaces de servicio web para incluir adjuntos como parte del mensaje SOAP.

  El enlace de servicio Web puede utilizar un protocolo de transporte SOAP/HTTP (SOAP a través de HTTP) o SOAP/JMS (SOAP a través de JMS). Independientemente del transporte (HTTP o JMS) utilizado para transportar los mensajes SOAP, los enlaces de servicio web siempre manejan las interacciones de petición/respuesta de forma síncrona.

- HTTP

  El enlace HTTP le permite acceder a un servicio externo mediante el protocolo HTTP, donde se utilicen mensajes que no sean SOAP, o donde se requiera acceso HTTP directo. Este enlace se utiliza cuando está trabajando con servicios Web basados en el modelo HTTP (es decir, servicios que utilizan operaciones de interfaz HTTP conocidas como, por ejemplo, GET, PUT, DELETE, etc.).

- Enterprise JavaBeans™ (EJB)

  Los enlaces EJB permiten a los componentes SCA interactuar con los servicios proporcionados por la lógica empresarial Java EE que se ejecuta en un servidor Java EE.

- EIS

  El enlace EIS (sistema de información de empresa), cuando se utiliza con un adaptador de recursos JCA, permite acceder a los servicios en un sistema de información de empresa o hacer que los servicios estén disponibles para el EIS.

- Enlaces JMS

  Los enlaces JMS (Java Message Service), JMS genéricos y MQ JMS (WebSphere MQ JMS) se utilizan para las interacciones con los sistemas de mensajería, donde la comunicación asíncrona mediante colas de mensajes es fundamental para la fiabilidad.

  Una exportación con uno de los enlaces JMS observa una cola para ver la llegada de un mensaje y envía la respuesta de forma asíncrona, si la hay, a la

cola de respuesta. Una importación con uno de los enlaces JMS crea y envía un mensaje a la cola JMS y observa una cola para ver la llegada de la respuesta, si la hay.

  – JMS

    El enlace JMS permite acceder al proveedor JMS incorporado de WebSphere.

  – JMS genérico

    El enlace JMS genérico permite acceder a un sistema de mensajería de un proveedor que no es IBM.

  – MQ JMS

    El enlace MQ JMS permite acceder al subconjunto JMS de un sistema de mensajería de WebSphere MQ. Utilice este enlace cuando el subconjunto JMS de funciones sea suficiente para la aplicación.

- MQ

  El enlace de WebSphere MQ permite la comunicación con las aplicaciones MQ nativas, con lo cual se pueden llevar a la infraestructura SOA (arquitectura orientada a servicios) y se proporciona acceso a la información de cabecera específica de MQ. Utilice este enlace cuando necesite utilizar funciones nativas MQ.

Visión general de los enlaces de exportación e importación

Una exportación permite crear servicios en un módulo de integración disponible para los clientes externos y una importación permite que los componentes SCA de un módulo de integración invoquen servicios externos. El enlace asociado con la exportación o la importación especifica la relación entre los mensajes de protocolo y los objetos empresariales. También especifica el modo en que se seleccionan las operaciones y los errores.

Flujo de información a través de una exportación

Una exportación recibe una petición, que está dirigida para el componente al cual está conectada la exportación, a través de un transporte específico determinado por el enlace asociado (por ejemplo, HTTP).

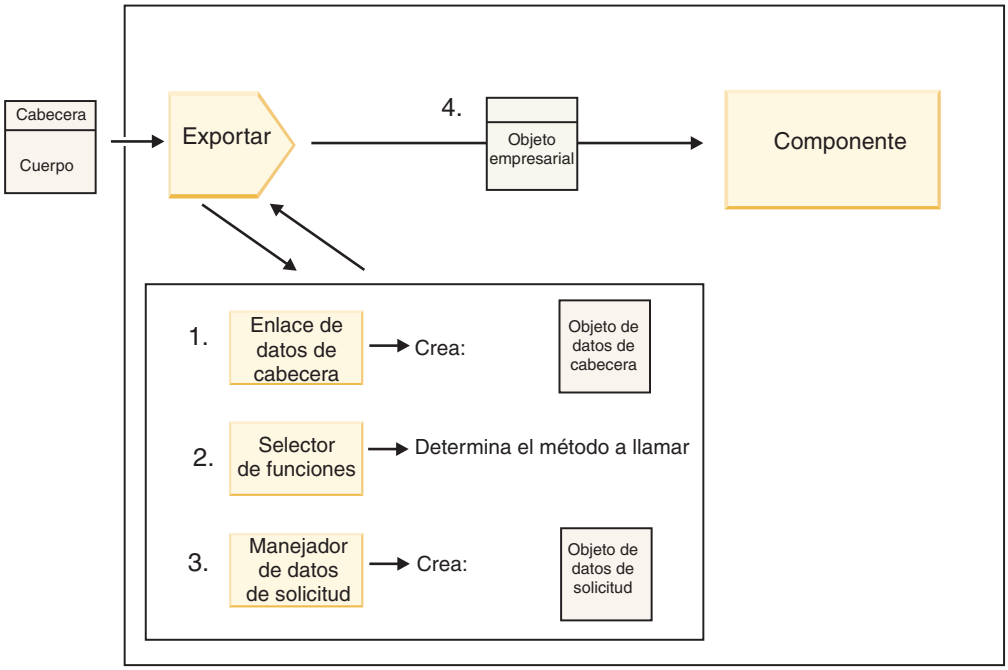


Figura 4. Flujo de una petición a través de la exportación a un componente

Cuando la exportación recibe la solicitud, se produce la secuencia de sucesos siguiente:

1. Sólo para los enlaces de WebSphere MQ, el enlace de datos de cabecera transforma la cabecera de protocolo en un objeto de datos de cabecera.
2. El selector de función determina el nombre de método nativo a partir del mensaje de protocolo. La configuración de exportación correlaciona el nombre del método nativo con el nombre de una operación efectuada en la interfaz de la exportación.
3. El manejador de datos de petición o el enlace de datos en el método transforma la petición en un objeto empresarial de petición.
4. La exportación invoca el método de componente con el objeto empresarial de solicitud.
   - El enlace de exportación HTTP, el enlace de exportación de servicio web y el enlace de exportación EJB invocan el componente SCA de forma síncrona.
   - Los enlaces de exportación JMS, JMS genéricos, MQ JMS, y de WebSphere MQ invocan el componente SCA de forma asíncrona.

Tenga en cuenta que una exportación puede propagar las cabeceras y las propiedades de usuario que recibe a través del protocolo, si se ha habilitado la propagación de contexto. Los componentes que están conectados con la exportación pueden acceder a estas cabeceras y propiedades de usuario. Para obtener más información, consulte el tema “Propagación” en el Centro de información de WebSphere Integration Developer.

Si es una operación bidireccional, el componente devuelve una respuesta.

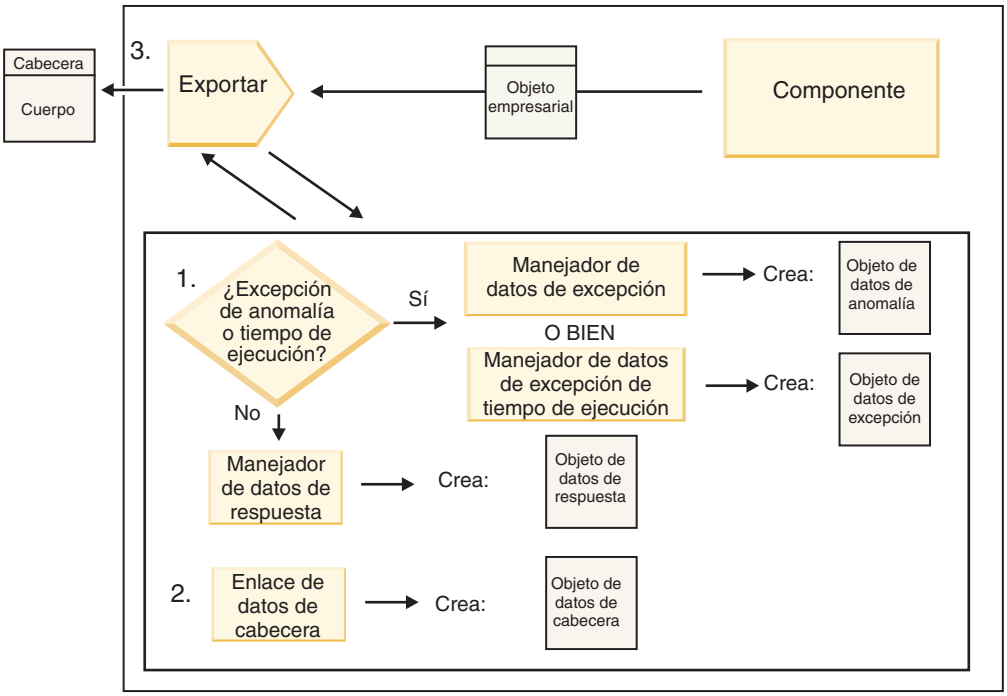


Figura 5. Flujo de retorno de una respuesta a través de la exportación

Se produce la secuencia de pasos siguiente:

1. Si el enlace de exportación recibe un mensaje de respuesta normal, el manejador de datos de respuesta o el enlace de datos en el método transforma el objeto empresarial en una respuesta.

   Si la respuesta es un error, el manejador de datos de error o el enlace de datos en el método transforma el error en una respuesta de error.

   Solamente en el caso de los enlaces de exportación HTTP, si la respuesta es una excepción de tiempo de ejecución, se invoca el manejador de datos de excepción de tiempo de ejecución, si está configurado.
2. Sólo para los enlaces de WebSphere MQ, el enlace de datos de cabecera transforma los objetos de datos de cabecera en cabeceras de protocolo.
3. La exportación envía la respuesta de servicio a través del transporte.

Flujo de información a través de una importación

Los componentes envían solicitudes a los servicios situados fuera del módulo, mediante una importación. La petición se envía, a través de un trasporte específico determinado por el enlace asociado.

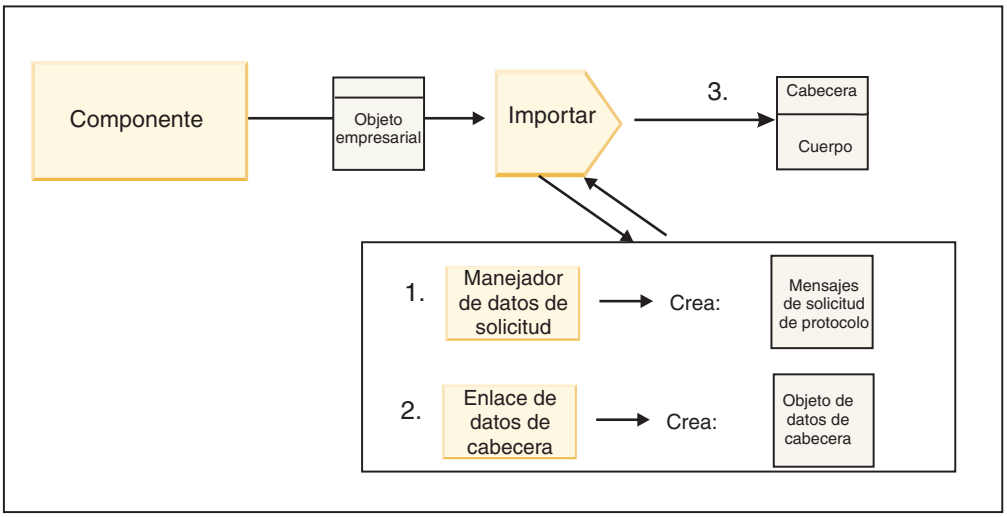


Figura 6. Flujo desde un componente a través de la importación a un servicio

El componente invoca la importación con un objeto empresarial de solicitud.

Nota:

- El componente que realiza la llamada debe invocar el enlace de importación HTTP, el enlace de importación de servicio web y el enlace de importación EJB de forma síncrona.
- Los enlaces de importación JMS, JMS genéricos, MQ JMS, y de WebSphere MQ deben invocarse de forma asíncrona.

Después de que el componente invoca la importación, se produce la siguiente secuencia de sucesos:

1. El manejador de datos de petición o el enlace de datos en el método transforma el objeto empresarial de petición en un mensaje de petición de protocolo.
2. Sólo para los enlaces de WebSphere MQ, el enlace de datos de cabecera en el método establece el objeto empresarial de cabecera en la cabecera de protocolo.
3. La importación invoca el servicio con la solicitud de servicio a través del transporte.

Si es una operación bidireccional, el servicio devuelve una respuesta y se produce la siguiente secuencia de pasos:

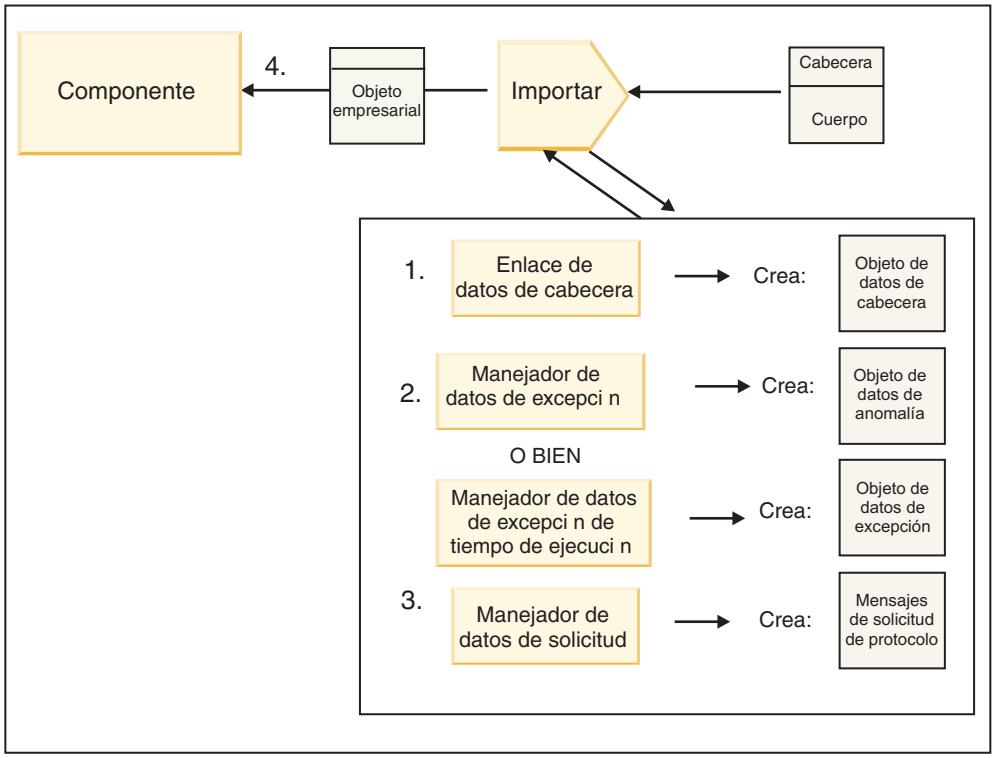


Figura 7. Flujo de retorno de una respuesta a través de la importación

1. Sólo para los enlaces de WebSphere MQ, el enlace de datos de cabecera transforma la cabecera de protocolo en un objeto de datos de cabecera.
2. Se determina si la respuesta es un error.
   - Si la respuesta es un error, el selector de error inspecciona el error para determinar con qué error de WSDL se correlaciona. A continuación, el manejador de datos de error en el método transforma el error en una respuesta de error.
   - Si la respuesta es una excepción de tiempo de ejecución, se invoca el manejador de datos de excepción de tiempo de ejecución, si está configurado.
3. El manejador de datos de respuesta o el enlace en el método transforma la respuesta en un objeto empresarial de respuesta.
4. La importación devuelve el objeto empresarial de respuesta al componente.

Configuración de los enlaces de exportación e importación

Uno de los aspectos clave de los enlaces de exportación e importación es la transformación de datos, que indica cómo se correlacionan (deserializan) los datos desde un formato conectado nativo a un objeto empresarial, o cómo se correlacionan (serializan) desde un objeto empresarial a un formato conectado nativo. Para los enlaces asociados con las exportaciones, también puede especificar un selector de función para indicar qué operación debe ejecutarse en los datos. Para los enlaces asociados con las exportaciones o importaciones, puede indicar cómo deben manejarse los errores que se producen durante el proceso.

Asimismo, especifique información específica del transporte en los enlaces. Por ejemplo, para un enlace HTTP, especifique el URL de punto final. Puede encontrar más información en el Centro de información de WebSphere Integration Developer. Por ejemplo, para el enlace HTTP, la información específica del transporte se

describe en los temas “Generación de un enlace de importación HTTP” y “Generación de un enlace de exportación HTTP”.

Transformación del formato de datos en importaciones y exportaciones

Cuando se configura un enlace de exportación o importación en WebSphere Integration Developer, una de las propiedades de configuración que especifica es el formato de los datos que utiliza el enlace.

- Para los enlaces de exportación, en los que una aplicación de cliente envía peticiones y recibe respuestas de un componente SCA, debe indicar el formato de los datos nativos. Dependiendo del formato, el sistema selecciona el manejador de datos o el enlace de datos correspondiente para transformar los datos nativos en un objeto empresarial (que utiliza el componente SCA) y, a la inversa, para transformar un objeto empresarial en datos nativos (que es la respuesta a la aplicación de cliente).
- Para los enlaces de importación, en los que un componente SCA envía peticiones y recibe respuestas de un servicio fuera del módulo, debe indicar el formato de los datos nativos. Dependiendo del formato, el sistema selecciona el manejador de datos o el enlace de datos correspondiente para transformar el objeto empresarial en datos nativos y viceversa.

WebSphere Process Server proporciona un conjunto de formatos de datos predefinidos y los manejadores de datos o enlaces de datos correspondientes que dan soporte a los formatos. También puede crear sus propios manejadores de datos personalizados y registrar el formato de datos para los manejadores de datos. Para obtener más información, consulte el tema “Desarrollo de manejadores de datos” en el Centro de información de WebSphere Integration Developer.

- Los *manejadores de datos* tienen un protocolo neutro y transforman datos de un formato a otro. En WebSphere Process Server, los manejadores de datos normalmente transforman datos nativos (por ejemplo, XML, CSV y COBOL) en un objeto empresarial, y un objeto empresarial en datos nativos. Como tiene un protocolo neutro, puede reutilizar el mismo manejador de datos con varios enlaces de exportación e importación. Por ejemplo, puede utilizar el mismo manejador de datos XML con un enlace de exportación o importación HTTP o con un enlace de exportación o importación JMS.
- Los *enlaces de datos* también transforman datos nativos en un objeto empresarial (y viceversa), pero son específicos del protocolo. Por ejemplo, un enlace de datos HTTP sólo puede utilizarse con un enlace de exportación o importación HTTP. A diferencia de los manejadores de datos, un enlace de datos HTTP no puede reutilizarse con un enlace de exportación o importación MQ.

  Nota: Tres enlaces de datos HTTP (HTTPStreamDataBindingSOAP, HTTPStreamDataBindingXML y HTTPServiceGatewayDataBinding) están en desuso como parte de WebSphere Process Server Versión 7.0. Utilice los manejadores de datos siempre que sea posible.

Como se ha indicado anteriormente, puede crear manejadores de datos personalizados, si es necesario. También puede crear enlaces de datos personalizados; no obstante, resulta recomendable crear manejadores de datos personalizados porque se pueden utilizar entre varios enlaces.

Manejadores de datos

Los manejadores de datos se configuran respecto a los enlaces de exportación e importación para transformar datos de un formato a otro mediante un protocolo

neutro. Se proporcionan varios manejadores de datos como parte del producto, pero también puede crear manejadores de datos propios, si es necesario. Puede asociar un manejador de datos con un enlace de exportación o importación en uno de estos dos niveles: puede asociarlo con todas las operaciones de la interfaz de la exportación o importación, o puede asociarlo con una operación específica para la petición o respuesta.

Manejadores de datos predefinidos

WebSphere Integration Developer se utiliza para especificar el manejador de datos que desee utilizar.

Los manejadores de datos predefinidos para que los utilice el usuario se enumeran en la tabla siguiente, que también describe cómo el manejador de datos transforma datos de entrada y salida.

Nota: Excepto allí donde se indique específicamente, dichos manejadores de datos se pueden utilizar con enlaces JMS, JMS genéricos, MQ JMS, WebSphere MQ y HTTP.
Consulte el tema "Manejadores de datos", en el Centro de información de WebSphere Integration Developer, para obtener información más detallada.

Tabla 1. Manejadores de datos predefinidos

Manejador de datos	Datos nativos a objeto empresarial	Objeto empresarial a datos nativos
ATOM	Analiza canales de información ATOM en un objeto empresarial de canal de información ATOM.	Serializa un objeto de negocio de canal de información ATOM en canales de información ATOM.
Delimitado	Analiza los datos delimitados en un objeto empresarial.	Serializa un objeto de negocio en datos delimitados, incluido el formato CSV.
Anchura fija	Analiza datos de anchura fija en un objeto empresarial.	Serializa un objeto empresarial en datos de anchura fija.
Gestionado por WTX	Delega la transformación del formato de datos a WebSphere Transformation Extender (WTX). El nombre de correlación WTX se deriva del manejador de datos.	Delega la transformación del formato de datos a WebSphere Transformation Extender (WTX). El nombre de correlación WTX se deriva del manejador de datos.
Gestionado por WTX Invoker	Delega la transformación del formato de datos al WebSphere Transformation Extender (WTX). El nombre de correlación WTX lo proporciona el usuario.	Delega la transformación del formato de datos al WebSphere Transformation Extender (WTX). El nombre de correlación WTX lo proporciona el usuario.
JAXB	Serializa los beans Java en un objeto de negocio utilizando las reglas de correlación definidas por la especificación JAXB (Java Architecture for XML Binding).	Deserializa un objeto de negocio en beans Java utilizando las reglas de correlación definidas por la especificación JAXB.

Tabla 1. Manejadores de datos predefinidos (continuación)

Manejador de datos	Datos nativos a objeto empresarial	Objeto empresarial a datos nativos
JAXWS **Nota:** El manejador de datos JAXWS sólo puede utilizase con el enlace EJB.	Utilizado por un enlace EJB para transformar un objeto Java de respuesta o un objeto Java de excepción en un objeto de negocio de respuesta utilizando las reglas de correlación definidas por la especificación de API de Java para servicios web XML (JAX-WS).	Utilizado por un enlace EJB para transformar un objeto de negocio a los parámetros del método Java de salida utilizando las reglas de correlación definidos por la especificación JAX-WS.
JSON	Analiza datos JSON en un objeto empresarial.	Serializa un objeto empresarial en datos JSON.
Cuerpo nativo	Analiza los bytes nativos, el texto, la correlación, la corriente o un objeto en uno de los cinco objetos de negocio básicos (texto, bytes, correlación, corriente u objeto).	Transforma los objetos de negocio básicos en byte, texto, correlación, corriente u objeto.
SOAP	Analiza el mensaje SOAP (y la cabecera) en un objeto empresarial.	Serializa un objeto empresarial en datos a un mensaje SOAP.
XML	Analiza datos XML en un objeto empresarial.	Serializa un objeto de negocio en datos XML.
UTF8XMLDataHandler	Analiza datos XML codificados UTF-8 en un objeto empresarial.	Serializa un objeto empresarial en datos XML codificados UTF-8 cuando se envía un mensaje.

Creación de un manejador de datos

Puede encontrar información detallada sobre la creación de un manejador de datos en el tema “Desarrollo de manejadores de datos” en el Centro de información de WebSphere Integration Developer.

Enlaces de datos

Los enlaces de datos se configuran respecto a los enlaces de exportación e importación para transformar datos de un formato a otro. Los enlaces de datos son específicos de un protocolo. Se proporcionan varios enlaces de datos como parte del producto, pero también puede crear enlaces de datos propios, si es necesario. Puede asociar un enlace de datos con un enlace de exportación o importación en uno de dos niveles; puede asociarlo con todas las operaciones de la interfaz de la exportación o importación, o puede asociarlo con una operación específica para la petición o respuesta.

WebSphere Integration Developer se utiliza para especificar el enlace de datos que desea utilizar o para crear un enlace de datos propio. Puede encontrar una descripción sobre la creación de enlaces de datos en la sección “Visión general de enlaces JMS, JMS de MQ y de JMS genérico” del centro de información de WebSphere Integration Developer.

Enlaces JMS

En la tabla siguiente se enumeran los enlaces de datos que se pueden utilizar con:

- Enlaces JMS
- Enlaces JMS genéricos
- Enlaces JMS de WebSphere MQ

La tabla también incluye una descripción de las tareas que los enlaces de datos realizan.

Tabla 2. Enlaces de datos predefinidos para enlaces JMS

Enlace de datos	Datos nativos a objeto empresarial	Objeto empresarial a datos nativos
Objeto Java serializado	Transforma el objeto Java serializado en un objeto empresarial (que se correlaciona como el tipo de entrada o salida en WSDL).	Serializa un objeto empresarial en el objeto serializado Java en el mensaje de objeto JMS.
Bytes incluidos	Extrae los bytes del mensaje de bytes de entrada JMS y los envuelve en el objeto empresarial JMSBytesBody.	Extrae los bytes del objeto empresarial JMSBytesBody y los envuelve en el mensaje de bytes JMS de salida.
Entrada de correlación envuelta	Extrae el nombre, el valor y la información de tipo de cada entrada del mensaje de correlación JMS de entrada y crea una lista de objetos de negocio MapEntry. A continuación, envuelve la lista en el objeto empresarial JMSMapBody.	Extrae el nombre, el valor y la información de tipo de la lista MapEntry en el objeto empresarial JMSMapBody y crea las entradas correspondientes en el mensaje de correlación JMS de salida.
Objeto envuelto	Extrae el objeto del mensaje de objeto JMS de entrada y lo envuelve en el objeto empresarial JMSObjectBody.	Extrae el objeto del objeto empresarial JMSObjectBody y lo envuelve en el mensaje de objeto JMS de salida.
Texto incluido	Extrae el texto del mensaje de texto JMS de entrada y lo envuelve en el objeto empresarial JMSTextBody.	Extrae el texto del objeto empresarial JMSTextBody y lo envuelve en el mensaje de texto JMS de salida.

Enlaces de WebSphere MQ

En la tabla siguiente se enumeran los enlaces de datos que se pueden utilizar con WebSphere MQ y se describen las tareas que los enlaces de datos realizan.

Tabla 3. Enlaces de datos predefinidos para enlaces de WebSphere MQ

Enlace de datos	Datos nativos a objeto empresarial	Objeto empresarial a datos nativos
Objeto Java serializado	Transforma el objeto Java serializado procedente del mensaje entrante en un objeto empresarial (que se correlaciona como el tipo de entrada o salida en WSDL).	Transforma un objeto empresarial en el objeto serializado Java en el mensaje de salida.

Tabla 3. Enlaces de datos predefinidos para enlaces de WebSphere MQ (continuación)

Enlace de datos	Datos nativos a objeto empresarial	Objeto empresarial a datos nativos
Bytes incluidos	Extrae los bytes del mensaje de bytes MQ sin estructurar y los envuelve en el objeto empresarial JMSBytesBody.	Extrae los bytes de un objeto empresarial JMSBytesBody y recorta los bytes en el mensaje de bytes MQ sin estructurar de salida.
Texto incluido	Extrae el texto del mensaje de texto MQ sin estructurar y lo envuelve en el objeto empresarial JMSTextBody.	Extrae el texto del objeto empresarial JMSTextBody y lo envuelve en un mensaje de texto MQ sin estructurar.
Entrada de corriente envuelta	Extrae el nombre y la información de tipo de cada entrada del mensaje de corriente JMS de entrada y crea una lista de objetos de negocio StreamEntry. A continuación, envuelve la lista en el objeto empresarial JMSStreamBody.	Extrae el nombre y la información de tipo de la lista StreamEntry en el objeto empresarial JMSStreamBody y crea las entradas correspondientes en el mensaje JMSStreamMessage de salida.

Además de los enlaces de datos listados en Tabla 3 en la página 19, WebSphere MQ también utiliza los enlaces de datos de cabecera. Consulte el centro de información de WebSphere Integration Developer para ver detalles.

Enlaces HTTP

En la tabla siguiente se enumeran los enlaces de datos que se pueden utilizar con HTTP y se describen las tareas que los enlaces de datos realizan.

Tabla 4. Enlaces de datos predefinidos para enlaces HTTP

Enlace de datos	Datos nativos a objeto empresarial	Objeto empresarial a datos nativos
Bytes incluidos	Extrae los bytes del cuerpo del mensaje HTTP de entrada y los envuelve en el objeto empresarial HTTPBytes.	Extrae los bytes del objeto empresarial HTTPBytes y los añade al cuerpo del mensaje HTTP de salida.
Texto incluido	Extrae el texto del cuerpo del mensaje HTTP de entrada y lo envuelve en el objeto de negocio HTTPText.	Extrae el texto del objeto de negocio HTTPText y lo añade al cuerpo del mensaje HTTP de salida.

Selectores de función en enlaces de exportación

Un selector de función para indicar qué operación debe ejecutarse en los datos para un mensaje de petición. Los selectores de función se configuran como parta de un enlace de exportación.

Consideremos una exportación SCA que expone una interfaz. La interfaz contiene dos operaciones: Crear y Actualizar. La exportación tiene un enlace JMS que lee de una cola.

Cuando llega un mensaje a la cola, se pasa la exportación a los datos asociados, pero, ¿qué operación de la interfaz de la exportación debe invocarse en el componente conectado? La operación viene determinada por la configuración del selector de función y del enlace de exportación.

El selector de función devuelve el nombre de función nativa (el nombre de función en el sistema cliente que ha enviado el mensaje). A continuación, el nombre de función nativa se correlaciona con el nombre de función u operación en la interfaz asociada a la exportación. Por ejemplo, en la siguiente figura, el selector de función devuelve el nombre de función nativa (CRT) del mensaje entrante, el nombre de función nativa se correlaciona con la operación Crear, y el objeto empresarial se envía al componente SCA con la operación Crear.

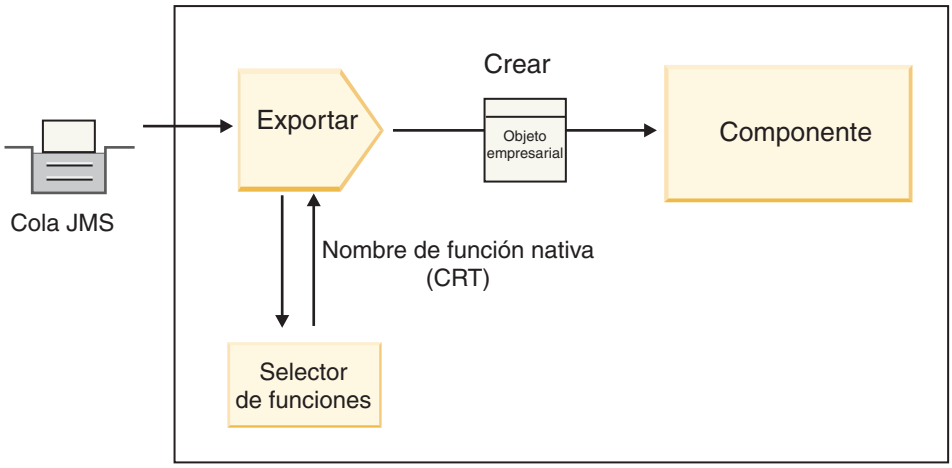


Figura 8. El selector de función

Si la interfaz sólo tiene una operación, no es necesario especificar ningún selector de función.

Hay disponibles varios selectores de función preempaquetados, que se enumeran en las secciones siguientes.

Enlaces JMS

En la tabla siguiente se enumeran los selectores de función que se pueden utilizar con:

- Enlaces JMS
- Enlaces JMS genéricos
- Enlaces JMS de WebSphere MQ

Tabla 5. Selectores de función predefinidos para enlaces JMS

Selector de función	Descripción
Selector de función JMS para enlaces de datos JMS simples	Utiliza la propiedad JMSType del mensaje para seleccionar el nombre de la operación.
Selector de función de la propiedad de cabecera JMS	Devuelve el valor de String Property de JMS, TargetFunctionName, de la cabecera.
Selector de función de pasarela de servicio JMS	Determina si la petición es una operación unidireccional o bidireccional, examinando la propiedad JMSReplyTo, establecida por el cliente.

Enlaces de WebSphere MQ

En la tabla siguiente se enumeran los selectores de función que se pueden utilizar con enlaces WebSphere MQ:

Tabla 6. Selectores de función predefinidos para enlaces de WebSphere MQ

Selector de función	Descripción
Selección de función handleMessage de MQ	Devuelve handleMessage como un valor que se correlaciona, utilizando los enlaces de método de exportación, al nombre de una operación de la interfaz.
MQ utiliza el selector de función por omisión de JMS.	Lee la operación nativa de la propiedad TargetFunctionName de la carpeta de una cabecera MQRFH2.
MQ utiliza el formato del cuerpo del mensaje como función nativa	Encuentra el campo Format de la última cabecera y devuelve dicho campo como un valor String.
Selección de función tipo de MQ	Crea un método en el enlace de exportación recuperando un URL que contiene las propiedades Msd, Set, Type y Format, encontradas en la cabecera MQRFH2.
Selector de función de pasarela de servicio MQ	Utiliza la propiedad MsgType de la cabecera MQMD para determinar el nombre de la operación.

Enlaces HTTP

En la tabla siguiente se enumeran los selectores de función que se pueden utilizar con enlaces HTTP.

Tabla 7. Selectores de función predefinidos para enlaces HTTP

Selector de función	Descripción
Selector de función HTTP basado en la cabecera TargetFunctionName	Utiliza la propiedad de la cabecera HTTP TargetFunctionName del cliente para determinar qué operación debe invocarse en tiempo de ejecución desde la exportación.
Selector de función HTTP basado en el URL y el método HTTP.	Utiliza la vía de acceso relativa del URL añadido con el método HTTP del cliente para determinar la operación nativa definida en la exportación.
Selector de función de pasarela de servicio HTTP basado en URL con un nombre de operación.	Determina el método para invocar basado en el URL si se ha añadido "operationMode = oneWay" al URL de petición.

Nota: También puede crear su propio selector de función utilizando WebSphere Integration Developer. En el Centro de información de WebSphere Integration Developer se proporciona información sobre cómo crear un selector de función. Por ejemplo, en el tema “Visión general de selectores de función MQ”, se puede encontrar una descripción de un selector de función para los enlaces de WebSphere MQ.

Manejo de errores

Puede configurar los enlaces de importación y exportación para que manejen los errores (por ejemplo, las excepciones empresariales) que se producen durante el proceso especificando manejadores de datos de error. Puede configurar un manejador de datos de error a tres niveles: puede asociar un manejador de datos de error con una anomalía, con una operación, o para todas las operaciones con un enlace.

Un manejador de datos de error procesa los datos de error y los transforma al formato correcto para que se puedan enviar en el enlace de exportación o importación.

- Para un enlace de exportación, el manejador de datos de error transforma el objeto empresarial de excepción enviado desde el componente en un mensaje de respuesta que pueda utilizarse en la aplicación cliente.
- Para un enlace de importación, el manejador de datos de error transforma los datos de error o el mensaje de respuesta enviado desde un servicio en un objeto empresarial de excepción que pueda utilizarse en el componente SCA.

Para los enlaces de importación, el enlace invoca el selector de error, que determina si el mensaje de respuesta es una respuesta normal, un error de empresa o una excepción de tiempo de ejecución.

Puede especificar un manejador de datos de error para un determinado error, para una operación y para todas las operaciones con un enlace.

- Si el manejador de datos de error se establece a los tres niveles, se invoca el manejador de datos asociado con un determinado error.
- Si los manejadores de datos de error se establecen en los niveles de operación y enlace, se invoca el manejador de datos asociado con la operación.

En WebSphere Integration Developer, se utilizan dos editores para especificar el manejo de errores. El editor de interfaz se utiliza para indicar si habrá un error en una operación. Una vez generado un enlace con esta interfaz, el editor en la vista de propiedades permite configurar cómo se manejará el error. Para obtener más información, consulte el tema “Selectores de error” en el Centro de información de WebSphere Integration Developer.

Cómo se manejan los errores en los enlaces de exportación

Cuando se produce un error durante el proceso de la petición de una aplicación de cliente, el enlace de exportación puede devolver la información de error al cliente. Configure el enlace de exportación para especificar cómo se debe procesar y devolver el error al cliente.

El enlace de exportación se configura utilizando WebSphere Integration Developer.

Durante el proceso de la solicitud, un cliente invoca una exportación con una solicitud, y la exportación invoca el componente SCA. Durante el proceso de la solicitud, el componente SCA puede devolver una respuesta empresarial, o puede generar una excepción empresarial de servicio o una excepción de tiempo de ejecución de servicio. Cuando esto ocurre, el enlace de exportación transforma la excepción en un mensaje de error y lo envía al cliente, tal como se muestra en la siguiente figura y se describe en las siguientes secciones.

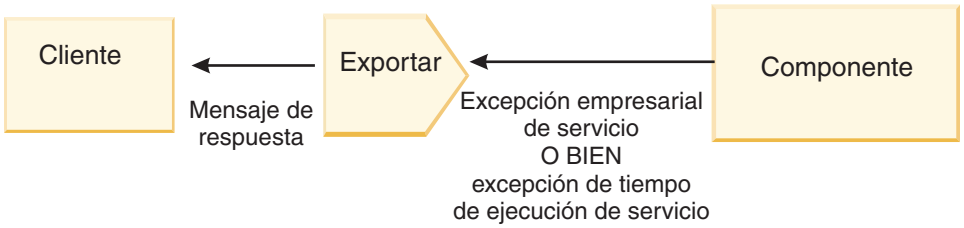


Figura 9. Cómo se envía la información de error desde el componente mediante el enlace de exportación al cliente

Puede crear un manejador de datos personalizado o un enlace de datos para manejar los errores.

Errores de empresa

Los errores de empresa son anomalías o excepciones de empresa que se producen durante el proceso.

Considere la siguiente interfaz, que tiene una operación createCustomer. Esta operación tiene definidos dos errores de empresa: CustomerAlreadyExists y MissingCustomerId.

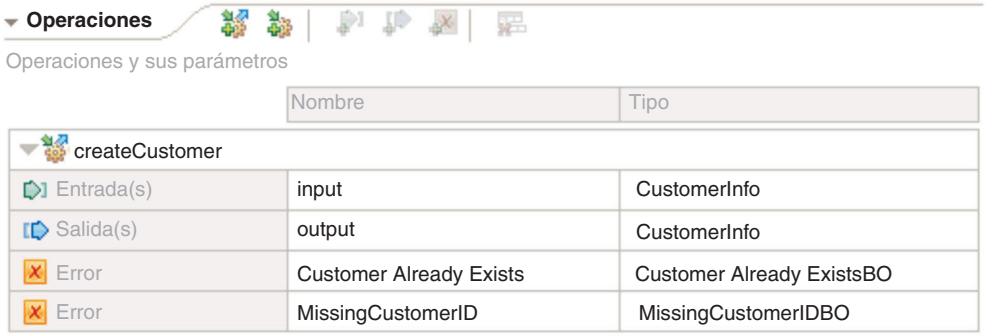


Figura 10. Interfaz con dos errores

En este ejemplo, si un sistema cliente envía una petición para crear un cliente (de este componente SCA) y el cliente ya existe, el componente genera un error CustomerAlreadyExists en la exportación. La exportación debe propagar este error de empresa de nuevo al cliente que realiza la llamada. Para ello, utiliza el manejador de datos de error que se ha configurado en el enlace de exportación.

Cuando el enlace de exportación recibe un error de empresa, se produce el siguiente proceso:

1. El enlace determina qué manejador de datos de error invocar para manejar el error. Si la excepción empresarial de servicio contiene el nombre del error, se invoca el manejador de datos configurado en el error. Si la excepción empresarial de servicio no contiene el nombre del error, el nombre del error se deriva haciendo coincidir los tipos de error.
2. El enlace invoca el manejador de datos de error con el objeto de datos de la excepción empresarial de servicio.
3. El manejador de datos de error transforma el objeto de datos de error en un mensaje de respuesta y lo devuelve al enlace de exportación.
4. La exportación devuelve el mensaje de respuesta al cliente.

Si la excepción empresarial de servicio contiene el nombre del error, se invoca el manejador de datos configurado en el error. Si la excepción empresarial de servicio no contiene el nombre del error, el nombre del error se deriva haciendo coincidir los tipos de error.

Excepciones de tiempo de ejecución

Una excepción de tiempo de ejecución es una excepción que se produce en la aplicación SCA durante el proceso de una petición que no se corresponde con un error de empresa. A diferencia de los errores de empresa, las excepciones de tiempo de ejecución no se definen en la interfaz.

En determinados escenarios, si lo desea, puede propagar estas excepciones de tiempo de ejecución a la aplicación de cliente para que esta pueda realizar la acción correspondiente.

Por ejemplo, si un sistema cliente envía una petición (al componente SCA) para crear un cliente y se produce un error de autorización durante el proceso de la solicitud, el componente genera una excepción de tiempo de ejecución. Esta excepción de tiempo de ejecución debe volver a propagarse al cliente que realiza la llamada, para que pueda realizar la acción correspondiente para la autorización. Para ello, se utiliza el manejador de datos de excepción de tiempo de ejecución configurado en el enlace de exportación.

Nota: Puede configurar un manejador de datos de excepción de tiempo de ejecución sólo en enlaces HTTP.

El proceso de una excepción de tiempo de ejecución es parecido al proceso de un error de empresa. Si se ha configurado un manejador de datos de excepción de tiempo de ejecución, se produce el siguiente proceso:

1. El enlace de exportación invoca el manejador de datos adecuado con la excepción de tiempo de ejecución de servicio.
2. El manejador de datos transforma el objeto de datos de error en un mensaje de respuesta y lo devuelve al enlace de exportación.
3. La exportación devuelve el mensaje de respuesta al cliente.

El manejo de errores y el manejo de excepciones de tiempo de ejecución son opcionales. Si no desea propagar los errores o las excepciones de tiempo de ejecución al cliente que realiza la llamada, no configure el manejador de datos de error o el manejador de datos de excepción de tiempo de ejecución.

Cómo se manejan los errores en los enlaces de importación

Un componente utiliza una importación para enviar una petición a un servicio fuera del módulo. Cuando se produce un error durante el proceso de la solicitud, el servicio devuelve el error al enlace de importación. Puede configurar el enlace de importación para especificar cómo se debe procesar y devolver el error al componente.

El enlace de importación se configura utilizando WebSphere Integration Developer. Puede especificar un manejador de datos de error (o un enlace de datos), así como un selector de error.

Manejadores de datos de error

El servicio que procesa la petición envía al enlace de importación la información de error, en forma de excepción o un mensaje de respuesta que contiene los datos de error.

El enlace de importación transforma la excepción de servicio o el mensaje de respuesta en una excepción empresarial de servicio o una excepción de tiempo de ejecución de servicio, tal como se muestra en la siguiente figura y se describe en las siguientes secciones.

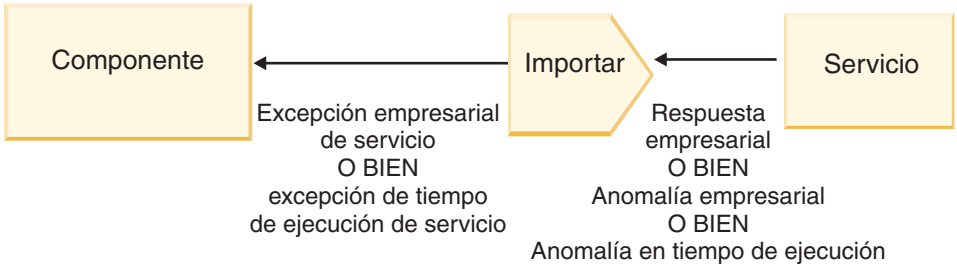


Figura 11. Cómo se envía la información de error desde el servicio mediante la importación al componente

Puede crear un manejador de datos personalizado o un enlace de datos para manejar los errores.

Selectores de error

Cuando configura un enlace de importación, puede especificar un selector de error. El selector de error determina si la respuesta de importación es una respuesta real, una excepción empresarial o un error de tiempo de ejecución. También determina, a partir del cuerpo o la cabecera de la respuesta, el nombre de error nativo, que la configuración del enlace correlaciona al nombre de un error de la interfaz asociada.

Hay disponibles dos tipos de selectores de error preempaquetados que se pueden utilizar con las importaciones JMS, MQ JMS, JMS genéricas, WebSphere MQ y HTTP:

Tabla 8. Selectores de error preempaquetados

Tipo de selector de error	Descripción
Basado en cabecera	Determina si un mensaje de respuesta es un error de empresa, una excepción de tiempo de ejecución, o un mensaje normal basado en las cabeceras del mensaje de respuesta de entrada.
SOAP	Determina si el mensaje SOAP de respuesta es una respuesta normal, un error de empresa o una excepción de tiempo de ejecución.

A continuación se muestran ejemplos de selectores de error basados en cabecera y el selector de error SOAP.

- Selector de error basado en cabecera

Si una aplicación quiere indicar que el mensaje de entrada es un error de empresa, deben haber dos cabeceras en el mensaje de entrada para los errores de empresa, tal como se muestra a continuación:

```
Header name = FaultType, Header value = Business
Header name = FaultName, Header value = <nombre de error nativo definido por el usuario>
```

Si una aplicación quiere indicar que el mensaje de entrada es una excepción de tiempo de ejecución, entonces debe haber una cabecera en el mensaje de entrada, tal como se muestra a continuación:

```
Header name = FaultType, Header value = Runtime
```

- Selector de error SOAP

  Una error de empresa debe enviarse como parte del mensaje SOAP con la cabecera SOAP personalizada siguiente. "CustomerAlreadyExists" es el nombre del error, en este caso.

  ```
  <ibmSoap:BusinessFaultName
  xmlns:ibmSoap="http://www.ibm.com/soap">CustomerAlreadyExists
  <ibmSoap:BusinessFaultName>
  ```

El selector de error es opcional. Si no especifica ningún selector de error, el enlace de importación no puede determinar el tipo de respuesta. Por tanto, el enlace la trata como si fuera una respuesta de empresa y llama al manejador de datos de respuesta o al enlace de datos de respuesta.

Puede crear un selector de error personalizado. Los pasos necesarios para crear un selector de error personalizo se proporcionan en el tema “Desarrollo de un selector de error personalizado” del Centro de información de WebSphere Integration Developer.

Errores de empresa

Un error de empresa puede producirse cuando hay un error en el proceso de una solicitud. Por ejemplo, si envía una petición para crear un cliente y el cliente ya existe, el servicio envía una excepción empresarial al enlace de importación.

Cuando el enlace recibe una excepción empresarial, los pasos del proceso dependen de si se ha configurado un selector de error para el enlace.

- Si no se ha configurado un selector de error, el enlace invoca el manejador de datos de respuesta o el enlace de datos de respuesta.
- Si se ha configurado un selector de error, se produce el siguiente proceso:
  1. El enlace de importación invoca el selector de error para determinar si la respuesta es un error de empresa, una respuesta de empresa, o un error de tiempo de ejecución.
  2. Si la respuesta es un error de empresa, el enlace de importación invoca el selector de error para proporcionar el nombre de error nativo.
  3. El enlace de importación determina el error WSDL que corresponda al nombre de error nativo que haya devuelto el selector de error.
  4. El enlace de importación determina el manejador de datos de error que se haya configurado para este error WSDL.
  5. El enlace de importación invoca este manejador de datos con los datos de error.
  6. El manejador de datos de error transforma los datos de error en un objeto de datos y lo devuelve al enlace de importación.
  7. El enlace de importación construye un objeto de excepción empresarial de servicio con el objeto de datos y el nombre de error.

8. La importación devuelve el objeto de excepción empresarial de servicio al componente.

Excepciones de tiempo de ejecución

Una excepción de tiempo de ejecución puede producirse cuando hay un problema de comunicación con el servicio. El proceso de una excepción de tiempo de ejecución es parecido al proceso de una excepción empresarial. Si se ha configurado un selector de error, se produce el siguiente proceso:

1. El enlace de importación invoca el manejador de datos de excepción de tiempo de ejecución con los datos de excepción.
2. El manejador de datos de excepción de tiempo de ejecución transforma los datos de excepción en un objeto de excepción de tiempo de ejecución de servicio y los devuelve al enlace de importación.
3. La importación devuelve el objeto de excepción tiempo de ejecución de servicio al componente.

Interoperatividad entre módulos SCA y servicios Open SCA

El paquete de características de IBM WebSphere Application Server V7.0 para Service Component Architecture (SCA) proporciona un modelo de programación sencillo, pero potente, para construir aplicaciones basadas en las especificaciones de Open SCA. Los módulos SCA de WebSphere Process Server utilizan los enlaces de importación y exportación para interoperar con servicios de Open SCA desarrollados en un entorno de Rational Application Developer y alojado por el paquete de características de WebSphere Application Server para Service Component Architecture.

Una aplicación SCA invoca una aplicación Open SCA a través de un enlace de importación. Una aplicación SCA recibe una llamada de una aplicación Open SCA a través de un enlace de exportación. Se muestra una lista de enlaces soportados en “Invocación de servicios en enlaces interoperables ” en la página 30.

Invocación de servicios Open SCA desde módulos SCA

Las aplicaciones SCA desarrolladas con WebSphere Integration Developer pueden invocar las aplicaciones Open SCA desarrolladas en un entorno de Rational Application Developer. En este apartado se proporciona un ejemplo de invocación de un servicio Open SCA desde un módulo SCA utilizando un enlace de importación SCA.

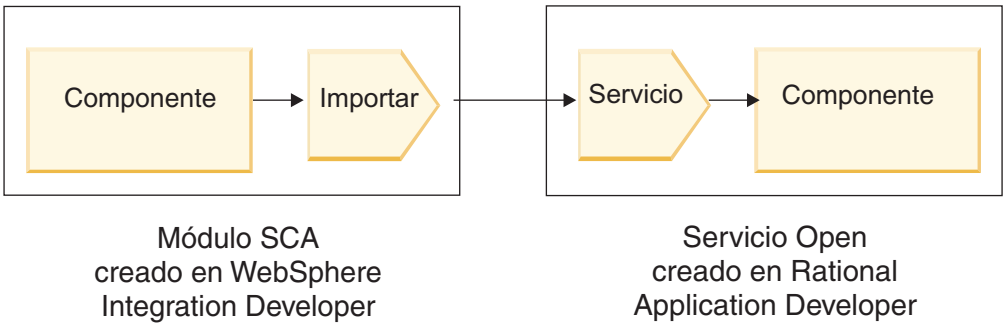


Figura 12. Componente de un módulo SCA que invoca un servicio Open SCA

No es necesaria ninguna configuración especial para invocar un servicio Open SCA.

Para conectarse a un servicio Open SCA a través de un enlace de importación SCA, proporcione el nombre de componente y el nombre del servicio Open SCA en el enlace de importación.

1. Para obtener el nombre del componente de destino y del servicio del compuesto Open SCA, realice los pasos siguientes:
   a. Asegúrese de que la pestaña **Propiedades** esté abierta pulsando **Ventana → Mostrar vista → Propiedades**.
   b. Abra el editor compuesto efectuando una doble pulsación en el diagrama compuesto que contiene el componente y el servicio. Por ejemplo, para un componente denominado **customer**, el diagrama compuesto es **customer.composite_diagram**.
   c. Pulse el componente de destino.
   d. En el campo **Nombre** de la pestaña **Propiedades**, escriba el nombre del componente de destino.
   e. Pulse el icono de servicio asociado con el componente.
   f. En el campo **Nombre** de la pestaña **Propiedades**, escriba el nombre del servicio.
2. Para configurar la importación de WebSphere Process Server para conectarlo al servicio Open SCA, realice los pasos siguientes:
   a. En WebSphere Integration Developer, vaya a la pestaña **Propiedades** de la importación SCA que desea conectar al servicio Open SCA.
   b. En el campo **Nombre de módulo**, entre el nombre de componente del paso 1d.
   c. En el campo **Nombre de la exportación**, entre el nombre de servicio del paso 1f.
   d. Guarde el trabajo pulsando Control+S.

Invocación de módulos SCA desde servicios Open SCA

Las aplicaciones Open SCA desarrolladas en un entorno de Rational Application Developer pueden invocar aplicaciones SCA desarrolladas con WebSphere Integration Developer. En este apartado se proporciona un ejemplo de invocación de un módulo SCA (a través de un enlace de exportación de SCA) de un servicio Open SCA.

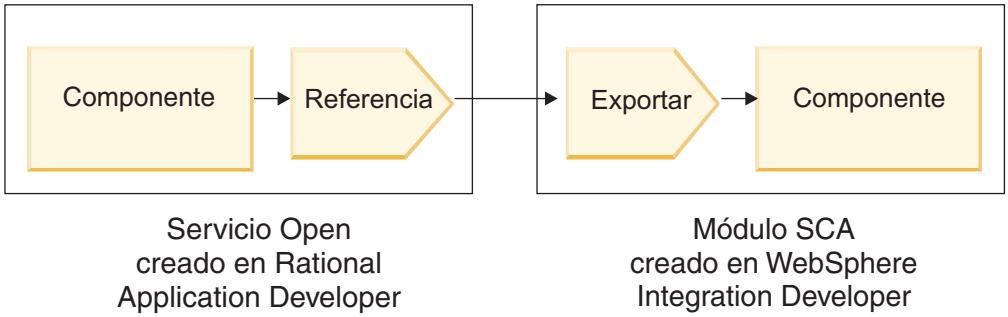


Figura 13. Servicio Open SCA que invoca el componente de un módulo SCA

Para conectarse a un componente SCA por medio de un enlace de referencia Open SCA, proporcione el nombre de módulo y el nombre de exportación.

1. Para obtener el nombre del módulo de destino y de la exportación, realice los pasos siguientes:
   a. En WebSphere Integration Developer, abra el módulo en el editor de ensamblaje efectuando una doble pulsación en el módulo.
   b. Pulse la exportación.
   c. En el campo **Nombre** de la pestaña **Propiedades**, escriba el nombre de la exportación.
2. Configure la referencia de Open SCA que desea conectar al módulo de WebSphere Process Server y la exportación:
   a. En Rational Application Developer, abra el editor compuesto efectuando una doble pulsación en el diagrama compuesto que contiene el componente y el servicio.
   b. Pulse el icono de referencia de la referencia de componente para visualizar las propiedades de referencia en la pestaña **Propiedades**.
   c. Pulse la pestaña **Enlace** situada en el lado izquierdo de la página.
   d. Pulse **Enlaces** y, a continuación, pulse **Añadir**.
   e. Seleccione el enlace **SCA**.
   f. En el campo **Uri**, entre el nombre de módulo de WebSphere Process Server, seguido de una barra inclinada ("/"), seguida del nombre de exportación (que ha determinado en el paso 1c).
   g. Pulse **Aceptar**.
   h. Guarde el trabajo pulsando Control+S.

Invocación de servicios en enlaces interoperables

Los siguientes enlaces están soportados para la interoperatividad con un servicio Open SCA.

- Enlace SCA

  Cuando un módulo SCA WebSphere Process Server invoca un servicio Open SCA a través de un enlace de importación SCA, están soportados los siguientes estilos de invocación:

  – Asíncrono (unidireccional)
  – Síncrono (petición/respuesta)

  La interfaz de importación SCA y la interfaz de servicio Open SCA deben utilizar una interfaz WSDL compatible con la interoperatividad de servicios web (WS-I).

  Tenga en cuenta que el enlace SCA soporta la transacción y la propagación del contexto de seguridad.
- El enlace de servicio web (JAX-WS) con el protocolo SOAP1.1/HTTP o SOAP1.2/HTTP

  La interfaz de importación SCA y la interfaz de servicio Open SCA deben utilizar una interfaz WSDL compatible con la interoperatividad de servicios web (WS-I).

  Además, están soportadas las siguientes calidades de servicio:

  – Transacción atómica de servicios web
  – Seguridad de servicios web
- Enlace EJB

  Se utiliza una interfaz Java para definir la interacción entre un módulo SCA y un servicio Open SCA, cuando se utiliza el enlace EJB.

Tenga en cuenta que el enlace EJB soporta la transacción y la propagación del contexto de seguridad.

- Enlaces JMS

  La interfaz de importación SCA y la interfaz de servicio Open SCA deben utilizar una interfaz WSDL compatible con la interoperatividad de servicios web (WS-I).

  Están soportados los siguientes proveedores JMS:

  – WebSphere Platform Messaging (Enlace JMS)
  – WebSphere MQ (Enlace JMS de MQ)

Nota: Los gráficos de negocio no funcionan entre sí en los enlaces SCA y, por lo tanto, no están soportados en las interfaces utilizadas para interoperar con el paquete de características de WebSphere Application Server para Service Component Architecture.

Tipos de enlace

Utilice los *enlaces* específicos del protocolo con las importaciones y exportaciones para especifica los medios de transporte de datos dentro o fuera de un módulo.

Selección de enlaces apropiados

Existen varios enlaces que están disponibles para adaptarse a las necesidades de la aplicación.

Los enlaces disponibles en WebSphere Integration Developer proporcionan una gama de opciones. Esta lista le ayuda a identificar cuándo un tipo de enlace podría ser más idóneo para las necesidades de la aplicación.

Considere un enlace *SCA* cuando sean aplicables estos factores:

- Todos los servicios están contenidos en módulos WebSphere Integration Developer; es decir, no hay servicios externos.
- Desea separar la función en diferentes módulos SCA que interactúan directamente entre ellos.
- Los módulos están unidos estrechamente.

Considere un enlace de *servicio web* cuando sean aplicables estos factores:

- Necesita acceder a un servicio externo a través de Internet o proporcionar un servicio a través de Internet.
- Los servicios están unidos ligeramente.
- Se prefiere la comunicación síncrona; es decir, una petición de un servicio puede esperar una respuesta de otro.
- El protocolo del servicio externo al que está accediendo o el servicio que desea proporcionar es SOAP/HTTP o SOAP/JMS.

Considere un enlace *HTTP* cuando sean aplicables estos factores:

- Necesita acceder a un servicio externo en Internet o proporcionar un servicio en Internet y está trabajando con otros servicios Web basados en el modelo HTTP (es decir, está utilizando operaciones de interfaz HTTP conocidas públicamente, por ejemplo GET, PUT, DELETE, etc.).
- Los servicios están unidos ligeramente.
- Se prefiere la comunicación síncrona; es decir, una petición de un servicio puede esperar una respuesta de otro.

Considere un enlace *EJB* cuando sean aplicables estos factores:

- El enlace es para un servicio importado que es a su vez un EJB o al que es necesario que accedan los clientes EJB.
- El servicio importado está unido ligeramente.
- Las interacciones EJB sin estado no son necesarias.
- Se prefiere la comunicación síncrona; es decir, una petición de un servicio puede esperar una respuesta de otro.

Considere un enlace *EIS* cuando sean aplicables estos factores:

- Necesita acceder a un servicio en un sistema EIS utilizando un adaptador de recursos.
- Se prefiere la transmisión de datos síncrona respecto a la asíncrona.

Considere un enlace *JMS* cuando sean aplicables estos factores:

Nota: Existen varios tipos de enlaces JMS. Si piensa intercambiar mensajes SOAP utilizando JMS, tenga en cuenta la posibilidad de utilizar el enlace de servicio Web con el protocolo SOAP/JMS. Consulte "Enlaces del servicio Web" en la página 33.

- Necesita acceder a un sistema de mensajería.
- Los servicios están unidos ligeramente.
- Se prefiere la transmisión de datos asíncrona respecto a la síncrona.

Considere un enlace *JMS genérico* cuando sean aplicables estos factores:

- Necesita acceder a un sistema de mensajería de proveedor que no es de IBM.
- Los servicios están unidos ligeramente.
- La fiabilidad es más importante que el rendimiento; es decir, se prefiere la transmisión de datos asíncrona respecto a la síncrona.

Considere un enlace *MQ* cuando sean aplicables estos factores:

- Necesita acceder a un sistema de mensajería WebSphere MQ y necesita utilizar las funciones nativas de MQ.
- Los servicios están unidos ligeramente.
- La fiabilidad es más importante que el rendimiento; es decir, se prefiere la transmisión de datos asíncrona respecto a la síncrona.

Considere un enlace *JMS de MQ* cuando sean aplicables estos factores:

- Necesita acceder a un sistema de mensajería WebSphere MQ pero puede hacerlo dentro de un contexto JMS; es decir, el subconjunto JMS de funciones es suficiente para la aplicación.
- Los servicios están unidos ligeramente.
- La fiabilidad es más importante que el rendimiento; es decir, se prefiere la transmisión de datos asíncrona respecto a la síncrona.

Enlaces SCA

Un enlace SCA (Service Component Architecture) permite a un servicio comunicarse con otros servicios de otros módulos. Una importación con un enlace SCA le permite acceder a un servicio de otro módulo SCA. Una exportación con un enlace SCA le permite ofrecer un servicio a otros módulos.

Utilice WebSphere Integration Developer para generar y configurar enlaces SCA en importaciones y exportaciones en módulos SCA.

Si los módulos se ejecutan en el mismo servidor o se despliegan en el mismo clúster, un enlace SCA es el enlace más fácil y rápido para utilizar.

Después de que se despliegue en el servidor el módulo que contiene el enlace SCA, puede utilizar la consola administrativa para ver la información sobre el enlace o, en el caso de un enlace de importación, para cambiar las propiedades seleccionadas del enlace.

Enlaces del servicio Web

Un enlace de servicio Web es la forma de transmitir mensajes de un componente SCA (Service Component Architecture) a un servicio Web (y viceversa).

Visión general de enlaces de servicio web

Un enlace de importación de servicios Web permite llamar a un servicio Web externo desde el componente SCA (Service Component Architecture). Un enlace de exportación de servicios Web permite exponer los componentes SCA a clientes como servicios Web.

Con un enlace de servicio web, acceda a los servicios externos utilizando los mensajes SOAP interoperables y las calidades de servicio (QoS).

Utilice WebSphere Integration Developer para generar y configurar enlaces de servicio Web en importaciones y exportaciones en módulos SCA. Los siguientes tipos de enlaces de servicio web están disponibles:

- SOAP1.2/HTTP y SOAP1.1/HTTP

  Estos enlaces se basan en la API de Java para los servicios web XML (JAX-WS), una API de programación Java para crear servicios web.

  – Utilice SOAP1.2/HTTP si el servicio web cumple la especificación SOAP 1.2.

  – Utilice SOAP1.1/HTTP si el servicio web es compatible con la especificación SOAP 1.1.

  Cuando seleccione uno de estos enlaces, puede enviar los adjuntos con los mensajes SOAP.

  Los enlaces de servicio web trabajan con los mensajes SOAP estándar. Sin embargo a través del uso de uno de los enlaces JAX-WS de servicio web, puede personalizar la forma en la que se analizan o escriben los mensajes SOAP. Por ejemplo, puede manejar elementos no estándar en mensajes SOAP o aplicar proceso adicional en el mensaje SOAP. Cuando configure el enlace, especifique un manejador de datos personalizados que realice este proceso en el mensaje SOAP.

- SOAP1.1/HTTP

  Utilice este enlace si desea crear servicios web que utilicen un mensaje codificado con SOAP basado en el RPC basado en la API de Java para XML (JAX-RPC).

- SOAP1.1/JMS

  Utilice este enlace para enviar o recibir mensajes SOAP utilizando un destino JMS (Java Message Service).

Independientemente del transporte (HTTP o JMS) que se utiliza para transportar el mensaje SOAP, los enlaces de servicio web siempre manejan las interacciones de petición/respuesta de forma síncrona. La hebra que realiza la invocación en el proveedor de servicios se bloquea hasta que se recibe una respuesta del proveedor. Consulte la “invocación síncrona” si desea más información sobre este estilo de invocación.

Importante: Las combinaciones siguientes de los enlaces de servicio Web no se pueden utilizar en exportaciones en el mismo módulo. Si necesita exponer componentes utilizando más de uno de estos enlaces de exportación, necesita tener cada uno en un módulo independiente y, a continuación, conectarse esos módulos a los componentes utilizando el enlace SCA:

- SOAP 1.1/JMS y SOAP 1.1/HTTP utilizando JAX-RPC
- SOAP 1.1/HTTP utilizando JAX-RPC y SOAP 1.1/HTTP utilizando JAX-WS
- SOAP 1.1/HTTP utilizando JAX-RPC y SOAP 1.2/HTTP utilizando JAX-WS

Tras desplegarse en el servidor el módulo SCA que contiene el enlace Web, podrá utilizar la consola administrativa para ver información sobre el enlace, o para cambiar propiedades seleccionadas de este último.

Nota: Los servicios web permiten a las aplicaciones interoperar utilizando descripciones estándar de servicios y formatos estándar para los mensajes que intercambian. Por ejemplo, los enlaces de importación y exportación de servicio web pueden interoperar con servicios que se han implementado mediante Web Services Enhancements (WSE) Versión 3.5 y Windows® Communication Foundation (WCF) Versión 3.5 for Microsoft® .NET. Cuando interopere con este tipo de servicios, debe asegurarse de lo siguiente:

- El archivo WSDL (lenguaje de descripción de servicios web) que se utiliza para acceder a una exportación de servicio web incluye un valor de acción SOAP no vacío para cada operación de la interfaz.
- El cliente de servicio Web establece la cabecera SOAPAction o la cabecera wsa:Action al enviar mensajes a una exportación de servicio Web.

Propagación de cabecera SOAP

Cuando se manejan mensajes SOAP, es posible que tenga que acceder a información de determinadas cabeceras SOAP de mensajes recibidos, asegúrese de que los mensajes con cabeceras SOAP se han enviado con valores específicos, o que permiten a las cabeceras SOAP pasar entre un módulo.

Cuando configure un enlace de servicio web en WebSphere Integration Developer, podrá indicar que desea que las cabeceras SOAP se propaguen.

- Cuando las peticiones se reciban en una exportación o las respuestas se reciban en una importación, se puede acceder a la información de cabecera SOAP, lo que permite que la lógica del módulo se base en los valores de cabecera y que permite que estas cabeceras se modifiquen.
- Cuando las peticiones se envían de una exportación o de respuestas enviadas de una importación, las cabeceras SOAP se pueden incluir en estos mensajes.

El formato y la presencia de las cabeceras SOAP propagadas podrían verse afectados por conjuntos de políticas configurados en la importación o exportación, tal como se explica en Tabla 9 en la página 36.

Para configurar la propagación de cabeceras SOAP para una importación o exportación, seleccione (desde la vista Propiedades de WebSphere Integration Developer) el separador **Propagar cabecera de protocolo** y seleccione las opciones que necesite.

Cabecera WS-Addressing

La cabecera WS-Addressing se puede propagar mediante el enlace de servicio web (JAX-WS).

Cuando propague la cabecera WS-Addressing, debe tener en cuenta la siguiente información:

- Si habilita la propagación para la cabecera WS-Addressing, la cabecera se propagará en el módulo bajo las circunstancias siguientes:
  - Cuando las peticiones se reciben en una exportación
  - Cuando las respuestas se reciben en una importación
- La cabecera WS-Addressing no se propaga en mensajes de salida desde WebSphere Process Server (es decir, la cabecera no se propaga cuando las peticiones se envían desde una importación o cuando las respuestas se envían desde la exportación).

Cabecera WS-Security

La cabecera WS-Security se puede propagar mediante el enlace de servicio web (JAX-WS) y, también, el enlace de servicio web (JAX-RPC).

La especificación WS-Security de servicios web describe las mejoras en las mensajería SOAP para proporcionar calidad de protección a través de la integridad de los mensajes, la confidencialidad de mensajes y una autenticación única de mensajes. Estos mecanismos se pueden utilizar para acomodar una gran variedad de modelos de seguridad y tecnologías de cifrado.

Cuando propague la cabecera WS-Security, debe tener en cuenta la siguiente información:

- Si habilita la propagación para la cabecera WS-Security, la cabecera se propagará entre el módulo bajo las circunstancias siguientes:
  - Cuando las peticiones se reciben en una exportación
  - Cuando las peticiones se envían desde una importación
  - Cuando las respuestas se reciben en una importación
- La cabecera *no* se propaga, de forma predeterminada, cuando las respuesta se envían desde la exportación. Sin embargo, si establece la propiedad JVM `WSSECURITY.ECHO.ENABLED` en `true`, la cabecera se propagará cuando las respuestas se envían desde la exportación. En este caso, si la cabecera WS-Security de la vía de acceso de la petición no se ha modificado, se podría hacer eco de las cabeceras WS-Security automáticamente desde las peticiones en las respuestas.
- El formato exacto del mensaje SOAP enviado desde una importación para una petición o desde una exportación para una respuesta podrían no coincidir exactamente con el mensaje SOAP que se recibió originalmente. Por este motivo, cualquier firma digital se debe considerar como no válido. Si es necesaria una firma digital en los mensajes enviados, se debe establecer utilizando el conjunto de políticas apropiado y las cabeceras WS-Security relacionadas con la firma digital en los mensajes recibidos se deben eliminar dentro del módulo.

Para propagar la cabecera WS-Security, debe incluir el esquema WS-Security con el módulo de aplicación. Consulte “Cómo incluir el esquema WS-Security en un módulo de aplicación ” en la página 37 para el procedimiento para incluir el esquema.

Cómo se propagan las cabeceras

La forma en la que se propagan las cabeceras depende del valor de la política de seguridad en el enlace de importación o exportación, tal como se muestra en Tabla 9:

Tabla 9. Cómo se pasan las cabeceras de seguridad

	Enlace de exportación sin política de seguridad	**Enlace de exportación con política de seguridad**
Enlace de importación sin política de seguridad	Las cabeceras de seguridad se pasan tal cual a través del módulo. No se descifran. Las cabeceras se envían en la salida del mismo modo en que se reciben. La firma digital podría ser no válida.	Las cabeceras de seguridad se descifran y se pasan a través del módulo con la verificación y autenticación de firma. Las cabeceras descifradas se envían en la salida. La firma digital podría ser no válida.
Enlace de importación con política de seguridad	Las cabeceras de seguridad se pasan tal cual a través del módulo. No se descifran. Las cabeceras no se deben propagar en la importación. De lo contrario, se produce un error debido a la duplicación.	Las cabeceras de seguridad se descifran y se pasan a través del módulo con la verificación y autenticación de firma. Las cabeceras no se deben propagar en la importación. De lo contrario, se produce un error debido a la duplicación.

Configure los conjuntos de políticas apropiados en los enlaces de exportación e importación, porque esto aísla el solicitante de servicio de cambios en la configuración o los requisitos QoS del proveedor de servicios. El tener las cabeceras SOAP estándar visibles en un módulo se puede utilizar para influir en el proceso (por ejemplo, el registro y el rastreo) en el módulo. La propagación de cabeceras SOAP entre un módulo desde un mensaje recibido en un mensaje enviado no significa que se reduzcan las ventajas del aislamiento del módulo.

Las cabeceras estándar como, por ejemplo, las cabeceras WS-Security, no se deben propagar en una petición para una importación o respuesta a una exportación cuando la importación o exportación tiene un conjunto de políticas asociado que normalmente produciría la generación de estas cabeceras. De lo contrario, se producirá un error debido a una duplicación de las cabeceras. En lugar de esto, las cabeceras se eliminarán de forma explícita o el enlace de importación o exportación configurado para impedir la propagación de las cabeceras de protocolo.

Acceso a cabeceras SOAP

Cuando un mensaje que contiene cabeceras SOAP se recibe desde una importación o exportación de servicio web, las cabeceras se colocan en la sección de cabeceras del objeto de mensaje de servicio (SMO). Puede acceder a la información de cabecera, tal como se describe en "Acceso a la información de cabecera SOAP del SMO".

Cómo incluir el esquema WS-Security en un módulo de aplicación

El siguiente procedimiento describe los pasos para incluir el esquema en el módulo de aplicación:

- Si el sistema en el que se ejecuta WebSphere Integration Developer tiene acceso a Internet, realice los pasos siguientes:
  1. En la perspectiva Business Integration, seleccione **Dependencias** para el proyecto.
  2. Expanda **Recursos predefinidos** y seleccione **Archivos de esquema WS-Security 1.0** o **Archivos de esquema WS-Security 1.1** para importar el esquema en el módulo.
  3. Limpie y vuelva a crear el proyecto.
- Si un sistema en el que se ejecuta WebSphere Integration Developer no tiene acceso a Internet, puede descargar el esquema en un segundo sistema que tenga acceso a Internet. A continuación, puede copiarlo en el sistema en el que se ejecuta WebSphere Integration Developer.
  1. Desde el sistema que tiene acceso a Internet, descargue el esquema remoto:
     a. Pulse **Archivo** → **Importar** → **Business Integration** → **WSDL y XSD**.
     b. Seleccione **WSDL remoto** o **Archivo XSD**.
     c. Importe los siguientes esquemas:

        http://www.w3.org/2003/05/soap-envelope/

        http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd

        http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd
  2. Copie los esquemas en el sistema que no tiene acceso a Internet.
  3. Desde el sistema que no tiene acceso a Internet, importe el esquema:
     a. Pulse **Archivo** → **Importar** → **Business Integration** → **WSDL y XSD**.
     b. Seleccione **WSDL local** o **Archivo XSD**.
  4. Cambie las ubicaciones del esquema para oasis-wss-wssecurity_secext-1.1.xsd:
     a. Abra el esquema en *ubicación_lugar_trabajo/nombre_módulo/*`StandardImportFilesGen/oasis-wss-wssecurity-secext-1.1.xsd`.
     b. Cambie:

        ```
        <xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
        schemaLocation='http://www.w3.org/2003/05/soap-envelope/'/>
        ```

        a:

        ```
        <xs:import namespace='http://www.w3.org/2003/05/soap-envelope'
        schemaLocation='../w3/_2003/_05/soap_envelope.xsd'/>
        ```

     c. Cambie:

        ```
        <xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
        schemaLocation='http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/xenc-schema.xsd'/>
        ```

        a:

        ```
        <xs:import namespace='http://www.w3.org/2001/04/xmlenc#'
        schemaLocation='../w3/tr/_2002/rec_xmlenc_core_20021210/xenc-schema.xsd'/>
        ```
  5. Cambie la ubicación del esquema para oasis-200401-wss-wssecurity-secext-1.0.xsd:
     a. Abra el esquema en *ubicación_lugar_trabajo/nombre_módulo/*`StandardImportFilesGen/oasis-200401-wss-wssecurity-secext-1.0.xsd`.
     b. Cambie:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="http://www.w3.org/TR/xmldsig-core/xmldsig-core-schema.xsd"/>
```

a:

```
<xsd:import namespace="http://www.w3.org/2000/09/xmldsig#"
schemaLocation="../w3/tr/_2002/rec_xmldsig_core_20020212/xmldsig-core-schema.xsd"/>
```

6. Limpie y vuelva a crear el proyecto.

Adjuntos en mensajes SOAP

Puede enviar y recibir mensajes SOAP que incluyen datos binarios (como archivos PDF o imágenes JPEG) como adjuntos. Los adjuntos puede ser *referenciados* (es decir, representados explícitamente como partes de mensaje en la interfaz de servicio) o *no referenciados* (en los que se pueden incluir los números y tipos arbitrarios de adjuntos).

Un adjunto referenciado puede estar representado de una de las formas siguientes:

- Como un elemento wsi:swaRef-typed en el esquema de mensaje

  Los adjuntos definidos utilizando el tipo wsi:swaRef type son compatibles con Web Services Interoperability Organization (WS-I) *Attachments Profile Versión 1.0* (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html), que define cómo se relacionan elementos de mensaje con partes MIME.
- Como parte de mensaje de nivel superior, mediante un tipo de esquema binario

  Los adjuntos representados como partes de mensaje de nivel superior son compatibles con la especificación de *Mensajes SOAP con adjuntos* (http://www.w3.org/TR/SOAP-attachments).

Un adjunto no referenciado se transporta en un mensaje SOAP sin ninguna representación en el esquema del mensaje.

En todos los casos, el enlace SOAP de WSDL debe incluir un enlace MIME para los adjuntos a utilizar y el tamaño máximo de los archivos adjuntos no debe superar los 20 MB.

Nota: Para enviar o recibir mensajes SOAP con adjuntos, debe utilizar uno de los enlaces de servicio web basados en la API de Java para servicios web XML (JAX-WS).

Adjuntos referenciados: elementos de tipo swaRef:

Puede enviar y recibir mensajes SOAP que incluyan adjuntos representados en la interfaz de servicio como elementos de tipo swaRef.

Un elemento de tipo swaRef se define en Web Services Interoperability Organization (WS-I) *Attachments Profile* Versión 1.0 (http://www.ws-i.org/Profiles/AttachmentsProfile-1.0.html), que define cómo están relacionados los elementos de mensaje con partes MIME.

Nota: No se garantiza que los mensajes SOAP producidos o consumidos se ajusten a WS-I Attachments Profile. En particular, no se soporta la "codificación de la parte de id de contenido," como se describe en la sección 3.8 de WS-I Attachments Profile 1.0.

En el mensaje SOAP, el cuerpo SOAP contiene un elemento de tipo swaRef que identifica el ID de contenido del adjunto.

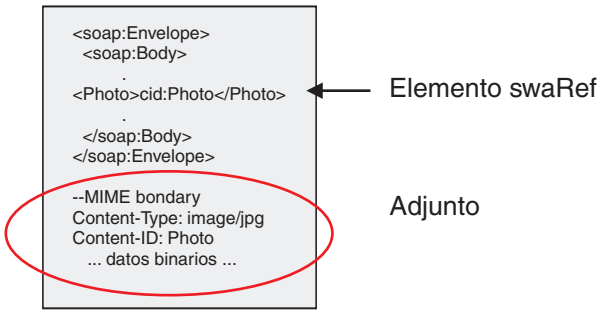


Figura 14. Un mensaje SOAP con un elemento swaRef

El WSDL para este mensaje SOAP contiene un elemento de tipo swaRef dentro de una parte de mensaje que identifica el adjunto.

```
<element name="sendPhoto">
      <complexType>
       <sequence>            <element name="Photo" type="wsi:swaRef"/>
       </sequence>       </complexType>          </element>
```

El WSDL también debe contener un enlace MIME que indica los mensajes de varias partes MIME que se deben utilizar.

Nota: El WSDL *no* incluya un enlace MIME para el elemento de mensaje de tipo swaRef específico, porque los enlaces MIME sólo se aplican en las partes de mensaje de nivel superior.

Los adjuntos representados como elementos de tipo swaRef se pueden propagar sólo entre componentes de flujo de mediación. Si otro tipo de componente debe acceder a un archivo adjunto, o el archivo adjunto se debe propagar a éste, utilice un componente de flujo de mediación para mover el archivo adjunto a una ubicación a la que pueda acceder ese componente.

Proceso de entrada de adjuntos

Utilice WebSphere Integration Developer para configurar un enlace de exportación para recibir el adjunto. Cree un módulo y sus operaciones e interfaz asociadas, que incluyen un elemento de tipo swaRef. Cree un enlace de servicio web (JAX-WS).

Nota: Consulte el tema “Trabajo con adjuntos” en el centro de información de WebSphere Integration Developer si desea más información.

Cuando un cliente pasa un mensaje SOAP con un adjunto swaRef a un componente SCA (Service Component Architecture), en primer lugar, el enlace de exportación de servicio web (JAX-WS) elimina el adjunto. A continuación, analiza la parte SOAP del mensaje y crea un objeto de negocio. Finalmente, el enlace establece el ID de contenido del adjunto en el objeto de negocio.

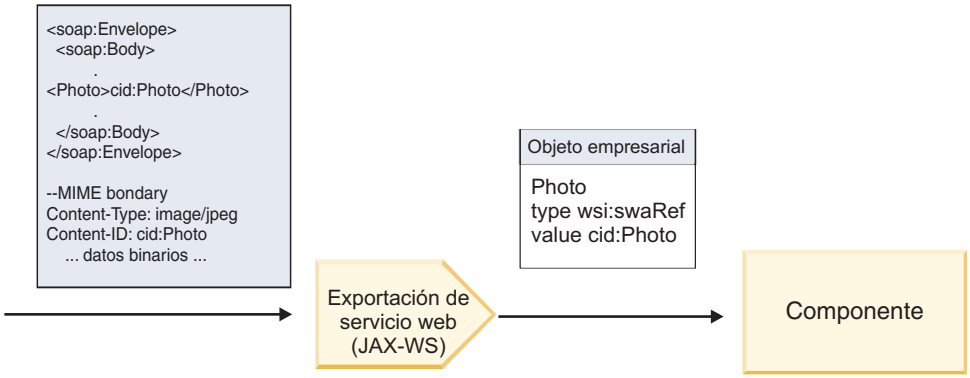


Figura 15. Cómo el enlace de exportación de servicio web (JAX-WS) procesa un mensaje SOAP con un adjunto swaRef

Acceso a los metadatos de adjunto en un componente de flujo de mediación

Tal como se indica en Figura 16, cuando los componentes acceden a los adjuntos swaRef, el identificador del contenido del adjunto aparece como un elemento del tipo swaRef.

Cada adjunto de un mensaje SOAP también tiene un elemento `attachments` correspondiente en SMO. Cuando se utiliza el tipo WS-I swaRef, el elemento `attachments` incluye el tipo de contenido de adjunto y el ID de contenido, así como los datos binarios reales del adjunto.

Para obtener el valor de un adjunto swaRefm, por lo tanto, es necesario obtener el valor del elemento del tipo swaRef y, a continuación, localizar el elemento `attachments` con el valor `contentID` correspondiente. Tenga en cuenta que el valor `contentID` normalmente tiene el prefijo `cid:` eliminado del valor swaRef.

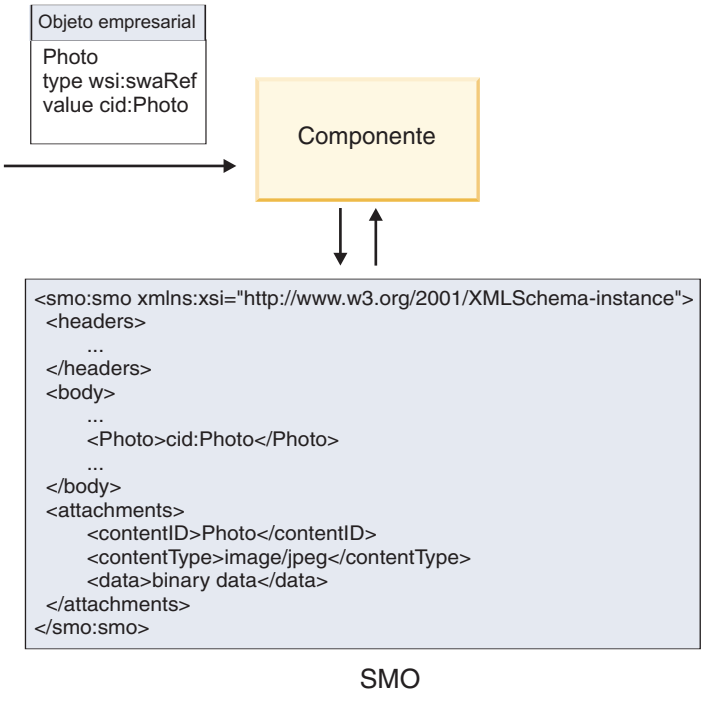


Figura 16. Cómo aparecen los adjuntos swaRef en SMO

Proceso de salida

Utilice WebSphere Integration Developer para configurar un enlace de importación de servicio web (JAX-WS) para invocar un servicio web externo. El enlace de importación se configura con un documento WSDL que describe el servicio web que se debe invocar y define el adjunto que se pasará al servicio web.

Cuando un enlace de importación de servicio web (JAX-WS) recibe un mensaje SCA, los elementos de tipo swaRef se envían como adjuntos, si la importación se conecta a un componente de flujo de mediación y el elemento de tipo swaRef tiene un elemento `attachments` correspondiente.

Para el proceso de salida, los elementos de tipo swaRef siempre se envían con sus valores de ID de contenido; sin embargo, el módulo de mediación debe asegurarse de que hay un elemento `attachments` correspondiente con un valor `contentID` coincidente.

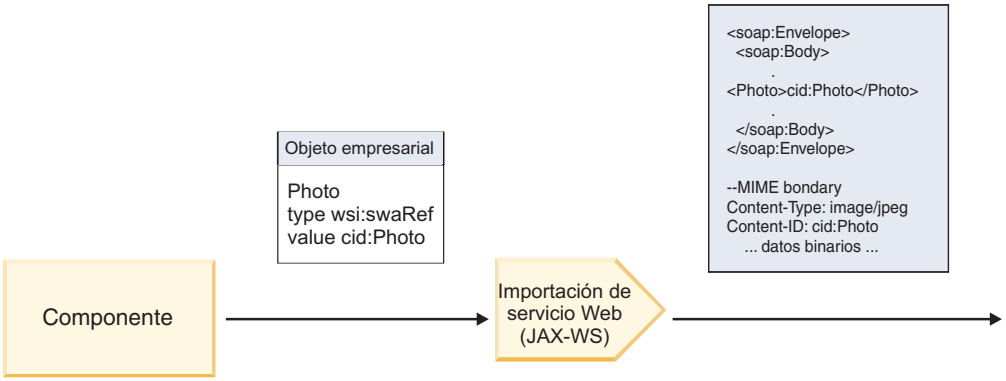


Figura 17. Cómo genera el enlace de importación de servicio web (JAX-WS) un mensaje SOAP con un adjunto swaRef

Definición de metadatos de adjunto en un componente de flujo de mediación

Si, en SMO, existe un valor del elemento del tipo swaRef y un elemento `attachments`, el enlace prepara el mensaje SOAP (con el adjunto) y lo envía a un destinatario.

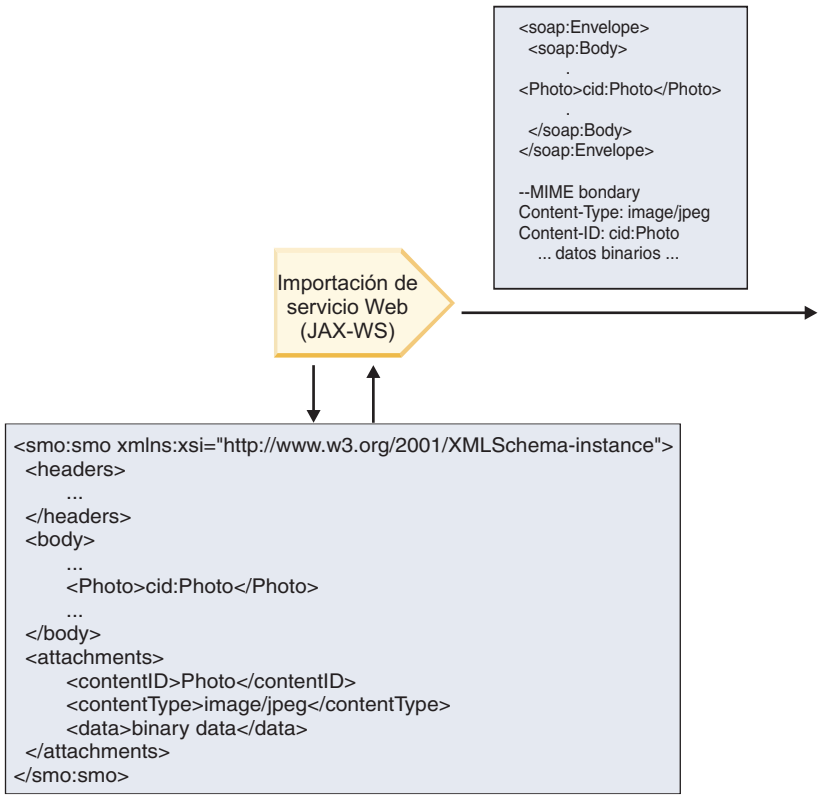


Figura 18. Cómo se accede a un adjunto swaRef en SMO para crear el mensaje SOAP

El elemento `attachments` está presente en SMO sólo si un componente de flujo de mediación está conectado directamente a la importación o exportación; no pasa entre otros tipos de componente. Si los valores son necesarios en un módulo que contiene otros tipos de componente, se debe utilizar un componente de flujo de mediación para copiar los valores en una ubicación en el módulo en la que se pueda acceder a éstos y otro componente de flujo de mediación utilizado para establecer los valores correctos antes de una invocación de salida mediante una importación de servicio web.

Importante: Tal y como se describe en "Representación XML de SMO," la primitiva de mediación Transformación XSL transforma mensajes mediante una transformación XSLT 1.0. La transformación opera en una serialización XML del SMO. La primitiva de mediación Transformación XSL permite especificar la raíz de la serialización, y el elemento raíz del documento XML refleja esta raíz.

Cuando se envían mensajes SOAP con archivos adjuntos, el elemento raíz que elija determina qué archivos adjuntos se propagan.

- Si se utiliza "/body" como la raíz de la correlación XML, se propagan de forma predeterminada todos los archivos adjuntos por la correlación.
- Si se utiliza "/" como la raíz de la correlación, se puede controlar la propagación de los archivos adjuntos.

Adjuntos referenciados: partes de mensaje de nivel superior:

Puede enviar y recibir mensajes SOAP que incluyen adjuntos binarios que se declaran como partes de la interfaz de servicio.

En un mensaje SOAP de varias partes MIME, el cuerpo SOAP es la primera parte del mensaje y los archivos adjuntos se encuentran en las partes subsiguientes. Las referencias a los adjuntos se incluyen en el cuerpo SOAP.

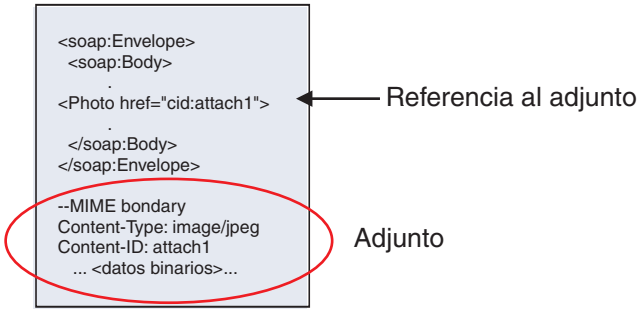


Figura 19. Un mensaje SOAP con un adjunto referenciado

¿Cuál es la ventaja de enviar o recibir un adjunto referenciado en un mensaje SOAP? Los datos binarios que conforman el adjunto (que a menudo es bastante grande) se alojan de forma separada del cuerpo del mensaje SOAP, de forma que no se deben analizar como XML. Esto genera un proceso más eficaz que si los datos binarios se incluyeran dentro de un elemento XML.

Proceso de entrada de adjuntos referenciados

Utilice WebSphere Integration Developer para configurar el enlace de exportación. Cree un módulo y sus operaciones e interfaz asociadas. A continuación, cree un enlace de servicio web (JAX-WS). La página Adjuntos referenciados visualiza todas las partes binarias de la operación creada y puede seleccionar qué partes serán adjuntos.

Nota: Sólo las partes del mensaje de nivel superior (es decir, los elementos definidos en WSDL portType como partes del mensaje de entrada o de salida) que tiene un tipo binario (ya sea base64Binary o hexBinary) se pueden enviar o recibir como adjuntos referenciados.
Consulte el tema “Trabajo con adjuntos” en el centro de información de WebSphere Integration Developer si desea más información.

Cuando un cliente pasa un mensaje SOAP con un adjunto a un componente SCA (Service Component Architecture), en primer lugar, el enlace de exportación de servicio web (JAX-WS) elimina el adjunto. A continuación, analiza la parte SOAP del mensaje y crea un objeto de negocio. Finalmente, el enlace establece el binario del adjunto en el objeto de negocio.

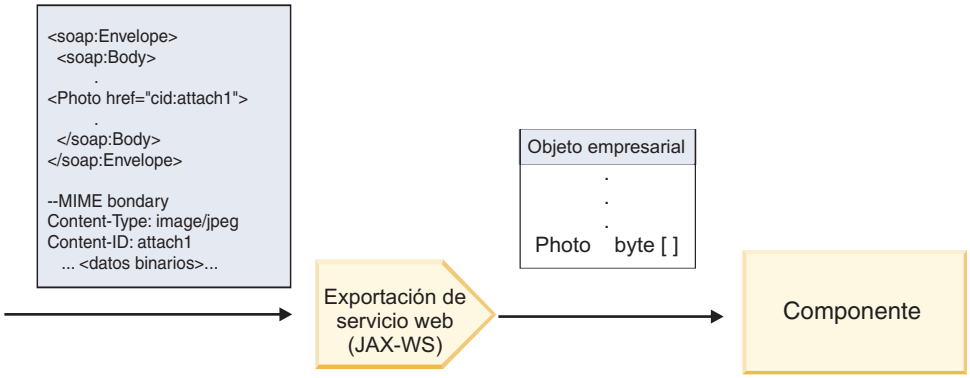


Figura 20. Cómo el enlace de exportación de servicio web (JAX-WS) procesa un mensaje SOAP con un adjunto referenciado

Acceso a los metadatos de adjunto en un componente de flujo de mediación

Tal como se muestra en Figura 20, cuando los componentes acceden a los adjuntos referenciados, los datos de adjunto aparecen como una matriz de bytes.

Cada adjunto referenciado de un mensaje SOAP también tiene un elemento `attachments` correspondiente en SMO. El elemento `attachments` incluye el tipo del contenido del adjunto y la vía de acceso del elemento del cuerpo del mensaje donde se incluye el adjunto.

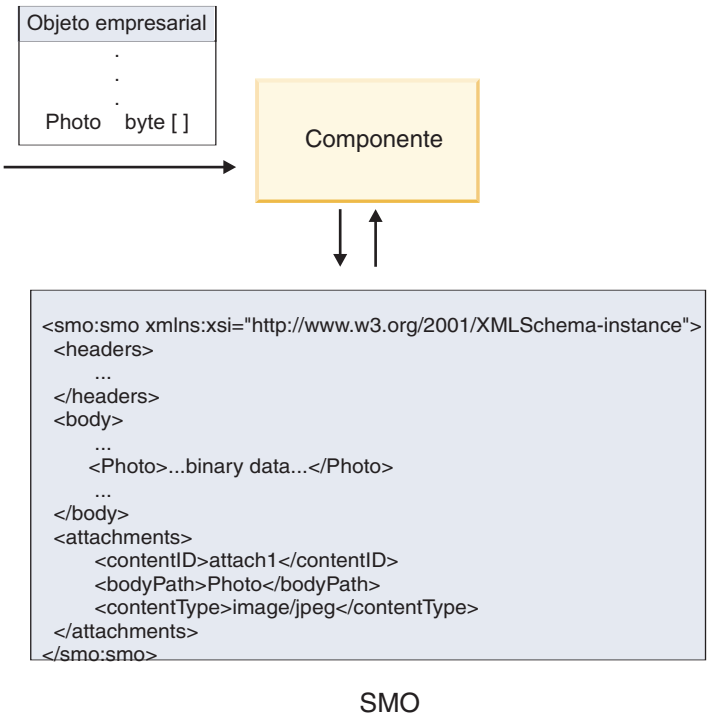


Figura 21. Cómo aparecen los adjuntos referenciados en SMO

Importante: La vía de acceso del elemento del cuerpo del mensaje no se actualiza automáticamente si el mensaje se transforma y se mueve el adjunto. Puede utilizar un flujo de mediación para actualizar el elemento `attachments` con la nueva vía de acceso (por ejemplo, como parte de la transformación o utilizando un sistema separado para establecer los elementos de un mensaje).

Proceso de salida de adjuntos referenciados

Utilice WebSphere Integration Developer para configurar un enlace de importación de servicio web (JAX-WS) para invocar un servicio web externo. El enlace de importación se configura con un documento WSDL que describe el servicio web que se debe invocar y define qué partes del mensaje se deben pasar como adjuntos.

Nota: La parte que representa un adjunto, tal como se define en el WSDL, debe ser un tipo sencillo (ya sea base64Binary o hexBinary). Si una parte se define mediante un complexType, dicha parte no se trata como un adjunto.

El enlace de importación utiliza la información en el SMO para determinar cómo se envían las partes binarias del mensaje de nivel superior como adjuntos.

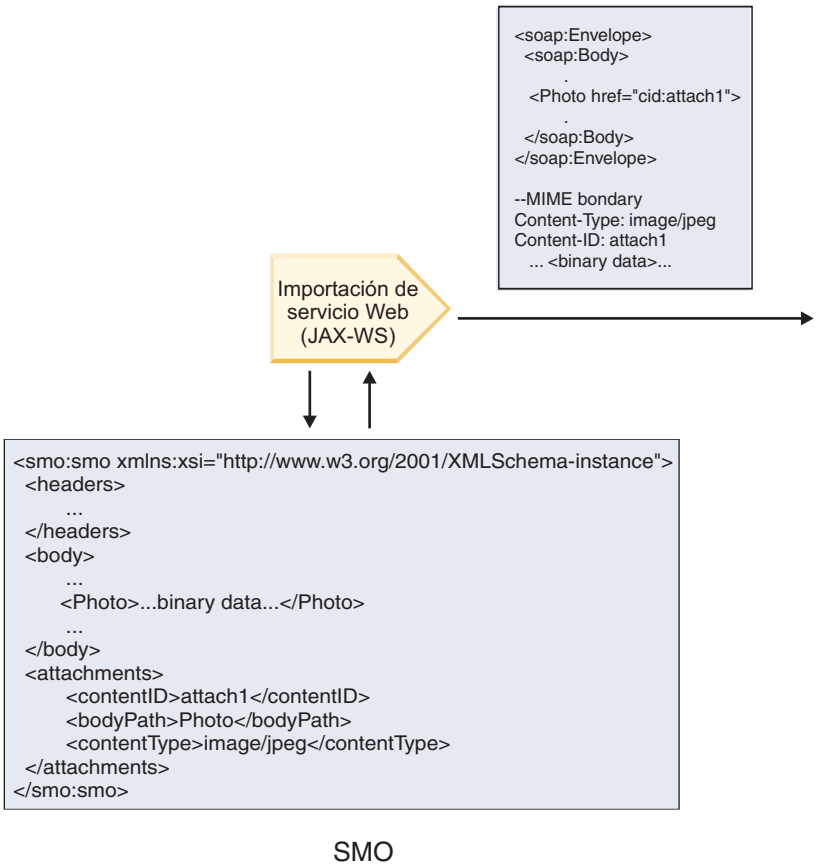


Figura 22. Cómo se accede al adjunto referenciado en SMO para crear el mensaje SOAP

El elemento `attachments` está presente en SMO sólo si un componente de flujo de mediación está conectado directamente a la importación o exportación; no pasa entre otros tipos de componente. Si los valores son necesarios en un módulo que contiene otros tipos de componente, se debe utilizar un componente de flujo de mediación para copiar los valores en una ubicación en el módulo en la que se pueda acceder a éstos y otro componente de flujo de mediación utilizado para establecer los valores correctos antes de una invocación de salida mediante una importación de servicio web.

El enlace utiliza una combinación de las siguientes condiciones para determinar cómo se envía el mensaje (o si se envía):

- Si hay un enlace WSDL MIME para la parte binaria del mensaje de nivel superior y, si existe, cómo se define el tipo de contenido
- Si hay un elemento `attachments` en el SMO cuyo valor `bodyPath` hace referencia a una parte binaria de nivel superior

Cómo se crean los adjuntos cuando existe un elemento `attachment` en el SMO

La tabla siguiente muestra cómo se crea un adjunto y cómo se envía si el SMO contiene un elemento `attachment` con un `bodyPath` que coincida con una parte del nombre del mensaje:

Tabla 10. Cómo se genera el adjunto

Estado del enlace WSDL MIME para la parte binaria del mensaje de nivel superior	**Cómo se crea y se envía el mensaje**
Presente con una de las acciones siguientes: • No hay tipo de contenido definido para la parte del mensaje • Hay definidos varios tipos de contenido • Tipo de contenido comodín definido	La parte de mensaje se envía como un adjunto. `Content-Id` se establece en el valor en el elemento de adjuntos (attachments) si está presente; de lo contrario, se genera uno. `Content-Type` se establece en el valor del elemento de adjuntos (attachments) si está presente; de lo contrario, se establece en `application/octet-stream`.
Presente con contenido único sin caracteres comodín para la parte del mensaje	La parte de mensaje se envía como un adjunto. `Content-Id` se establece en el valor en el elemento de adjuntos (attachments) si está presente; de lo contrario, se genera uno. `Content-Type` se establece en el valor del elemento de adjuntos (attachments) si está presente; de lo contrario, se establece en el tipo definido en el elemento de contenido WSDL MIME.
No presente	La parte de mensaje se envía como un adjunto. `Content-Id` se establece en el valor en el elemento de adjuntos (attachments) si está presente; de lo contrario, se genera uno. `Content-Type` se establece en el valor del elemento de adjuntos (attachments) si está presente; de lo contrario, se establece en `application/octet-stream`. **Nota:** Enviar partes del mensaje como adjuntos si no se han definido como tales en WSDL puede romper la compatibilidad con el perfil 1.0 de WS-I Attachments y se debe evitar siempre que sea posible.

Cómo se crean adjuntos cuando no existe ningún elemento `attachment` en el SMO

La tabla siguiente muestra cómo se crea un adjunto y cómo se envía si el SMO no contiene ningún elemento `attachment` con un `bodyPath` que coincida con una parte de nombre del mensaje:

Tabla 11. Cómo se genera el adjunto

Estado del enlace WSDL MIME para la parte binaria del mensaje de nivel superior	Cómo se crea y se envía el mensaje
Presente con una de las acciones siguientes: • No hay tipo de contenido definido para la parte del mensaje • Hay definidos varios tipos de contenido • Tipo de contenido comodín definido	La parte de mensaje se envía como un adjunto. Se genera `Content-Id`. `Content-Type` es establece en `application/octet-stream`.
Presente con contenido único sin caracteres comodín para la parte del mensaje	La parte de mensaje se envía como un adjunto. Se genera `Content-Id`. `Content-Type` se establece en el tipo definido en el elemento de contenido WSDL MIME.
No presente	La parte del mensaje no se envía como un adjunto.

Importante: Tal y como se describe en "Representación XML de SMO," la primitiva de mediación Transformación XSL transforma mensajes mediante una transformación XSLT 1.0. La transformación opera en una serialización XML del SMO. La primitiva de mediación Transformación XSL permite especificar la raíz de la serialización, y el elemento raíz del documento XML refleja esta raíz.

Cuando se envían mensajes SOAP con archivos adjuntos, el elemento raíz que elija determina qué archivos adjuntos se propagan.

- Si se utiliza "/body" como la raíz de la correlación XML, se propagan de forma predeterminada todos los archivos adjuntos por la correlación.
- Si se utiliza "/" como la raíz de la correlación, se puede controlar la propagación de los archivos adjuntos.

Adjuntos no referenciados:

Puede enviar y recibir archivos adjuntos *no referenciados* que no estén declarados en la interfaz de servicio.

En un mensaje SOAP de varias partes MIME, el cuerpo SOAP es la primera parte del mensaje y los archivos adjuntos se encuentran en las partes subsiguientes. No se incluye ninguna referencia al archivo adjunto en el cuerpo SOAP.

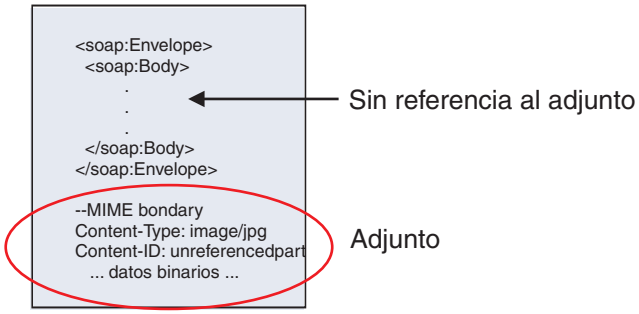


Figura 23. Un mensaje SOAP con un archivo adjunto no referenciado

Puede enviar un mensaje SOAP con un archivo adjunto no referenciado mediante una exportación de servicio Web a una importación de servicio Web. El mensaje de salida, que se envía al servicio Web de destino, contiene el archivo adjunto.

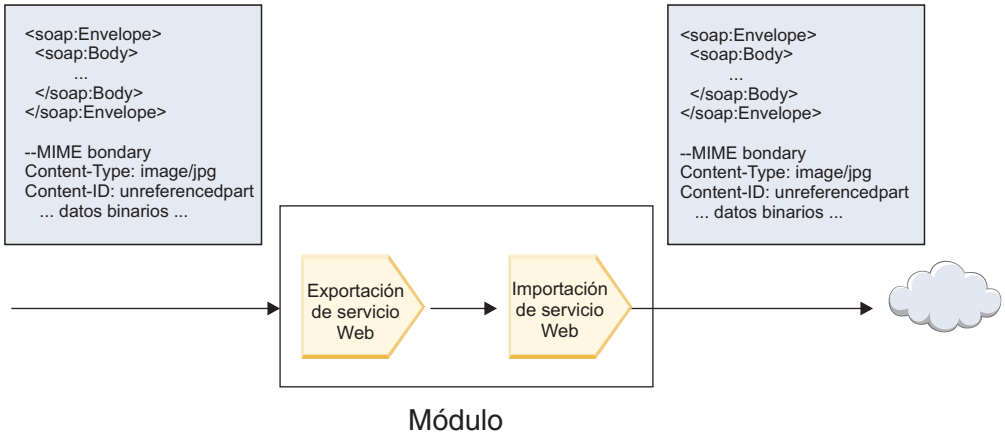


Figura 24. Un archivo adjunto que pasa a través de un módulo SCA

En la Figura 24, el mensaje SOAP, con el archivo adjunto, pasa sin modificaciones.

También puede modificar el mensaje SOAP utilizando un componente de flujo de mediación. Por ejemplo, puede utilizar el componente de flujo de mediación para extraer datos del mensaje SOAP (datos binarios en el cuerpo del mensaje, en este caso) y crear un mensaje SOAP con archivos adjuntos. Los datos se procesan como parte del elemento de archivos adjuntos de un objeto de mensaje de servicio (SMO).

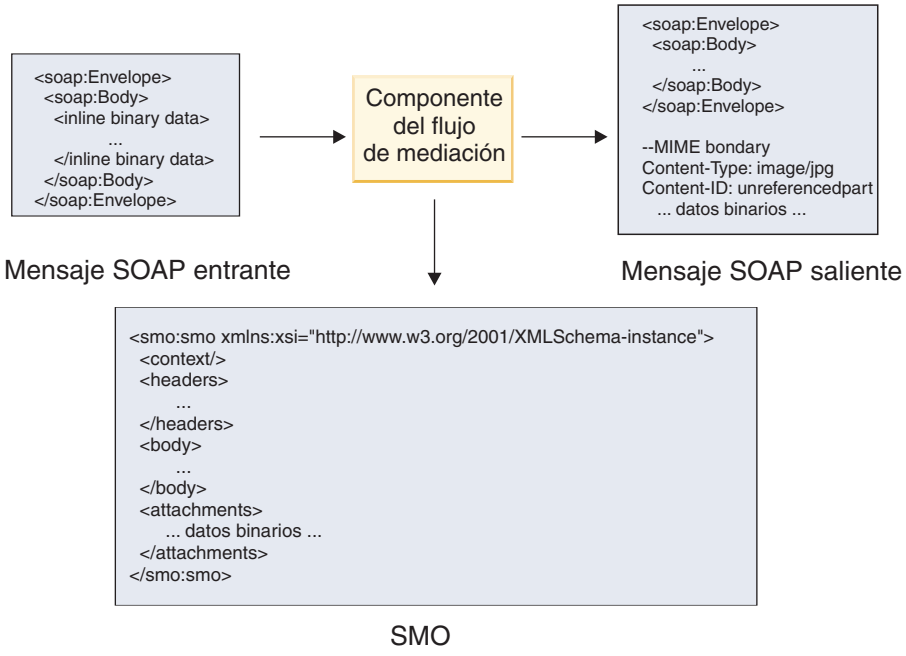


Figura 25. Un mensaje procesado por un componente de flujo de mediación

En sentido inverso, el componente de flujo de mediación puede transformar el mensaje entrante extrayendo y codificando el archivo adjunto y, a continuación, transmitiendo el mensaje sin archivos adjuntos.

En lugar de extraer los datos de un mensaje SOAP entrante para formar un mensaje SOAP con archivos adjuntos, puede obtener los datos de archivo adjunto de un origen externo, como por ejemplo una base de datos.

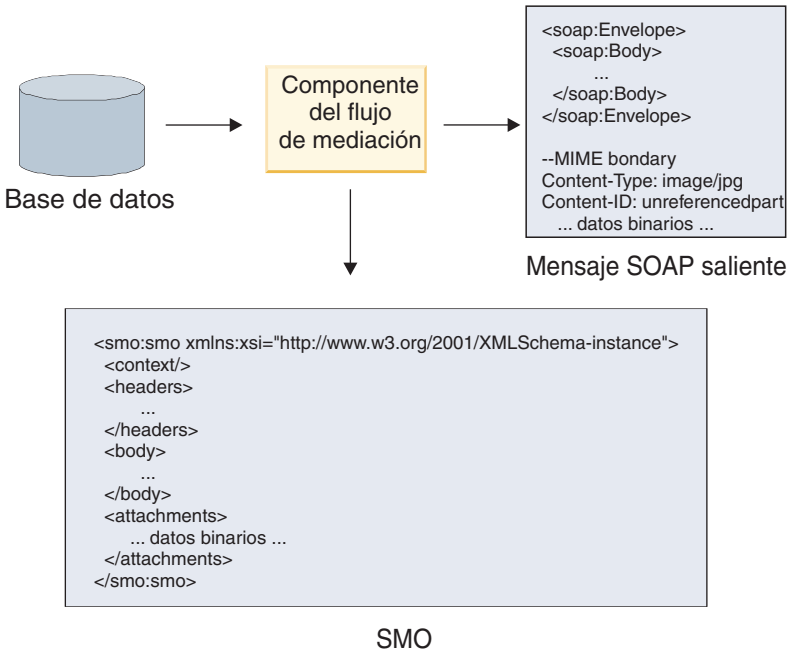


Figura 26. Un archivo adjunto obtenido de una base de datos y añadido al mensaje SOAP

En sentido inverso, el componente de flujo de mediación puede extraer el archivo adjunto de un mensaje SOAP entrante y procesar el mensaje (por ejemplo, almacenar el archivo adjunto en una base de datos).

Los archivos adjuntos no referenciados se pueden propagar sólo entre componentes de flujo de mediación. Si otro tipo de componente debe acceder a un archivo adjunto, o el archivo adjunto se debe propagar a éste, utilice un componente de flujo de mediación para mover el archivo adjunto a una ubicación a la que pueda acceder ese componente.

Importante: Tal y como se describe en "Representación XML de SMO," la primitiva de mediación Transformación XSL transforma mensajes mediante una transformación XSLT 1.0. La transformación opera en una serialización XML del SMO. La primitiva de mediación Transformación XSL permite especificar la raíz de la serialización, y el elemento raíz del documento XML refleja esta raíz.

Cuando se envían mensajes SOAP con archivos adjuntos, el elemento raíz que elija determina qué archivos adjuntos se propagan.

- Si se utiliza "/body" como la raíz de la correlación XML, se propagan de forma predeterminada todos los archivos adjuntos por la correlación.
- Si se utiliza "/" como la raíz de la correlación, se puede controlar la propagación de los archivos adjuntos.

Uso del enlace de estilo de documento WSDL con mensajes de varias partes

La organización WS-I (Web Services Interoperability - Interoperatividad de servicios web) tiene definido un conjunto de normas en relación a la manera en que se deben describir los servicios Web por medio de un WSDL y a la manera en se deben formar los mensajes SOAP correspondientes, a fin de asegurar la interoperatividad.

Estas normas se especifican en *Basic Profile Versión 1.1* de WS-I (http://www.ws-i.org/Profiles/BasicProfile-1.1.html).

En particular, para un enlace SOAP de estilo de documento, el cumplimiento del perfil WS-I necesita que, en un documento WSDL, para una operación que utiliza el estilo de documento, sólo se enlace una parte de mensaje individual al cuerpo de SOAP y que el mensaje SOAP correspondiente a éste contenga un solo elemento hijo que coincida con la parte con la que ha enlazado de este modo.

Esto significa que, cuando se utiliza un enlace SOAP de estilo de documento para una operación cuyos mensajes (de entrada, salida o error) se definen con varias partes, sólo se deberá enlazar una de esas partes al cuerpo de SOAP a fin de ajustarse a los estándares de WS-I Basic Profile 1.1.

El planteamiento siguiente se utiliza cuando se generan descripciones WSDL para exportaciones con enlaces de servicio Web (JAX-WS y JAX-RPC) en este caso:

- La primera parte de mensaje se enlaza al cuerpo SOAP.
- Para el enlace JAX-WS, todas las demás partes de mensaje de tipo "hexBinary" o "base64Binary" se enlazan como adjuntos referenciados. Consulte "Adjuntos referenciados: partes de mensaje de nivel superior" en la página 42.
- Todas las demás partes de mensaje se enlaza como cabeceras SOAP.

Los enlaces de importación JAX-RPC y JAX-WS se ajustarán a los estándares del enlace SOAP en un documento WSDL existente con mensajes de estilo de

documento de varias partes incluso si éste enlaza varias partes al cuerpo SOAP; sin embargo, no podrá generar clientes de servicio Web para dichos documentos WSDL en Rational Application Developer.

Nota: El enlace JAX-RPC no soporta adjuntos.

Por consiguiente el patrón recomendado cuando se utilizan mensajes de varias partes con una operación que tiene enlace SOAP de estilo de documento es:

1. Utilice el estilo incluido de documento/literal. En este caso, los mensajes siempre tienen una sola parte; sin embargo, los adjuntos pueden no tener referencia (como se describe en “Adjuntos no referenciados” en la página 47) o pueden ser de tipo swaRef (como se describe en “Adjuntos referenciados: elementos de tipo swaRef” en la página 38) en este caso.
2. Utilice el estilo de RPC/literal. En este caso, no hay restricciones en el enlace WSDL en términos de número de partes enlazadas al cuerpo SOAP; el mensaje SOAP que se produce tiene siempre un solo hijo que representa la operación que se está invocando, donde las partes de mensaje son hijos de ese elemento.
3. Para el enlace JAX-WS, haga que la primera parte de mensaje no sea de tipo "hexBinary" o "base64Binary" y todas las demás partes de uno de esos dos tipos se enlazarán a continuación como adjuntos.
4. Cualquier otro caso estará sujeto al comportamiento descrito más arriba.

Nota: Al recibir mensajes SOAP de estilo de documento de varias partes con adjuntos referenciados, el enlace JAX-WS espera que cada adjunto referenciado esté representado por un elemento hijo de cuerpo SOAP con un atributo href cuyo valor identifique la conexión por el ID de contenido. El enlace JAX-WS envía adjuntos referenciados para dichos mensajes del mismo modo. Este comportamiento no se ajusta a los estándares de WS-I Basic Profile. El perfil de adjuntos de WS-I define una "codificación de partes de id de contenido", que permite omitir el elemento hijo con el atributo href y, por consiguiente, hace que estos mensajes se ajusten a los estándares de Basic Profile. El enlace JAX-WS no soporta el envío o la recepción de mensajes que utilizan la codificación de partes de id de contenido. Para asegurar que los mensajes se ajusten a los estándares, siga el planteamiento 1 o 2 de la lista anterior o evite el uso de adjuntos referenciados para dichos mensajes y utilice en su lugar adjuntos no referenciados o de tipo swaRef.

Enlaces HTTP

El enlace HTTP está diseñado para proporcionar una conectividad SCA (Service Component Architecture) con HTTP. En consecuencia, las aplicaciones HTTP existentes o recién desarrolladas pueden participar en entornos de arquitectura orientada a servicios (SOA - Service Oriented Architecture).

HTTP (protocolo de transporte de hipertexto) es un protocolo ampliamente utilizado para transportar información en la Web. Cuando trabaje con una aplicación externa que utilice el protocolo HTTP, necesitará un enlace HTTP. El enlace HTTP maneja la transformación de los datos pasados, como un mensaje en un formato nativo, a un objeto de empresa de una aplicación SCA. El enlace HTTP también puede transformar los datos extraídos como un objeto empresarial, al formato nativo que espera la aplicación externa, para un mensaje de entrada.

Nota: Si desea interactuar con clientes y servicios que utilizan el protocolo SOAP/HTTP de servicios Web, tenga en cuenta la posibilidad de utilizar uno de

los enlaces de servicio Web, que proporcionan funcionalidad adicional respecto al manejo de calidades de servicio estándares de servicios Web.

En la lista siguiente se describen algunos escenarios comunes para utilizar el enlace HTTP:

- Los servicios alojados en SCA pueden invocar aplicaciones HTTP utilizando una importación HTTP.
- Los servicios alojados en SCA pueden exponerse a sí mismos como aplicaciones habilitadas para HTTP, de modo que los clientes HTTP pueden utilizarlos, mediante una exportación HTTP.
- WebSphere Process Server y WebSphere Enterprise Service Bus pueden comunicarse entre sí a través de una infraestructura HTTP y, por lo tanto, los usuarios pueden gestionar sus comunicaciones según los estándares corporativos.
- WebSphere Process Server y WebSphere Enterprise Service Bus pueden actuar como mediadores de comunicaciones HTTP, transformando y direccionando los mensajes, que mejora la integración de aplicaciones que utilicen una red HTTP.
- WebSphere Process Server y WebSphere Enterprise Service Bus se pueden utilizar para hacer de puente entre el protocolo HTTP y otros protocolos como, por ejemplo, SOAP/servicios Web HTTP, adaptadores de recursos basados en JCA (Java Connector Architecture), JMS, etc.

Puede encontrar información detallada sobre la creación de enlaces de importación y exportación HTTP en el Centro de información de WebSphere Integration Developer. Consulte los temas **Desarrollo de aplicaciones de integración → Acceso a servicios externos con HTTP**.

Visión general de enlaces HTTP

El enlace HTTP proporciona conectividad a las aplicaciones alojadas bajo un protocolo HTTP. Su función es arbitrar la comunicación entre las aplicaciones HTTP y permite que, desde un módulo, se pueda llamar a las aplicaciones alojadas bajo un protocolo HTTP.

Enlaces de importación HTTP

El enlace de importación HTTP proporciona conectividad de salida desde aplicaciones SCA (Service Component Architecture) a un servidor o aplicaciones HTTP.

La importación invoca un URL de punto final HTTP. El URL se puede especificar de una de tres maneras:

- El URL se puede establecer dinámicamente en las cabeceras HTTP por medio del URL de alteración dinámico.
- El URL se puede establecer dinámicamente en el elemento de dirección de destino SMO.
- El URL se puede especificar como una propiedad de configuración en la importación.

Esta invocación, por naturaleza, siempre es síncrona.

Aunque las invocaciones HTTP son siempre del tipo petición-respuesta, la importación HTTP da soporte tanto a las operaciones unidireccionales como bidireccionales, y no tiene en cuenta la respuesta en el caso de que se trate de una operación unidireccional.

Enlaces de exportación HTTP

El enlace de exportación HTTP proporciona conectividad de entrada de aplicaciones HTTP a una aplicación SCA.

Un URL se define en la exportación HTTP. Las aplicaciones HTTP que quieren enviar mensajes de petición a la exportación, utilizan este URL para invocar la exportación.

La exportación HTTP también da soporte a los mandatos ping.

Enlaces HTTP en el tiempo de ejecución

Una importación con un enlace HTTP en tiempo de ejecución envía una petición, con o sin datos, en el cuerpo del mensaje desde la aplicación SCA al servicio Web externo. La petición se efectúa desde la aplicación SCA al servicio Web externo, tal como se muestra en la Figura 27.

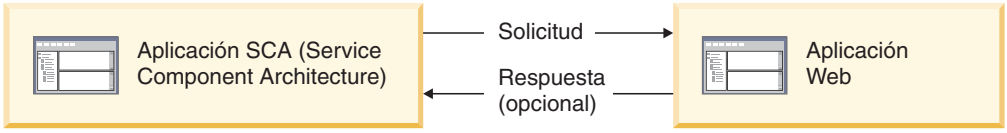


Figura 27. Flujo de una petición de la aplicación SCA a la aplicación Web.

Opcionalmente, la importación con el enlace HTTP puede recibir datos procedentes de la aplicación Web, en respuesta a la petición.

Con una exportación, una aplicación cliente efectúa la petición a un servicio Web, tal como se muestra en la Figura 28.

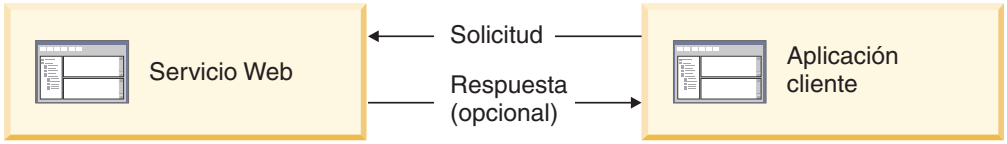


Figura 28. Flujo de una petición del servicio Web a la aplicación cliente.

El servicio Web es una aplicación Web que se ejecuta en el servidor. La exportación se implementa en dicha aplicación Web como un servlet, de forma que el cliente envía su petición a una dirección URL. El servlet pasa la petición a la aplicación SCA en el tiempo de ejecución.

Opcionalmente, la exportación puede enviar datos a la aplicación cliente en respuesta a la petición.

Cabeceras HTTP

Los enlaces de importación y exportación HTTP permiten la configuración de cabeceras HTTP y sus valores para ser utilizados para los mensajes de salida. La importación HTTP utiliza estas cabeceras para peticiones y la exportación HTTP las utiliza para las respuestas.

Las cabeceras configuradas estáticamente y la información de control tienen preferencia sobre los valores establecidos de forma dinámica durante la ejecución.

Sin embargo, los valores de control del URL de alteración dinámico, la versión y el método alteran temporalmente los valores estáticos, que, de lo contrario, se consideran los valores por omisión.

El enlace soporta la naturaleza dinámica del URL de importación HTTP determinando el valor del URL de destino HTTP, la versión y el método en el tiempo de ejecución. Estos valores se determinan mediante la extracción del valor de referencia de punto final, URL de alteración dinámico, versión y método.

- En el caso de Referencia de punto final, utilice las API com.ibm.websphere.sca.addressing.EndpointReference o establezca el campo /headers/SMOHeader/Target/address en la cabecera SMO.
- Para el URL de alteración dinámico, la versión y el método, utilice la sección de parámetros de control HTTP del mensaje SCA (Service Component Architecture). Tenga en cuenta que el URL de alteración dinámico tiene prioridad sobre la referencia de punto final de destino; sin embargo, dado que la referencia de punto final se aplica a través de los enlaces es el planteamiento preferido y se deberá utilizar donde sea posible.

Nota: Consulte Conceptos relacionados para obtener información sobre invocación dinámica y para obtener información específica sobre el formato, la sintaxis y la utilización de URL.

La información de control y cabecera para los mensajes de salida bajo los enlaces de importación y exportación HTTP se procesa en el orden siguiente:

1. La información de cabecera y control excluyendo el URL de alteración dinámico de HTTP, la versión y el método del mensaje SCA (prioridad más baja)
2. Modificaciones desde la consola administrativa en el nivel de exportación/importación
3. Modificaciones desde la consola administrativa en el nivel de método de la exportación o la importación
4. Dirección de destino especificada por medio de la referencia de punto final o la cabecera SMO
5. URL de alteración dinámico, versión y método del mensaje SCA
6. La información de cabeceras y control del manejador de datos o del enlace de datos (prioridad mayor)

La exportación e importación HTTP sólo llenará los parámetros de control y las cabeceras de dirección de entrada con los datos procedentes del mensaje entrante (HTTPExportRequest y HTTPImportResponse) si la propagación de cabecera de protocolo se ha establecido en `True`. Y a la inversa, la exportación e importación HTTP sólo leerá y procesará las cabeceras de salida y los parámetros de control (HTTPExportResponse y HTTPImportRequest) si la propagación de cabecera de protocolo se ha establecido en `True`.

Nota: Los cambios del manejador de datos o del enlace de datos efectuados en las cabeceras, o los parámetros de control, de la respuesta de importación o petición de exportación, no alterarán las instrucciones de proceso del mensaje dentro del enlace de importación o exportación, y deben utilizarse sólo para propagar los valores modificados a los componentes SCA en sentido descendente.

El servicio de contexto es responsable de propagar el contexto (incluidas las cabeceras de protocolo, por ejemplo la cabecera HTTP, y el contexto de usuario, por ejemplo el ID de cuenta) en una vía de acceso de invocación SCA. Durante el desarrollo en WebSphere Integration Developer, puede controlar la propagación de

contexto mediante las propiedades de importación y exportación. Para obtener más detalles, consulte la información de enlaces de importación y exportación en el Centro de información de WebSphere Integration Developer.

Estructuras de cabecera HTTP proporcionadas y soporte

Tabla 12 detalla los parámetros de petición/respuesta para las peticiones y respuestas de importación HTTP y exportación HTTP.

Tabla 12. Información de cabecera HTTP proporcionada

Nombre de control	**Petición de importación HTTP**	**Respuesta de importación HTTP**	**Petición de exportación HTTP**	**Respuesta de exportación HTTP**
URL	Se omite	No establecida	Se lee en el mensaje de petición **Nota:** La serie de consulta también forma parte del parámetro de control URL.	Se omite
Versión (valores posibles: 1.0, 1.1; el valor por omisión es 1.1)	Se omite	No establecida	Se lee en el mensaje de petición	Se omite
Método	Se omite	No establecida	Se lee en el mensaje de petición	Se omite
URL de alteración dinámico	Si se establece en el manejador de datos o en el enlace de datos, altera temporalmente el URL de importación HTTP. Se escribe en el mensaje en la línea de petición. **Nota:** La serie de consulta también forma parte del parámetro de control URL.	No establecida	No establecida	Se omite
Versión de alteración dinámica	Si se establece, altera temporalmente la versión de importación HTTP. Se escribe en el mensaje en la línea de petición.	No establecida	No establecida	Se omite

Tabla 12. Información de cabecera HTTP proporcionada (continuación)

Nombre de control	Petición de importación HTTP	Respuesta de importación HTTP	Petición de exportación HTTP	Respuesta de exportación HTTP
Método de alteración dinámico	Si se establece, altera temporalmente el método de importación HTTP. Se escribe en el mensaje en la línea de petición.	No establecida	No establecida	Se omite
Tipo de soporte (este parámetro de control lleva parte del valor de la cabecera HTTP Content-Type).	Si está presente, se escribe en el mensaje como parte de la cabecera Content-Type. **Nota:** Este valor del elemento de control debe ser proporcionado por el manejador de datos o el enlace de datos.	Se lee en el mensaje de respuesta, cabecera Content-Type	Se lee en el mensaje de petición, cabecera Content-Type	Si está presente, se escribe en el mensaje como parte de la cabecera Content-Type. **Nota:** Este valor del elemento de control debe ser proporcionado por el manejador de datos o el enlace de datos.
Conjunto de caracteres (el valor por omisión es UTF-8)	Si está presente, se escribe en el mensaje como parte de la cabecera Content-Type. **Nota:** Este valor del elemento de control debe ser proporcionado por el enlace de datos.	Se lee en el mensaje de respuesta, cabecera Content-Type	Se lee en el mensaje de petición, cabecera Content-Type	Soportada; se escribe en el mensaje como parte de la cabecera Content-Type. **Nota:** Este valor del elemento de control debe ser proporcionado por el enlace de datos.
Codificación de transferencia (valores posibles: chunked, identity; el valor por omisión es identity)	Si está presente, se escribe en el mensaje como parte de una cabecera y controla cómo se codifica la transformación del mensaje.	Se lee en el mensaje de respuesta	Se lee en el mensaje de petición	Si está presente, se escribe en el mensaje como parte de una cabecera y controla cómo se codifica la transformación del mensaje.
Codificación de contenido (valores posibles: gzip, x-gzip, deflate, identity; el valor por omisión es identity)	Si está presente, se escribe en el mensaje como una cabecera y controla cómo se codifica la carga útil.	Se lee en el mensaje de respuesta	Se lee en el mensaje de petición	Si está presente, se escribe en el mensaje como una cabecera y controla cómo se codifica la carga útil.

Tabla 12. Información de cabecera HTTP proporcionada (continuación)

Nombre de control	**Petición de importación HTTP**	**Respuesta de importación HTTP**	**Petición de exportación HTTP**	**Respuesta de exportación HTTP**
Content-Length	Se omite	Se lee en el mensaje de respuesta	Se lee en el mensaje de petición	Se omite
StatusCode (valor por omisión: 200)	No soportado	Se lee en el mensaje de respuesta	No soportado	Si está presente, se escribe en el mensaje en la línea de respuesta
ReasonPhrase (valor por omisión: OK)	No soportado	Se lee en el mensaje de respuesta	No soportado	Se ignora el valor de control. El valor de la línea de respuesta del mensaje se genera a partir de StatusCode.
Autenticación (contiene varias propiedades)	Si está presente, se utiliza para construir la cabecera de autenticación básica. **Nota:** El valor de esta cabecera se codificará sólo en el protocolo HTTP. En la SCA, se descodificará y se pasará como texto legible.	No se aplica	Se lee en la cabecera de autenticación básica del mensaje de petición. La presencia de esta cabecera no indica que el usuario se haya autenticado. La autenticación se debe controlar en la configuración del servlet. **Nota:** El valor de esta cabecera se codificará sólo en el protocolo HTTP. En la SCA, se descodificará y se pasará como texto legible.	No se aplica
Proxy (contiene varias propiedades: Host, Port, Authentication)	Si está presente, se utiliza para establecer una conexión a través del proxy.	No se aplica	No se aplica	No se aplica

Tabla 12. Información de cabecera HTTP proporcionada (continuación)

Nombre de control	Petición de importación HTTP	Respuesta de importación HTTP	Petición de exportación HTTP	Respuesta de exportación HTTP
SSL (contiene varias propiedades: Keystore, Keystore Password, Trustore, Trustore Password, ClientAuth)	Si se ha rellenado y el URL de destino es HTTPS, se utiliza para establecer una conexión a través de SSL.	No se aplica	No se aplica	No se aplica

Enlaces de datos HTTP

Para cada correlación diferente de datos entre un mensaje de la arquitectura SCA (Arquitectura de componente de servicio) y un mensaje del protocolo HTTP, debe configurarse un manejador de datos o un enlace de datos HTTP. Los manejadores de datos proporcionan una interfaz de enlaces neutros que permite volver a utilizar los enlaces de transporte y representar el enfoque recomendado. Los enlaces de datos son específicos de un enlace de transporte concreto. Se proporcionan clases de enlace de datos específicos de HTTP; también puede escribir manejadores de datos personalizados o enlaces de datos.

Nota: Las tres clases de enlace de datos HTTP descritas en este tema (HTTPStreamDataBindingSOAP, HTTPStreamDataBindingXML y HTTPServiceGatewayDataBinding) están en desuso como parte de WebSphere Process Server Versión 7.0. En lugar de utilizar los enlaces de datos descritos en este tema, considere los siguientes manejadores de datos:

- Utilice SOAPDataHandler en lugar de HTTPStreamDataBindingSOAP.
- Utilice UTF8XMLDataHandler en lugar de HTTPStreamDataBindingXML
- Utilice GatewayTextDataHandler en lugar de HTTPServiceGatewayDataBinding

Los enlaces de datos se proporcionan para ser utilizados con importaciones HTTP y exportaciones HTTP: enlace de datos binarios, enlace de datos XML y enlace de datos SOAP. En las operaciones unidireccionales no se necesitan enlaces de datos de respuesta. Un enlace de datos se representa mediante el nombre de una clase Java cuyas instancias pueden convertirse de HTTP a ServiceDataObject y viceversa. Se debe utilizar un selector de función en una exportación que, junto con los enlaces de método, puede determina qué enlace de datos se utiliza y qué operación se invoca. Los enlaces de datos proporcionados son:

- Enlaces de datos binarios, los cuales tratan el cuerpo como datos binarios sin estructura. La implementación del esquema XSD del enlace de datos binarios es la siguiente:

```
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:tns="http://com.ibm.websphere.http.data.bindings/schema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="HTTPBaseBody">
    <xsd:sequence/>
  </xsd:complexType>

  <xsd:complexType name="HTTPBytesBody">
    <xsd:complexContent>
```

```
      <xsd:extension base="tns:HTTPBaseBody">
        <xsd:sequence>
          <xsd:element name="value" type="xsd:hexBinary"/>
        </xsd:sequence>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>
```

- Enlaces de datos XML, los cuales soportan el cuerpo como datos XML. La implementación del enlace de datos XML es similar al enlace de datos JMS XML y no tiene limitaciones en el esquema de la interfaz.
- Enlaces de datos SOAP, los cuales soportan el cuerpo como datos SOAP. La implementación del enlace de datos SOAP no tiene limitaciones en el esquema de la interfaz.

Implementación de enlaces de datos HTTP personalizado

Esta sección describe cómo implementar un enlace de datos HTTP personalizado.

Nota: El enfoque recomendado es la implementación de un manejador de datos personalizado, ya que puede volver a utilizarse con enlaces de transporte.

HTTPStreamDataBinding es la interfaz principal para manejar los mensajes HTTP personalizados. La interfaz se ha diseñado para permitir el manejo de grandes cargas útiles. Sin embargo, para que dicha implementación funcione, este enlace de datos debe devolver la información de control y las cabeceras antes de escribir el mensaje en la secuencia.

Los métodos y su orden de ejecución, listado a continuación, deben ser implementados por el enlace de datos personalizado.

Para personalizar un enlace de datos, escriba una clase que implemente HTTPStreamDataBinding. El enlace de datos debería tener cuatro propiedades privadas:

- private DataObject pDataObject
- private HTTPControl pCtrl
- private HTTPHeaders pHeaders
- private yourNativeDataType nativeData

El enlace HTTP invoca el enlace de datos personalizado en el orden siguiente:

- Proceso saliente (DataObject a formato nativo):
  1. setDataObject(...)
  2. setHeaders(...)
  3. setControlParameters(...)
  4. setBusinessException(...)
  5. convertToNativeData()
  6. getControlParameters()
  7. getHeaders()
  8. write(...)
- Proceso entrante (formato nativo a DataObject):
  1. setControlParameters(...)
  2. setHeaders(...)
  3. convertFromNativeData(...)

4. isBusinessException()
5. getDataObject()
6. getControlParameters()
7. getHeaders()

Debe invocar setDataObject(...) en convertFromNativeData(...) para establecer el valor de dataObject, el cual pasa de ser dato nativo a propiedad privada de "pDataObject".

```
public void setDataObject(DataObject dataObject)
   throws DataBindingException {
  pDataObject = dataObject;

}
public void setControlParameters(HTTPControl arg0) {
  this.pCtrl = arg0;
}

public void setHeaders(HTTPHeaders arg0) {
  this.pHeaders = arg0;
}
/*
* Add http header "IsBusinessException" in pHeaders.
* Two steps:
* 1.Remove all the header with name IsBusinessException (case-insensitive) first.
*   This is to make sure only one header is present.
* 2.Add the new header "IsBusinessException"
*/
public void setBusinessException(boolean isBusinessException) {
  //remove all the header with name IsBusinessException (case-insensitive) first.
  //This is to make sure only one header is present.
  //add the new header "IsBusinessException", code example:
  HTTPHeader header=HeadersFactory.eINSTANCE.createHTTPHeader();
  header.setName("IsBusinessException");
  header.setValue(Boolean.toString(isBusinessException));
  this.pHeaders.getHeader().add(header);

}
public HTTPControl getControlParameters() {
  return pCtrl;
}
public HTTPHeaders getHeaders() {
  return pHeaders;
}
public DataObject getDataObject() throws DataBindingException {
  return pDataObject;
}
/*
* Get header "IsBusinessException" from pHeaders, return its boolean value
*/
public boolean isBusinessException() {
  String headerValue = getHeaderValue(pHeaders,"IsBusinessException");
  boolean result=Boolean.parseBoolean(headerValue);
  return result;
}
public void convertToNativeData() throws DataBindingException {
  DataObject dataObject = getDataObject();
  this.nativeData=realConvertWorkFromSDOToNativeData(dataObject);
}
public void convertFromNativeData(HTTPInputStream arg0){
  //Customer-developed method to
  //Read data from HTTPInputStream
  //Convert it to DataObject
  DataObject dataobject=realConvertWorkFromNativeDataToSDO(arg0);
  setDataObject(dataobject);
```

```
}
public void write(HTTPOutputStream output) throws IOException {
  if (nativeData != null)
  output.write(nativeData);
}
```

Enlaces EJB

Los enlaces de importación EJB (Enterprise JavaBeans) permiten a los componentes SCA (Service Component Architecture) invocar servicios proporcionados por la lógica empresarial Java EE que se ejecuta en un servidor Java EE. Los enlaces de exportación EJB permite exponer a los componentes SCA como Enterprise JavaBeans, de forma que la lógica empresarial Java EE puede invocar componentes SCA que, de lo contrario, no están disponibles.

Enlaces de importación EJB

Los enlaces de importación EJB permiten a un módulo SCA llamar a implementaciones EJB especificando la forma a través de la que se enlaza el módulo que consume con el EJB externo. La importación de servicios de una implementación EJB externa permite a los usuarios conectar su lógica empresarial en el entorno de WebSphere Process Server y participar en un proceso de negocio.

Utilice WebSphere Integration Developer para crear enlaces de importación EJB. Puede utilizar cualquiera de estos procedimientos para generar los enlaces:

- Crear la importación EJB utilizando el asistente de servicios externos

  Puede utilizar el asistente de servicios externos en WebSphere Integration Developer para crear una importación EJB basada en una implementación existente. El asistente de servicio externo crea servicios basados en los criterios que proporcione. Genera objetos de negocio, interfaces y archivos de importación basándose en los servicios descubiertos.
- Crear una importación EJB utilizando el editor de ensamblajes.

  Puede crear una importación EJB dentro de un diagrama de ensamblaje utilizando el editor de ensamblajes de WebSphere Integration Developer. Desde la paleta, puede utilizar una importación o utilizar una clase Java para crear el enlace EJB.

La importación generada tiene enlaces de datos que realizan la conexión Java-WSDL, en lugar de necesitar un componente de puente Java. Puede conectar directamente un componente con una referencia de lenguaje de descripción de servicios web (WSDL) a la importación EJB que se comunica con un servicio basado en EJB utilizando una interfaz Java.

La importación EJB puede interactuar con la lógica empresarial Java EE utilizando el modelo de programación EJB 2.1 o el modelo de programación EJB 3.0.

La invocación de la lógica empresarial Java EE puede ser local (sólo para EJB 3.0) o remota.

- La invocación local se utiliza cuando se desea llamar a la lógica empresarial Java EE que reside en el mismo servidor que la importación.

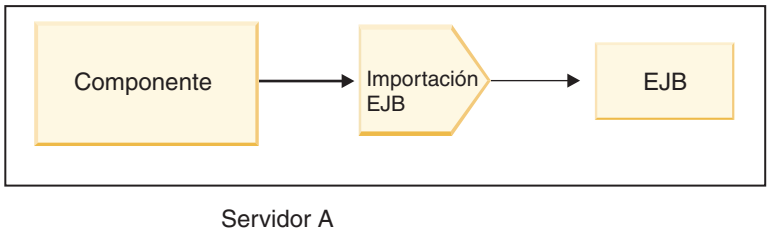


Figura 29. Invocación local de un EJB (sólo EJB 3.0)

- La invocación remota se utiliza cuando se desea llamar a la lógica empresarial Java EE que no reside en el mismo servidor que la importación.

  Por ejemplo, en la siguiente figura, una importación EJB utiliza la invocación de método remoto sobre protocolo Internet InterORB (RMI/IIOP) para invocar un método EJB en otro servidor.

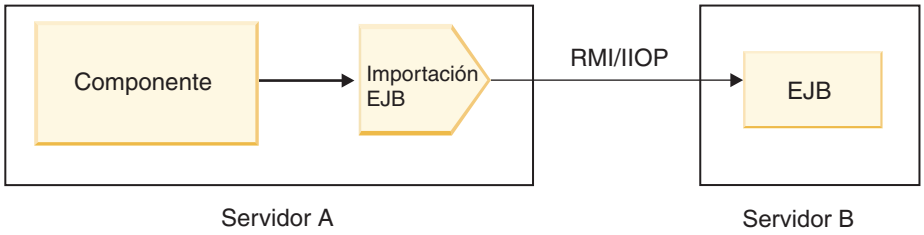


Figura 30. Invocación remota de un EJB

Cuando configura el enlace EJB, WebSphere Integration Developer utiliza el nombre JNDI para determinar el nivel del modelo de programación EJB y el tipo de invocación (local o remota).

Los enlaces de importación EJB contienen los siguientes componentes principales:

- Manejador de datos JAX-WS
- Selector de error de EJB
- Selector de función de importación EJB

Si el escenario del usuario no se basa en la correlación JAX-WS, es posible que necesite un manejador de datos personalizados, un selector de función y un selector de errores para realizar las tareas que, de lo contrario, completarían los componentes que forman parte de los enlaces de importación EJB. Esto incluye la correlación realizada normalmente por el algoritmo de correlación personalizado.

Enlaces de exportación EJB

Las aplicaciones Java EE externas pueden invocar un componente SCA a través de un enlace de exportación EJB. La utilización de una exportación EJB le permite exponer componentes SCA de forma que las aplicaciones Java EE externas pueden invocar estos componentes utilizando el modelo de programación EJB.

Nota: La exportación EJB es un bean sin estado.

Puede utilizar WebSphere Integration Developer para crear enlaces EJB. Puede utilizar cualquiera de estos procedimientos para generar los enlaces:

- Crear enlaces de exportación EJB utilizando el asistente de servicio externo

  Puede utilizar el asistente de servicio externo en WebSphere Integration Developer para crear un servicio de exportación EJB basado en una

implementación existente. El asistente de servicio externo crea servicios basados en los criterios que proporcione. Genera objetos de negocio, interfaces y archivos de exportación basándose en los servicios descubiertos.

- Crear enlaces de exportación EJB utilizando el editor de ensamblajes

  Puede crear una exportación EJB utilizando el editor de ensamblajes WebSphere Integration Developer.

Puede generar el enlace a partir de un componente SCA existente, o puede generar una exportación con un enlace EJB para una interfaz Java.

- Cuando genere una exportación para un componente SCA existente que tiene una interfaz WSDL existente, se asigna una interfaz Java a la exportación.
- Cuando genere una exportación para una interfaz Java, puede seleccionar una interfaz WSDL o Java para la exportación.

  Nota: Una interfaz Java utilizada para crear una exportación EJB tiene las siguientes limitaciones respecto a los objetos (parámetros y excepciones de entrada y salida) pasadas como parámetros en una llamada remota:
  - Deben ser del tipo concreto (en lugar de un tipo de interfaz o abstracto).
  - Deben ser compatibles con la especificación de Enterprise JavaBean. Se deben serializar y tener el constructor sin argumentos predeterminado, y se debe poder acceder a todas las propiedades a través de los métodos getter y setter.

    Consulte el sitio web de Sun Microsystems, Inc. en http://java.sun.com si desea información sobre la especificación de Enterprise JavaBean.

  Además, la excepción debe ser una excepción comprobada, heredada de java.lang.Exception y debe ser singular (es decir, no soporta el lanzamiento de varios tipos de excepción comprobada).

  También debe tener en cuenta que la interfaz empresarial de un EnterpriseBean Java es una interfaz Java sencilla y no debe ampliar javax.ejb.EJBObject o javax.ejb.EJBLocalObject. Los métodos de la interfaz empresarial no deben lanzar la excepción java.rmi.Remote.Exception.

Los enlaces de exportación EJB pueden interactuar con la lógica empresarial Java EE utilizando el modelo de programación EJB 2.1 o el modelo de programación EJB 3.0.

La invocación puede ser local (sólo para EJB 3.0) o remota.

- La invocación local se utiliza cuando la lógica empresarial Java EE llama a un componente SCA que reside en el mismo servidor que la exportación.
- La invocación remota se utiliza cuando la lógica empresarial Java EE no reside en el mismo servidor que la exportación.

  Por ejemplo, en la siguiente figura, un EJB utiliza RMI/IIOP para llamar a un componente SCA en un servidor diferente.

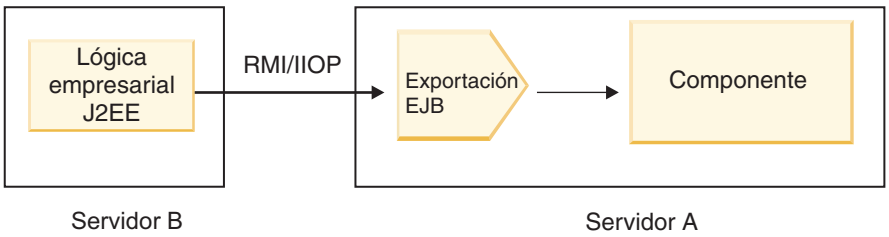


Figura 31. Llamada remota de un cliente a un componente SCA a través de una exportación EJB

Cuando configura el enlace EJB, WebSphere Integration Developer utiliza el nombre JNDI para determinar el nivel del modelo de programación EJB y el tipo de invocación (local o remota).

Los enlaces de exportación EJB contienen los siguientes componentes principales:

- Manejador de datos JAX-WS
- Selector de función de exportación EJB

Si el escenario de usuario no se basa en la correlación JAX-WS, es posible que necesite un manejador de datos personalizados y un selector de función para realizar las tareas que, de lo contrario, se completarían con componentes que forman parte de los enlaces de exportación EJB. Esto incluye la correlación realizada normalmente por el algoritmo de correlación personalizado.

Propiedades de enlace EJB

Los enlaces de importación EJB utilizan los nombres JNDI configurados para determinar el nivel de modelo de programación EJB y el tipo de invocación (local o remota). Los enlaces de importación y exportación EJB utilizan el manejador de datos JAX-WS para la transformación de datos. El enlace de importación EJB utiliza un selector de función de importación EJB y un selector de errores EJB y el enlace de exportación EJB utiliza un selector de función de exportación EJB.

Nombres JNDI y enlaces de importación EJB:

Cuando configura el enlace EJB en una importación, WebSphere Integration Developer utiliza el nombre JNDI para determinar el nivel de modelo de programación EJB y el tipo de invocación (local o remota).

Si no se ha especificado ningún nombre JNDI, se utiliza el enlace de interfaz EJB predeterminado. Los nombres predeterminados que se han creado dependen de si se invoca un JavaBean de EJB 2.1 o un JavaBean de EJB 3.0.

Nota: Consulte el centro de información de WebSphere Application Server si desea información más detallada sobre los convenios de denominación: Visión general de enlaces de aplicación EJB 3.0.

- JavaBean EJB 2.1

  El nombre JNDI predeterminado seleccionado previamente por WebSphere Integration Developer es el enlace de EJB 2.1 predeterminado, que adopta el formato `ejb/` más la interfaz de inicio, separados por barras inclinadas.

  Por ejemplo, para la interfaz inicial de un JavaBean EJB 2.1 para com.mycompany.myremotebusinesshome, el enlace predeterminado es:

  ```
  ejb/com/mycompany/myremotebusinesshome
  ```

  Para EJB 2.1, sólo está soportada la invocación EJB remota.

- JavaBean EJB 3.0

  El nombre JNDI predeterminado seleccionado previamente por WebSphere Integration Developer para el JNDI local en el nombre de clase plenamente cualificado de la interfaz local precedido por `ejblocal:`. Por ejemplo, para la interfaz plenamente cualificada de la interfaz local com.mycompany.mylocalbusiness, la JNDI EJB 3.0 preseleccionada es:

  ```
  ejblocal:com.mycompany.mylocalbusiness
  ```

  Para la interfaz remota com.mycompany.myremotebusiness, la JNDI EJB 3.0 preseleccionada es la interfaz plenamente cualificada:

  ```
  com.mycompany.myremotebusiness
  ```

  Los enlaces de aplicación predeterminados de EJB 3.0 se describen en la siguiente ubicación: Visión general de enlaces de aplicación EJB 3.0.

  WebSphere Integration Developer utilizará el nombre "corto" como la ubicación de JNDI predeterminada para los EJB que utilizan el modelo de programación de versión 3.0.

  Nota: Si la referencia JNDI desplegada del EJB de destino es diferente de la ubicación de enlace JNDI predeterminado porque se ha utilizado o configurado una correlación personalizada, el nombre JNDI de destino se debe especificar correctamente. Puede especificar el nombre WebSphere Integration Developer antes del despliegue, o , para el enlace de importación, puede cambiar el nombre en la consola administrativa (después del despliegue) para coincidir con el nombre JNDI del EJB de destino.

Si desea más información sobre cómo crear enlaces EJB, consulte la sección dedicada a Trabajar con enlaces EJB en el centro de información de WebSphere Integration Developer.

Manejador de datos JAX-WS:

El enlace de importación EJB (Enterprise JavaBeans) utiliza el manejador de datos JAX-WS para convertir los objetos de negocio de petición en parámetros de objeto Java y para convertir el valor de retorno de objeto Java en el objeto de negocio de respuesta. El enlace de exportación EJB utiliza el manejador de datos JAX-WS para convertir los EJB de petición en objetos de negocio de petición y para convertir el objeto de negocio de respuesta en un valor de retorno.

Este manejador de datos correlaciona los datos de la interfaz WSDL especificada por SCA con la interfaz Java de EJB de destino (y viceversa) utilizando la especificación de la API de Java para los servicios web de XML (JAX-WS) y la especificación de arquitectura Java Java para enlaces XML (JAXB).

Nota: El soporte actual está limitado a las especificaciones JAX-WS 2.1.1 y JAXB 2.1.3.

El manejador de datos especificado en el nivel de enlace EJB se utiliza para realizar el proceso de peticiones, errores y excepciones de tiempo de ejecución.

Nota: Para los errores, se puede establecer un manejador de datos determinado para cada error, especificando la propiedad de configuración aultBindingType. Esto altera temporalmente el valor especificado en el nivel del enlace EJB.

El manejador de datos JAX-WS se utiliza, por omisión, cuando el enlace EJB tiene una interfaz WSDL. Este manejador de datos no puede utilizarse para transformar un mensaje SOAP que representa una invocación JAX-WS de un objeto de datos.

El enlace de importación EJB utiliza un manejador de datos para transformar un objeto de datos en una matriz de objetos Java (Object[]). Durante las comunicaciones de salida, tiene lugar el siguiente proceso:

1. El enlace EJB establece el tipo esperado, el elemento esperado y el nombre de método de destino en el BindingContext para que coincidan con los especificados en el WSDL.
2. El enlace EJB invoca el método de transformación del objeto de datos que requiere la transformación de datos.
3. El manejador de datos devuelve un Object[] que representa los parámetros del método (en el orden de su definición en el método).
4. El enlace EJB utiliza el Object[] para invocar el método en la interfaz EJB de destino.

El enlace también prepara un Object[] para procesar la respuesta de la invocación EJB.

- El primer elemento del Object[] es el valor de retorno de la invocación de método Java.
- Los valores siguientes representan los parámetros de entrada para el método.

Esto es necesario para soportar los tipos de entrada/salida y salida de parámetros.

Para los parámetros del tipo salida, los valores se deben devolver en el objeto de datos de respuesta.

El manejador de datos procesa y transforma los valores encontrados en el Object[] y, a continuación, devuelve una respuesta al objeto de datos.

El manejador de datos da soporte a xs:AnyType, xs:AnySimpleType y xs:Any, junto con otros tipos de datos XSD. Para habilitar el soporte para xs:Any, utilice `@XmlAnyElement (lax=true)` para la propiedad JavaBean en el código Java, tal como se muestra en el siguiente ejemplo:

```
public class TestType {
 private Object[] object;

 @XmlAnyElement (lax=true)
 public Object[] getObject() {
  return object;
 )

 public void setObject (Object[] object) {
  this.object=object;
 )
)
```

Esto convierte el objeto de propiedad en TestType en un campo xs:any. El valor de la clase Java utilizado en el campo xs:any debe tener la anotación `@XmlAnyElement`. Por ejemplo, si Dirección es la clase Java que se utiliza para rellenar la matriz de objetos, la clase Dirección debe tener la anotación `@XmlRootElement`.

Nota: Para personalizar la correlación del tipo XSD con los tipos Java definidos por la especificación JAX-WS, cambie las anotaciones JAXB para que se adapten a sus necesidades empresariales. El manejador de datos JAX-WS soporta xs:any, xs:anyType y xs:anySimpleType.

Se aplican las siguientes restricciones para el manejador de datos JAX-WS:

- El manejador de datos no incluye el soporte para la anotación `@WebParam` del atributo de cabecera.
- El espacio de nombres de los archivos de esquema de objetos de negocio (archivos XSD) no incluye la correlación por omisión del nombre de paquete Java. La anotación `@XMLSchema` en package-info.java tampoco funciona. La única forma de crear un XSD con un espacio de nombres es utilizar las anotaciones `@XmlType` y `@XmlRootElement`. `@XmlRootElement` define el espacio de nombres de destino de destino para el elemento global en los tipos JavaBean.
- El asistente de importación EJB no crea los archivos XSD para las clases no relacionadas. La versión 2.0 no soporta la anotación `@XmlSeeAlso`, así pues, si no se hace referencia directamente a la clase hijo desde la clase padre, no se crea un XSD. La solución a este problema es ejecutar SchemaGen para este tipo de clases hijo.

  SchemaGen es un programa de utilidad de línea de comandos de línea de mandatos (situado en el directorio *WPS_Install_Home*/bin) proporcionado para crear archivos XSD para un bean de Java determinado. Estos XSD se deben copiar manualmente en el módulo para que funciona la solución.

Selector de errores de EJB:

Se determina el selector de errores de EJB, si una invocación EJB ha generado un error, una excepción de tiempo de ejecución o una respuesta satisfactoria.

Si se detecta un error, el selector de errores de EJB devuelve el nombre del error nativo al tiempo de ejecución del enlace, de forma que el manejador de datos JAX-WS pueda convertir el objeto de excepción en un objeto de negocio con anomalía.

En una respuesta satisfactoria (sin errores), el enlace de importación EJB ensambla una matriz de objetos Java (Object[]) para devolver los valores.

- El primer elemento del Object[] es el valor de retorno de la invocación de método Java.
- Los valores siguientes representan los parámetros de entrada para el método.

Esto es necesario para soportar los tipos de entrada/salida y salida de parámetros.

En los escenarios de excepción, el enlace ensambla un Object[] y el primer elemento representa la excepción generada por el método.

El selector de errores puede devuelve los valores siguientes:

Tabla 13. Valores de retorno

Tipo	Valor de retorno	Descripción
Error	`ResponseType.FAULT`	Se devuelve cuando el Object[] pasado contiene un objeto de excepción.
Excepción de tiempo de ejecución	`ResponseType.RUNTIME`	Se devuelve si el objeto de excepción no coincide con ninguno de los tipos de excepción declarados en el método.
Respuesta normal	`ResponseType.RESPONSE`	Se devuelve en todos los demás casos.

Si el selector de errores devuelve un valor de `ResponseType.FAULT`, se devuelve el nombre de error nativo. El enlace utiliza este nombre de error nativo para determinar el nombre de anomalía WSDL correspondiente del modelo e invocar el manejador de datos de error correcto.

Selector de función EJB:

Los enlaces EJB utilizan un selector de función de importación (para el proceso de salida) o un selector de función de exportación (para el proceso de entrada) para determinar el método EJB para llamar.

Selector de función de importación

Para el proceso de salida, el selector de función de importación deriva el tipo de método EJB basado en el nombre de la operación invocada por el componente SCA que está conectado a la importación EJB. El selector de función consulta la anotación @WebMethod en la clase Java anotada JAX-WS generada por WebSphere Integration Developer para determinar el nombre de la operación de destino asociada.

- Si está presente la anotación @WebMethod, el selector de función utiliza la anotación @WebMethod para determinar la correlación correcta del método Java para el método WSDL.
- Si falta la anotación @WebMethod, el selector de función da por supuesto que el nombre del método Java es el mismo que el nombre de la operación invocada.

Nota: Este selector de función sólo es válido para una interfaz de tipo WSDL en una importación EJB, no para una interfaz de tipo Java en una importación EJB.

El selector de función devuelve un objeto java.lang.reflect.Method que representa el método de la interfaz EJB.

La selector de función utiliza una matriz de objetos Java (Object[]) para contener la respuesta del método de destino. El primer elemento del Object[] es un método Java con el nombre del WSDL, y el segundo elemento en el Object[] es el objeto de negocio de entrada.

Selector de función de exportación

Para el proceso de entrada, el selector de función de exportación deriva el método de destino que se debe invocar desde el método Java.

El selector de función de exportación correlaciona el nombre de la operación Java invocada por el cliente EJB con el nombre de la operación de la interfaz del componente de destino. El nombre de método se devuelve como una serie y lo resuelve el tiempo de ejecución de SCA, en función del tipo de interfaz del componente de destino.

Enlaces EIS

Los enlaces EIS (Sistema de información de empresa) proporcionan conectividad entre los componentes SCA y un EIS externo. Esta comunicación se consigue a través de las exportaciones e importaciones EIS que dan soporte a los adaptadores de recursos JCA 1.5 y a WebSphere Adapters.

Los componentes SCA pueden necesitar la transferencia de datos desde o hacia un EIS externo. Cuando se crea un módulo SCA que requiere este tipo de

conectividad, debe incluir (además del componente SCA) una importación o exportación con un enlace EIS para comunicarse con un EIS externo específico.

Los adaptadores de recursos de WebSphere Integration Developer se utilizan en el contexto de una importación o una exportación. Primero desarrolla una importación o una exportación con el asistente de servicio externo y, al desarrollarla, incluye el adaptador de recursos. Una importación EIS, que permite a la aplicación invocar un servicio en un sistema EIS, o una exportación EIS, que permite a una aplicación en un sistema EIS invocar un servicio desarrollado en WebSphere Integration Developer, se crean con un adaptador de recurso particular. Por ejemplo, crearía una importación con el adaptador JD Edwards para invocar un servicio en el sistema JD Edwards.

Cuando utilice el asistente del servicio externo, la información del enlace EIS se creará de forma automática. También puede utilizar otra herramienta, el editor de ensamblaje, para añadir o modificar la información del enlace. Consulte el Centro de información de WebSphere Integration Developer para obtener más información.

Tras desplegarse en el servidor el módulo SCA que contiene el enlace EIS, podrá utilizar la consola administrativa para ver información sobre el enlace, o para configurar éste último.

Visión general de los enlaces EIS

El enlace EIS (sistema de información de empresa), cuando se utiliza con un adaptador de recursos JCA, permite acceder a los servicios en un sistema de información de empresa o hacer que los servicios estén disponibles para el EIS.

En el ejemplo siguiente se muestra un módulo SCA típico con el nombre ContactSyncModule sincroniza la información de contacto entre un sistema Siebel y un sistema SAP.

1. El componente SCA denominado ContactSync escucha (mediante una exportación de aplicación EIS denominada Contacto Siebel) los cambios en los contactos Siebel.
2. El propio componente SCA ContactSync utiliza una aplicación SAP (mediante una importación de aplicación EIS) para actualizar la información de contacto SAP según corresponda.

Como las estructuras de datos utilizadas para almacenar contactos son distintas en los sistemas Siebel y SAP, el componente SCA ContactSync debe proporcionar una correlación.

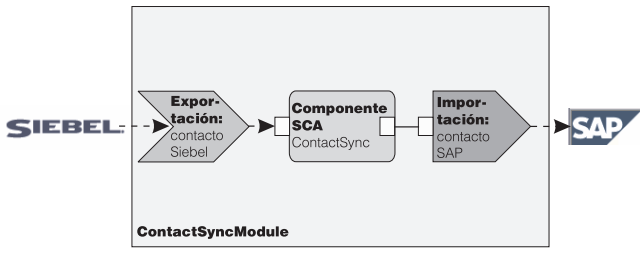


Figura 32. Flujo desde un sistema Siebel a uno SAP

La importación Contacto Siebel y la importación Contacto SAP tienen configurados los adaptadores de recursos adecuados.

Características clave de los enlaces EIS

Una importación EIS es una importación SCA (Service Component Architecture) que permite a los componentes del módulo de SCA utilizar los servicios EIS definidos fuera del módulo de SCA. Una importación EIS se utiliza para transferir datos del componente SCA a un EIS externo; una exportación EIS se utiliza para transferir datos del EIS externo a un componente SCA.

Importaciones

El rol de la importación EIS es hacer de puente entre los componentes SCA y los sistemas EIS externos. Las aplicaciones externas se pueden tratar como una importación EIS. En este caso, la importación EIS envía datos al EIS externo y, opcionalmente, recibe datos de respuesta.

La importación EIS proporciona a los componentes SCA una vista uniforme de las aplicaciones externas al módulo. Esto permite que los componentes se comuniquen con un EIS externo como, por ejemplo, SAP, Siebel o PeopleSoft, utilizando un modelo SCA coherente.

En el lado del cliente de la importación existe una interfaz, expuesta por la aplicación de importación EIS, con uno o más métodos, cada uno de los cuales toma objetos de datos como argumentos y valores de retorno. En el lado de la implementación, existe una CCI (Common Client Interface) implementada por un adaptador de recursos.

La implementación en tiempo de ejecución de una importación EIS conecta la interfaz del lado del cliente y la CCI. La importación correlaciona la invocación del método en la interfaz de la invocación en la CCI.

Los enlaces se crean a tres niveles: el enlace de interfaz, que utiliza los enlaces de método contenidos, que a su vez utilizan enlaces de datos.

El enlace de interfaz relaciona la interfaz de la importación con la conexión con el sistema EIS que proporciona la aplicación. Esto refleja el hecho de que la instancia específica del EIS proporciona el conjunto de aplicaciones, representado por la interfaz, y la conexión proporciona acceso a esta instancia. El elemento de enlace contiene propiedades con suficiente información para crear la conexión (estas propiedades forman parte de la instancia javax.resource.spi.ManagedConnectionFactory).

El enlace de método asocia el método con la interacción específica con el sistema EIS. Para JCA, la interacción está caracterizada por el conjunto de propiedades de la implementación de interfaz javax.resource.cci.InteractionSpec. El elemento de interacción del enlace de método contiene estas propiedades, junto con el nombre de la clase, lo que proporciona información suficiente para realizar la interacción. El enlace de método utiliza enlaces de datos que describen la correlación del argumento y el resultado del método de interfaz con la representación EIS.

El escenario de ejecución de una importación EIS es el siguiente:

1. El método en la interfaz de importación se invoca utilizando el modelo de programación SCA.
2. La petición, que alcanza la importación EIS, contiene el nombre del método y los argumentos.
3. La importación crea primero una implementación de enlace de interfaz y, a continuación, utilizando los datos del enlace de importación, una

ConnectionFactory, y asocia ambas. Es decir, la importación invoca setConnectionFactory en el enlace de interfaz.

4. Se crea la implementación de enlace de método que coincide con el método invocado.
5. La instancia javax.resource.cci.InteractionSpec se crea y se rellena y, a continuación, se utilizan enlaces de datos para enlazar los argumentos de método con un formato conocido para el adaptador de recursos.
6. La interfaz CCI se utiliza para realizar la interacción.
7. Cuando se devuelve la llamada, se utilizan los enlaces de datos para crear el resultado de la invocación y el resultado se devuelve al llamante.

Exportaciones

El rol de la exportación EIS es hacer de puente entre un componente SCA y un EIS externo. Las aplicaciones externas se pueden tratar como una exportación EIS. En este caso, la aplicación externa envía los datos en forma de notificaciones periódicas. Una exportación EIS se puede considerar una aplicación de suscripción que escucha una solicitud externa de un EIS. El componente SCA que utiliza la exportación EIS la ve como una aplicación local.

La exportación EIS proporciona a los componentes SCA una vista uniforme de las aplicaciones externas al módulo. Esto permite que los componentes se comuniquen con un EIS como, por ejemplo, SAP, Siebel o PeopleSoft, utilizando un modelo SCA coherente.

La exportación presenta una implementación de receptor que recibe solicitudes del EIS. El receptor implementa una interfaz receptora específica de adaptador de recursos. La exportación también contiene un componente que implementa la interfaz, expuesto al EIS a través de la exportación.

La implementación en tiempo de ejecución de una exportación EIS conecta el receptor con el componente que implementa la interfaz. La exportación correlaciona la solicitud EIS con la invocación de la operación correspondiente en el componente. Los enlaces se crean a tres niveles: un enlace de receptor, que utiliza el enlace de método nativo contenido, que a su vez utiliza un enlace de datos.

El enlace de receptor relaciona el receptor que recibe las solicitudes con el componente expuesto mediante la exportación. La definición de exportación contiene el nombre del componente; el tiempo de ejecución lo localiza y le envía las solicitudes.

El enlace de método nativo asocia el método nativo o el tipo de suceso recibido por el receptor con la operación implementada por el componente expuesto a través de la exportación. No existe ninguna relación entre el método invocado en el receptor y el tipo de suceso; todos los sucesos llegan a través de uno o varios métodos del receptor. El enlace de método nativo utiliza el selector de funciones definido en la exportación para extraer el nombre de método nativo de los datos de entrada y enlaces de datos para enlazar el formato de datos del EIS con un formato conocido para el componente.

El escenario en tiempo de ejecución de una exportación EIS es el siguiente:

1. La solicitud EIS activa la invocación del método en la implementación del receptor.

2. El receptor localiza e invoca la exportación y le pasa todos los argumentos de invocación.
3. La exportación crea la implementación del enlace de receptor.
4. La exportación crea una instancia del selector de funciones y lo establece en el enlace de receptor.
5. La exportación inicializa enlaces de método nativo y los añade al enlace de receptor. Para cada enlace de método nativo, también se inicializan los enlaces de datos.
6. La exportación invoca el enlace del receptor.
7. El enlace de receptor localiza los componentes exportados y utiliza el selector de funciones para recuperar el nombre de método nativo.
8. Este nombre se utiliza para localizar el enlace de método nativo que, a continuación, invoca el componente de destino.

El estilo de interacción del adaptador permite al enlace de exportación EIS invocar el componente de destino de forma asíncrona (valor por omisión) o síncrona.

Adaptadores de recurso

Primero desarrolla una importación o una exportación con el asistente de servicio externo y, al desarrollarla, incluye un adaptador de recursos. Los Adaptadores que incorpora WebSphere Integration Developer utilizados para acceder a sistemas CICS, IMS, JD Edwards, PeopleSoft, SAP y Siebel están pensados únicamente con fines de desarrollo y de pruebas. Esto significa que puede usarlos para desarrollar y probar sus aplicaciones.

Cuando despliegue su aplicación, necesitará Adaptadores en tiempo de ejecución con licencia para ejecutar su aplicación. Sin embargo, cuando cree el servicio, puede incluir el adaptador con su servicio. Es posible que la licencia de su adaptador le permita usar el adaptador incluido en tiempo de ejecución. Estos adaptadores se ajustan a los estándares de Java EE Connector Architecture (JCA 1.5). JCA es un estándar abierto utilizado en Java EE para la conectividad EIS. JCA incluye una infraestructura gestionada; es decir, el servidor de aplicaciones proporciona Calidad del servicio (QoS), lo que ofrece gestión de ciclo de vida y seguridad a las transacciones. También se ajustan a la especificación Enterprise Metadata Discovery con excepción del adaptador de recursos IBM CICS ECI y del conector IBM IMS para Java.

El asistente también admite los adaptadores de WebSphere Business Integration, un conjunto antiguo de adaptadores.

Recursos Java EE

El módulo EIS, un módulo de SCA que sigue el patrón del módulo EIS, puede desplegarse en la plataforma Java EE.

El despliegue del módulo EIS en la plataforma Java EE genera una aplicación, que está preparada para su ejecución, comprimida como un archivo EAR y desplegada en el servidor. Se crean todos los artefactos y recursos Java EE; se configura la aplicación y está preparada para ejecutarse.

Propiedades dinámicas de la especificación de conexión y la especificación de interacción JCA

El enlace EIS puede aceptar entrada para la InteractionSpec y la ConnectionSpec especificadas utilizando un objeto de datos hijo bien definido que acompaña a la

carga útil. Esto permite interacciones dinámicas de solicitud-respuesta con un adaptador de recursos a través de la InteractionSpec y la autenticación de componentes a través de la ConnectionSpec.

javax.cci.InteractionSpec transporta información sobre cómo se debe manejar la petición de interacción con el adaptador de recursos. También puede incluir información sobre cómo se ha conseguido la interacción después de la petición. Estas comunicaciones de dos direcciones a través de las interacciones se conocen también como *conversaciones*.

El enlace EIS espera que la carga útil que será el argumento en el adaptador de recursos contenga un objeto de datos hijo denominado `properties`. Este objeto de datos de propiedad contendrá pares de nombre/valor, donde el nombre de las propiedades de especificación de interacción tendrá un formato específico. Las normas del formato son las siguientes:

- Los nombres deben empezar con el prefijo `IS`, seguido del nombre de propiedad. Por ejemplo, una especificación de interacción con una propiedad JavaBeans llamada `InteractionId` especificará el nombre de propiedad como `ISInteractionId`.
- El par de nombre/valor representa el nombre y el valor del tipo simple de la propiedad de especificación de interacción.

En este ejemplo, una interfaz especifica que la entrada de una operación es un objeto de datos `Account`. Esta interfaz invoca un servicio de enlace de importación EIS con el objetivo de enviar y recibir una propiedad dinámica InteractionSpec denominada `workingSet` con el valor `xyz`.

El gráfico de empresa o los objetos empresariales del servidor contienen un objeto empresarial `properties` subyacente que permite el envío de datos específicos del protocolo con la carga útil. Este objeto empresarial `properties` está incorporado y no se debe especificar en el esquema XML cuando se construye un objeto empresarial. Sólo se debe crear y utilizar. Si tiene sus propios tipos de datos definidos basados en un esquema XML, deberá especificar un elemento `properties` que contenga los pares de nombre/valor esperados.

```
BOFactory dataFactory = (BOFactory) \
serviceManager.locateService("com/ibm/websphere/bo/BOFactory");
 //Envoltura para interfaces de estilo acomodado de literal de documento,
 //saltar a la carga útil si no es literal de documento
 DataObject docLitWrapper = dataFactory.createByElement /
 ("http://mytest/eis/Account", "AccountWrapper");
```

Cree la carga útil.

```
DataObject account = docLitWrapper.createDataObject(0);
  DataObject accountInfo = account.createDataObject("AccountInfo");
  //Llevar a cabo la configuración de la carga útil


  //Construir los datos de propiedades para la interacción dinámica

  DataObject properties = account.createDataObject("properties");
```

Para el nombre workingSet, establezca el valor esperado (`xyz`).

```
properties.setString("ISworkingSet", "xyz");


  //Invocar el servicio con argumento

  Service accountImport = (Service) \
```

```
serviceManager.locateService("AccountOutbound");
DataObject result = accountImport.invoke("createAccount", docLitWrapper);

//Obtener la propiedad devuelta
DataObject retProperties = result.getDataObject("properties");

String workingset = retProperties.getString("ISworkingSet");
```

Puede utilizar las propiedades ConnectionSpec en la autenticación de componentes dinámica. Se aplican las mismas normas que antes, excepto que el prefijo de nombre de la propiedad debe ser `CS` (en lugar de `IS`). Las propiedades ConnectionSpec no son de dos direcciones. El mismo objeto de datos `properties` puede contener propiedades IS y CS.

Para utilizar las propiedades ConnectionSpec, establezca la propiedad `resAuth` especificada en el enlace de importación en `Application`. Asimismo, asegúrese de que el adaptador de recursos dé soporte a la autorización de componentes. Consulte el capítulo 8 de J2EE Connector Architecture Specification para obtener más información.

Clientes externos con enlaces EIS

El servidor puede enviar mensajes o recibir mensajes de clientes externos utilizando enlaces EIS.

Un cliente externo, por ejemplo, un portal Web o un EIS, necesita enviar un mensaje a un módulo SCA en el servidor o necesita que lo invoque un componente desde el servidor.

El cliente invoca la importación EIS del mismo modo que invoca cualquier otra aplicación, utilizando la interfaz DII (Dynamic Invocation Interface) o la interfaz Java.

1. El cliente externo crea una instancia del ServiceManager y busca la importación EIS utilizando su nombre de referencia. El resultado de la búsqueda es una implementación de la interfaz de servicio.
2. El cliente crea un argumento de entrada, un objeto de datos genérico, que se crea dinámicamente utilizando el esquema de objeto de datos. Este paso se realiza utilizando la implementación de la interfaz DataFactory del objeto de datos de servicio.
3. El cliente externo invoca el EIS y obtiene los resultados necesarios.

De forma alternativa, el cliente puede invocar la importación EIS utilizando la interfaz Java.

1. El cliente crea una instancia del ServiceManager y busca la importación EIS utilizando su nombre de referencia. El resultado de la búsqueda es una interfaz Java de la importación del EIS.
2. El cliente crea un argumento de entrada y un objeto de datos escrito.
3. El cliente invoca el EIS y obtiene los resultados necesarios.

La interfaz de exportación EIS define la interfaz del componente SCA exportado que está disponible para los servicios EIS externos. Esta interfaz se puede considerar la interfaz que invocará una aplicación externa (por ejemplo, SAP o PeopleSoft) a través de la implementación del tiempo de ejecución del servicio de exportación EIS.

La exportación utiliza EISExportBinding para enlazar los servicios exportados a la aplicación EIS externa. Permite suscribirse a una aplicación contenida en el módulo

SCA para que escuche las solicitudes de servicio EIS. El enlace de exportación de EIS especifica la correlación entre la definición de sucesos de entrada, tal como la entiende el adaptador de recursos (utilizando las interfaces Java EE Connector Architecture) y la invocación de las operaciones SCA.

EISExportBinding requiere que los servicios EIS externos se basen en contratos de entrada JCA (Java EE Connector Architecture 1.5). EISExportBinding requiere que se especifique un manejador de datos o un enlace de datos a nivel de enlace o a nivel de método.

Enlaces JMS

Un proveedor de JMS (Java Message Service) permite la mensajería basada en la API y el modelo de programación JMS (Java Messaging Service). Proporciona fábricas de conexiones Java EE para crear conexiones para destinos JMS y para enviar y recibir mensajes.

Se proporcionan estos enlaces JMS:

- Enlace de proveedor SIB (Service Integration Bus) compatible con JMS JCA 1.5 (*enlace JMS*).
- Enlaces JMS genéricos no de JCA compatibles con JMS 1.1 (*enlace JMS genérico*).
- Enlace JMS de WebSphere MQ, que proporciona soporte de proveedor JMS para WebSphere MQ y permite la interoperatividad de la aplicación Java EE (*enlace JMS de WebSphere MQ*)

Los enlaces de exportación e importación JMS permiten que un módulo SCA (Service Component Architecture) realice llamadas y reciba mensajes de sistemas JMS externos.

También están soportados los enlaces de WebSphere MQ (enlace de *WebSphere MQ*) que permiten a los usuarios MQ nativos manejar los mensajes arbitrarios de mensajes de entrada y de salida (WebSphere MQ obligatorio).

Los enlaces de importación y exportación JMS proporcionan integración con las aplicaciones JMS que utilizan el proveedor JMS SIB basado en JCA 1.5, que forma parte de WebSphere Application Server. Los demás adaptadores de recursos JMS basados en JCA 1.5 no están soportados.

Visión general de enlaces JMS

Los enlaces JMS proporcionan conectividad entre el entorno SCA (Service Component Architecture) y los sistemas JMS.

Enlaces JMS

Los principales componentes de los enlaces de importación JMS y exportación JMS son:

- Adaptador de recursos: permite la conectividad gestionada y bidireccional entre un módulo SCA y sistemas JMS externos.
- Conexiones: encapsulan una conexión virtual entre un cliente y una aplicación de proveedor.
- Destinos: utilizados por un cliente para especificar el destino de los mensajes que produce o el origen de los mensajes que consume.
- Datos de autenticación: utilizados para proteger el acceso al enlace.

Enlaces de importación JMS

Los enlaces de importación JMS permiten importar una aplicación JMS externa para utilizarla dentro del módulo SCA. Los enlaces de importación JMS permiten a los componentes del módulo SCA comunicarse con los servicios proporcionados por las aplicaciones JMS externas.

Las conexiones del proveedor JMS asociado de los destinos JMS se crean mediante una fábrica de conexiones JMS. Utilice los objetos administrativos de la fábrica de conexiones para gestionar fábricas de conexiones de JMS para el proveedor de mensajería por omisión.

La interacción con los sistemas JMS externos incluye la utilización de destinos para enviar peticiones y recibir respuestas.

Se da soporte a dos tipos de escenarios de uso para los enlaces de importación JMS, dependiendo del tipo de operación que se invoque:

- Unidireccional: la importación JMS coloca un mensaje en el destino de envío configurado en el enlace de importación. No se envía nada al campo replyTo de la cabecera JMS.
- Bidireccional (petición y respuesta): la importación JMS coloca un mensaje en el destino de envío y persiste la respuesta que recibe del componente SCA.

  El enlace de importación se puede configurar (utilizando el campo **Esquema de correlación de respuesta** en WebSphere Integration Developer) para esperar el ID de correlación de mensajes de respuesta que se ha copiado del ID de mensaje de petición (el predeterminado), o desde el ID de correlación de mensajes de petición. El enlace de importación también se puede configurar para utilizar un destino de respuesta dinámico temporal para correlacionar respuestas con peticiones. Se crea un destino temporal para cada petición y la importación utiliza este destino para recibir la respuesta.

  El destino indicado en receive se establece en la propiedad de la cabecera replyTo del mensaje de salida. Se despliega un receptor de mensajes para escuchar en el destino de recepción y cuando se recibe una respuesta, el receptor de mensajes vuelve a pasar la respuesta al componente.

Para los escenarios unidireccionales y bidireccionales, se pueden especificar propiedades dinámicas y estáticas. Las propiedades estáticas se pueden establecer desde el enlace de método de importación JMS. Algunas de estas propiedades tienen significados especiales para el tiempo de ejecución JMS de SCA.

Resulta importante tener en cuenta que JMS es un enlace asíncrono. Si un componente llamante invoca una importación JMS de forma síncrona (para una operación bidireccional), el componente llamante se bloquea hasta que el servicio JMS devuelve la respuesta.

Figura 33 en la página 77 ilustra cómo se enlaza la importación con el servicio externo.

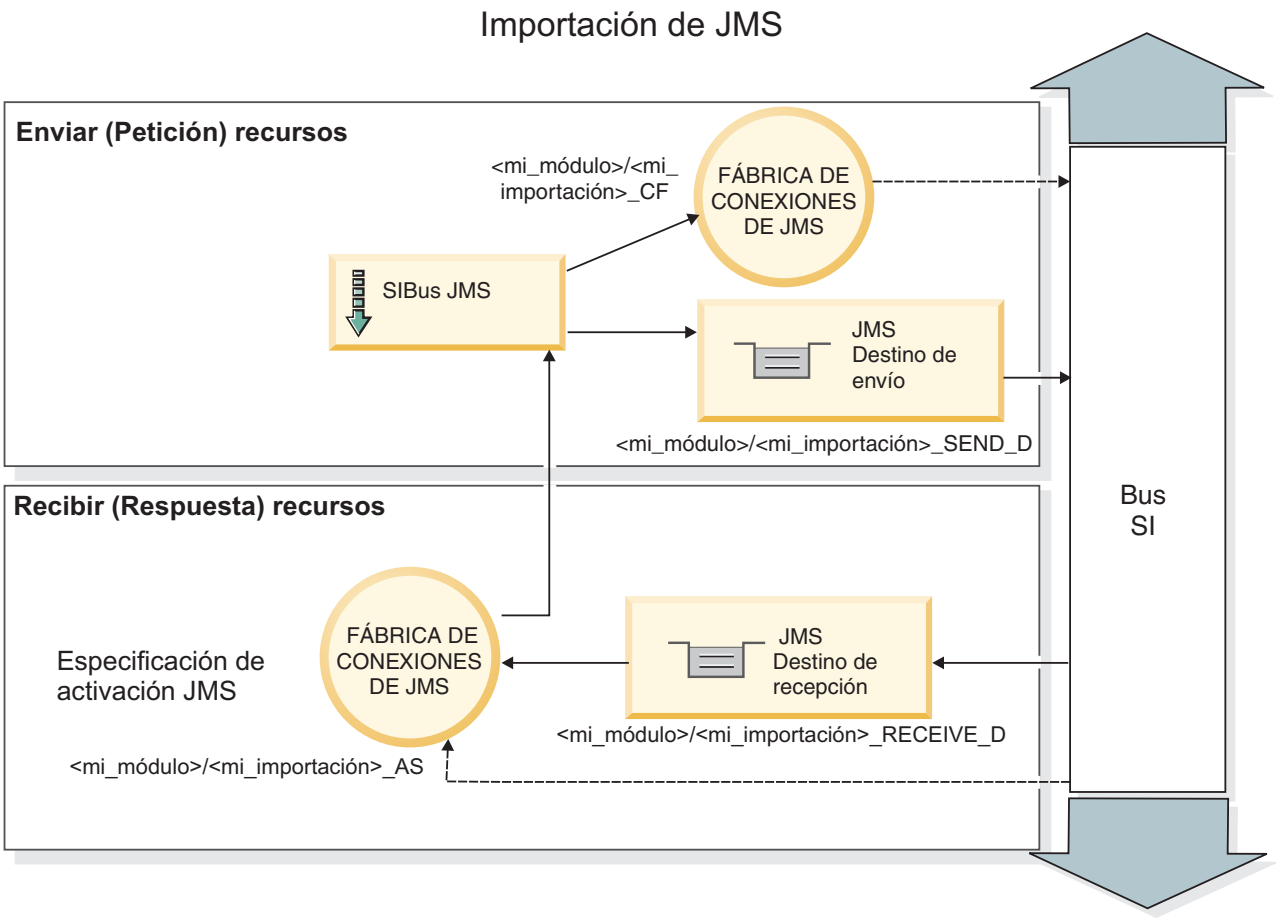


Figura 33. Recursos de enlace de importación JMS

Enlaces de exportación JMS

Los enlaces de exportación JMS proporcionan el medio mediante el cual los módulos SCA pueden ofrecer servicios a aplicaciones JMS externas.

La conexión que forma parte de una exportación JMS es una especificación de activación configurable.

Una exportación JMS ha enviado y recibido destinos.

- El destino indicado en receive es el lugar donde debe colocarse el mensaje de entrada del componente de destino.
- El destino indicado en send es el lugar al que se enviará la respuesta, a menos que el mensaje de entrada lo altere temporalmente mediante la propiedad de cabecera replyTo.

Se despliega un receptor de mensajes para escuchar las peticiones de entrada del destino receive especificado en el enlace de exportación. El destino especificado en el campo send se utiliza para enviar la respuesta a la petición de entrada si el componente invocado proporciona una respuesta. El destino especificado en el campo replyTo del mensaje de entrada altera temporalmente el destino especificado en el campo send.

Figura 34 ilustra cómo se enlaza el solicitante externo con la exportación.

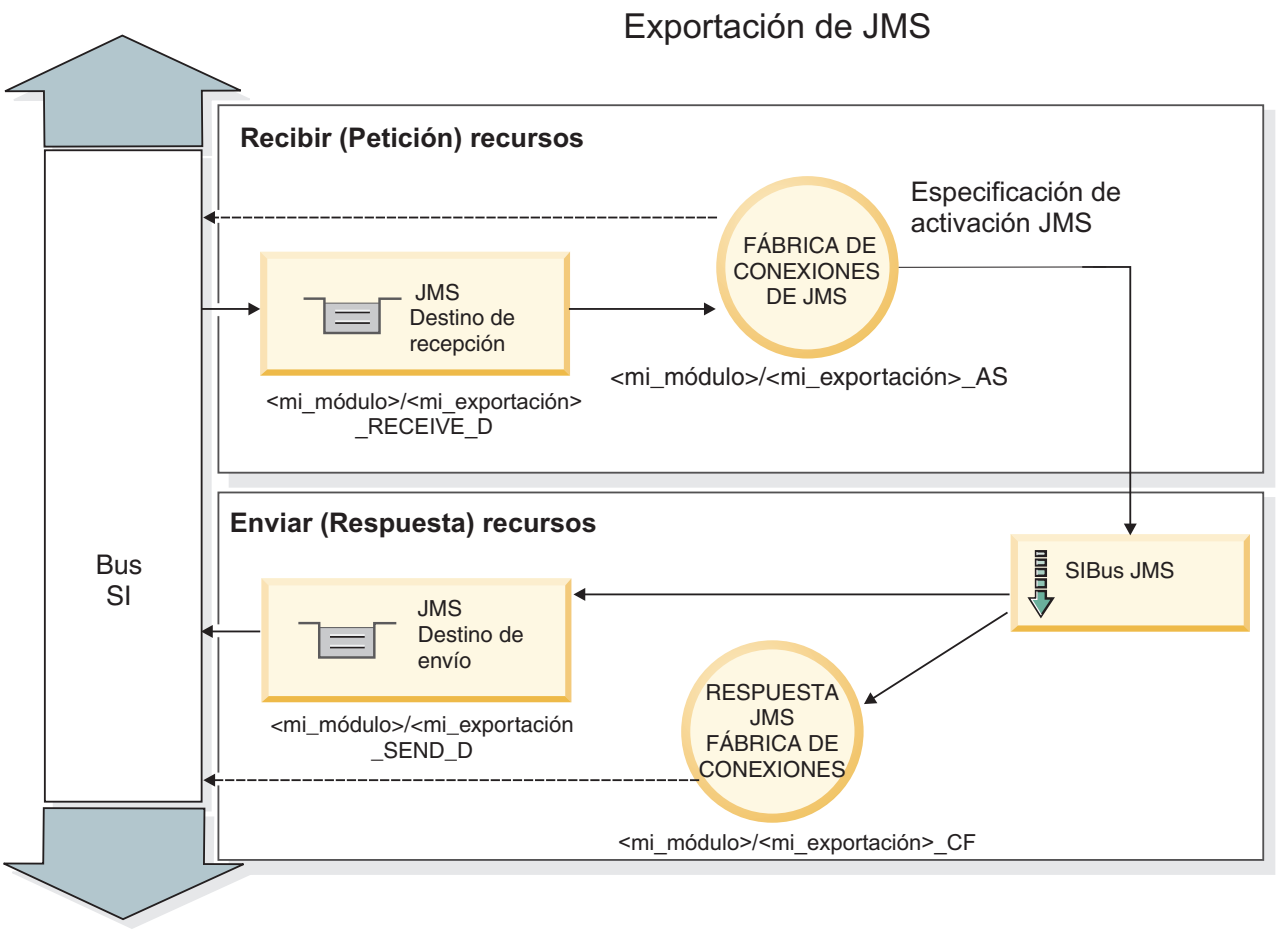


Figura 34. Recursos de enlace de exportación JMS

Integración JMS y adaptadores de recursos

El servicio JMS (Java Message Service) proporciona integración mediante un adaptador de recursos basado en JMS JCA 1.5 disponible. Se proporciona soporte completo para la integración JMS para el adaptador de recursos JMS SIB (Service Integration Bus).

Utilice un proveedor de JMS para el adaptador de recursos JCA 1.5 cuando desee integrarse con un sistema JMX externo compatible con JCA 1.5. Los servicios externos compatibles con JCA 1.5 pueden recibir y enviar mensajes para la integración con los componentes SCA (Service Component Architecture) utilizando el adaptador de recursos JMS SIB.

El uso de adaptadores de recursos JCA 1.5 específicos de otros proveedores no está soportado.

Los módulos JMS no se pueden desplegar en un entorno Java SE. Dichos módulos sólo se despliegan en un entorno Java EE.

Características clave de los enlaces JMS

Las características claves de los enlaces de importación y exportación JMS incluyen cabeceras y recursos Java EE creados.

Cabeceras especiales

Se utilizan las propiedades de cabecera especiales en las importaciones y exportaciones JMS para indicar al destino cómo debe manejar el mensaje.

Por ejemplo, TargetFunctionName se correlaciona desde el método nativo al método de operación.

Recursos Java EE

Se crean varios recursos Java EE cuando se despliegan las importaciones y exportaciones JMS en un entorno Java EE.

ConnectionFactory
Utilizado por los clientes para crear una conexión con el proveedor de JMS.

ActivationSpec
Las importaciones la utilizan para recibir las respuestas a una petición. Las exportaciones la utilizan cuando configuran los puntos finales de mensaje que representan los receptores de mensajes en sus interacciones con el sistema de mensajería.

Destinos

- Destino de envío: en una importación, es el lugar donde se envía la solicitud o el mensaje saliente; en una exportación, es el lugar donde se enviará el mensaje de respuesta, si no se reemplaza por el campo de cabecera JMSReplyTo en el mensaje de entrada.
- Destino de recepción: donde se debe colocar el mensaje de entrada; con las importaciones, se trata de una respuesta; con las exportaciones, se trata de una solicitud.
- Destino de devolución de llamada: el destino del sistema JMS SCA utilizado para almacenar la información de correlación. Ni lea ni escriba en este destino.

La tarea de instalación crea la ConnectionFactory y tres destinos. Además, crea la ActivationSpec para permitir que el receptor de mensajes en tiempo de ejecución escuche las respuestas en el destino de recepción. Las propiedades de estos recursos se especifican en el archivo de exportación o de importación.

Cabeceras JMS

Un mensaje JMS contiene dos tipos de cabeceras: la cabecera del sistema JMS y varias propiedades JMS. Se puede acceder a ambos tipos de cabeceras, bien en un módulo de mediación del objeto de mensajes de servicios (SMO), o bien mediante la API ContextService.

Cabecera del sistema JMS

La cabecera del sistema JMS se representa en el SMO mediante el elemento JMSHeader, que contiene todos los campos que se encuentran, normalmente, en una cabecera JMS. Aunque éstos pueden modificarse en la mediación (o en ContextService), algunos campos de cabecera del sistema JMS establecidos en el SMO no se propagarán en el mensaje JMS de salida, ya que el sistema, o los valores estáticos, los alteran temporalmente.

Los campos clave de la cabecera del sistema JMS que se pueden actualizar en la mediación (o en ContextService) son:

- **JMSType** y **JMSCorrelationID**: valores de las propiedades de la cabecera del mensaje predefinidas específicas.
- **JMSDeliveryMode**: valores de la modalidad de entrega (persistentes o no persistentes). El valor por omisión es persistente).
- **JMSPriority**: valor de prioridad (0 a 9). El valor por omisión es JMS_Default_Priority).

Propiedades JMS

Las propiedades JMS se representan en el SMO como entradas de la lista Propiedades. Las propiedades se pueden añadir, actualizar o suprimir dentro de una mediación, o bien utilizando la API ContextService.

Las propiedades también pueden establecerse estáticamente en el enlace JMS. Las propiedades que se establecen estáticamente alteran temporalmente los valores (que tengan el mismo nombre) que se establecen dinámicamente.

Las propiedades de usuario propagadas desde otros enlaces (por ejemplo, un enlace HTTP) generarán una salida en el enlace JMS en forma de propiedades JMS.

Valores de propagación de cabecera

La propagación de las propiedades y la cabecera del sistema JMS bien sea a desde el mensaje JMS de entrada a los componentes ubicados en sentido descendente, o bien sea desde los componentes ubicados en sentido ascendente al mensaje JMS de salida, se pueden controlar mediante el distintivo Propagate Protocol Header del enlace.

Cuando se establece Propagate Protocol Header, la información de cabecera puede fluir al mensaje o al componente de destino, tal como se describe en la lista siguiente:

- Petición de exportación JMS

  La cabecera JMS recibida en el mensaje se propagará a los componentes de destino por medio del servicio de contexto. Las propiedades JMS recibidas en el mensaje se propagará a los componentes de destino por medio del servicio de contexto.
- Respuesta de exportación JMS

  Cualquiera de los campos de cabecera JMS establecidos en el servicio de contexto se utilizará en el mensaje de salida, si no lo han alterado temporalmente las propiedades establecidas en el enlace de exportación de JMS. Cualquiera de las propiedades establecidas en el servicio de contexto se utilizará en el mensaje de salida, si no la han alterado temporalmente las propiedades establecidas en el enlace de exportación de JMS.
- Petición de importación JMS

  Cualquiera de los campos de cabecera JMS establecidos en el servicio de contexto se utilizará en el mensaje de salida, si no lo han alterado temporalmente las propiedades establecidas en el enlace de importación de JMS. Cualquiera de las propiedades establecidas en el servicio de contexto se utilizará en el mensaje de salida, si no la han alterado temporalmente las propiedades establecidas en el enlace de importación de JMS.
- Respuesta de importación JMS

La cabecera JMS recibida en el mensaje se propagará a los componentes de destino por medio del servicio de contexto. Las propiedades JMS recibidas en el mensaje se propagará a los componentes de destino por medio del servicio de contexto.

Esquema de correlación de destinos de respuesta dinámicos temporales de JMS

El esquema de correlación de destinos de respuesta dinámicos temporales provoca que se cree una cola o tema dinámica exclusiva para cada petición enviada.

El destino de respuesta estático especificado en la importación se utiliza para derivar la naturaleza de la cola o el tema de destino dinámico temporal. Esto se establece en el campo **ReplyTo** de la petición y la importación JMS escucha respuestas en dicho destino. Cuando se recibe la respuesta, se vuelve a poner en la cola en el destino de respuesta estático para el proceso asíncrono. El campo **CorrelationID** de la respuesta no se utiliza y no se debe establecer.

Problemas transaccionales

Cuando se utiliza un destino dinámico temporal, la respuesta se debe consumir en la misma hebra que la respuesta enviada. La petición se debe enviar fuera de la transacción global y se debe confirmar antes de ser recibida por el servicio de programa de fondo y se devuelve una respuesta.

Persistencia

Las colas dinámicas temporales son entidades de vida corta y no garantizan el mismo nivel de persistencia asociada a una cola o un tema estático. Una cola o un tema dinámico temporal no sobrevivirá un renicio de servidor y no generará mensajes. Después de que el mensaje se haya vuelto a poner en la cola del destino de respuesta estático, conserva la persistencia definida en el mensaje.

Tiempo de espera

La importación espera a recibir la respuesta en el destino de respuesta dinámico temporal para una cantidad de tiempo fijo. Este intervalo de tiempo se tomará del calificador de tiempo Caducidad de respuesta SCA, si está establecido, de lo contrario, el tiempo toma el valor predeterminado de 60 segundos. Si se excede el tiempo de espera, la importación lanza una excepción `ServiceTimeoutRuntimeException`.

Clientes externos

El servidor puede enviar mensajes o recibir mensajes de clientes externos utilizando enlaces JMS.

Un cliente externo (como un portal Web o un sistema de empresa) puede enviar un mensaje a un módulo SCA del servidor o lo puede invocar un componente desde el servidor.

Los componentes de exportación JMS despliegan los receptores de mensajes para escuchar las peticiones que entran en el destino de recepción especificado en el enlace de exportación. El destino especificado en el campo de envío se utiliza para enviar la respuesta a la petición de entrada si la aplicación invocada proporciona una respuesta. De esta forma, un cliente externo puede invocar aplicaciones con el enlace de exportación.

Las importaciones JMS se enlazan y pueden entregar mensajes a los clientes externos. Este mensaje puede necesitar o no una respuesta del cliente externo.

Trabajo con clientes externos:

Es posible que un cliente externo (es decir, un cliente que está fuera del servidor) necesite interactuar con una aplicación instalada en el servidor.

Acerca de esta tarea

Considere un escenario muy sencillo en el cual un cliente externo desea interactuar con una aplicación en el servidor. En la figura se describe un escenario típico sencillo.

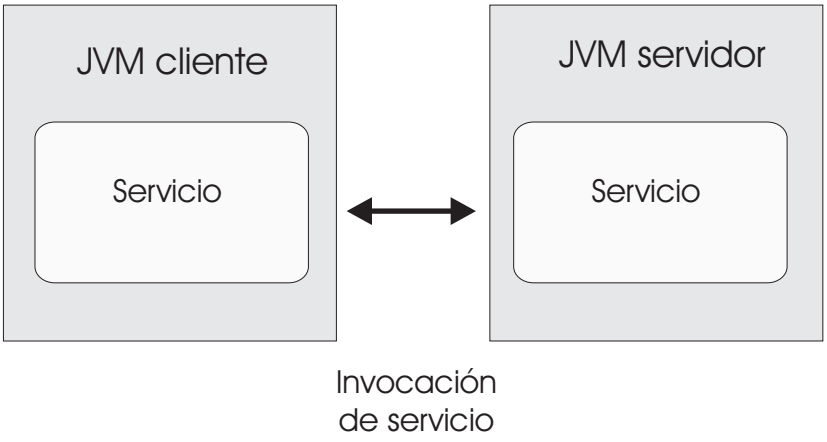


Figura 35. Escenario de caso de uso sencillo: el cliente externo interactúa con la aplicación de servidor

La aplicación SCA incluye una exportación con un enlace JMS; que hace que la aplicación esté disponible para los clientes externos.

Cuando tenga un cliente externo en una máquina virtual Java (JVM) independiente del servidor, debe seguir varios pasos para crear una conexión e interactuar con una exportación JMS. El cliente obtiene un InitialContext con los valores correctos y busca los recursos a través de JNDI. A continuación, el cliente utiliza la especificación JMS 1.1 para acceder a los destinos y enviar y recibir mensajes en los destinos.

Los nombres JNDI por omisión de los recursos creados automáticamente por el tiempo de ejecución aparecen en el tema de configuración de esta sección. No obstante, si tiene recursos creados previamente, utilice esos nombres JNDI.

Procedimiento

1. Configure los destinos JMS y la fábrica de conexiones para enviar el mensaje.
2. Asegúrese de que el contexto JNDI, el puerto para el adaptador de recursos SIB y el puerto de la rutina de arranque de mensajería sean correctos.

   El servidor utiliza algunos puertos por omisión, pero si existen más servidores instalados en ese sistema, se crean puertos alternativos durante la instalación para evitar conflictos con otras instancias de servidor. También puede utilizar la consola administrativa para determinar los puertos que el servidor está utilizando. Vaya a **Servidores** → **Servidores de aplicaciones** →

nombre_su_servidor → **Configuración** y pulse **Puertos** en **Comunicación**. A continuación, puede editar el puerto que se está utilizando.

3. El cliente obtiene un contexto inicial con los valores correctos y busca los recursos a través de JNDI.
4. Utilizando las especificaciones JMS 1.1, el cliente accede a los destinos y a los mensajes de envío y recepción en los destinos.

Resolución de problemas de enlaces JMS

Puede diagnosticar y solucionar problemas con los enlaces JMS.

Excepciones de implementación

Como respuesta a las distintas condiciones de error, la implementación de la importación y exportación JMS puede devolver uno de estos dos tipos de excepciones:

- Excepción empresarial de servicio: esta excepción se devuelve si se ha producido la excepción especificada de error en la interfaz empresarial de servicio (tipo de puerto WSDL).
- Excepción de tiempo de ejecución de servicio: se produce en todos los demás casos. En la mayoría de los casos, la excepción cause contendrá la excepción (JMSException) original.

  Por ejemplo, la importación espera sólo un mensaje de respuesta para cada mensaje de solicitud. Si llega más de una respuesta o si llega una respuesta con demora (una para la que se ha pasado el tiempo de caducidad de respuesta de SCA), se genera una excepción de tiempo de ejecución de servicio. Se retrotrae la transacción y el mensaje de respuesta se vuelve a poner fuera de la cola o lo gestiona el Gestor de sucesos con error.

Condiciones de error principales

Las condiciones de error principales de los enlaces JMS quedan determinadas por la semántica de transacciones, por la configuración del proveedor JMS o por la referencia a un comportamiento existente de otros componentes. Las primeras condiciones de error son:

- No se puede conectar al proveedor JMS o al destino.

  Si no se puede conectar al proveedor JMS para recibir mensajes, no se podrá iniciar el receptor de mensajes. Esta condición se anotará cronológicamente en las anotaciones cronológicas de WebSphere Application Server. Los mensajes permanentes permanecerán en el destino hasta que se recuperen satisfactoriamente (o caduquen).

  Si no se puede conectar al proveedor JMS para enviar mensajes de salida, provocará la retrotracción de la transacción que controla el envío.
- No se puede analizar un mensaje de entrada o construir un mensaje de salida.

  Una anomalía en el enlace de datos o en el manejador de datos provoca la retrotracción de la transacción que controla el trabajo.
- No se puede enviar el mensaje de salida.

  Una anomalía al enviar un mensaje provoca una retrotracción de la transacción correspondiente.
- Varios mensajes de respuesta con demora o mensajes inesperados.

  La importación espera sólo un mensaje de respuesta para cada mensaje de solicitud. Además, el periodo de tiempo válido en el que se puede recibir una respuesta viene determinado por el calificador de caducidad de respuesta de SCA en la solicitud. Cuando llega una respuesta o se supera el tiempo de espera,

se suprime el registro de correlaciones. Si llegan mensajes de respuesta de forma inesperada o llegan con demora, se genera una excepción de tiempo de espera de servicio.

- Excepción en tiempo de ejecución de tiempo de espera excedido de servicio provocado por una respuesta con demora al utilizar el esquema de correlación de destino de respuesta dinámica temporal.

  La importación de JMS provocará un tiempo de espera excedido después de un periodo de tiempo determinado por el calificador de caducidad de respuesta de SCA, o si esto no es establece utilizará como valor por omisión 60 segundos.

Mensajes SCA basados en JMS que no aparecen en el gestor de sucesos con anomalía

Si los mensajes SCA originados en una interacción JMS fallan, los mensajes deberán aparecer en el gestor de sucesos con anomalía. Si los mensajes no aparecen en el gestor de sucesos con error, asegúrese de que el destino SIB subyacente del destino JMS tenga un valor máximo de entregas con error mayor que 1. Si se establece este valor en 2 o más, se habilita la interacción con el gestor de sucesos con error durante las invocaciones de SCA para los enlaces JMS.

Manejo de excepciones

El modo en que está configurado el enlace de datos determina cómo se tratan las excepciones generadas por los manejadores de datos o los enlaces de datos. Además, la naturaleza del flujo de mediación dicta el comportamiento del sistema cuando se genera una excepción de este estilo.

Pueden surgir varios problemas cuando su enlace llama a un manejador de datos o enlace de datos. Por ejemplo, un manejador de datos puede recibir un mensaje que tiene una carga útil dañada, o puede tratar de leer un mensaje que tenga un formato incorrecto.

El modo en que el enlace gestiona una excepción de este estilo viene determinado por el modo de implementar el manejador de datos o el enlace de datos. El comportamiento recomendado es diseñar su propio enlace de datos para producir una excepción `DataBindingException`.

Cuando se produce cualquier excepción en tiempo de ejecución, incluida `DataBindingException`:

- Si el flujo de mediación está configurado para ser transaccional, el mensaje JMS , por omisión, se almacena en el Gestor de sucesos anómalos para la repetición o supresión manual.

  Nota: Puede cambiar la modalidad de recuperación en el enlace, de modo que el mensaje se retrotraiga en lugar de almacenarse en el Gestor de sucesos anómalos.
- Si el flujo de mediación no es transaccional, el flujo de mediación se anota y se pierde el mensaje.

La situación es similar para un manejador de datos. Dado que el enlace de datos invoca el manejador de datos, cualquier excepción del manejador de datos se incluye en una excepción de enlace de datos. Por lo tanto, se le notifica una excepción `DataHandlerException` como una `DataBindingException`.

Enlaces JMS genéricos

El enlace JMS genérico proporciona conectividad a los proveedores de terceros compatibles con JMS 1.1. El funcionamiento de los enlaces JMS genéricos es similar a los enlaces JMS.

El servicios proporcionado mediante un enlace JMS permite a un módulo SCA (Service Component Architecture) realizar llamadas o recibir mensajes de sistemas externos. El sistema puede ser un sistema JMS externo.

El enlace JMS genérico proporciona integración con los proveedores JMS no de JCA compatibles con la versión 1.5 que soportan JMS 1.1 e implementa el recurso del servidor de aplicaciones JMS. El enlace JMS genérico soporta estos proveedores de JMS (incluidos Oracle AQ, TIBCO, SonicMQ, WebMethods, BEA WebLogic y WebSphere MQ) que no dan soporte a JCA 1.5, pero sí que dan soporte a Application Server Facility de la especificación JMS 1.1. Este enlace no soporta el proveedor JMS incorporado de WebSphere (SIBJMS), que es un proveedor JMS de JCA 1.5; al utilizar el proveedor, utilice los "Enlaces JMS" en la página 75.

Utilice este enlace genérico cuando integre con un sistema basado en JMS, no de JCA, compatible con la versión 1.5, dentro de un entorno SCA. Las aplicaciones externas de destino pueden posteriormente recibir mensajes y enviar mensajes para la integración con un componente SCA.

Visión general de enlaces de JMS genérico

Los enlaces JMS genéricos son enlaces JMS no de JCA que proporcionan conectividad entre el entorno de SCA (Service Component Architecture) y los sistemas JMS que cumplen con JMS 1.1 y que implementan el recurso del servidor de aplicaciones JMS.

Enlaces JMS genéricos

Los aspectos más relevantes de los enlaces de importación y exportación JMS genéricos son:

- Puerto receptor: permite que los proveedores JMS no basados en JCA recibir mensajes y asignarlos a un Bean controlado por mensajes (MDB)
- Conexiones: encapsulan una conexión virtual entre un cliente y una aplicación de proveedor.
- Destinos: utilizados por un cliente para especificar el destino de los mensajes que produce o el origen de los mensajes que consume.
- Datos de autenticación: utilizados para proteger el acceso al enlace.

Enlaces de importación JMS genéricos

Los enlaces de importación JMS genéricos permiten a los componentes del módulo SCA comunicarse con los servicios proporcionados por proveedores de JMS compatibles con sistemas no JCA 1.5.

La parte de la conexión de una importación JMS es una fábrica de conexiones. Una fábrica de conexiones es el objeto que utiliza un cliente para crear una conexión con un proveedor, encapsula un conjunto de parámetros de configuración de conexión definidos por un administrador. Cada fábrica de conexiones es una instancia de la interfaz ConnectionFactory, QueueConnectionFactory o TopicConnectionFactory.

La interacción con los sistemas JMS externos incluye la utilización de destinos para enviar peticiones y recibir respuestas.

Se da soporte a dos tipos de escenarios de uso para los enlaces de importación JMS genéricos, dependiendo del tipo de operación que se invoque:

- Unidireccional: la importación JMS genérica coloca un mensaje en el destino de envío configurado en el enlace de importación. No se envía nada al campo replyTo de la cabecera JMS.
- Bidireccional (petición y respuesta): la importación JMS genérica coloca un mensaje en el destino de envío y persiste la respuesta que recibe del componente SCA.

  El destino indicado en receive se establece en la propiedad de la cabecera replyTo del mensaje de salida. Se despliega un bean MDB (Message Driven Bean) para escuchar en el destino de recepción y cuando se recibe una respuesta, el MDB vuelve a pasar la respuesta al componente.

  El enlace de importación se puede configurar (mediante el campo **Esquema de correlación de respuesta** de WebSphere Integration Developer) para que espere que el ID de correlación de mensaje de respuesta se haya copiado del ID mensaje de petición (el valor por omisión) o del ID de correlación de mensaje de petición.

Para los escenarios unidireccionales y bidireccionales, se pueden especificar propiedades dinámicas y estáticas. Las propiedades estáticas se pueden establecer desde el enlace de método de importación JMS genérica. Algunas de estas propiedades tienen significados especiales para el tiempo de ejecución JMS de SCA.

Resulta importante tener en cuenta que el JMS genérico es un enlace asíncrono. Si un componente llamante invoca una importación JMS genérica de forma síncrona (para una operación bidireccional), el componente llamante se bloquea hasta que el servicio JMS devuelve la respuesta.

Figura 36 en la página 87 ilustra cómo se enlaza la importación con el servicio externo.

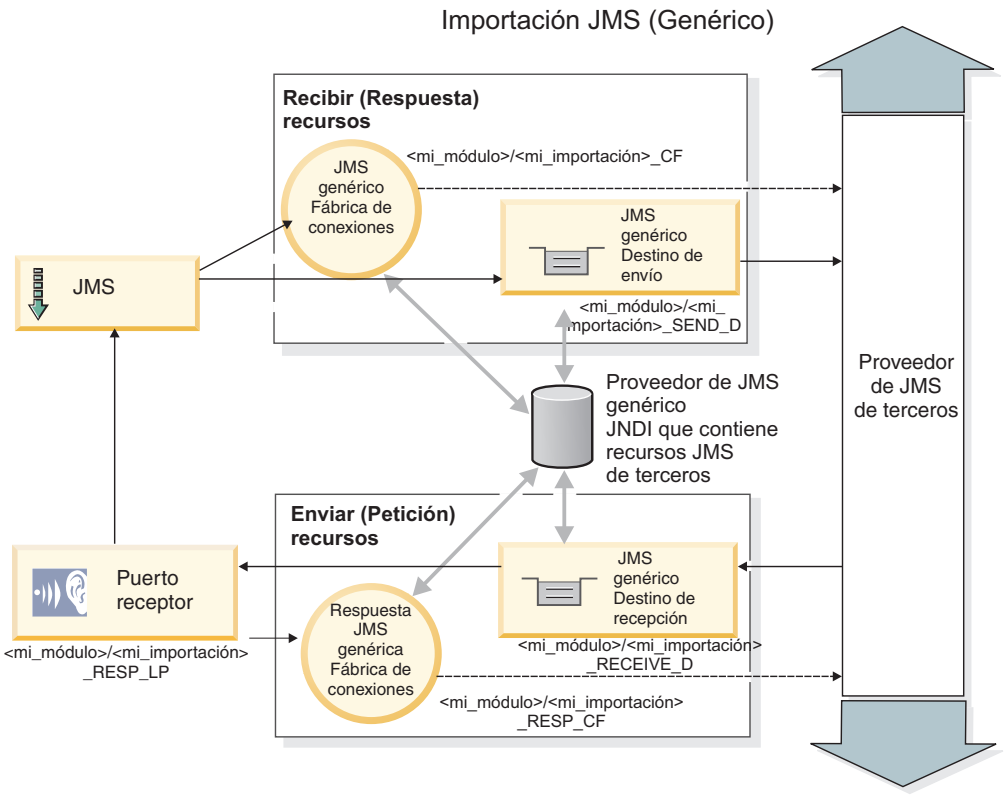


Figura 36. Recursos de enlace de importación de JMS genérico

Enlaces de exportación de JMS genéricos

Los enlaces de exportación JMS genéricos proporcionan el medio mediante el cual los módulos SCA pueden ofrecer servicios a aplicaciones JMS externas.

La parte de conexión de una exportación JMS está formada por una ConnectionFactory y un ListenerPort.

Una exportación JMS genérica ha enviado y recibido destinos.

- El destino indicado en receive es el lugar donde debe colocarse el mensaje de entrada del componente de destino.
- El destino indicado en send es el lugar al que se enviará la respuesta, a menos que el mensaje de entrada lo altere temporalmente mediante la propiedad de cabecera replyTo.

Se despliega un MDB para que escuche las peticiones de entrada en el destino indicado en receive especificado en el enlace de exportación.

- El destino especificado en el campo send se utiliza para enviar la respuesta a la petición de entrada si el componente invocado proporciona una respuesta.
- El destino especificado en el campo replyTo del mensaje de entrada altera temporalmente el destino especificado en el campo send.
- En los escenarios de petición y respuesta, se puede configurar el enlace de importación (mediante el campo **Esquema de correlación de respuesta** de WebSphere Integration Developer) para que espere que la respuesta copie el ID de mensaje de petición en el campo ID de correlación del mensaje de respuesta (valor por omisión), o que la respuesta pueda copiar el ID de correlación de petición en el campo ID de correlación del mensaje de respuesta.

Figura 37 ilustra cómo se enlaza el solicitante externo con la exportación.

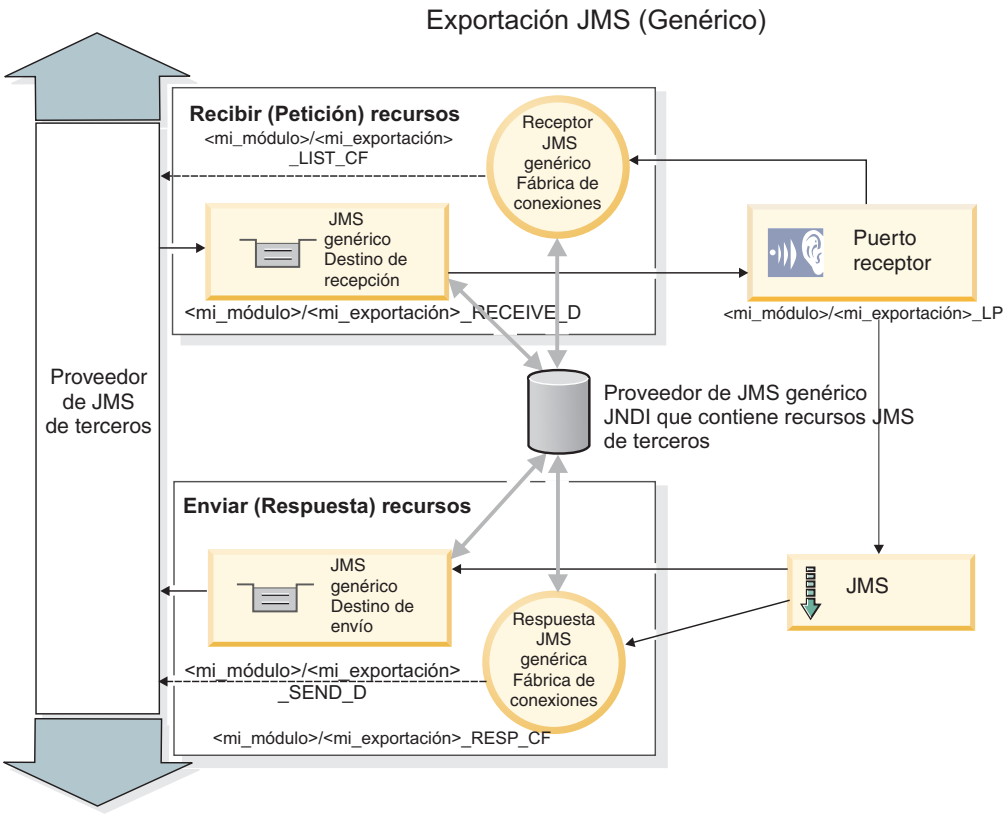


Figura 37. Recursos de enlace de exportación de JMS genérico

Características clave de los enlaces JMS genéricos

Las características del enlace de exportación e importación JMS genérico son coherentes con las de los enlaces de importación JMS incorporados de WebSphere y JMS de MQ. Las características clave incluyen definiciones de cabecera y el acceso a los recursos Java EE existentes. No obstante, debido a su naturaleza genérica, no hay opciones de conectividad específicas del proveedor JMS y este enlace tiene una posibilidad limitada de generar recursos durante el despliegue y la instalación.

Importaciones genéricas

Al igual que la aplicación de importación JMS de MQ, la implementación JMS genérica es asíncrona y soporta tres invocaciones: unidireccional, bidireccional (también conocida como de petición-respuesta) y devolución de llamada.

Cuando se despliega la importación JMS, se despliega un bean controlado por mensajes (MDB) proporcionado por el entorno de ejecución. El MDB escucha las respuestas al mensaje de petición. El MDB se asocia (escucha) al destino enviado con la petición en el campo de cabecera replyTo del mensaje JMS.

Exportaciones genéricas

Los enlaces de exportación JMS genéricos son distintos de los enlaces de exportación EIS en el manejo de la devolución de los resultados. Una exportación

JMS genérica envía explícitamente la respuesta al destino replyTo especificado en el mensaje de entrada. Si no se especifica ninguno, se utiliza el destino de envío.

Cuando se despliega la exportación JMS genérica, se despliega un bean controlado por mensajes (un MDB distinto del utilizado para las importaciones JMS genéricas). Este escucha las peticiones de entrada en el destino de recepción y envía las peticiones para que las procese el tiempo de ejecución de SCA.

Cabeceras especiales

Se utilizan las propiedades de cabecera especiales en las importaciones y exportaciones de JMS genéricos para indicar al enlace de destino cómo debe manejar el mensaje.

Por ejemplo, el selector de función por omisión utiliza la propiedad TargetFunctionName para identificar el nombre de la operación en la interfaz de exportación que se está invocando.

Nota: El enlace de importación se puede configurar para establecer la cabecera TargetFunctionName en el nombre de cada operación.

Recursos Java EE

Se crea una serie de recursos Java EE cuando se despliega un enlace JMS en un entorno Java EE.

- El puerto de receptor para escuchar en el destino (respuesta) de recepción (sólo bidireccional) las importaciones y en el destino de recepción (petición) las exportaciones
- La fábrica de conexiones JMS genérica para outboundConnection (importación) e inboundConnection (exportación)
- El destino JMS genérico para los destinos de envío (importación) y recepción (exportación; sólo bidireccional)
- La fábrica de conexiones JMS genérica para responseConnection (sólo bidireccional y opcional; de lo contrario, se utiliza outboundConnection para las importaciones y se utiliza inboundConnection para las exportaciones)
- El destino JMS genérico para el destino de recepción (importación) y envío (exportación) (sólo bidireccional)
- El destino JMS de devolución de llamada del proveedor de mensajería por omisión se utiliza para acceder al destino de cola de devolución de llamada SIB (sólo bidireccional)
- La fábrica de conexiones JMS de devolución de llamada del proveedor de mensajería por omisión se utiliza para acceder al destino JMS de devolución de llamada (sólo bidireccional)
- El destino de cola de devolución de llamada SIB se utiliza para almacenar información acerca del mensaje de petición para utilizarlo durante el proceso de la respuesta (sólo bidireccional)

La tarea de instalación crea ConnectionFactory, los tres destinos y ActivationSpec a partir de la información de los archivos de importación y exportación.

Cabeceras JMS genéricas

Las cabeceras JMS genéricas son objetos SDO (Service Data Objects) que contienen todas las propiedades de los mensajes JMS genéricos. Estas propiedades pueden ser del mensaje de entrada o pueden ser las propiedades que se aplicarán al mensaje de salida.

Un mensaje JMS contiene dos tipos de cabeceras: la cabecera del sistema JMS y varias propiedades JMS. Se puede acceder a ambos tipos de cabeceras, bien en un módulo de mediación del objeto de mensajes de servicios (SMO), o bien mediante la API ContextService.

Las propiedades siguientes se establecen estáticamente en methodBinding:

- JMSType
- JMSCorrelationID
- JMSDeliveryMode
- JMSPriority

El enlace JMS genérico también da soporte a la modificación dinámica de las cabeceras JMS y de las propiedades del mismo modo que los enlaces JMS y los enlaces JMS de MQ.

Algunos proveedores JMS genéricos imponen restricciones sobre las propiedades que puede establecer la aplicación y sus combinaciones. Debe consultar la documentación del producto de terceros para obtener más información. No obstante, se ha añadido una propiedad adicional a methodBinding, ignoreInvalidOutboundJMSProperties, que permite propagar cualquier excepción.

Las cabeceras JMS genéricas y las propiedades de los mensajes se utilizan únicamente cuando se ha activado el conmutador del enlace SCDL de SCA (Service Component Architecture). Cuando se activa el conmutador, se propaga la información de contexto. Por omisión, este conmutador está encendido. Para impedir que se propague la información de contexto, cambie el valor a `false`.

Cuando la propagación de contexto está habilitada, la información de cabecera puede fluir al mensaje o al componente de destino. Para activar y desactivar la propagación del contexto, especifique `true` o `false` para el atributo contextPropagationEnabled de los enlaces de importación y exportación. Por ejemplo:

```
<esbBinding xsi:type="eis:JMSImportBinding" contextProgagationEnabled="true">
```

El valor por omisión es `true`.

Resolución de problemas de enlaces JMS genéricos

Puede diagnosticar y corregir problemas con enlaces JMS genéricos.

Excepciones de implementación

Como respuesta a las distintas condiciones de error, la implementación de la importación y exportación JMS genérica puede devolver uno de estos dos tipos de excepciones:

- Excepción empresarial de servicio: esta excepción se devuelve si se ha producido la excepción especificada de error en la interfaz empresarial de servicio (tipo de puerto WSDL).

- Excepción de tiempo de ejecución de servicio: se produce en todos los demás casos. En la mayoría de los casos, la excepción cause contendrá la excepción (JMSException) original.

Resolución de problemas de caducidad de mensajes JMS genéricos

Los mensajes de petición de un proveedor de JMS caducan.

La *caducidad de petición* hace referencia a la caducidad de un mensaje de petición de un proveedor JMS cuando se alcanza la hora definida por JMSExpiration en el mensaje de petición. Tal como sucede con otros enlaces JMS, el enlace JMS genérico gestiona la caducidad estableciendo la del mensaje de devolución de llamada efectuado por la importación en el mismo valor que para la solicitud de salida. La notificación de la caducidad del mensaje de devolución de llamada indicará que el mensaje de petición ha caducado y que se debe notificar al cliente mediante una excepción empresarial.

No obstante, si el destino de la devolución de llamada pasa al proveedor de terceros, este tipo de caducidad de petición no está soportada.

La *caducidad de respuesta* hace referencia a la caducidad de un mensaje de respuesta de un proveedor JMS cuando se alcanza la hora definida por JMSExpiration en el mensaje de respuesta.

La caducidad de respuesta para el enlace JMS genérico no está soportada porque el comportamiento exacto de la caducidad de un proveedor JMS de terceros no está definido. Sin embargo, puede comprobar que la respuesta no haya caducado cuando se reciba.

Para los mensajes de petición de salida, el valor JMSExpiration se calculará a partir del tiempo que se haya esperado y de los valores requestExpiration contenidos en asyncHeader, si se han establecido.

Resolución de problemas de errores de fábrica de conexiones JMS genéricos

Cuando define determinados tipos de fábricas de conexiones en su proveedor JMS genérico, es posible que reciba un mensaje de error cuando intenta iniciar una aplicación. Puede modificar la fábrica de conexiones externa para evitar este problema.

Al iniciar una aplicación, puede recibir el siguiente mensaje de error:

```
El tipo JMSConnectionFactory
del puerto de escucha MDB no coincide con el tipo JMSDestination
```

Este problema puede surgir cuando define las fábricas de conexiones externas. Específicamente la excepción se puede generar cuando crea una fábrica de conexiones de temas JMS 1.0.2, en lugar de una fábrica de conexiones JMS 1.1 (unificada) (es decir, una que pueda dar soporte a las comunicaciones de punto a punto y de publicación/suscripción).

Para solucionar este problema, lleve a cabo los pasos siguientes:

1. Acceda al proveedor JMS genérico que está utilizando.
2. Sustituya la fábrica de conexiones de temas JMS 1.0.2 que ha definido por una fábrica de conexiones JMS 1.1 (unificada).

Cuando inicie la aplicación con la fábrica de conexiones JMS 1.1 que acaba de definir, ya no recibirá ningún mensaje de error.

Mensajes SCA basados en JMS genéricas que no aparecen en el gestor de sucesos con anomalía

Si los mensajes SCA originados en una interacción JMS genérica fallan, los mensajes deberán aparecer en el gestor de sucesos con anomalía. Si los mensajes no aparecen en el gestor de sucesos con error, asegúrese de que el valor de la propiedad de máximo de reintentos en el puerto receptor subyacente sea mayor o igual que 1. Si establece este valor en 1 o más, se habilita la interacción con el gestor de sucesos con error durante las invocaciones de SCA para los enlaces JMS genéricos.

Manejo de excepciones

El modo en que está configurado el enlace de datos determina cómo se tratan las excepciones generadas por los manejadores de datos o los enlaces de datos. Además, la naturaleza del flujo de mediación dicta el comportamiento del sistema cuando se genera una excepción de este estilo.

Pueden surgir varios problemas cuando su enlace llama a un manejador de datos o enlace de datos. Por ejemplo, un manejador de datos puede recibir un mensaje que tiene una carga útil dañada, o puede tratar de leer un mensaje que tenga un formato incorrecto.

El modo en que el enlace gestiona una excepción de este estilo viene determinado por el modo de implementar el manejador de datos o el enlace de datos. El comportamiento recomendado es diseñar su propio enlace de datos para producir una excepción `DataBindingException`.

La situación es similar para un manejador de datos. Dado que el enlace de datos invoca el manejador de datos, cualquier excepción del manejador de datos se incluye en una excepción de enlace de datos. Por lo tanto, se le notifica una excepción `DataHandlerException` como una `DataBindingException`.

Cuando se produce cualquier excepción de tiempo de ejecución, incluida la excepción `DataBindingException`:

- Si el flujo de mediación se ha configurado para ser transaccional, el mensaje JMS se almacena en el gestor de sucesos anómalos de forma predeterminada para una respuesta o supresión manual.

  Nota: Puede cambiar la modalidad de recuperación en el enlace, de forma que el mensaje se retrotrae, en lugar de almacenarse en el gestor de sucesos anómalos.
- Si el flujo de mediación no es transaccional, el flujo de mediación se anota y se pierde el mensaje.

La situación es similar para un manejador de datos. Puesto que el enlace de datos llama al manejador de datos, se produce una excepción de manejador de datos dentro de una excepción de enlace de datos. Por lo tanto, se informa de una excepción `DataHandlerException` como una `DataBindingException`.

Enlaces JMS de WebSphere MQ

El enlace JMS de WebSphere MQ proporciona integración con aplicaciones externas que utilizan un proveedor basado en JMS de WebSphere MQ.

Utilice los enlaces de exportación e importación de WebSphere MQ para integrarse directamente con los sistemas JMS o JMS MQ externos desde el entorno de servidor. Esto elimina la necesidad de utilizar las características de Enlace MQ o Enlace de cliente de Service Integration Bus.

Cuando un componente interactúa con un servicio basado en JMS de WebSphere MQ, a través de una importación, el enlace de importación JMS de WebSphere MQ utiliza un destino al que se enviarán los datos y un destino donde se puede recibir la respuesta. La conversión de los datos a y desde un mensaje JMS se realiza mediante el manejador de datos JMS o el componente Data Binding Edge.

Cuando un módulo SCA proporciona un servicio a clientes JMS de WebSphere MQ, el enlace de exportación JMS de WebSphere MQ utiliza un destino donde se puede recibir la petición y donde se puede enviar la respuesta. La conversión de los datos a y desde un mensaje JMS se realiza a través del manejador de datos o el enlace de datos JMS.

El selector de función proporciona una correlación con la operación del componente de destino que se va a invocar.

Visión general de enlaces JMS de WebSphere MQ

El enlace JMS de WebSphere MQ proporciona integración con las aplicaciones externas que utilizan el proveedor de JMS de WebSphere MQ.

Tareas administrativas de WebSphere MQ

Se espera que el administrador del sistema WebSphere MQ cree el gestor de colas WebSphere MQ subyacente, que utilizará los enlaces JMS de WebSphere MQ, antes de ejecutar una aplicación que contiene estos enlaces.

Enlaces de importación JMS de WebSphere MQ

La importación JMS de WebSphere MQ permite a los componentes del módulo SCA comunicarse con los servicios proporcionados por los proveedores basados en JMS de WebSphere MQ. Debe utilizar una versión soportada de WebSphere MQ. Puede encontrar los requisitos detallados de hardware y software en las páginas de soporte de IBM.

Se da soporte a dos tipos de situaciones de uso para los enlaces de importación JMS de WebSphere MQ, dependiendo del tipo de operación que se invoque:

- Unidireccional: la importación JMS de WebSphere MQ coloca un mensaje en el destino de envío configurado en el enlace de importación. No se envía nada al campo replyTo de la cabecera JMS.
- Bidireccional (petición y respuesta): la importación JMS de WebSphere MQ coloca un mensaje en el destino de envío.

  El destino indicado en receive se establece en el campo de la cabecera replyTo. Se despliega un bean MDB (Message Driven Bean) para escuchar en el destino de recepción y cuando se recibe una respuesta, el MDB vuelve a pasar la respuesta al componente.

  El enlace de importación se puede configurar (mediante el campo **Esquema de correlación de respuesta** de WebSphere Integration Developer) para que espere que el ID de correlación de mensaje de respuesta se haya copiado del ID mensaje de petición (el valor por omisión) o del ID de correlación de mensaje de petición.

Para los escenarios de uso unidireccionales y bidireccionales, se pueden especificar propiedades de cabecera dinámicas y estáticas.Las propiedades estáticas se pueden establecer desde el enlace de método de importación JMS. Algunas de estas propiedades tienen significados especiales para el tiempo de ejecución JMS de SCA.

Resulta importante tener en cuenta que JMS de WebSphere MQ es un enlace asíncrono. Si un componente llamante invoca una importación JMS de WebSphere MQ de forma síncrona (para una operación bidireccional), el componente llamante se bloquea hasta que el servicio JMS devuelve la respuesta.

Figura 38 ilustra cómo se enlaza la importación con el servicio externo.

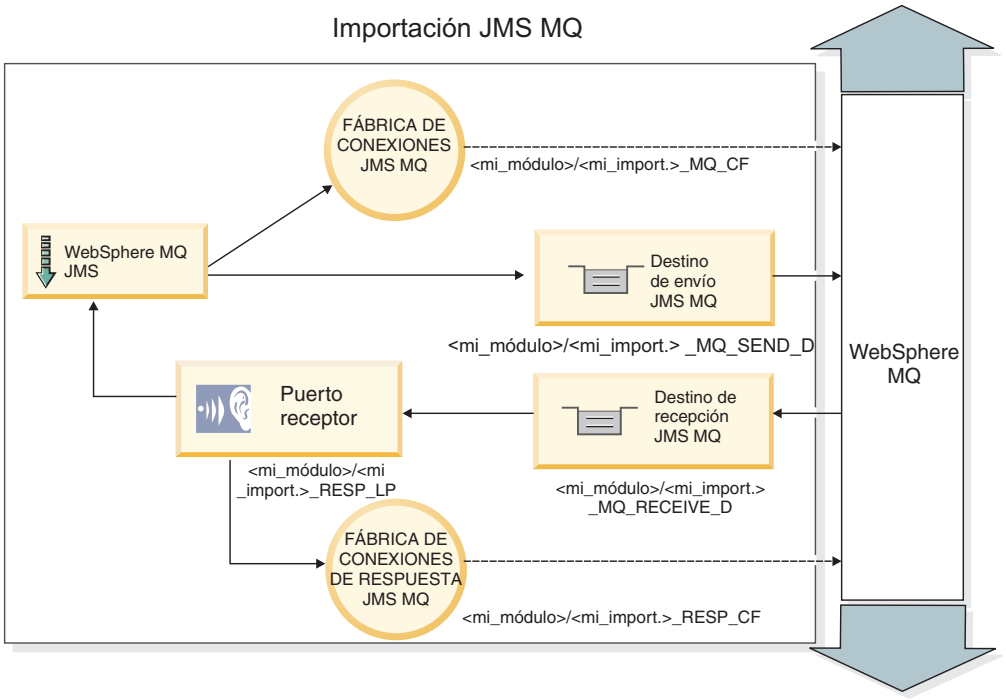


Figura 38. Recursos de enlace de importación de JMS de WebSphere MQ

Enlaces de exportación JMS de WebSphere MQ

El enlace de exportación JMS de WebSphere MQ proporciona el medio mediante el cual los módulos SCA pueden ofrecer servicios a aplicaciones JMS externas en el proveedor de JMS basado en WebSphere MQ.

Se despliega un MDB para que escuche las peticiones de entrada en el destino indicado en receive especificado en el enlace de exportación. El destino especificado en el campo send se utiliza para enviar la respuesta a la petición de entrada si el componente invocado proporciona una respuesta. El destino especificado en el campo replyTo del mensaje de respuesta altera temporalmente el destino especificado en el campo send.

Figura 39 en la página 95 ilustra cómo se enlaza el solicitante externo con la exportación.

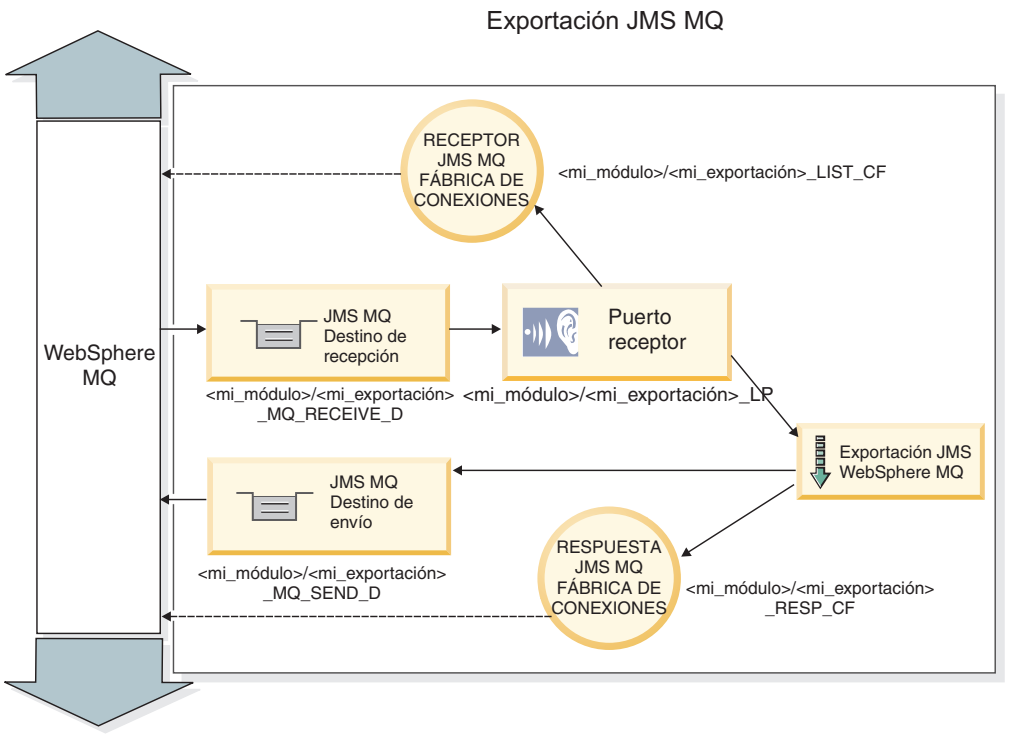


Figura 39. Recursos de enlace de exportación de JMS de WebSphere MQ

Nota: La Figura 38 en la página 94 y la Figura 39 ilustran cómo una aplicación de una versión anterior de WebSphere Process Server se enlaza a un servicio externo. Para aplicaciones desarrolladas para WebSphere Process Server Versión 7.0, se utiliza la Especificación de activación en lugar del Puerto receptor y de la Fábrica de conexiones.

Características clave de los enlaces JMS de WebSphere MQ

Entre las características clave de un enlace JMS de WebSphere MQ se incluyen las cabeceras, los artefactos Java EE y los recursos Java EE creados.

Cabeceras

Una cabecera de mensaje JMS contiene varios campos predefinidos que incluyen valores que utilizan los clientes y los proveedores para identificar y direccionar mensajes. Pueden usarse propiedades de enlace para configurar estas cabeceras con valores fijos o se pueden especificar dinámicamente en tiempo de ejecución.

JMSCorrelationID
Enlaza con un mensaje relacionado. Normalmente, este campo se establece en la serie del identificador del mensaje al que se está respondiendo.

TargetFunctionName
Esta cabecera la utiliza uno de los selectores de función para identificar la operación que se está invocando. Establecer la propiedad de cabecera JMS TargetFunctionName en un mensaje enviado a una exportación JMS permite utilizar este selector de función. La propiedad se puede establecer directamente en aplicaciones cliente de JMS o al conectar una importación con un JMS enlazando a esa exportación. En este caso, el enlace de importación de JMS debe configurarse para establecer la cabecera TargetFunctionName de cada operación de la interfaz en el nombre de la operación.

Esquemas de correlación

Los enlaces JMS de WebSphere MQ proporcionan varios esquemas de correlación que se utilizan para determinar cómo correlacionar los mensajes de petición con los mensajes de respuesta.

RequestMsgIDToCorrelID
El JMSMessageID se copia en el campo JMSCorrelationID. Éste es el valor por omisión.

RequestCorrelIDToCorrelID
El JMSCorrelationID se copia en el campo JMSCorrelationID.

Recursos Java EE

Se crea una serie de recursos Java EE cuando se despliega una importación de JMS de MQ en un entorno Java EE.

Parámetros

Fábrica de conexiones MQ
Utilizada por los clientes para crear una conexión con el proveedor de JMS de MQ.

Fábrica de conexiones de respuesta
Utilizada por el tiempo de ejecución JMS MQ SCA cuando el destino de envío está en un gestor de colas diferente que el destino de recepción.

Especificación de activación
Una especificación de activación JMS se asocia con uno o más beans controlados por mensajes y proporciona la configuración necesaria para que ellos reciban mensajes.

Destinos

- Destino de envío:
  - Importaciones: donde se envía la petición o el mensaje de salida.
  - Exportaciones: donde se enviará el mensaje de respuesta, si no se reemplaza por el campo de cabecera JMSReplyTo en el mensaje de entrada.
- Destino de recepción:
  - Importaciones: donde se debe colocar la respuesta o el mensaje de entrada.
  - En las exportaciones, donde se debe colocar el mensaje de entrada o de petición.

Cabeceras JMS

Un mensaje JMS contiene dos tipos de cabeceras: la cabecera del sistema JMS y varias propiedades JMS. Se puede acceder a ambos tipos de cabeceras, bien en un módulo de mediación del objeto de mensajes de servicios (SMO), o bien mediante la API ContextService.

Cabecera del sistema JMS

La cabecera del sistema JMS se representa en el SMO mediante el elemento JMSHeader, que contiene todos los campos que se encuentran, normalmente, en una cabecera JMS. Aunque éstos pueden modificarse en la mediación (o en ContextService), algunos campos de cabecera del sistema JMS establecidos en el

SMO no se propagarán en el mensaje JMS de salida, ya que el sistema, o los valores estáticos, los alteran temporalmente.

Los campos clave de la cabecera del sistema JMS que se pueden actualizar en la mediación (o en ContextService) son:

- **JMSType** y **JMSCorrelationID**: valores de las propiedades de la cabecera del mensaje predefinidas específicas.
- **JMSDeliveryMode**: valores de la modalidad de entrega (persistentes o no persistentes). El valor por omisión es persistente).
- **JMSPriority**: valor de prioridad (0 a 9). El valor por omisión es JMS_Default_Priority).

Propiedades JMS

Las propiedades JMS se representan en el SMO como entradas de la lista Propiedades. Las propiedades se pueden añadir, actualizar o suprimir dentro de una mediación, o bien utilizando la API ContextService.

Las propiedades también pueden establecerse estáticamente en el enlace JMS. Las propiedades que se establecen estáticamente alteran temporalmente los valores (que tengan el mismo nombre) que se establecen dinámicamente.

Las propiedades de usuario propagadas desde otros enlaces (por ejemplo, un enlace HTTP) generarán una salida en el enlace JMS en forma de propiedades JMS.

Valores de propagación de cabecera

La propagación de las propiedades y la cabecera del sistema JMS bien sea a desde el mensaje JMS de entrada a los componentes ubicados en sentido descendente, o bien sea desde los componentes ubicados en sentido ascendente al mensaje JMS de salida, se pueden controlar mediante el distintivo Propagate Protocol Header del enlace.

Cuando se establece Propagate Protocol Header, la información de cabecera puede fluir al mensaje o al componente de destino, tal como se describe en la lista siguiente:

- Petición de exportación JMS

  La cabecera JMS recibida en el mensaje se propagará a los componentes de destino por medio del servicio de contexto. Las propiedades JMS recibidas en el mensaje se propagará a los componentes de destino por medio del servicio de contexto.
- Respuesta de exportación JMS

  Cualquiera de los campos de cabecera JMS establecidos en el servicio de contexto se utilizará en el mensaje de salida, si no lo han alterado temporalmente las propiedades establecidas en el enlace de exportación de JMS. Cualquiera de las propiedades establecidas en el servicio de contexto se utilizará en el mensaje de salida, si no la han alterado temporalmente las propiedades establecidas en el enlace de exportación de JMS.
- Petición de importación JMS

  Cualquiera de los campos de cabecera JMS establecidos en el servicio de contexto se utilizará en el mensaje de salida, si no lo han alterado temporalmente las propiedades establecidas en el enlace de importación de JMS. Cualquiera de las propiedades establecidas en el servicio de contexto se utilizará

en el mensaje de salida, si no la han alterado temporalmente las propiedades establecidas en el enlace de importación de JMS.

- Respuesta de importación JMS

  La cabecera JMS recibida en el mensaje se propagará a los componentes de destino por medio del servicio de contexto. Las propiedades JMS recibidas en el mensaje se propagará a los componentes de destino por medio del servicio de contexto.

Clientes externos

El servidor puede enviar mensajes o recibirlos de clientes externos utilizando los enlaces JMS de WebSphere MQ.

Un cliente externo (como un portal Web o un sistema de empresa) puede enviar un mensaje a un componente SCA de la aplicación mediante una exportación, o lo puede invocar un componente SCA de la aplicación mediante una importación.

El enlace de exportación JMS de WebSphere MQ despliega los beans controlados por mensajes (MDB) para escuchar las peticiones procedentes del destino especificado en receive especificado en el enlace de exportación. El destino especificado en el campo send se utiliza para enviar la respuesta a la petición de entrada si la aplicación invocada proporciona una respuesta. De esta forma, un cliente externo puede invocar aplicaciones a través del enlace de exportación.

Las importaciones de JMS de WebSphere MQ se enlazan con clientes externos y pueden entregar mensajes a éstos. Este mensaje puede necesitar o no una respuesta del cliente externo.

Puede encontrar más información sobre cómo interactuar con clientes externos utilizando WebSphere MQ en el Centro de información de WebSphere MQ.

Resolución de problemas de enlaces JMS de WebSphere MQ

Puede diagnosticar y solucionar problemas con los enlaces JMS de WebSphere MQ.

Excepciones de implementación

Como respuesta a las distintas condiciones de error, la implementación de la importación y exportación de JMS puede devolver uno de estos dos tipos de excepciones:

- Excepción empresarial de servicio: esta excepción se devuelve si se ha producido la excepción especificada de error en la interfaz empresarial de servicio (tipo de puerto WSDL).
- Excepción de tiempo de ejecución de servicio: se produce en todos los demás casos. En la mayoría de los casos, la excepción cause contendrá la excepción (JMSException) original.

  Por ejemplo, la importación espera sólo un mensaje de respuesta para cada mensaje de solicitud. Si llega más de una respuesta o si llega una respuesta con demora (una para la que se ha pasado el tiempo de caducidad de respuesta de SCA), se genera una excepción de tiempo de ejecución de servicio. Se retrotrae la transacción y el mensaje de respuesta se vuelve a poner fuera de la cola o lo gestiona el Gestor de sucesos con error.

Mensajes SCA basados en JMS de WebSphere MQ que no aparecen en el gestor de sucesos con error

Si los mensajes SCA originados en una interacción JMS de WebSphere MQ fallan, los mensajes deberán aparecer en el gestor de sucesos con error. Si los mensajes no aparecen en el gestor de sucesos con error, asegúrese de que el valor de la propiedad de máximo de reintentos en el puerto receptor subyacente sea mayor o igual que 1. Si establece este valor en 1 o más, se habilita la interacción con el gestor de sucesos con error durante las invocaciones de SCA para los mensajes JMS de MQ.

Escenarios de utilización errónea: comparación con los enlaces de WebSphere MQ

El enlace JMS de WebSphere MQ se ha diseñado para interoperar con las aplicaciones JMS desplegadas en WebSphere MQ, que expone los mensajes de acuerdo con el modelo de mensaje JMS. La importación y exportación de WebSphere MQ, sin embargo, están diseñadas principalmente para interoperar con las aplicaciones WebSphere MQ nativas y exponer el contenido completo del cuerpo del mensaje WebSphere MQ en las mediaciones.

Los escenarios siguientes se deben crear utilizando el enlace JMS de WebSphere MQ, no el enlace de WebSphere MQ:

- Invocación de un bean controlado por mensaje JMS (MDB) desde un módulo SCA, donde el MDB se despliega en el proveedor de JMS de WebSphere MQ. Utilice una importación de JMS de WebSphere MQ.
- Permitir que se invoque el módulo de SCA desde un servlet de componentes Java EE o desde EJB por medio de JMS. Utilice una exportación de JMS de WebSphere MQ.
- Mediación de los contenidos de un MapMessage JMS, en tránsito entre WebSphere MQ. Utilice una exportación y una importación de JMS de WebSphere MQ junto con el enlace de datos o manejador de datos adecuado.

Existen situaciones en las cuales es posible que se espera la interoperatividad entre el enlace WebSphere MQ y el enlace JMS de WebSphere MQ. En particular, cuando realiza un puente entre aplicaciones Java EE y no Java EE de WebSphere MQ, utilice una exportación de WebSphere MQ y una importación de JMS de WebSphere MQ (o viceversa) junto con los enlaces de datos apropiados o los módulos de mediación (o ambos).

Manejo de excepciones

El modo en que está configurado el enlace de datos determina cómo se tratan las excepciones generadas por los manejadores de datos o los enlaces de datos. Además, la naturaleza del flujo de mediación dicta el comportamiento del sistema cuando se genera una excepción de este estilo.

Pueden surgir varios problemas cuando su enlace llama a un manejador de datos o enlace de datos. Por ejemplo, un manejador de datos puede recibir un mensaje que tiene una carga útil dañada, o puede tratar de leer un mensaje que tenga un formato incorrecto.

El modo en que el enlace gestiona una excepción de este estilo viene determinado por el modo de implementar el manejador de datos o el enlace de datos. El comportamiento recomendado es diseñar su propio enlace de datos para producir una excepción `DataBindingException`.

La situación es similar para un manejador de datos. Dado que el enlace de datos invoca el manejador de datos, cualquier excepción del manejador de datos se incluye en una excepción de enlace de datos. Por lo tanto, se le notifica una excepción `DataHandlerException` como una `DataBindingException`.

Cuando se produce cualquier excepción de tiempo de ejecución, incluida la excepción `DataBindingException`:

- Si el flujo de mediación se ha configurado para ser transaccional, el mensaje JMS se almacena en el gestor de sucesos anómalos de forma predeterminada para una respuesta o supresión manual.

  Nota: Puede cambiar la modalidad de recuperación en el enlace, de forma que el mensaje se retrotrae, en lugar de almacenarse en el gestor de sucesos anómalos.
- Si el flujo de mediación no es transaccional, el flujo de mediación se anota y se pierde el mensaje.

La situación es similar para un manejador de datos. Puesto que el enlace de datos llama al manejador de datos, se produce una excepción de manejador de datos dentro de una excepción de enlace de datos. Por lo tanto, se informa de una excepción `DataHandlerException` como una `DataBindingException`.

Enlaces de WebSphere MQ

El enlace de WebSphere MQ proporciona la conectividad SCA (Service Component Architecture) con las aplicaciones WebSphere MQ.

Utilice los enlaces de exportación e importación de WebSphere MQ para integrarse directamente con un sistema basado en WebSphere MQ desde el entorno del servidor. Esto elimina la necesidad de utilizar las características de Enlace MQ o Enlace de cliente de Service Integration Bus.

Cuando un componente interactúa con un servicio de WebSphere MQ a través de una importación, el enlace de importación de WebSphere MQ utiliza una cola a la que se enviarán los datos y una cola donde se puede recibir la respuesta.

Cuando un módulo SCA proporciona un servicio a clientes de WebSphere MQ, el enlace de exportación de WebSphere MQ utiliza una cola donde se puede recibir la petición y se puede enviar la respuesta. El selector de función proporciona una correlación con la operación del componente de destino que se va a invocar.

La conversión de los datos de carga útil a y desde un mensaje MQ se realiza mediante el manejador de datos del cuerpo MQ o el enlace de datos. La conversión de los datos de la cabecera a y desde un mensaje MQ se realiza mediante el enlace de datos de la cabecera MQ.

Para obtener información sobre las versiones soportadas de WebSphere MQ, consulte el sitio Web de requisitos del sistema de WebSphere Process Server.

Visión general de enlaces WebSphere MQ

El enlace de WebSphere MQ proporciona integración con las aplicaciones basadas en MQ nativo.

Tareas administrativas de WebSphere MQ

El administrador del sistema WebSphere MQ debe crear el gestor de colas WebSphere MQ subyacente, que utilizarán los enlaces de WebSphere MQ antes de

ejecutar una aplicación que contenga estos enlaces.

Tareas administrativas de WebSphere

Debe establecer la propiedad **Vía de acceso de biblioteca nativa** del adaptador de recursos MQ en Websphere en la versión de WebSphere MQ soportada por el servidor y reinicie el servidor. Esto garantiza que se utilicen las bibliotecas de una versión soportada de WebSphere MQ. Puede encontrar los requisitos detallados de hardware y software en las páginas de soporte de IBM.

Enlaces de importación WebSphere MQ

El enlace de importación de WebSphere MQ permite a los componentes del módulo SCA comunicarse con los servicios proporcionados por las aplicaciones externas basadas en WebSphere MQ. Debe utilizar una versión soportada de WebSphere MQ. Puede encontrar los requisitos detallados de hardware y software en las páginas de soporte de IBM.

La interacción con los sistemas WebSphere MQ externos incluye la utilización de colas para enviar peticiones y recibir respuestas.

Se da soporte a dos tipos de escenarios de uso para los enlaces de importación de WebSphere MQ, dependiendo del tipo de operación que se invoque:

- Unidireccional: la importación de WebSphere MQ coloca un mensaje en la cola configurada en el campo **Cola de destino de envío** del enlace de importación. No se envía nada al campo replyTo de la cabecera MQMD.
- Bidireccional (petición y respuesta): la importación de WebSphere MQ coloca un mensaje en la cola configurada en el campo **Cola de destino de envío**.

  La cola de recepción se establece en el campo de la cabecera MQMD replyTo. Se despliega un bean MDB (Message Driven Bean) para escuchar en la cola de recepción y cuando se recibe una respuesta, el MDB vuelve a pasar la respuesta al componente.

  El enlace de importación se puede configurar (mediante el campo **Esquema de correlación de respuesta** para esperar a que el ID de correlación de mensaje de respuesta se haya copiado del ID mensaje de petición (el valor por omisión) o del ID de correlación de mensaje de petición.

Resulta importante tener en cuenta que WebSphere MQ es un enlace asíncrono. Si un componente llamante invoca una importación de WebSphere MQ de forma síncrona (para una operación bidireccional), el componente llamante se bloquea hasta que el servicio WebSphere MQ devuelve la respuesta.

Figura 40 en la página 102 ilustra cómo se enlaza la importación con el servicio externo.

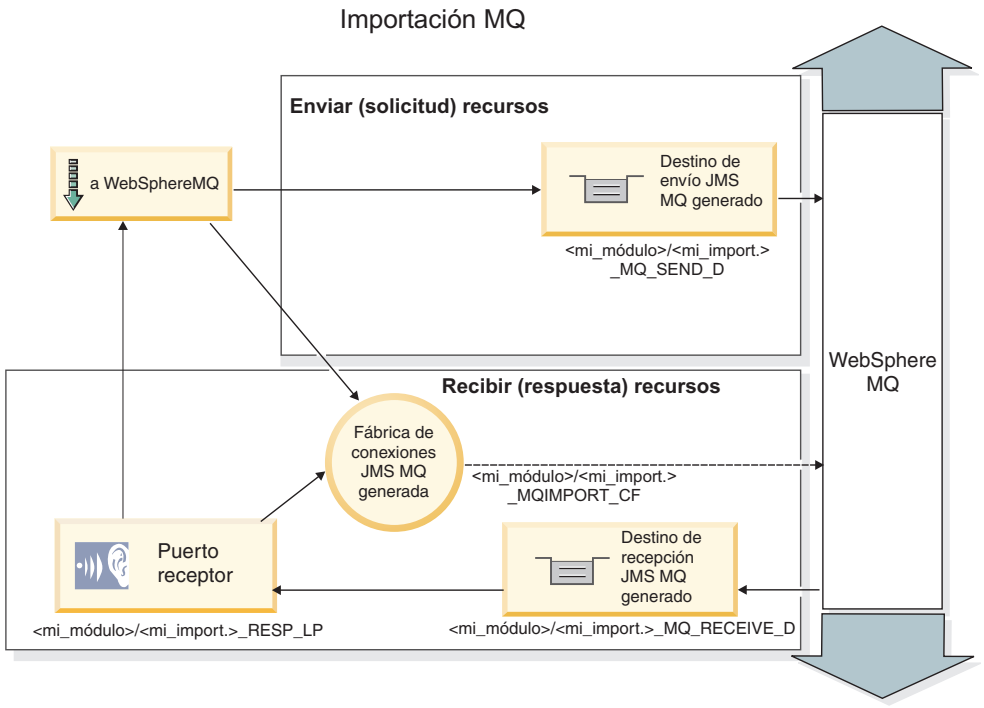


Figura 40. Recursos de enlace de importación de WebSphere MQ

Enlaces de exportación de WebSphere MQ

El enlace de exportación de WebSphere MQ proporciona el medio mediante el cual los módulos SCA pueden ofrecer servicios a aplicaciones externas basadas en WebSphere MQ.

Se despliega un MDB para que escuche las peticiones de entrada en la **cola de destino de recepción** especificada en el enlace de exportación. La cola especificada en el campo **Cola de destino de envío** se utiliza para enviar la respuesta a la petición de entrada si el componente invocado proporciona una respuesta. La cola especificada en el campo replyTo del mensaje de respuesta altera temporalmente la cola especificada en el campo **Cola de destino de envío**.

Figura 41 en la página 103 ilustra cómo se enlaza el solicitante externo con la exportación.

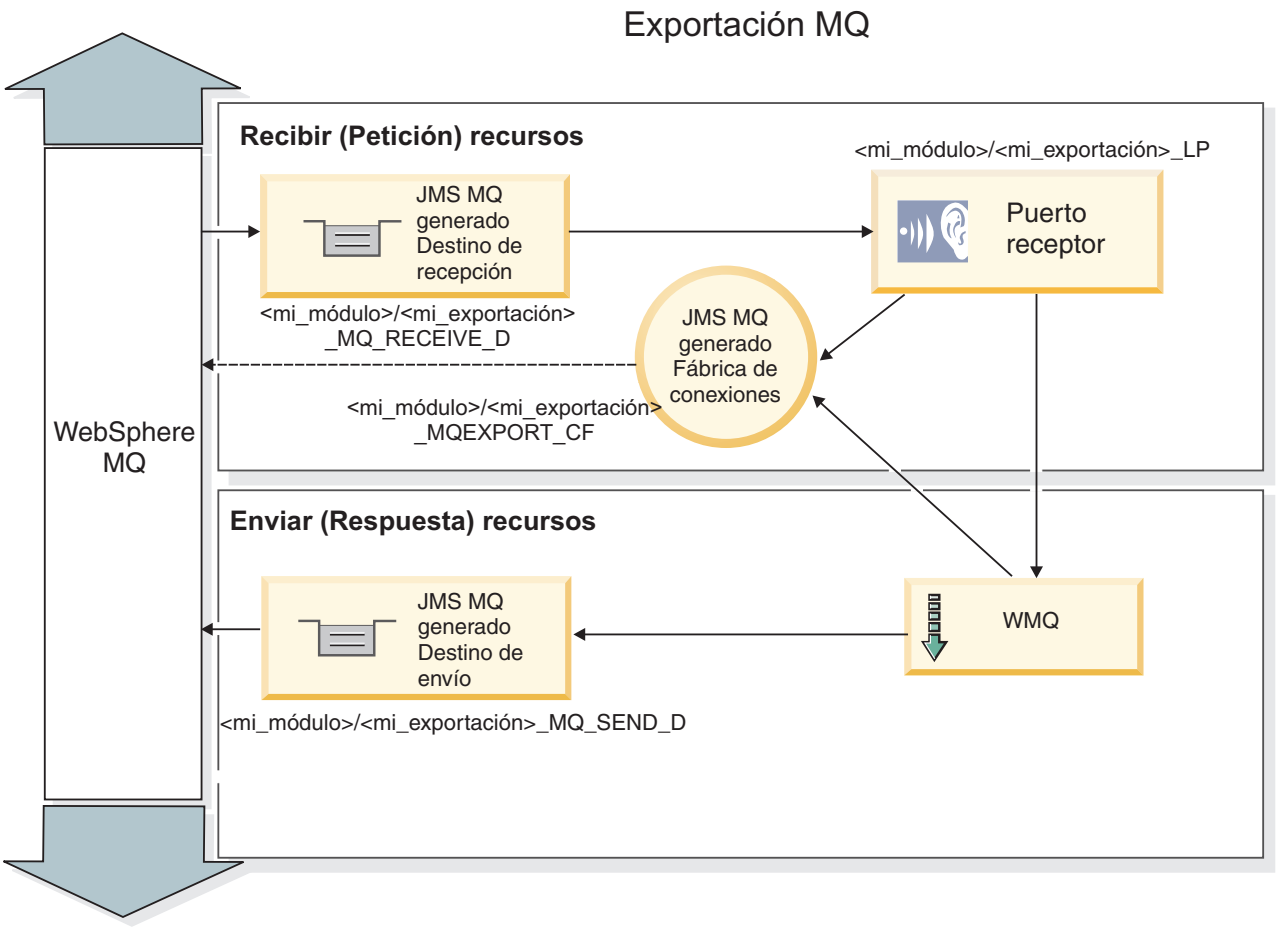


Figura 41. Recursos de enlace de exportación de WebSphere MQ

Nota: La Figura 40 en la página 102 y la Figura 41 ilustran cómo una aplicación de una versión anterior de WebSphere Process Server se enlaza a un servicio externo. Para aplicaciones desarrolladas para WebSphere Process Server Versión 7.0, se utiliza la Especificación de activación en lugar del Puerto receptor y de la Fábrica de conexiones.

Características clave de un enlace WebSphere MQ

Entre las características clave de un enlace WebSphere MQ se incluyen las cabeceras, los artefactos Java EE y los recursos Java EE creados.

Esquemas de correlación

Una aplicación de petición/respuesta de WebSphere MQ puede utilizar una de las diferentes técnicas para correlacionar los mensajes de respuesta con las peticiones, creadas alrededor de los campos de MQMD MessageID y CorrelID. En la gran mayoría de los casos, el solicitante permite que el gestor de colas seleccione un MessageID y espera que la aplicación de respuesta lo copie en el CorrelID de la respuesta. En muchos casos, la aplicación de petición y de respuesta sabe de forma implícita qué técnica de correlación se ha de utilizar. Ocasionalmente, la aplicación de respuesta reconocerá los diferentes distintivos del campo Report de la petición que describe cómo se manejan estos campos.

Los enlaces de exportación para los mensajes WebSphere MQ se pueden configurar con las opciones siguientes:

Opciones del MsgId de respuesta:

Nuevo MsgID
Permite al gestor de colas seleccionar un MsgId exclusivo para la respuesta (valor por omisión).

Copiar de MsgID de petición
Copia el campo MsgId desde el campo MsgId de la petición.

Copiar del mensaje SCA
Establece el MsgId que se ha de transportar en las cabeceras de WebSphere MQ en el mensaje de respuesta de SCA o deja que el gestor de colas defina un nuevo Id si el valor no existe.

Opciones como Informe
Inspecciona el campo Report del MQMD de la petición para obtener una sugerencia acerca de cómo manejar MsgId. Las opciones MQRO_NEW_MSG_ID y MQRO_PASS_MSG_ID están soportadas y se comportan como Nuevo MsgId y Copiar de MsgID de petición, respectivamente.

Opciones del CorrelId de respuesta:

Copiar de MsgID de petición
Copia el campo CorrelId desde el campo MsgId de la petición (valor por omisión).

Copiar de CorrelID de petición
Copia el campo CorrelId desde el campo CorrelId de la petición.

Copiar del mensaje SCA
Establece el CorrelId que se ha de transportar en las cabeceras de WebSphere MQ en el mensaje de respuesta SCA o lo deja en blanco si el valor no existe.

Opciones como Informe
Inspecciona el campo Report del MQMD de la petición para obtener una sugerencia acerca de cómo manejar CorrelId. Las opciones MQRO_COPY_MSG_ID_TO_CORREL_ID y MQRO_PASS_CORREL_ID están soportadas y se comportan como Copiar desde y Copiar de CorrelID de petición, respectivamente.

Los enlaces de importación para los mensajes WebSphere MQ se pueden configurar con las opciones siguientes:

Opciones del MsgId de respuesta:

Nuevo MsgID
Permite al gestor de colas seleccionar un MsgId exclusivo para la petición (valor por omisión).

Copiar del mensaje SCA
Establece el MsgId que se ha de transportar en las cabeceras de WebSphere MQ en el mensaje de solicitud de SCA o deja que el gestor de colas defina un nuevo Id si el valor no existe.

Opciones del correlación de respuesta:

La respuesta tiene el CorrelID copiado de MsgId
Espera que el mensaje de respuesta tenga un campo CorrelId establecido como el MsgId de la petición (por omisión).

La respuesta tiene el MsgID copiado de MsgId
Espera que el mensaje de respuesta tenga un campo MsgId establecido como el MsgId de la petición.

La respuesta tiene el CorrelID copiado de CorrelId
Espera que el mensaje de respuesta tenga un campo CorrelId establecido como el CorrelId de la petición.

Recursos Java EE

Se crea una serie de recursos Java EE cuando se despliega un enlace WebSphere MQ en un entorno Java EE.

Parámetros

Fábrica de conexiones MQ
Utilizada por los clientes para crear una conexión con el proveedor MQ de WebSphere.

Fábrica de conexiones de respuesta
Utilizada por el tiempo de ejecución SCA de MQ cuando el destino de envío está en un gestor de colas diferente que el destino de recepción.

Especificación de activación
Una especificación de activación JMS se asocia con uno o más beans controlados por mensajes y proporciona la configuración necesaria para que ellos reciban mensajes.

Destinos

- Destino de envío: donde se envía la petición o el mensaje de salida (importación); donde se enviará el mensaje de respuesta (exportación), si no se reemplaza con el campo de cabecera ReplyTo en el mensaje de entrada.
- Destino de recepción: donde se debe colocar el mensaje de respuesta/petición o de entrada.

Cabeceras de WebSphere MQ

Las cabeceras WebSphere MQ incorporan determinados convenios para la conversión de mensajes SCA (Service Component Architecture).

Un mensaje WebSphere MQ está formado por una cabecera de sistema (el MQMD), cero o más cabeceras MQ adicionales (del sistema o personalizadas), y un cuerpo de mensaje. Si existen varias cabeceras de mensajes en el mensaje, el orden de las cabeceras es importante.

Cada cabecera contiene información que describe la estructura de la siguiente cabecera. El MQMD describe la primera cabecera.

Cómo se analizan las cabeceras MQ

Para analizar las cabeceras MQ se utiliza un enlace de datos de cabecera MQ. Se da soporte, de forma automática, a las cabeceras siguientes:

- MQRFH
- MQRFH2
- MQCIH
- MQIIH

Las cabeceras que empiezan con MQH se manejan de forma distinta. Los campos específicos de la cabecera no se analizan; permanecen como bytes no analizados.

Para otras cabeceras MQ, puede escribir enlaces de datos de cabecera MQ personalizados para analizar aquéllas.

Cómo se accede a las cabeceras MQ

Se puede acceder a las cabeceras MQ del producto en una de dos maneras:

- A través del objeto de mensaje de servicio (SMO) en una mediación.
- A través de la API ContextService.

Las cabeceras MQ se representan internamente con el elemento MQHeader de SMO. MQHeader es un contenedor de los datos de cabecera que amplía MQControl pero que contiene un elemento de valor de anyType. Contiene el MQMD, MQControl (la información de control del cuerpo del mensaje MQ), y una lista de otras cabeceras MQ.

- MQMD representa los contenidos de la descripción del mensaje WebSphere MQ, excepto para la información que determina la estructura y la codificación del cuerpo.
- MQControl contiene información que determina la estructura y la codificación de un cuerpo de mensaje.
- Las MQHeader contienen una lista de objetos MQHeader.

La cadena de la cabecera MQ se despliega de forma que, dentro del SMO, cada cabecera MQ incluye su propia información de control (CCSID, codificación y formato). Las cabeceras pueden añadirse o suprimirse fácilmente, sin por ello alterar otros datos de la cabecera.

Definición de campos en MQMD

Puede actualizar el MQMD utilizando la API del contexto o a través del objeto de mensaje de servicio (SMO) en una mediación. Los campos siguientes se propagan automáticamente en el mensaje MQ de salida:

- Cifrado
- Juego de caracteres codificado
- Formato
- Notificar
- Caducidad
- Comentarios
- Prioridad
- Persistencia
- ID de correlación
- Indicadores de mensajes

Configure el enlace MQ en una Importación o Exportación para propagar las siguientes propiedades en el mensaje MQ de salida:

MsgID
Establezca el **ID de mensaje de solicitud** en copiar desde el mensaje SCA.

MsgType
Desactive el recuadro de selección **Establecer tipo de mensaje en MQMT_DATAGRAM o MQMT_REQUEST para la operación petición-respuesta**.

ReplyToQ
Desactive el recuadro de selección **Alterar temporalmente la respuesta de una cola del mensaje de solicitud**.

ReplyToQMgr
Desactive el recuadro de selección **Alterar temporalmente la respuesta de una cola del mensaje de solicitud**.

Desde la versión 7.0 en adelante, los campos de contexto se pueden alterar temporalmente utilizando una propiedad personalizada en la definición del destino JMS. Establezca la propiedad personalizada MDCTX con el valor `SET_IDENTITY_CONTEXT` en el destino de envío para propagar los campos siguientes en el mensaje MQ de salida:

- UserIdentifier
- AppIdentityData

Establezca la propiedad personalizada MDCTX con el valor `SET_ALL_CONTEXT` en el destino de envío para propagar las siguientes propiedades en el mensaje MQ de salida:

- UserIdentifier
- AppIdentityData
- PutApplType
- PutApplName
- ApplOriginData

Algunos campos no se propagan en el mensaje MQ de salida. Los campos siguientes se alteran temporalmente durante el envío del mensaje:

- BackoutCount
- AccountingToken
- PutDate
- PutTime
- Offset
- OriginalLength

Adición estática de MQCIH a un enlace WebSphere MQ

WebSphere Process Server soporta la adición de datos de cabecera MQCIH de forma estática sin utilizar un módulo de mediación.

Hay varias maneras de añadir información de cabecera MQCIH a un mensaje (por ejemplo, utilizando el primitivo de mediación del método setter de cabecera). Es posible que resulte útil añadir esta información de cabecera estáticamente, sin utilizar un módulo de mediación adicional. La información de cabecera estática, incluido el nombre de programa CICS, el ID de transacción y otros detalles de cabecera de formato de datos, se puede definir y añadir como parte del Enlace de WebSphere MQ.

WebSphere MQ, el puente MQ CICS y CICS deben estar configurados para que se añada estáticamente los datos de cabecera de MQCIH.

Puede utilizar WebSphere Integration Developer para configurar la importación de WebSphere MQ con valores estáticos que son necesarios para la información de cabecera MQCIH.

Cuando llega un mensaje y lo procesa la importación de WebSphere MQ, se realiza una comprobación para ver si la información de cabecera MQCIH ya está presente en el mensaje. Si MQCIH existe, se utilizan los valores estáticos definidos en la importación de WebSphere MQ para alterar temporalmente los valores dinámicos correspondientes en el mensaje. Si MQCIH no existe, se crea una en el mensaje y se añaden los valores estáticos definidos en la importación de WebSphere MQ.

Los valores estáticos definidos en la importación de WebSphere MQ son específicos de un método. Puede especificar diferentes valores MQCIH estáticos para diferentes métodos en la misma importación de WebSphere MQ.

Este recurso no se utiliza para proporcionar valores por omisión si MQCIH no contiene información de cabecera específica porque un valor estático definido en la importación de WebSphere MQ alterará temporalmente un valor correspondiente proporcionado en el mensaje de entrada.

Clientes externos

WebSphere Process Server puede enviar mensajes o recibirlos de clientes externos mediante los enlaces WebSphere MQ.

Un cliente externo (por ejemplo, un portal Web o un sistema de empresa) puede enviar un mensaje a un componente SCA de la aplicación mediante una exportación, o lo puede invocar un componente SCA de la aplicación mediante una importación.

El enlace de exportación de WebSphere MQ despliega los beans controlados por mensajes (MDB) para escuchar las peticiones procedentes del destino especificado en receive especificado en el enlace de exportación. El destino especificado en el campo send se utiliza para enviar la respuesta a la petición de entrada si la aplicación invocada proporciona una respuesta. De esta forma, un cliente externo puede invocar aplicaciones a través del enlace de exportación.

Las importaciones WebSphere MQ se enlazan con clientes externos y pueden entregar mensajes a éstos. Este mensaje puede necesitar o no una respuesta del cliente externo.

Puede encontrar más información sobre cómo interactuar con clientes externos utilizando WebSphere MQ en el Centro de información de WebSphere MQ.

Resolución de problemas de enlaces WebSphere MQ

Puede diagnosticar y solucionar las anomalías y las condiciones de error que se producen con los enlaces WebSphere MQ.

Condiciones de error principales

Las condiciones de error principales de los enlaces WebSphere MQ quedan determinadas por la semántica de transacciones, por la configuración de WebSphere MQ o por la referencia a un comportamiento existente de otros componentes. Las primeras condiciones de error son:

- No se puede conectar a la cola o al gestor de colas de WebSphere MQ.

  Si no se puede conectar a WebSphere MQ para recibir mensajes, no se podrá iniciar el puerto receptor MDB. Esta condición se anotará cronológicamente en

las anotaciones cronológicas de WebSphere Application Server. Los mensajes persistentes permanecerán en la cola de WebSphere MQ hasta que se hayan recuperado correctamente (o hayan caducado mediante WebSphere MQ).

Si no se puede conectar a WebSphere MQ, para enviar mensajes de salida, se provocará la retrotracción de la transacción que controla el envío.

- No se puede analizar un mensaje de entrada o construir un mensaje de salida.

  Una anomalía en el enlace de datos provoca la retrotracción de la transacción que controla el trabajo.
- No se puede enviar el mensaje de salida.

  Una anomalía al enviar un mensaje provoca una retrotracción de la transacción correspondiente.
- Varios mensajes de respuesta o mensajes inesperados.

  La importación espera sólo un mensaje de respuesta para cada mensaje de solicitud. Si llega más de una respuesta o si llega una respuesta con demora (una para la que se ha pasado el tiempo de caducidad de respuesta de SCA), se genera una excepción en tiempo de ejecución de servicio. Se retrotrae la transacción y el mensaje de respuesta se vuelve a poner fuera de la cola o lo gestiona el Gestor de sucesos anómalos.

Escenarios de utilización incorrecta: comparación con los enlaces de JMS de WebSphere MQ

La importación y exportación de WebSphere MQ se han diseñado principalmente para interoperar con aplicaciones WebSphere MQ nativas, y exponer el contenido completo del cuerpo del mensaje WebSphere MQ en las mediaciones. El enlace de JMS de WebSphere MQ, sin embargo, se ha diseñado para interoperar con las aplicaciones JMS desplegadas en WebSphere MQ, que expone los mensajes de acuerdo con el modelo de mensaje JMS.

Los escenarios siguientes se deben crear utilizando el enlace JMS de WebSphere MQ, no el enlace de WebSphere MQ:

- Invocación de un bean controlado por mensaje JMS (MDB) desde un módulo de SCA, donde el MDB se despliega en el proveedor de JMS de WebSphere MQ. Utilice una importación de JMS de WebSphere MQ.
- Permitir que se invoque el módulo de SCA desde un servlet de componentes Java EE o desde EJB por medio de JMS. Utilice una exportación de JMS de WebSphere MQ.
- Mediación de los contenidos de un MapMessage JMS, en tránsito entre WebSphere MQ. Utilice una exportación y una importación de JMS de WebSphere MQ junto con el enlace de datos adecuado.

Existen situaciones en las cuales es posible que se espera la interoperatividad entre el enlace WebSphere MQ y el enlace JMS de WebSphere MQ. En particular, cuando realiza un puente entre aplicaciones Java EE y no Java EE de WebSphere MQ, utilice una exportación de WebSphere MQ y una importación de JMS de WebSphere MQ (o viceversa) junto con los enlaces de datos apropiados o los módulos de mediación (o ambos).

Mensajes sin entregar

Si WebSphere MQ no puede entregar un mensaje a su destino previsto (por ejemplo, debido a errores de configuración), envía los mensajes en su lugar a una cola de mensajes no entregados nominada.

Al hacerlo, se añade una cabecera de mensaje en espera al inicio del cuerpo del mensaje. Esta cabecera contiene las razones de la anomalía, el destino original y otra información.

Mensajes SCA basados en MQ que no aparecen en el gestor de sucesos con anomalía

Si los mensajes SCA originados debido a una anomalía en una interacción WebSphere MQ, los mensajes deberán aparecer en el gestor de sucesos con anomalía. Si dichos mensajes no aparecen en el gestor de sucesos con anomalía, asegúrese de que el destino WebSphere MQ tenga un valor máximo de entregas con error mayor que 1. Si establece este valor en 2 o más, se permite la interacción con el gestor de sucesos con anomalía durante las invocaciones de SCA para los mensajes WebSphere.

Los sucesos anómalos MQ se repiten en el gestor de colas erróneo

Cuando una fábrica de conexiones predefinida se va a utilizar para conexiones de salida, las propiedades de la conexión deben coincidir con las definidas en la especificación de activación utilizada para las conexiones de entrada.

La fábrica de conexiones predefinida se utiliza para crear una conexión al reproducir un suceso anómalo y, por lo tanto, se debe configurar para utilizar el mismo gestor de colas desde el que se ha recibido originalmente el mensaje.

Manejo de excepciones

El modo en que está configurado el enlace de datos determina cómo se tratan las excepciones generadas por los manejadores de datos o los enlaces de datos. Además, la naturaleza del flujo de mediación dicta el comportamiento del sistema cuando se genera una excepción de este estilo.

Pueden surgir varios problemas cuando su enlace llama a un manejador de datos o enlace de datos. Por ejemplo, un manejador de datos puede recibir un mensaje que tiene una carga útil dañada, o puede tratar de leer un mensaje que tenga un formato incorrecto.

El modo en que el enlace gestiona una excepción de este estilo viene determinado por el modo de implementar el manejador de datos o el enlace de datos. El comportamiento recomendado es diseñar su propio enlace de datos para producir una excepción `DataBindingException`.

La situación es similar para un manejador de datos. Dado que el enlace de datos invoca el manejador de datos, cualquier excepción del manejador de datos se incluye en una excepción de enlace de datos. Por lo tanto, se le notifica una excepción `DataHandlerException` como una `DataBindingException`.

Cuando se produce cualquier excepción de tiempo de ejecución, incluida la excepción `DataBindingException`:

- Si el flujo de mediación se ha configurado para ser transaccional, el mensaje JMS se almacena en el gestor de sucesos anómalos de forma predeterminada para una respuesta o supresión manual.

  Nota: Puede cambiar la modalidad de recuperación en el enlace, de forma que el mensaje se retrotrae, en lugar de almacenarse en el gestor de sucesos anómalos.

- Si el flujo de mediación no es transaccional, el flujo de mediación se anota y se pierde el mensaje.

La situación es similar para un manejador de datos. Puesto que el enlace de datos llama al manejador de datos, se produce una excepción de manejador de datos dentro de una excepción de enlace de datos. Por lo tanto, se informa de una excepción `DataHandlerException` como una `DataBindingException`.

Limitaciones de los enlaces

Los enlaces tienen algunas limitaciones por lo que respecta a su uso que se listan aquí.

Limitaciones del enlace MQ

El enlace MQ tiene algunas limitaciones por lo que respecta a su uso que se listan aquí.

Sin distribución de mensajes publicación-suscripción

El enlace MQ no admite actualmente el método de publicación-suscripción para la distribución de mensajes aunque WMQ admite el método de publicación-suscripción. No obstante, el enlace JMS de MQ sí que admite este método de distribución.

Colas de recepción compartidas

Varios enlaces de exportación e importación WebSphere MQ esperan que cualquier mensaje presente en su cola de recepción configurada tenga como objetivo dicha exportación o importación. Los enlaces de importación y exportación se deben configurar con las siguientes consideraciones:

- Cada importación MQ debe tener una cola de recepción diferente porque el enlace de importación MQ asume que todos los mensajes de la cola de recepción son respuestas a peticiones que ha enviado. Si más de una importación comparte la cola de recepción, las respuestas podrían ser recibidas por la importación errónea y no se podrán correlacionar con el mensaje de solicitud original.
- Cada exportación MQ debe tener una cola de recepción diferente, porque, de lo contrario, no puede predecir qué exportación obtendrá cualquier mensaje de solicitud.
- Las importaciones y exportaciones MQ pueden señalar la misma cola de envío.

Limitaciones de los enlaces JMS, MQ JMS y JMS genéricos

Los enlaces JMS y MQ JMS tienen algunas limitaciones.

Implicaciones de la generación de enlaces predeterminados

Las limitaciones de uso de los enlaces JMS, JMS MQ y JMS genéricos se exponen en las secciones siguientes:

- Implicaciones de la generación de enlaces predeterminados
- Esquema de correlación de respuesta
- Soporte bidireccional

Cuando genere un enlace, se llenarán varios campos automáticamente como valores predeterminados, si no decide especificar los valores usted mismo. Por ejemplo, se creará automáticamente el nombre de una fábrica de conexiones. Si sabe que pondrá su aplicación en un servidor y accederá a ella remotamente con un cliente, al crear el enlace debería especificar nombres JNDI en lugar de aceptar

los valores predeterminados, ya que probablemente querrá controlar estos valores mediante la consola de administración en tiempo de ejecución.

No obstante, si ha aceptado los valores predeterminados y luego se encuentra con que no puede acceder a su aplicación desde un cliente remoto, puede utilizar la consola de administración para definir explícitamente el valor de la fábrica de conexiones. Localice el campo de puntos finales del proveedor en los valores de la fábrica de conexiones y añada un valor como, por ejemplo <nombre_host_servidor>:7276 (si utiliza el número de puerto predeterminado).

Esquema de correlación de respuesta

Si utiliza el esquema de correlación de respuesta de ID de correlación a ID de correlación, que se utiliza para correlacionar mensajes en una operación de solicitud-respuesta, debe disponer de un ID de correlación dinámico en el mensaje.

Para crear un ID de correlación dinámico en un módulo de mediación utilizando el editor de flujos de mediación, añada un nodo XSLT antes de realizar la importación con el enlace JMS. Abra el editor de correlaciones XSLT. Las cabeceras de arquitectura de componentes de servicio conocidos estarán disponibles en el mensaje de destino. Arrastre y suelte un campo que contenga un ID exclusivo en el mensaje de origen al ID de correlación de la cabecera JMS del mensaje de destino.

Soporte bidireccional

Java Naming and Directory Interface (JNDI) sólo admite caracteres ASCII para los nombres en tiempo de ejecución.

Colas de recepción compartidas

Varios enlaces de exportación e importación esperan que cualquier mensaje presente en su cola de recepción configurada tenga como objetivo dicha exportación o importación. Los enlaces de importación y exportación se deben configurar con las siguientes consideraciones:

- Cada enlace de importación debe tener una cola de recepción diferente porque el enlace de importación asume que todos los mensajes de la cola de recepción son respuestas a las peticiones que ha enviado. Si más de una importación comparte la cola de recepción, las respuestas podrían ser recibidas por la importación errónea y no se podrán correlacionar con el mensaje de solicitud original.
- Cada exportación debe tener una cola de recepción diferente, porque, de lo contrario, no puede predecir qué exportación obtendrá cualquier mensaje de solicitud.
- Las importaciones y exportaciones pueden señalar la misma cola de envío.

Capítulo 3. Guías y técnicas de programación

Esta sección incluye guías y ejemplos de programación.

Los subtemas siguientes proporcionan información para la programación de diversos componentes, aplicaciones y soluciones de integración empresarial.

Importante: Vea la sección de consulta del Centro de información para obtener detalles de las API (interfaces de programación de aplicaciones) y las SPI(interfaces de programación del sistema) admitidas por WebSphere Process Server y WebSphere Enterprise Service Bus.

Programación de SCA (Service Component Architecture)

SCA (Service Component Architecture) proporciona un sencillo, pero potente, modelo de programación para construir aplicaciones basadas en una arquitectura orientada a servicios (SOA).

Lenguaje de definición de componente de servicio

El lenguaje de definición de componente de servicio (SCDL) es un lenguaje basado en XML utilizado para describir los elementos SCA (Service Component Architecture) como, por ejemplo, módulos, componentes, referencias, importaciones y exportaciones.

Los distintos tipos de artefacto que existen en SCA se han diseñado para soportar algunos de los requisitos básicos de esta arquitectura orientada a servicios. Para empezar, SCA necesita un mecanismo para definir un componente de servicio básico. Una vez que haya un mecanismo para definir componentes de servicio, es importante poder hacer que estos servicios estén disponibles para los clientes, tanto dentro como fuera del módulos SCA actual. Además de esto, debe existir una construcción diseñada para importar y hacer referencia a los servicios externos al módulo SCA actual. Finalmente, SCA proporciona construcciones para componer servicios y módulos en aplicaciones mayores.

Las definiciones SCDL se organizan entre varios archivos. Por ejemplo, en una aplicación de aprobación de crédito, se puede almacenar el SCDL para la interfaz y la implementación en un archivo llamado `CreditApproval.component`. Las referencias se pueden incluir en el archivo `CreditApproval.component` (en línea) o en un archivo `sca.references` separado situado en la raíz del módulo. Cualquier referencia autónoma se coloca en el archivo `sca.references`. Las referencias autónomas pueden ser utilizadas por artefactos no SCA (JSP) dentro del mismo módulo SCA para invocar el componente SCA.

Tabla 14. Artefactos primarios que conforman un módulo de servicio SCA

Artefacto	Definición SCDL
Definición de módulo	• Se incluye en el archivo `sca.module` en el JAR del proyecto SCA de raíz

Tabla 14. Artefactos primarios que conforman un módulo de servicio SCA (continuación)

Artefacto	Definición SCDL
Componentes de servicio	• Un módulo puede contener 0..n definiciones de servicio • Cada definición de componente se incluye en un archivo `<NOMBRE_SERVICIO>.component`
Importaciones	• Un módulo puede contener 0..n definiciones de importación • Cada definición de importación se incluye en un archivo `<NOMBRE_IMPORTACIÓN>.import`
Exportaciones	• Un módulo puede contener 0..n definiciones de exportación • Cada definición de exportación se incluye en un archivo `<NOMBRE_EXPORTACIÓN>.export`
Referencias	• Dos tipos de referencias – En línea (se incluye dentro de una definición de componente de servicio) – Autónomo • Cada definición de componente se incluye en un archivo `<NOMBRE_SERVICIO>.reference`
Otros artefactos	• Otros artefactos incluyen: clases Java, archivos WSDL, otros archivos XSD de artefactos, BPEL.

Cuando se crea una aplicación SCA, WebSphere Integration Developer se ocupa de la generación de las definiciones SCDL apropiadas. Sin embargo, una familiaridad básica con SCDL puede ayudarle a comprender la arquitectura general y ayudarle cuando se depuran aplicaciones.

Definición de módulo

SCA (Service Component Architecture) define un modelo de despliegue estándar para los componentes empaquetados en un módulo de servicio. El archivo `sca.module` contiene la definición del módulo.

Un módulo SCA no es simplemente otro tipo de paquete. En WebSphere Process Server, un módulo de servicio SCA es equivalente a un archivo EAR de Java EE y varios otros submódulos Java EE. Los elementos Java EE como, por ejemplo, un archivo WAR, se pueden empaquetar junto con el módulo SCA. Los artefactos no SCA (JSP y otros) también se pueden empaquetar con un módulo de servicio SCA, esto les permite invocar servicios SCA mediante el modelo de programación de cliente SCA que utiliza un tipo especial de referencia llamada referencia autónoma.

Aquí aparece un ejemplo de un archivo `sca.module`:

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:module xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"
name="CreditApproval"/>
```

El diagrama muestra un módulo en WebSphere Integration Developer junto con su definición de módulo SCDL asociado que puede ver en un editor. En este ejemplo, el tipo de módulo es un módulo de mediación.

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:module xmlns:mt="http://www.ibm.com/xmlns/prod/websphere/scdl/moduletype/7.0.0"
  Xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"name="StockQuote">
  <mt:moduleType type="com.ibm.ws.sca.scdl.moduletype.mediation" version="1.0.0"/>
</Scdl:module>
```

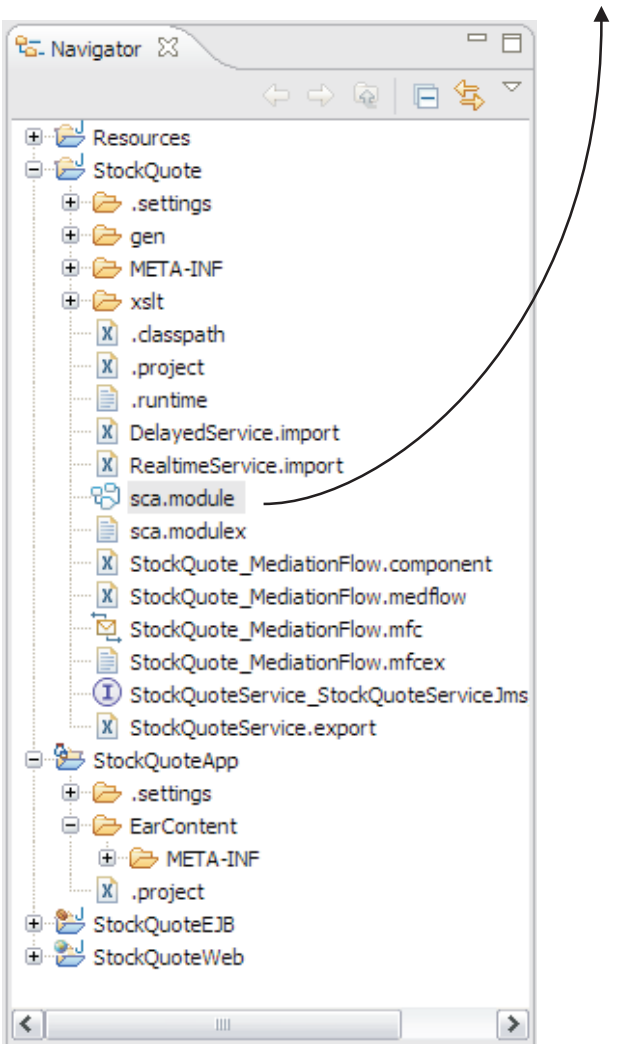


Figura 42. Visualización de la relación entre un módulo de WebSphere Integration Developer y la definición de módulo

Definición de componente

La definición de componente de servicio se incluye en un archivo llamado `<NOMBRE_SERVICIO>.component`. Los componentes SCA con sus dependencias asociadas se pueden definir y empaquetar juntas en unidades desplegables.

Esta figura proporciona un consulta más detallada en la definición del componente de servicio. Cada componente de servicio debe tener un nombre exclusivo dentro del módulo SCA y debe coincidir con la vía de acceso de archivo relativa a la raíz

del módulo. Tal como se ha especificado en la diapositiva anterior, la definición del componente de servicio se incluye en un archivo llamado `<NOMBRE_SERVICIO>.component`.

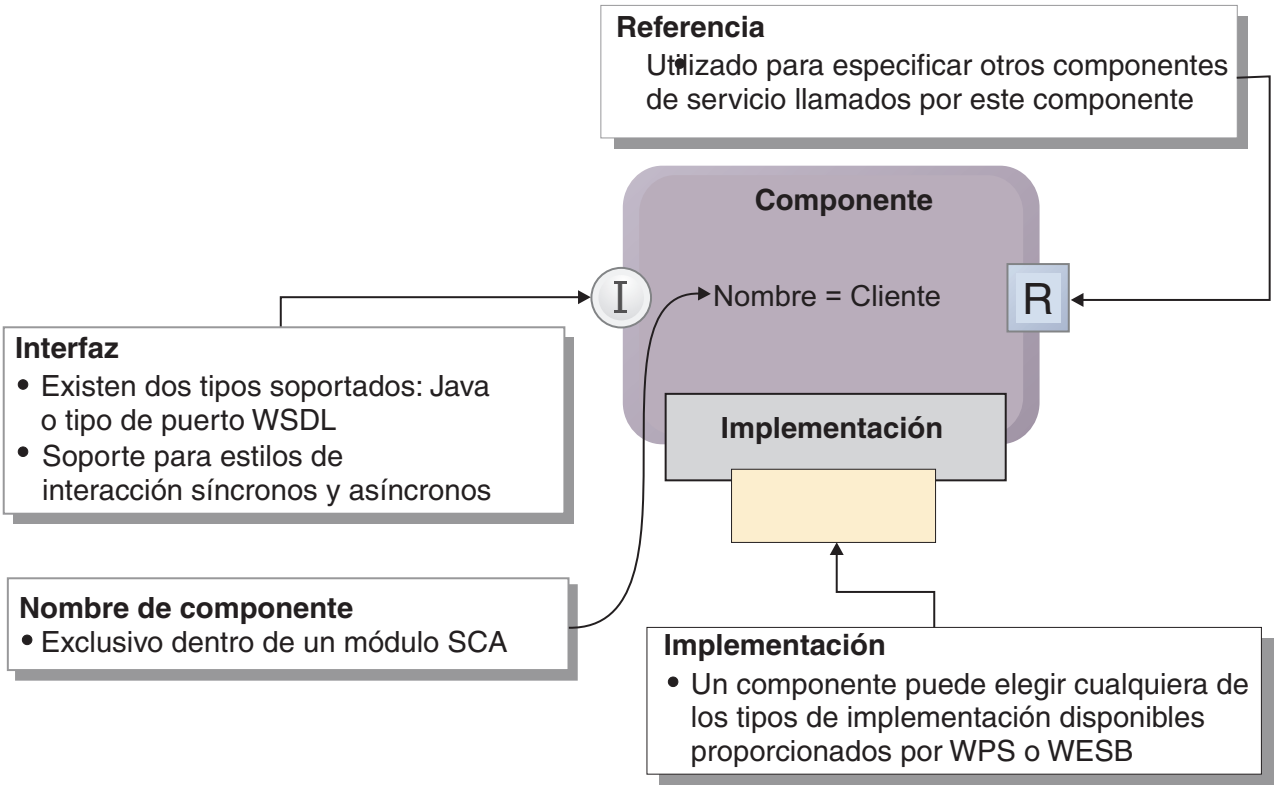
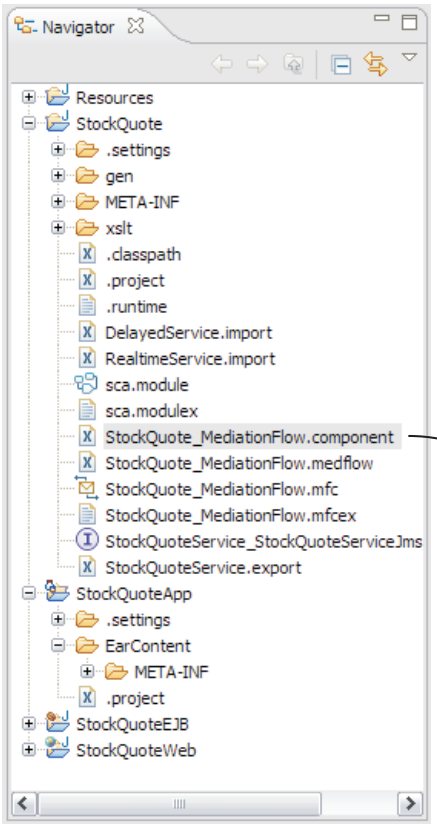


Figura 43. Definición de componente de servicio que incluye el nombre de componente, la implementación, las interfaces y las referencias

Cada componente de servicio debe tener un nombre exclusivo dentro del módulo SCA y debe coincidir con la vía de acceso de archivo relativa a la raíz del módulo. A continuación, cada componente de servicio puede tener una o más interfaces asociadas, que pueden ser definiciones de interfaz del tipo de puerto WSDL o Java. Las interfaces asociadas a un componente de servicio puede soportar un estilo de interacción síncrono o asíncrono con los clientes que llaman al servicio.

Cada componente de servicio se puede implementar de distintas formas, especificadas por la definición de la implementación. Finalmente, los componentes de servicio pueden invocar otros componentes de servicio o importaciones definidos en el módulo de servicio actual. En este caso, la referencia apropiada se debe definir para indicar qué servicio se utiliza. A menudo, este tipo de referencia está en línea con la definición del componente de servicio, aunque se puede colocar, de forma alternativa, en el archivo de referencias autónomo. Cada definición de componente de servicio puede tener cero o más referencias a otros servicios a los que llama el componente de servicio que se está definiendo.

Aquí aparece un ejemplo que muestra la definición para el componente StockQuote_MediationFlow. Tenga en cuenta que el componente incluye definiciones para una interfaz WSDL, dos referencias y una implementación.




```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mfc="http://www.ibm.com/xmlns/prod/websphere/scdl/mfc/7.0.0"
        xmlns:ns1="http://Resources/StockQuoteService"
        xmlns:ns2="http://stockquote.samp.sibx.websphere.ibm.com/DelayedService/"
        xmlns:ns3="http://stockquote.samp.sibx.websphere.ibm.com/RealtimeService/"
        xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"
        Xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/7.0.0"
        DisplayName="StockQuote_MediationFlow" name="StockQuote_MediationFlow">
 <interfaces>
  <interface xsi:type="wsdl:WSDLPortType" portType="ns1:StockQuoteService"/>
 </interfaces>
 <references>
  <reference name="DelayedServicePortTypePartner">
   <interface xsi:type="wsdl:WSDLPortType" portType="ns2:DelayedServicePortType">
    <method name="getQuote"/>
   </interface>
   <wire target="DelayedService"/>
  </reference>
  <reference name="RealtimeServicePortTypePartner">
   <interface xsi:type="wsdl:WSDLPortType" portType="ns3:RealtimeServicePortType">
    <Method name="getQuote"/>
   </interface>
   <wire target="RealtimeService"/>
  </reference>
 </references>
 <implementation xsi:type="mfc:MediationFlowImplementation" mfcFile="StockQuote_MediationFlow.mfc"/>
</Scdl:component>
```

Figura 44. Ejemplo de una definición de componente en el lenguaje de definición de componente de servicio

Definición de importación

La definición de importación se incluye en un archivo llamado `<NOMBRE_IMPORTACIÓN>.import`. Las importaciones SCA permiten a los clientes de un módulo SCA acceder a servicios que están fuera del módulo SCA actual.

Al igual que los componentes de servicio, las importaciones tienen un nombre y un conjunto de 1..N interfaces con las que se asocian. Las importaciones también tienen un atributo de enlace, que se utiliza para describir cómo se enlace el servicio externo con el módulo actual. Los tipos de enlace común se indican en el diagrama.

Las importaciones se pueden considerar como un tipo especial de componente de servicio en un módulo SCA. Las importaciones son destinos válidos de una definición de conexión para una referencia de servicio. Esto significa que para un cliente que invoca un servicio de destino, el modelo de programación es el mismo, independientemente de si la referencia apunta a una importación o a otro componente de servicio.

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:import xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ns1="http://stockquote.samp.sibx.websphere.ibm.com/DelayedService/"
        xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"
        xmlns:webservice="http://www.ibm.com/xmlns/prod/websphere/scdl/webservice/7.0.0"
        xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/7.0.0"
        displayName="DelayedService" name="DelayedService">
  <interfaces>
    <interface xsi:type="wsdl:WSDLPortType" portType="ns1:DelayedServicePortType">
      <method name="getQuote"/>
    </interface>
  </interfaces>
  <esbBinding xsi:type="webservice:WebServiceImportBinding"
        endpoint="http://localhost:9080/DelayedService/services/DelayedServiceSOAP"
        port="ns1:DelayedServiceSOAP" service="ns1:DelayedService"/>
</scdl:import>
```

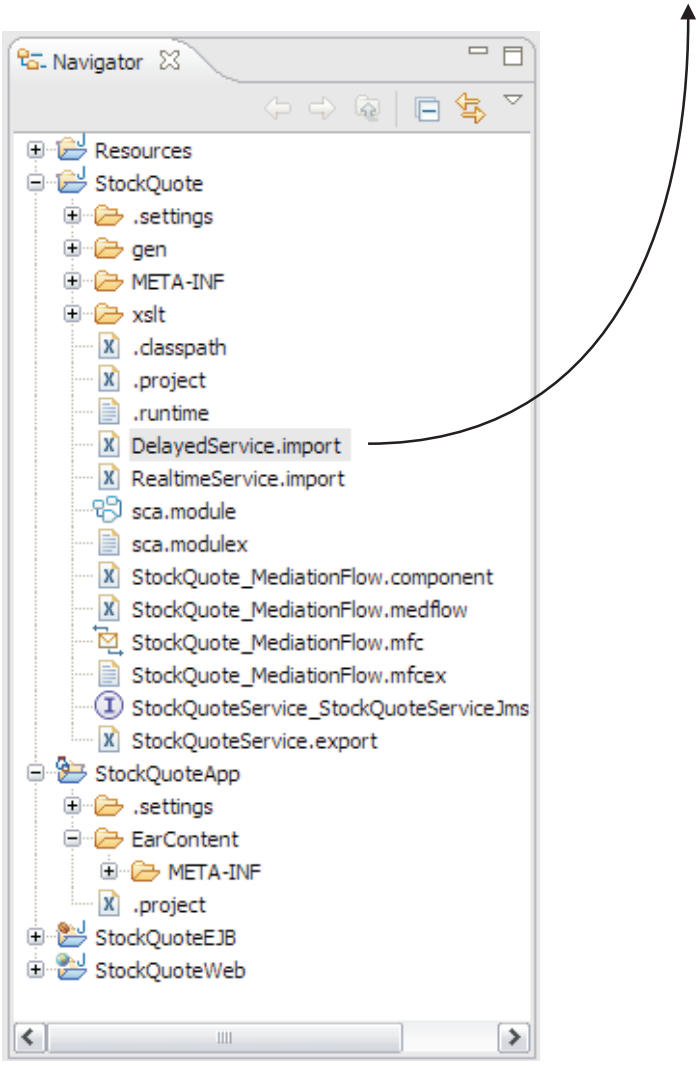
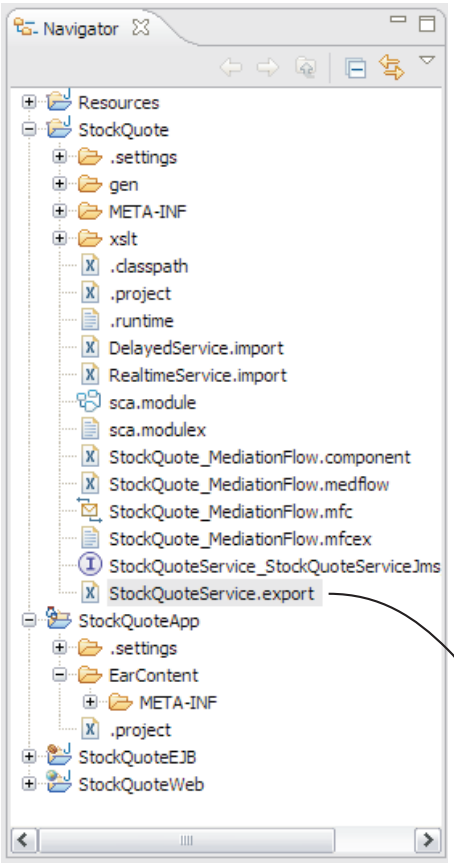


Figura 45. Ejemplo de una definición de importación en lenguaje de definición de componente de servicio

Definición de exportación

La definición de exportación se incluye en un archivo llamado `<NOMBRE_EXPORTACIÓN>.export`. Las exportaciones SCA proporcionan acceso a componentes de servicio definidos en un módulo SCA para ser utilizado por los clientes que están fuera del módulo SCA actual.

Los componentes de exportación incluyen un nombre y un atributo de destino, que dan nombre al componente de servicio que se va a exportar. Al igual que los componentes de importación, las exportaciones tienen un atributo de enlace que indica cómo se enlace el servicio de forma externa. Los tipos de enlace común se indican en el diagrama.




```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:export xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:_="http://Resources/StockQuoteService/Binding"
xmlns:ns1="http://Resources/StockQuoteService"
        xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"
        xmlns:webservice="http://www.ibm.com/xmlns/prod/websphere/scdl/webservice/7.0.0"
        Xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/7.0.0"
        displayName="StockQuoteService" name="StockQuoteService" target="StockQuote_MediationFlow">
 <interfaces>
  <interface xsi:type="wsdl:WSDLPortType" portType="ns1:StockQuoteService">
   <method name="getQuote"/>
  </interface>
 </interfaces>
```

Figura 46. Ejemplo de una definición de exportación en lenguaje de definición de componente de servicio

Definición de referencia

Los clientes SCA y no SCA que llaman a un componente de servicio necesitan una referencia a dicho servicio para poder invocarlo. Las referencias se pueden definir como autónomas en el archivo `sca.reference` o como en línea dentro de una definición de composición de servicio.

Cada referencia tiene un nombre, utilizado para buscar el servicio apropiado por un cliente que utiliza el modelo de programación de cliente. Además del nombre, una referencia también incluye un elemento de interfaz. La multiplicidad para una referencia indica cuántas definiciones de conexión pueden dar nombre a esta referencia como el origen. Finalmente, la definición de conexión especifica el nombre del componente de servicio de destino o importación que resuelve la referencia.

Existen dos métodos para definir referencias. El primer método es poner en línea la referencia en la definición del componente de servicio. Mediante el uso de este enfoque, las referencias sólo están disponibles para el componente de servicio en el que se incluyen las referencias. Otro enfoque es incluir las definiciones de referencia dentro del archivo de referencias autónomas. Para este enfoque, las referencias pueden ser utilizadas por un cliente no SCA o por otro componente del módulo. Un ejemplo de un componente no SCA que utiliza una referencia del archivo de referencias autónomas es un componente de interfaz de usuario como, por ejemplo, un JSP que necesita la capacidad de invocar un servicio particular. Para poder invocar un componente de servicio, el cliente necesita una referencia para que pueda utilizar el tiempo de ejecución SCA para buscar el servicio apropiado para invocar.

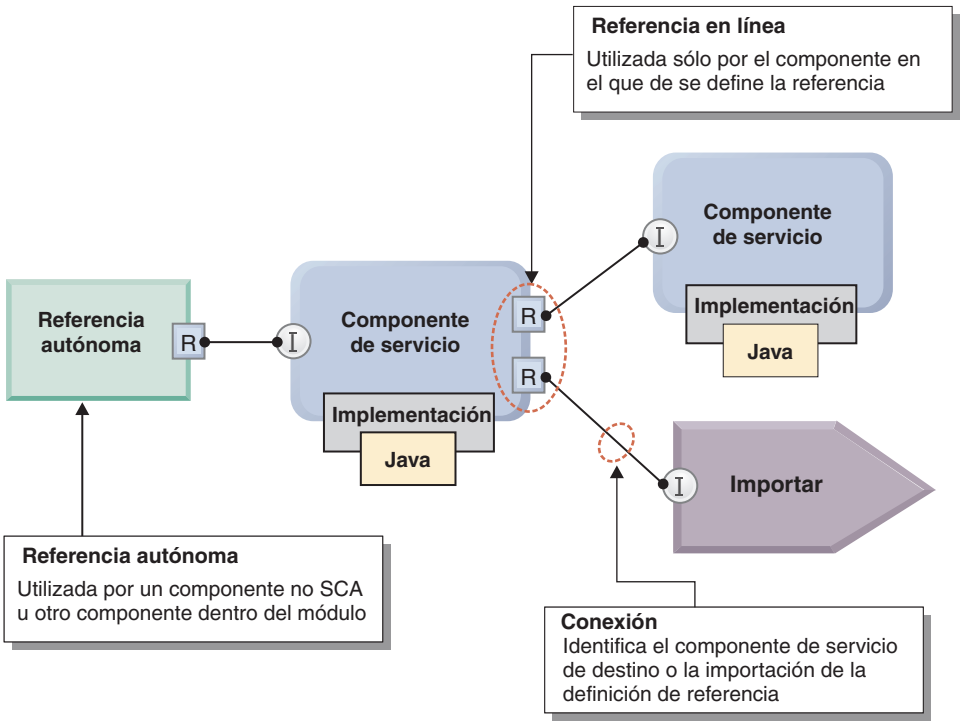


Figura 47. Los clientes pueden llamar a un componente de servicio utilizando las referencias autónomas o en línea.

Aquí aparece un ejemplo que muestra la definición para el componente que incluye dos referencias en línea: DelayedServicePortTypePartner y

RealtimeServicePortTypePartner. Tenga en cuenta que el componente incluye definiciones para una interfaz WSDL, dos referencias y una implementación.

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        Xmlns:mfc="http://www.ibm.com/xmlns/prod/websphere/scdl/mfc/7.0.0"
        Xmlns:ns1="http://Resources/StockQuoteService"
        xmlns:ns2="http://stockquote.samp.sibx.websphere.ibm.com/DelayedService/"
        xmlns:ns3="http://stockquote.samp.sibx.websphere.ibm.com/RealtimeService/"
        Xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"
        Xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/7.0.0"
        DisplayName="StockQuote_MediationFlow" name="StockQuote_MediationFlow">
 <Interfaces>
  <interface xsi:type="wsdl:WSDLPortType" portType="ns1:StockQuoteService"/>
 </Interfaces>
 <references>
  <reference name="DelayedServicePortTypePartner">
   <interface xsi:type="wsdl:WSDLPortType" portType="ns2:DelayedServicePortType">
    <method name="getQuote"/>
   </interface>
   <wire target="DelayedService"/>
  </reference>
  <reference name="RealtimeServicePortTypePartner">
   <interface xsi:type="wsdl:WSDLPortType" portType="ns3:RealtimeServicePortType">
    <Method name="getQuote"/>
   </interface>
   <wire target="RealtimeService"/>
  </reference>
 </references>
 <implementation xsi:type="mfc:MediationFlowImplementation" mfcFile="StockQuote_MediationFlow.mfc"/>
</Scdl:component>
```

Figura 48. Ejemplo de definiciones de referencia en línea

Fundamentos del modelo de programación SCA

El concepto de un *componente* de software forma la base del modelo de programación SCA (Service Component Architecture). Un componente es una unidad que implementa cierta lógica y que hace que estén disponibles componentes a través de una interfaz. Es posible que un componente también requiera los servicios que otros componentes han convertido en disponibles. En ese caso, el componente expone una *referencia* a dichos servicios.

En SCA, cada componente debe exponer al menos una interfaz. El diagrama de ensamblaje mostrado en la Figura 49 en la página 123 tiene tres componentes. Cada componente tiene una interfaz que se representa mediante la letra I dentro de un círculo. Un componente también puede hacer referencia a otros componentes. Las referencias se representan mediante la letra R, dentro de un cuadrado. A continuación, las referencias y las interfaces se enlazan en un diagrama de ensamblaje. Fundamentalmente, el desarrollador de integración “resuelve” las referencias conectándolas con las interfaces de los componentes que implementan la lógica necesaria.

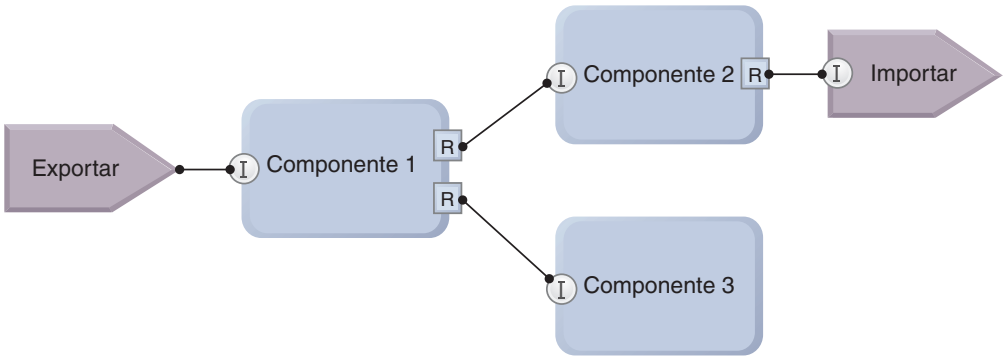


Figura 49. Diagrama de ensamblaje

Invocación de componentes SCA

Para proporcionar acceso a los servicios que se deban invocar, el modelo de programación SCA incluye una clase *ServiceManager*, que permite a los desarrolladores buscar los servicios disponibles, por nombre. He aquí un fragmento de código Java típico en el que se ilustra la búsqueda de servicios. La clase ServiceManager se utiliza para obtener una referencia al servicio BOFactory, que es un servicio proporcionado por el sistema:

```
//Obtener singleton de ServiceManager
ServiceManager smgr = new ServiceManager();
//Acceder al servicio BOFactory
BOFactory bof =(BOFactory)
        smgr.locateService("com/ibm/websphere/bo/BOFactory");
```

Nota: El paquete de ServiceManager es `com.ibm.websphere.sca`.

Los desarrolladores pueden utilizar un mecanismo similar para obtener referencias a sus propios servicios, especificando el nombre del servicio referenciado en el método *locateService*. Tras haber obtenido una referencia a un servicio mediante la clase ServiceManager, puede invocar cualquiera de las operaciones disponibles en dicho servicio, de una forma que sea independiente del protocolo de invocación y del tipo de implementación.

Existen tres estilos de invocación distintos para llamar a los componentes SCA:

- **Invocación síncrona**: cuando se utiliza este estilo de invocación, el llamante espera síncronamente a que se devuelva la respuesta. Este estilo es el mecanismo clásico de invocación.
- **Invocación asíncrona**: este mecanismo permite al llamante invocar un servicio sin esperar a que la respuesta se genere inmediatamente. En lugar de obtener la respuesta, el llamante obtiene un "tiquet", que se puede utilizar para recuperar la respuesta. El llamante recupera la respuesta llamando a una operación especial, que debe proporcionar el receptor de la llamada, con este fin.
- **Invocación asíncrona con devolución de llamada**: este estilo de invocación es como el anterior, pero delega la responsabilidad de devolver la respuesta al receptor de la llamada. El llamante necesita exponer una operación especial (la operación de devolución de llamada) que el receptor de la llamada puede invocar cuando la respuesta esté preparada.
- **Invocación asíncrona con respuesta aplazada**: en este estilo de invocación, el cliente invoca un servicio y sigue con el proceso hasta más adelante, cuando el cliente realiza una petición para capturar la respuesta.

Importaciones

A veces son los componentes, o las funciones, que estén disponibles en los sistemas externos como, por ejemplo, aplicaciones de herencia u otras implementaciones externas, los encargados de proporcionar la lógica empresarial. En dichos casos, el desarrollador de integración no puede resolver la referencia conectando una referencia a un componente que contenga la implementación que necesita para conectar la referencia a un componente que “señale a” la implementación externa. Dicho tipo de componente se denomina *importación*. Cuando defina una importación, necesitará especificar cómo se puede acceder al servicio externo en términos de ubicación y protocolo de invocación.

Exportaciones

De forma similar, si las aplicaciones externas deben acceder al componente, que es el caso más probable, debe hacer que resulte accesible. Para ello utilice un componente especial que exponga la lógica al "mundo exterior". Dicho tipo de componente se denomina *exportación*. También se invocan las exportaciones de forma síncrona o asíncrona.

Referencias autónomas

En WebSphere Process Server, un módulo de servicio SCA se empaqueta como un archivo EAR Java EE que contiene otros varios submódulos Java EE. Los elementos Java EE como, por ejemplo, un archivo WAR, se pueden empaquetar junto el módulo SCA. Los artefactos que no son SCA como, por ejemplo, JSP, también se pueden empaquetar junto con un módulo de servicio SCA. Este paquete les permite invocar servicios SCA a través del modelo de programación de cliente SCA utilizando un tipo especial de componente denominado referencia autónoma.

El modelo de programación SCA es altamente enunciativo. Los desarrolladores de integración pueden configurar aspectos como, por ejemplo, un comportamiento transaccional de las invocaciones, la propagación de credenciales de seguridad, si una invocación debe ser síncrona o asíncrona de un modo enunciativo, directamente en el diagrama de ensamblaje. El tiempo de ejecución SCA, no los desarrolladores, es el responsable de implementar el comportamiento especificado en estos modificadores. La flexibilidad enunciativa de SCA es una de las características más potentes de este modelo de programación. Los desarrolladores pueden concentrarse en implementar la lógica empresarial, en lugar de en cómo direccionar los aspectos técnicos como, por ejemplo, ser capaz de acomodar mecanismos de invocación asíncronos. De todos estos aspectos se encarga, automáticamente, el tiempo de ejecución de SCA.

Calificadores

Los calificadores determinan la interacción entre clientes de servicio y servicios de destino. Los calificadores se pueden especificar en referencias de componente de servicio, interfaces e implementaciones y, normalmente, son externos a una implementación.

Las distintas categorías de calificadores incluyen las siguientes:

- Transacción, que especifica el modo en que se manejan los contextos transaccionales en una invocación SCA.
- Sesión de actividad, que especifica cómo se propagan los contextos de sesión de actividad.

- Seguridad, que especifica los permisos.
- La fiabilidad asíncrona proporciona las normas para la entrega de mensajes asíncrona.

SCA permite que se apliquen estos calificadores de calidad de servicio (QoS) a los componentes de forma enunciativa (sin que se requiera programación alguna ni efectuar cambios en el código de implementación de servicios). Puede añadir calificadores de servicio utilizando WebSphere Integration Developer. Normalmente, se aplican los calificadores QoS cuando se está preparado para considerar el despliegue de soluciones.

Modelo de programación de cliente

El modelo de programación de cliente SCA se ha diseñado para localizar un servicio, crear objetos de datos, invocar un servicio y manejar excepciones que surgen por el componente invocado.

El modelo de programación de cliente SCA proporciona dos funciones primarias para clientes. El modelo de programación expone una interfaz que permite a los clientes localizar los servicios del módulo actual, y una vez localizado un servicio, el modelo de programación de cliente proporciona un método para que el cliente invoque las operaciones en dicho servicio.

Los clientes localizan los servicios utilizando la clase ServiceManager. Existen unas pocas formas para crear una instancia de la clase ServiceManager, en función del ámbito de búsqueda que desee para el servicio.

La interfaz de clave que deben conocer los clientes para localizar los servicios es `com.ibm.websphere.sca.ServiceManager`. Esta interfaz incluye un método locateService que devuelve una referencia a la implementación del servicio para el servicio solicitado. El parámetro string que se pasa en el método locateService representa el nombre de referencia para el servicio que desea localizar el cliente. La documentación Java para el modelo de programación SCA se incluye en el centro de información de WebSphere Process Server y, también, se incluye si elige instalar la documentación Java como parte de la instalación de WebSphere Process Server.

Una vez que un cliente ha localizado el servicio apropiado, existen dos tipos de modelos de invocación que se pueden utilizar para realizar una llamada a una operación o método que ofrece el servicio. En primer lugar, existe un estilo de *invocación dinámica* de interacción. La interfaz de clave para este estilo de interacción es `com.ibm.websphere.sca.Service`. El método invoke() en esta interfaz adopta el nombre de la operación que va a invocar, junto con los parámetros necesarios para llamar a dicha operación.

```
public Interface MyService {
public String someMethod(String input);

Service myService = (Service) serviceManager.locateService("myService");
DataObject input = ...
DataObject result = (DataObject) myService.invoke("someMethod", input);
```

Los clientes también pueden utilizar un método de *invocación estática (tipo seguro)* para llamar a una operación particular asociada a un servicio. Este tipo de invocación sólo funciona para las definiciones de interfaz que se especifican como Java. En esta situación, el cliente distribuye la devolución de la llamada locateService() a la interfaz apropiada y puede seguir realizando las llamadas del método apropiado del tipo seguro en dicha interfaz.

```
public Interface MyService {
public String someMethod(String input);

MyService myService = (MyService) serviceManager.locateService("myService");
String input = ...
String result = myService.someMethod(input);
```

Interfaces

Un componente de servicio tiene una o más interfaces con las que se asocia. Las interfaces asociadas a un componente de servicio anuncian las operaciones de negocio asociadas a este servicio.

Todos los componentes tienen interfaces del tipo WSDL. Sólo los componentes Java soportan las interfaces del tipo Java. Si un componente, importación o exportación, tiene más de una interfaz, todas las interfaces deben ser del mismo tipo.

```
<?xml version="1.0" encoding="UTF-8"?>
<scdl:component xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:mfc="http://www.ibm.com/xmlns/prod/websphere/scdl/mfc/7.0.0"
        xmlns:ns1="http://HelloWorldLibrary/HelloWorld" xmlns:ns2="http://HelloService/HelloService"
        xmlns:scdl="http://www.ibm.com/xmlns/prod/websphere/scdl/7.0.0"
        Xmlns:wsdl="http://www.ibm.com/xmlns/prod/websphere/scdl/wsdl/7.0.0"
        displayName="HelloWorldMediation" name="HelloWorldMediation">
 <Interfaces>
  <interface xsi:type="wsdl:WSDLPortType" portType="ns1:HelloWorld">
   <scdl:interfaceQualifier xsi:type="scdl:JoinTransaction" value="true"/>
  </Interface>
 </interfaces>
 <references>
  <reference name="HelloServicePartner">
   <interface xsi:type="wsdl:WSDLPortType" portType="ns2:HelloService"/>
   <scdl:referenceQualifier xsi:type="scdl:SuspendTransaction" value="false"/>
   <scdl:referenceQualifier xsi:type="scdl:Reliability"/>
   <scdl:referenceQualifier xsi:type="scdl:DeliverAsyncAt" value="commit"/>
   <wire target="HelloServiceImport"/>
  </Reference>
 </references>
 <implementation xsi:type="mfc:MediationFlowImplementation" mfcFile="HelloWorldMediation.mfc">
  <scdl:implementationQualifier xsi:type="scdl:Transaction" value="global"/>7
```

Figura 50. Ejemplo de una definición de interfaz dentro de una definición de componente SCDL

Estas interfaces se pueden especificar como interfaces Java o como interfaces de tipo de puerto WSDL. Sin embargo, no puede combinar interfaces Java e interfaces del tipo de puerto WSDL en la misma definición de componente de servicio. Los argumentos y tipos de retornos para estas interfaces se han especificado como tipos Java sencillos, clases Java, DataObjects de servicio o esquema XML (para las interfaces de tipo de puerto WSDL).

Se puede llamar a un componente de forma síncrona o asíncrona; esto es independiente de si la implementación es síncrona o asíncrona. Las interfaces de componente se han definido en la forma síncrona y el soporte asíncrono también se ha generado para éstas. Puede especificar un estilo de interacción preferido como síncrono o asíncrono. El tipo asíncrono anuncia a los usuarios de la interfaz que contiene, como mínimo, una operación que tarda una cantidad considerable de tiempo para completarse. Como consecuencia, el servicio que realiza la llamada debe evitar mantener una transacción abierta mientras espera a que se complete la operación y envíe su respuesta. El estilo de interacción se aplica a todas las operaciones de la interfaz.

Los componentes de distintos módulos se pueden conectar juntos publicando los servicios como exportaciones que tienen sus interfaces y arrastrando las exportaciones hasta el diagrama de ensamblaje necesario para crear importaciones. Al conectar componentes, también puede especificar calificadores de calidad de servicio en las implementaciones, referencias de socio e interfaces del componente.

Las importaciones tienen interfaces que son las mismas o un subconjunto de las interfaces del servicio remoto con las que están asociadas, de forma que se puede llamar a estos servicios remotos. Las importaciones se utilizan en una aplicación, exactamente del mismo modo que los componentes locales. Esto proporciona un modelo de ensamblaje uniforme para todas las funciones, independientemente de sus ubicaciones o implementaciones. El enlace de importación no debe estar definido durante el desarrollo; se puede realizar durante el despliegue.

Las exportaciones tienen interfaces que son las mismas o un subconjunto de las interfaces del componente con las que están asociadas, de forma que se puede llamar al servicio publicado. Una exportación arrastrada desde otro módulo a un diagrama de ensamblaje crea automáticamente una importación.

Desarrollo de módulos de servicio

Un componente de servicio debe estar contenido en un módulo de servicio. Desarrollar módulos de servicio para contener componentes de servicio es un elemento clave al proporcionar servicios a otros módulos.

Antes de empezar

Esta tarea da por supuesto que un análisis de los requisitos muestra que implementar un componente de servicio para su uso por otros módulos es beneficioso.

Acerca de esta tarea

Después de analizar los requisitos, tal vez decida que proporcionar y utilizar componentes de servicio es una manera eficaz de procesar información. Si determina que los componentes de servicio reutilizables se benefician del entorno, cree un módulo de servicio para contener los componentes de servicio.

Procedimiento

1. Identifique componentes de servicio que otros módulos pueden utilizar.

   Una vez que haya identificado los componentes de servicio, continúe en "Desarrollo de componentes de servicio".
2. Identifique componentes de servicio en una aplicación que puedan utilizar componentes de servicio en otros módulos de servicio.

   Una vez que haya identificado los componentes de servicio y sus componentes de destino, continúe en "Invocación de componentes" o "Invocación dinámica de componentes".
3. Conecte los componentes de cliente con los componentes de destino mediante cables.

Visión general del desarrollo de módulos:

Un módulo es una unidad de despliegue básica para una aplicación de WebSphere Process Server. Un módulo puede contener componentes, bibliotecas y módulos intermedios que utiliza la aplicación.

El desarrollo de módulos implica asegurarse de que los componentes, módulos intermedios y bibliotecas (colecciones de artefactos a los que el módulo hace referencia) que la aplicación necesita están disponibles en el servidor de producción.

WebSphere Integration Developer es la herramienta principal para desarrollar módulos para su despliegue en WebSphere Process Server. Aunque puede desarrollar módulos en otros entornos, es mejor utilizar WebSphere Integration Developer.

WebSphere Process Server da soporte a los módulos para servicios de empresa y módulos de mediación. Tanto los módulos como los módulos de mediación tipos de módulo SCA (Service Component Architecture). Un módulo de mediación permite la comunicación entre aplicaciones transformando la invocación de servicio a un formato que sólo entiende el destino, pasando la petición al destino y devolviendo el resultado al originador. Un módulo para un servicio de empresa implementa la lógica de un proceso empresarial. No obstante, un módulo también puede incluir la misma lógica de mediación que se puede empaquetar en un módulo de mediación.

Las secciones siguientes tratan de cómo implementar y actualizar módulos para WebSphere Process Server.

Componentes

Los módulos SCA contienen componentes, que son los bloques básicos para encapsular una lógica de empresa reutilizable. Los componentes proporcionan y consumen servicios y están asociados con interfaces, referencias e implementaciones. La interfaz define un contrato entre un componente de servicio y un componente llamante.

Con WebSphere Process Server, un módulo puede exportar un componente de servicio para su uso por otros módulos o importar un componente de servicio para su uso. Para invocar un componente de servicio, el módulo llamante hace referencia a la interfaz para el componente de servicio. Las referencias a las interfaces se resuelven configurando las referencias del módulo llamante a sus interfaces respectivas.

Para desarrollar un módulo, debe realizar las actividades siguientes:

1. Defina o identifique interfaces para los componentes del módulo.
2. Defina o maneje objetos empresariales utilizados por los componentes.
3. Defina o modifique componentes a través de sus interfaces.

   Nota: Un componente se define mediante su interfaz.
4. Opcional: Exporte o importe componentes de servicio.
5. Cree un archivo EAR (Enterprise Archive) para el tiempo de ejecución. Cree el archivo utilizando la característica EAR de exportación de WebSphere Integration Developer o el mandato serviceDeploy.

Tipos de desarrollo

WebSphere Process Server proporciona un modelo de programación de componentes para facilitar un paradigma de programación orientada a servicios. Para utilizar este modelo, un proveedor exporta interfaces de un componente de servicio para que un consumidor pueda importar esas interfaces y utilizar el

componente de servicio como si fuese local. Un desarrollador utiliza interfaces de tipo fuerte o de tipo dinámico para implementar o invocar el componente de servicio. Las interfaces y sus métodos se describen en la sección Referencias de este Centro de información.

Después de instalar los módulos de servicio en los servidores, puede utilizar la consola administrativa para cambiar el componente de destino para una referencia de una aplicación. El nuevo destino debe aceptar el mismo tipo de objeto empresarial y realizar la misma operación que solicita la referencia de la aplicación.

Consideraciones sobre el desarrollo de componentes de servicio

Al desarrollar un componente de servicio, formúlese las preguntas siguientes:

- ¿Este componente de servicio se exportará y lo utilizará otro módulo?

  En caso afirmativo, asegúrese de que otro módulo pueda utilizar la interfaz que defina para el componente.
- ¿El componente de servicio tardará relativamente mucho tiempo en ejecutarse?

  Si la respuesta es afirmativa, estudie implementar una interfaz asíncrona al componente de servicio.
- ¿Es ventajoso descentralizar el componente de servicio?

  Si la respuesta es afirmativa, estudie tener una copia del componente de servicio en un módulo desplegado en un clúster de servidores para beneficiarse del proceso paralelo.
- ¿La aplicación requiere una mezcla de recursos de compromiso de 1 fase y de 2 fases?

  En caso afirmativo, asegúrese de habilitar el soporte de último participante para la aplicación.

  Nota: Si crea la aplicación mediante WebSphere Integration Developer o crea el archivo EAR instalable mediante el mandato serviceDeploy, estas herramientas habilitan automáticamente el soporte de la aplicación. Consulte el tema "Utilización de recursos de compromiso de una fase y de dos fases en la misma transacción" en el Centro de información de WebSphere Application Server Network Deployment.

Desarrollo de componentes de servicio:

Desarrolle componentes de servicio para proporcionar lógica reutilizable a varias aplicaciones en el servidor.

Antes de empezar

En esta tarea se da por supuesto que ya se ha desarrollado e identificado el proceso que es útil para varios módulos.

Acerca de esta tarea

Varios módulos pueden utilizar un componente de servicio. Al exportar un componente de servicio, queda a disposición de otros módulos que hagan referencia al componente de servicio mediante una interfaz. En esta tarea se describe cómo construir el componente de servicio para que otros modelos puedan utilizarlo.

Nota: Un solo componente de servicio puede contener varias interfaces.

Procedimiento

1. Defina el objeto de datos para mover datos entre el llamante y el componente de servicio.

   El objeto de datos y su tipo forman parte de la interfaz entre los llamantes y el componente de servicio.
2. Defina una interfaz que los llamantes utilizarán para hacer referencia al componente de servicio.

   Esta definición de interfaz indica el componente de servicio y lista los métodos disponibles en el componente de servicio.
3. Genere la clase que implementa la llamada al servicio.
4. Desarrolle la implementación de la clase generada.
5. Guarde las interfaces e implementaciones de componentes con una extensión .java.
6. Empaquete el módulo de servicio y los recursos necesarios en un archivo JAR.

   Consulte “Instalación de un módulo en un servidor de producción” en este centro de información para ver una descripción de los pasos 6 a 8.
7. Ejecute el mandato serviceDeploy para crear un archivo EAR instalable que contenga la aplicación.
8. Instale la aplicación en el nodo de servidor.
9. Opcional: Configure los cables entre los llamantes y el componente de servicio correspondiente, si se llama a un componente de servicio de otro módulo de servicio.

   La sección de “Administración” de este centro de información describe la configuración de los cables.

Ejemplos de desarrollo de componentes

Este ejemplo muestra un componente de servicio síncrono que implementa un único método, CustomerInfo. La primera sección define la interfaz para el componente de servicio que implementa un método denominado getCustomerInfo.

```
public interface CustomerInfo {
 public Customer getCustomerInfo(String customerID);
}
```

El siguiente bloque de código implementa el componente de servicio.

```
public class CustomerInfoImpl implements CustomerInfo {
 public Customer getCustomerInfo(String customerID) {
  Customer cust = new Customer();

  cust.setCustNo(customerID);
  cust.setFirstName("Victor");
  cust.setLastName("Hugo");
  cust.setSymbol("IBM");
  cust.setNumShares(100);
  cust.setPostalCode(10589);
  cust.setErrorMsg("");

  return cust;
 }
}
```

x

La sección siguiente es la implementación de la clase asociada con StockQuote.

```
public class StockQuoteImpl implements StockQuote {

 public float getQuote(String symbol) {


     return 100.0f;
 }
}
```

Qué hacer a continuación

Invoque el servicio.

Invocación de componentes:

Los componentes con módulos pueden utilizar componentes en cualquier nodo de un clúster de WebSphere Process Server.

Antes de empezar

Antes de invocar un componente, asegúrese de que el módulo que contiene el componente esté instalado en WebSphere Process Server.

Acerca de esta tarea

Los componentes pueden utilizar cualquier componente de servicio disponible en un clúster de WebSphere Process Server utilizando el nombre del componente y pasando el tipo de datos que espera el componente. La invocación de un componente en este entorno implica localizar y, a continuación, crear la referencia en el componente necesario.

Nota: Un componente de un módulo puede invocar un componente en el mismo módulo, conocido como una invocación dentro del módulo. Implemente llamadas externas (invocaciones intermódulos) exportando la interfaz del componente que se proporciona e importando la interfaz en el componente llamante.

Importante: Cuando se invoca un componente que reside en un servidor distinto de aquél donde se ejecuta el módulo llamante, debe realizar configuraciones adicionales en los servidores. Las configuraciones necesarias dependen de si se llama al componente de forma asíncrona o síncrona. La manera de configurar los servidores de aplicaciones en este caso se describen en tareas relacionadas.

Procedimiento

1. Determine los componentes que necesita el módulo llamante.

   Tome nota del nombre de la interfaz dentro de un componente y el tipo de datos que la interfaz necesita.
2. Defina un objeto de datos.

   Aunque la entrada o el retorno puede ser una clase Java, lo óptimo es un objeto de datos de servicio.
3. Localice el componente.
   a. Utilice la clase ServiceManager para obtener las referencias disponibles en el módulo llamante.
   b. Utilice el método locateService() para encontrar el componente.

      Según el componente, la interfaz puede ser un tipo de puerto WSDL (Web Service Descriptor Language) o una interfaz Java.

4. Invoque el componente de manera síncrona.

   Puede invocar el componente mediante una interfaz Java o utilice el método invoke() para invocar el componente de manera dinámica.
5. Procese la devolución.

   El componente puede generar una excepción, de manera que el cliente tiene que poder procesar esa posibilidad.

Ejemplos de invocación de un componente

En el ejemplo siguiente se crea una clase ServiceManager.

```
ServiceManager serviceManager = new ServiceManager();
```

El ejemplo siguiente utiliza la clase ServiceManager para obtener una lista de componentes de un archivo que contenga las referencias de componente.

```
InputStream myReferences = new FileInputStream("MyReferences.references");
ServiceManager serviceManager = new ServiceManager(myReferences);
```

El código siguiente localiza un componente que implementa la interfaz StockQuote Java.

```
StockQuote stockQuote = (StockQuote)serviceManager.locateService("stockQuote");
```

El código siguiente localiza un componente que implementa una interfaz de tipo de puerto Java o WSDL. El módulo llamante utiliza la interfaz Service para interactuar con el componente.

Consejo: Si el componente implementa una interfaz Java, el componente puede invocarse utilizando la interfaz o el método invoke().

```
Service stockQuote = (Service)serviceManager.locateService("stockQuote");
```

En el ejemplo siguiente se muestra MyValue, código que llama a otro componente.

```
public class MyValueImpl implements MyValue {

 public float myValue throws MyValueException {

  ServiceManager serviceManager = new ServiceManager();

     // variables
         Customer customer = null;
         float quote = 0;
         float value = 0;

          // invocar
         CustomerInfo cInfo =
      (CustomerInfo)serviceManager.locateService("customerInfo");
         customer = cInfo.getCustomerInfo(customerID);

     if (customer.getErrorMsg().equals("")) {

          // invocar
       StockQuote sQuote =
      (StockQuote)serviceManager.locateService("stockQuote");
       Ticket ticket =  sQuote.getQuote(customer.getSymbol());
    // ... hacer otra cosa...
       quote =  sQuote.getQuoteResponse(ticket, Service.WAIT);

         // asignar
          value = quote * customer.getNumShares();
} else {
```

```
        // lanzar
        throw new MyValueException(customer.getErrorMsg());
    }
    // responder
        return value;
 }
}
```

Qué hacer a continuación

Configure los cables entre las referencias al módulo llamante y las interfaces de componente.

Invocación dinámica de un componente:

Cuando un módulo invoca un componente que tiene una interfaz de tipo de puerto WSDL (Service Descriptor Language), el módulo debe invocar el componente de forma dinámica utilizando el método invoke().

Antes de empezar

En esta tarea se da por supuesto que un componente llamante invoca a un componente de forma dinámica.

Acerca de esta tarea

Con una interfaz de tipo de puerto WSDL, un componente llamante debe utilizar el método invoke() para invocar el componente. Un módulo llamante también puede invocar un componente que tiene una interfaz Java de esta manera.

Procedimiento

1. Determine el módulo que contiene el componente necesario.
2. Determine la matriz que el componente necesita.

   La matriz de entrada puede ser de uno de los tres tipos siguientes:
   - Tipos o matrices Java primitivos en mayúsculas de este tipo
   - Clases o matrices Java de las clases
   - SDO (Service Data Objects)
3. Defina una matriz para que contenga la respuesta del componente.

   La matriz de respuesta puede ser de los mismos tipos que la matriz de entrada.
4. Utilice el método invoke() para invocar el componente necesario y pasar el objeto de matriz al componente.
5. Procese el resultado.

Ejemplos de invocación dinámica de un componente

En el ejemplo siguiente, un módulo utiliza el método invoke() para llamar a un componente que utiliza tipos de datos Java primitivos en mayúsculas.

```
Service service = (Service)serviceManager.locateService("multiParamInf");

  Reference reference = service.getReference();

  OperationType methodMultiType =
    reference.getOperationType("methodWithMultiParameter");

  Type t = methodMultiType.getInputType();

  BOFactory boFactory = (BOFactory)serviceManager.locateService
```

```
    ("com/ibm/websphere/bo/BOFactory");

  DataObject paramObject = boFactory.createbyType(t);

  paramObject.set(0,"input1")
  paramObject.set(1,"input2")
  paramObject.set(2,"input3")

  service.invoke("methodMultiParamater",paramObject);
```

En el ejemplo siguiente se utiliza el método invoke con una interfaz de tipo de puerto WSDL de destino.

```
Service serviceOne = (Service)serviceManager.locateService("multiParamInfWSDL");

 DataObject dob = factory.create("http://MultiCallWSServerOne/bos", "SameBO");
   dob.setString("attribute1", stringArg);

 DataObject wrapBo = factory.createByElement
  ("http://MultiCallWSServerOne/wsdl/ServerOneInf", "methodOne");
   wrapBo.set("input1", dob); //wrapBo encapsula todos los parámetros de methodOne
   wrapBo.set("input2", "XXXX");
   wrapBo.set("input3", "yyyy");

 DataObject resBo= (DataObject)serviceOne.invoke("methodOne", wrapBo);
```

Estilos de invocación

Con SCA, puede invocar los componentes de servicio utilizando los estilos de programación síncrono y asíncrono. Puede ensamblar los módulos en las soluciones generales donde los canales asíncronos entre los componentes de servicio y los módulos pueden aumentar el rendimiento general y la flexibilidad del sistema.

Un componente expone las interfaces de nivel empresarial a su lógica empresarial de aplicación de forma que se puede utilizar o invocar el servicio. La interfaz de un componente define las operaciones a las que se puede llamar y los datos que se pasan como, por ejemplo, argumentos de entrada, valores devueltos y excepciones. Asimismo, una importación y una exportación tienen interfaces de forma que se puede invocar el servicio publicado.

Todos los componentes tienen interfaces del tipo WSDL. Sólo los componentes de Java soportan las interfaces del tipo Java. Si un componente, importación o exportación, tiene más de una interfaz, todas las interfaces deben ser del mismo tipo.

Se puede llamar a un componente de forma síncrona o asíncrona; independientemente de si la implementación es síncrona o asíncrona. Las interfaces de componente se han definido en la forma síncrona y el soporte asíncrono también se ha generado para éstas. Puede especificar un estilo de interacción preferido como síncrono o asíncrono. El tipo asíncrono anuncia a los usuarios de la interfaz que contiene, como mínimo, una operación que tarda una cantidad considerable de tiempo para completarse. Como consecuencia, el servicio que realiza la llamada debe evitar mantener una transacción abierta mientras espera a que se complete la operación y envíe su respuesta. El estilo de interacción se aplica a todas las operaciones de la interfaz.

Cuando se crean aplicaciones en WebSphere Integration Developer, lo mejor es establecer de forma explícita el estilo de invocación que utilizan cada uno de los componentes para llamarse entre sí. Como mínimo, desea saber qué estilos de invocación se utilizan en la aplicación, mientras realiza el análisis de rendimiento o

desarrolla la estrategia de manejo de errores. Ciertamente, debe comprender las interacciones de la aplicación, cuando considere/establezca los límites de la transacción. A menudo, los usuarios se sorprenden de encontrar que definir o determinar los estilos de invocación entre los componentes no es una tarea fácil, como pudiera parecer. Esta sección explica cómo establecer o determinar qué estilo de invocación se utiliza durante la ejecución, basándose en características específicas de la aplicación.

Los estilos de invocación que proporciona SCA son:

- Síncrono
- Asíncrono utilizando la operación unidireccional
- Asíncrono con devolución de llamada
- Asíncrono con respuesta diferida

La API SCA que se utiliza en la implementación de Java para realizar la invocación determina qué estilo de invocación aparece durante la ejecución:

- `invoke()`: síncrono
- `invokeAsync()`: asíncrono
- `invokeAsyncWithCallback()`: asíncrono

En general, cuando se considera una interacción de un componente (de origen o cliente) con otro (de destino), el cliente de servicio determina qué tipo de invocación se utiliza. Por ejemplo, si el componente de origen es un componente Java™, el estilo de invocación del origen al destino está determinado a través de la API de invocación SCA particular que se utiliza en la implementación como, por ejemplo, invoke(), invokeAsync() o invokeAsyncWithCallback(). Cada uno de los otros componentes proporcionados en WebSphere Process Server tiene un conjunto de reglas que se utiliza para determinar si una invocación es síncrona o asíncrona.

Algunos componentes/importaciones se consideran asíncronos:

- BPEL de larga ejecución
- Tareas de usuario
- Importaciones MQ/MQ
- Importaciones JMS/Genéricos
- Importaciones JMS/JMS

Todas las invocaciones a componentes o importaciones de estos tipos deben ser invocaciones asíncronas. Si el componente que realiza la llamada (o de origen) inicia la interacción de forma síncrona, SCA conmuta la interacción para que sea asíncrona.

Invocación síncrona:

Las interfaces de componente de servicio (SCA) siempre se definen en el formato síncrono. Para cada interfaz síncrona, se puede generar una o más interfaces asíncronas.

Cuando se invoca un componente de servicio de forma síncrona, tanto el cliente (consumidor) y el proveedor de servicios se ejecutan en la misma hebra. EL componente que realiza la llamada dentro de WebSphere Process Server se bloquea hasta que se recibe una respuesta del proveedor.

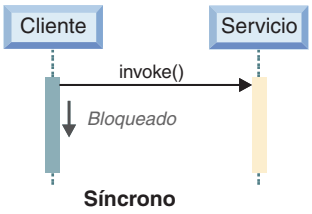


Figura 51. Invocación síncrona

Invocación asíncrona:

WebSphere Process Server entrega un modelo de programación potente para desarrollar aplicaciones asíncronas. Con la invocación asíncrona en SCA, hay tres tipos de estilos de interacción asíncrona disponibles: unidireccional, respuesta diferida y solicitud con devolución de llamada. Con los tres tipos de invocación asíncrona, el cliente recupera el control inmediatamente del tiempo de ejecución SCA con una llamada `invokeAsync()`.

Además de sus API publicadas y herramientas para desarrollar programas asíncronos utilizando Java, WebSphere Process Server también se suministra con varios enlaces de mensajería asíncrona incorporada y componentes asíncronos incorporados.

Hay tres maneras diferentes que el cliente puede capturar la respuesta posteriormente. En la primera, el cliente puede elegir descartar la respuesta por completo o si se trata de un llamada a un método vacío. En este caso, se dice que la invocación asíncrona es *unidireccional*. Otra opción es que el cliente llame a `invokeAsync()` y, a continuación, siga el proceso durante algún tiempo hasta que el cliente realice una solicitud de capturar la respuesta. Este caso de ejemplo se denomina *respuesta diferida*. Por último, el cliente también tiene la opción de realizar una *solicitud con devolución de llamada* asíncrona. Para ello, el cliente debe implementar primero la interfaz ServiceCallback. Luego, después de llamar a `invokeAsync()`, el tiempo de ejecución SCA realiza una devolución de llamada al manejador ServiceCallback para proporcionar la respuesta al cliente.

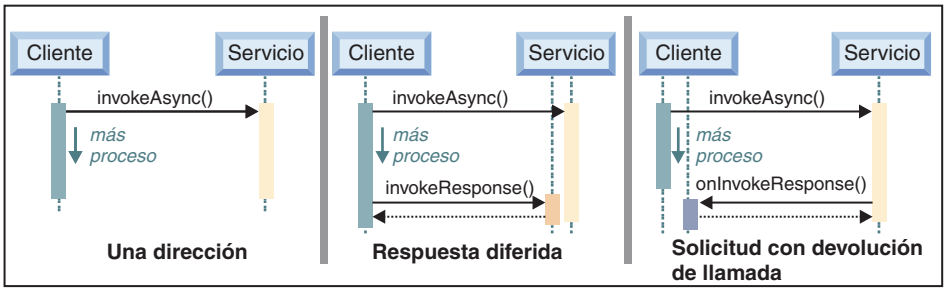


Figura 52. Modelos de invocación asíncrona

Las interfaces SCA siempre se definen en formato síncrono. Para cada interfaz síncrona, se puede generar una o varias interfaces asíncronas. Cuando el cliente elige un mecanismo de devolución de llamada, el componente de cliente tiene que implementar una clase: `<nombre de interfaz>.Callback.java`. La interfaz de esta clase se deriva de la interfaz del componente actual que el cliente desea utilizar.

Interacciones SCA:

SCA soporta la invocación síncrona y asíncrona de módulos. Los desarrolladores tienen la opción de seleccionar las interfaces apropiadas y los métodos de invocación para sus interacciones SCA.

El diagrama resume los distintos tipos de interfaz, los métodos y modelos de invocación soportados y cómo se pasan los datos entre cliente y servicio.

Para la invocación síncrona, los datos se pasan a través de la referencia dentro del mismo módulo SCA, mientras que para las llamadas asíncronas se pasan los datos por el valor. Asimismo, la tabla resume cuando es posible utilizar la invocación dinámica o segura de tipo basándose en el tipo de interfaz. Los métodos de invocación dinámica siempre están disponibles para el tipo de puerto WSDL o las interfaces Java. Sin embargo, para que estén disponibles los métodos de invocación segura de tipo para el cliente, se debe utilizar un tipo de interfaz Java para la definición de interfaz en la referencia de cliente apropiada.

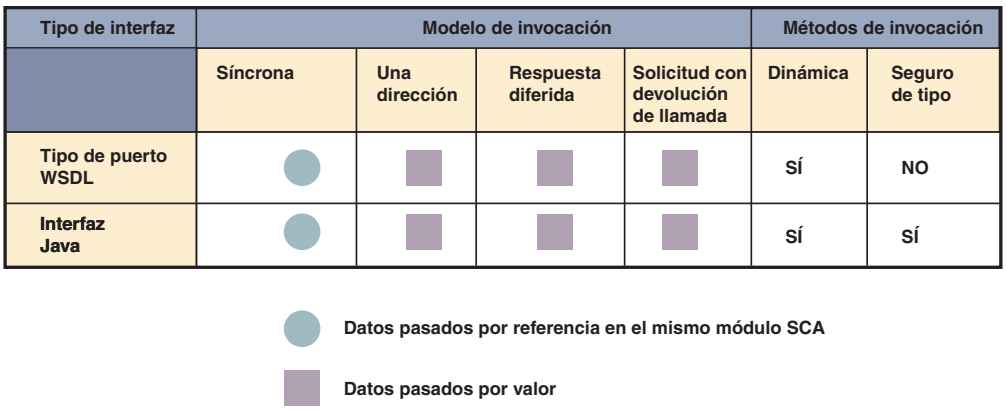


Figura 53. Resumen de modelos de invocación junto con los métodos soportados para pasar datos

Invocación de cliente dinámica

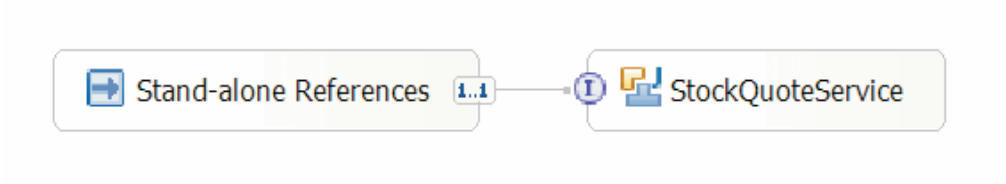
Existe una serie de métodos e interfaces clave necesarios para soportar la interacción síncrona y asíncrona cuando se utiliza la invocación de cliente dinámica.

Tabla 15. Resumen de métodos e interfaces clave para la invocación de cliente dinámica

Interfaz	Métodos	Descripción
Servicio	Object invoke(Service String, Object)	Se utiliza para invocar peticiones de servicio síncronas
	Ticket invokeAsync(String, Object)	Se utiliza para invocar peticiones de servicio asíncronas **unidireccionales** o **de respuesta aplazada**.
	Ticket invokeAsyncWithCallback(String, Object)	Se utiliza para invocar las peticiones de servicio asíncronas con **solicitud con devolución de llamada**. El cliente debe implementar la interfaz ServiceCallback
	Object invokeResponse(Ticket, long)	Se utiliza para obtener respuesta en el caso de la invocación **con respuesta aplazada**
ServiceCallback	void onInvokeResponse(Ticket, Object, Exception)	El cliente debe implementar la interfaz de devolución de llamada utilizando la invocación de servicio asíncrona de **petición con devolución de llamada**

Manejo de excepciones para la invocación síncrona:

Cuando se invoca un componente de servicio de forma síncrona, tanto el cliente como el proveedor de servicios se ejecutan en la misma hebra. El destino puede devolver un mensaje de respuesta, una excepción, o nada (en una operación unidireccional) al cliente. Si el resultado es una excepción, puede ser una excepción de negocio o una excepción del sistema. El cliente, en este caso, puede ser un código de aplicación o alguna forma de código de sistema.



Aquí aparece un cliente de ejemplo que invoca un componente Java declarado con una interfaz JType. La interfaz tiene un método declarado del modo siguiente:

```
public interface StockQuote {
 float getQuote(String symbol) throws InvalidSymbolException;
}
```

El código cliente tiene un aspecto parecido a este:

```
try {
 float quote = StockQuoteService.getQuote(String symbol);
 } catch (InvalidSymbolException s) {
 System.out.println(This is business exception declared in the Java interface.);
 } catch (ServiceRuntimeException e) {
 System.out.println(Unchecked system exception detected);
}
```

En este escenario, la primera excepción InvalidSymbolException indica que la petición ha alcanzado el proveedor de servicios, que no reconoce la entrada del cliente. El servicio lanza una excepción de negocio que indica que el símbolo proporcionado no es válido. Esta excepción de negocio es la única declarada por la firma de método.

Las excepciones JType como InvalidSymbolException sólo se capturan con los clientes que utilizan una referencia JType.

Además de la excepción de negocio declarada, el cliente puede recibir excepciones de sistema. Por ejemplo, si el sistema de la bolsa experimenta un problema, el servicio podría fallar para obtener la cotización con algunas excepciones no comprobadas. Cuando dicha excepción es lanzada por el servicio, se devuelve una excepción ServiceRuntimeException al cliente, y éste podría desear determinar la causa subyacente. El siguiente fragmento de código muestra cómo puede obtener esta información:

```
try {
 float quote = StockQuoteService.getQuote(String symbol);
 } catch (ServiceRuntimeException e) {
 Throwable t = e.getCause();
 if (t instanceof RemoteException) {
 system.out.println(RemoteException en sistema. Detalles: + e.toString());
}
```

Manejo de excepciones para la invocación asíncrona:

Cuando se invoca un componente de servicio de forma asíncrona, tanto el cliente como el proveedor de servicios se ejecutan en distintas hebras y se pueden producir condiciones de error en cualquier hebra. El cliente puede experimentar una excepción del sistema durante la invocación, o el proveedor de servicios puede experimentar una excepción empresarial o de sistema mientras se presta servicio a la solicitud.

En WebSphere Process Server, siempre hay un equivalente asíncrono de la interfaz síncrona.

Aquí aparece un ejemplo de una interfaz asíncrona:

```
public interface StockQuoteAsync {
public Ticket getQuoteAsync(String arg0);
public Ticket getQuoteAsyncWithCallback(String arg0);
public float getQuoteResponse(Ticket ticket, long timeout) throws InvalidSymbolException;
}
```

Aquí aparece el código cliente para una llamada que utiliza el patrón de invocación para la respuesta aplazada:

```
Ticket ticket = stockQuote.getQuoteAsync(symbol);
try {
 quote = stockQuote.getQuoteResponse(ticket, Service.WAIT);
 } catch (InvalidSymbolException s) {
 System.out.println(This is business exception declared in the interface.);
 } catch (ServiceRuntimeException e) {
 System.out.println(Unchecked system exception detected);
}
```

Al igual que en una invocación síncrona, InvalidSymbolException indica que la petición ha alcanzado el proveedor de servicios, que ha lanzado una excepción de negocio que indica que el símbolo no es válido. Esta excepción de negocio es la única declarada por la interfaz. Las excepciones JType como InvalidSymbolException sólo se capturan con clientes que utilizan una referencia JType.

Además de la excepción de negocio declarada, el cliente puede recibir excepciones de sistema como, por ejemplo, un error de conexión que se produce cuando se envía el mensaje. El cliente no puede recibir excepciones de sistema que se producen en la hebra del servicio (la hebra del servicio de la invocación asíncrona). De acuerdo con el modelo de programación asíncrona SCA, las excepciones de tiempo de ejecución que se producen en el componente de destino no se devuelven al componente de origen.

Caso de excepción en el manejo de excepciones asíncronas

Existe una excepción de la regla del modelo de programación asíncrona SCA que las excepciones de tiempo de ejecución que se producen en el componente de destino no se devuelven al componente de origen. Si el componente de origen es un componente de Business Process o de Staff Process, las excepciones del sistema que se producen en el componente del servicio de destino se devuelven al interlocutor. Esta capacidad permite a los diseñadores de procesos de negocio modelar y capturar excepciones de sistema y ejecutar la lógica de error si un cliente BPEL devuelve una excepción del sistema.

Consideraciones al invocar servicios en distintos servidores:

Una de las ventajas de la arquitectura orientada a servicios es la capacidad de los clientes de utilizar los servicios que existen en otros módulos de servicio. Para

equilibrar la carga de trabajo de forma equitativa, puede instalar las aplicaciones en distintos servidores de una célula y estas aplicaciones pueden residir en diferentes servidores físicos.

Una de las ventajas de WebSphere Process Server es la capacidad de distribuir la carga de trabajo de la aplicación entre varios servidores de una célula. Esta distribución permite un mejor equilibrio de carga de trabajo entre los distintos servidores de la célula y maximiza la capacidad de mantenimiento de los recursos del sistema porque sólo hay una copia de una aplicación de servicio dentro del servidor. De esta forma, una aplicación en el servidor A puede requerir un servicio instalado en el servidor B de la célula. Para utilizar los servicios de esta forma, debe configurar las comunicaciones entre los servidores. El tipo de configuración que realiza depende de si la llamada del componente del servicio invoca al servicio de forma asíncrona o síncrona.

Los temas relacionados describen cómo configurar los sistemas para ambas invocaciones, las asíncronas y las síncronas.

Configuración de servidores para invocar servicios de forma asíncrona:

Para permitir que componentes de servicios de distintos servidores se comuniquen, debe configurar los servidores de forma similar. Este tema describe la configuración realizada para permitir la comunicación para aplicaciones que invocan servicios de forma asíncrona en un servidor diferente.

Antes de empezar

La tarea asume que ya ha instalado WebSphere Process Server en los sistemas para los que está configurando las comunicaciones pero todavía no ha instalado las aplicaciones afectadas. Está utilizando una consola administrativa que puede examinar y cambiar la configuración de los servidores involucrados.

Acerca de esta tarea

Antes de instalar una aplicación que requiera los servicios del componente de un servicio instalado en otro sistema, debe configurar los sistemas para que puedan comunicar las solicitudes. Para módulos de servicios que utilizan invocaciones asíncronas, el proceso involucra buses externos y mediaciones SIB (Service Integration Bus).

Nota: Para esta tarea, el módulo del servicio que realiza la invocación reside en el Sistema A y el destino reside en el Sistema B.

Para esta tarea, Figura 54 en la página 141 contiene la información que debe utilizarse en la configuración.

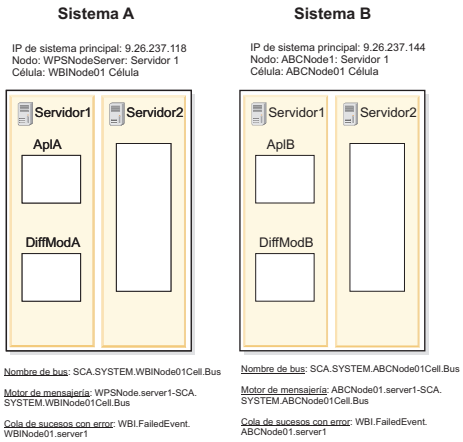


Figura 54. Invocación de un servicio en un sistema diferente

Nota: Para simplificar las cosas, sólo se muestran los servidores involucrados en esta comunicación de cada célula y los servidores residen en una máquina física diferente.

Procedimiento

1. Recopilar información sobre cada servidor involucrado en la comunicación. Necesita la información siguiente para los servidores de origen y de destino:
   - Dirección IP del sistema principal
   - Célula
   - Nodo
   - Servidor
   - Nombre de bus
   - Motor de mensajería
   - Nombre de cola de sucesos con anomalía
2. Instale la aplicación.
3. Cree un bus externo en cada servidor que apunte al otro servidor y establezca el tipo de definición de direccionamiento en `Directo, enlace del bus de integración de servicios`.

   Consulte el tema Conexión de buses de integración de servicios para utilizar mensajería de punto a punto en el centro de información de WebSphere Application Server Network Deployment, versión 7 para obtener más información.

   En el ejemplo, la configuración del bus externo y del enlace de mediación SIB en el Sistema A sería:

   ```
   Nombre del bus de integración de servicios al que conectarse (el bus externo):
   SCA.SYSTEM.ABCNode01Cell.Bus
   Motor de mensajería de la pasarela en el bus externo:
   ABCNode01.server1-SCA.SYSTEM.ABCNode01Cell.Bus
   Nombre del enlace de bus de integración de servicios: TestCrossCell
   Puntos finales del proveedor de bus de integración de rutina de carga:
   9.26.237.144:7277:BootstrapBasicMessaging
   ```

   La configuración del bus externo y del enlace de mediación SIB en el Sistema B sería:

   ```
   Nombre del bus de integración de servicios al que conectarse (el bus externo):
   SCA.SYSTEM.WBINode01Cell.Bus
   Motor de mensajería de la pasarela en el bus externo:
   WPSNode.server1-SCA.SYSTEM.WBINode01Cell.Bus
   ```

```
Nombre del enlace de bus de integración de servicios: TestCrossCell
Puntos finales del proveedor de bus de integración de rutina de carga:
9.26.237.118:7276:BootstrapBasicMessaging
```

Atención: El número de puerto de la rutina de carga es el puerto de dirección del punto final SIB. Si ha habilitado la seguridad, debe utilizar el puerto de dirección de punto final SIB seguro.

4. Sincronizar los enlaces de mediación SIB reiniciando los servidores.

   Debería ver mensajes como:

   ```
   [9/25/09 8:04:23:406 CDT] 00000034 SibMessage I [:] CWSIT0032I:
   El enlace de bus de integración de servicios TestCrossCell desde el motor de mensajería
   WPSNode01.server1-SCA.SYSTEM.WPSNode01Cell.Bus in bus SCA.SYSTEM.WPSNode01Cell.Bus
   al motor de mensajería ABCNode01.server1-SCA.SYSTEM. ABCNode01Cell.Bus en el bus SCA.SYSTEM.
   ABCNode01Cell.Bus iniciado.
   ```

5. Visualice los destinos para cada módulo de servicio.
6. Modifique la vía de acceso de envío por omisión de los destinos de salida del módulo de servicio de invocación que debe conectarse a los destinos en el otro sistema.

   Seleccione **Aplicaciones > Módulos SCA**, elija un módulo y pulse **Destinos de bus de sistema SCA**.

   El destino que va a conectarse tiene `importlink` en el nombre de destino, por ejemplo en el Sistema A el destino sería `sca/AppA/importlink/test/sca/cros/simple/custinfo/CustomerInfo`. Modifique la vía de acceso prefijando el nombre de bus externo al nombre de destino. En el ejemplo, el nombre de bus externo del segundo sistema es `SCA.SYSTEM.ABCNode01Cell.Bus`. El resultado es

   ```
   SCA.SYSTEM.ABCNode01Cell.Bus:sca/AppA/importlink/
   test/sca/cros/simple/custinfo/CustomerInfo
   ```

7. Opcional: Añada roles de emisor a los buses externos si ha habilitado la seguridad en los sistemas. Asegúrese de definir el usuario que utiliza cada aplicación en ambos sistemas desde el indicador de mandatos del sistema operativo. El mandato para añadir el rol es:

   ```
   wsadmin $AdminTask addUserToForeignBusRole -bus nombreBus
     -foreignBus nombreBusExterno -role nombreRol -user nombreUsuario
   ```

   Donde:

   nombreBus
   : Es el nombre del bus en el sistema en el que escribe el mandato.

   nombreBusExterno
   : Es el bus externo al que añade el usuario.

   nombre_usuario
   : Es el ID de usuario que se añade al bus externo.

Qué hacer a continuación

Inicie las aplicaciones.

Configuración de servidores para invocar servicios de forma síncrona:

Cuando el componente de un servicio invoca el componente de otro servicio de forma síncrona, se debe configurar el componente del servicio que realiza la invocación para que apunte al sistema que ejecuta el destino de tal modo que el servicio pueda comunicar los resultados al componente del servicio que realiza la invocación.

Antes de empezar

La tarea asume que ya ha instalado WebSphere Process Server en los sistemas para los que está configurando las comunicaciones pero todavía no ha instalado las aplicaciones afectadas. Está utilizando una consola administrativa que puede examinar y cambiar la configuración de los servidores involucrados.

Acerca de esta tarea

El componente de un servicio que invoca otro servicio de forma síncrona puede comunicarse con el destino sólo si se configura el nombre JNDI (Java Naming and Directory Interface) de exportación en el sistema de destino a un nombre JNDI del sistema que realiza la invocación.

Nota: Para esta tarea, el módulo del servicio que realiza la invocación reside en el Sistema A y el destino reside en el Sistema B.

Para esta tarea, Figura 55 contiene la información que debe utilizarse en la configuración.

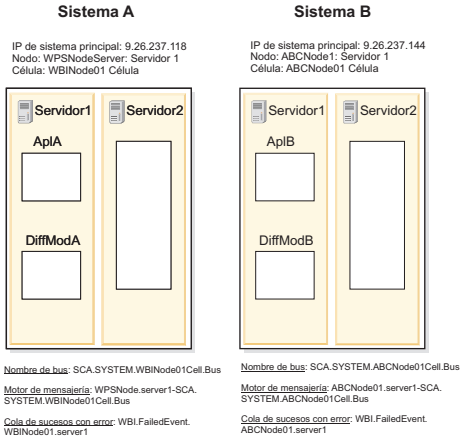


Figura 55. Invocación de un servicio en un sistema diferente

Nota: Para simplificar las cosas, sólo se muestran los servidores involucrados en esta comunicación de cada célula y los servidores residen en una máquina física diferente.

Procedimiento

1. Instale las aplicaciones en cada servidor.
2. Cree un enlace de espacio de nombres nuevo en el sistema que realiza la invocación (Sistema A en el ejemplo) que apunte a la exportación del sistema de destino.

   En el panel **Enlaces de espacio de nombres**, seleccione un ámbito de célula y pulse **Aplicar**. Con el ámbito cambiado, pulse **Nuevo** en la pantalla para crear el enlace nuevo.

   En el asistente, especifique lo siguiente (los valores son apropiados para la configuración de ejemplo):

   a. El tipo de enlace es CORBA
   b. Las propiedades básicas son:

- El identificador del enlace es una serie exclusiva, en nuestro ejemplo: `sca_import_test_sca_cross_simple_custinfo_CustomerInfo`
- El nombre del espacio de nombres es el nombre JNDI del EJB (Enterprise Java Bean) que está invocando en el sistema de destino, por ejemplo:

  `sca/AppB/export/test/sca/cros/simple/custinfo/CustomerInfo`

  Esto denomina la interfaz de exportación en el sistema de destino.
- La URL Corbaname es la dirección IP y el número de puerto del servicio de denominación en el sistema de destino:

  `corbaname:iiop:`*sistema_principal:puerto/RaízServidorServicioNombre#<interfaz.calificada.paquete>*

  Nota: Para WebSphere Process Server, el puerto es `BOOTSTRAP_ADDRESS`.

  Por ejemplo:

  ```
  corbaname:iiop:9.26.237.144:2809/NameServiceServerRoot#sca/
    AppB/export/test/sca/cros/simple/custinfo/CustomerInfo
  ```

Cuando termine, pulse **Siguiente** y compruebe los valores en la página **Resumen**. Después de la comprobación, pulse **Finalizar**.

Su sistema mostrará el enlace nuevo.

3. Guarde los cambios pulsando **Guardar**.
4. Si su configuración de célula cruzada se compone de servidores en el mismo sistema principal con le mismo nombre y encuentra anomalías de búsqueda de JNDI con `NameNotFoundException`, entonces deberá establecer una propiedad del sistema.

   Siga las direcciones de Problemas de acceso a aplicaciones bajo la subcabecera *Se están utilizando dos servidores con el mismo nombre ejecutados en el mismo sistema principal para interoperar*.

Qué hacer a continuación

Inicie las aplicaciones.El componente del servicio en el Sistema A puede invocar ahora de forma síncrona el servicio en el Sistema B.

Calificadores

Los calificadores son una parte importante de SCA porque permiten a los desarrolladores colocar los requisitos de calidad de servicio en el tiempo de ejecución de WebSphere Process Server.

Existen varias categorías diferentes de calificadores disponibles en SCA. Estas categorías de calificadores son: transacción, sesión de actividad, seguridad y fiabilidad asíncrona.

Cada calificador SCA tiene un ámbito particular dentro de la definición SCDL para un componente en el que se puede especificar el calificador. Por ejemplo, se pueden especificar algunos calificadores en el nivel de referencias, mientras que otros puede que sólo sean válidos en el nivel de interfaces o implementación. La tabla lista los distintos calificadores que están disponibles y el ámbito válido para cada uno. Los calificadores se clasifican a través del tipo de calidad de servicio que proporcionan como, por ejemplo, transacción o seguridad.

Tabla 16. Resumen de calificadores

Tipo	Calificador	Ámbito	Descripción
Transacción	transaction	Implementación	**global**: debe estar presente una transacción global para ejecutar el componente **local**: no debe existir una transacción global para ejecutar el componente **any**: el componente no se ve afectado por el estado transaccional **aplicación local**: el proceso de componente se produce dentro de un contenedor de transacciones locales WebSphere que está gestionado por la aplicación
	joinTransaction	Interfaz	**true**: el contenedor que lo incluye se une a la transacción de cliente **false** (valor predeterminado): el contenedor que lo incluye no se une a la transacción de cliente
	suspendTransaction	Referencia	**true**: las invocaciones síncronas del componente de destino no se ejecutan dentro de una transacción global de cliente. **false**: las invocaciones síncronas del componente de destino se ejecutan dentro de la transacción global de cliente
	deliverAsyncAt	Referencia	**call**: las invocaciones asíncronas de un servicio de destino se producen inmediatamente **commit**: las invocaciones asíncronas de un servicio de destino se producen como parte de una transacción global

Tabla 16. Resumen de calificadores (continuación)

Tipo	Calificador	Ámbito	Descripción
Respuesta asíncrona	reliability	Referencia	Especifica el nivel de calidad de servicio para la entrega de mensajería asíncrona. La fiabilidad puede adoptar uno de los siguientes valores: **bestEffort** o **assured**
	requestExpiration	Referencia	Especifica la longitud de tiempo (milisegundos) después de la que se debe descartar una petición asíncrona, si no se ha entregado.
	responseExpiration	Referencia	Especifica la duración (milisegundos) entre la hora en la que se envía una petición y la hora en la que se recibe una respuesta o devolución de llamada
Seguridad	securityIdentity	Implementación	El **permiso** especifica un nombre lógico para la identidad bajo la cual se ejecuta la implementación durante la ejecución.
	securityPermission	Interfaces, Interfaz, Método	La identidad del emisor debe tener el rol especificado desde este calificador para que tenga el permiso para ejecutar la interfaz o el método

Tabla 16. Resumen de calificadores (continuación)

Tipo	Calificador	Ámbito	Descripción
Sesión de actividad	activitySession	Implementación	**true**: debe haber un ActivitySession establecido para poder ejecutar este componente **false**: el componente se ejecuta sin ninguna sesión de actividad**any**: el componente no depende de si hay o no un ActivitySession
	joinActivitySession	Interfaz	**true**: el contenedor que lo incluye se une al ActivitySession de cliente **false**: el contenedor que lo incluye no se une al ActivitySession de cliente
	suspendActivitySession	Referencia	**true**: los métodos del componente de destino NO se ejecutan como parte de cualquier ActivitySession de cliente **false**: los métodos del componente de destino se ejecutan como parte de cualquier ActivitySession de cliente

Calificadores de transacción

Los calificadores de transacción permiten a los desarrolladores solicitar un entorno transaccional particular para los componentes de servicios de un módulo SCA. A continuación, aparece un resumen de estos calificadores:

transacción
El calificador de transacción se establece en el ámbito de implementación de un componente de servicio. Este calificador se puede establecer en 'global', 'local' (valor predeterminado), o 'any'. Si está establecido en global, el componente se ejecuta en el contexto de una transacción global. Si en una invocación está presente una transacción global, el componente se añade a este ámbito de transacción global. Si está establecido en local, el componente se ejecuta en el contexto de una transacción local. Finalmente, si el valor se establece en any, si hay una transacción global, el componente se une al ámbito de transacción global actual. Sin embargo, si no hay una transacción global, el componente se ejecuta en el contexto de una transacción local.

joinTransaction
El calificador joinTransaction se establece en el ámbito de interfaz de un componente de servicio. Este calificador se puede establecer en true o false (el valor false es el valor predeterminado). Si se establece en true, indica al tiempo de ejecución que no suspenda una transacción global (si está presente) en el límite de la interfaz. Si se establece en false, indica al

tiempo de ejecución que suspenda una transacción global (si está presente) en el límite de interfaz. La exposición del calificador transaccional joinTransaction en una interfaz proporciona los metadatos que pueden utilizar los ensambladores y desplegadores para asegurarse de que la aplicación ensamblada se comporta como es necesario. Los motivos para decidir si un componente de destino se federa con una transacción propagada dependen del ensamblador y el desplegador además de los tiempos de ejecución dinámicos.

suspendTransaction
El calificador suspendTransaction se establece en el nivel de referencia de un componente de servicio e identifica si se debe suspender una transacción global antes de invocar el servicio de destino asociado a la referencia. Este calificador se puede establecer en true o false (valor predeterminado).

deliveryAsyncAt
El calificador deliveryAsyncAt es como el calificador suspendTransaction, excepto en que pertenece a las interacciones asíncronas y no a los tipos síncronos,como en el caso con suspendTransaction. El calificador deliverAsyncAt puede tener el valor call (predeterminado) o commit. Si se establece en call, indica al tiempo de ejecución que el mensaje para la interacción asíncrona se debe confirmar en la cola inmediatamente cuando se haya realizado la llamada. El valor de commit indica que el mensaje se debe confirmar en la cola como parte de una transacción asociada a la unidad de trabajo actual.

Calificadores de respuesta asíncrona

Existen tres calificadores disponibles para indicar la calidad de servicio para la respuesta asíncrona. Cada uno de los calificadores de respuesta asíncrona se especifican en el ámbito de referencia. Lo que aparece a continuación es un resumen de calificadores de respuesta asíncrona:

fiabilidad
El calificador de fiabilidad se utiliza para especificar el nivel de calidad de servicio para la entrega de mensajería asíncrona. La fiabilidad se puede establecer en bestEffort o assured (valor predeterminado).

requestExpiration
El calificador requestExpiration se utiliza para especificar la duración de tiempo que debe retener el tiempo de ejecución una petición asíncrona, si todavía no se ha entregado. Después del tiempo indicado para este calificador, dado en milisegundos, se descarta esta petición.

responseExpiration
El calificador responseExpiration se utiliza para especificar la duración de tiempo que el tiempo de ejecución debe conservar una respuesta asíncrona o debe proporcionar una devolución de llamada. El valor de este calificador se proporciona en milisegundos.

Calificadores de seguridad

Existen dos calificadores disponibles para indicar la calidad de servicio relacionada con la seguridad. A continuación, aparece un resumen de estos calificadores:

securityIdentity
El calificador securityIdentity se utiliza para especificar la identidad de seguridad bajo la cual se debe ejecutar la implementación del componente

de servicio durante la ejecución. Este calificador se debe colocar en el ámbito de implementación para el componente de servicio y el valor proporcionado debe coincidir con el nombre lógico para la identidad bajo la que se ejecutará el componente.

securityPermission
El calificador securityPermission se especifica en el nivel de interfaces, incluido el nivel de interfaz o método. El valor para este calificador indica que un emisor de este servicio debe tener el rol que se ha especificado para invocar el servicio.

Para ambos calificadores, securityPermission y securityIndentity, la implementación subyacente para estos calificadores se basa en los conceptos de Java EE existentes.

Calificadores de sesión de actividad

El conjunto de calificadores de sesión de actividad son similares a los calificadores de transacción presentados anteriormente. El servicio ActivitySession es una ampliación del modelo de programación de WebSphere que puede proporcionar una unidad de trabajo alternativa cuando se compara con transacciones globales. De hecho, un contexto de sesión de actividad puede tener una vida más larga que una transacción global e, incluso, puede incluir transacciones globales. A continuación aparece un resumen de los calificadores de sesión de actividad:

joinActivitySession
El calificador joinActivitySession se establece en el nivel de interfaz e indica si el componente se debe unir a la sesión de actividad de un emisor de cliente. Existen dos valores para este calificador, true y false (valor predeterminado). Si se establece en true, indica que el tiempo de ejecución no debe suspender una sesión de actividad si está presente cuando se invoca el componente. Si se establece en false, indica que una sesión de actividad se debe suspender antes de invocar el componente.

activitySession
El calificador activitySession se especifica en el nivel de implementación y se utiliza para indicar si una sesión de actividad debe existir o no para poder ejecutar el componente de servicio con el que está asociado. Este calificador se puede establecer en ‘true', ‘false' o 'any‘ (valor predeterminado). Si está establecido en true, indica que el componente se ejecutará como parte de una sesión de actividad. Si está establecido en false, el componente no se debe ejecutar como parte de una sesión de actividad. Esto significa que joinActivitySession también se debe establecer en false para cualquier interfaz especificada para el componente. Finalmente, si este calificador se establece en any, el componente se ejecutará como parte de una sesión de actividad si está presente, de lo contrario, no lo hará.

suspendActivitySession
El calificador suspendActivitySession se establece en el nivel de referencia y se utiliza para indicar si se va a llamar o no a un servicio de destino asociado con una referencia como parte de la llamada de una sesión de actividad. Si está establecido en true, la sesión de actividad se suspende y los métodos del componente de destino no se ejecutarán como parte de la sesión de actividad de cliente. Si se establece en false (valor predeterminado), la sesión de actividad no se suspende y los métodos del componente de destino se ejecutarán como parte de ActivitySession de cliente.

Técnicas de programación SCA

Esta sección proporciona ejemplos de técnicas de programación SCA (Service Component Architecture).

Normas de tiempo de ejecución utilizadas para la conversión de Java a Service Data Objects

Para alterar temporalmente el código generado o determinar posibles excepciones en tiempo de ejecución relacionadas con las conversiones de Java a Service Data Object (SDO), es importante tener un conocimiento de las normas correspondientes. La mayoría de las conversiones son directas, pero hay algunos casos complejos en que la ejecución proporciona la mejor posibilidad cuando convierte el código generado.

Tipos y clases básicos

Durante la ejecución se realiza una conversión directa entre Service Data Objects y tipos y clases Java básicos. Entre los tipos y clases básicos se incluyen:

- Char o java.lang.Character
- Boolean
- Java.lang.Boolean
- Byte o java.lang.Byte
- Short o java.lang.Short
- Int o java.lang.Integer
- Long o java.lang.Long
- Float o java.lang.Float
- Double o java.lang.Double
- Java.lang.String
- Java.math.BigInteger
- Java.math.BigDecimal
- Java.util.Calendar
- Java.util.Date
- Java.xml.namespace.QName
- Java.net.URI
- Byte[]

Clases y matrices Java definidas por el usuario

Cuando se realiza la conversión de una clase o matriz Java a un SDO, se crea un objeto de datos durante la ejecución que tiene un URI generado invirtiendo el nombre de paquete del tipo Java y con un tipo igual al nombre de la clase Java. Por ejemplo, la clase Java `com.ibm.xsd.Customer` se convierte a un objeto SDO y URI `http://xsd.ibm.com` con el tipo Customer. Durante la ejecución se inspecciona entonces el contenido de los miembros de la clase Java y se asignan los valores a las propiedades del SDO.

Al convertir un SDO a un tipo Java, durante la ejecución se genera el nombre de paquete invirtiendo el URI y con el nombre del tipo igual al tipo del SDO. Por ejemplo, el objeto de datos con tipo Customer y URI `http://xsd.ibm.com` genera una instancia del paquete Java `com.ibm.xsd.Customer`. Durante la ejecución luego se extraen los valores de las propiedades del SDO y se asignan esas propiedades a los campos de la instancia de la clase Java.

Cuando la clase Java es una interfaz definida por el usuario, debe alterar temporalmente el código generado y proporcionar una clase concreta de la que se pueda crear una instancia durante la ejecución. Si durante la ejecución no se puede crear la clase concreta, se produce una excepción.

Java.lang.Object

Cuando un tipo Java es java.lang.Object el tipo generado es xsd:anyType. Un módulo puede invocar esta interfaz con cualquier SDO. Durante la ejecución se intenta crear una instancia de una clase concreta del mismo modo que para las clases y matrices Java definidas por el usuario, si durante la ejecución se puede encontrar esa clase. De lo contrario, se pasa el SDO a la interfaz Java durante la ejecución.

Aun cuando el método devuelva un tipo java.lang.Object, se convertirá durante la ejecución a un SDO sólo si el método devuelve un tipo concreto. Durante la ejecución se utiliza una conversión similar a la que se utiliza para convertir las clases y matrices Java definidas por el usuario a objetos SDO, como se describe en el párrafo siguiente.

Cuando se realiza la conversión de una clase o matriz Java a un SDO, se crea un objeto de datos durante la ejecución que tiene un URI generado invirtiendo el nombre de paquete del tipo Java y con un tipo igual al nombre de la clase Java. Por ejemplo, la clase Java `com.ibm.xsd.Customer` se convierte a un objeto SDO y URI `http://xsd.ibm.com` con el tipo Customer. Durante la ejecución se inspecciona entonces el contenido de los miembros de la clase Java y se asignan los valores a las propiedades del SDO.

En cualquier caso, si durante la ejecución no se puede completar la conversión se produce una excepción.

Clases de contenedor Java.util

Al convertir a una clase de contenedor Java concreta, como Vector, HashMap, HashSet y por el estilo, se crea una instancia de la clase de contenedor adecuada durante la ejecución. Durante la ejecución se utiliza un método similar al que se utiliza para las clases y matrices Java definidas por el usuario para llenar la clase de contenedor. Si durante la ejecución no se puede localizar una clase Java concreta, se llena la clase de contenedor con el SDO.

Al convertir clases de contenedor Java concretas a objetos SDO durante la ejecución, se utilizan los esquemas generados que se muestran en la “conversión de Java a XML”.

Interfaces Java.util

Para determinadas interfaces de contenedor del paquete `java.util`, se crean instancias de las clases concretas siguientes durante la ejecución:

Tabla 17. Conversión del tipo WSDL a la clase Java

Interfaz	**Clases concretas por omisión**
Colección	HashSet
Correlación	HashMap
Lista	ArrayList
Conjunto	HashSet

Alteración temporal de una conversión de Service Data Object a Java

Puede que a veces la conversión que el sistema crea entre un SDO (Service Data Object) y un objeto de tipo Java no satisfaga sus necesidades. Utilice este procedimiento para sustituir la implementación por omisión por la suya propia.

Antes de empezar

Asegúrese de que ha generado el WSDL para la conversión del tipo Java utilizando WebSphere Integration Developer o el mandato genMapper.

Acerca de esta tarea

Puede alterar temporalmente un componente generado que correlaciona un tipo WSDL con un tipo Java sustituyendo el código generado por código que satisfaga sus necesidades. Considere utilizar su propia correlación si ha definido sus propias clases Java. Utilice este procedimiento para hacer los cambios.

Procedimiento

1. Localice el componente generado. El componente se denomina *java_class*Mapper.component.
2. Edite el componente utilizando un editor de texto.
3. Convierta en comentario el código generado y proporcione su propio método.

   No cambie el nombre de archivo que contiene la implementación del componente.

Ejemplo

A continuación figura un ejemplo de un componente generado que se va a sustituir:

```
 private Object datatojava_get_customerAcct(DataObject myCustomerID,
    String integer)
{

  // Puede alterar temporalmente este código para la correlación personalizada.
  // Convierta en comentario este código y escriba código personalizado.

  // También puede cambiar el tipo Java que se pasa al
  // conversor, que este último intenta crear.


  return SDOJavaObjectMediator.data2Java(customerID, integer) ;

}
```

Qué hacer a continuación

Copie el componente y otros archivos al directorio en el que reside el módulo contenedor y conecte el componente de WebSphere Integration Developer o genere un archivo EAR (Enterprise Archive) con el mandato serviceDeploy.

Alteración temporal de la implementación de la Service Component Architecture generada

Puede que a veces la conversión que el sistema crea entre un código Java y SDO (Service Data Object) no satisfaga sus necesidades. Utilice este procedimiento para sustituir la implementación por omisión SCA (Service Component Architecture) por la suya propia.

Antes de empezar

Asegúrese de que ha generado la conversión del tipo Java a WSDL (Web Services Definition Language) utilizando WebSphere Integration Developer o el mandato genMapper.

Acerca de esta tarea

Puede alterar temporalmente un componente generado que correlaciona un tipo Java con un tipo WSDL sustituyendo el código generado por código que satisfaga sus necesidades. Considere utilizar su propia correlación si ha definido sus propias clases Java. Utilice este procedimiento para hacer los cambios.

Procedimiento

1. Localice el componente generado. El componente se denomina *java_class*Mapper.component.
2. Edite el componente utilizando un editor de texto.
3. Convierta en comentario el código generado y proporcione su propio método.

   No cambie el nombre de archivo que contiene la implementación del componente.

Ejemplo

A continuación figura un ejemplo de un componente generado que se va a sustituir:

```
private DataObject javatodata_setAccount_output(Object myAccount) {

  // Puede alterar temporalmente este código para la correlación personalizada.
  // Convierta en comentario este código y escriba código personalizado.

  // También puede cambiar el tipo Java que se pasa al
  // conversor, que este último intenta crear.


 return SDOJavaObjectMediator.java2Data(myAccount);


}
```

Qué hacer a continuación

Copie el componente y otros archivos al directorio en el que reside el módulo contenedor y conecte el componente de WebSphere Integration Developer o genere un archivo EAR (Enterprise Archive) con el mandato serviceDeploy.

Propagación de cabecera de protocolo a partir de enlaces de exportación que no sean SCA

El servicio de contexto es responsable de propagar el contexto (incluidas las cabeceras de protocolo, como la cabecera JMS y el contexto de usuario, como el ID de cuenta) junto con una vía de invocación SCA (Service Component Architecture). El servicio de contexto ofrece un conjunto de API y valores configurables.

Cuando la propagación del servicio de contexto es bidireccional, el contexto de respuesta siempre graba encima del contexto actual. Al ejecutar una invocación de un componente SCA a otro, una respuesta contiene un contexto distinto. Un componente de servicio tiene un contexto de entrada, pero cuando se invoca otro

servicio, éste último graba encima del contexto de salida original. El contexto de respuesta se convierte en el nuevo contexto.

Cuando la propagación del servicio de contexto es unidireccional, el contexto original sigue siendo el mismo.

El ciclo de vida del servicio de contexto está asociado a una invocación. Una petición tiene un contexto asociado, y el ciclo de vida de dicho contexto está enlazado al proceso de dicha petición en particular. Cuando dicha petición finaliza el proceso, el ciclo de vida de dicho contexto también finaliza.

Para un proceso BPEL (Business Process Execution Language) de breve ejecución, el contexto de respuesta sobrescribe el contexto de la petición. Recibe el contexto de respuesta de la primera petición y la pasa a la siguiente petición. Para un proceso BPEL (Business Process Execution Language) de larga ejecución, la infraestructura BPEL descarta el contexto de la respuesta. Almacena el contexto original y utiliza dicho contexto cuando efectúa otras llamadas de salida.

Los servicios de contexto tienen normas y tablas configurables que dictan el comportamiento del enlace. Para obtener más información, consulte la documentación de API y SPI generada que hay disponible en la sección Consulta. Durante el desarrollo en WebSphere® Integration Developer, puede establecer el servicio de contexto en las propiedades de importación y exportación. Para obtener más detalles, consulte la información de enlaces de importación y exportación en el Centro de información de WebSphere Integration Developer.

Programación de objetos de negocio

Los objetos de negocio son contenedores para datos de aplicación como, por ejemplo, un cliente o una factura. Los datos se intercambian entre los componentes a través de objetos de negocio. La estructura subyacente de un objeto de negocio es una definición de esquema XML (XSD), se proporciona un acceso mediante programa a los objetos de negocio a través de interfaces de objeto de negocio en WebSphere. De forma global, estos aspectos del objeto de negocio, su representación estructural, sus interfaces programáticas y su comportamiento y manipulación dentro de SCA (Service Component Architecture) y la infraestructura de objeto de negocio, que proporciona unos métodos potentes y coherentes para describir y entregar datos de negocio en la solución.

Esta guía proporciona información sobre los objetos de negocio de programación, incluidas las descripciones de áreas de problemas del manejo de las construcciones de esquema para algunas características. Si desea más información sobre cómo se define un objeto de negocio, las directrices de desarrollo de objetos de negocios y cómo utilizar las API de programación de objeto de negocio, consulte los artículos de la sección "Información relacionada".

Información relacionada

Web Services Description Language (WSDL) 1.1

Introducción a SDO (Service Data Objects)

Examen de objetos empresariales en WebSphere Process Server

Modelo de programación

La sección del modelo de programación de objeto de negocio describe cómo se encapsulan los tipos básicos de datos dentro de la infraestructura del objeto de negocio IBM. Para facilitar la creación y manipulación de objetos de negocio, la infraestructura del objeto de negocio amplía las especificaciones SDO (objetos de datos de servicio) proporcionando un conjunto de servicios Java.

Trabajo con la infraestructura del objeto empresarial de IBM

La infraestructura del objeto de negocio describe cómo las aplicaciones crean modelos de datos durante el tiempo de ejecución, integradas en el tiempo de ejecución y representadas en memoria.

La Tabla 1 resume cómo se implementan los tipos básicos de datos en la infraestructura del objeto de negocio.

Tabla 18. Abstracciones de datos y las correspondientes implementaciones

Abstracción de datos	Implementación	Descripción
Datos de instancia	Objeto de negocio	Los objetos de negocio son el mecanismo primario para representar las entidades de negocio, o las definiciones de mensaje literal documento, que habilita todo a partir de un objeto básico sencillo con propiedades escalares en un gráfico de objetos o una jerarquía compleja de gran tamaño. Un objeto de negocio es un corolario directo del concepto DataObject SDO.

Tabla 18. Abstracciones de datos y las correspondientes implementaciones (continuación)

Abstracción de datos	Implementación	Descripción
Metadatos de instancia	Gráfico de empresa	Los gráficos de negocio son envoltorios que se añaden en torno a un objeto de negocio sencillo o una jerarquía de objetos de negocio para proporcionar posibilidades adicionales, como el transporte de información de resumen de cambios y de resumen de sucesos relacionada con los objetos de negocio del gráfico de empresa. Un gráfico de negocio es un corolario directo del concepto DataGraph SDO, excepto que proporciona más que una simple cabecera de resumen de cambios única.
Metadatos de tipo	Metadatos empresariales Metadatos de tipo de objeto de negocio	Los metadatos de tipo de objeto de negocio son los metadatos que se pueden añadir a las definiciones de objeto de negocio para ampliar su valor en el tiempo de ejecución. Estos elementos de metadatos se añaden a la definición de esquema XML del objeto de negocio, así como los elementos conocidos `xs:annotation` y `xs:appinfo`.
Servicios	Servicios de objeto de negocio (API)	Los servicios de objeto de negocio son un conjunto de capacidades proporcionadas por encima de las posibilidades básicas proporcionadas por objetos de datos de servicio. Los ejemplos son los servicios como, por ejemplo, create, copy, equality y serialization. Estas API se encuentran en el paquete `com.ibm.websphere.bo`.

La infraestructura del objeto de negocio de WebSphere Process Server es una ampliación del estándar SDO. Por tanto, los objetos de negocio intercambiados entre los componentes de WebSphere Process Server son instancias de la clase `commonj.sdo.DataObject`. Sin embargo, la infraestructura del objeto de negocio de WebSphere Process Server añade varios servicios y funciones que simplifican y enriquecen la funcionalidad básica de DataObject.

Para facilitar la creación y manipulación de objetos de negocio, la infraestructura del objeto de negocio de WebSphere amplía las especificaciones SDO

proporcionando un conjunto de servicios de Java. Estos servicios forman parte del paquete denominado `com.ibm.websphere.bo`.

- **BOFactory**: el servicio clave que proporciona varias maneras de crear instancias de objetos de negocio.
- **BOXMLSerializer**: proporciona maneras de "hinchar" un objeto empresarial a partir de una corriente o escribir el contenido de un objeto empresarial, en formato XML, en una corriente.
- **BOCopy**: proporciona métodos que crean copias de objetos de negocio (semántica "profunda" y "superficial").
- **BODataObject**: le proporciona acceso a los aspectos de objeto de datos de un objeto empresarial como, por ejemplo, el resumen de cambios, el gráfico de empresa, y el resumen de sucesos.
- **BOXMLDocument**: el programa frontal del servicio que le permite manipular el objeto empresarial como un documento XML.
- **BOChangeSummary** y **BOEventSummary**: simplifica el acceso y la manipulación de la parte del resumen de cambios y del resumen de sucesos de un objeto empresarial.
- **BOEquality**: un servicio que le permite determinar si dos objetos de negocio contienen la misma información. Da soporte a la igualdad profunda y superficial.
- **BOType** y **BOTypeMetaData**: estos servicios materializan las instancias del tipo `commonj.sdo` y le permite manipular los metadatos asociados. De ahí que las instancias "Type" se puedan utilizar para crear objetos "por tipo" (del inglés "type").
- **BOInstanceValidator** : valida los datos de un objeto empresarial para ver si se ajustan al estándar XSD.

Modelado de objetos de negocio

Los datos de objeto de negocio que fluyen a través del tiempo de ejecución de WebSphere Process Server se ha modelado utilizando los esquemas XML. los esquemas XML son una alternativa para las definiciones de tipo de documento (DTD) y se pueden utilizar para ampliar las funciones de las áreas de tipos, herencia y presentación de datos. El esquema XML proporciona una formato para modelar los tipos de datos que son un estándar del sector, ampliamente adoptado y es una plataforma y lenguaje neutral.

Definición del espacio de nombres de destino:

La mayoría de los problemas de comunicaciones y de negocio que puede resolver el XML requiere una combinación de varios vocabularios de XML. XML tiene un mecanismo para cualificar los nombres que se deben asignar a distintos espacios de nombres como, por ejemplo, los espacios de nombres que se aplican a sectores diferentes. En XML, un identificador universal de recursos (URI) proporciona un nombre exclusivo para asociar con las definiciones de elemento, atributo y tipo en un esquema XML.

Existen dos requisitos para el espacio de nombres de destino del objeto de negocio:

- El tiempo de ejecución y las herramientas de gestión de procesos de negocio prefieren los espacios de nombres de destino que se parecen a `http://www.foo.com/xyz` respecto a `urn:foo:com:xyz`.
- La infraestructura del objeto de negocio requiere un espacio de nombres de destino para los objetos de negocio.

Definición de objeto de negocio:

WebSphere Process Server proporciona un mecanismo flexible para definir o importar objetos de negocio.

Básicamente, existen tres formatos diferentes de esquema XML que reconoce WebSphere Process Server como una definición de objeto de negocio:

- Definición de tipo complejo de nivel superior
- Definición de tipo complejo anónimo de nivel superior
- Elemento de nivel superior que hace referencia a un tipo complejo determinado

Definición de tipo complejo de nivel superior

A continuación aparece un ejemplo de una definición de tipo complejo de nivel superior:

```
<complexType name="ProductType">
 <sequence>        <element name="name" type="string"/>
  <element name="color" type="string" maxOccurs="unbounded"/>
 </sequence>    </complexType>
```

La importación o definición de objetos de negocio que están todos definidos utilizando sólo las definiciones de tipo complejo es el esquema más flexible y con más capacidad de gestión. La parte positiva de este modelo es una biblioteca de tipos que permite la reutilización. La reutilización se puede realizar a través de tres métodos diferentes.

- Primer método: los nuevos tipos se pueden crear utilizando el modelo de derivación de tipo complejo (de ampliación o restricción).
- Segundo método: los nuevos tipos de agregación se pueden crear utilizando los tipos complejos existentes y los tipos sencillos disponibles como primitivos.
- Tercer método: se pueden crear nuevas definiciones de documento complejo, como los tipos complejos de agregación.

La otra implicación de los objetos de negocio definidos como tipos complejos es que cuando el tipo complejo es utilizado por los tipos de componente `JService` para transferir datos dentro del tiempo de ejecución, para poder mantener la compatibilidad con WS-I, se debe crear un elemento que haga referencia al tipo complejo con nombre.

Definición de tipo complejo anónimo de nivel superior

A continuación aparece un ejemplo de la definición de tipo complejo anónimo de nivel superior:

```
<element name="Product">
 <complexType>
  <sequence>
   <element name="name" type="string"/>
  <element name="color" type="string" maxOccurs="unbounded"/>
  </sequence>
 </complexType>
</element>
```

Si los objetos de negocio importados son todos definiciones de elemento anónimo, éstos están preparados para incluirse en las invocaciones `JService`. Sin embargo, no son reutilizables de forma inherente.

Elemento de nivel superior que hace referencia a un tipo con nombre

Esto es un ejemplo de un elemento de nivel superior que hace referencia a un tipo con nombre:

```
<element name="product" type="prod:ProdType"/>
```

Los objetos de negocio que hacen referencia a tipos complejos con nombre podrían ser frecuentes en un entorno que ya ha definido operaciones WSDL que requieren definiciones de elemento. En este escenario, es importante considerar la posible disposición del tipo complejo y las definiciones de elemento:

- Los elementos pueden cohabitar con sus definiciones de tipo complejo en el mismo archivo esquema XML.
- Los elementos pueden cohabitar con sus definiciones de tipo complejo incorporadas en un archivo WSDL.
- Los elementos se pueden definir en el esquema XML `A.xsd`, mientras que su definición del tipo complejo se define en el archivo de esquema XML `B.xsd`.
- Los elementos se pueden incorporar en un archivo WSDL, que hace referencia a una definición de tipo complejo definida en un archivo de esquema XML.

Ejemplo

El ejemplo demuestra todos los mecanismos para definir un objeto de negocio combinados de forma conjunta.

```
<schema
 targetNamespace="http://www.app.com/Address"
 xmlns:addr="http://www.app.com/Address"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns="http://www.w3.org/2001/XMLSchema">

 <complexType name="Address">
  <sequence>
   <element name="street1" type="string"/>
   <element name="street2" type="string"/>
   <element name="city" type="string"/>
   <element name="state" type="string"/>
   <element name="zip" type="string"/>
  </sequence>
 </complexType>

 <element name="homeAddress" type="addr:Address"/>
 <element name="workAddress" type="addr:Address"/>
 <element name="otherAddress" type="addr:Address"/>

 <element name="individualContact">
  <complexType>
   <sequence>
    <element name="firstName" type="string"/>
    <element name="lastName" type="string"/>
    <element ref="addr:HomeAddress"/>
    <element ref="addr:WorkAddress"/>
    <element ref="addr:OtherAddress"/>
   </sequence>
  </complexType>
 </element>

 <element name="businessContact">
  <complexType>
   <sequence>
  <element name="name" type="string"/>
    <element ref="addr:WorkAddress"/>
   </sequence>
```

```
  </complexType>
 </element>

 <element name="chairmanOfTheBoard">
  <complexType>
   <sequence>
    <element name="startDate" type="date"/>
    <element ref="addr:IndividualContact"/>
    <element ref="addr:BusinessContact"/>
   </sequence>
  </complexType>
 </element>
</schema>
```

Las siguientes directrices conforman el método preferido para definir los objetos de negocio:

- Los elementos se definen utilizando tipos con nombres; los tipos anónimos no se recomiendan.
- Los elementos y las definiciones de tipo complejo no cohabitan en el mismo esquema XML o el mismo archivo WSDL. Esta práctica no recomienda la reutilización del tipo.
- Los tipos complejos se han definido en archivos de esquema XML, no en definiciones WSDL, creando una biblioteca de tipos como concepto. De nuevo, este tipo de definición habilita y recomienda la reutilización del tipo complejo.
- Las definiciones de elemento se crean según sea necesario para hacer referencia a una sola definición de tipo complejo. Por ejemplo, la definición de un elemento dentro de WSDL es un patrón que se recomienda.
- Las definiciones de elemento normalmente utilizan el mismo espacio de nombres de destino que su definición de tipo complejo.

Definición de propiedad de objeto de negocio:

El esquema XML proporciona tipos complejos, tipos sencillos y atributos, que se utilizan para crear objetos de negocio.

Las definiciones de tipo complejo, definiciones de tipo complejo anónimo y los elementos que hacen referencia a las definiciones de tipo complejo se utilizan para definir los objetos de negocio externos. El término propiedad se utiliza para definir los datos de un objeto de negocio. El término se deriva del término de propiedad del objeto de datos de servicio y se define a través de la interfaz `commonj.sdo.Property`. Es sinónimo al concepto de un atributo.

Una propiedad puede ser simple o compleja. Una propiedad simple se puede definir como un atributo de esquema XML, o como un elemento de esquema XML con un tipo que es un tipo sencillo de esquema XML. Una propiedad compleja puede hacer referencia a otro objeto de negocio, o puede definir una estructura compleja dentro del objeto de negocio actual.

Esta soportado el sistema del tipo de esquema XML completo.

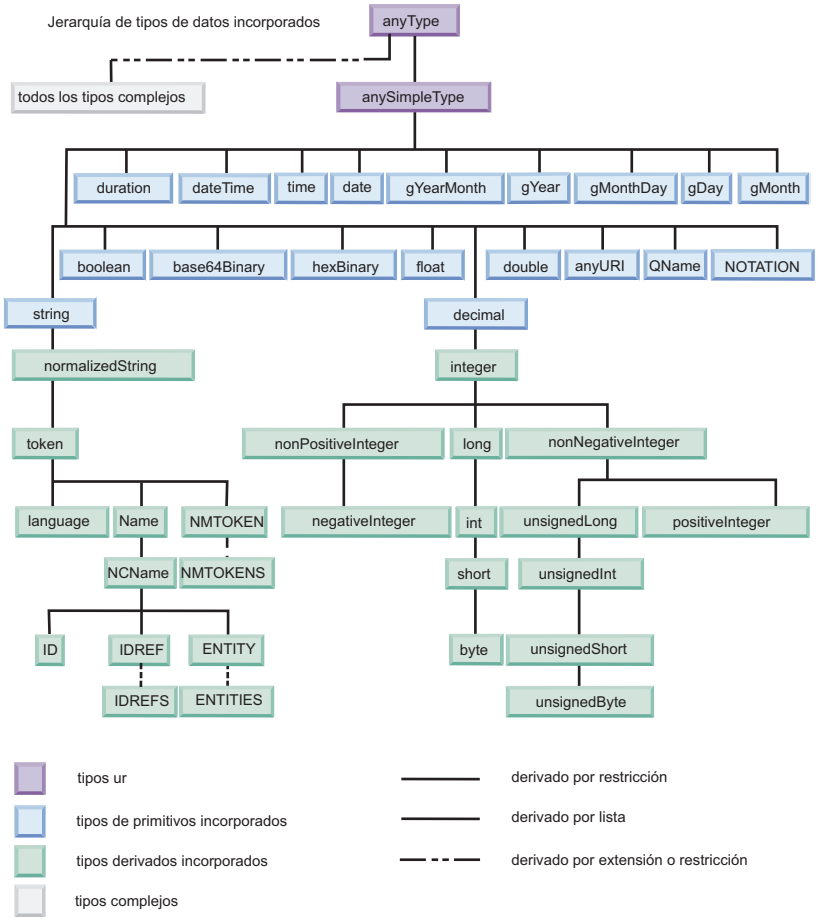


Figura 56. Tipos simples de esquema XML

Artefactos XSD y WSDL soportados:

Cuando se importa un WSDL o un esquema en un proyecto en WebSphere Integration Developer, los objetos de negocio representados en el WSDL o el esquema se pueden utilizar para desarrollar un módulo. Sin embargo, es importante tener en cuenta que sólo determinados artefactos de un esquema se representan como *objetos de negocio* (por ejemplo, elementos de nivel raíz/superior y tipos complejos denominados). Determinados artefactos como, por ejemplo, tipos complejos anónimos anidados, *no* se representan como objetos de negocio. Estas restricciones son un resultado de a qué artefactos se puede acceder en el esquema XML. Por ejemplo, si importa un esquema que ha generado un solo objeto de negocio, lo más probable es que el resto de los elementos fueran tipos complejos anónimos. La siguiente información detalla qué artefactos XSD y WSDL generan objetos de negocio.

Objetos de negocio de definiciones XSD importadas

Cuando se importa un esquema XML en un proyecto, sólo se representan determinados artefactos como objetos de negocio. Las listas siguientes muestran qué artefactos están soportados durante el tiempo de ejecución y el momento de creación:

Los artefactos XSD que generan objetos de negocio en el momento de creación:

- Tipos complejos definidos en el nivel de raíz

- Elementos definidos en el nivel raíz con tipos complejos anónimos

Estos artefactos generan tipos sencillos definidos por el usuario en el momento de creación a los que se puede hacer referencia a través de objetos de negocio:

- Tipos sencillos definidos en el nivel raíz
- Elementos definidos en el nivel raíz con tipos sencillos anónimos

Objetos de negocio de archivos WSDL importados

Cuando se importa en un proyecto una definición WSDL que incluye un esquema XSD en línea, sólo se representan determinados artefactos como objetos de negocio. Las listas siguientes muestran qué artefactos están soportados durante el tiempo de ejecución y el momento de creación:

Artefactos XSD en línea que generan objetos de negocio en el momento de creación:

- Tipos complejos definidos en el nivel de raíz
- Los elementos definidos en el nivel raíz con tipos complejos anónimos Y el nombre del elemento no contienen los nombres de ninguna operación/mensaje (ya que estos elementos podrían ser elementos envueltos-doc-lit de los que WebSphere Integration Developer elimina el envoltorio automáticamente).

Estos artefactos generan tipos sencillos definidos por el usuario en el momento de creación a los que se puede hacer referencia a través de objetos de negocio:

- Tipos sencillos definidos en el nivel raíz
- Elementos definidos en el nivel raíz con tipos sencillos anónimos

Objetos de negocio de tiempo de ejecución de artefactos XSD

Estos artefactos generan objetos de negocio durante el tiempo de ejecución:

- Tipos complejos definidos en el nivel de raíz
- Elementos definidos en el nivel raíz con tipos complejos anónimos
- Elementos definidos en el nivel raíz que hacen referencia a un tipo complejo

Objetos de negocio de tiempo de ejecución de archivos WSDL

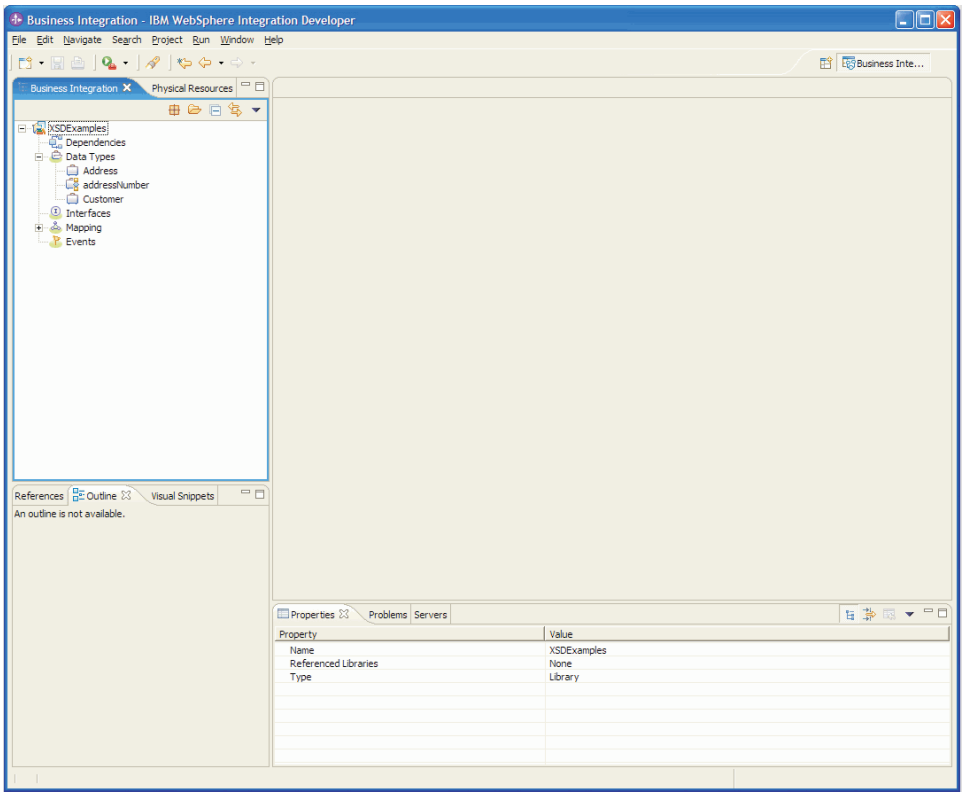
Estos artefactos generan objetos de negocio durante el tiempo de ejecución:

- Tipos complejos definidos en el nivel de raíz
- Los elementos definidos en el nivel raíz con tipos complejos anónimos Y el nombre del elemento no contienen los nombres de ninguna operación/mensaje (ya que estos elementos podrían ser elementos envueltos-doc-lit de los que WebSphere Integration Developer elimina el envoltorio automáticamente).
- Elementos definidos en el nivel raíz que hacen referencia a un tipo complejo

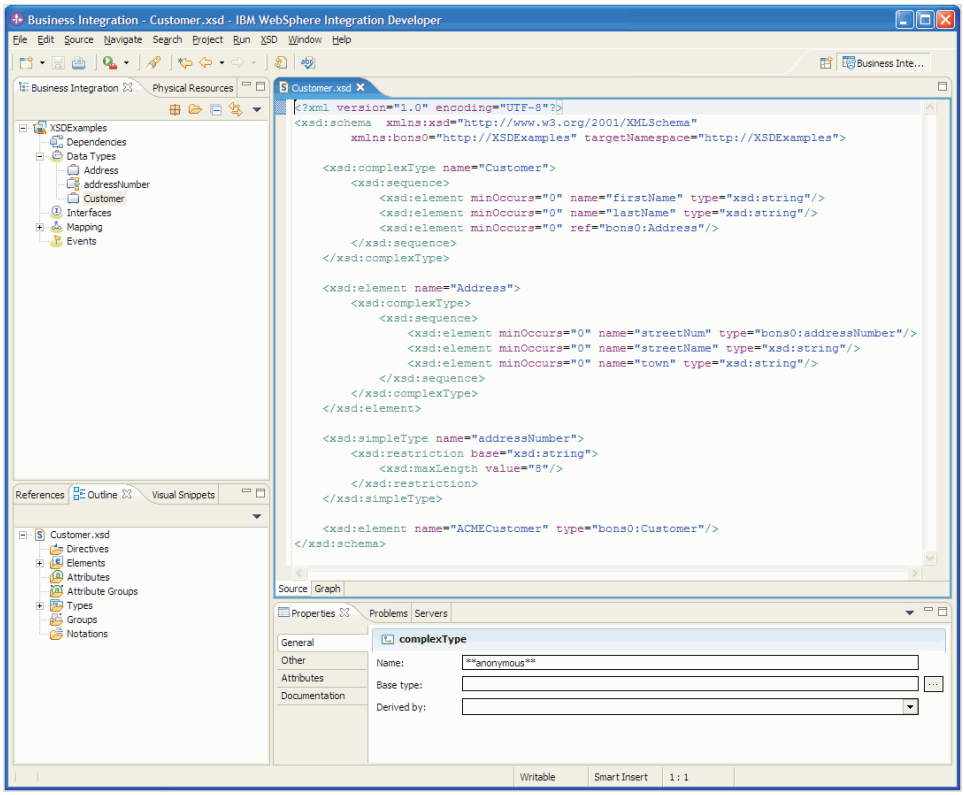
Ejemplos

1. **Ejemplo XSD (soportado en el momento de creación)**

   Este ejemplo muestra un proyecto (XSDExamples) en la vista de Business Integration con los objetos de negocio visualizados:



Esto muestra el archivo Customer.xsd en el editor de esquemas XSD:



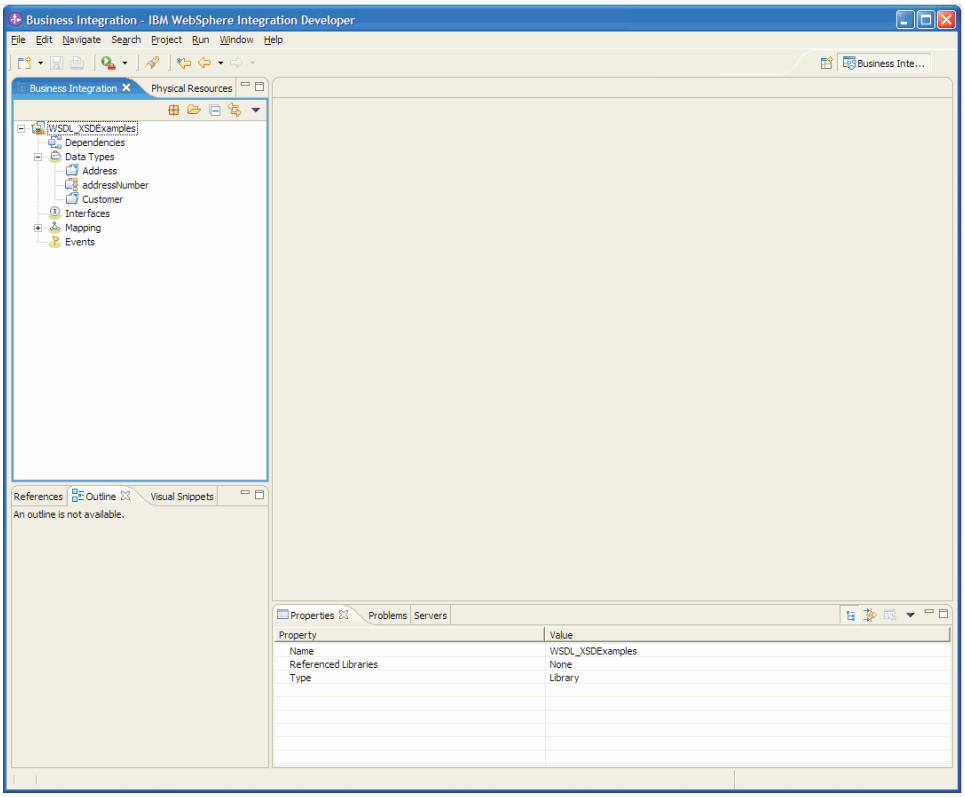
Los ejemplos anteriores ilustran el siguiente soporte:

Tabla 19. Soporte de artefacto XSD

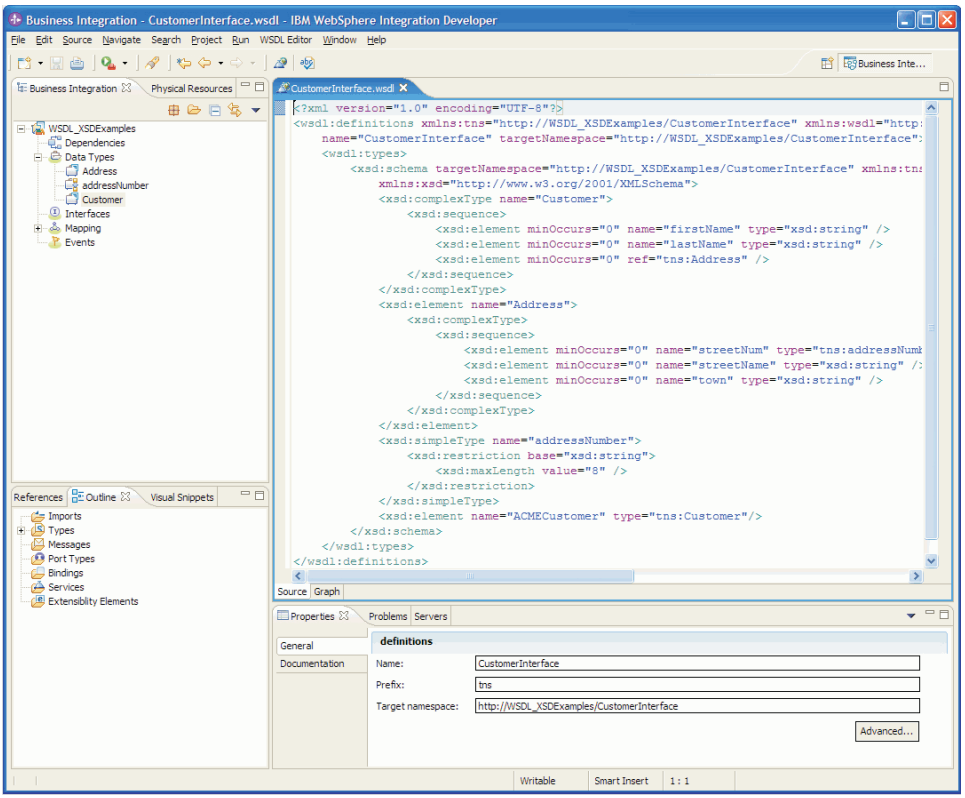
Soporte XSD	Artefacto XSD del ejemplo anterior
Tipos complejos definidos en el nivel de raíz	Customer
Elementos definidos en el nivel raíz con tipos complejos anónimos	Address
Elementos definidos en el nivel raíz con tipos sencillos definidos por el usuario	addressNumber

2. **Ejemplo de WSDL (soportado en el momento de creación)**

   Este ejemplo muestra un proyecto (WSDL_XSDExamples) en la vista de Business Integration con los objetos de negocio visualizados:



   Esta captura de pantalla muestra el archivo CustomerInterface.wsdl abierto en el editor de WSDL:



Los ejemplos anteriores ilustran el siguiente soporte:

Tabla 20. Soporte de artefacto WSDL

Soporte de WSDL	**Artefacto XSD en línea del ejemplo anterior**
Tipos complejos definidos en el nivel de raíz	Customer
Los elementos definidos en el nivel raíz con tipos complejos anónimos Y el nombre del elemento *no* contienen los nombres de ninguna operación/mensaje (ya que estos podrían ser los elementos envueltos-doc-lit de los que WebSphere Integration Developer podría eliminar el envoltorio automáticamente)	Address
Los elementos definidos en el nivel raíz con tipos sencillos definidos por el usuario	addressNumber

3. **Ejemplo de tiempo de ejecución**

   Los ejemplos anteriores ilustran el siguiente soporte de tiempo de ejecución:

Tabla 21. Soporte de artefacto de tiempo de ejecución

Soporte de tiempo de ejecución	**Artefacto XSD o XSD en línea en los ejemplos anteriores**
Todo lo anterior en los ejemplos 1 y 2 (excepto para addressNumber ya que los tipos sencillos no son objetos de negocio)	Véase arriba (ejemplos 1 y 2)
Los elementos definidos en el nivel raíz que hacen referencia a un tipo complejo	ACMECustomer (mostrado en los ejemplos 1 y 2)

Objetos de negocio sencillos y jerárquicos:

A los objetos de negocio se les puede dar un modelo sencillo o uno jerárquico.

Objeto de negocio sencillo

Un *objeto de negocio sencillo* contiene uno o más atributos sencillos y una lista de verbos soportados. Un atributo sencillo representa un valor como, por ejemplo, Serie o Entero o Fecha. Todos los atributos sencillos tienen una solacardinalidad. Si el objeto de negocio es un objeto de negocio específico de la aplicación, un objeto de negocio sencillo puede representar una entidad en una aplicación o en un estándar de tecnología.

Objeto de negocio jerárquico

Las definiciones de *objeto de negocio jerárquico* definen la estructura de varias entidades relacionadas, que encapsulan no sólo cada entidad individual, sino que también aspectos de la relación entre las entidades. Además de contener, como mínimo, un atributo sencillo, el objeto empresarial jerárquico tiene uno o más atributos que son complejos (es decir, el propio atributo contiene uno o más objetos empresariales, denominados *objetos empresariales hijo*). El objeto de negocio que contiene el atributo complejo se denomina *objeto de negocio padre*.

Existen dos tipos de relaciones entre los objetos empresariales padre e hijo.

- Una sola cardinalidad: cuando un atributo de un objeto empresarial padre representa un único objeto empresarial hijo. El tipo de atributo se establece en el nombre del objeto de negocio hijo y la cardinalidad se establece en uno.
- Varias cardinalidades: cuando un atributo del objeto de negocio padre representa una matriz de objetos de negocio hijo. El tipo del atributo se establece en el nombre del objeto de negocio hijo y la cardinalidad se establece en *n*.

A su vez, cada objeto de negocio hijo puede contener atributos que contienen un objeto de negocio hijo, o una matriz de objetos de negocio, etc. El objeto de negocio en la parte superior de la jerarquía, que no tiene ningún padre, se denomina *objeto de negocio de nivel superior*. Cualquier objeto de negocio único, independiente de sus objetos de negocio hijo que podría contener (o podría estar incluido en éste), se denomina *objeto de negocio individual*.

Ejemplo

El siguiente ejemplo ayuda a ilustrar la diferencia entre un objeto de negocio sencillo y un objeto de negocio jerárquico. El diagrama contiene un objeto de negocio sencillo, denominado Product. El objeto de negocio se representa en la memoria con el tipo de objeto de datos de servicio `commonj.sdo.DataObject` (a menos que se haya generado de forma estática). Este objeto empresarial sencillo tiene un conjunto de atributos que se han modelado como tipos sencillos de esquema XML, así como un atributo que se ha modelado como una lista de tipos sencillos.

El diagrama también ilustra un objeto de negocio Product, en combinación con el objeto de negocio ProductCategory, para crear un objeto de negocio jerárquico más complejo. Este objeto de negocio tiene un objeto de negocio de nivel superior (ProductCategory), así como un objeto de negocio incluido (Product).

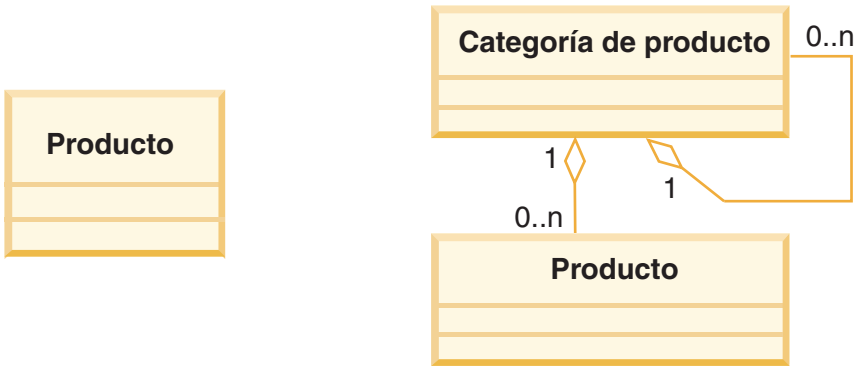


Figura 57. Comparación de objetos de negocio sencillos y jerárquicos

Aquí aparece un ejemplo de la definición del objeto de negocio sencillo para el objeto de negocio Product. El objeto de negocio Product define dos propiedades, *Name* e *Inventory*, tienen los tipos sencillos de esquema XML `xs:string` y `xs:int`. Además, Product también demuestra la definición de la propiedad List *Color*, que es una lista de los tipos sencillos `xs:string`.

```
<schema>
 targetNamespace="http://www.scm.com/ProductTypes"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema">

 <complexType name="Product">
  <sequence>
   <element name="name" type="string"/>
   <element name="inventory" type="int"/>
   <element name="color" type="string" maxOccurs="unbounded"/>
  </sequence>
   </complexType>
</schema>
```

Aquí aparece un ejemplo del objeto de negocio jerárquico ProductCategory. La definición define dos objetos de negocio diferentes, ProductCategory y Product. El objeto de negocio jerárquico ProductCategory define la propiedad, *Name* y, también, define una lista de objetos de negocio del tipo `Product` o `ProductCategory`.

```
<schema>
 targetNamespace="http://www.scm.com/ProductCategoryTypes"
 xmlns:pc="http://www.scm.com/ProductCategoryTypes"
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
   xmlns="http://www.w3.org/2001/XMLSchema">
 elementFormDefault="qualified">

 <complexType name="ProductCategory">
  <sequence>
   <element name="name" type="string"/>
   <choice>
    <element name=="productCategory"
     type="pc:ProductCategory"
     maxOccurs="unbounded"/>
    <element name=="product"
     type="pc:Product"
     maxOccurs="unbounded"/>
```

```
   </choice>
  </sequence>
   </complexType>

 <complexType name="Product">
  <sequence>
   <element name="name" type="string"/>
   <element name=="inventory" type="int"/>
   <element name=="color" type="string" maxOccurs="unbounded"/>
  </sequence>
   </complexType>

</schema>
```

Características de objeto de negocio:

Los objetos de negocio tienen características inherentes que amplían su uso dentro de la infraestructura del objeto de negocio.

Cardinalidad

La cardinalidad de las propiedades está definida a través de los facets `minOccurs` y `maxOccurs` del esquema XML estándar para los tipos sencillos y complejos y el atributo `use` para los atributos.

Valores de propiedad predeterminados

La capacidad para proporcionar los valores predeterminados en el esquema XML para los atributos y los tipos sencillos en un objeto de negocio está soportada por la infraestructura del objeto de negocio. Este soporte está visible durante la creación cuando los tipos de propiedad sencillos de un objeto empresarial reflejan sus valores predeterminados.

Puede adoptar un valor de cero

Se puede definir un elemento en un esquema XML para que tenga un valor nulo. La infraestructura del objeto de negocio permite establecer las propiedades que *pueden adoptar el valor cero* en un valor `Null` durante el tiempo de ejecución.

Definición de clave

Varios subsistemas pueden utilizar la información de clave del objeto de negocio como, por ejemplo, relaciones, la secuenciación y el aislamiento. Sin embargo, cada uno de estos subsistemas puede definir su propio mecanismo de clave independiente de la definición clave del objeto empresarial. Puesto que el lenguaje del modelo subyacente que ha sacado el máximo partido a través de los objetos empresariales es el esquema XML, existe un soporte de primera clase para las definiciones de clave dentro del lenguaje de modelo. Sin embargo, este soporte dentro del lenguaje de modelado no está completamente soportado en el tiempo de ejecución SDO.

xs:ID, xs:IDREF y xs:IDREFS

Estos tipos se han añadido al esquema XML básicamente para proporcionar una vía de acceso de actualización para los DTD. Cada tipo complejo puede tener entre 0 o 1 elementos/atributos con un tipo como `xs:ID`. Los ID deben ser exclusivos en todo un documento, en contraposición a una clave primaria en una base de datos, por ejemplo, que debe ser exclusiva con respecto al ámbito de la tabla. Como ejemplo, un documento compatible no puede utilizar el mismo valor de ID para

identificar tanto Product, como ProductCategory. A menudo, los elementos sortean esta restricción anexando como prefijo el nombre del tipo complejo al valor de la clave. Un atributo del tipo `IDREF` debe contener un valor que coincida con uno de los valores de ID del documento actual. Además, el esquema XML se proporciona para una construcción que es un elemento que puede tener un tipo para contener una lista de referencias de ID, `xs:IDREFs`.

xs:unique, xs:key, xs:keyref
El esquema XML ha introducido un nuevo estilo que habilita las definiciones de clave y las referencias de clave. La construcción `xs:unique` permite a un usuario definir que 1 o más campos de un elemento debe ser único dentro de un ámbito particular del elemento (que representa todo el documento). La construcción `xs:key` es una variante de `xs:unique` con la limitación adicional que son necesarios los elementos referenciados. La construcción `xs:keyref` se utiliza para identificar que el valor de un elemento debe tener un nombre de clave o de construcción exclusiva.

Las construcciones `unique`, `key` y `keyref` tienen varias ventajas sobre el conjunto `ID`, `IDREF` y `IDREFS`, incluidas:

- Pueden definir claves compuestas.
- Pueden definir restricciones restricción unicidad que son relativas a una parte del documento.

Aunque no es necesario que un objeto de negocio tenga una clave definida, se recomienda encarecidamente. Las aplicaciones pueden utilizar los objetos de negocio que no definen una clave. Este escenario es un modelo de uso común en muchos modelos de uso de aplicaciones céntricas Java EE, donde los JavaBeans se pasan y exponen entre el servlet y los contenedores EJB sin la especificación de una clave. Sin embargo, estos objetos de negocio que no definen una clave no pueden interactuar con los subsistemas que requieren una. Esta situación limita su capacidad de sacar partido de las calidades de servicio de WebSphere Process Server.

Modelado de gráficos de negocio

Los gráficos de negocio están relacionados con los objetos de negocio casi del mismo modo que los SDO DataGraphs están relacionados con los SDO DataObjects. Cuando un objeto de negocio de nivel superior requiere enriquecimiento para poder utilizar los servicios proporcionados por WebSphere Process Server, se envuelven con un gráfico de negocio. El envoltorio del gráfico de negocio proporciona la adición del valor adicional añadiendo las cabeceras de datos para almacenar la información de forma lógica en la memoria, o de forma física cuando se serializa el gráfico de negocio.

Nota: Si está migrando una aplicación de WebSphere InterChange Server o migrando adaptadores, puede necesitar utilizar los gráficos de negocio.

Modelos de uso de gráfico de negocio:

Dos modelos de uso primario representan las capacidades fundamentales proporcionadas por el gráfico de negocio: el soporte delta y la imagen posterior.

El *soporte delta* es la capacidad habilitada por el SDO 1.0, donde se capturan los cambios de un gráfico de objeto de negocio en una cabecera especial llamada *resumen de cambios*.

Una *imagen posterior* es un gráfico de negocio que captura el estado actual de datos empresariales en un sistema EIS, normalmente, como resultado de un cambio en dichos datos del EIS. Una imagen posterior permite que los cambios en los sistemas EIS sean capturados y publicados en el tiempo de ejecución.

Para proporcionar estos dos conceptos fundamentales, los gráficos de negocio presentan y proporcionan los siguientes conceptos:

- Un *gráfico de negocio con plantilla* es un gráfico de negocio que tiene un tipo específico para un tipo de gráfico de objeto negocio que lo envuelve.
- El *resumen de cambios* se proporciona para capturar los cambios implícitos, los cambios explícitos, para ambos Delta y, también, para los modelos de uso de imagen posterior.
- El soporte de *resumen de cambios explícitos* es proporcionado por las intefaces de programación de gráfico de negocio que habilitan el componente BPM como, por ejemplo, adaptadores, mediadores, correlaciones y relaciones para modificar explícitamente la cabecera Resumen de cambios.
- El *resumen de sucesos* se proporciona para soportar la captura de anotaciones basadas en instancia sobre los datos del gráfico de objeto de negocio y, específicamente, para contener identificadores de suceso de objeto.
- El gráfico de negocio proporciona un soporte de *verbo* que habilita los componentes en el tiempo de ejecución para eliminar la clave del tipo de suceso para realizar funciones de valor añadido.
- Los *verbos soportados* es la noción de limitar y ampliar el conjunto de verbos autorizados que se puede especificar para un gráfico de negocio.
- Los *identificadores de suceso de objeto* están soportados por el gráfico de negocio permitiendo que se identifiquen de forma exclusiva todos los objetos de un gráfico. Esta capacidad es necesaria para algunos de los componentes que proporcionan características de valor añadido.

Definición de modelo de gráfico de negocio:

Para permanecer de forma no intrusiva en un modelo desarrollado de forma externa de un objeto de negocio, la capacidad del gráfico de negocio se acomoda en el objeto de negocio original. Se utiliza un gráfico de negocio con un nombre de patrón para envolver el objeto de negocio original con el esquema de gráfico de negocio avanzado.

El *gráfico de negocio con plantilla* se crea ampliando el tipo complejo de gráfico de negocio proporcionado por el tiempo de ejecución de la infraestructura del objeto de negocio y añadiendo un elemento que delega el objeto de negocio original. El diagrama muestra un modelo UML para el gráfico de negocio. El gráfico de negocio es abstracto, lo que proporciona sólo el conjunto estándar de cabeceras que se añaden al objeto de negocio de nivel superior.

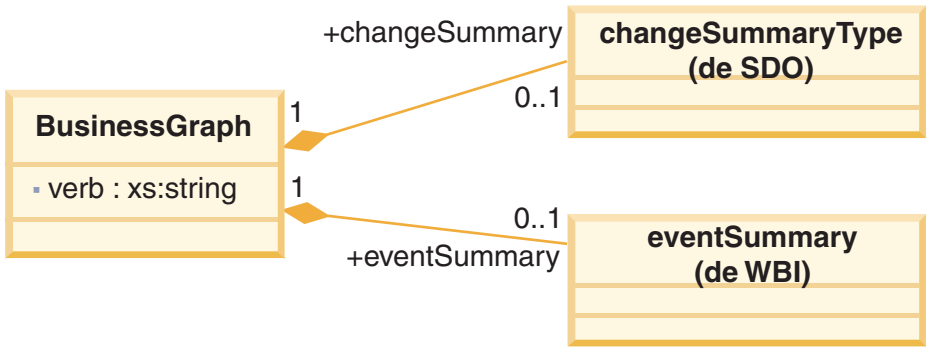


Figura 58. Tipo complejo de gráfico de negocio

El modelo de esquema XML para el tipo complejo XML de gráfico de negocio abstracto:

```
<schema
 targetNamespace="http://www.ibm.com/xmlns/prod/websphere/bo/6.0.0"
 xmlns:bo="http://www.ibm.com/xmlns/prod/websphere/bo/6.0.0"
 xmlns:sdo="commonj.sdo"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">

 <import namespace="commonj.sdo" schemaLocation="DataGraph.xsd"/>

 <complexType name="BusinessGraph" abstract="true">
  <sequence>
   <element name="changeSummary" type="sdo:ChangeSummaryType"
    minOccurs="0" maxOccurs="1"/>
   <element name="eventSummary" type="bo:EventSummary"
    minOccurs="0" maxOccurs="1"/>
   <element name="property" type="bo:ValuesType"
    minOccurs="0"/>
  </sequence>
  <anyAttribute namespace="##other" processContents="lax"/>
 </complexType>

 <complexType name="EventSummary">
  <sequence>
   <any namespace="##any" processContents="lax"
    minOccurs="0" maxOccurs="unbounded"/>
  </sequence>
 </complexType>

 <complexType name="ValuesType">
  <complexContent>
   <extension base="ecore:EClass"/>
  </complexContent>
 </complexType>

 <attribute name="name" type="string"/>
</schema>
```

Los gráficos de negocio sólo son un concepto de nivel superior porque existen para añadir un conjunto de cabeceras a un objeto de nivel de nivel superior existente, y no se puede crear un modelo en un patrón de diseño recursivo, a diferencia de los

objetos de negocio que sí pueden hacerlo. Los gráficos de negocio se pueden aplicar a cualquier objeto de negocio, pero en una aplicación, dicho objeto de negocio pasa a ser un objeto de negocio de nivel superior.

Esto es un ejemplo de un gráfico de negocio que envuelve un objeto de negocio denominado ProductCategory, ProductCategory es un objeto de negocio jerárquico que contiene un objeto de negocio hijo denominado Product.

```
<schema
 targetNamespace="http://www.scm.com/ProductCategoryTypes/ProductCategoryBG"
 xmlns:pcbg="http://www.scm.com/ProductCategoryTypes/ProductCategoryBG"
 xmlns:pc="http://www.scm.com/ProductCategoryTypes"
 xmlns:bo="http://www.ibm.com/xmlns/prod/websphere/bo/6.0.0"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">

 <import namespace="http://www.ibm.com/xmlns/prod/websphere/bo/6.0.0"
  schemaLocation="BusinessGraph.xsd"/>

 <import namespace="http://www.scm.com/ProductCategoryTypes"
  schemaLocation="ProductCategoryTypes.xsd"/>

 <complexType name="ProductCategoryBG">
  <complexContent>
   <extension base="bo:BusinessGraph">
      <sequence>
     <element name="verb" minOccurs="0" maxOccurs="1"/>
     <element name="productCategory"
      type="pc:ProductCategory"
      minOccurs="0" maxOccurs="1"/>
    </sequence>
   </extension>
  <complexContent>
   <complexType>
</schema>
```

El patrón recomendado para un gráfico de negocio con plantilla generado es:

- El gráfico de negocio con plantilla se define utilizando un tipo complejo con nombre que amplía el esquema `bo:BusinessGraph` por restricción (este patrón proporciona restricciones en el valor Verb permitido.
- El nombre del gráfico de negocio con plantilla es el nombre del objeto de negocio de nivel superior, con la serie “BG” anexada.
- El espacio de nombres de destino de destino del gráfico de negocio con plantilla está compuesto por el espacio de nombres de destino del objeto de negocio que se está envolviendo, seguido por un “/”, denominado a través del tipo complejo del gráfico de negocio con plantilla.
- La definición del tipo complejo del gráfico de negocio con plantilla se coloca en su propio archivo de esquema XML cuyo nombre se corresponde al nombre del tipo complejo.

Instancia de modelo de gráfico de negocio:

El gráfico de negocio de nivel superior se representa en la memoria casi del mismo modo que un objeto de negocio, por un SDO 1.0 DataObject, específicamente, la clase `commonj.sdo.DataObject`.

El gráfico de negocio está compuesto por más de simplemente un objeto de contención de gráfico de negocio. También contiene dos cabeceras, además del

objeto de negocio de nivel superior. Ninguna de las cabeceras se representa en la memoria como DataObject y ninguna de las cabeceras permite un mecanismo de acceso de API DataObject.

Cabecera de resumen de cambios

Una prestación proporcionada por los gráficos de negocio es la capacidad de rastrear implícitamente los cambios en el objeto de negocio, ya que el gráfico de negocio se pasa entre varios procesos de negocio distintos. Cuando un proceso cambia en el gráfico de negocio, se genera un registro de cambios en la memoria. Una vez serializado el gráfico de negocio, el registro de cambios se escribe en un formato que habilita el proceso de texto para ver los tipos de cambios que se han realizado en el gráfico de negocio. Esta técnica habilita el adaptador y los servicios de mediador de datos para actualizar de forma eficaz sus almacenes de datos de persistencia optimizando los datos en los que se centra.

Además, el resumen de cambios también se utiliza cuando un adaptador genera un suceso de imagen posterior para describir los datos que se actualizaron en un sistema EIS. En particular, la capacidad del resumen de cambios para anotar los cambios de objeto y propiedad se utiliza en el modelo de uso Imagen posterior (así como también el verbo para el gráfico de negocio).

Uso del resumen de cambios implícitos

Cuando las aplicaciones provocan cambios en el objeto de negocio, las aplicaciones pueden activan el registro de cambios, de forma que los sucesos de cambio se registran automáticamente en el resumen de cambios. El resumen de cambios se incluye en el gráfico de negocio en un nivel de igual del objeto de negocio de nivel superior. Los sucesos de cambio se definen en dos niveles, para los objetos de negocio y para las propiedades de objeto de negocio. Los objetos de negocio tienen tipos de cambio crear, actualizar y suprimir, mientras que las propiedades tienen tipos de cambio establecer y no establecer. Los sucesos de cambio sólo se rastrean para las modificaciones en la parte del objeto de negocio del gráfico de negocio. Un cambio en el resumen de sucesos no genera ninguna actualización implícita del resumen de cambios para el gráfico de negocio.

Modificación del resumen de cambios explícitos

Hay varios modelos de uso en los componentes de WebSphere Process Server que requieren la capacidad de escribir de forma explícita en la cabecera del resumen de cambios. Por ejemplo, un adaptador que genera un sucesos de EIS crea explícitamente los tipos de cambio de objeto y, potencialmente, los tipos de cambio de propiedad. Una correlación de objeto de negocio específico de aplicación/un objeto de negocio general (ASBO/GBO) transforma el resumen de cambios de un gráfico de negocio a otro, creando una nueva versión del resumen de cambios en un gráfico de negocio de salida. Esta prestación es proporcionada por la infraestructura del objeto de negocio.

Cabecera del resumen de sucesos

El resumen de sucesos proporciona el **ObjectEventID**, que es el mecanismo utilizado para identificar de forma exclusiva una instancia de un objeto que aparece en el tiempo de ejecución. Esta información se transporta en el resumen de sucesos, donde el identificador exclusivo se asocia a un DataObject dado en la jerarquía de objetos de negocio del gráfico de negocio.

La información de suceso también se puede transportar en el resumen de sucesos. Esta información es una serie que se puede utilizar para añadir metadatos adicionales asociados a cada objeto de la jerarquía de objetos de negocio para el gráfico de negocio. Un modelo de uso potencial para la información de sucesos es marcar los objetos de negocio contenidos con un verbo que no sea ninguno de los otros verbos estándar Create, Update y Delete soportados por el resumen de cambios.

Cabecera de verbo

Si el verbo se establece en el gráfico de negocio, la parte de los datos del objeto de negocio del gráfico d e negocio transporta un conjunto de datos de imagen posterior de EIS. Si el verbo contiene un valor, existen tres posibilidades con respecto a la granularidad del suceso de imagen posterior:

- El resumen de cambios está vacío. Esta situación se produce cuando un EIS conoce el tipo de actualización en un conjunto de datos, pero no tiene específicos en los que se hayan creado, actualizado o suprimido los objetos en el gráfico. El resultado es que una mediación, correlación, relación o adaptador en sentido descendente debe utilizar la información del gráfico de negocio además de los datos adicionales para determinar la actualización real para realizar.
- El resumen de cambios tiene anotaciones de suceso de cambio de nivel de objeto. Este caso es típico donde el sistema EIS reconoce lo que ha sucedido en cada objeto del gráfico de negocio, pero no tiene granularidad para determinar si las propiedades específicas de los objetos se han actualizado.
- El resumen de cambios tiene anotaciones de suceso de cambio de nivel de objeto y anotaciones get/set de nivel de propiedad. Esta situación es el caso más granular, y todos los adaptadores se esfuerzan por obtener este nivel de una imagen posterior si su sistema EIS hace que estén disponibles los datos apropiados. La ventaja de una imagen posterior especificada por completo es que permite que se produzca una gestión de sucesos de cambio de propiedad implícita. Por lo tanto, es mucho más fácil para los gráficos de negocio de imagen posterior interoperar con los gráficos de negocio basados en Delta desconectados.

Modelado de metadatos de tipo de objeto de negocio

Los metadatos de tipo de objeto de negocio se pueden añadir a definiciones de objeto de negocio para ampliar su valor en el tiempo de ejecución. La infraestructura de objeto de negocio le permite diseñar, anotar y convertir metadatos de tipo de objeto de negocio.

La infraestructura de objeto de negocio proporciona un mecanismo que permite:

- A los metadatos combinarse con un objeto de negocio de una forma coherente y relativamente no intrusiva
- Una política preceptiva para los desarrolladores para definir las estructuras de anotación compleja
- Una política preceptiva para los desarrolladores y desplegadores de objeto de negocio para anotar un objeto de negocio con metadatos de instancia que cumplen con las estructuras de anotación compleja predefinidas
- Un conjunto de las API para transformar las anotaciones durante la ejecución en una estructura DataObject fácil de utilizar

Este proceso implica, como mínimo, tres roles diferentes:

- El primer rol es el del diseñador de metadatos de tipo de objeto de negocio. Este rol diseña la estructura de los metadatos. Por ejemplo, la infraestructura del

objeto de negocio desempeña este rol y define un par de características de metadatos para anotar un objeto de negocio. Se espera que los adaptadores como PeopleSoft, Siebel y SAP desempeñen el rol del diseñador de metadatos para anotar el objeto de negocio con información específica de la aplicación.

- El segundo rol es el del diseñador o desplegador de objeto de negocio. Este rol utiliza la estructura de los metadatos del tipo de objeto de negocio y la política definida por la infraestructura de metadatos del tipo de objeto de negocio para anotar las definiciones de objeto de negocio con metadatos.
- Si el objeto de negocio se ha anotado correctamente, el tercer rol puede utilizar las API de metadatos de objeto de negocio para validar e inspeccionar los metadatos durante la ejecución, y convertirlos en una estructura de gráfico DataObject práctica y navegable.

Además, la infraestructura de objeto de negocio se proporciona para las siguientes características de metadatos:

- Definiciones de propiedad de clave primaria compuesta y clave foránea
- Anotaciones de metadatos de verbo soportado de nivel superior

Si desea más detalles, consulte estos otros temas.

Representación de metadatos del tipo de objeto de negocio:

La infraestructura del objeto de negocio define un mecanismo a través del cual los metadatos del tipo de objeto de negocio se pueden combinar en un esquema importado descendente, o desarrollado de forma externa.

El mecanismo que se utiliza para combinar los metadatos del tipo de objeto de datos son las anotaciones de esquema XML y la estructura `appInfo`. No obstante, esta política requiere la modificación de la definición del esquema original, lo hace de una forma que se habilitan las anotaciones y `appinfo` para conmutar fácilmente entre vistas o para su eliminación completa. Este esquema de identificación se realiza utilizando el atributo de origen en el código `xs:appinfo` cuyo valor es el espacio de nombres de destino que define los metadatos de tipo de objeto de negocio.

Por ejemplo, asuma que el siguiente esquema se ha importado en el tiempo de ejecución. El esquema describe un objeto de negocio, denominado Product, que incluye anotaciones de cliente.

```
<schema>
 targetNamespace="http://www.scm.com/ProductTypes"
 xmlns:p="http://www.scm.com/ProductTypes"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
   elementFormDefault="qualified

 <complexType name="Product">
  <annotation>
   <appInfo>
    <SCMEditor value="Bottom" type="Anchor"/>
     </appInfo>
   <documentation>
     Describe el Producto SCM
   </documentation>
  </annotation>

  <sequence>
   <element name="id" type="ID"/>
   <element name="description" type="string" default="DefaultDescription"/>
```

```
   <element name="sku" type="p:Sku"/>
  </sequence>
   </complexType>

 <simpleType name="Sku">
  <restriction base="string">   <pattern value="\d{3}-[A-Z]{2}"/>
    </restriction>
</simpleType> </schema>
```

Diseño de metadatos de tipo de objeto empresarial:

Puede diseñar una estructura de metadatos para que albergue metadatos de tipo de objeto empresarial.

Acerca de esta tarea

Para diseñar la estructura de metadatos, siga estos pasos.

Procedimiento

1. Cree un archivo de esquemas XML con un espacio de nombres de destino válido. Este paso se utiliza cuando se añaden los metadatos de la instancia a la definición del objeto empresarial.
2. Defina un tipo complejo denominado para definir cada parte independiente de metadatos que pueda añadirse a un objeto empresarial. La definición de tipo complejo se utiliza para definir la estructura de `DataObject` escrito dinámicamente que se utiliza para leer los metadatos de la instancia. El nombre del tipo complejo se utiliza cuando se añaden los metadatos de la instancia a la definición del objeto empresarial.

Ejemplo

Este ejemplo muestra una porción de metadatos de tipo de objeto empresarial desarrollada para un adaptador PeopleSoft.

```
<schema>
 targetNamespace="http://www.ibm.com/xmlns/prod/websphere/adapter/psft/
  PSFTBODefinitionASI/7.0.0"
 xmlns:psft="http://www.ibm.com/xmlns/prod/websphere/adapter/psft/
  PSFTBODefinitionASI/7.0.0"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">

 <complexType name="PSFTBODefinitionASI">
  <sequence>
   <element name="hostname" type="string"/>
   <element name="ipaddress" type="string"/>
  </sequence>
 </complexType>
</schema>
```

Anotación de una definición de objeto de empresa:

Puede utilizar la sintaxis proporcionada por la infraestructura de objeto empresarial para anotar una definición de objeto empresarial.

Acerca de esta tarea

La infraestructura de objeto empresarial no intenta definir el mecanismo por el cual los datos de metadatos de la instancia son conectados a la definición del objeto empresarial. Sin embargo, define la sintaxis de los metadatos de la instancia.

Para anotar la definición del objeto empresarial siga estas pautas.

Procedimiento

1. Si la parte de la definición del objeto empresarial a la que van a conectarse los metadatos de la instancia todavía no tiene una anotación, añada una. Si ya tiene una anotación, utilice la existente.
2. Cree un código `xs:appinfo` separado dentro del código `xs:annotation`, y defina el atributo de origen al espacio de nombres definido por los metadatos del tipo de objeto empresarial que se añaden.
3. Dentro del código `xs:appinfo`, defina QName utilizando un prefijo de espacio de nombres de destino seguido del nombre de la definición de tipo complejo. Anexe el atributo QName definido con `xmlns:` seguido por el prefijo de espacio de nombres de destino utilizado anteriormente. Establezca el valor de este atributo QName en el espacio de nombres de destino de los metadatos del tipo de objeto empresarial que se están utilizando.
4. Añada los datos de la estructura y de la instancia que proceden de la definición de los metadatos del tipo de objeto empresarial. Cierre todos los códigos si es necesario.

Ejemplo

Cuando se importa el objeto, se define para que necesite un conjunto de anotaciones de PeopleSoft. Este ejemplo muestra la misma definición de objeto empresarial con los metadatos de PeopleSoft añadidos.

```
<schema>
 targetNamespace="http://www.scm.com/ProductTypes"
 xmlns:p="http://www.scm.com/ProductTypes"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified

 <complexType name="Product">
  <annotation>
   <appinfo source="http://www.ibm.com/xmlns/prod/websphere/Adapter/PSFT">
    <psft:PSFTMetadata
     xmlns:psft="http://www.ibm.com/xmlns/prod/websphere/Adapter/PSFT">
     <hostname>mumbai</hostname>
     <ipaddress>9.29.1.1</ipaddress>
    <psft:PSFTMetadata>
   </appInfo>
   <appInfo>
    <SCMEditor value="Bottom" type="Anchor"/>
   </appInfo>
   <documentation>
    Describes the SCM Product
   </documentation>
  </annotation>

  <sequence>
   <element name="id" type="ID"/>
   <element name="description" type="string" default="DefaultDescription"/>
   <element name="sku" type="p:Sku"/>
  </sequence>
 </complexType>
```

```
 <simpleType name="Sku">
  <restriction base="string">   <pattern value="\d{3}-[A-Z]{2}"/>
  </restriction>
</simpleType> </schema>
```

Conversión de una anotación en DataObjects:

La infraestructura de objeto empresarial ofrece la posibilidad de transformar anotaciones en una estructura DataObject utilizable.

Acerca de esta tarea

Las anotaciones asociadas con una definición de objeto empresarial pueden leerse en tiempo de ejecución utilizando el conjunto API específicas de implementación de SDO. Sin embargo, el problema con estas API es que devuelven un BLOB (Binary Large Object). No obstante, la infraestructura de objeto empresarial ofrece una utilidad que, si se han seguido los patrones de anotaciones recomendados, lee el BLOB, lo valida y lo transforma en una estructura DataObject utilizable.

Para convertir una anotación en DataObject.

Procedimiento

1. Obtenga una anotación.
2. Utilice BOTypeMetadata para convertir dicha anotación en DataObject de SDO. La implementación BOTypeMetadata está disponible como singleton utilizando la instancia BOTypeMetadata.INSTANCE.

Ejemplo

El ejemplo siguiente muestra cómo utilizar las API para obtener una anotación y después utilizar la API BOTypeMetadata para convertir dicha anotación en DataObject de SDO. Los metadatos se definen en `BOTypeMetadata.xsd`.

```
<schema>
 targetNamespace="http://www.ibm.com/xmlns/prod/websphere/botm/7.0.0"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">

 <complexType name="VerbInfo">
  <sequence>
   <element name="verbInfo" type="string"/>
  </sequence>
 </complexType>
</schema>
```

Una de las funciones de la infraestructura de objeto empresarial es la adición de verbos soportados adicionales y sus metadatos correspondientes para habilitar capacidades controladas por metadatos en tiempo de ejecución. La capacidad se soporta mediante la utilización de verbos y los metadatos de VerbInfo para anotar los verbos con metadatos adicionales. El ejemplo siguiente muestra los lugares donde los metadatos de VerbInfo serían añadidos para cada valor de posible de verbo.

```
<schema
 targetNamespace="http://www.scm.com/ProductCategoryTypes/ProductCategoryBG"
 xmlns:pcbg="http://www.scm.com/ProductCategoryTypes/ProductCategoryBG"
 xmlns:pc="http://www.scm.com/ProductCategoryTypes"
 xmlns:sdo="commonj.sdo"
 xmlns:bo="http://www.ibm.com/xmlns/prod/websphere/bo/7.0.0"
```

```
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified">

 <import namespace="http://www.ibm.com/xmlns/prod/websphere/bo/7.0.0"
  schemaLocation="BusinessGraph.xsd"/>

 <import namespace="commonj.sdo"
  schemaLocation="DataGraph.xsd"/>

 <import namespace="http://www.scm.com/ProductCategoryTypes"
  schemaLocation="ProductCategoryTypes.xsd"/>

 <complexType name="ProductCategoryBG">
  <complexContent>
   <extension base="bo:BusinessGraph">
    <sequence>

     <element name="verb"
      minOccurs="0" maxOccurs="1">
      <simpleType>
 <restriction base="string">        <enumeration value="Create">
        <annotation>
         <appinfo source="http://www.ibm.com/xmlns/prod/websphere/botm/7.0.0">
          <botm:VerbInfo
           xmlns:botm="http://www.ibm.com/xmlns/prod/websphere/botm/7.0.0">
           <verbInfo>Metadata relating to Create</verbInfo>
          </botm:VerbInfo>
         </appinfo>
        </annotation>
       </enumeration>
       <enumeration value="Retrieve">
        <annotation>
         <appinfo source="http://www.ibm.com/xmlns/prod/websphere/botm/7.0.0">
          <botm:VerbInfo
           xmlns:botm="http://www.ibm.com/xmlns/prod/websphere/botm/7.0.0">
           <verbInfo>Metadata relating to Retrieve</verbInfo>
          </botm:VerbInfo>
         </appinfo>
        </annotation>
       </enumeration>
      </restriction>
     </simpleType>
    </element>

    <element name="productCategory" type="pc:ProductCategory"
     minOccurs="0" maxOccurs="1"/>

   </sequence>
  </extension>
 </complexContent>
</complexType>

 <element name="productCategoryBG" type="pcbg:ProductCategoryBG"/>

</schema>
```

Programación mediante los servicios de objeto de negocio

Para facilitar la creación y manipulación de objetos de negocio, la infraestructura del objeto de negocio amplía las especificaciones SDO proporcionando un conjunto de servicios Java. Estos servicios forman parte del paquete denominado `com.ibm.websphere.bo`.

Esto es una breve descripción de los servicios de objeto de negocio.

Tabla 22. Servicios de objeto empresarial

Servicio	Descripción
BOChangeSummary	Proporciona mejoras a la interfaz del resumen de cambios SDO para gestionar la cabecera del resumen de cambios del gráfico de negocio.
BOCopy	Facilita la copia de un gráfico de objetos de negocio o un gráfico de negocio que contiene un gráfico de objetos de negocio.
BODataObject	Proporciona mejoras a la interfaz del objeto de datos SDO.
BOEquality	Proporciona la capacidad para determinar si dos gráficos de negocio u objetos de negocio son equivalentes.
BOEventSummary	Proporciona la interfaz para gestionar el contenido de la cabecera del resumen de sucesos del gráfico de negocio.
BOFactory	Proporciona la capacidad para crear un gráfico de negocio o un objeto de negocio.
BOType	Proporciona un mecanismo para obtener el tipo SDO de un gráfico de negocio o un objeto de negocio que duplica lo que proporciona `Class.forName()` para los nombres de clase Java.
BOTypeMetadata	Proporciona la capacidad de tomar un objeto binario grande de anotación (BLOB) que es compatible con el patrón de metadatos del tipo BO y lo transforma en un conjunto de DataObjects SDO (y realiza la transformación de reserva).
BOXMLDocument/BOXMLSerializer	Proporciona los mecanismos para crear y representar un documento XML en memoria, y para serializar y deserializar un documento XML.

Tabla 22. Servicios de objeto empresarial (continuación)

Servicio	Descripción
BOInstanceValidator	Proporciona un recurso para validar una instancia de objeto de negocio respecto a su definición XSD. Los objetos de negocio pueden estar presentes con distintos formatos. Pueden ser objetos de negocio sencillos o pueden tener el envoltorio de un modelo de gráfico de negocio avanzado. En determinados escenarios de Business Integration, los objetos de negocio están en la sección suprimida del resumen de cambios. Estos objetos de negocio controlan las lógicas de negocio en sentido descendente. La precisión de los objetos de negocio se debe garantizar en todos los casos. Existen dos estilos soportados para BOInstanceValidator: • Validación programática explícita: se proporciona un servicio de sistema para validar los objetos de negocio a través de un conjunto de las API de programación. • Validación de interfaz implícita: esta validación está habilitada/inhabilitada a través de WebSphere Integration Developer en las interfaces de destino a través de un calificador de interfaz SCA.

Validación de documentos XML

Los documentos XML y los objetos empresariales se pueden validar mediante el servicio de validación.

Además, otros servicios requieren ciertos estándares mínimos o se genera una excepción de tiempo de ejecución. Uno de éstos es `BOXMLSerializer`.

Puede utilizar `BOXMLSerializer` para validar documentos antes de que los procese una petición de servicio. `BOXMLSerializer` valida la estructura de los documentos XML para determinar si hay presente alguno de los tipos de errores siguientes:

- Los documentos XML no válidos como, por ejemplo, aquellos que en los que faltan ciertos códigos de elemento.
- Los documentos XML con formato incorrecto como, por ejemplo, aquellos en los que faltan los códigos de cierre.
- Documentos que contienen errores de análisis como, por ejemplo, errores en la declaración de la entidad.

Cuando `BOXMLSerializer` descubra un error, se generará una excepción en la que se especificarán los detalles del problema.

La validación se puede llevar a cabo para la importación y exportación de documentos XML, para los servicios siguientes:

- HTTP
- Servicios Web JAXRPC
- Servicios Web JAX-WS
- Servicios JMS

- Servicios MQ

Para los servicios HTTP, JAXRPC y JAX-WS, `BOXMLSerializer` generará las excepciones del modo siguiente:

- Importaciones –
  1. El componente SCA invoca el servicio.
  2. El servicio invoca un URL de destino.
  3. El URL de destino responde con una excepción XML no válida.
  4. El servicio falla, generando una excepción de tiempo de ejecución y un mensaje.
- Exportaciones –
  1. El cliente del servicio invoca la exportación del servicio.
  2. El cliente del servicio envía datos XML no válidos.
  3. La exportación no se puede realizar para este servicio y genera una excepción y un mensaje.

En el caso de los servicios de mensajería JMS y MQ, las excepciones se generan del modo siguiente:

- Importaciones –
  1. La importación invoca el servicio JMS o MQ.
  2. El servicio devuelve una respuesta.
  3. El servicio devuelve una excepción XML no válida.
  4. La importación falla y genera un mensaje.
- Exportaciones –
  1. El cliente MQ o JMS invoca una exportación.
  2. El cliente envía datos XML no válidos.
  3. La exportación falla y genera una excepción y un mensaje.

Puede ver los registros para cualquiera de los mensajes generados por una excepción de validación XML. Los ejemplos siguientes son mensajes generados por código XML inadecuado que `BOXMLSerializer` ha validado.

- Importación JAXWS

```
javax.xml.ws.WebServiceException: org.apache.axiom.om.OMException:
 javax.xml.stream.XMLStreamException: el tipo de elemento "TestResponse" debe ir
 seguido de una de las especificaciones de atributo siguientes: ">" o "/>".

javax.xml.ws.WebServiceException: org.apache.axiom.soap.SOAPProcessingException:
 El primer elemento debe contener el nombre local, Envelope
```

- Importación JAXRPC

```
[9/11/08 15:16:27:417 CDT] 0000003e ExceptionUtil E
 CNTR0020E: EJB ha generado una excepción (no declarada)
  inesperada durante la invocación del método
  "transactionNotSupportedActivitySessionNotSupported" en el bean
  "BeanId(WSXMLValidationApp#WSXMLValidationEJB.jar#Module, null)".
  Datos de la excepción: WebServicesFault
 faultCode: {http://schemas. xmlsoap.org/soap/envelope/}Server.generalException
 faultString: org.xml.sax.SAXParseException: el tipo de elemento "TestResponse"
   debe ir seguido de una de las especificaciones de
  atributo siguientes: ">" o "/>". Mensaje que se analiza:
    <?xml version="1.0"?><TestResponse
  xmlns="http://WSXMLValidation"<firstName>Bob</firstName>
    <lastName>Smith</lastName></TestResponse>
 faultActor: null
 faultDetail:
```

```
[9/11/08 15:16:35:135 CDT] 0000003f ExceptionUtil E   CNTR0020E: EJB ha generado una
  excepción (no declarada) inesperada durante la invocación del método
  "transactionNotSupportedActivitySessionNotSupported" en el bean
  "BeanId(WSXMLValidationApp#WSXMLValidationEJB.jar#Module, null)".
 Datos de la excepción: WebServicesFault
 faultCode: {http://schemas. xmlsoap.org/soap/envelope/}Server.generalException
 faultString: org.xml.sax.SAXException: WSWS3066E: Error: Se esperaba 'envelope'
   pero se ha encontrado TestResponse.
  Mensaje que se analiza: <?xml version="1.0"?><TestResponse
   xmlns="http://WSXMLValidation">
  <firstName>Bob</firstName><middleName>John</middleName>
   <lastName>Smith</lastName>
   </TestResponse>
 faultActor: null
 faultDetail:
```

- Exportación JAXRPC/JAXWS

```
[9/11/08 15:35:13:401 CDT] 00000064 WebServicesSe E
  com.ibm.ws.webservices.engine.transport.http.WebServicesServlet
  getSoapAction WSWS3112E:
   Error: Se está generando WebServicesFault porque falta SOAPAction.
                   WebServicesFault
 faultCode: Client.NoSOAPAction
 faultString: WSWS3147E: Error: ¡falta la cabecera SOAPAction!
 faultActor: null
 faultDetail:
```

Para obtener más información sobre los servicios de validación, consulte la interfaz `BOInstanceValidator`, en la documentación de API y SPI generada, en la sección Consulta.

Técnicas de programación

Estas técnicas ilustran cómo programar de forma eficaz objetos de negocio utilizando la infraestructura del objeto empresarial.

Matrices en los objetos empresariales

Puede definir matrices para un elemento en un objeto empresarial de forma que el elemento pueda contener más de una instancia de datos.

Puede utilizar un tipo List para crear una matriz para un elemento con un nombre único en un objeto empresarial. Esto le permitirá utilizar dicho elemento para que contenga varias instancias de datos. Por ejemplo, puede utilizar una matriz para almacenar varios números de teléfono dentro de un elemento llamado `telephone` que se haya definido como una serie en su envoltorio de objeto empresarial. También puede definir el tamaño de la matriz especificando el número de instancias de datos mediante el valor `maxOccurs`. El código de ejemplo siguiente muestra cómo se crearía una matriz que puede contener tres instancias de datos para dicho elemento:

```
<xsd:element name="telephone" type="xsd:string" maxOccurs="3"/>
```

Esto creará un índice de lista para el elemento `telephone`, que puede contener hasta tres instancias de datos. También puede utilizar el valor `minOccurs` si tiene pensado que la matriz tenga sólo un elemento.

La matriz resultante consta de dos elementos:

- el contenido de la matriz
- la propia matriz.

Para poder crear esta matriz, no obstante, necesita efectuar un paso intermedio definiendo un envoltorio. Este envoltorio, de hecho, sustituye la propiedad del elemento por un objeto empresarial. En el ejemplo anterior, puede volver a crear un objeto `ArrayOfTelephone` para definir el elemento `telephone` como una matriz. El código de ejemplo siguiente muestra cómo se llevaría a cabo esta tarea:

```
<?xml version="1.0" encoding="UTF-8"?>
  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element name="Customer">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="name" type="xsd:string"/>
          <xsd:element name="ArrayOfTelephone" type="ArrayOfTelephone"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="ArrayOfTelephone">
      <xsd:sequence maxOccurs="3">
        <xsd:element name="telephone" type="xsd:string" nillable="true"/>
      </xsd:sequence>
    </xsd:complexType>
 </xsd:schema>
```

Ahora, el elemento `telephone` aparece como un hijo del objeto envoltorio `ArrayOfTelephone`.

Tenga en cuenta que en el ejemplo anterior, el elemento `telephone` contiene una propiedad llamada `nillable`. Puede establecer esta propiedad en `true` si desea que ciertos elementos del índice de la matriz no contengan datos. El código de ejemplo siguiente muestra cómo pueden representarse los datos de una matriz:

```
<Customer>
   <name>Bob</name>
   <ArrayOfTelephone>
     <telephone>111-1111</telephone>
     <telephone xsi:nil="true"/>
     <telephone>333-3333</telephone>
   </ArrayOfTelephone>
 </Customer>
```

En este caso, el primer y el tercer elemento del índice de la matriz para el elemento `telephone` contienen datos, mientras que el segundo elemento no contiene ningún dato. Si no hubiera utilizado la propiedad `nillable` para el elemento `telephone`, tendría que haber hecho que los dos primeros elementos contuvieran datos.

Puede utilizar las API de secuencia de SDO (Service Data Object) en WebSphere Process Server como método alternativo a manejar secuencias en las matrices de objeto empresarial. En el siguiente código de ejemplo se creará una matriz para el elemento `telephone` con datos idénticos a los mostrados más arriba:

```
DataObject customer = ...
 customer.setString("name", "Bob");

 DataObject tele_array = customer.createDataObject("ArrayOfTelephone");
 Sequence seq = tele_array.getSequence();  // La matriz está secuenciada
 seq.add("telephone", "111-1111");
 seq.add("telephone", null);
 seq.add("telephone", "333-3333");
```

Puede devolver los datos para un índice de matriz de elemento dado utilizando un código similar al del ejemplo siguiente:

```
String tele3 = tele_array.get("telephone[3]");  // tele3 = "333-3333"
```

En este ejemplo, una serie llamada tele3 devolverá los datos "333-3333".

Puede especificar los elementos de datos para la matriz del índice de lista utilizando el ancho fijo o los datos delimitados ubicados en una cola de mensaje JMS o MQ. También puede efectuar esta tarea utilizando un archivo de texto sin formato que contenga los datos con el formato correcto.

Creación de objetos empresariales anidados

Puede utilizar la función setWithCreate para crear objetos empresariales anidados dentro de un objeto empresarial padre.

Puede crear objetos empresariales anidados a partir de un objeto empresarial padre sin tener que escribir código en el que se detallen los objetos hijo intermedios. Por ejemplo, puede establecer un objeto empresarial anidado, que esté dos niveles por debajo del objeto padre, sin tener que definir un objeto empresarial dependiente que esté un nivel por debajo del objeto padre. Utilice la función setWithCreate para realizar esta tarea para:

- una única instancia
- varias instancias
- un valor comodín
- un grupo de modelos

En los temas siguientes se describe cómo puede realizar cada una de las mismas.

Instancia individual de un objeto empresarial anidado:

Utilice la función setWithCreate para crear una instancia individual de objeto empresarial anidado.

Antes de empezar

En el código de ejemplo siguiente se muestra cómo, normalmente, tendría que crear código para un objeto (hijo) intermedio a partir de un objeto de nivel superior (padre) para, así, poder crear un objeto de tercer nivel (nieto). El archivo XSD tendría un aspecto parecido al siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="Parent">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="child" type="Child"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Child">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="grandChild" type="GrandChild"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="GrandChild">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Acerca de esta tarea

Si ha utilizado el método "de arriba a abajo" tradicional para establecer los datos de objeto empresarial, tendrá que procesar el siguiente código especificando los objetos hijo y nieto antes de definir los datos en el objeto nieto:

```
DataObject parent = ...
DataObject child = parent.createDataObject("child");
DataObject grandchild = child.createDataObject("grandChild");
grandchild.setString("name", "Bob");
```

Puede utilizar un método más eficiente mediante la función `setWithCreate` para definir, simultáneamente, el objeto nieto y establecer sus datos, sin tener que especificar el objeto hijo intermedio. El código de ejemplo siguiente muestra cómo se llevaría a cabo esta tarea:

```
DataObject parent = ...
parent.setString("child/grandchild/name", "Bob");
```

Resultados

Los datos del objeto empresarial de nivel inferior se establecen sin tener que hacer referencia al objeto empresarial de nivel intermedio. Se produce una excepción si la vía de acceso no es válida.

Creación de varias instancias de objetos empresariales anidados:

Utilice la función `setWithCreate` para crear varias instancias de objeto empresarial anidado.

Antes de empezar

El archivo XSD de ejemplo siguiente contiene objetos anidados ubicados un nivel (hijo) y dos niveles (nieto) por debajo del objeto empresarial superior (padre):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="Parent">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="child" type="Child" maxOccurs="5"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Child">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="grandChild" type="GrandChild"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="GrandChild">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Tenga en cuenta que el objeto padre puede tener hasta cinco objetos hijo, tal como se especifica en el valor `maxOccurs`.

Acerca de esta tarea

Puede crear una lista con una política que sea más rigurosa que no permitirá que falten secuencias en una matriz. Puede utilizar el método `setWithGet` y, al mismo tiempo, especificar los datos que aparecerán en un elemento de índice de lista concreto:

```
DataObject parent = ...
parent.setString("child[3]/grandchild/name", "Bob");
```

En este caso, la matriz resultante sería del tamaño tres, pero los valores de los elementos de índice de lista `child[1]` y `child[2]` no están definidos. Es posible que desee que los elementos sean un valor nulo o bien que tengan un valor de datos asociado. En el escenario anterior, se generará una excepción porque no se han definido los valores para los dos primeros elementos de índice de la matriz.

Puede resolver esta situación definiendo los valores en el índice de la lista. Si el elemento de índice hace referencia a un elemento existente de la matriz y si dicho elemento no es nulo (es decir, contiene datos), se utilizará. Si es nulo, se creará y utilizará. Si el índice de la lista supera en uno el tamaño de la lista, se creará y añadirá un nuevo valor. En el código de ejemplo siguiente se muestra qué ocurrirá en una lista que es del tamaño dos, donde `child[1]` se ha designado como nulo y `child[2]` contiene datos:

```
DataObject parent = ...
// child[1] = nulo
// child[2] = hijo existente
// Este código funcionará porque child[1] es nulo y se creará.
parent.setString("child[1]/grandchild/name", "Bob");

// Este código funcionará porque child[2] existe y se utilizará.
parent.setString("child[2]/grandchild/name", "Dan");

// Este código funcionará porque la lista de hijos es del tamaño 2, y al añadir
// un elemento de lista más, se aumentará el tamaño de la lista.
parent.setString("child[3]/grandchild/name", "Sam");
```

Resultados

Ha alterado temporalmente los valores de los dos elementos existentes y ha añadido un tercer elemento al índice de la lista. Si, no obstante, añade otro elemento que no sea del tamaño cuatro, o que sea mayor que el tamaño especificado en `maxOccurs`, se generará una excepción. La política más rigurosa de este método se demuestra en el código de ejemplo siguiente.

Nota: Se supone que el código que aparece a continuación se ha añadido al código anterior existente:

```
// Este código generará una excepción porque la lista es del tamaño 3
// y no ha creado ningún elemento para aumentar el tamaño hasta 4.
parent.setString("child[5]/grandchild/name", "Billy");
```

Utilización de un objeto empresarial anidado definido mediante un comodín:

Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

Acerca de esta tarea

La función `setWithCreate` utilizada para definir objetos empresariales anidados para una o varias instancias no funciona si utiliza un valor de comodín de `xsd:any` en el objeto de datos de servicio. Esto se ilustra en el siguiente código de ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:complexType name="Parent">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="child" type="xsd:anyType"/>
    </xsd:sequence>
  </xsd:complexType>

</xsd:schema>
```

Resultados

Se generará una excepción si el objeto de datos hijo no existe.

Utilización de objetos empresariales en grupos de modelos:

Cree los patrones de la vía de acceso de grupo de modelos cuando trabaje con objetos de negocio anidados que forman parte de un grupo de modelos.

Acerca de esta tarea

Los modelos de grupos utilizan el código `xsd:choice`, que puede utilizar para crear objetos de negocio a partir de un objeto empresarial padre. No obstante, la infraestructura del objeto de negocio puede provocar conflictos de denominación que pueden generar una excepción. El siguiente código de ejemplo ilustra cómo se puede producir conflictos de denominación:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://MultipleGroup">
  <xsd:complexType name="MultipleGroup">
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="child1" type="Child"/>
        <xsd:element name="child2" type="Child"/>
      </xsd:choice>
      <xsd:element name="separator" type="xsd:string"/>
      <xsd:choice>
        <xsd:element name="child1" type="Child"/>
        <xsd:element name="child2" type="Child"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Nota: puede haber varias instancias de los elementos llamados "child1" y "child2",

Utilice los patrones de vía de acceso de objetos de datos de servicio (SDO) para los grupos de modelos para resolver estos conflictos.

Resultados

Se obtendrán muchas matrices que utilizan el patrón de vía de acceso de SDO que se utiliza para manejar grupos de modelos, tal como se muestra en el siguiente código de ejemplo:

```
set("child1/grandchild/name", "Bob");

set("child11/grandchild/name", "Joe");
```

Diferenciación de elementos con nombre idéntico

Debe proporcionar nombres exclusivos para atributos y elementos de objetos empresariales.

En la infraestructura SDO (Service Data Object), los elementos y atributos se crean como propiedades. En los siguientes ejemplos de código, las XSD crean tipos que tienen una propiedad llamada foo:

```
<xsd:complexType name="ElementFoo">
  <xsd:sequence>
    <xsd:element name="foo" type="xsd:string" default="elem_value"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="AttributeFoo">
  <xsd:attribute name="foo" type="xsd:string" default="attr_value"/>
</xsd:complexType>
```

En estos casos, puede acceder a la propiedad utilizando XPath (XML Path Language). Sin embargo, los tipos de esquema válidos pueden tener un atributo y elemento con el mismo nombre, como en el siguiente ejemplo:

```
<xsd:complexType name="DuplicateNames">
  <xsd:sequence>
    <xsd:element name="foo" type="xsd:string" default="elem_value"/>
  </xsd:sequence>
  <xsd:attribute name="foo" type="xsd:string" default="attr_value"/>
</xsd:complexType>
```

En XPath, debe poder diferenciar elementos con nombres idénticos de atributos. Esto se consigue comenzando uno de los nombres con un signo (@). En el siguiente fragmento de código se muestra cómo acceder a un elemento y atributo con nombre idéntico:

```
1 DataObject duplicateNames = ...

2 // Muestra "elem_value"
3 System.out.println(duplicateNames.get("foo"));

4 // Muestra "attr_value"
5 System.out.println(duplicateNames.get("@foo"));
```

Utilice este esquema de denominación para todos los métodos que tienen un valor String que es un SDO XPath.

Soporte de grupo de modelos (todos, opción, secuencia y referencias de grupo):

La especificación de SDO requiere que los grupos de modelos (todos, opción, secuencia y referencias de grupo) se expandan en su lugar y que no describan tipos ni propiedades.

Básicamente, esto significa que ninguna de estas estructuras que están dentro de las mismas estructuras contenedoras se "presentan simultáneamente". Esta

"presentación simultánea" coloca todas las estructuras hijo en el mismo nivel, que puede producir problemas de nombres duplicados en un SDO cuya estructura se derive de los datos presentados simultáneamente. Cuando una XSD no presenta simultáneamente los grupos, sigue habiendo una separación de duplicados que están incluidos en distintos padres.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://MultipleGroup">
  <xsd:complexType name="MultipleGroup">
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="option1" type="xsd:string"/>
        <xsd:element name="option2" type="xsd:string"/>
      </xsd:choice>
      <xsd:element name="separator" type="xsd:string"/>
      <xsd:choice>
        <xsd:element name="option1" type="xsd:string"/>
        <xsd:element name="option2" type="xsd:string"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Dado que las múltiples apariciones de option1 y option2 están incluidas en distintos bloques de opciones e incluso tienen un elemento separador entre ellos, XSD y XML no tienen ningún problema para distinguirlos. Pero cuando SDO presente simultáneamente estos grupos, todas las propiedades de opciones están bajo el mismo contenedor de MultipleGroup.

Incluso sin nombres duplicados, también está en problema semántico que provoca la presentación simultánea de estos grupos. Por ejemplo, examine la siguiente XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://SimpleChoice">
  <xsd:complexType name="SimpleChoice">
    <xsd:sequence>
      <xsd:choice>
        <xsd:element name="option1" type="xsd:string"/>
        <xsd:element name="option2" type="xsd:string"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Solicitar al usuario que renombre nombres duplicados o añada anotaciones especiales a las XSD no es práctico porque en muchos casos, como en los estándares y esquemas del sector, el usuario no controla las XSD que utiliza.

Para crear coherencia para todas las propiedades, los objetos empresariales incluyen un método para acceder a cada aparición individual de las propiedades con nombre duplicadas a través de XPath. Siguiendo el convenio de denominación de la infraestructura de objeto de negocio, a todos los nombres de propiedad duplicados que se encuentre se les añadirá el siguiente dígito sin utilizar a su nombre, por ejemplo, la siguiente XSD:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://TieredGroup">
  <xsd:complexType name="TieredGroup">
    <xsd:sequence>
      <xsd:choice minOccurs="0">
        <xsd:sequence>
```

```
          <xsd:element name="low" minOccurs="1"
     maxOccurs="1" type="xsd:string"/>
          <xsd:choice minOccurs="0">
            <xsd:element name="width" minOccurs="0"
      maxOccurs="1" type="xsd:string"/>
            <xsd:element name="high" minOccurs="0"
      maxOccurs="1" type="xsd:string"/>
          </xsd:choice>
        </xsd:sequence>
        <xsd:element name="high" minOccurs="1"
    maxOccurs="1" type="xsd:string"/>
          <xsd:sequence>
            <xsd:element name="width" minOccurs="1"
     maxOccurs="1" type="xsd:string"/>
            <xsd:element name="high" minOccurs="0"
     maxOccurs="1" type="xsd:string"/>
          </xsd:sequence>
          <xsd:sequence>
            <xsd:element name="center" minOccurs="1"
     maxOccurs="1" type="xsd:string"/>
            <xsd:element name="width" minOccurs="0"
     maxOccurs="1" type="xsd:string"/>
          </xsd:sequence>
       </xsd:choice>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

La XSD anterior produce el siguiente modelo DataObject:

```
DataObject - TieredGroup
 Property[0] - low - string
 Property[1] - width - string
 Property[2] - high - string
 Property[3] - high1 - string
 Property[4] - width1 - string
 Property[5] - high2 - string
 Property[6] - center - string
 Property[7] - width2 - string
```

Donde **width**, **width1** y **width2** son los nombres de las propiedades de nombre width empezando por la primera en la XSD hacia abajo y lo mismo con **high**, **high1**, **high2**.

Estos nombres de propiedad nuevos son los nombres utilizados como referencia y XPath, y no afectan al contenido serializado. Los nombres "true" de cada una de estas propiedades que aparecen en el XML serializado son los valores dados en la XSD. Entonces, para la instancia XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:TieredGroup xsi:type="p:TieredGroup"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://TieredGroup">
  <width>foo</width>
  <high>bar</high>
</p:TieredGroup>
```

Para acceder a estas propiedades podría utilizar el siguiente código:

```
 DataObject tieredGroup = ...

 // Muestra "foo"
 System.out.println(tieredGroup.get("width1"));

 // Muestra "bar"
 System.out.println(tieredGroup.get("high2"));
```

Diferenciación de propiedades con nombre idéntico

Cuando varias XSD con el mismo espacio de nombres definen los mismos tipos con nombre, accidentalmente se puede hacer referencia a un tipo incorrecto.

Address1.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Address">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="city" type="xsd:string"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

Address2.xsd:
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Address">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="state" type="xsd:string"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Los objetos empresariales no dan soporte a nombres duplicados para ninguna estructura de XSD global (como complexType, simpleType, element, attribute, etc.) a través de las API BOFactory.create(). Estas estructuras globales duplicadas se pueden seguir creando como hijos de otras estructuras si se utilizan las API adecuadas, tal como se muestra en los siguientes ejemplos.

Customer1.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://Customer1"
 targetNamespace="http://Customer1">
  <xsd:import schemaLocation="./Address1.xsd"/>
  <xsd:complexType name="Customer">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="address" type="Address"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Customer2.xsd:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://Customer2"
 targetNamespace="http://Customer2">
  <xsd:import schemaLocation="./Address2.xsd"/>
  <xsd:complexType name="Customer">
    <xsd:sequence>
      <xsd:element minOccurs="0" name="address" type="Address"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Al llenar los dos campos de dirección de cliente y después llamar a BOFactory.create() para realizar la dirección, los tipos de objetos empresariales resultantes pueden establecerse incorrectamente. Puede evitarlo llamando a la API

createDataObject("address") en el DataObject Customer. Esto garantizará que se genere un hijo del tipo correcto porque los objetos empresariales seguirán la schemaLocation de la importación.

```
DataObject customer1 = ...

// Forma incorrecta de crear una dirección hijo
// Esto podría crear un tipo de dirección Address1.xsd Address o maybe Address2.xsd
DataObject incorrect = boFactory.create("", "Address");
customer1.set("address", incorrect);

// Corrija la forma de crear la dirección hijo
// Esto garantizará que se cree un tipo de dirección Address1.xsd
customer1.createDataObject("address");
```

Cómo resolver los nombres de propiedad que contienen puntos

Los nombres de propiedad en una XSD pueden incluir un punto (".") como uno de los caracteres válidos, mientras que en un SDO también se utilizan para mostrar la indexación en un propiedad de múltiple cardinalidad. Esto puede causar problemas de resolución en determinadas situaciones.

Los nombres de propiedad en los SDO (Service Data Objects) se basan en los nombres de los elementos y atributo que se generan a partir de la XSD. Los objetos empresariales manejarán el carácter "." adecuadamente, con una excepción: si una XSD tiene una propiedad de única cardinalidad denominada "<name>.<#>" y una propiedad de múltiple cardinalidad llamada "<name>".

Un XPath como "foo.0" no se resolvería adecuadamente si hubiera una propiedad de única cardinalidad llamada "foo.0" y una propiedad de múltiple cardinalidad llamada "foo". En este caso, la propiedad de única cardinalidad llamada "foo.0" sería la que se resolviera. Aunque esto debería ocurrir raramente, puede evitarlo completamente si utiliza la sintaxis "foo[1]" para acceder a su propiedad de múltiple cardinalidad. Los SDO no darán soporte a la sintaxis "." para la indexación, por lo que debería utilizar "[]" para la indexación.

Serialización y deserialización de uniones con xsi:type:

En XSD, una unión es una forma de fusionar espacios léxicos de varios tipos de datos simples conocidos como miembros.

En el siguiente ejemplo XSD muestra una unión que tiene los miembros de un entero y una fecha.

```
<xsd:simpleType name="integerOrDate">
 <xsd:union memberTypes="xsd:integer xsd:date"/>
</xsd:simpleType>
```

Estos múltiples tipos pueden causar confusión durante la deserialización y al manipular los datos.

Los objetos empresariales dan soporte a SDO utilizando xsi:type para la serialización y seguirán el mismo algoritmo para determinar el tipo en la deserialización si xsi:type no existe en los datos XML.

Por lo tanto, para garantizar que los datos (el número "42" en este ejemplo) se deserializarán como entero, puede utilizar el xsi:type especificado en el XML de entrada. También puede ordenar la lista de miembros de la unión de la XSD para que el entero sea anterior a la serie. En el siguiente ejemplo se muestra cómo se implementan ambos métodos:

```
<integerOrString xsi:type="xsd:integer">42</integerOrString>

<xsd:simpleType name="integerOrString">
 <xsd:union memberTypes="xsd:integer xsd:string"/>
</xsd:simpleType>
```

De igual modo, si el usuario deseaba que los datos se deserializarán como una serie, cualquiera de los siguientes cambios podría causar ese comportamiento:

```
<integerOrString xsi:type="xsd:string">42</integerOrString>

<xsd:simpleType name="integerOrString">
 <xsd:union memberTypes="xsd:string xsd:integer"/>
</xsd:simpleType>
```

Tenga en cuenta que si un tipo serie es el primer miembro de la unión, nunca tiene pérdida de información. También puede mantener datos que siempre se elegirán mediante el algoritmo ningún xsi:type. Si desea utilizar un tipo distinto que no sea la serie, debe utilizar xsi:type en el XML o volver a ordenar los tipos de miembro de la XSD para que los otros miembros tengan la oportunidad de aceptar datos.

Soporte para objetos de negocio nulos

Este escenario implica un sistema externo que se comunica con WebSphere Process Server a través de un XML con envoltorio dentro de un mensaje SOAP. Cuando el elemento incluido puede adoptar un valor cero `xsi:nil="true"`, el DataObject resultante que se crea en WebSphere Process Server es nulo.

Aquí aparece un ejemplo que ilustra un mensaje XML con un elemento que puede adoptar el valor cero.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope">
 <soap:Body>
  <p:Employee xmlns:p="http://www.mycompany.com" xmlns:xsi="http://www.w3.org"
   xsi:nil="true"/>
 </soap:Body>
</soap:Envelope>
```

Donde `Employee` se define como:

```
<element name="Employee" nillable="true">
...
</element>
```

En este caso, el objeto de negocio que se genera y se envía desde la exportación es nulo. Por ejemplo, si cualquier componente en sentido descendente tiene operaciones invocadas, la salida de dicha operación es nula.

Nota: Estos objetos no se pueden pasar en las correlaciones de objetos empresariales porque las correlaciones de objetos empresariales no pueden correlacionar los campos desde un objeto nulo.

Utilización del objeto Sequence para establecer el orden de datos

Algunas XSD se definen de una forma que hace que el orden en el que aparecen los datos en el XML tengan una importancia especial.

Un ejemplo de la importancia del orden en las XSD es el contenido mixto. Si los datos de texto aparecen antes o después de un elemento, puede tener un significado distinto a si se produjera en una ubicación distinta. Para estas

situaciones, SDO genera un objeto conocido como Secuencia, que se utiliza para establecer los datos de una forma ordenada.

Las secuencias de SDO no deben confundirse con secuencias de XSD. Las secuencias de XSD sólo son grupos de modelos que se presentan simultáneamente antes de la generación de modelos de SDO. La presencia de una secuencia de XSD no está relacionada con la presencia de una secuencia de SDO.

Las siguientes condiciones de una XSD harán que se genere una secuencia de SDO:

Un complexType con contenido mixto:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://MixedContent"
 targetNamespace="http://MixedContent">
  <xsd:complexType name="MixedContent" mixed="true">
    <xsd:sequence>
      <xsd:element name="element1" type="xsd:string" minOccurs="0"/>
      <xsd:element name="element2" type="xsd:string" minOccurs="0"/>
      <xsd:element name="element3" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="MixedContent" type="tns:MixedContent"/>
</xsd:schema>
```

Un esquema que tiene 1 o más códigos <any/>:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://AnyElemAny"
 targetNamespace="http://AnyElemAny">
  <xsd:complexType name="AnyElemAny">
    <xsd:sequence>
      <xsd:any/>
      <xsd:element name="marker1" type="xsd:string"/>
      <xsd:any/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Una matriz de grupos de modelos (una referencia a todos, opción, secuencia o grupo con maxOccurs > 1):

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://ModelGroupArray">
  <xsd:complexType name="ModelGroupArray">
    <xsd:sequence maxOccurs="3">
      <xsd:element name="element1" type="xsd:string"/>
      <xsd:element name="element2" type="xsd:string"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

En el código <all/> de maxOccurs <= 1 que contiene más de un elemento:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://All">
  <xsd:complexType name="All">
    <xsd:all>
      <xsd:element name="element1" type="xsd:string"/>
```

```
      <xsd:element name="element2" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
</xsd:schema>
```

Información específica sobre la utilización conjunta de <any/> y la secuencia se tratará en el tema listado al final de esta página. La información general que aparece en el resto de esta sección describirá cómo trabajar con otras condiciones de secuencia, aunque también se aplicará a <any/>.

¿Cómo puedo saber si mi DataObject tiene una secuencia?:

Existen dos API simples que se pueden seleccionar para determinar si un DataObject está secuenciado: DataObject noSequence y DataObject withSequence.

Utilice DataObject noSequence y DataObject withSequence tal como se muestra en el siguiente ejemplo:

```
DataObject noSequence = ...
DataObject withSequence = ...

// Muestra false
System.out.println(noSequence.getType().isSequenced());

// Muestra true
System.out.println(withSequence.getType().isSequenced());

// Muestra true
System.out.println(noSequence.getSequence() == null);

// Muestra false
System.out.println(withSequence.getSequence() == null);
```

¿Por qué necesito saber que un DataObject tiene una secuencia?:

Si trabaja en un DataObject que tiene una secuencia, es importante saber el orden en el que se establecen los datos. Debido a esto, es necesario ir con cuidado cuando se establece el orden de los valores.

Un DataObject que no está secuenciado permite que se establezca un orden de acceso aleatorio. Esto funciona como una correlación donde todas las claves ese establecen en los mismos valores. No importa en qué orden se establezcan las claves, los datos de la correlación son los mismos y se serializarán con XML de forma idéntica.

Cuando un DataObject está secuenciado, el orden en el que se establecieron los datos está registrado en la secuencia, de forma parecida a cómo se añaden los datos a la lista. Esto proporciona dos formas de acceder a los datos, por pares de nombre/valor (las API de DataObject) y por el orden en el que se ha establecido (las API de secuencia). Puede utilizar las API set(...) o Sequence add(...) de DataObject para conservar la estructura. Este orden afecta a la forma en que el XML se serializa.

Por ejemplo, fíjese en la XSD del código <all/> siguiente. Cuando los métodos set se invocan en el siguiente orden, se produce el siguiente XML cuando se serializa:

```
DataObject all = ...
all.set("element1", "foo");
all.set("element2", "bar");

<?xml version="1.0" encoding="UTF-8"?>
```

```
<p:All xsi:type="p:All"
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xmlns:p="http://All">
  <element1>foo</element1>
  <element2>bar</element2>
</p:All>
```

En cambio, si los métodos set se invocan en el orden opuesto, se creará el siguiente XML cuando se serialice el objeto empresarial:

```
 DataObject all = ...
 all.set("element2", "bar");
 all.set("element1", "foo");

<?xml version="1.0" encoding="UTF-8"?>
<p:All xsi:type="p:All"
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xmlns:p="http://All">
  <element2>bar</element2>
  <element1>foo</element1>
</p:All>
```

Si alguna vez es necesario cambiar el orden de la secuencia , la clase Sequence tiene métodos básicos para añadir, eliminar y mover que permiten al usuario alterar el orden de la secuencia.

¿Cómo puedo trabajar con contenido mixto?:

Para un contenido mixto, la secuencia tiene una API específica para añadir texto:addText(...).

Las demás API funcionan de la misma forma con texto que con propiedades. La API getProperty(int) devolverá un valor nulo para datos de texto de contenido mixto. El siguiente ejemplo de código de contenido mixto se puede utilizar para imprimir todo el texto de contenido mixto de un DataObject:

```
 DataObject mixedContent = ...
 Sequence seq = mixedContent.getSequence();

 for (int i=0; i < seq.size(); i++)
 {
     Property prop = seq.getProperty(i);
     Object value = seq.getValue(i);

     if (prop == null)
     {
         System.out.println("Found mixed content text: "+value);
     }
     else
     {
         System.out.println("Found Property "+prop.getName()+": "+value);
     }
 }
```

¿Cómo puedo trabajar con una matriz de grupo de modelos?:

Una matriz de grupo de modelos se crea cuando un grupo de modelos tiene un valor para maxOccurs > 1.

Ya que los grupos de modelos se presentan simultáneamente y no se expresan en un DataObject, las propiedades dentro del grupo de modelos pasan a ser varias propiedades de cardinalidad de modo que sus métodos isMany() devuelven true si todavía no son true. Sus facetas minOccurs y maxOccurs pasan a multiplicarse por

el valor del grupo de modelos que lo contiene. Choice multiplicará la faceta maxOccurs de la misma forma que los demás grupos de modelos, pero siempre utilizará 0 como valor de multiplicación para minOccurs porque es posible que no se haya seleccionado ningún dato de una opción.

Por ejemplo, la siguiente XSD tiene una matriz de grupos de modelos:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://ModelGroupArray">
  <xsd:complexType name="ModelGroupArray">
    <xsd:sequence minOccurs="2" maxOccurs="5">
      <xsd:element name="element1" type="xsd:string"/>
      <xsd:element name="element2" type="xsd:string"
   minOccurs="0" maxOccurs="3"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Tal como se indica, **element1** y **element2** ahora tendrán múltiple cardinalidad para que un objeto usuario get(...) devuelva una lista. **Element1** tiene un minOccurs por omisión de 1 y maxOccurs de 1. **Element2** tiene un minOccurs de 0 y un maxOccurs de 3. En el siguiente ejemplo, los nuevos minOccurs y maxOccurs serán los siguientes:

```
DataObject - ModelGroupArray
 Property[0] - element1 - minOccurs=(2*1)=2 - maxOccurs=(5*1)=5
 Property[1] - element2 - minOccurs=(2*0)=0 - maxOccurs=(5*3)=15
```

Si el tipo fuera Choice:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  targetNamespace="http://ModelGroupArray">
  <xsd:complexType name="ModelGroupArray">
    <xsd:choice minOccurs="2" maxOccurs="5">
      <xsd:element name="element1" type="xsd:string"/>
      <xsd:element name="element2" type="xsd:string"
   minOccurs="0" maxOccurs="3"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Generaría los siguientes minOccurs debido a la exclusión de la opción de que sólo se puede seleccionar **element1** cada vez o sólo se puede seleccionar **element2** cada vez, por lo que para pasar la validación los dos necesitan poder tener 0 apariciones:

```
DataObject - ModelGroupArray
 Property[0] - element1 - minOccurs=(0*1)=0 - maxOccurs=(5*1)=5
 Property[1] - element2 - minOccurs=(0*0)=0 - maxOccurs=(5*3)=15
```

Utilización de cualquier tipo de datos

Esta sección proporciona técnicas de programación para utilizar cualquier tipo de datos.

Utilización de AnySimpleType para tipos simples:

Las API de SDO no gestionan AnySimpleType de forma distinta a cualquier otro tipo simple (string, int, boolean, etc.).

Las únicas diferencias entre anySimpleType y los otros tipos simples son sus datos de instancias y la serialización/deserialización. Estos pueden ser conceptos

internos sólo para el objeto empresarial y se utilizan para determinar si los datos correlacionados a o desde el campo son válidos. Si en un tipo string se invocara un método set(...), los datos primero se convertirían en una serie y el tipo de datos original se perdería:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://StringType">
  <xsd:complexType name="StringType">
    <xsd:sequence>
      <xsd:element name="foo" type="xsd:string"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

 DataObject stringType = ...

 // Establecer los datos en String
 stringType.set("foo", "bar");

 // Los datos de instancia siempre serán de tipo String, independientemente del
 // conjunto de datos
 // Muestra "java.lang.String"
 System.out.println(stringType.get("foo").getClass().getName());

 // Establecer los datos en Integer
 stringType.set("foo", new Integer(42));

 // Los datos de instancia siempre serán de tipo String, independientemente del
 // conjunto de datos
 // Muestra "java.lang.String"
 System.out.println(stringType.get("foo").getClass().getName());
```

En cambio, anySimpleType no pierde el tipo de datos original de lo que se establece:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://AnySimpleType">
  <xsd:complexType name="AnySimpleType">
    <xsd:sequence>
      <xsd:element name="foo" type="xsd:anySimpleType"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

 DataObject anySimpleType = ...

 // Establecer los datos en String
 stringType.set("foo", "bar");

 // Los datos de instancia siempre serán del tipo de datos utilizado en el conjunto
 // Muestra "java.lang.String"
 System.out.println(stringType.get("foo").getClass().getName());

 // Establecer los datos en Integer
 stringType.set("foo", new Integer(42));

 // Los datos de instancia siempre serán del tipo de datos utilizado en el conjunto
 // Muestra "java.lang.Integer"
 System.out.println(stringType.get("foo").getClass().getName());
```

Este tipo de datos también lo mantiene xsi:type a lo largo de la serialización y deserialización. Por consiguiente, cada vez que serialice un elemento anySimpleType, tendrá un xsi:type que coincida con lo definido en la especificación de SDO en función de su tipo Java:

En el siguiente ejemplo se serializa el objeto empresarial anterior, por lo que los datos se parecerían a lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:StringType xsi:type="p:StringType"
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xmlns:xsd=http://www.w3.org/2001/XMLSchema
 xmlns:p="http://StringType">
  <foo xsi:type="xsd:int">42</foo>
<p:StringType></p:StringType>
```

xsi:type se utilizará durante la deserialización para cargar datos como la clase de instancia Java adecuada. Si no se especifica ningún xsi:type, el tipo de deserialización por omisión será una serie.

Para los otros tipos simples, la determinación de la posibilidad de correlación es una constante. Por ejemplo, un booleano siempre se puede correlacionar con una serie. Sin embargo, AnySimpleType puede contener cualquiera de los tipos simples, por lo que una correlación puede o no ser posible en función de los datos de instancia en el campo.

Utilice el URI y el nombre del tipo de propiedad para determinar si una propiedad es de tipo anySimpleType. Serán "commonj.sdo" y "Object". Para determinar si es válido insertar los datos en anySimpleType, copie para ver si no es una instancia de un DataObject. Todos los datos que se pueden representar como una String y no es un DataObject pueden establecerse en un campo anySimpleType.

Esto conduce a las siguientes reglas de correlación:

- anySimpleType se puede relacionar con anySimpleType.
- cualquier otro tipo simple siempre se puede correlacionar con anySimpleType.
- anySimpleType siempre se puede correlacionar con una serie porque todos los tipos simples deben poder convertirse en una serie.
- anySimpleType puede o no puede correlacionarse con ninguno de los otros tipos simples, en función de su valor en el objeto empresarial. Esto significa que esta correlación no se puede determinar en el momento del diseño, sólo durante la ejecución.

Información relacionada

Asignación de y a xs:any

Utilización de AnyType para tipos complejos:

Las API SDO no manejan el código anyType de forma distinta a cómo manejan cualquier otro tipo complejo.

Las únicas diferencias entre anyType y cualquier otro tipo complejo están en los datos de la instancia y la serialización/deserialización, que deben ser conceptos internos sólo para el objeto empresaria, y al determinar si los datos que se correlacionan a o desde el campo son válidos. Los tipos complejos se limitan a un único tipo: Customer, Address, etc. Sin embargo, anyType admite todos los DataObject independientemente del tipo. Si maxOccurs > 1, cada DataObject de la lista puede ser de un tipo distinto.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://AnyType">
  <xsd:complexType name="AnyType">
    <xsd:sequence>
```

```
      <xsd:element name="person" type="xsd:anyType"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://Customer">
  <xsd:complexType name="Customer">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://Employee" targetNamespace="http://Employee">
  <xsd:complexType name="Employee">
    <xsd:sequence>
      <xsd:element name="id" type="xsd:string"/>
 </xsd:sequence>
  </xsd:complexType>
</xsd:schema>

 DataObject anyType = ...
 DataObject customer = ...
 DataObject employee = ...

 // Establecer la persona en un cliente
 anyType.set("person", customer);

 // Los datos de instancia serán un cliente
 // Muestra "Customer"
 System.out.println(anyType.getDataObject("person").getName());

 // Establecer la persona en un empleado
 anyType.set("person", employee);

 // Los datos de instancia serán un empleado
 // Muestra "Employee"
 System.out.println(anyType.getDataObject("person").getName());
```

Al igual que anySimpleType, anyType utiliza xsi:type durante la serialización para asegurarse de que se mantiene el tipo previsto de DataObject cuando se deserializa. Cuando establece en "Customer," el XML se parecería a:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:AnyType xsi:type="p:AnyType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:customer="http://Customer"
    xmlns:p="http://AnyType">
  <person xsi:type="customer:Customer">
    <name>foo</name>
  </person>
</p:AnyType>
```

Y cuando se establece en "Employee":

```
<?xml version="1.0" encoding="UTF-8"?>
<p:AnyType xsi:type="p:AnyType"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:employee="http://Employee"
    xmlns:p="http://AnyType">
```

```
  <person xsi:type="employee:Employee">
    <id>foo</id>
  </person>
</p:AnyType>
```

AnyType también permite establecer valores de tipo simple a través de DataObjects del envoltorio. Estos DataObjects de envoltorio tienen una sola propiedad llamada "value" (element) que mantiene el valor de tipo simple. Las API de SDO se han alterado temporalmente para empaquetar y desempaquetar automáticamente estos tipos simples y DataObjects de envoltorio cuando utilizan las API get<Type>/set<Type>. Las API get/set de conversión de no tipo no realizarán este empaquetamiento.

```
DataObject anyType = ...

// Al llamar a una API set<Type> en una propiedad anyType hace que
// se cree automáticamente un envoltorio DataObject
anyType.setString("person", "foo");

// Las API get/set normales no se alteran temporalmente, por lo que devolverán
// el envoltorio DataObject
DataObject wrapped = anyType.get("person");

// El envoltorio DataObject tendrá la propiedad "value"
// Muestra "foo"
System.out.println(wrapped.getString("value"));

// La API get<Type> desempaquetará automáticamente el DataObject
// Muestra "foo"
System.out.println(anyType.getString("person"));
```

Cuando el envoltorio DataObject se serializa, éste se serializará como la correlación anySimpleType de clases de instancia Java con tipos XSD en el campo xsi:type. Por lo que este valor se serializaría como:

```
<?xml version="1.0" encoding="UTF-8"?>
<p:AnyType xsi:type="p:AnyType"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:p="http://AnyType">
  <person xsi:type="xsd:string">foo</person>
</p:AnyType>
```

Si no se indica ningún xsi:type o se indica un xsi:type incorrecto, se generará una excepción. Además del empaquetamiento automático, el envoltorio se puede crear manualmente para utilizarlo con la API de set() a través de BOFactory createDataTypeWrapper(Type, Object) donde Type es el tipo simple de SDO de los datos que se van a empaquetar y Object son los datos a empaquetar.

```
Type stringType = boType.getType("http://www.w3.org/2001/XMLSchema", "string");
DataObject stringType = boFactory.createByMessage(stringType, "foo");
```

Para determinar si DataObject es un tipo de envoltorio, se puede llamar a BOType isDataTypeWrapper(Type).

```
DataObject stringType = ...
boolean isWrapper = boType.isDataTypeWrapper(stringType.getType());
```

Para los demás tipos complejos, para mover datos de un campo a otro, los datos deben ser del mismo tipo. Sin embargo, AnyType puede contener cualquiera de los tipos simples, por lo que un movimiento directo sin una correlación puede o no ser posible en función de los datos de instancia en el campo.

Puede utilizar el URI y el nombre del tipo de propiedad para determinar si una propiedad es de tipo anyType. Serán "commonj.sdo" y "DataObject". Todos los datos son válidos para insertar en un anyType. Esto conduce a las siguientes reglas de correlación:

- anyType siempre se puede correlacionar con anyType.
- cualquier tipo complejo se puede correlacionar con anyType.
- cualquier tipo simple se puede correlacionar con anyType.
- anyType puede o no puede correlacionarse con ninguno de los otros tipos simples o complejos, en función de su valor en la instancia de BO. Esto significa que esta correlación no se puede determinar en el momento del diseño, sólo durante la ejecución.

Utilización de Any para establecer elementos globales para tipos complejos:

Puede utilizar el código <any/> para establecer elementos globales en un tipo complejo.

Una aparición del código any hace que el método DataObject Type isOpen() y el método isSequenced() devuelva true. Si el valor de maxOccurs es > 1 en un código any, no tiene ningún efecto en la estructura de DataObject; sólo se utiliza como información durante la validación. De forma parecida, la aparición de varios códigos any en un tipo no cambia la estructura del DataObject; sólo se utilizan para validar la ubicación de datos abiertos establecidos.

¿Cómo puedo saber si mi DataObject tiene alguna etiqueta?:

Puede determinar fácilmente si las instancias de un DataObject tienen valores any establecidos dentro comprobando las propiedades de la instancia para ver si alguna de las propiedades open son atributos.

DataObject no proporciona un mecanismo para determinar si un tipo DataObject tiene un código any. DataObjects sólo tienen el concepto de "open" que se aplican tanto a any como a anyAttribute y se permite añadir libremente propiedades any. La presencia de un código any causa que un DataObject tengan isOpen() = true y isSequenced() = true, podría tener sólo un código anyAttribute y una de las razones para secuenciarse que se tratan en los temas de secuencias. El siguiente ejemplo demuestra estos conceptos:

```
DataObject dobj = ...

// Comprobar si el tipo es open, si no lo es no puede tener
// establecidos valores any en el mismo.
boolean isOpen = dobj.getType().isOpen();

if (!isOpen) return false;  // No hay establecidos valores any

// Propiedades Open se añaden a la lista de propiedades de instancia, pero no
// la lista de propiedades, por lo que si se comparan sus tamaños se podrá
// determinar fácilmente si se ha establecido algún dato open
int instancePropertyCount = dobj.getInstanceProperties().size();
int definedPropertyCount = dobj.getType().getProperties().size();

// Si es igual, no hay establecido ningún contenido abierto
if (instancePropertyCount == definedPropertyCount) return false;

// Comprobar las propiedades de contenido abierto para determinar si any
// son elementos
for (int i=definedPropertyCount; i < instancePropertyCount; i++)
{
```

```
    Property prop = (Property)dobj.getInstanceProperties().get(i);
    if (boXsdHelper.isElement(prop))
    {
        return true;  // Se ha encontrado un valor any
    }
}

return false;  // No tiene establecido ningún valor any
```

¿Cómo puedo obtener/establecer valores?:

Efectuar una llamada get sobre datos que se establecieron en un campo any se puede llevar a cabo de la misma forma que con cualquier otro valor de elemento si se conoce el nombre.

Puede llevar a cabo una llamada get con el XPath "<name>" y se resolverá. Si no se conoce el nombre, el valor se puede encontrar comprobando las propiedades de la instancia tal como se ha efectuado anteriormente. Si hay varios códigos any, o un código any con maxOccurs > 1, en su lugar deberá utilizarse la secuencia de DataObject si es importante determinar en qué código any se originan los datos.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://AnyElemAny"
 targetNamespace="http://AnyElemAny">
  <xsd:complexType name="AnyElemAny">
    <xsd:sequence>
      <!-- Manejar todos los códigos any de una forma -->
      <xsd:any maxOccurs="3"/>
      <xsd:element name="marker1" type="xsd:string"/>
      <!-- Manejar este código any de otra forma -->
      <xsd:any/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Dado que el código <any/> hace que el DataObject esté secuenciado, determinar qué valor any se ha establecido se puede llevar a cabo comprobando la secuencia para la posición de las propiedades any.

Puede determinar a qué código any pertenecen los datos de la instancia para la siguiente XSD utilizando el siguiente código:

```
DataObject anyElemAny = ...
Seqeuence seq = anyElemAny.getSequence();

// Hasta que encontramos el elemento marker1, todos los datos open
// encontrados pertenecen al primer código any
boolean foundMarker1 = false;

for (int i=0; i<seq.size(); i++)
{
    Property prop = seq.getProperty(i);

    // Comprobar si la propiedad es una propiedad open
    if (prop.isOpenContent())
    {
        if (!foundMarker1)
        {
            // Debe ser el primer any por que aparece
            // antes del elemento marker1
            System.out.println("Found first any data: "+seq.getValue(i));
        }
        else
        {
```

```
            // Debe ser el segundo any por que aparece
            // después del elemento marker1
            System.out.println("Found second any data: "+seq.getValue(i));
         }
    }
    else
    {
        // Debe ser el elemento marker1
        System.out.println("Found marker1 data: "+seq.getValue(i));
        foundMarker1 = true;
    }
}
```

Establecer un valor <any/> se lleva a cabo creando una propiedad de elemento global y añadiendo ese valor a la secuencia.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://GlobalElems"
 targetNamespace="http://GlobalElems">
  <xsd:element name="globalElement1" type="xsd:string"/>
  <xsd:element name="globalElement2" type="xsd:string"/>
</xsd:schema>

 DataObject anyElemAny = ...
 Seqeuence seq = anyElemAny.getSequence();

 // Obtener la propiedad de elemento global para globalElement1
 Property globalProp1 = boXsdHelper.getGlobalProperty(http://GlobalElems,
 "globalElement1", true);

 // Obtener la propiedad de elemento global para globalElement2
 Property globalProp2 = boXsdHelper.getGlobalProperty(http://GlobalElems,
 "globalElement2", true);

 // Añadir los datos a la secuencia para el primer any
 seq.add(globalProp1, "foo");
 seq.add(globalProp1, "bar");

 // Añadir los datos para marker1
 seq.add("marker1", "separator");  // or anyElemAny.set("marker1", "separator")

 // Añadir los datos a la secuencia para el segundo any
 seq.add(globalProp2, "baz");

 // Ahora se podrá acceder a los datos mediante una llamada get
 System.out.println(dobj.get("globalElement1"); // Muestra "[foo, bar]"
 System.out.println(dobj.get("marker1");        // Muestra "separator"
 System.out.println(dobj.get("globalElement2"); // Muestra "baz"
```

¿Cuáles son correlaciones válidas para datos en un any?:

Un código <any/> es un conjunto de pares nombre/valor. La única correlación válida que se puede determinar en el momento del diseño para <any/> es otro <any/> o anyType que tenga el mismo valor maxOccurs.

De forma individual, los valores incluidos en una instancia de DataObject para any son tipos complejos básicos que siguen todas las reglas de correlación de tipos complejos. Algunos de estos tipos complejos pueden ser tipos simples empaquetados, de forma que seguirán las reglas de la correlación de tipos simples.

Utilización de AnyAttribute para establecer atributos globales para tipos complejos:

El código <anyAttribute/> permite establecer cualquier número de atributos globales para un tipo complejo.

Parecido al código <any/>, la aparición del código <anyAttribute/> hace que el método DataObject Type isOpen() devuelva true. Sin embargo, a diferencia del código <any/>, el código <anyAttribute/> no causa que DataObject se secuencie porque los atributos en XSD no son construcciones ordenadas.

¿Cómo puedo saber si miDataObject tiene una etiqueta anyAttribute?:

Puede determinar fácilmente si las instancias de un DataObject tienen valores anyAttribute establecidos dentro comprobando las propiedades de la instancia para ver si alguna de las propiedades open son atributos.

DataObject no proporciona un mecanismo para determinar si un tipo DataObject tiene un código anyAttribute. Los DataObject sólo tienen el concepto de "open" que se aplica tanto a any como a <anyAttribute/> y se permite añadir libremente propiedades any. Si un DataObject tiene isOpen() = true y isSequenced() = false, debe tener un código anyAttribute, si isOpen() = true y isSequenced() = true, el tipo DataObject Type puede o no tener un código anyAttribute.

DataObject proporciona métodos de consulta de metadatos para responder mediante programación a ésta y a otras preguntas sobre la estructura XSD que se utilizó para generar el DataObject. Se pueden realizar consultas del modelo InfoSet si es necesario saber si hay algún código anyAttribute. Puesto que anyAttribute es singular y es o no es true, los objetos empresariales también proporcionarán un método BOXSDHelper hasAnyAttribute(Type) para permitir la determinación en lo referente a si se establece un atributo open en este DataObject producirá un resultado válido. En el siguiente ejemplo se muestran estos conceptos:

```
DataObject dobj = ...

// Comprobar si el tipo es open, si no lo es no puede tener
// establecidos valores anyAttribute en el mismo.
boolean isOpen = dobj.getType().isOpen();

if (!isOpen) return false;  // No hay establecidos valores anyAttribute

// Propiedades Open se añaden a la lista de propiedades de instancia, pero no
// la lista de propiedades, por lo que si se comparan sus tamaños se podrá
// determinar fácilmente si se ha establecido algún dato open
int instancePropertyCount = dobj.getInstanceProperties().size();
int definedPropertyCount = dobj.getType().getProperties().size();

// Si es igual, no hay establecido ningún contenido abierto
if (instancePropertyCount == definedPropertyCount) return false;

// Comprobar las propiedades de contenido abierto para determinar si any
// son atributos
for (int i=definedPropertyCount; i<instancePropertyCount; i++)
{
    Property prop = (Property)dobj.getInstanceProperties().get(i);
    if (boXsdHelper.isAttribute(prop))
    {
        return true;  // Se ha encontrado un valor anyAttribute
    }
}

return false;  // No tiene establecido ningún valor anyAttribute
```

¿Cómo puedo obtener/establecer valores AnyAttribute?:

Establecer un valor <anyAttribute/> se hace de la misma forma que establecer un <any/>, pero en lugar de un elemento global se utiliza un atributo global.

Efectuar una llamada get sobre datos que se establecieron en un campo anyAttribute se puede llevar a cabo de la misma forma que con cualquier otro valor de atributo si se conoce el nombre. Puede llevar a cabo una llamada get con el XPath "@<name>" y se resolverá. Si no se conoce el nombre, si utiliza el código anterior los valores pueden repetirse y se puede acceder a los mismos de uno en uno. El código de ejemplo siguiente lo muestra:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:tns="http://AnyAttrOnlyMixed"
 targetNamespace="http://AnyAttrOnly">
  <xsd:complexType name="AnyAttrOnly">
    <xsd:sequence>
      <xsd:element name="element" type="xsd:string"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:schema>

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://GlobalAttrs">
  <xsd:attribute name="globalAttribute" type="xsd:string"/>
</xsd:schema>

 DataObject dobj = ...

 // Obtener la propiedad de atributo global que se va a establecer
 Property globalProp = boXsdHelper.getGlobalProperty(http://GlobalAttrs,
 "globalAttribute", false);

 // Establecer el valor de dobj, como cualquier otro dato
 dobj.set(globalProp, "foo");

 // Ahora se podrá acceder a los datos mediante una llamada get
 System.out.println(dobj.get("@globalAttribute")); // Muestra "foo"
```

¿Cuáles son correlaciones válidas para datos en un anyAttribute?:

El código AnyAttribute es similar al código any y es un conjunto de pares nombre/valor. Por lo tanto, la única correlación válida para anyAttribute es otro anyAttribute.

De forma individual, los valores contenidos en los datos anyAttribute son tipos simples básicos que siguen todas las reglas de la correlación de tipos simples

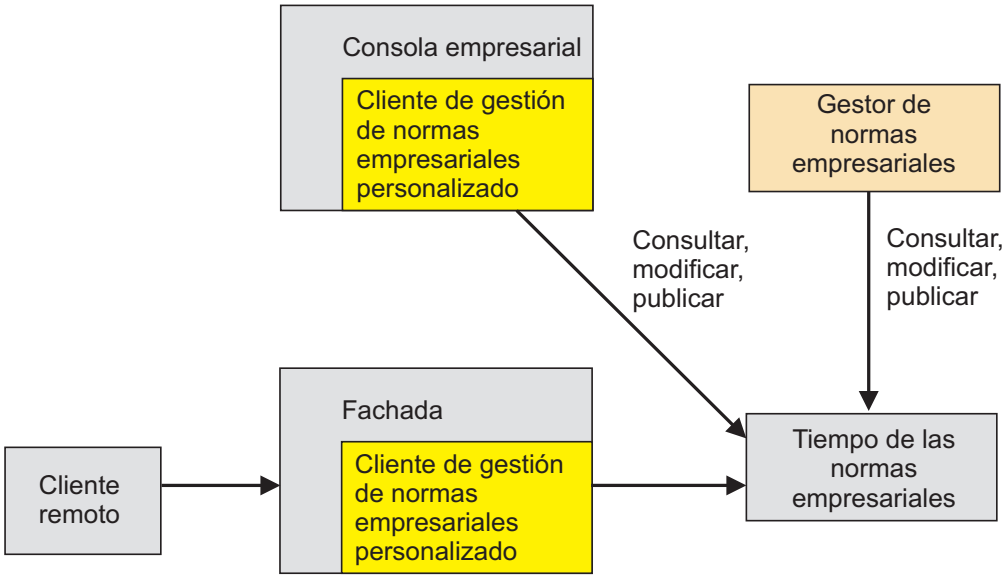
Programación de la gestión de normas empresariales

Se proporcionan clases públicas de gestión de normas empresariales para crear clientes de gestión personalizados o para automatizar cambios en normas empresariales.

Las clases de gestión de normas empresariales podrían usarse en una aplicación web combinadas con otras funciones de gestión para cosas como procesos de negocio o tareas de usuario y así gestionar todos los componentes desde un mismo cliente. Se puede usar cualquier cliente de gestión personalizado junto con la aplicación web del gestor de normas empresariales que incluye WebSphere Process Server. Las clases también podrían usarse para automatizar cambios a normas

empresariales desde dentro de una aplicación. Por ejemplo, se podrían cambiar normas empresariales como resultado de que un proceso de negocio que esté usando las normas empresariales supere algún umbral o límite.

Las clases de gestión de normas empresariales deben utilizarse en una aplicación instalada en WebSphere Process Server. Las clases no ofrecen una interfaz remota, aunque pueden envolverse en una fachada que posteriormente se expone sobre un protocolo específico para la ejecución remota.



Esta guía de programación está compuesta de dos secciones principales y un apéndice. La primera sección explica el modelo de programación y cómo usar las distintas clases. Se incluyen diagramas de clase que muestran la relación entre clases. La segunda sección incluye ejemplos sobre cómo usar las clases para realizar acciones como buscar grupos de normas empresariales, planificar un destino de norma nuevo y modificar un conjunto de normas o una tabla de decisiones. El apéndice contiene clases adicionales que se utilizaron en los ejemplos para simplificar operaciones comunes y ejemplos adicionales de creación de consultas complejas para buscar grupos de normas empresariales empleando comodines.

Esta información de guía de programación sobre las clases, también está disponible en formato HTML Javadoc en WebSphere Process Server v6.1 y en el entorno de pruebas incluido con WebSphere Integration Developer v6.1. Esta documentación en Javadoc está disponible en ${Directorio de instalación de WebSphere Process Server}\web\apidocs o en ${Directorio de instalación de WebSphere Integration Developer}\runtimes\bi_v61\web\apidocs. Los paquetes com.ibm.wbiserver.brules.mgmt.* contienen toda la información.

Modelo de programación

Las normas empresariales de WebSphere Business Integration se crean con dos herramientas de creación y lo emiten el tiempo de ejecución de la norma. Los tres comparten el mismo modelo para los artefactos de normas empresariales.

La compartición del modelo se consideró crítica no sólo para facilitar el mantenimiento futuro, sino para conseguir un modelo de programación coherente para el usuario final. Compartir este modelo obligaba a hacer compromisos entre las necesidades de las herramientas de escritorio, el tiempo de ejecución y la

creación, todos con objetivos claros que debían cumplir para satisfacer a sus entornos respectivos y estas necesidades a veces entraban en conflicto entre ellas. Los artefactos descritos anteriormente como parte del modelo de programación general representan un equilibrio para cumplir las necesidades de estos entornos distintos.

La modificación de normas empresariales está limitada a sólo aquellos elementos que están definidos con plantillas en los conjuntos de normas y en las tablas de decisiones además de la tabla de selección de operación (fechas y destinos efectivos). La creación de conjuntos de normas y de tablas de decisiones nuevos sólo se admite a través de la copia de un conjunto de normas o de tabla de de decisiones existente. El propio componente del grupo de normas empresariales no puede crearse dinámicamente en tiempo de ejecución con excepción de las propiedades definidas por el usuario y los valores de descripción. Los cambios que deben realizarse al componente (por ejemplo, añadir una operación nueva) deben hacerse utilizando WebSphere Integration Developer y después volver a desplegarse o reinstalarse en el servidor.

Grupo de normas empresariales

La clase BusinessRuleGroup representa el componente del grupo de normas empresariales. La clase BusinessRuleGroup se puede considerar el objeto raíz que contiene conjuntos de normas y tablas de decisiones.

A los conjuntos de normas y de tablas de decisiones sólo se puede acceder a través del grupo de normas empresariales con el que están asociados. En la clase se incluyen métodos para recuperar información acerca del grupo de normas empresariales y para acceder a los conjuntos de normas y a las tablas de decisiones. Utilizando los métodos, se puede recuperar la información siguiente:

- Espacio de nombres de destino
- Nombre del grupo de normas empresariales
- Nombre de visualización
- Sincronización de nombre/nombre de visualización
- Descripción
- Huso horario de presentación que indica si las fechas se deben mostrar en formato UTC o de manera local para el sistema
- Las operaciones definidas en la interfaz asociada con el grupo de normas empresariales
- Propiedades personalizadas definidas en el grupo de normas empresariales

Se puede acceder a los diferentes conjuntos de normas y tablas de decisiones asociados con el grupo de normas empresariales mediante la operación del grupo de normas empresariales.

También existen métodos que permiten que la información se actualice en el grupo de normas empresariales. Utilizando los métodos se puede actualizar la información siguiente:

- Descripción
- Nombre de visualización
- Sincronización de nombre/nombre de visualización
- Propiedades personalizadas definidas en el grupo de normas empresariales

El nombre de visualización del grupo de normas empresariales se puede establecer explícitamente o se puede establecer en el valor del nombre con el método setDisplayNameIsSynchronizedToName.

Los demás valores no se pueden modificar, ya que forman parte de la definición de los componentes del grupo de normas empresariales y los cambios en estos valores obligarían a repetir el despliegue y la instalación.

La clase del grupo de normas empresariales también proporciona un método de renovación. Este método realizará una llamada al almacenamiento persistente o al repositorio en que están almacenadas las normas empresariales y devolverá el grupo de normas empresariales y todos los conjuntos de normas empresariales y tablas de decisiones asociados con la información persistida. El grupo de normas empresariales devuelto es la última copia y el objeto anterior es obsoleto.

El método isShell se puede utilizar para indicar si una instancia de grupo de normas empresariales es de una versión que no está admitida por el tiempo de ejecución actual. Por ejemplo, si se creó un cliente web con las clases de gestión de normas empresariales actuales y en el futuro se añaden capacidades nuevas al grupo de normas empresariales que no son compatibles por las clases, se creará un grupo de normas empresariales shell cuando se recupere el grupo de normas empresariales. Esto permite que el cliente web continúe trabajando con normas empresariales que sean compatibles y seguir recuperando grupos de normas empresariales con atributos y funcionalidades limitados. Cuando isShell es verdadero, sólo devolverán valores los métodos getName, getTargetNameSpace, getProperties, getPropertyValue y getProperty. Todos los demás métodos provocarán una UnsupportedOperationException. Además de usar el método isShell, también se puede comprobar el tipo de BusinessRuleGroup si se trata de una instancia de BusinessRuleGroupShell para poder determinar si se trata de una versión admitida.

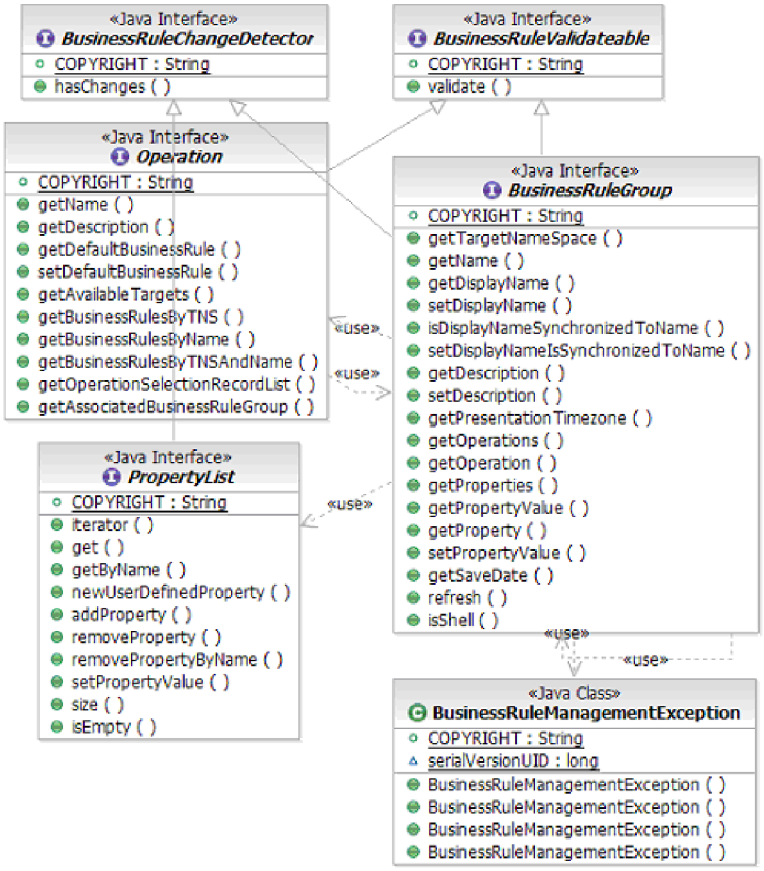


Figura 59. Diagrama de clase de BusinessRuleGroup y clases relacionadas

Propiedades del grupo de normas empresariales

Las propiedades de los grupos de normas empresariales están pensadas para utilizarlas para gestionar grupos de normas empresariales. Las propiedades establecidas en grupos de normas empresariales se pueden utilizar en consultas para devolver sólo un subconjunto de grupos de normas empresariales que primero se mostrarán y después modificarán.

Todas las propiedades son del tipo serie y se definen como parejas de nombre-valor. Cada propiedad sólo puede definirse una vez en un grupo de normas empresariales. Para cada propiedad definida, también debe tener definido un valor. El valor de propiedad puede ser una serie vacía o tener una longitud cero, pero no un valor nulo. Establecer una propiedad como un valor nulo equivale a suprimir dicha propiedad.

También se puede acceder a las propiedades de un grupo de normas empresariales en un conjunto de normas o en una tabla de decisiones en tiempo de ejecución. Ello permite establecer un valor individual en el grupo de normas empresariales que se utilice en varios conjuntos de normas o tablas de decisiones del grupo de normas empresariales. Sólo las propiedades definidas en el grupo de normas empresariales están disponibles para los conjuntos de normas y las tablas de decisiones incluidos.

Existen dos tipos de propiedades, del sistema y definidas por el usuario. El número de propiedades del sistema o definidas por el usuario no está limitado en un grupo de normas empresariales. Las propiedades del sistema se utilizan para contener información de componentes específica como la versión del modelo de normas utilizadas al definir la lógica de la norma. Esta información del sistema se expone en propiedades para permitir su consulta entre estos campos. Las propiedades del sistema empiezan con un prefijo IBMSystem y son de sólo lectura a través del grupo de normas empresariales y de clases de propiedad. Las propiedades del sistema no se pueden añadir, cambiar ni suprimir. Un ejemplo de una propiedad del sistema sería:

Nombre de propiedad	Valor de propiedad
IBMSystemVersion	6.2.0

Nota: Los valores de nombre, espacio de nombres y nombre de visualización para un grupo de normas empresariales se tratan como propiedades del sistema para propósitos de consulta y formarán parte de la lista de propiedades que se pueden recuperar para un grupo de normas empresariales con el método getProperties. Sin embargo, estas propiedades no se definen como elementos de propiedad reales en el artefacto del grupo de normas empresariales y no se consideran propiedades en WebSphere Integration Developer ya que se definen con elementos independientes y exclusivos en el grupo de normas empresariales. Se proporcionan exclusivamente para ofrecer más opciones de consulta.

Las propiedades definidas por el usuario están disponibles para utilizarlas para contener cualquier información específica del cliente y también se pueden utilizar en consultas para grupos de normas empresariales. Las propiedades definidas por el usuario están disponibles para lectura y escritura.

Las propiedades de un grupo de normas empresariales se pueden recuperar individualmente o como lista (objeto PropertyList). Con PropertyList, se proporcionan métodos para recuperar propiedades individuales y añadiendo y eliminando propiedades definidas por el usuario.

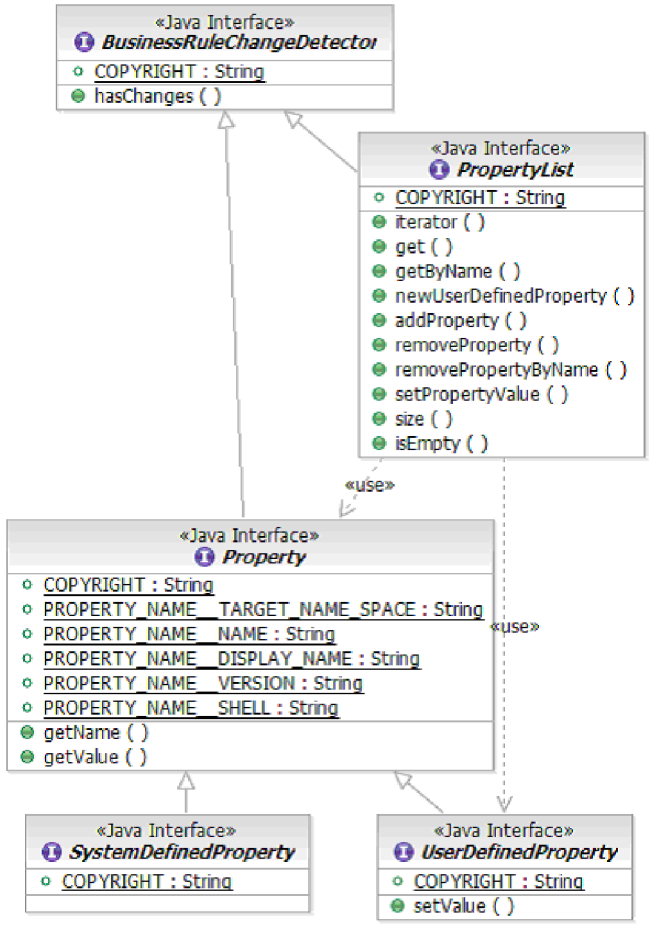


Figura 60. Diagrama de clase de Property y clases relacionadas

Operación

Las operaciones son los puntos de inicio para llegar a conjuntos de normas empresariales y tablas de decisiones que modificar. Las operaciones de un grupo de normas empresariales coinciden las operaciones que se listan en el WSDL asociado con el componente de grupo de normas empresariales.

Para cada operación, existen destinos distintos, cada uno de los cuales es una norma empresarial (conjunto de normas o tabla de decisiones):

- Destino por omisión (opcional)
- Lista de destinos planificados por rangos de fecha/hora (OperationSelectionRecord)
- Lista de todos los destinos disponibles que se pueden usar para esa operación

Todas las operaciones deben tener especificado al menos un destino de norma empresarial. Este destino puede ser un OperationSelectionRecord con fechas inicial y final específicas en que debe planificarse que el destino esté activo. La operación también puede tener un único conjunto de destinos por omisión que se utiliza durante la ejecución cuando no se encuentra ningún destino de norma empresarial planificada que coincida. La clase Operation incluye métodos para recuperar y establecer el destino de norma empresarial por omisión además de recuperar la lista (OperationSelectionRecordList) de destinos de normas empresariales

planificados. Además del destino de normas empresariales por omisión, y de los destinos de normas empresariales planificados, existe una lista de todos los destinos de normas empresariales disponibles para la operación. Esta lista incluirá aquellos destinos de normas empresariales que están planificados y el establecido por omisión, además de cualquier otro conjunto de normas o tabla de decisión que no están planificados para esta operación. Un conjunto de normas o una tabla de decisiones no planificados están asociados a la operación a través de la lista de destinos disponibles por el hecho de compartir implícitamente la información de la operación. Todos los destinos de normas empresariales deben admitir los mensajes de entrada y salida para sus operaciones. En cada operación exclusiva de un interfaz, los conjuntos de normas y las tablas de decisiones de una operación son exclusivos respecto de los de otra operación.

Cualquiera de los distintos conjuntos de normas y tablas de decisiones de la lista de destinos disponibles se puede planificar para que estén activos mediante la creación de un OperationSelectionRecord. Junto con el conjunto de normas o tabla de decisiones particular de la lista de destinos disponible, se debe especificar una fecha inicial y una fecha final. La fecha inicial debe ser anterior a la final. Las fechas pueden ser para un intervalo de tiempo que abarque la fecha actual, además del pasado y el futuro. El intervalo de tiempo de las fechas no se puede solapar con ningún otro OperationSelectionRecords en cuento se añade a OperationSelectionRecordList y se publica. Los valores de las fechas fecha inicial y final son del tipo Java.util.Date. Los valores que se especifiquen se considerarán valores UTC según la clase java.util.Date. Cuando OperationSelectionRecord está completo, se puede añadir a OperationSelectionRecordList para que se planifique junto con otros destinos de normas empresariales. Pueden existir huecos entre los intervalos de tiempo de distintos OperationSelectionRecords. Cuando se encuentra un hueco durante la ejecución, se utiliza el destino por omisión. Si no se ha especificado ningún destino por omisión, se generará una excepción. Se recomienda que especifique siempre un destino de norma empresarial por omisión.

Un destino de norma empresarial planificado puede eliminarse de la lista de destinos planificados eliminando el OperationSelectionRecord de la OperationSelectionRecordList. Al eliminar un OperationSelectionRecord no se eliminarán los destinos de norma empresarial de la lista de destinos de norma empresarial disponibles y no se eliminará ningún otro OperationSelectionRecords que tenga planificado el mismo destino de norma empresarial.

Además de recuperar un conjunto de normas o una tabla de decisiones mediante OperationSelectionRecordList o la lista de destinos disponibles, la clase Operation también permite recuperar destinos de normas empresariales por valores de propiedad nombre y espacio de nombres destino. Mediante los métodos de la clase Operation, se pueden consultar estos conjuntos de normas y tablas de decisiones que se listan en los destinos disponibles para esa operación. Los conjuntos de normas y las tablas de decisiones que pudieran tener valores de nombre y espacio de nombres destino coincidentes, pero que sean parte de las listas de destinos disponibles de otras operaciones, no se incluirán en el conjunto de resultados. Para mayor comodidad se incluyen los métodos getBusinessRulesByName, getBusinessRulesByTNS y getBusinessRulesByTNSAndName, que simplifican la recuperación de conjuntos de normas y tablas de de decisiones específicos.

La clase Operation incluye métodos que admiten lo siguiente:

- Recuperar el nombre de operación.
- Recuperar la descripción de operación.
- Recuperar y establecer el destino de norma empresarial por omisión.

- Recuperar los destinos de norma empresarial planificados (OperationSelectionRecordList).
- Recuperar la lista de todos los destinos de norma empresarial disponibles.
- Recuperar un conjunto de normas o tablas de decisiones a partir de la lista de todos los destinos disponibles por nombre o espacio de nombres destino.
- Recuperar el grupo de normas empresariales al que está asociado la operación.

La clase OperationSelectionRecordList incluye métodos que admiten lo siguiente:
- Recuperar un OperationSelectionRecord determinado por valor de índice.
- Eliminar un OperationSelectionRecord determinado por valor de índice.
- Añadir un OperationSelectionRecord nuevo a la lista.

La clase OperationSelectionRecord incluye métodos que admiten lo siguiente:
- Recuperar y establecer la fecha inicial.
- Recuperar y establecer la fecha final.
- Recuperar y establecer el destino de la norma empresarial.
- Recuperar la operación a la que está asociada OperationSelectionRecord.

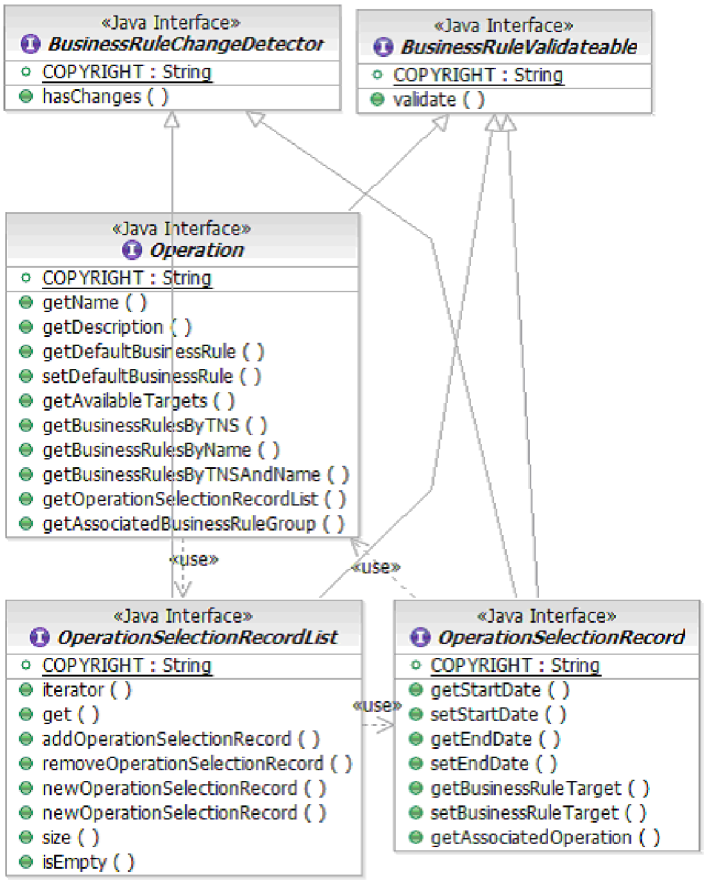


Figura 61. Diagrama de clase para Operation y clases Relacionadas

Norma empresarial

Las clases RuleSet y DecisionTable están basadas en una clase BusinessRule genérica con métodos que proporcionan información que está disponible en conjuntos de normas y tablas de decisiones.

Los conjuntos de normas y las tablas de decisiones, al igual que los artefactos de grupos de normas empresariales, tienen un nombre y un espacio de nombres de destino. La combinación de estos valores debe ser exclusiva al compararla con otros conjuntos de normas y tablas de decisiones. Por ejemplo, dos conjuntos de normas puede compartir el mismo valor de espacio de nombres de destino, pero deben tener nombres distintos o un conjunto de normas y una tabla de decisiones podrían tener el mismo nombre, pero tener diferentes valores de espacio de nombres de destino.

Se puede realizar una copia de una norma empresarial a partir de una existente en situaciones en que se necesite planificar una norma similar en un momento específico con valores de parámetro distintos, para normas construidas a partir de plantillas. Las normas nuevas no pueden crearse partiendo totalmente de cero ya que debe existir una clase Java de respaldo que proporcione la implementación para la norma empresarial. La clase Java de respaldo sólo se crea en el momento del despliegue. Cuando se crea una norma nueva, se agrega a la lista de destinos disponibles para la operación que está asociada a la norma original. Sin embargo, la norma adicional no se persiste hasta que se publica el grupo de normas empresariales con el que está asociada la operación.

La nueva norma empresarial debe tener un espacio de nombres de destino o un nombre distintos del de la norma original. El nombre de visualización de la norma empresarial nueva puede seguir siendo el mismo que el de la original ya que la combinación de nombre y espacio de nombres proporciona un valor clave para identificar la norma empresarial. Dentro de la norma empresarial, se pueden modificar los distintos valores de parámetros que se han definido con una plantilla. Se puede planificar la norma empresarial en un momento determinado con OperationSelectionRecordList o como destino por omisión con la operación asociada a la norma empresarial.

La clase BusinessRule ofrece métodos que admiten lo siguiente:

- Recuperar el espacio de nombres de destino
- Recuperar el nombre del conjunto de normas o de la tabla de decisiones
- Recuperar y establecer el nombre de visualización del conjunto de normas o de la tabla de decisiones
- Recuperar el tipo de norma empresarial, ya sea conjunto de normas o tabla de decisiones
- Recuperar y establecer la descripción para la norma empresarial
- Recuperar la operación con la que está asociada la norma empresarial
- Crear una copia de la norma empresarial con un nombre y/o espacio de nombres de destino distintos

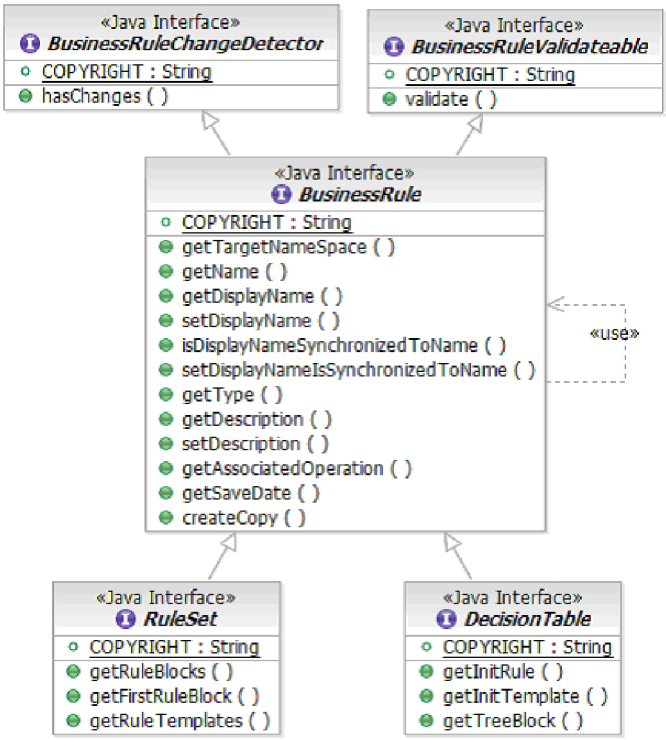


Figura 62. Diagrama de clase de BusinessRule y clases relacionadas

Conjunto de normas

Un conjunto de normas es un tipo de norma empresarial. Los conjuntos de normas normalmente se utilizan cuando resulta necesario ejecutar varias normas según valores condicionales distintos. Los conjuntos de normas están compuestos por un bloque de normas y plantillas de normas. El bloque de normas (`RuleBlock`) contiene las normas "if-then" y de acción que componen la lógica del conjunto de normas.

La clase `RuleSet` incluye métodos que admiten lo siguiente:

- Recuperar una lista de bloques de normas para el conjunto de normas
- Recuperar una lista de plantillas de normas definida en el conjunto de normas

Actualmente cada conjunto de normas sólo puede tener un bloque de normas, mientras que pueden haber varias plantillas de norma definidas en el conjunto de normas. El bloque de normas contiene el conjunto de normas que se ejecutarán cuando éste se invoque. El bloque de normas permite modificar el orden de las normas. Un bloque de normas debe tener definida al menos una norma. Las normas (`Rule`) se pueden definir como normas de instancia de plantilla (TemplateInstanceRule) o codificarse. Si se ha definido una norma "if-then" o de acción con una plantilla, se pueden eliminar del bloque de normas. Si se ha creado una instancia nueva de norma con una plantilla, se puede añadir al bloque de normas.

Si una norma está codificada y no se definió con una plantilla, no se puede modificar ni eliminarse del bloque de normas. Lo que se espera de estas normas es que se hayan diseñado para que siempre formen parte de la lógica del conjunto de normas y no se cambien ni repitan dentro de la lógica.

Cuando se crea una norma nueva con una plantilla, debe tener un valor de nombre exclusivo. La lista de normas existentes se puede recuperar y comprobar antes de crear la norma.

En las normas "if-then" y de acción codificadas, sólo se puede recuperar el nombre y la presentación. La presentación es una serie que se puede usar para mostrar información sobre la norma en aplicaciones cliente. Para normas "if-then" y de acción que se definen con una plantilla, se puede recuperar el nombre y la presentación, además de información adicional. Se pueden recuperar y cambiar valores de parámetro específicos. Con una plantilla (`RuleSetRuleTemplate`) definida en el conjunto de normas, se puede crear otra instancia de la norma dentro del conjunto de normas y se pueden establecer valores de parámetros. Por ejemplo, si tiene una norma que dice que un cliente de un nivel de estado determinado recibe un descuento de un importe específico. Esta lógica podría definirse con una plantilla de norma única y después repetirse cambiando los valores de los parámetros según el nivel de estado (oro, plata, bronce, etc.) y el importe del descuento (15%, 10%, 5%, etc).

Los parámetros para una norma definida con una plantilla son específicos para la instancia de la norma. La plantilla sólo define una presentación estándar y el número de parámetros para la norma. Todas las normas definidas con una plantilla pueden tener valores distintos, tal como se explica en e ejemplo sobre descuentos para estados de cliente distintos.

La clase RuleBlock incluye métodos que admiten lo siguiente:

- Recuperar una norma por índice
- Añadir una norma que se definió con una plantilla
- Eliminar una norma definida con una plantilla
- Modificar el orden de una norma en una posición o ir a una ubicación de índice específica

La clase `RuleSetRule` incluye métodos que admiten lo siguiente:

- Recuperar el nombre de la norma
- Recuperar el nombre de visualización de la norma
- Recuperar la presentación de usuario
- Recuperar el bloque de normas

La clase `RuleSetRuleTemplate` ofrece métodos que admiten lo siguiente:

- Crear una instancia de plantilla de norma a partir de esta definición de plantilla
- Recuperar el conjunto de normas padre

La clase `TemplateInstanceRule` incluye métodos que admiten lo siguiente:

- Recuperar los parámetros de la norma
- Recuperar la definición de plantilla que definió la norma

La clase `Template` incluye métodos que permiten lo siguiente:

- Recuperar el ID de plantilla
- Recuperar el nombre
- Recuperar y establecer el nombre de visualización
- Recuperar y establecer la descripción
- Recuperar los parámetros de esta plantilla

- Recuperar la presentación de usuario

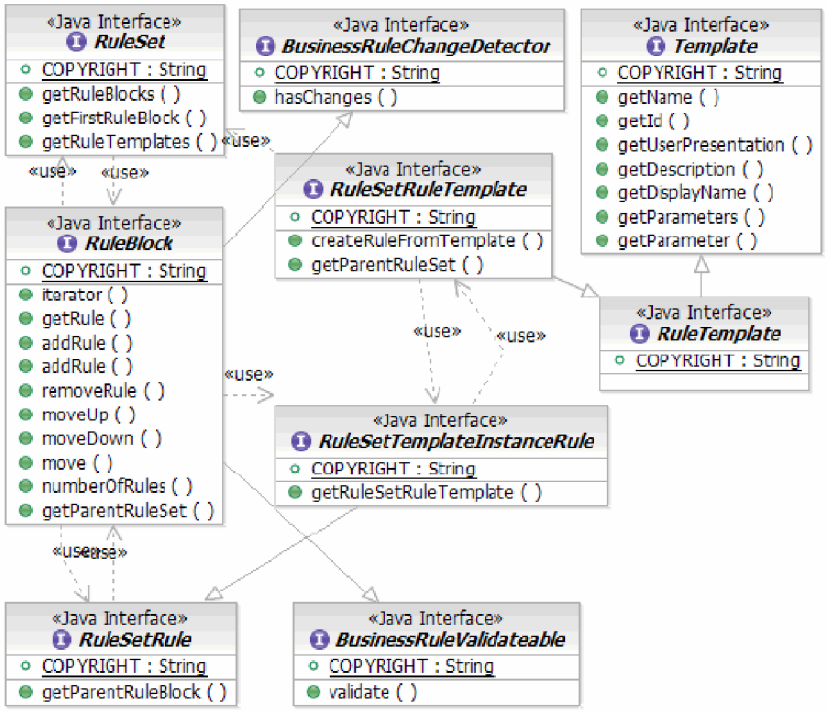


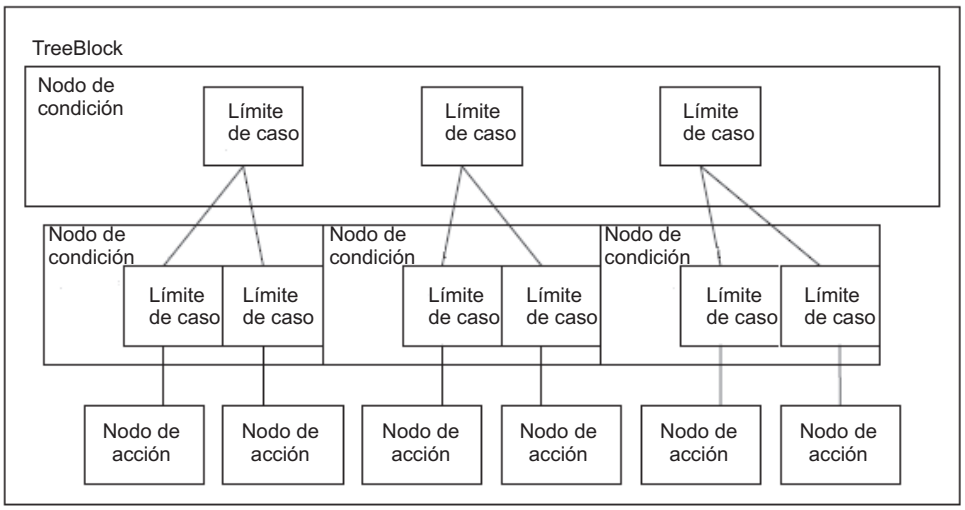
Figura 63. Diagrama de clase de BusinessRule y clases relacionadas

Tabla de decisiones

Las tablas de decisiones son otro tipo de norma empresarial que se puede gestionar y modificar. Las tablas de decisiones suelen utilizarse cuando hay un número coherente de condiciones que deben evaluarse y un conjunto específico de acciones que emitir cuando se cumplen las condiciones.

Las tablas de decisiones son parecidas a los árboles de decisiones, aunque en este caso están equilibradas. Las tablas de decisiones siempre tienen el mismo número de condiciones que evaluar y acciones que realizar, cualquiera que sea el conjunto de ramas que se resuelvan como verdaderas. Un árbol de decisiones puede tener una rama con más condiciones que evaluar que otra rama.

Las tablas de decisiones están estructuradas como árbol de nodos y las define un `TreeBlock`. Existen `TreeNodes` distintos que componen el `TreeBlock`. Los `TreeNodes` pueden ser nodos de condición o de acción. Los nodos de condición son las ramas de evaluación. Al final de las ramas, existen nodos de acción que tienen las acciones de árbol apropiadas que emitir en caso que todas las condiciones se evalúen como verdaderas. Puede existir cualquier número de niveles de nodos de condición, pero sólo un nivel de nodos de acción.



Las tablas de decisión también podrían tener una norma de inicialización (norma init) que se puede emitir antes de comprobar las condiciones de la tabla.

La clase `DecisionTable` incluye métodos que admiten lo siguiente:

- Recuperar el bloque de árbol de nodos de árbol (nodos de condición y de acción)
- Recuperar la instancia de norma de inicialización
- Recuperar la plantilla de norma de inicialización, si estuviera definida

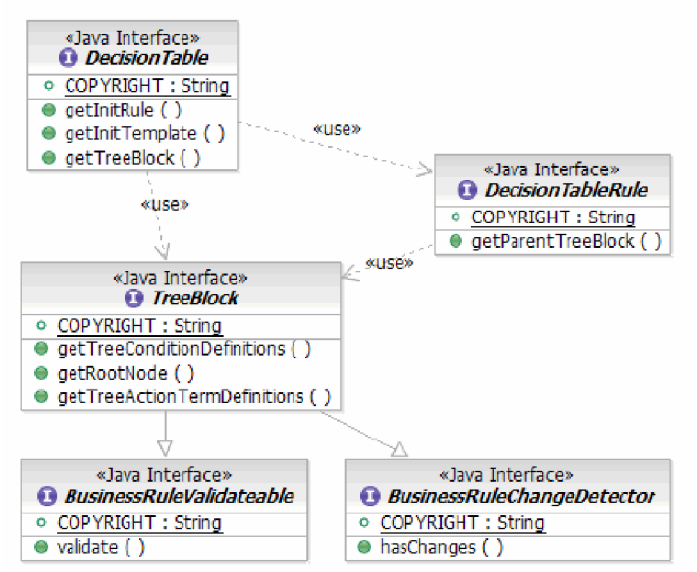
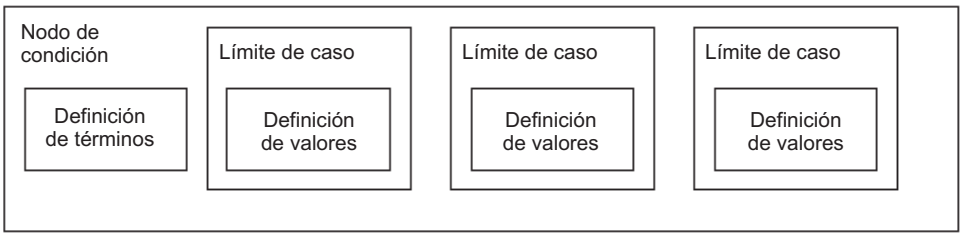


Figura 64. Diagrama de clase de DecisionTable y clases relacionadas

El `TreeBlock` de una tabla de decisión contiene la condición distinta y los nodos de acción. Cada nodo de condición (`ConditionNode`) tiene una definición de término (`TreeConditionTermDefinition`) y entre uno y n límites de caso (`CaseEdge`). La definición del término contiene el operando de la izquierda para la expresión de la condición. Los límites de caso contienen las definiciones de valor que son los

operandos derechos distintos que utilizar en la expresión de condición. Por ejemplo, en la expresión (status == “oro”) la definición de término sería “status” y “oro” sería la definición de valor en el límite de caso. Todos los límites de caso de un nodo de condición comparten la definición de término y sólo se diferencian en el valor (`TreeConditionValueDefinition`). Continuando con el ejemplo, otro límite de caso del nodo de condición podría tener un valor “plata”. Esto se utilizaría también en una expresión (status == “plata”). La única excepción a este comportamiento es si se ha definido un "otherwise" para el nodo de condición. Con un "otherwise" no existe ninguna definición de valor, ya que se utiliza si todos los demás límites de caso del nodo de condición se evalúan como falsos. Mientras un "otherwise" no es un límite de caso, tiene un `TreeNode` que se puede recuperar.



Para la definición de término, la presentación de usuario se puede recuperar y utilizar en aplicaciones cliente. La presentación para la definición de término normalmente sólo es una representación del operando izquierdo (en nuestro ejemplo estado) y no contiene ningún contenedor. Para los límites de caso, se puede utilizar una plantilla para definir la definición de valor (`TreeConditionValueTemplate`). Una instancia de definición de valor de plantilla (`TemplateInstanceExpression`) contiene los valores de parámetro que se utilizan para ejecución y se pueden modificar. Si se realiza un intento de recuperar la definición de plantilla del valor para una `TreeConditionValueDefinition` que no se definió con una plantilla, se devolverá un valor nulo. Si no se ha utilizado una plantilla para definir la condición del valor, se puede seguir recuperando una presentación de usuario y utilizarla en aplicaciones cliente, si se especificó en tiempo de creación.

La clase `TreeBlock` incluye métodos que admiten lo siguiente:

- Recuperar el nodo raíz del árbol
- Recuperar las definiciones de término de condición para el bloque de árbol
- Recuperar las definiciones del término de acción para el bloque de árbol

El nodo raíz del árbol es del tipo `TreeNode` y, a partir de aquí, se puede producir la navegación de la tabla de decisiones. La clase `TreeNode` incluye métodos que admiten lo siguiente:

- Determinar si un nodo es una cláusula otherwise
- Recuperar el nodo padre para el nodo de árbol actual (nodo de condición o de acción)
- Recuperar el nodo raíz del árbol que contiene el nodo del árbol actual

La clase `ConditionNode` incluye métodos que admiten lo siguiente:

- Recuperar los límites de caso
- Recuperar la definición de término

- Recuperar el caso otherwise
- Recuperar las plantillas de las condiciones de valor de los límites de caso para el nodo de condición
- Añadir al nodo un valor de condición basado en una plantilla
- Eliminar un valor de condición basado en una plantilla

La clase `CaseEdge` incluye métodos que admiten lo siguiente:
- Recuperar la lista de plantillas de valor que están disponibles para la definición de valor
- Recuperar el nodo hijo (nodo de condición o de acción)
- Recuperar la instancia de la definición de plantilla asociada con la definición de valor
- Recuperar la definición de valor directamente sin recuperar la plantilla
- Establecer el valor para que la definición use una definición específica de instancia de plantilla

La clase `TreeConditionTermDefinition` incluye métodos que admiten lo siguiente:
- Recuperar las plantillas de definición de valor definidas para el nodo de condición
- Recuperar la presentación de usuario del término de condición

La clase `TreeConditionDefinition` incluye métodos que admiten lo siguiente:
- Recuperar la definición de término para el nodo de condición
- Recuperar las definiciones de valor de condición para el nodo de condición de todos los límites de caso
- Recuperar la orientación (fila o columna)

La clase `TreeConditionValueDefinition` incluye métodos que admiten lo siguiente:
- Recuperar la expresión de instancia de plantilla específica para el valor
- Recuperar el usuario

La clase `Template` incluye métodos que admiten lo siguiente:
- Recuperar el ID de sistema para la plantilla
- Recuperar el nombre de la plantilla
- Recuperar los parámetros definidos para la plantilla
- Recuperar la presentación para la plantilla

La clase `TreeConditionValueTemplate` proporciona un método que admite lo siguiente:
- Crear una nueva instancia de valor de condición de plantilla

La clase `TemplateInstanceExpression` incluye métodos que admiten lo siguiente:
- Recuperar los parámetros para la instancia de plantilla
- Recuperar la plantilla (`TreeConditionValueTemplate` en el caso de un límite de caso de una tabla de decisiones) que se utilizó para definir la instancia

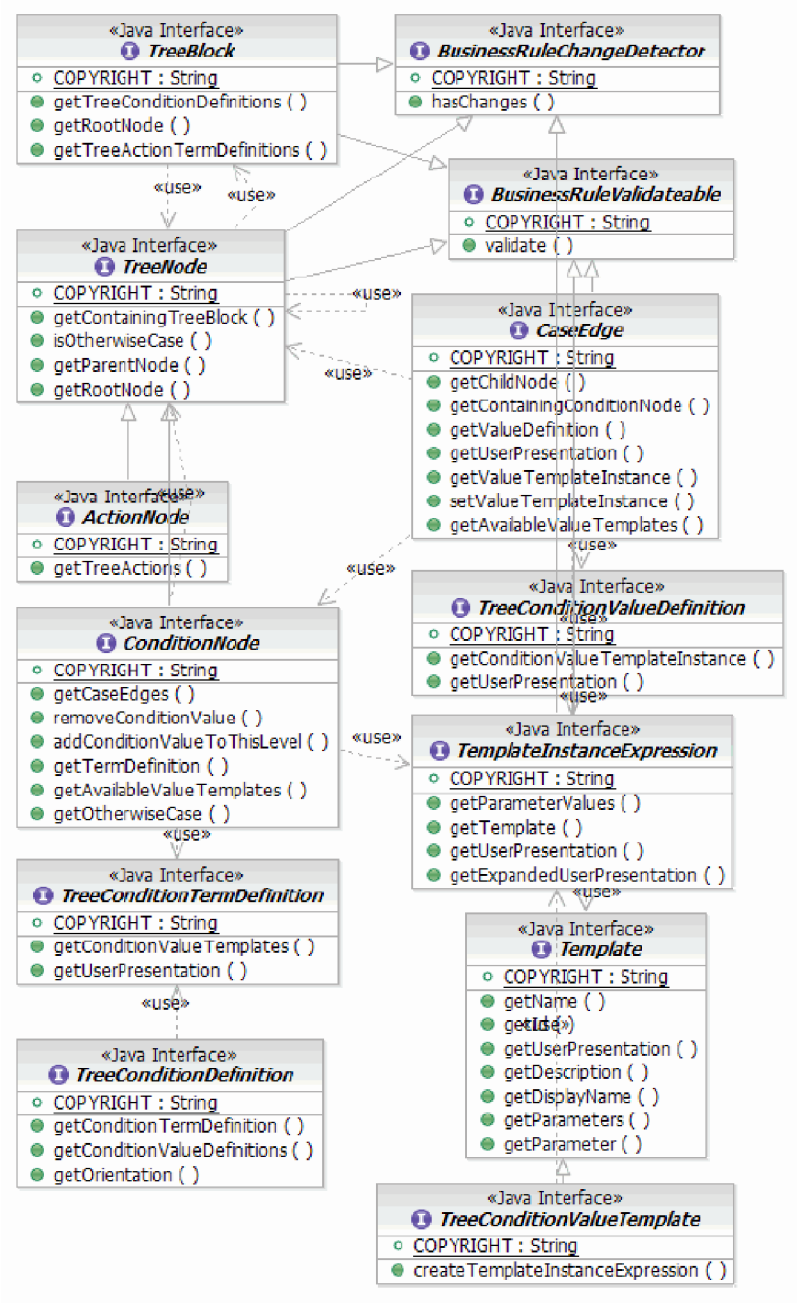


Figura 65. Diagrama de clase para TreeNode y clases relacionadas

Cuando se añade un límite de caso nuevo a un nodo de condición, el límite de caso nuevo debe usar una plantilla para definir el valor. Por ejemplo, si se fuera a añadir un límite de caso nuevo “bronce” para comprobar ‘status’, sería necesario usar la plantilla apropiada (`TreeConditionValueTemplate`) para crear una TemplateInstanceExpression nueva, estableciendo el valor de parámetro en “bronce”.

Cuando se añade un límite de caso nuevo, también se le añadirá automáticamente un nodo de condición hijo. Este nodo de condición hijo contendrá límites de caso que están basados en las definiciones de límite de caso que se han definido para nodos de condición a ese mismo nivel. Si se utilizan plantillas o valores codificados en límites de caso, también se utilizarán en los límites de caso del nodo de

condición hijo. El nodo de condición hijo que se añade automáticamente también tendrá sus propios nodos de condición hijos creados automáticamente. Estos nodos de condición hijos también tendrán nodos de condición hijos y así sucesivamente hasta que se hayan recreado todos los niveles de nodos de condición.

Además de los nodos de condición, una tabla de decisiones y más específicamente un bloque de árbol, también contiene un nivel de nodos de acción (`ActionNode`). Los nodos de acción son nodos de hoja y residen al final de la rama de nodos de condición y los límites de caso. Si todos los valores de condición de una línea de límites de caso se resuelven como verdaderos, se accede a un nodo de acción. El nodo de acción tendrá definido al menos una acción (`TreeAction`). Para la acción, existirá una definición de término y otra de valor. Al igual que con los nodos de condición, la definición de término (`TreeActionTermDefinition`) es el lado izquierdo de la expresión y la definición de valor (`TemplateInstanceExpression`) es el lado derecho de la expresión. Por ejemplo, para los nodos de condición distintos que comprobaban el estado, podría haber acciones para definir el descuento. Si la condición era (status == "oro"), la acción puede ser (discountValue = 0,90). Para la acción el "discountValue" sería la definición del término y "= 0.90" sería la definición del valor.

La definición del término de una acción de árbol se comparte con otras acciones de árbol de otros nodos de acción. Debido a que cada rama de límites de caso accede a una acción, se utilizan las mismas definiciones de término. Sin embargo, las definiciones de valor pueden ser distintas por acción de árbol y nodo de acción. Por ejemplo, el discountValue de un estado "oro" puede ser "0,90"; aunque el "discountValue" de un estado "plata" puede ser "0,95".

Los nodos de acción pueden tener acciones de árbol múltiples que tienen una definición de término y definición de valor independientes. Por ejemplo, si se estaba determinando el descuento de un automóvil de alquiler, además de establecer el discountValue, también puede desear asignar un nivel específico de automóvil. Podría crearse otra acción de árbol para establecer el término "carSize" en "tamaño completo" si el estado era "oro" además de establecer "discountValue" en "0,90".

La definición de valor en una acción de árbol se puede crear a partir de una plantilla (`TreeActionValueTemplate`). La definición de plantilla contiene una expresión (`TemplateInstanceExpression`) que tiene los parámetros para la expresión.

Además de cambiar los parámetros, se puede modificar la definición de valor completa por una instancia de definición de valor nueva que se crea con otra plantilla que se definió para la acción de árbol.

Si una definición de valor no se crea con una plantilla, no se puede cambiar. Para aplicaciones cliente, se puede utilizar la presentación de usuario en la visualización si se especificó en el momento de la creación.

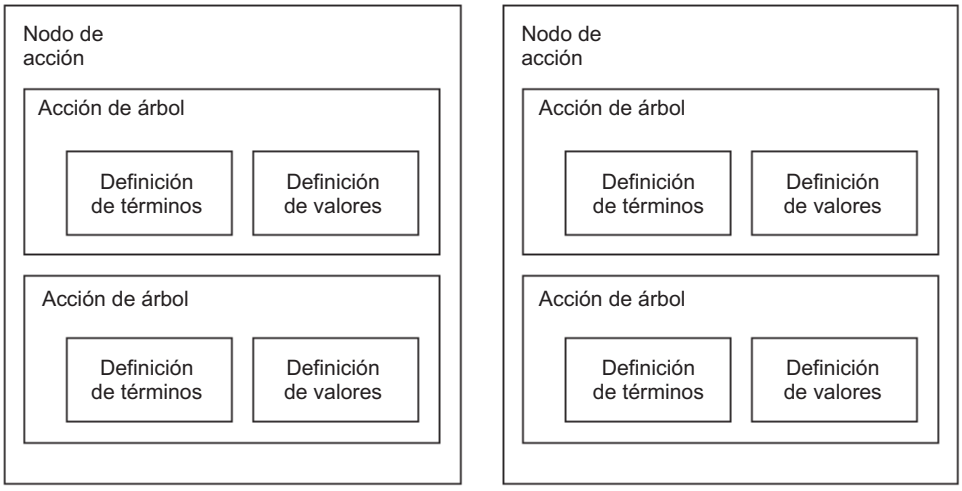
Para las definiciones de término de las acciones de árbol, si se ha especificado una presentación de usuario, también las pueden utilizar aplicaciones cliente.

Cuando se añade un límite de caso nuevo a un nodo de condición y se crean los diferentes nodos de condición hijos, también se crearán los nodos de acción. A diferencia de los nodos de condición hijos y los límites de caso que se crean en base a la definición de los límites de caso ya definidos para ese nivel, los nodos de acción no heredan automáticamente un diseño existente. En el nodo de acción sólo

se crean contenedores TreeActions vacíos. Debe utilizarse una plantilla (TreeActionValueTemplate) para completar la definición de acción creando una TemplateInstanceExpression para al menos una definición de término para el nodo de acción. Hasta que se establezca la acción de árbol con una TemplateInstanceExpression, la acción de árbol tendrá valores nulos especificados para el valor de presentación de usuario y el valor de instancia de plantilla.

Al crear una condición nueva que provoque en ActionNodes nuevos, los nodos de acción se añadirán a la derecha de acciones existentes para el nodo de condición inmediatamente superior. Por ejemplo, si se añade un estado "rubí" a la tabla de decisiones y debería tener un descuento específico, la condición para comprobar el estado se añade a la derecha de "oro", "plata" y "bronce". El nodo de acción para el descuento de "rubí" se añadirá a la derecha de los nodos de acción que corresponden a los límites de caso "oro", "plata" y "bronce".

Al establecer acciones de árbol nuevas para nodos de acción, un algoritmo que examina el nodo de acción de más a la derecha del límite de caso inferior devolverá el nodo de acción con una acción de árbol vacío. También se puede comprobar que la acción de árbol tenga valores nulos para los valores de presentación de usuario y de instancia de plantilla. Cuando se obtiene la acción de árbol, se puede establecer con la instancia correcta de una TreeActionValueTemplate.



La clase ActionNode proporciona un método que admite lo siguiente:

- Recuperar una lista de las acciones de árbol definidas

La clase TreeAction incluye métodos que admiten lo siguiente:

- Recuperar una lista de las plantillas de valores disponibles definidas para la acción de árbol
- Recuperar la definición de término
- Recuperar la instancia de plantilla de valor definida para la acción de árbol
- Recuperar la presentación de usuario para el valor si no se utilizó una plantilla de valor
- Comprobar si la acción es una invocación de servicio SCA (método isValueNotApplicable)

- Sustituir la instancia de plantilla de valor por una instancia nueva

La clase `TreeActionTermDefinition` incluye métodos que admiten lo siguiente:
- Recuperar la presentación de usuario para la definición del valor del término
- Recuperar una lista de las plantillas de valor disponibles para la acción de árbol
- Comprobar si la acción es una invocación de servicio SCA (método `isTermNotApplicable`)

La clase `Template` incluye métodos que admiten lo siguiente:
- Recuperar el ID de sistema para la plantilla
- Recuperar el nombre de la plantilla
- Recuperar los parámetros definidos para la plantilla
- Recuperar la presentación para la plantilla

La clase `TreeActionValueTemplate` proporciona un método que admite lo siguiente:
- Crear una instancia de plantilla de valor nueva a partir de la definición de la plantilla

La clase `TemplateInstanceExpression` incluye métodos que admiten lo siguiente:
- Recuperar los parámetros para la instancia de plantilla
- Recuperar la plantilla (`TreeActionValueTemplate` en el caso de una acción de árbol de una tabla de decisiones) que se utilizó para definir la instancia

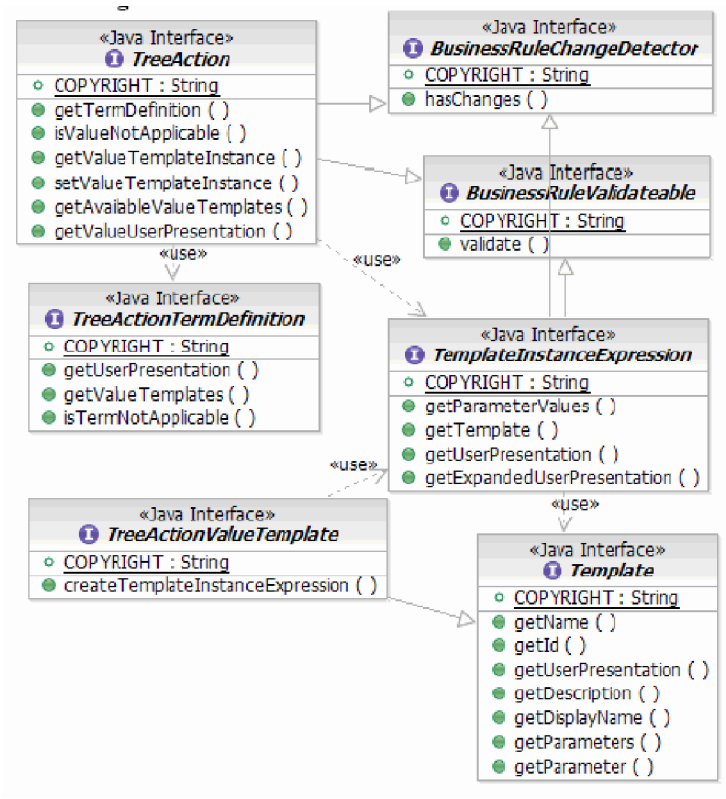


Figura 66. Diagrama de clase de TreeAction y clases relacionadas

La definición de una norma de inicialización para una tabla de decisiones sigue la misma estructura que una norma en un conjunto de normas. La norma de inicialización se puede definir con una plantilla (`DecisionTableRuleTemplate`).

Si no se creó una norma de inicialización en el momento de la creación, no se puede añadir cuando la norma está desplegada.

La clase `Rule` incluye métodos que admiten lo siguiente:

- Recuperar el nombre de la norma
- Recuperar la presentación de usuario para la norma
- Recuperar la presentación de usuario para la norma con los parámetros distintos completados para la norma

La clase `DecisionTableRule` proporciona un método que admite lo siguiente:

- Recuperar el bloque de árbol que contiene la norma de inicialización

La clase `DecisionTableRuleTemplate` proporciona un método que admite lo siguiente:

- Recuperar la tabla de decisiones que contiene la plantilla

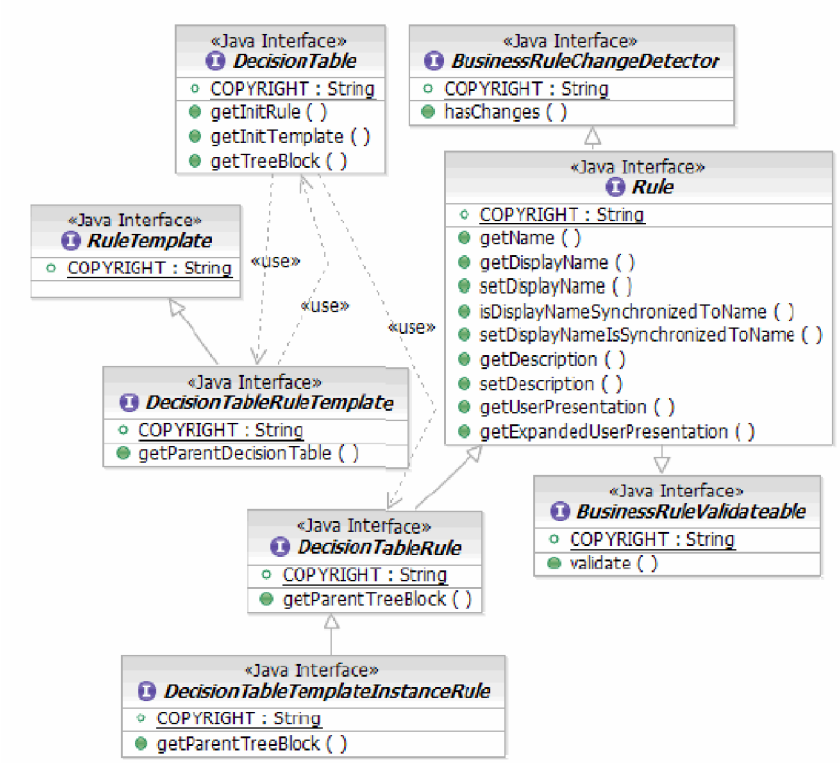


Figura 67. Diagrama de clase para DecisionTableRule y clases relacionadas

Plantillas y parámetros

Las plantillas de los conjuntos de normas y tablas de decisiones están basados en una definición común. Las plantillas tienen parámetros y una presentación de usuario. Los valores de parámetro de plantilla están definidos para permitir que se realicen cambios en la norma cuando se haya desplegado.

El valor de presentación de usuario proporciona un valor de cadena que se puede utilizar para mostrar la norma y parámetros distintos de manera intuitiva para el usuario. La presentación de usuario, que es una serie, tiene contenedores que permiten sustituir y mostrar correctamente los distintos valores de los parámetros. Los contenedores tienen el formato {<índice de parámetro>}. Por ejemplo, si la serie de presentación para el valor de inicialización es "El descuento base es {0} %", el contenedor {0} podría sustituirse por el valor del parámetro. La serie de presentación no se puede cambiar por la norma o la definición de plantilla. Sin embargo, los valores del contenedor se pueden modificar con los valores del parámetro en una aplicación cliente según la definición de la plantilla. Las distintas plantillas incluyen un práctico método (`getExpandedUserPresentation`) que devuelve una serie que tiene todos los valores de parámetro correctamente situados en la serie.

Todos los valores de parámetro tienen un tipo de datos específico, aunque al recuperar y establecer un valor de parámetro se utiliza un objeto de serie. El valor del parámetro se puede considerar una serie al sustituir lo en la presentación de usuario y también al establecer el parámetro con un valor nuevo. El parámetro se convierte en el tipo de datos correcto en tiempo de ejecución para emitir correctamente la norma en tiempo de ejecución. Durante la validación, el valor del parámetro se comparará con el tipo de datos para garantizar que sea correcto. Por ejemplo, si un parámetro es del tipo `boolean` y se establece en "T", la validación no reconocerá este valor y devolverá un problema.

En la definición de plantilla, se pueden restringir los valores de los parámetros. Las restricciones se pueden definir como rango o en una enumeración. Las restricciones para el parámetro se aplicarán cuando se valide la norma. Si no se utilizó una plantilla para la definición del valor, sólo estará disponible una presentación de usuario. Una definición de valor no puede tener una plantilla y una presentación de usuario. Si se utiliza una plantilla, la presentación de la definición de plantilla es la única presentación que está disponible.

La clase `Template` incluye métodos que permiten lo siguiente:

- Recuperar el ID de plantilla
- Recuperar el nombre
- Recuperar los parámetros
- Recuperar la presentación de usuario

La clase `Parameter` incluye métodos que permiten lo siguiente:

- Recuperar el nombre del parámetro
- Recuperar el tipo de datos del parámetro
- Recuperar la restricción para el parámetro
- Recuperar la plantilla que define el parámetro
- Crear un valor de parámetro

La clase `ParameterValue` incluye métodos que permiten lo siguiente:

- Recuperar el nombre del parámetro
- Recuperar el valor del parámetro
- Establecer el valor del parámetro

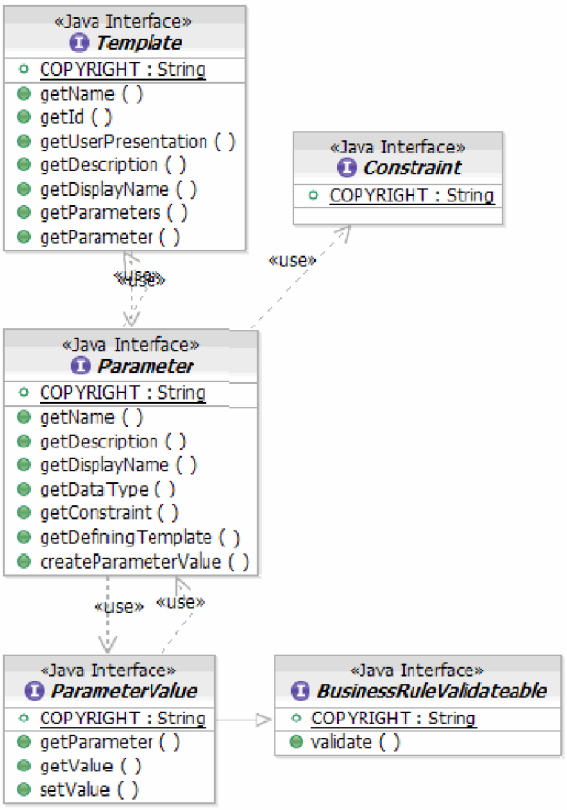


Figura 68. Diagrama de clase para Template, Parameter y clases relacionadas

Validación

Muchos de los objetos principales tienen un método de validación que permite comprobar la corrección y la finalización de los artefactos antes de publicarlos.

La validación que se produce al realizar los cambios a través de las clases API sólo es un subconjunto adecuado de la validación general que se produce durante serviceDeploy o al editar los artefactos en WebSphere Integration Developer. Ello es debido a las restricciones que ya se aplican al grupo de normas empresariales y que limitan los aspectos que son editables en tiempo de ejecución. El usuario de las clases puede validar la tabla de selección de grupos de normas empresariales, el conjunto de normas o las tablas de decisiones siempre que sea necesario (el propio componente del grupo de normas no puede editarse en tiempo de ejecución). Cuando se publica un grupo de normas empresariales, la tabla de selección de grupos de normas empresariales, los conjuntos de normas y las tablas de decisiones se validarán antes de publicarlos en el depósito.

Si los artefactos no son válidos, se generará una ValidationException con una lista de los problemas de validación. Los distintos problemas de validación se documentan en la sección Manejo de excepciones.

Seguimiento de cambios

En todos los objetos hay disponible un método `hasChanges` para comprobar si se han producido modificaciones en el objeto y en cualquier objeto que contenga.

Este método se puede usar para comprobar los cambios y publicar solamente un grupo de normas empresariales si tiene elementos que hayan cambiado.

BusinessRuleManager

La clase `BusinessRuleManager` es la principal para trabajar con los grupos de normas empresariales, conjuntos de normas y tablas de decisiones.

`BusinessRuleManager` tiene métodos que permiten recuperar grupos de normas empresariales por nombre, espacio de nombres de destino o propiedades personalizadas. También tiene un método para publicar cambios que se han realizado a grupos de normas empresariales, conjuntos de normas o tablas de decisiones.

La clase `BusinessRuleManager` incluye métodos que admiten lo siguiente:

- Recuperar todos los grupos de normas empresariales
- Recuperar los grupos de normas empresariales de un espacio de nombres destino específico
- Recuperar grupos de normas empresariales de un nombre específico
- Recuperar los grupos de normas empresariales de un nombre y espacio de nombres destino específicos
- Recuperar grupos de normas empresariales que contienen una propiedad específica
- Recuperar grupos de normas empresariales que contienen propiedades específicas
- Publicar grupos de normas empresariales

Figura 69. Diagrama de clase para BusinessRuleManager y el paquete

Consulta de componente del grupo de normas

El componente del grupo de normas puede tener propiedades definidas por el usuario (pares de nombre/valor) que se pueden utilizar para restringir la lista de grupos de normas empresariales que devuelve la clase. Estos son los campos que se pueden usar en la consulta y en cualquier combinación:

- Espacio de nombres destino de componente de grupo de normas empresariales
- Nombre de componente del grupo de normas empresariales

- Nombre de propiedad
- Valor de propiedad

Cada nombre de propiedad sólo puede definirse una vez por grupo de normas empresariales.

La función de consulta que admite esta clase es un pequeño subconjunto del lenguaje SQL completo. El usuario no proporciona la sentencia SQL, proporciona los valores como parámetros para una sola propiedad o para una estructura de árbol que contiene la información para una consulta de propiedad múltiple en forma de nodos. Hay nodos de operador lógico y nodos de consulta de propiedad que implementan la interfaz `QueryNode`. Los nodos de operador lógico especifican los operadores booleanos (AND, OR, NOT). Estos se crean mediante `QueryNodeFactory`. Como parte de la creación de estos nodos de operador lógico, se deben especificar los operadores izquierdo y derecho con las clases `QueryNode` adicionales. Estos nodos pueden ser de consulta de propiedad u otro nodo de operador lógico. Si se pasa un nodo de consulta de propiedad, contendrá el nombre, valor y operador (EQUAL (==), NOT_EQUAL (!=) LIKE o NOTLIKE) de la propiedad. El `QueryNode` general lo analiza la clase y se realiza una consulta en los datos subyacentes del almacenamiento persistente.

Las búsquedas con comodines se admiten cuando se utilizan los operadores LIKE y NOTLIKE. En las búsquedas con comodín se admiten los caracteres '%' y '_'. El carácter '%' se utiliza cuando existe un número infinito de caracteres que se desconocen o que no deberían considerarse en la búsqueda. Por ejemplo, si se deseara realizar una búsqueda de todos los grupos de normas empresariales que tienen una propiedad con un nombre de departamento y valor que empiece con "Norte", el valor se especificaría como "Norte%". Otro ejemplo, supongamos que se desean todos los departamentos cuyo valor termina en "Región". El valor sería "%Región". El carácter '%' también se puede utilizar en mitad de una serie. Por ejemplo, si existieran grupos de normas empresariales que tuvieran valores de "NorthCentralRegion", "NorthEastRegion" y "NorthWestRegion", se podría especificar un valor de "North%Region".

El carácter '_' se utiliza cuando existe un carácter individual que es desconocido o no debería considerarse al realizar búsquedas. Por ejemplo, si se deseara una búsqueda de todos los grupos de normas empresariales cuya propiedad de Department tuviera los valores "Dept1North", "Dept2North", "Dept3North" y "Dept4North", se podría especificar un valor "Dept_North" y se devolverían los cuatro grupos de normas empresariales con estas propiedades. El carácter '_' se puede usar varias veces en un valor de búsqueda indicando en cada instancia que hay un carácter que ignorar. El carácter '_' se puede utilizar al principio o al final de un valor. Por ejemplo, si se tuvieran que ignorar dos caracteres en un valor, se podrían usar dos '_' como en "Dept__outh".

Para tratar '%' y '_' como caracteres literales y no como comodines se debe especificar un carácter de escape '\' delante del '%' o '_'. Por ejemplo, si el nombre de propiedad era "%Descuento", para usarlo en una consulta, sería necesario especificar "\%Descuento". Si se va a utilizar el carácter '\' como carácter literal, se debe utilizar otro carácter de escape '\' como en "Orders\\Customer". Si se encuentra un carácter '\' individual sin que le siga un carácter '%', '_' o '\' a continuación, se generará una IllegalArgumentException.

Los caracteres comodín sólo se pueden utilizar en el operador de la izquierda (valor de propiedad). Los caracteres comodín no se pueden utilizar en nombres de propiedades.

Durante las búsquedas de un valor de propiedad específico o de una búsqueda de valores que no coincidan con una propiedad, la ausencia de una propiedad provoca que el artefacto se ignore al considerar la búsqueda. Por ejemplo, si existen tres grupos de normas (A, B y C) y sólo dos (A y B) tienen una propiedad llamada "Departamento" con valores distintos ("Contabilidad" y "Envíos" respectivamente) una búsqueda de todos los grupos de normas empresariales que no tienen una propiedad "Departamento" que es igual a "Contabilidad" sólo devolverá el grupo de normas empresariales que tiene la propiedad "Departamento" definida pero no es igual a "Contabilidad" (grupo de normas empresariales B). El grupo de normas empresariales (C) que no tiene la propiedad "Departamento", no se devolverá, ya que no tiene definida la propiedad de ninguna manera.

Cuando se utilizan propiedades para buscar, existen dos propiedades especiales llamadas *IBMSystemName* e *IBMSystemTargetNameSpace* que se pueden utilizar para buscar según el nombre y el espacio de nombres de un artefacto. Estos valores también se pueden recuperar con los métodos `getName` y `getTargetNameSpace`.

La clase admite los métodos siguientes para consultar:

```
List getBRGsByTNS (string tNSName, Operator op, int skip, int threshold)
List getBRGByName(string Name, Operator op, int skip, int threshold)
List getBRGsByTNSAndName (string tNSName, Operator, tNSOp, string
 name, Operator nameOp, int skip, int threshold)
List getBRGsBySingleProperty (string propertyName, string propertyValue,
 Operator op, int skip, int threshold)
List getBRGsByProperties (QueryNode queryTree, int skip, int threshold)
```

Los parámetros 'skip' y 'threshold' ofrecen al usuario la posibilidad de captar una lista de resultados parciales de hasta el umbral especificado. Un valor de cero para estos dos parámetros devolverá la lista de resultados completa. El cursor no se mantiene en el conjunto de resultados desde una llamada de consulta. Si se utiliza un valor omitido, se podrían haber realizado incorporaciones o supresiones al conjunto de resultados y que una solicitud posterior devuelva grupos de normas empresariales que estuvieran en conjunto de resultados anterior.

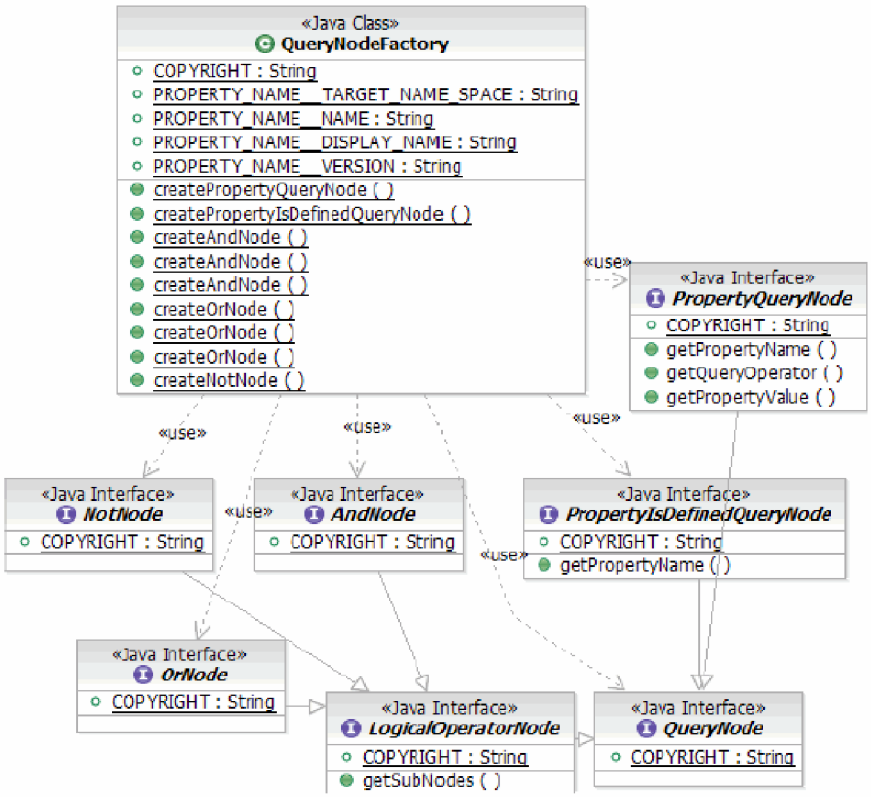


Figura 70. Diagrama de clase para QueryNodeFactory y clases relacionadas

Los nodos del árbol permiten al usuario especificar una expresión de búsqueda utilizando operadores booleanos, comodines (% y escape) y la pareja propiedad/valor. El operador sólo es válido para los valores, el operador para la propiedad siempre es igual a (==).

Publicación

La publicación de los cambios en las normas empresariales se realiza a nivel del componente del grupo de normas empresariales. El usuario puede publicar 1...n componentes del grupo de normas empresariales. Antes de que se realice una operación de publicación, se realiza una acción de validación en el grupo de normas empresariales y en los distintos objetos contenidos en el grupo de normas empresariales (tabla de selección de operación, conjuntos de normas, tabla de decisiones, etc). Todas las peticiones de publicación se producirán dentro de una transacción individual y si se encuentran excepciones durante la validación o la publicación de la base de datos, la transacción se retrotrae y no se publica en el repositorio ningún cambio para ningún grupo de normas empresariales. Esto permite que los cambios que dependen entre ellos dentro de un componente individual (por ejemplo, una tabla de selección de operaciones y un conjunto de normas) o dependencias entre componentes se produzcan dentro de una operación atómica.

En el momento de la publicación, se realizará una comprobación para garantizar que los elementos que se van a publicar no han sido modificados por otra transacción. Para reducir las posibilidades de un conflicto, el método de publicación dará al usuario la posibilidad de elegir publicar todos los artefactos (se hayan cambiado o no) o sólo aquellos que se cambiaron en el grupo de normas

empresariales. El comportamiento por omisión será publicar todos los artefactos. Si la opción está establecida para publicar todos los artefactos y durante ese tiempo otra transacción cambiara los artefactos, se generaría una `ChangeConflictException`. Especificar publicar sólo los artefactos que han cambiado reducirá las posibilidad de conflictos. Publicar sólo los artefactos que se cambiaron podría provocar que dos usuarios envíen cambios al repositorio para dos artefactos distintos de un grupo de normas empresariales (por ejemplo, dos conjuntos de normas) lo que podría introducir cambios incompatibles en el grupo de normas empresariales. Debido a esta situación potencial, esta opción se debe utilizar con precaución.

Manejo de excepciones

Se pueden producir excepciones cuando se llama a la validación en un artefacto o cuando se publica un artefacto. Cuando se produce un error de validación, se genera `ValidationException` con una lista de problemas. Si existe un problema durante la publicación debido a que otra transacción publica los mismos artefactos, se genera una `ChangeConflictException`. En cualquier momento en que se detecta que otra transacción está cambiando un artefacto, se genera la excepción `ChangeConflictException`.

Existe una `SystemPropertyNotChangeableException` que se genera cuando se intenta cambiar una propiedad que duplica un nombre de propiedad del sistema. Las propiedades del sistema no se pueden cambiar.

Existe una `ChangesNotAllowedException` que se genera cuando se intenta una operación de establecer en un artefacto que se está publicando.

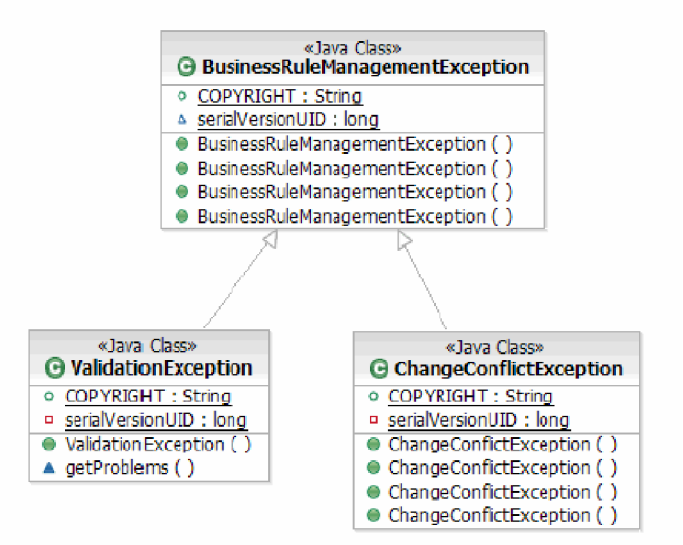


Figura 71. Diagrama de clase para BusinessRuleManagementException y clases relacionadas

Problemas de grupos de normas empresariales

Problemas que se pueden producir cuando se valida un grupo de normas empresariales o al intentar publicarlo cuando una parte del mismo no es válida.

Tabla 23. Problemas de grupos de normas empresariales

Excepción	Descripción
ProblemBusRuleNotInAvailTargetList	Problema que se produce cuando se especifica una norma como norma empresarial por omisión para una tabla de selección de operación, pero el artefacto de tabla no se encuentra en la lista de destinos disponibles para esa operación. Para evitar este problema, debe especificarse una norma empresarial válida de la lista de destinos disponibles.
ProblemDuplicatePropertyName	Este problema se produce cuando se intenta publicar una propiedad que es un duplicado de una propiedad del sistema o de una definida por el usuario en un grupo de normas empresariales. Para evitar este problema se debe utilizar un nombre de propiedad exclusivo.
ProblemOperationContainsNoTargets	Problema que se produce cuando una operación no tiene establecido un destino de norma por omisión o ningún destino de norma planificado. Para evitar este problema, la operación debería establecerse con al menos un destino de norma, ya sea por omisión o en una hora planificada.
ProblemOverlappingRanges	Problema que se produce cuando un registro de selección de operación tiene una fecha inicial o final que se solapa con el rango de fecha inicial y final de otro registro de selección de operación. Este solapamiento en rangos de fechas impide que el tiempo de ejecución de la norma empresarial encuentre el destino de norma correcto que invocar. Para evitar este problema, se deben comprobar la fecha inicial o la fecha final de los otros registros de selección de un operación para garantizar que no existe un solapamiento.
ProblemStartDateAfterEndDate	Este problema se produce cuando la fecha inicial de un registro de selección de operación es posterior a la fecha final de ese registro de selección. Este problema se puede producir en cualquier registro de selección de operación excepto el por omisión, que no tiene fecha inicial ni final. Para evitar este problema, especifique una fecha inicial después de la final de un registro de selección de operación.
ProblemTargetBusRuleNotSet	Problema que se produce cuando un registro de selección de operación tiene especificada una norma que no está en la lista de normas de destino disponibles. Para evitar este problema, debe especificarse una norma de la lista de destinos disponibles.
ProblemTNSAndNameAlreadyInUse	Problema que se produce cuando se crea una norma empresarial nueva con un espacio de nombres destino y un nombre que ya utiliza un conjunto de normas o una tabla de decisiones. Se realiza una comprobación en todos los conjuntos de normas y tablas de decisiones asociados al grupo de normas empresariales actual, además de cualquier artefacto de norma almacenado en el depósito. Para evitar este problema, se debe utilizar un espacio de nombres destino o un nombre distintos.
ProblemWrongOperationForOpSelectionRecord	Problema que se produce cuando se añade un registro de selección de operación nuevo a una lista de registros de selección de operación y la operación del nuevo registro no coincide con la de los registros de la lista. Para evitar este problema se debe crear una operación nueva utilizando el método `newOperationSelectionRecord` en el objeto de lista del registro de selección de operación correcto.

Problemas de conjuntos de normas y tablas de decisiones

Tabla 24. Problemas de conjuntos de normas y tablas de decisiones

Excepción	Descripción
ProblemInvalidBooleanValue	Problema que se produce cuando un parámetro de una plantilla de normas de un conjunto de normas o un valor de acción o de condición de una tabla de decisiones reciben un valor distinto a "true" o "false" para un parámetro de tipo Boolean. Ejemplos de valores de parámetro incorrectos serían "T" o "F". Para evitar este problema, utilice los valores "true" o "false" cuando trabaje con un parámetro de tipo Boolean.
ProblemParmNotDefinedInTemplate	Problema que se produce cuando se especifica un valor para un parámetro de plantilla y el parámetro no está definido en la lista de parámetros válidos para la plantilla. Los parámetros se deben comprobar antes de establecerlos en la plantilla. Puede ocurrir en las plantillas RuleTemplate, TreeActionValueTemplate o TreeConditionValueTemplate.
ProblemParmValueListContainsUnexpected Value	Problema que se produce cuando se pasan parámetros válidos con una plantilla, pero existen demasiados parámetros. Debe reducirse el número de parámetros. Puede ocurrir en las plantillas RuleTemplate, TreeActionValueTemplate o TreeConditionValueTemplate.
ProblemRuleBlockContainsNoRules	Este problema se produce cuando se eliminan todas las normas de un bloque de un conjunto de normas y se intenta validar o publicar el conjunto de normas. Un bloque de normas de un conjunto de normas debe tener al menos una norma.
ProblemTemplateNotAssociatedWithRuleSet	Problema que se produce cuando se intenta añadir una norma a un conjunto de normas y ésta se creó con una plantilla que no está definida con ese conjunto de normas. Para evitar este problema, cuando se cree una norma nueva, debe usarse una plantilla que se haya definido en el conjunto de normas.
ProblemRuleNameAlreadyInUse	Problema que se produce cuando se intenta añadir una norma a un bloque de un conjunto de normas que tiene el mismo nombre que una existente en el bloque de normas. Para evitar este problema, deben comprobarse los nombres de las normas antes de añadir una norma nueva.
ProblemTemplateParameterNotSpecified	Este problema se produce cuando no se incluye un parámetro al actualizar una plantilla para una norma en un conjunto de normas o en un valor de acción o condición de una tabla de decisiones. Para evitar este problema, deben especificarse todos los parámetros para una plantilla.
ProblemTypeConversionError	Este problema se produce cuando no se puede convertir un parámetro para una plantilla al tipo apropiado. Todos los parámetros se consideran como objetos de series y después se convierten al tipo del parámetro (boolean, byte, short, int, long, float y double). Este error se produce cuando el valor de parámetro de serie no se puede convertir al tipo especificado para este parámetro. Para evitar este problema, se debe especificar una serie que se pueda convertir en el tipo de parámetro (boolean, byte, short, int, long, float y double).

Tabla 24. Problemas de conjuntos de normas y tablas de decisiones (continuación)

Excepción	Descripción
ProblemValueViolatesParmConstraints	Este problema se produce cuando un parámetro no está incluido en la enumeración o el rango de valores que se han definido dentro de la plantilla para ese parámetro. Este problema se puede producir en parámetros restringidos con enumeraciones o rangos en plantillas de normas de un conjunto de normas o en plantillas de valor de acción o de condición de una tabla de decisiones. Para evitar este problema, debe usarse un valor que esté incluido en la enumeración.
ProblemInvalidActionValueTemplate	Problema que se produce cuando se intenta establecer una instancia de plantilla en la definición de valor de una acción de árbol, pero la plantilla correspondiente no está disponible para esa acción de árbol. Para evitar este problema, utilice la plantilla correcta para crear una definición de valor en una acción de árbol.
ProblemInvalidConditionValueTemplate	Problema que se produce cuando se intenta establecer una instancia de plantilla en la definición de condición de un límite de caso, pero la plantilla correspondiente no está disponible para ese límite de caso. Para evitar este problema, utilice la plantilla correcta para crear una definición definición de condición en un límite de caso.
ProblemTreeActionIsNull	Este problema se produce cuando se crea un valor de condición nuevo y no se ha establecido una acción con una instancia de plantilla. Cree una instancia de plantilla nueva utilizando una plantilla de ActionNode, y establézcala en la lista de `TreeActions`.

Autorización

Las clases no admiten ningún nivel de autorización. Depende de la acplicación cliente utilizar las clases para añadir su propia forma de autorización.

Ejemplos

Se incluyen varios ejemplos que muestran cómo se pueden usar las distintas clases para recuperar grupos de normas empresariales y para realizar modificaciones en los conjunto de normas y en las tablas de decisiones. Los ejemplos se incluyen en un archivo de intercambio de proyectos (ZIP) que se puede importar en WebSphere Integration Developer, donde podrá examinarse y reutilizarse.

El intercambio de proyectos contiene varios proyectos.

- **BRMgmtExamples** – Proyecto de módulo con artefactos de normas empresariales que se utilizan en los distintos ejemplos.
- **BRMgmt** – Proyecto Java con los ejemplos situados en el paquete com.ibm.websphere.sample.brules.mgmt.
- **BRMgmtDriverWeb** – Proyecto web con interface para ejecutar los ejemplos.

También se incluyen ejemplos como archivo EAR (`BRMgmtExamples.ear`) que se puede emitir una vez cuando se ha instalado en WebSphere Process Server. Se incluye una interfaz web con los ejemplos. La interfaz web se ha diseñado para que sea sencilla ya que los ejemplos se centran en el uso de las clases para recuperar artefactos, realizar modificaciones y publicar cambios. No está pensada para ser una interfaz web de funcionamiento sofisticado. Sin embargo, las clases se pueden usar fácilmente para construir interfaces web robustas o usarse en otras aplicaciones Java centradas en modificar las normas empresariales.

Nota: Puede descargar el archivo EAR y el intercambio de proyecto de ejemplo de Business Rule Management Programming Guide for WebSphere Process Server V6.1.

La aplicación de ejemplo se puede instalar en WebSphere Process Server v6.1 y se puede acceder a la página de índice en:

`http://<nombre_sistema_ppral>:<puerto>/BRMgmtDriverWeb/`

Por ejemplo, `http://localhost:9080/BRMgmtDriverWeb/`

A medida que se emitan los ejemplos, se realizarán cambios en los artefactos de las normas. Si se emiten todos los ejemplos, será necesario reinstalar la aplicación para ver otra vez los mismos resultados para todos los ejemplos.

Los ejemplos se explican con detalle con ejemplos de código completos además del resultado tal como se verá en un navegador web.

Se han creado varias clases adicionales para realizar operaciones comunes y ayudar en la visualización de información dentro de la aplicación web de ejemplo. Consulte el apéndice para ver más información sobre las clases `Formatter` y `RuleArtifactUtility`.

Para comprender totalmente estos ejemplos, le resultará útil ver un estudio de los distintos artefactos que hay en WebSphere Integration Developer.

Ejemplo 1: recuperar e imprimir todos los grupos de normas empresariales

Este ejemplo recuperará todos los grupos de normas empresariales e imprimirá los atributos, las propiedades y las operaciones de cada grupo de normas empresariales.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;
```

Para las clases de gestión de normas empresariales, asegúrese de usar las clases del paquete com.ibm.wbiserver.brules.mgmt y no las de com.ibm.wbiserver.brules ni las de otros paquetes. Estos otros paquetes son para clases internas de IBM.

```
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import
com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;

public class Example1 {
static Formatter out = new Formatter();
static public String executeExample1()
{
try
{
out.clear();
```

La clase BusinessRuleManager es la principal para recuperar grupos de normas empresariales y para publicar cambios en grupos de normas empresariales. Esto incluye trabajar y modificar cualquier artefacto de normas como los conjuntos de normas y las tablas de decisiones. Existen varios métodos en la clase

BusinessRuleManager que simplifican la recuperación de grupos de normas empresariales específicos por nombre y espacio de nombres y propiedades.

```
// Recuperar todos los grupos de normas empresariales
List<BusinessRuleGroup> brgList = BusinessRuleManager
  .getBusinessRuleGroups(0, 0);

Iterator<BusinessRuleGroup> iterator = brgList.iterator();

BusinessRuleGroup brg = null;
// Iterar a través de la lista de grupos de normas empresariales
while (iterator.hasNext())
{
  brg = iterator.next();
  // Emitir atributos para cada grupo de normas empresariales
  out.println1nBold("Grupo de normas empresariales");
```

Se pueden recuperar y mostrar los atributos básicos del grupo de normas empresariales.

```
out.println("Nombre: " + brg.getName());
out.println("Espacio de nombres: " +
brg.getTargetNameSpace());
out.println("Nombre de visualización: " +
brg.getDisplayName());
out.println("Descripción: " + brg.getDescription());
out.println("Huso horario de presentación: "
     + brg.getPresentationTimezone());
out.println("Fecha de guardado: " + brg.getSaveDate());
```

También se pueden recuperar y modificar las propiedades del grupo de normas empresariales.

```
PropertyList propList = brg.getProperties();

Iterator<Property> propIterator =
propList.iterator();
Property prop = null;
// Emitir nombres y valores de propiedad
while (propIterator.hasNext())
{
prop = propIterator.next();
out.println("Nombre de propiedad: " +
prop.getName());
out.println("Valor de propiedad: " +
prop.getValue());
}
```

Las operaciones para el grupo de normas empresariales también están disponibles y son la manera de recuperar los artefactos de normas empresariales como conjuntos de normas y tablas de decisiones.

```
List<Operation> opList = brg.getOperations();

Iteration<Operation> opIterator = opList.iterator();
Operation op = null;
// Emitir operaciones para el grupo de normas empresariales
while (opIterator.hasNext())
{
op = opIterator.next();
out.println("Operación: " + op.getName());
}
out.println("");}
} catch (BusinessRuleManagementException e)
{
e.printStackTrace();
out.println(e.getMessage());
```

```
}
return out.toString();
}
}
```

Salida en el navegador web para el ejemplo 1.

```
Ejecutando
ejemplo 1

Grupo de normas empresariales
Nombre: ApprovalValues
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: ApprovalValues
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: Department
Valor de propiedad: Accounting
Nombre de propiedad: RuleType
Valor de propiedad: regulatory
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: ApprovalValues
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: ApprovalValues
Operación: getApprover

Grupo de normas empresariales
Nombre: ConfigurationValues
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: ConfigurationValues
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: Department
Valor de propiedad: General
Nombre de propiedad: RuleType
Valor de propiedad: mensajes
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: ConfigurationValues
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: ConfigurationValues
Operación: getMessages

Grupo de normas empresariales
Nombre: DiscountRules
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: DiscountRules
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: Department
Valor de propiedad: Accounting
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: RuleType
Valor de propiedad: monetary
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
```

```
Nombre de propiedad: IBMSystemName
Valor de propiedad: DiscountRules
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: DiscountRules
Operación: calculateOrderDiscount
Operación: calculateShippingDiscount
```

Ejemplo 2: recuperar e imprimir grupos de normas empresariales, conjuntos de normas y tablas de decisiones

Además de la función del ejemplo 1, este ejemplo imprimirá la tabla de selección para cada operación y después el destino por omisión de la norma empresarial (conjunto de normas o tabla de decisiones) y las otras normas empresariales planificadas para la operación. Imprime los conjuntos de normas y las tablas de decisiones.

La mayor parte del ejemplo es el mismo, pero se incluye para proveer la información completa.

```
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecord;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecordList;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
public class Example2
{
 status Formatter out = new Formatter();
 static public String executeExample2()
 {
  try
  {
    out.clear();
```

En este ejemplo se recupera un grupo de normas empresariales específico por nombre.

```
  // Recuperar todos los grupos de normas empresariales
  List<BusinessRuleGroup> brgList = BusinessRuleManager
    .getBRGsByName("DiscountRules",
    QueryOperator.EQUAL, 0, 0);

  Iterator<BusinessRuleGroup> iterator = brgList.iterator();

  BusinessRuleGroup brg = null;
  // Iterar a través de la lista de grupos de normas empresariales
  while (iterator.hasNext())
   {
   brg = iterator.next();
   // Emitir atributos para cada grupo de normas empresariales
   out.println1nBold("Grupo de normas empresariales");
   out.println("Nombre: " + brg.getName());
   out.println("Espacio de nombres: " +
    brg.getTargetNameSpace());
   out.println("Nombre de visualización: " +
    brg.getDisplayName());
   out.println("Descripción: " + brg.getDescription());
   out.println("Huso horario de presentación: "
    + brg.getPresentationTimezone());
```

```
out.println("Fecha de guardado: " + brg.getSaveDate());

PropertyList propList = brg.getProperties();

Iterator<Property> propIterator =
propList.iterator();
Property prop = null;
// Emitir nombres y valores de propiedad
while (propIterator.hasNext())
{
  prop = propIterator.next();
  out.println("Nombre de propiedad: " +
  prop.getName());
  out.println("Valor de propiedad: " +
  prop.getValue());
}
```

Para cada operación, hay una tabla de selección que tiene una lista de los distintos artefactos de normas y la planificación de cuándo están activos. Se puede especificar una norma empresarial por omisión para cada operación. No es necesario especificar una norma empresarial por omisión ni que haya una norma empresarial planificada, aunque debe haber al menos una norma empresarial por omisión o planificada. Debido a este soporte, es mejor comprobar si hay un `nulo` antes de utilizar la norma empresarial por omisión o comprobar el tamaño de `OperationSelectionRecordList`.

```
List<Operation> opList = brg.getOperations();

Iterator<Operation> opIterator = opList.iterator();
Operation op = null;
out.println("");
out.printlnBold("Operaciones");
// Emitir operaciones para el grupo de normas empresariales
while (opIterator.hasNext())
 {
   op = opIterator.next();
   out.printBold("Operación: ");
   out.println(op.getName());

   // Recuperar la norma empresarial por omisión para la operación
   BusinessRule defaultRule =
   op.getDefaultBusinessRule();
   // Si se encuentra la norma por omisión, imprimir la norma empresarial
   // utilizando el método apropiado para el tipo de norma
   if (defaultRule != null)
   {
     out.printlnBold("Destino por omisión:");
```

La norma empresarial por omisión es del tipo `RuleSet` o `DecisionTable` y se puede enviar al tipo correcto para procesar el artefacto de norma.

```
     if (defaultRule instanceof RuleSet)
        out.println(RuleArtifactUtility.
        intRuleSet(defaultRule));
     else
        out.print(RuleArtifactUtility.
        tDecisionTable(defaultRule));
   }
   OperationSelectionRecordList
   opSelectionRecordList = op
     .getOperationSelectionRecordList()
     ;

   Iterator<OperationSelectionRecord>
```

```
      opSelRecordIterator = opSelectionRecordList
        .iterator();
      OperationSelectionRecord record = null;
```

OperationSelectionRecord está compuesto por el artefacto de norma y la planificación en que el artefacto de norma estará activo.

```
      while (opSelRecordIterator.hasNext())
      {
   out.printlnBold("Destino planificado:");
        record = opSelRecordIterator.next();

        out.println("Fecha inicial: " +
        record.getStartDate()
          + " - Fecha final: " +
          record.getEndDate());
        BusinessRule ruleArtifact = record
          .getBusinessRuleTarget();

        if (ruleArtifact instanceof RuleSet)
          out.println(RuleArtifactUtility.pr
          intRuleSet(ruleArtifact));
        else
          out.print(RuleArtifactUtility.prin
          tDecisionTable(ruleArtifact));
       }
    }
  }
  out.println("");
 } catch (BusinessRuleManagementException e)
 {
 e.printStackTrace();
 out.println(e.getMessage());
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 2.

```
Grupo de normas empresariales
Nombre: DiscountRules
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: DiscountRules
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: Department
Valor de propiedad: Accounting
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: RuleType
Valor de propiedad: monetary
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: DiscountRules
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: DiscountRules

Operaciones
Operación: calculateOrderDiscount
Destino por omisión:
Conjunto de normas
Nombre: calculateOrderDiscount
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
```

Norma: CopyOrder
Nombre de visualización: CopyOrder
Descripción: nulo
Presentación de usuario expandida: nulo
Presentación de usuario: nulo
Norma: FreeGiftInitialization
Nombre de visualización: FreeGiftInitialization
Descripción: nulo
Presentación de usuario expandida: ID de producto para obsequio = 5001AE80
Cantidad = 1 Coste = 0.0 Descripción = Obsequio para pedido con descuento
Presentación de usuario: ID de producto para obsequio = {0}
Cantidad = {1} Coste = {2}
Descripción = {3}
Nombre de parámetro: param0
Valor de parámetro: 5001AE80
Nombre de parámetro: param1
Valor de parámetro: 1
Nombre de parámetro: param2
Valor de parámetro: 0.0
Nombre de parámetro: param3
Valor de parámetro: Obsequio para pedido con descuento
Norma: Rule1
Nombre de visualización: Rule1
Descripción: nulo
Presentación de usuario expandida: si el cliente tiene el estado "oro", aplicar
un descuento de 20,0 e incluir un obsequio
Presentación de usuario: si el cliente tiene el estado "{0}", aplicar un
descuento de {1} e incluir un obsequio
Nombre de parámetro: param0
Valor de parámetro: gold
Nombre de parámetro: param1
Valor de parámetro: 20.0
Norma: Rule2
Nombre de visualización: Rule2
Descripción: nulo
Presentación de usuario expandida: si customer.status == silver,
ofrecer un descuento de 15,0
Presentación de usuario: si customer.status == {0}, ofrecer un descuento de {1}
Nombre de parámetro: param0
Valor de parámetro: silver
Nombre de parámetro: param1
Valor de parámetro: 15.0
Norma: Rule3
Nombre de visualización: Rule3
Descripción: plantilla para clientes que no tienen status "gold"
Presentación de usuario expandida: si customer.status == bronze,
ofrecer un descuento de 10,0
Presentación de usuario: si customer.status == {0}, ofrecer un descuento de {1}
Nombre de parámetro: param0
Valor de parámetro: bronze
Nombre de parámetro: param1
Valor de parámetro: 10.0

Operación: calculateShippingDiscount
Destino por omisión:
Tabla de decisiones
Nombre: calculateShippingDiscount
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules

Norma de inicialización: Rule1
Nombre de visualización: Rule1
Descripción: nulo
Presentación de usuario expandida: nulo
Presentación de usuario: nulo

Ejemplo 3: recuperar grupos de normas empresariales por propiedades múltiples con el operador AND

Este ejemplo también es parecido al 1, pero sólo recuperará los grupos de normas empresariales que tengan una propiedad llamada Department y un valor de "accounting", y una propiedad llamada RuleType y un valor de "regulatory".

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.AndNode;
import com.ibm.wbiserver.brules.mgmt.query.PropertyQueryNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryNodeFactory;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example3
{
static Formatter out = new Formatter();
static public String executeExample3()
{
  try
  {
    out.clear();
```

Las consultas de grupos de normas empresariales están compuestas por nodos de consulta que siguen una estructura de árbol. Cada nodo de consulta tiene un término en el lado izquierdo y un término en el lado derecho y una condición. Cada término y término de la derecha puede ser otro nodo de consulta. En este ejemplo, el grupo de normas empresariales se recupera por la combinación de dos valores de propiedad.

```
   // Recuperar grupos de normas empresariales según dos codiciones
   // Crear PropertyQueryNodes para cada condición
   PropertyQueryNode propertyNode1 = QueryNodeFactory
     .createPropertyQueryNode("Department",
     QueryOperator.EQUAL,"Accounting");
   PropertyQueryNode propertyNode2 = QueryNodeFactory
     .createPropertyQueryNode("RuleType", QueryOperator.EQUAL,
     "regulatory");
   // Combinar los dos PropertyQueryNodes con un nodo AND
   AndNode andNode =
   QueryNodeFactory.createAndNode(propertyNode1, propertyNode2);

   // Usar andNode en la búsqueda de grupos de normas empresariales
   List<BusinessRuleGroup> brgList = BusinessRuleManager
     .getBRGsByProperties(andNode, 0, 0);

   Iterator<BusinessRuleGroup> iterator = brgList.iterator();

   BusinessRuleGroup brg = null;
   // Iterar a través de la lista de grupos de normas empresariales
   while (iterator.hasNext())
   {
     brg = iterator.next();
     // Emitir atributos para cada grupo de normas empresariales
     out.println1nBold("Grupo de normas empresariales");
     out.println("Nombre: " + brg.getName());
     out.println("Espacio de nombres: " +
     brg.getTargetNameSpace());
     out.println("Nombre de visualización: " + brg.getDisplayName());
```

```
    out.println("Descripción: " + brg.getDescription());
    out.println("Huso horario de presentación: "
    + brg.getPresentationTimezone());
    out.println("Fecha de guardado: " + brg.getSaveDate());

    PropertyList propList = brg.getProperties();

    Iterator<Property> propIterator =
    propList.iterator();
    Property prop = null;
    // Emitir nombres y valores de propiedad
    while (propIterator.hasNext())
    {
      prop = propIterator.next();
      out.println("\t Nombre de propiedad: " +
      prop.getName());
      out.println("\t Valor de propiedad: " +
      prop.getValue());
    }
  }
 } catch (BusinessRuleManagementException e)
 {
 e.printStackTrace();
 out.println(e.getMessage());
 }
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 3.

Ejecución del ejemplo 3

```
Grupo de normas empresariales
Nombre: ApprovalValues
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: ApprovalValues
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: Department
Valor de propiedad: Accounting
Nombre de propiedad: RuleType
Valor de propiedad: regulatory
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: ApprovalValues
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: ApprovalValues
```

Ejemplo 4: recuperar grupos de normas empresariales por propiedades múltiples con el operador OR

Este ejemplo es parecido al 3, pero sólo recuperará los grupos de normas empresariales que tengan una propiedad llamada Department y un valor de “accounting”, o una propiedad llamada RuleType y un valor de “monetary”.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;
```

```
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.OrNode;
import com.ibm.wbiserver.brules.mgmt.query.PropertyQueryNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryNodeFactory;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example4
{
static Formatter out = new Formatter();
static public String executeExample4()
{
  try
  {
    out.clear();
```

Distintas propiedades componen la consulta y devuelven grupos de normas empresariales distintos.

```
    // Recuperar grupos de normas empresariales según dos codiciones
    // Crear PropertyQueryNodes para cada condición
    PropertyQueryNode propertyNode1 = QueryNodeFactory
      .createPropertyQueryNode("Department",
      QueryOperator.EQUAL,"Accounting");
    PropertyQueryNode propertyNode2 = QueryNodeFactory
      .createPropertyQueryNode("RuleType",
      QueryOperator.EQUAL,"monetary");
    // Combinar los dos PropertyQueryNodes con un nodo OR
    OrNode orNode =
    QueryNodeFactory.createOrNode(propertyNode1,
      propertyNode2);
    // Usar orNode en la búsqueda de grupos de normas empresariales
    List<BusinessRuleGroup> brgList = BusinessRuleManager
      .getBRGsByProperties(orNode, 0, 0);

    Iterator<BusinessRuleGroup> iterator = brgList.iterator();

    BusinessRuleGroup brg = null;
    // Iterar a través de la lista de grupos de normas empresariales
    while (iterator.hasNext())
    {
      brg = iterator.next();
      // Emitir atributos para cada grupo de normas empresariales
      out.println1nBold("Grupo de normas empresariales");
      out.println("Nombre: " + brg.getName());
      out.println("Espacio de nombres: " +
      brg.getTargetNameSpace());
      out.println("Nombre de visualización: " + brg.getDisplayName());
      out.println("Descripción: " + brg.getDescription());
      out.println("Huso horario de presentación: "
        + brg.getPresentationTimezone());
      out.println("Fecha de guardado: " + brg.getSaveDate());

      PropertyList propList = brg.getProperties();

      Iterator<Property> propIterator =
      propList.iterator();
      Property prop = null;
      // Emitir nombres y valores de propiedad
      while (propIterator.hasNext())
      {
        prop = propIterator.next();
        out.println("\t Nombre de propiedad: " +
```

```
        prop.getName());
        out.println("\t Valor de propiedad: " +
        prop.getValue());
      }
      out.println("");
    }
  } catch (BusinessRuleManagementException e)
 {
   e.printStackTrace();
   out.println(e.getMessage());
 }
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 4.

Ejecución del ejemplo 4

```
Grupo de normas empresariales
Nombre: ApprovalValues
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: ApprovalValues
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: Department
Valor de propiedad: Accounting
Nombre de propiedad: RuleType
Valor de propiedad: regulatory
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: ApprovalValues
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: ApprovalValues

Grupo de normas empresariales
Nombre: DiscountRules
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: DiscountRules
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: Department
Valor de propiedad: Accounting
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: RuleType
Valor de propiedad: monetary
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: DiscountRules
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: DiscountRules
```

Ejemplo 5: recuperar grupos de normas empresariales con una consulta compleja

Este ejemplo es una combinación de los ejemplos 3 y 4, y está pensado para mostrar cómo se pueden crear consultas más complejas. En este ejemplo, se lleva a

cabo una búsqueda con una consulta que combina 2 condiciones. La primera condición de consulta es recuperar aquellos grupos de normas empresariales que tienen una propiedad llamada Departamento y un valor de “General” o una propiedad llamada MissingProperty y un valor de “somevalue”. A continuación se combina esta condición de consulta con un operador AND y una condición en que la propiedad se llama RuleType y tiene un valor de “messages”.

Hay más ejemplos de consultas de grupos de normas empresariales en el apéndice.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Property;
import com.ibm.wbiserver.brules.mgmt.PropertyList;
import com.ibm.wbiserver.brules.mgmt.query.AndNode;
import com.ibm.wbiserver.brules.mgmt.query.OrNode;
import com.ibm.wbiserver.brules.mgmt.query.PropertyQueryNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryNodeFactory;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example5
{
static Formatter out = new Formatter();
static public String executeExample5()
{
  try
  {
    out.clear();

    // Recuperar grupos de normas empresariales según tres codiciones, donde
    // dos de las condiciones se combinan en un nodo OR
    // Crear PropertyQueryNodes para cada condición del nodo OR
    PropertyQueryNode propertyNode1 = QueryNodeFactory
      .createPropertyQueryNode("Department",
      QueryOperator.EQUAL,"General");
    PropertyQueryNode propertyNode2 = QueryNodeFactory
      .createPropertyQueryNode("MissingProperty",
    QueryOperator.EQUAL, "SomeValue");
    // Combinar los dos PropertyQueryNodes con un nodo OR
    OrNode orNode =
    QueryNodeFactory.createOrNode(propertyNode1, propertyNode2);
    // Crear el tercer PropertyQueryNode
      PropertyQueryNode propertyNode3 = QueryNodeFactory
      .createPropertyQueryNode("RuleType",
    QueryOperator.EQUAL,"messages");
```

La combinación de la izquierda se combina con la condición de la derecha con un nodo AND. AndNode es la raíz del árbol de consulta.

```
    // Combinar nodo OR con el tercer PropertyQueryNode
    AndNode AndNode andNode =
    QueryNodeFactory.createAndNode(propertyNode3, orNode);

    List<BusinessRuleGroup> brgList = BusinessRuleManager
      .getBRGsByProperties(andNode, 0, 0);

    Iterator<BusinessRuleGroup> iterator = brgList.iterator();

    BusinessRuleGroup brg = null;
    // Iterar a través de la lista de grupos de normas empresariales
    while (iterator.hasNext())
```

```
    {
      brg = iterator.next();
      // Emitir atributos para cada grupo de normas empresariales
      out.printlnBold("Grupo de normas empresariales");
      out.println("Nombre: " + brg.getName());
      out.println("Espacio de nombres: " +
        brg.getTargetNameSpace());
      out.println("Nombre de visualización: " + brg.getDisplayName());
      out.println("Descripción: " + brg.getDescription());
      out.println("Huso horario de presentación: "
        + brg.getPresentationTimezone());
      out.println("Fecha de guardado: " + brg.getSaveDate());
      PropertyList propList = brg.getProperties();

      Iterator<Property> propIterator =
      propList.iterator();
      Property prop = null;
      // Emitir nombres y valores de propiedad
      while (propIterator.hasNext())
      {
        prop = propIterator.next();
        out.println("\t Nombre de propiedad: " +
        prop.getName());
        out.println("\t Valor de propiedad: " +
        prop.getValue());
      }
    }
   }   catch (BusinessRuleManagementException e)
  {
  e.printStackTrace();
  out.println(e.getMessage());
  }
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 5.

```
Ejecución
del ejemplo 5

Grupo de normas empresariales
Nombre: ConfigurationValues
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de visualización: ConfigurationValues
Descripción: nulo
Huso horario de presentación: LOCAL
Fecha de guardado: Sun Jan 06 17:56:51 CST 2008
Nombre de propiedad: IBMSystemVersion
Valor de propiedad: 6.2.0
Nombre de propiedad: Department
Valor de propiedad: General
Nombre de propiedad: RuleType
Valor de propiedad: mensajes
Nombre de propiedad: IBMSystemTargetNameSpace
Valor de propiedad: http://BRSamples/com/ibm/websphere/sample/brules
Nombre de propiedad: IBMSystemName
Valor de propiedad: ConfigurationValues
Nombre de propiedad: IBMSystemDisplayName
Valor de propiedad: ConfigurationValues
```

Ejemplo 6: actualizar una propiedad de grupo de normas empresariales y publicarla

En este ejemplo, se actualizará una propiedad de un grupo de normas empresariales y después se publicará el grupo de normas empresariales.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.UserDefinedProperty;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example6
{
static Formatter out = new Formatter();

static public String executeExample6()
{
 try
 {
   out.clear();
   out.println1nBold("Conjunto de normas empresariales antes de la publicación:");
   // Recuperar grupos de normas empresariales según un solo valor de propiedad
   List<BusinessRuleGroup> brgList = BusinessRuleManager
     .getBRGsBySingleProperty("Department",
     QueryOperator.EQUAL,"General", 0, 0);

   if (brgList.size() > 0)
   {
     // Obtener el primer grupo de normas empresariales de la lista
     BusinessRuleGroup brg = brgList.get(0);
     // Recuperar la propiedad del grupo de normas empresariales
     UserDefinedProperty userDefinedProperty =
     (UserDefinedProperty) brg
       .getProperty("Department");

     out.println("Grupo de normas empresariales: " + brg.getName());
     out.println("Valor de propiedad de departamento: "
     + brg.getProperty("Department").getValue());
```

El método `getProperty` devuelve una propiedad por referencia y los cambios realizados en la propiedad se efectúan directamente en el grupo de normas empresariales.

```
     // Modificar el valor de propiedad en brg
     // Esto actualiza el valor de propiedad directamente en el objeto brg
     userDefinedProperty.setValue("GeneralConfig");
     // Utilizar la lista original o crear una lista nueva
     // de grupos de normas empresariales.
     List<BusinessRuleGroup> publishList = new
     ArrayList<BusinessRuleGroup>();
     // Añadir el grupo de normas empresariales cambiado a la lista
     publishList.add(brg);
```

La clase `BusinessRuleManager` se utiliza para publicar los cambios realizados en un grupo de normas empresariales. Para publicar el cambio, se pasa una lista al método de publicación `BusinessRuleManager` aunque sólo se esté publicando un elemento.

```
     // Publicar la lista con el grupo de normas empresariales actualizado
     BusinessRuleManager.publish(publishList, true);
```

```
     out.println("");

     // Recuperar el grupo de normas empresariales otra vez para verificar que
     // se publicaron los cambios
     out.println1nBold("Grupo de normas empresariales después de la publicación:");
     brgList = BusinessRuleManager
     .getBRGsBySingleProperty("Department",
     QueryOperator.EQUAL, "GeneralConfig", 0, 0);

     brg = brgList.get(0);

     out.println("Grupo de normas empresariales: " + brg.getName());
     // Mostrar el valor de propiedad para mostrar el cambio
     out.println("Valor de propiedad de departamento: "
     + brg.getProperty("Department").getValue());
   }
 } catch (BusinessRuleManagementException e)
 {
 e.printStackTrace();
 out.println(e.getMessage());
 }
return out.toString();
}
}
```

Ejemplo

Salida en el navegador web para el ejemplo 6.

```
Ejecución
del ejemplo 6

Grupo de normas empresariales antes de la publicación:
Grupo de normas empresariales: ConfigurationValues
Valor de propiedad de departamento: General

Grupo de normas empresariales después de la publicación:
Grupo de normas empresariales: ConfigurationValues
Valor de propiedad de departamento: GeneralConfig
```

Ejemplo 7: Actualizar propiedades en varios grupos de normas empresariales y publicarlas

En este ejemplo, se actualizar propiedades en varios grupos de normas empresariales antes de publicar.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.UserDefinedProperty;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example7
{
static Formatter out = new Formatter();

static public String executeExample7()
{
  try
  {
    out.clear();
    out.println1nBold("Conjunto de normas empresariales antes de la publicación:");
```

```
    // Recuperar grupos de normas empresariales según un solo valor de propiedad
    List<BusinessRuleGroup> brgList = BusinessRuleManager
      .getBRGsBySingleProperty("Department",
      QueryOperator.EQUAL, "Accounting", 0, 0);

    Iterator<BusinessRuleGroup> iterator = brgList.iterator();

    BusinessRuleGroup brg = null;

    // Utilizar la lista original o crear una lista nueva
    // de grupos de normas empresariales.
    List<BusinessRuleGroup> publishList = new
    ArrayList<BusinessRuleGroup>();

    // Iterar por todos los grupos de normas empresariales y
    // modificar la propiedad
    while (iterator.hasNext())
    {
      // Recuperar la propiedad del grupo de normas empresariales
      brg = iterator.next();

      out.println("Grupo de normas empresariales: " + brg.getName());

      // Recuperar la propiedad del grupo de normas empresariales
      UserDefinedProperty prop = (UserDefinedProperty) brg
        .getProperty("Department");
      out.println("Valor de propiedad de departamento: "
      +
      brg.getProperty("Department").getValue())
      ;

      // Modificar el valor de propiedad en brg
      // Esto actualiza el valor de propiedad directamente en el objeto brg
      prop.setValue("Finance");
```

Se añaden a la lista todos los grupos de normas empresariales cambiados.

```
      // Añadir el grupo de normas empresariales cambiado a la lista
      publishList.add(brg);
      }

      // Publicar la lista con el grupo de normas empresariales actualizado
      BusinessRuleManager.publish(publishList, true);

      out.println("");

      // Recuperar los grupos de normas empresariales otra vez para verificar que
      // se publicaron los cambios
    out.println1nBold("Grupo de normas empresariales después de la publicación:");
      brgList = BusinessRuleManager
        .getBRGsBySingleProperty("Department",
        QueryOperator.EQUAL,
        "Finance", 0, 0);
      iterator = brgList.iterator();

      while (iterator.hasNext())
      {
        brg = iterator.next();
        out.println("Grupo de normas empresariales: " +
        brg.getName());
        out.println("Valor de propiedad de departamento: "
         +
         brg.getProperty("Department").getVa
         lue());
      }
} catch (BusinessRuleManagementException e)
{
```

```
  e.printStackTrace();
  out.println(e.getMessage());
}
return out.toString();
}
}
```

Ejemplo

Salida en el navegador web para el ejemplo 7.

```
Ejecución
del ejemplo 7

Grupo de normas empresariales antes de la publicación:
Grupo de normas empresariales: ApprovalValues
Valor de propiedad de departamento: Accounting
Grupo de normas empresariales: DiscountRules
Valor de propiedad de departamento: Accounting

Grupo de normas empresariales después de la publicación:
Grupo de normas empresariales: ApprovalValues
Valor de propiedad de departamento: Finance
Grupo de normas empresariales: DiscountRules
Valor de propiedad de departamento: Finance
```

Ejemplo 8: Cambiar la norma empresarial por omisión de un grupo de normas empresariales

En este ejemplo, la norma empresarial por omisión se cambia por otra que forma porta de la lista de destinos disponibles para una operación específica.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example8
{
static Formatter out = new Formatter();

static public String executeExample8()
{
  try
  {
    out.clear();

  // Recuperar un grupo de normas empresariales por espacio de nombres destino
  // y nombre
  List<BusinessRuleGroup> brgList = BusinessRuleManager
    .getBRGsByTNSAndName(
      "http://BRSamples/com/ibm/websphere
      /sample/brules",
      QueryOperator.EQUAL,
      "DiscountRules",
      QueryOperator.EQUAL, 0, 0);

  if (brgList.size() > 0)
  {
    out.printlnBold("Conjunto de normas empresariales antes de la publicación:");
```

```
// Obtener el primer grupo de normas empresariales de la lista
// Debería ser el único grupo de normas empresariales de la lista, ya que
// la combinación de espacio de nombres destino y nombre es exclusiva
BusinessRuleGroup brg = brgList.get(0);

out.print("Grupo de normas empresariales: ");
out.println(brg.getName());

// Obtener la operación del grupo de normas empresariales cuya
// norma empresarial por omisión se actualizará
Operation op =
brg.getOperation("calculateShippingDiscount");
```

La norma empresarial por omisión se recupera antes de actualizarla con otra norma que forme parte de la lista de destinos disponibles para la operación. Los conjuntos de normas y las tablas de decisiones son específicos de operaciones y sólo aquellos artefactos de normas empresariales que están en una operación se pueden establecer como por omisión o para planificarse en otro momento en la operación.

```
// Recuperar la norma empresarial por omisión para la operación
BusinessRule defaultRule =
op.getDefaultBusinessRule();
out.print("Norma por omisión: ");
out.println(defaultRule.getName());

// Obtener la lista de normas empresariales disponibles para esta operación
List<BusinessRule> ruleList =
op.getAvailableTargets();

Iterator<BusinessRule> iterator =
ruleList.iterator();
BusinessRule rule = null;

// Buscar una norma empresarial que sea diferente de la
// norma empresarial
// por omisión
while (iterator.hasNext())
{
  rule = iterator.next();
  if
  (!defaultRule.getName().equals(rule.getName()))
  {
```

La norma empresarial por omisión se establece en el objeto de operación. Establecer la norma empresarial por omisión en un valor nulo eliminará la norma empresarial por omisión anterior de la operación, aunque se recomienda que todas las operaciones tengan una especificada.

```
    // Establecer la norma empresarial por omisión por
    // una distinta.
    // Este cambio se realiza en el objeto de la operación
    // directamente
    op.setDefaultBusinessRule(rule);
    break;
  }
}
// Utilizar la lista original o crear una lista nueva
// de grupos de normas empresariales.
List<BusinessRuleGroup> publishList = new
ArrayList<BusinessRuleGroup>();
// Añadir el grupo de normas empresariales cambiado a la lista
publishList.add(brg);
   // Publicar la lista con el grupo de normas empresariales actualizado
BusinessRuleManager.publish(publishList, true);
```

```
    out.println("");

    // Recuperar los grupos de normas empresariales otra vez para verificar que
    // se publicaron los cambios

    out.printlnBold("Grupo de normas empresariales después de la publicación:");
    brgList = BusinessRuleManager
    .getBRGsByTNSAndName(
      "http://BRSamples/com/ibm/websphere/sample/brules",
      QueryOperator.EQUAL, "DiscountRules",
      QueryOperator.EQUAL, 0, 0);

    brg = brgList.get(0);
    out.println("Grupo de normas empresariales: " + brg.getName());
    op = brg.getOperation("calculateShippingDiscount");

    // Recuperar la norma empresarial por omisión para la operación
    defaultRule = op.getDefaultBusinessRule();
    out.print("Norma por omisión: ");
    out.println(defaultRule.getName());
   }
 } catch (BusinessRuleManagementException e)
 {
   e.printStackTrace();
   out.println(e.getMessage());
 }
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 8.

```
Ejecución
del ejemplo 8

Grupo de normas empresariales antes de la publicación:
Grupo de normas empresariales: DiscountRules
Norma por omisión: calculateShippingDiscount

Grupo de normas empresariales después de la publicación:
Grupo de normas empresariales: DiscountRules
Norma por omisión: calculateShippingDiscountHoliday
```

Ejemplo 9: Planificar otra norma para una operación de un grupo de normas empresariales

En este ejemplo, se planifica una norma empresarial para que esté activa durante una hora a partir del momento de la publicación para una operación determinada.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecordList;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecord;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
```

```
public class Example9 {
static Formatter out = new Formatter();

static public String executeExample9()
{
 try
 {
   out.clear();

   // Recuperar un grupo de normas empresariales por espacio de nombres destino
   // y nombre
   List<BusinessRuleGroup> brgList = BusinessRuleManager
     .getBRGsByTNSAndName(
     "http://BRSamples/com/ibm/websphere
     /sample/brules",
     QueryOperator.EQUAL,
     "DiscountRules",
     QueryOperator.EQUAL, 0, 0);

   if (brgList.size() > 0)
   {
     out.println("");
     out.printlnBold("Conjunto de normas empresariales antes de la publicación:");
     // Obtener el primer grupo de normas empresariales de la lista
      // Debería ser el único grupo de normas empresariales de la
     list as
     // la combinación de espacio de nombres destino y nombre es exclusiva
     BusinessRuleGroup brg = brgList.get(0);

     // Obtener la operación del grupo de normas empresariales que
     // tendrá planificada una norma empresarial nueva
     Operation op =
     brg.getOperation("calculateShippingDiscount");

     printOperationSelectionRecord(op);
     // Obtener la lista de normas empresariales disponibles para esta operación
     List<BusinessRule> ruleList =
     op.getAvailableTargets();

     // Obtener la primera norma de la lista, ya que se planificará
     // para la operación
     BusinessRule rule = ruleList.get(0);

     // Obtener la lista de normas empresariales planificadas
     OperationSelectionRecordList opList = op
       .getOperationSelectionRecordList();
     // Crear una fecha de finalización en el futuro para la norma empresarial
     Date future = new Date();
     long futureTime = future.getTime() + 3600000;
```

En una norma planificada nueva, se pueden especificar fechas de inicio y de finalización junto con la norma. Si la fecha de inicio se establece en nulo, esto indica que la norma estará activa justo cuando se publique. Si se establece en nulo la fecha final, la norma no tendrá fecha de finalización. No se permite solapar planificaciones y esto se puede comprobar llamando al método de validación en la operación.

```
     // Crear la norma empresarial planificada nueva con la
     // fecha actual, lo que significa que esta norma estará activa
     // de manera inmediata en el intervalo comprendido entre su
     // publicación y la fecha futura.
     newOperationSelectionRecord(new Date(),
       new Date(futureTime), rule);
     // Añadir la norma empresarial planificada nueva a la lista de
     // normas empresariales
     opList.addOperationSelectionRecord(newRecord);
```

Valide la operación para garantizar que no existe ningún solapamiento.

```
     // Validar la lista para garantizar que no existe solapamiento
     List<Problem> problems = op.validate();
     if (problems.size() == 0)
     {
       // Utilizar la lista original o crear una lista nueva
       // de grupos de normas empresariales.
       List<BusinessRuleGroup> publishList = new
       ArrayList<BusinessRuleGroup>();
       // Añadir el grupo de normas empresariales cambiado a la lista
       publishList.add(brg);
    // Publicar la lista con el grupo de
       rule group
       BusinessRuleManager.publish(publishList, true);
       out.println("");

       // Recuperar los grupos de normas empresariales otra vez para verificar que
       // se publicaron los cambios
    out.println1nBold("Grupo de normas empresariales después de la publicación:");
       brgList =
       BusinessRuleManager.getBRGsByTNSAndName(
         "http://BRSamples/com/ibm/websphere
         /sample/brules",
         QueryOperator.EQUAL,
         "DiscountRules",
         QueryOperator.EQUAL, 0, 0);
         brg = brgList.get(0);

         op =
         brg.getOperation("calculateShippingDiscount");

         printOperationSelectionRecord(op);
       }
       // en caso contrario, manejar el error de validación
     }
 } catch (BusinessRuleManagementException e)
 {
 e.printStackTrace();
 out.println(e.getMessage());
 }
 return out.toString();
}
/*
Método para imprimir el registro de selección de operación de una operación. Los
start date and end date are printed as well as the name of the rule
artifact for the scheduled time.
*/
private static void printOperationSelectionRecord(Operation op)
{
 OperationSelectionRecordList opSelectionRecordList = op
   .getOperationSelectionRecordList();
 Iterator<OperationSelectionRecord> opSelRecordIterator =
 opSelectionRecordList
   .iterator();
 OperationSelectionRecord record = null;
 while (opSelRecordIterator.hasNext())
 {
   out.println1nBold("Destino planificado:");
   record = opSelRecordIterator.next();
   out.println("Fecha inicial: " + record.getStartDate()
   + " - Fecha final: " + record.getEndDate());
   BusinessRule ruleArtifact = record.getBusinessRuleTarget();
   out.println("Norma: " + ruleArtifact.getName());
   }
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 9.

```
Ejecución
del ejemplo 9

Grupo de normas empresariales antes de la publicación:
Destino planificado:
Fecha inicial: Thu Dec 01 00:00:00 CST 2005 - Fecha final: Sun Dec 25 00:00:00 CST 2005
Norma: calculateShippingDiscountHoliday

Grupo de normas empresariales después de la publicación:
Destino planificado:
Fecha inicial: Thu Dec 01 00:00:00 CST 2005 - Fecha final: Sun Dec 25 00:00:00 CST 2005
Norma: calculateShippingDiscountHoliday
Destino planificado:
Fecha inicial: Mon Jan 07 21:08:31 CST 2008 - Fecha final: Mon Jan 07 22:08:31 CST 2008
Norma: calculateShippingDiscount
```

Ejemplo 10: Modificar un valor de parámetro de una plantilla de un conjunto de normas

En este ejemplo, una instancia de norma definida con una plantilla se modifica cambiando un valor de parámetro y después publicando.

```
package com.ibm.websphere.sample.brules.mgmt;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;

public class Example10
{
static Formatter out = new Formatter();

static public String executeExample10()
{
  try
  {
    out.clear();

    // Recuperar un grupo de normas empresariales por espacio de nombres y
    nombre
    List<BusinessRuleGroup> brgList = BusinessRuleManager
      .getBRGsByTNSAndName(
        "http://BRSamples/com/ibm/websphere
        /sample/brules",
        QueryOperator.EQUAL,
        "ApprovalValues",
        QueryOperator.EQUAL, 0, 0);
    if (brgList.size() > 0)
    {
      // Obtener el primer grupo de normas empresariales de la lista
      // Debería ser el único grupo de normas empresariales de la
      lista, ya que la
      // combinación de espacio de nombres destino y nombre es
```

```
exclusiva
BusinessRuleGroup brg = brgList.get(0);
// Obtener la operación del grupo de normas empresariales que
// tiene la norma empresarial que se modificará, ya que
// las normas empresariales están asociadas a una operación
// específica
Operation op = brg.getOperation("getApprover");

// Obtener la norma empresarial en la operación que
se modificará
List<BusinessRule> ruleList =
op.getBusinessRulesByName(
  "getApprover", QueryOperator.EQUAL, 0,
  0);

if (ruleList.size() > 0)
{
  out.println("");
  out.printlnBold("Conjunto de normas antes de la publicación:");
  // Obtener la norma que modificar. Las normas son
  exclusivas respecto a
  // espacio de nombre destino y nombre, pero para este
  ejemplo
  // sólo hay una norma empresarial llamada
  "getApprover"
  RuleSet ruleSet = (RuleSet) ruleList.get(0);
  out.print(RuleArtifactUtility.printRuleSet(rule
  Set));
```

Todas las normas de un conjunto de normas están en un bloque de normas. Sólo se admite un bloque de normas y debe usarse el método `getFirstRuleBlock` para recuperar el bloque de normas.

```
      // Un conjunto de normas tiene todas
las normas definidas en un
      bloque de normas
      RuleBlock ruleBlock =
      ruleSet.getFirstRuleBlock();

      Iterator<RuleSetRule> ruleIterator =
      ruleBlock.iterator();

      // Iterar por las normas del bloque de normas
      para encontrar la
      // instancia de norma llamada "LargeOrderApprover"
      while (ruleIterator.hasNext())
      {
        RuleSetRule rule = ruleIterator.next();
```

Si una norma no está definida con una plantilla de normas, sólo tiene una presentación Web que se puede recuperar. No se pueden realizar actualizaciones a una norma que no esté definida con una plantilla. Si el nombre de la norma es desconocido, es mejor comprobar si ésta se ha definido con una plantilla.

```
      // La norma se
debe haber definido con una
      plantilla
      // para que se pueda cambiar. Comprobar
      si la norma actual
      // está basada en una plantilla.
      if (rule instanceof
      RuleSetTemplateInstanceRule)
      {
```

Utilice el objeto TemplateInstance para crear la norma.

```
        // Obtener la instancia de la plantilla de la norma
        RuleSetTemplateInstanceRule
        templateInstance =
        (RuleSetTemplateInstanceRule) rule;

        // Comprobar la instancia de la norma
        que coincide con
        // la norma que modificar
        if
        (templateInstance.getName().equals(
          "LargeOrderApprover"))
        {
```

Para la instancia de plantilla, sólo se pueden modificar valores de parámetros. Los parámetros se modifican recuperando `ParameterValue` y estableciéndolo a un valor apropiado. Debido a que se pasa `ParameterValue` por referencia, la actualización se realiza directamente en la norma, conjunto de normas y grupo de normas empresariales.

```
          // Obtener el parámetro de la
          instancia de norma
          ParameterValue parameter =
          templateInstance
            .getParameterValue("par
            am2");

          // Modificar el valor del
          parámetro
          parameter.setValue("superviso
          r");
          break;
        }
       }
    }
    // Utilizar la lista original o crear una lista nueva
    // de grupos de normas empresariales.
    List<BusinessRuleGroup> publishList = new
    ArrayList<BusinessRuleGroup>();

    // Añadir el grupo de normas empresariales cambiado a la lista
    publishList.add(brg);

       // Publicar la lista con el grupo de normas empresariales actualizado
     BusinessRuleManager.publish(publishList, true);

    out.println("");
    // Recuperar los grupos de normas empresariales otra vez para verificar que
    // se publicaron los cambios
    out.printlnBold("Conjunto de normas después de la publicación:");

    brgList = BusinessRuleManager
      .getBRGsByTNSAndName(
        "http://BRSamples/com/ibm/websphere/sample/brules",
        QueryOperator.EQUAL, "ApprovalValues",
        QueryOperator.EQUAL, 0, 0);

    brg = brgList.get(0);
    op = brg.getOperation("getApprover");
    ruleList = op.getBusinessRulesByName(
      "getApprover", QueryOperator.EQUAL, 0,0);

    ruleSet = (RuleSet) ruleList.get(0);
    out.print(RuleArtifactUtility.printRuleSet(ruleSet));
    }
  }
  } catch (BusinessRuleManagementException e)
  {
```

```
    e.printStackTrace();
    out.println(e.getMessage());
  }
  return out.toString();
  }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 10.

```
Ejecución
del ejemplo 10

Conjunto de normas antes de la publicación:
Conjunto de normas
Nombre: getApprover
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Norma: LargeOrderApprover
Nombre de visualización: LargeOrderApprover
Descripción: nulo
Presentación de usuario expandida: si el número de artículos pedidos es
superior a 10 y el pedido es superior a 5000 USD, entonces se requiere la
aprobación del gestor
Presentación de usuario: si el número de artículos pedidos es superior a
{0} y el pedido es superior a ${1}, entonces se requiere la aprobación del {2}
Nombre de parámetro: param0
Valor de parámetro: 10
Nombre de parámetro: param1
Valor de parámetro: 5000
Nombre de parámetro: param2
Valor de parámetro: manager
Norma: DefaultApprover
Nombre de visualización: DefaultApprover
Descripción: nulo
Presentación de usuario expandida: approver = peer
Presentación de usuario: approver = {0}
Nombre de parámetro: param0
Valor de parámetro: peer

Conjunto de normas después de la publicación:
Conjunto de normas
Nombre: getApprover
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Norma: LargeOrderApprover
Nombre de visualización: LargeOrderApprover
Descripción: nulo
Presentación de usuario expandida: si el número de artículos pedidos es
superior a 10 y el pedido es superior a 5000 USD, entonces se requiere la
aprobación del supervisor
Presentación de usuario: si el número de artículos pedidos es superior a
{0} y el pedido es superior a ${1}, entonces se requiere la aprobación del {2}
Nombre de parámetro: param0
Valor de parámetro: 10
Nombre de parámetro: param1
Valor de parámetro: 5000
Nombre de parámetro: param2
Valor de parámetro: supervisor
Norma: DefaultApprover
Nombre de visualización: DefaultApprover
Descripción: nulo
Presentación de usuario expandida: approver = peer
Presentación de usuario: approver = {0}
Nombre de parámetro: param0
Valor de parámetro: peer
```

Ejemplo 11: añadir una norma nueva a partir de una plantilla a un conjunto de normas

En este ejemplo, se añade una norma nueva a partir de una plantilla a un conjunto de normas. Antes de que se cree la instancia de norma nueva, se crean sus parámetros.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.Parameter;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRuleTemplate;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class Example11
{
static Formatter out = new Formatter();

static public String executeExample11()
{
 try
 {
  out.clear();
  // Recuperar un grupo de normas empresariales por espacio de nombres y
  nombre
  List<BusinessRuleGroup> brgList = BusinessRuleManager
   .getBRGsByTNSAndName(
   "http://BRSamples/com/ibm/websphere
   /sample/brules",
   QueryOperator.EQUAL,
   "ApprovalValues",
   QueryOperator.EQUAL, 0, 0);

  if (brgList.size() > 0)
  {
   // Obtener el primer grupo de normas empresariales de la lista
   // Debería ser el único grupo de normas empresariales de la
   lista, ya que la
   // combinación de espacio de nombres destino y nombre es
   exclusiva
   BusinessRuleGroup brg = brgList.get(0);
   // Obtener la operación del grupo de normas empresariales que
   // tiene la norma empresarial que se modificará, ya que
   // las normas empresariales están asociadas a una operación
   // específica
   Operation op = brg.getOperation("getApprover");

   // Obtener la norma empresarial en la operación que
   se modificará
   List<BusinessRule> ruleList =
   op.getBusinessRulesByName(
    "getApprover", QueryOperator.EQUAL, 0,0);

   if (ruleList.size() > 0)
   {
```

```
out.println("");
out.printlnBold("Conjunto de normas antes de la publicación:");
// Obtener la norma que modificar. Las normas son únicas por
// espacio de nombre destino y nombre, pero para este ejemplo
// sólo hay una norma empresarial llamada
"getApprover"
RuleSet ruleSet = (RuleSet) ruleList.get(0);
out.print(RuleArtifactUtility.printRuleSet(rule
Set));
```

Para añadir una norma nueva al conjunto de normas, debe localizarse la plantilla adecuada en el conjunto de normas y crear una instancia a partir de la plantilla. La plantilla puede localizarse por su nombre.

```
// Obtener la lista de plantillas de normas
ListRuleSetRuleTemplate> ruleTemplates =
ruleSet
 .getRuleTemplates();

Iterator<RuleSetRuleTemplate> templateIterator
= ruleTemplates
 .iterator();

while (templateIterator.hasNext())
{
 RuleSetRuleTemplate template =
 templateIterator.next();

 // Localizar la plantilla que usar para crear una norma nueva
 if
 (template.getName().equals("Template_Larg
 eOrder"))
 {
```

Para una instancia de plantilla, debe crearse una lista de parámetros.

```
 // Crear una lista para los parámetros de esta instancia
 // de norma de plantilla
 List<ParameterValue> paramList =
 new ArrayList<ParameterValue>();

 // A partir de la definición de plantilla, obtener un parámetro específico
 // y establecer el valor
 Parameter param =
 template.getParameter("param0");
 ParameterValue paramValue = param
  .createParameterValue("
  20");

 // Añadir parámetro a la lista
 paramList.add(paramValue);

 // Obtener el parámetro siguiente y establecer el valor
 param = template.getParameter("param1");
 paramValue =
 param.createParameterValue("7500");

 // Añadir parámetro a la lista
 paramList.add(paramValue);

 // Obtener el parámetro siguiente y establecer el valor
 param =
 template.getParameter("param2");
  paramValue = param
  .createParameterValue("
```

```
 2nd-line manager");

// Añadir parámetro a la lista
paramList.add(paramValue);
```

Una vez creados los parámetros, se puede crear la instancia de la plantilla.

```
// Crear la instancia de la norma de plantilla con la lista
// de parámetros
RuleSetTemplateInstanceRule
 templateInstance = template
 .createRuleFromTemplate
 ("ExtraLargeOrder",
 paramList);
// Obtener el bloque de normas para el conjunto de normas
RuleBlock ruleBlock =
ruleSet.getFirstRuleBlock();
```

Cuando se crea la instancia de plantilla, ésta se puede añadir al bloque de normas. Cuando se ha añadido al bloque de normas, se puede solicitar entre las demás instancias de normas de plantilla.

```
 // Añadir la norma de plantilla al bloque de normas
 ruleBlock.addRule(templateInstance)
 ;

 break;
 }
}

// Utilizar la lista original o crear una lista nueva
// de grupos de normas empresariales.
List<BusinessRuleGroup> publishList = new
ArrayList<BusinessRuleGroup>();

// Añadir el grupo de normas empresariales cambiado a la
list
publishList.add(brg);

// Publicar la lista con el grupo de
rule group
BusinessRuleManager.publish(publishList, true);

out.println("");

// Recuperar los grupos de normas empresariales otra vez para verificar que
// se publicaron los cambios
out.printlnBold("Conjunto de normas después de la publicación:");

brgList = BusinessRuleManager
.getBRGsByTNSAndName(
  "http://BRSamples/com/ibm/websphere
  /sample/brules",
  QueryOperator.EQUAL,
  "ApprovalValues",
  QueryOperator.EQUAL, 0, 0);

brg = brgList.get(0);
op = brg.getOperation("getApprover");
ruleList = op.getBusinessRulesByName(
     "getApprover", QueryOperator.EQUAL,
     0, 0);

ruleSet = (RuleSet) ruleList.get(0);
out.print(RuleArtifactUtility.printRuleSet(rule
Set));
}
```

```
  }
 } catch (BusinessRuleManagementException e)
 {
  e.printStackTrace();
  out.println(e.getMessage());
 }
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 11.

```
Ejecución
del ejemplo 11

Conjunto de normas antes de la publicación:
Conjunto de normas
Nombre: getApprover
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Norma: LargeOrderApprover
Nombre de visualización: LargeOrderApprover
Descripción: nulo
Presentación de usuario expandida: si el número de artículos pedidos es
superior a 10 y el pedido es superior a 5000 USD, entonces se requiere la
aprobación del supervisor
Presentación de usuario: si el número de artículos pedidos es superior a
{0} y el pedido es superior a ${1}, entonces se requiere la aprobación del {2}
Nombre de parámetro: param0
Valor de parámetro: 10
Nombre de parámetro: param1
Valor de parámetro: 5000
Nombre de parámetro: param2
Valor de parámetro: supervisor
Norma: DefaultApprover
Nombre de visualización: DefaultApprover
Descripción: nulo
Presentación de usuario expandida: approver = peer
Presentación de usuario: approver = {0}
Nombre de parámetro: param0
Valor de parámetro: peer

Conjunto de normas después de la publicación:
Conjunto de normas
Nombre: getApprover
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
Norma: LargeOrderApprover
Nombre de visualización: LargeOrderApprover
Descripción: nulo
Presentación de usuario expandida: si el número de artículos pedidos es
superior a 10 y el pedido es superior a 5000 USD, entonces se requiere la
aprobación del supervisor
Presentación de usuario: si el número de artículos pedidos es superior a
{0} y el pedido es superior a ${1}, entonces se requiere la aprobación del {2}
Nombre de parámetro: param0
Valor de parámetro: 10
Nombre de parámetro: param1
Valor de parámetro: 5000
Nombre de parámetro: param2
Valor de parámetro: supervisor
Norma: DefaultApprover
Nombre de visualización: DefaultApprover
Descripción: nulo
Presentación de usuario expandida: approver = peer
Presentación de usuario: approver = {0}
Nombre de parámetro: param0
```

```
Valor de parámetro: peer
Norma: ExtraLargeOrder
Nombre de visualización:
Descripción: nulo
Presentación de usuario expandida: si el número de artículos pedidos es
superior a 20 y el pedido es superior a 7500 USD, entonces se requiere la
aprobación del gestor de segunda línea
Presentación de usuario: si el número de artículos pedidos es superior a
{0} y el pedido es superior a ${1}, entonces se requiere la aprobación del {2}
Nombre de parámetro: param0
Valor de parámetro: 20
Nombre de parámetro: param1
Valor de parámetro: 7500
Nombre de parámetro: param2
Valor de parámetro: 2nd-line manager
```

Ejemplo 12: modificar una plantilla en una tabla de decisiones cambiando el valor de un parámetro y después publicando

En este ejemplo, una condición y una acción, ambas definidas con plantillas, se modifican en una tabla de decisiones cambiando los valores de los parámetros antes de publicarlas.

La manera más sencilla de modificar condiciones y acciones de una tabla de decisiones es utilizar nombres exclusivos para las plantillas en cada nivel de condición y para cada acción. Los nombres exclusivos se pueden buscar y después se pueden aplicar cambios a instancias de plantilla definidas con la plantilla. Cuando se realizan cambios a una instancia de plantilla de una plantilla determinada, se actualizarán todos los valores de condición definidos con esa plantilla a ese nivel. En las expresiones de acción, cada instancia es exclusiva y cambiar una no cambia las demás.

Para este ejemplo, existen diversos métodos adicionales que se crearon para simplificar la localización de un límite de caso específico para actualizar, buscar el valor de parámetro específico y buscar la expresión de acción definida con una plantilla determinada.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Vector;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.Template;
import com.ibm.wbiserver.brules.mgmt.dtable.ActionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.CaseEdge;
import com.ibm.wbiserver.brules.mgmt.dtable.ConditionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTable;
import com.ibm.wbiserver.brules.mgmt.dtable.TemplateInstanceExpression;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeAction;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeBlock;
import
com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionValueDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeNode;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;

public class Example12 {
static Formatter out = new Formatter();
```

```
static public String executeExample12()
{
 try
 {
   out.clear();
    // Recuperar un grupo de normas empresariales por espacio de nombres y
   name
   List<BusinessRuleGroup> brgList = BusinessRuleManager
    .getBRGsByTNSAndName(
    "http://BRSamples/com/ibm/websphere
    /sample/brules",
    QueryOperator.EQUAL,
    "ConfigurationValues",
    QueryOperator.EQUAL, 0, 0);

   if (brgList.size() > 0)
   {
    // Obtener el primer grupo de normas empresariales de la lista
      // Debería ser el único grupo de normas empresariales de la
    list as
   // combinación de espacio de nombres destino y nombre es
    unique
    BusinessRuleGroup brg = brgList.get(0);

    // Obtener la operación del grupo de normas empresariales que
    // tiene la norma empresarial que se modificará, ya que
    // las normas empresariales están asociadas a una operación
    // específica
    Operation op = brg.getOperation("getMessages");

    // Obtener las normas empresariales disponibles para esta operación
    List<BusinessRule> ruleList =
    op.getAvailableTargets();

    // Para esta operación sólo hay una norma empresarial y es la
    // empresa que deseamos actualizar
    DecisionTable decisionTable = (DecisionTable)
    ruleList.get(0);
    out.println("");
    out.printlnBold("Tabla de decisiones antes de la publicación:");
    out
     .print(RuleArtifactUtility
     .printDecisionTable(decisionT
      able));
```

La norma de inicialización y la condición y las acciones están contenidos en un bloque de árbol. Con el bloque de árbol, se puede recuperar el nodo raíz.

```
    // Obtener el bloque de árbol que contiene todas las condiciones
    // y acciones para la tabla de decisiones
    TreeBlock treeBlock = decisionTable.getTreeBlock();
    // Desde el bloque de árbol, obtener el nodo de árbol que es el
    // punto de inicio para navegar a través de la tabla de decisiones
    TreeNode treeNode = treeBlock.getRootNode();
```

La condición que actualizar se definió con una plantilla con el nombre “Condition Value Template 2.1”. El método `getCaseEdge` buscará recursivamente desde el TreeNode hasta el límite de caso apropiado para encontrar el límite de caso en que se define la plantilla. El método espera que el nivel al que se define la plantilla sea conocido además del nivel actual. Este método se puede utilizar para buscar el límite de caso con una plantilla utilizando un nombre específico, en situaciones en que se utilice el mismo nombre en varios límites de caso.

```
// Buscar el límite de caso a nivel 1 por debajo del raíz con
// plantilla específica con un valor de parámetro que tiene
// un nombre específico. Como empezamos en la parte superior,
// la profundidad actual es 0
CaseEdge caseEdge = getCaseEdge(treeNode, "param0",
"Condition Value Template 2.1", 1, 0);
```

Una vez encontrado el límite de caso, se puede recuperar `ConditionValueTemplateInstance` para la condición.

```
if (caseEdge != null)
{
 // Se encontró el límite de caso. Obtener la definición de valor del
 // límite de caso
 TreeConditionValueDefinition condition =
  caseEdge
  .getValueDefinition();
 // Get the condition expression defined with a
  plantilla
  TemplateInstanceExpression conditionExpression
  = condition
   .getConditionValueTemplateInstance(
 );
```

Con `ConditionValueTemplateInstance`, se puede recuperar el valor de parámetro apropiado y después actualizarlo con el método `getParameterValue`.

```
// Obtener la plantilla para la expresión
Template conditionTemplate =
conditionExpression
 .getTemplate();

 // Comprobar que la plantilla es correcta, ya que es posible tener
 // varias plantillas para un valor de condición, pero solo una
 // aplicada
 if (conditionTemplate.getName().equals(
  "Condition Value Template 2.1"))
 {
  // Obtener el valor de parámetro
  ParameterValue parameterValue =
  getParameterValue("param0",
   conditionExpression);

  // Establecer el valor de parámetro nuevo
  parameterValue.setValue("info");
 }
```

A continuación se pueden recuperar las expresiones de acción definidas con plantillas que necesitan actualizarse. El método `getActionExpressions` devolverá todas las acciones que están definidas con la plantilla que se llama Action Value Template 1.

```
ConditionNode conditionNode = (ConditionNode)
treeNode;

// Obtener el nodo de árbol de límites de caso
ListCaseEdge> caseEdges =
conditionNode.getCaseEdges();

// Crear una lista que contenga todas las expresiones de acción que
// también necesitan actualizarse. Because every
action is
// independiente de las demás, aunque la plantilla
// se comparta, todas deben actualizarse.
List<TemplateInstanceExpression> expressions =
new Vector<TemplateInstanceExpression>();
```

```
// Recuperar todas las expresiones
for (CaseEdge edge : caseEdges)
{
  getActionExpressions("Action Value
  Template 1", edge,
  expressions);
}
```

Con la lista de expresiones de acción, se puede actualizar cada elemento. En las expresiones de acción definidas con plantillas, se puede actualizar el valor de parámetro correcto.

```
// Obtener el parámetro correcto en cada expresión
for (TemplateInstanceExpression expression
expressions)
{
 for (ParameterValue parameterValue :
 expression
  .getParameterValues())
 {
  // Comprobar los parámetros correctos aunque sólo hay
  // un parámetro en nuestra plantilla
  if
   (parameterValue.getParameter().getN
   ame().equals("param0")) {
    String value =
    parameterValue.getValue();
    parameterValue.setValue("Info
    "
    +
    value.substring(value.
    indexOf(":"),
    value.length()));
 }
 }
}
 // Con el valor de condición y las acciones
updated, the
// puede publicar el grupo de normas empresariales.
// Utilizar la lista original o crear una lista nueva
// de grupos de normas empresariales.
List<BusinessRuleGroup> publishList = new
ArrayList<BusinessRuleGroup>();

 // Añadir el grupo de normas empresariales cambiado a la
list
publishList.add(brg);

 // Publicar la lista con el grupo de
rule group
BusinessRuleManager.publish(publishList, true);

out.println("");

// Recuperar los grupos de normas empresariales otra vez para verificar que
// se publicaron los cambios
out.printlnBold("Tabla de decisiones después de la publicación:");
brgList =
BusinessRuleManager.getBRGsByTNSAndName(
 "http://BRSamples/com/ibm/websphere
 /sample/brules",
 QueryOperator.EQUAL,
 "ConfigurationValues",
 QueryOperator.EQUAL, 0, 0);

brg = brgList.get(0);
op = brg.getOperation("getMessages");
```

```
   ruleList = op.getAvailableTargets();

   decisionTable = (DecisionTable)
   ruleList.get(0);
   out.print(RuleArtifactUtility
    .printDecisionTable(decisionTable))
    ;
    }
   }
 } catch (BusinessRuleManagementException e)
 {
   e.printStackTrace();
   out.println(e.getMessage());
 }
 return out.toString();
}

/*
Método para navegar recursivamente a través de una tabla de decisiones y localizar
un límite de caso que tenga una plantilla con un nombre determinado y que contenga un
parámetro específico que cambiar. Este método asume que se conoce el nivel (profundidad)
de la tabla de decisiones del valor que se debe cambiar y que se rastrea el nivel
actual (currentDepth) *
*/
static private CaseEdge getCaseEdge(TreeNode node, String pName,
  String templateName, int depth, int currentDepth)
{
 // Comprobar si el nodo actual es una acción. Eso es una indicación
 // de que esta rama de la tabla de decisiones se ha agotado
 // al buscar el límite de caso
 if (node instanceof ActionNode)
 {
  return null;
 }

 // Obtener los límites de caso para este nodo
 List<CaseEdge> caseEdges = ((ConditionNode) node).getCaseEdges();
 for (CaseEdge caseEdge : caseEdges)
 {

 // Comprobar si se ha alcanzado el nivel correcto
 if (currentDepth < depth)
  {
   // Moverse abajo un nivel y después llamar a getCaseEdge de nuevo
   // para procesar ese nivel
   currentDepth++;
   return getCaseEdge(caseEdge.getChildNode(), pName,
    templateName, depth, currentDepth);
  } else
  {
   // Se ha llegado al nivel correcto. Obtener
   la condición para
   // comprobar si las plantillas con esa condición
   // coinciden con la búsqueda de plantilla
   TreeConditionValueDefinition condition = caseEdge
    .getValueDefinition();

   // Obtener la expresión para la condición que se ha definido
   // con una plantilla
   TemplateInstanceExpression expression = condition
    .getConditionValueTemplateInstance();
   // Obtener la plantilla a partir de la expresión
      Template template = expression.getTemplate();

   // Comprobar si se trata de la plantilla buscada
    if (template.getName().equals(templateName))
    {
```

```
     // Se comprueba que la plantilla coincide
      return caseEdge;
    } else
     caseEdge = null;
   }
 }
 return null;
}

/*
Este método comprobará los distintos valores de parámetro de una expresión y si se
encuentra el correcto, devolverá ese valor de parámetro.
*/
private static ParameterValue getParameterValue(String pName,
  TemplateInstanceExpression expression)
{
 // Comprobar que la expresión no es nula, ya que ello indicaría
 // que la expresión que se pasó probablemente no se había definido
 // con una plantilla y no tenía ningún parámetro que comprobar.
 if (expression != null) {
  // Obtener los valores de parámetro para la expresión
  List<ParameterValue> parameterValues = expression
    .getParameterValues();

  for (ParameterValue parameterValue : parameterValues)
  {
   // Para los parámetros distintos, comprobar que coincidan con
   // el valor de parámetro encontrado

   if
    (parameterValue.getParameter().getName().equals(pName
   ))
   {
    // Devolver el valor de parámetro que coincidía
      return parameterValue;
   }
  }
 }
return null;
}
/*
Este método encuentra todas las expresiones de acción que están definidas con una
plantilla determinada. Trabaja recursivamente a través de un límite de caso y añade
expresiones de acción que coinciden con el parámetro de expresiones.
*/

private static void getActionExpressions(String templateName,
CaseEdge next, List<TemplateInstanceExpression>
expressions)
{
 ActionNode actionNode = null;
 TreeNode treeNode = next.getChildNode();

 // Comprobar si el nodo actual está a nivel del nodo de acción.
 if (treeNode instanceof ConditionNode)
 {
  List<CaseEdge> caseEdges = ((ConditionNode) treeNode)
   .getCaseEdges();

  Iterator<CaseEdge> caseEdgesIterator =
  caseEdges.iterator();

  // Examinar todos los límites de caso para encontrar las
  // expresiones de acción
  while (caseEdgesIterator.hasNext())
   {
    getActionExpressions(templateName,
```

```
    caseEdgesIterator.next(),
     expressions);
   }
 } else {
   // ActionNode encontrado
   actionNode = (ActionNode) treeNode;

   List<TreeAction> treeActions = actionNode.getTreeActions();
   // Comprobar que al menos hay un treeAction especificado para
   // la expresión y examinar las expresiones comprobando
   // si éstas se han creado con una plantilla
   // determinada
   if (!treeActions.isEmpty())
   {

    Iterator<TreeAction> iterator =
    treeActions.iterator();

   while (iterator.hasNext())
    {
     TreeAction treeAction = iterator.next();
     TemplateInstanceExpression expression =
     treeAction
      .getValueTemplateInstance();

     Template template = expression.getTemplate();

     if (template.getName().equals(templateName))
     {
      // Expression found with matching
      plantilla
      expressions.add(expression);
     }
    }
   }
  }
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 12.

```
Ejecución
del ejemplo 12

Conjunto de normas antes de la publicación:
Tabla de decisiones
Nombre: getMessages
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules

Tabla de decisiones después de la publicación:
Tabla de decisiones
Nombre: getMessages
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
```

Ejemplo 13: añadir un valor de condición y acciones a una tabla de decisiones

En este ejemplo, se añaden un valor de condición y una acción a una tabla de decisiones. Los valores de condición se pueden añadir a una tabla de decisiones mediante el uso de una plantilla.

Cuando se añade un valor de condición a un nodo de condición, está añadiendo un límite de caso. El límite de caso nuevo se añade al final de la lista de límites de caso. Para el valor de condición, se debe especificar una expresión de instancia de

plantilla que tenga establecidos los valores de parámetro apropiados. Para especificar la expresión de instancia de plantilla, tendrá que usar una plantilla específica. Se recomienda dar a las plantillas de cada nivel de nodo de condición un nombre exclusivo, para recuperar las plantillas correctas para ese tipo de condición. Utilizar una definición de plantilla única podría dificultar la determinación del nivel al que se está añadiendo la condición.

Cuando se establece el valor de condición de un nodo de condición, en realidad se estará añadiendo el valor de condición con la misma instancia de plantilla a todos los nodos de condición del mismo nivel. Esto se realiza como parte del equilibrado de la tabla de decisiones. Además, como parte de añadir un valor de condición nuevo, se añadirán nodos de acción nuevos. Estos nodos de acción tienen acciones de árbol con valores nulos especificados para la presentación de usuario y la expresión de instancia de plantilla. Debido a que el valor de condición se puede añadir a un nodo de condición que no tenga un nodo de acción como nodo hijo, la adición de un nodo de condición puede provocar mayor número de nodos de acción. El número de nodos de acción está basado en el nivel en que se añade el nodo de condición, el número de nodos de condición de ese nivel y el número de nodos de condición de cada nivel hijo.

Para encontrar los nodos de acción que se han creado, se puede realizar una búsqueda de nodos de acción con acciones de árbol que tengan presentaciones de usuario y expresiones de instancia de plantilla nulas. Se puede usar una `TreeActionValueTemplate` para crear una expresión que se puede establecer en el `TreeAction`. Este patrón sería necesario repetirlo para todos los nodos de acción nuevos.

Para este ejemplo, se proporcionaron dos métodos como ayuda para la configuración de las acciones de árbol nuevas. `getEmptyActionNode` busca recursivamente un nodo de acción vacío desde el nodo de condición actual y getParameterValue devuelve el valor de un parámetro que se especificó por nombre.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.Parameter;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.Template;
import com.ibm.wbiserver.brules.mgmt.ValidationException;
import com.ibm.wbiserver.brules.mgmt.dtable.ActionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.CaseEdge;
import com.ibm.wbiserver.brules.mgmt.dtable.ConditionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTable;
import com.ibm.wbiserver.brules.mgmt.dtable.TemplateInstanceExpression;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeAction;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeActionTermDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeActionValueTemplate;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeBlock;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionValueTemplate;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeNode;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
```

```
public class Example13
{
 static Formatter out = new Formatter();

 static public String executeExample13()
 {
  try
  {
   out.clear();

   // Recuperar un grupo de normas empresariales por espacio de nombres
   // y nombre
   List<BusinessRuleGroup> brgList = BusinessRuleManager
    .getBRGsByTNSAndName(
   "http://BRSamples/com/ibm/websphere/sample/brules",
     QueryOperator.EQUAL,"ConfigurationValues",
     QueryOperator.EQUAL, 0, 0);

   if (brgList.size() > 0)
    {
     // Obtener el primer grupo de normas empresariales de la
     // lista. Debería ser el único grupo de
     // normas empresariales de la lista ya que la combinación
     // de espacio de nombres destino y nombre es exclusiva
     BusinessRuleGroup brg = brgList.get(0);

     // Obtener la operación del grupo de normas
     // empresariales que tiene la norma empresarial que se
     // modificará, ya que las normas empresariales están
     // asociadas a una operación determinada
     Operation op = brg.getOperation("getMessages");

     // Obtener las normas empresariales disponibles para
     // esta operación
     List<BusinessRule> ruleList =
      op.getAvailableTargets();

     // Para esta operación sólo hay una norma
     // empresarial, y es la que deseamos
     // actualizar

     DecisionTable decisionTable = (DecisionTable)
      ruleList.get(0);
      out.println\nBold("Tabla de decisiones antes de la publicación:");
      out.print(RuleArtifactUtility
      .printDecisionTable(decisionTable));
```

Es necesario localizar el nivel en el que se va a añadir el valor de condición. Normalmente esto se pasa como parámetro, ya que la interfaz de usuario o la aplicación que utilizan las clases saben donde añadir la condición.

```
     // Obtener el bloque de árbol que contiene todas las
     // condiciones y las acciones para la tabla de
     // decisiones
     TreeBlock treeBlock =
      decisionTable.getTreeBlock();

     // Desde el bloque de árbol, obtener el nodo de árbol que
     // es el punto de inicio para para navegar por
     // la tabla de decisiones
     ConditionNode conditionNode = (ConditionNode)
      treeBlock.getRootNode();

     // Obtener los límites de caso para este nodo que es
     // el primer nivel de la condiciones
     List<CaseEdge> caseEdges =
```

```
 conditionNode.getCaseEdges();

// Obtener el límite de caso al que se
// añadirá la nueva condición
CaseEdge caseEdge = caseEdges.get(0);

// Para el límite de caso, obtener el nodo de condición
// para recuperar la plantilla para la
// condición
conditionNode = (ConditionNode)
    caseEdge.getChildNode();

// Obtener las plantillas para la condición
List<TreeConditionValueTemplate>
 treeValueConditionTemplates = conditionNode
 .getAvailableValueTemplates();

Iterator<TreeConditionValueTemplate>
 treeValueConditionTemplateIterator =
 treeValueConditionTemplates.iterator();

TreeConditionValueTemplate conditionTemplate =
 null;
```

Mediante el uso de nombres de plantilla únicos en cada nivel de nodo de condición en la tabla de decisiones, puede garantizar más fácilmente que el valor de condición se está añadiendo al valor de nodo de condición correcto.

```
// Buscar la plantilla que debería usarse
 while
(treeValueConditionTemplateIterator.hasNext())
{
 conditionTemplate =
  treeValueConditionTemplateIterator
   .next();
 if (conditionTemplate.getName().equals(
  "Condition Value Template
  2.1"))
  {
   // Plantilla encontrada
   break;
  }
  conditionTemplate = null;
}
if (conditionTemplate != null)
{
```

Cuando se encuentra la plantilla correcta, se puede crear una instancia y establecerse el parámetro apropiado antes de añadirse al nodo de condición.

```
// Obtener la definición de parámetro de la
// plantilla
Parameter conditionParameter =
conditionTemplate.getParameter("param0");

// Crear una instancia de valor de parámetro que
// utilizar en una instancia de plantilla de condición
// nueva
ParameterValue conditionParameterValue =
  conditionParameter
  .createParameterValue("fatal");

List<ParameterValue>
  conditionParameterValues = new
  ArrayList<ParameterValue>();

// Añadir el valor de parámetro a una lista
```

```
conditionParameterValues
 .add(conditionParameterValue);

// Crear una instancia de plantilla de condición
// nueva con el valor del parámetro
TemplateInstanceExpression
 newConditionValue =
 conditionTemplate
  .createTemplateInstanceExpression(c
 onditionParameterValues);
// Añadir la instancia de plantilla de condición a
// este nodo de condición
conditionNode

.addConditionValueToThisLevel(newConditionValue);
// Cuando se añade un nodo de condición, hay
// nodos de acción nuevos que se crean
// y vacíos. Éstos deben rellenarse con
// instancias de plantillas de acción. Al
// buscar cada nodo de acción vacío
// desde el nivel padre, se pueden encontrar
// todos los nuevos nodos de acción vacíos.
conditionNode = (ConditionNode)
conditionNode.getParentNode();
```

Con el valor de condición añadido al nodo de condición, deben establecerse las acciones de árbol de los nodos de acción nuevos con `TreeActionValueTemplate`. Primer hay que localizar el nodo de acción vacío para los límites de caso. Utilice el nodo de condición padre para garantizar que a medida que itere por los nodos de condición, encontrará todos los nodos de acción.

```
// Obtener los límites de caso para el nodo padre
caseEdges = conditionNode.getCaseEdges();

Iterator<CaseEdge> caseEdgesIterator =
caseEdges.iterator();

while (caseEdgesIterator.hasNext())
{
 // Para cada límite de caso, recuperar un
 // nodo de acción vacío si existe
 ActionNode actionNode =
 getEmptyActionNode(caseEdgesIterator
 .next());

 // Comprobar si todas las acciones están completadas
 if (actionNode != null)
 {
```

Cuando se encuentra un nodo de acción con acciones de árbol vacías, se debe establecer la acción de árbol con una `TreeActionValueTemplate`. Primero se debe localizar la plantilla y después especificar los parámetros antes de crear una instancia de plantilla. Cuando se ha creado la instancia de plantilla, la acción de árbol se puede actualizar. En este ejemplo el parámetro se estableció con un valor procedente de otra acción de árbol de otro nodo de acción bajo el mismo nodo de condición. En otras tablas de decisión en que otra acción de árbol pudiera no tener un valor que se pueda usar para crear los valores de parámetro nuevos, el valor se tendrá que pasar como parámetro desde la aplicación.

```
  // Obtener la lista de las
  // acciones de árbol. Éstas
  // no son las acciones
  // reales, sino los
  // contenedores para las
```

```
// acciones
List<TreeAction>
treeActionList = actionNode
 .getTreeActions();

List<TreeActionTermDefinition>
treeActionTermDefinitions =
 treeBlock
.getTreeActionTermDefinitions();

List<TreeActionValueTemplate>
 treeActionValueTemplates =
 treeActionTermDefinitions
 .get(0).getValueTemplates();

TreeActionValueTemplate
 actionTemplate = null;

for (TreeActionValueTemplate
  tempActionTemplate :
  treeActionValueTemplates)
  {

  if
  (tempActionTemplate.get
   Name().equals(
   "Action Value
   Template 1"))
   {
    actionTemplate =
    tempActionTemplate;
    break;
   }
  }

  if (actionTemplate != null)
  {
   // Obtener otra acción
   // que está bajo
   // el nodo de condición
   // padre para
   // usar el valor como
   // base para
   // el mensaje de error en
   // el nodo de
   // acción nuevo. Subir
   // hasta el
   // nodo de condición
   // padre primero
   ConditionNode
   parentNode =
   (ConditionNode)
   actionNode
    .getParentNode();

   // Obtener el primer límite
   // de caso del
   // nodo padre, ya que esta
   // acción siempre
   // estará completada
   // pues las acciones nuevas
   // se añaden al final
   // de la lista de
   // límites de caso.
   CaseEdge caseE =
    parentNode.getCas
    eEdges().get(
```

```
 0);

// El nodo hijo es un
// nodo de acción
// y al mismo
// nivel que el nodo de
// acción nuevo.
ActionNode aNode =
(ActionNode) caseE
 .getChildNode();

// Obtener la lista de las
// acciones de árbol
TreeAction
 existingTreeAction =
 aNode
 .getTreeActions()
 .get(0);

// Obtener la expresión de
// instancia
// de plantilla para la
// acción de árbol
// desde la que puede
// recuperar el
// parámetro

TemplateInstanceExpression
existingExpression =
 existingTreeAction
 .getValueTemplateInstance();

ParameterValue
existingParameterValue =
getParameterValue(
  "param0",
existingExpression);

String actionValue =
existingParameterValue
 .getValue();

// Crear el mensaje
// nuevo a partir
// del mensaje de la
// acción de árbol
// existente
actionValue = "Fatal"
  +
actionValue.substring(actionValue
 .indexOf(":"), actionValue
 .length());
 Parameter
 actionParameter =
 actionTemplate
 .getParameter("param0");

 // Obtener el parámetro
 // desde la plantilla
 ParameterValue
 actionParameterValue =
  actionParameter
 .createParameterValue(actionValue);

 // Añadir el parámetro a
 // una lista de plantillas
 List<ParameterValue>
```

```
          actionParameterValues = new
          ArrayList<ParameterValue>();

        actionParameterValues.add(actionParameterValue);

          // Crear una instancia de
          // acción de árbol nueva

         TemplateInstanceExpression
          treeAction = actionTemplate
    .createTemplateInstanceExpression(actionParameterValues);

         // Establecer la acción de árbol
         // en el nodo de acción
         // estableciéndola en la lista
         // de acción de árbol
```

Aquí se actualiza la acción de árbol del nodo de acción.

```
          treeActionList.get(0)
          .setValueTemplateInstance(
          treeAction);
          }
         }
       }
   }
   // Con el valor de condición y las acciones
   // actualizados, el grupo de normas empresariales se puede
   // publicar.
   // Utilizar la lista original o crear una lista nueva
   // de grupos de normas empresariales.
    List<BusinessRuleGroup> publishList = new
     ArrayList<BusinessRuleGroup>();

   // Añadir el grupo de normas empresariales cambiado a la
   // lista
    publishList.add(brg);

   // Publicar la lista con el grupo de
   // normas empresariales actualizado

   BusinessRuleManager.publish(publishList, true);

   brgList =
    BusinessRuleManager.getBRGsByTNSAndName(
    "http://BRSamples/com/ibm/websphere/sample/brules",
    QueryOperator.EQUAL, "ConfigurationValues",
    QueryOperator.EQUAL, 0, 0);
    brg = brgList.get(0);
    op = brg.getOperation("getMessages");
    ruleList = op.getAvailableTargets();
    decisionTable = (DecisionTable)
     ruleList.get(0);
    out.println1nBold("Tabla de decisiones después de la publicación:");
    out
     .print(RuleArtifactUtility
      .printDecisionTable(decisionTable));
   }
  } catch (ValidationException e)
  {
   List<Problem> problems = e.getProblems();

   out.println("Problem = " +
   problems.get(0).getErrorType().name());

   e.printStackTrace();
   out.println(e.getMessage());
```

```
 } catch (BusinessRuleManagementException e)
 {
  e.printStackTrace();
  out.println(e.getMessage());
 }
 return out.toString();
}

/*
* Este método busca el límite de caso actual para cualquier nodo
* de acción que tenga acciones de árbol vacías. Un nodo de acción
* vacío se encuentra buscando al final de la lista de límites de
* caso y comprobando si el nodo de acción tiene acciones de árbol
* que tengan una presentación de usuario nula y una
* TemplateInstanceExpression.
*/
private static ActionNode getEmptyActionNode(CaseEdge next)
{
 ActionNode actionNode = null;
 TreeNode treeNode = next.getChildNode();

if (treeNode instanceof ConditionNode)
{
 List<CaseEdge> caseEdges = ((ConditionNode) treeNode)
  .getCaseEdges();

 if (caseEdges.size() > 1)
 {
  // Obtener el límite de caso de más a la derecha, ya que
  // la condición nueva y por tanto las acciones vacías están en
  // el extremo derecho de los límites de caso
  actionNode = getEmptyActionNode(caseEdges
   .get(caseEdges.size() - 1));

  if (actionNode != null)
  {
   return actionNode;
  }
}
} else
{
 actionNode = (ActionNode) treeNode;

 List<TreeAction> treeActions =
 actionNode.getTreeActions();

 if (!treeActions.isEmpty())
 {
 if
 ((treeActions.get(0).getValueUserPresentation() == null)
  &&
 (treeActions.get(0).getValueTemplateInstance() == null))
  {
   return actionNode;
  }
 }
  `actionNode = null;
 }
 return actionNode;
}
/*
* Este método comprobará los distintos valores de parámetro de
* una expresión y si se encuentra el correcto, devolverá ese
* valor de parámetro.
*/
private static ParameterValue getParameterValue(String pName,
  TemplateInstanceExpression expression)
```

```
{
 ParameterValue parameterValue = null;

 // Comprobar que la expresión no es nula, ya que ello indicaría
 // indicate that the expression that was passed in was
 // probably not defined with a template and does not have
 // any parameters to check.
 if (expression != null)
 {
  // Obtener los valores de parámetro para la expresión
  List<ParameterValue> parameterValues = expression
   .getParameterValues();
  Iterator<ParameterValue> parameterIterator =
   parameterValues
   .iterator();

  // Para los parámetros distintos, comprobar que
  // coincidan con el valor de parámetro encontrado
  while (parameterIterator.hasNext())
  {
   parameterValue = parameterIterator.next();

   if
   (parameterValue.getParameter().getName().equals(pName))
   {
   // Devolver el valor del parámetro que
   // coincidió
   return parameterValue;
   }
 }
}
 return parameterValue;
}
}
```

Ejemplo

Salida en el navegador web para el ejemplo 13.

```
Ejecución
del ejemplo 13

Tabla de decisiones antes de la publicación:
Tabla de decisiones
Nombre: getMessages
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules

Tabla de decisiones después de la publicación:
Tabla de decisiones
Nombre: getMessages
Espacio de nombres: http://BRSamples/com/ibm/websphere/sample/brules
```

Ejemplo 14: Manejar errores de un conjunto de normas

Este ejemplo se centra en la manera de interceptar problemas en un conjunto de normas y averiguar qué problema se ha producido para que se pueda mostrar el mensaje apropiado o se pueda llevar a cabo la acción apropiada para corregir la situación.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
```

```
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.ValidationException;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import
com.ibm.wbiserver.brules.mgmt.problem.ProblemStartDateAfterEndDate;
import com.ibm.wbiserver.brules.mgmt.problem.ValidationError;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class Example14 {
static Formatter out = new Formatter();

static public String executeExample14() {
 try {
   out.clear();

    // Recuperar un grupo de normas empresariales por espacio de nombres y
   name
   List<BusinessRuleGroup> brgList = BusinessRuleManager
     .getBRGsByTNSAndName(
     "http://BRSamples/com/ibm/websphere
     /sample/brules",
     QueryOperator.EQUAL,
     "ApprovalValues",
     QueryOperator.EQUAL, 0, 0);

   if (brgList.size() > 0) {
     // Obtener el primer grupo de normas empresariales de la lista
      // Debería ser el único grupo de normas empresariales de la
     list as
   // combinación de espacio de nombres destino y nombre es
     exclusivo
     BusinessRuleGroup brg = brgList.get(0);
     out.println("Business Rule Group retrieved");

     // Obtener la operación del grupo de normas empresariales que
     // tiene la norma empresarial que se modificará, ya que
     // las normas empresariales están asociadas a una operación
     // específica
     Operation op = brg.getOperation("getApprover");

     // Recuperar norma específica por nombre
     List<BusinessRule> ruleList =
     op.getBusinessRulesByName(
       "getApprover", QueryOperator.EQUAL, 0,
       0);

     // Obtener la norma específica
     RuleSet ruleSet = (RuleSet) ruleList.get(0);
     out.println("Conjunto de normas recuperado");

     RuleBlock ruleBlock = ruleSet.getFirstRuleBlock();

     Iterator<RuleSetRule> ruleIterator =
     ruleBlock.iterator();

     // Buscar por las normas para encontrar la que cambiar
     while (ruleIterator.hasNext()) {
       RuleSetRule rule = ruleIterator.next();
```

```
      // Comprobar que la norma se definió con una plantilla
      // ya que así puede cambiarse.
      if (rule instanceof
      RuleSetTemplateInstanceRule) {
       // Obtener la instancia de norma de plantilla
       RuleSetTemplateInstanceRule
       templateInstance =
       (RuleSetTemplateInstanceRule) rule;
       // Comprobar la instancia de norma de plantilla correcta
       if (templateInstance.getName().equals(
         "LargeOrderApprover")) {
```

Para provocar un problema, este ejemplo establece un parámetro en un valor que no es compatible para la expresión. El parámetro está esperando un entero, pero se pasa una serie.

```
         // Obtener el parámetro de la
         template instance
         ParameterValue parameter =
         templateInstance
           .getParameterValue("par
           am1");

         // Set an incorrect value for this
         parámetro
         // Esto causará un error de validación
         parameter.setValue("$3500");
         out.println("Incorrect parameter
         value set");
         break;
       }
     }
   }
   // A este código nunca debería llegarse debido al error
   // introducido
   // anteriormente

  // Con el valor de condición y las acciones actualizadas, se
   business
   // la empresa puede publicar el
   // grupo de normas empresariales.
   // Utilizar la lista original o crear una lista nueva
   // de grupos de normas empresariales.
   List<BusinessRuleGroup> publishList = new
   ArrayList<BusinessRuleGroup>();

   // Añadir el grupo de normas empresariales cambiado a la lista
   publishList.add(brg);

      // Publicar la lista con el grupo de normas empresariales actualizado
   BusinessRuleManager.publish(publishList, true);
}
```

Se puede interceptar una `ValidationException` y, a partir de la excepción, se pueden recuperar los problemas. Para cada problema, se puede comprobar el error para determinar cuál se ha producido. Se puede imprimir un mensaje o llevar a cabo la acción apropiada.

```
} catch (ValidationException e) {
out.println("Error de validación");

List<Problem> problems = e.getProblems();

Iterator<Problem> problemIterator = problems.iterator();

// Comprobar en la lista de problemas el error apropiado y
// realizar la acción apropiada, por ejemplo, informar del error
```

```
              // o corregirlo
  while (problemIterator.hasNext()) {
    Problem problem = problemIterator.next();
    ValidationError error = problem.getErrorType();

    // Comprobar el valor de error específico
    if (error == ValidationError.TYPE_CONVERSION_ERROR) {
      // Manejar el error informando del problema
      out
        .println("Problema: valor incorrecto introducido para un parámetro");
      return out.toString();
    }
    // else if....
    // Se puede realizar comprobaciones para otros errores y se puede
    // mostrar el mensaje de error o llevar a cabo la acción apropiados
    // para corregir el problema
  }
} catch (BusinessRuleManagementException e) {
  out.println("Error producido.");
  e.printStackTrace();
}
return out.toString();
}
}
```

Ejemplo

Salida en el navegador web para el ejemplo 14.

Ejecución del ejemplo 14

```
Grupo de normas empresariales recuperado
Conjunto de normas recuperado
Error de validación
Problema: se introdujo un valor incorrecto para un parámetro
```

Ejemplo 15: manejar errores en un grupo de normas empresariales

Este ejemplo es parecido al ejemplo 14 ya que muestra cómo manejar problemas que se producen cuando se publica un grupo de normas empresariales. Muestra cómo se puede determinar el problema e imprimir el mensaje o llevar a cabo la acción correctos.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleGroup;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManagementException;
import com.ibm.wbiserver.brules.mgmt.BusinessRuleManager;
import com.ibm.wbiserver.brules.mgmt.Operation;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecord;
import com.ibm.wbiserver.brules.mgmt.OperationSelectionRecordList;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.ValidationException;
import com.ibm.wbiserver.brules.mgmt.problem.Problem;
import
com.ibm.wbiserver.brules.mgmt.problem.ProblemStartDateAfterEndDate;
import com.ibm.wbiserver.brules.mgmt.query.QueryOperator;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
```

```
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class Example15
{
static Formatter out = new Formatter();

static public String executeExample15()
{
 try
 {
  out.clear();

    // Recuperar un grupo de normas empresariales por espacio de nombres y
 name
 List<BusinessRuleGroup> brgList = BusinessRuleManager
  .getBRGsByTNSAndName(
  "http://BRSamples/com/ibm/websphere
  /sample/brules",
  QueryOperator.EQUAL,
  "ApprovalValues",
  QueryOperator.EQUAL, 0, 0);
  if (brgList.size() > 0)
  {
   // Obtener el primer grupo de normas empresariales de la lista
   // Debería ser el único grupo de normas empresariales de la
   lista, ya que la
   // combinación de espacio de nombres destino y nombre es
   exclusiva
   BusinessRuleGroup brg = brgList.get(0);
   out.println("Business Rule Group retrieved");

   // Obtener la operación del grupo de normas empresariales que
   // tiene la norma empresarial que se modificará, ya que
   // las normas empresariales están asociadas a una operación
   // específica
   Operation op = brg.getOperation("getApprover");

   // Recuperar norma específica por nombre
   List<BusinessRule> ruleList =
   op.getBusinessRulesByName(
    "getApprover", QueryOperator.EQUAL, 0,
    0);

   // Obtener la norma específica
   RuleSet ruleSet = (RuleSet) ruleList.get(0);
    out.println("Conjunto de normas recuperado");

   RuleBlock ruleBlock = ruleSet.getFirstRuleBlock();

   Iterator<RuleSetRule> ruleIterator =
    ruleBlock.iterator();

   // Buscar por las normas para encontrar la que cambiar
   while (ruleIterator.hasNext())
   {
    RuleSetRule rule = ruleIterator.next();

    // Comprobar que la norma se definió con una plantilla
    // ya que así puede cambiarse.
    if (rule instanceof
    RuleSetTemplateInstanceRule)
    {
     // Obtener la instancia de norma de plantilla
     RuleSetTemplateInstanceRule
     templateInstance =
```

```
    (RuleSetTemplateInstanceRule) rule;

       // Comprobar la instancia de norma de plantilla correcta
    if (templateInstance.getName().equals(
      "LargeOrderApprover"))
    {
        // Obtener el parámetro de la
     template instance
     ParameterValue parameter =
     templateInstance
      .getParameterValue("par
      am1");

     // Establecer el valor para este parámetro
     // Este valor está en el formato correcto y
     // no causará un error de validación
     parameter.setValue("4000");
     out.println("Valor de parámetro de conjunto de normas establecido correctamente");
     break;
     }
   }
 }
```

Para garantizar que un conjunto de normas es correcto, se puede llamar al método de validación. El método de validación está disponible en todos los objetos y devolverá una lista de problemas que se pueden comprobar para determinar el problema. Cuando se llama a la validación de un objeto, también se llama al método de validación en todos los objetos que contiene.

```
// Validar los cambios realizados en el conjunto de normas
List<Problem> problems = ruleSet.validate();
out.println("Conjuntos de normas validado");

// No deberían producirse errores para este caso de prueba, aunque
// compruebe si hay algún problema y a continuación
// realice la acción adecuada para recuperar o informar
// del error
if (problems != null)
{
 Iterator<Problem> problemIterator =
 problems.iterator();

 while (problemIterator.hasNext())
 {
  Problem problem = problemIterator.next();

  if (problem instanceof
  ProblemStartDateAfterEndDate)
  {
   out
   .println("Incorrect
    value entered for a
     parameter");
   return out.toString();
  }
 }
} else
{
 out.println("No se encontraron problemas para el conjunto de normas");
}
// Obtener la lista de destinos de norma disponibles
List<BusinessRule> ruleList2 =
op.getAvailableTargets();

// Obtener la primera norma que se planificará incorrectamente
BusinessRule rule = ruleList2.get(0);
```

```
    // La condición de error será establecer la hora final de una
    // norma planificada en 1 hora antes de la hora de inicio
    // Esto causará un error de validación
    Date future = new Date();
    long futureTime = future.getTime() - 360000;

    // Obtener la lista de selección de la operación para que añada el
    // elemento planificado incorrectamente
    OperationSelectionRecordList opList = op
    .getOperationSelectionRecordList();

    // Crear una nueva instancia de norma planificada
    // No se genera ningún error hasta que se valida o se produce la
    // publicación, ya que podrían realizarse más cambios
    OperationSelectionRecord newRecord = opList
     .newOperationSelectionRecord(new Date(),
     new Date(
     futureTime), rule);
```

Cuando se añade el registro con un conjunto de fechas incorrecto, esto no causa un error. Es posible que se produzca solapamientos o que no se establezcan registros de selección para la operación, ya que nos encontramos en proceso de realizar cambios. El error se encontrará cuando se publique el grupo de normas empresariales con el registro de selección de la operación. El método de validación se llama antes de que los objetos se publiquen y se generarán excepciones si existe algún error.

```
    // Añadir la instancia de norma planificada a la operación
    // Aquí tampoco hay errores
    opList.addOperationSelectionRecord(newRecord);
    out.println("Añadido registro de selección nuevo con planificación incorrecta");
// Con el valor de condición y las acciones actualizadas, se
    business
    // puede publicar el
    // grupo de normas empresariales.
    // Utilizar la lista original o crear una lista nueva
    // de grupos de normas empresariales.
    List<BusinessRuleGroup> publishList = new
    ArrayList<BusinessRuleGroup>();

    // Añadir el grupo de normas empresariales cambiado a la lista
    publishList.add(brg);

    // Publicar la lista con el grupo de normas empresariales actualizado
    BusinessRuleManager.publish(publishList, true);
   }
} catch (ValidationException e) {
out.println("Error de validación");

List<Problem> problems = e.getProblems();

Iterator<Problem> problemIterator = problems.iterator();
// Podrían haber varios problemas
// Repasar los problemas y manjera cada uno o
// informar del problema.
while (problemIterator.hasNext())
{
 Problem problem = problemIterator.next();

 // Cada problema es un tipo distinto que puede compararse
 if (problem instanceof ProblemStartDateAfterEndDate)
 {
  out
  .println("Planificación de norma incorrecta. Fecha inicial posterior a la final.");
  return out.toString();
  }
```

```
    // else if....
    // Se puede realizar comprobaciones para otros errores y se puede
    // mostrar el mensaje de error o llevar a cabo la acción apropiados
    // para corregir el problema
  }
 }catch (BusinessRuleManagementException e)
 {
  out.println("Error producido.");
  e.printStackTrace();
 }
 return out.toString();
 }
}
```

Ejemplo

Salida en el navegador web para el ejemplo 15.

```
Ejecución
del ejemplo 15

Grupo de normas empresariales recuperado
Conjunto de normas recuperado
Valor de parámetro de conjunto de normas establecido correctamente
Conjunto de normas validado
Error de validación
Planificación de norma incorrecta. Fecha inicial posterior a la final.
```

Ejemplos de consultas adicionales

Los ejemplos siguientes no se incluyen con la aplicación que contiene los ejemplos 1 al 15, sin embargo ofrecen más ejemplos sobre la creación de consultas para recuperar grupos de normas empresariales.

En estos ejemplos, se utilizan propiedades y valores de comodín ('_', '%') distintos con operadores distintos (AND, OR, LIKE, NOT_LIKE, EQUAL y NOT_EQUAL).

Ejemplo

En los ejemplos se llevan a cabo consultas en combinaciones distintas de cuatro grupos de normas empresariales. Es importante comprender los distintos atributos y propiedades de los grupos de normas empresariales que se utilizan en las consultas.

```
Nombre: BRG1
Espacio de nombres destino: http://BRG1/com/ibm/br/rulegroup
Propiedades:
organización, 8JAA
departamento, reclamaciones
ID, 00000567
región: Región Sur-central
jefe: Joe Bean

Nombre: BRG2
Espacio de nombres destino: http://BRG2/com/ibm/br/rulegroup
Propiedades:
organización, 7GAA
departamento, contabilidad
ID, 0000047
ID_cert45, ABC
región: Región Norte

Nombre: BRG3
Espacio de nombres destino: http://BRG3/com/ibm/br/rulegroup
Propiedades:
organización, 7FAB
```

```
departamento, finanzas
ID, 0000053
ID_app45, DEF
región: Región Norte-central

Nombre: BRG4
Espacio de nombres destino: http://BRG4/com/ibm/br/rulegroup
Propiedades:
organización, 7HAA
departamento, envíos
ID, 0000023
ID_app45, GHI
región: Región Sur
```

Consulta por una sola propiedad:

A continuación figura un ejemplo de una consulta por una sola propiedad.

```
List<BusinessRuleGroup> brgList = null;

brgList = BusinessRuleManager.getBRGsBySingleProperty(
  "department", QueryOperator.EQUAL,
  "accounting", 0, 0);
// Devuelve BRG2
```

Consulta de grupos de normas empresariales por propiedades y el comodín (%) al principio y al final del valor:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por propiedades y el comodín (%) al principio y al final del valor.

```
// Query Prop AND Prop
QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
  "region", QueryOperator.LIKE,
  "%Region");

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode(
  "ID", QueryOperator.LIKE,
  "000005%");

QueryNode queryNode =
QueryNodeFactory.createAndNode(leftNode,
  rightNode);

brgList =
BusinessRuleManager.getBRGsByProperties(queryNode, 0, 0);
// Devuelve BRG1 y BRG3
```

Consulta de grupos de normas empresariales por propiedades y el comodín ('_'):

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por propiedades y el comodín (%).

```
brgList = BusinessRuleManager.getBRGsBySingleProperty("ID",
QueryOperator.LIKE, "00000_3", 0, 0);

// Devuelve BRG3 y BRG4
```

Consulta de un grupo de normas empresariales por propiedades con varios comodines ('_' y '%'):

Este es un ejemplo de un grupo de normas empresariales por propieades con varios comodines (‘_’ y ‘%’).

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("region",
QueryOperator.LIKE, "__uth%Region",
  0, 0);

// Devuelve BRG1 y BRG4
```

Consulta de grupos de normas empresariales por el operador NOT_LIKE y el comodín (‘_’):

A continuación figura un ejemplo de una consulta de un grupo de normas empresariales por el operador NOT_LIKE y el comodín (‘_’).

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("organization",
QueryOperator.NOT_LIKE,
  "7__A", 0, 0);

// Devuelve BRG1 y BRG3
brgList =
BusinessRuleManager.getBRGsBySingleProperty("organization",
QueryOperator.NOT_LIKE,
  "7%", 0, 0);

// Devuelve BRG1
```

Consulta de grupos de normas empresariales por el operador NOT_EQUAL:

A continuación figura un ejemplo de una consulta de un grupo de normas empresariales por el operador NOT_EQUAL.

```
brgList =
BusinessRuleManager.getBRGsBySingleProperty("department",
QueryOperator.NOT_EQUAL,
  "claims", 0, 0);
// Devuelve BRG1
```

Consulta de grupos de normas empresariales por PropertyIsDefined:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por PropertyIsDefined.

```
PropertyIsDefinedQueryNode node =
QueryNodeFactory.createPropertyIsDefinedQueryNode("manager"
);

brgList = BusinessRuleManager.getBRGsByProperties(node, 0,
0);

// Devuelve BRG1
```

Consulta de grupos de normas empresariales por NOT PropertyIsDefined:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por NOT PropertyIsDefined.

```
// NOT Prop
QueryNode node =
QueryNodeFactory.createPropertyIsDefinedQueryNode("manager"
);

NotNode notNode = QueryNodeFactory.createNotNode(node);
```

```
brgList = BusinessRuleManager.getBRGsByProperties(notNode,
0, 0);

// Devuelve BRG1
```

Consulta de grupos de normas empresariales por varias propiedades con un solo nodo NOT:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades con un solo nodo NOT.

```
// Prop AND NOT Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.EQUAL, "accounting");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID",
  QueryOperator.LIKE, "00000%");

AndNode andNode = QueryNodeFactory.createAndNode(leftNode,
  notNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG2
```

Consulta de grupos de normas empresariales por varias propiedades con varios nodos NOT combinados con el operador AND:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades con varios nodos NOT combinados con un operador AND.

```
// NOT Prop AND NOT Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
QueryOperator.EQUAL, "accounting");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE, "cla%");

NotNode notNode2 =
QueryNodeFactory.createNotNode(leftNode);

AndNode andNode = QueryNodeFactory.createAndNode(notNode,
notNode2);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG1 y BRG2
```

Consulta de grupos de normas empresariales por varias propiedades con varios nodos NOT combinados con el operador OR:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades con varios nodos NOT combinados con un operador OR.

```
// NOT Prop OR NOT Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE, "acc%");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
  "department", QueryOperator.EQUAL,
  "claims");

NotNode notNode2 =
QueryNodeFactory.createNotNode(leftNode);

OrNode orNode = QueryNodeFactory.createOrNode(notNode,
notNode2);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

//Devuelve BRG1, BRG2, BRG3 y BRG4
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con varios operadores AND:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con varios operadores AND.

```
// (Prop AND Prop) AND (Prop AND Prop)
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE, "acc%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.EQUAL, "7GAA");

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(leftNode,rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("ID",
QueryOperator.LIKE,"000004_");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",
QueryOperator.EQUAL,
  "NorthRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNode =
QueryNodeFactory.createAndNode(andNodeLeft,andNodeRight);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG2
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND y OR:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND y OR.

```
// (Prop AND Prop) OR (Prop AND NOT Prop)
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE, "acc%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.EQUAL, "7GAA");

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(leftNode,rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.EQUAL, "8JAA");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE, "%lRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,notNode);

OrNode orNode = QueryNodeFactory.createOrNode(andNodeLeft,
andNodeRight);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Devuelve BRG2 y BRG3
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND y NOT:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND y NOT.

```
// Prop AND NOT (Prop AND Prop)
QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID",
QueryOperator.LIKE, "000005%");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
QueryOperator.EQUAL,
  "8JAA");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",QueryOper
ator.LIKE,
  "%lRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

NotNode notNode =
QueryNodeFactory.createNotNode(andNodeRight);
```

```
AndNode andNode = QueryNodeFactory.createAndNode(leftNode,
notNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG3
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores NOT y OR:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores NOT y OR.

```
// NOT (Prop AND Prop) OR Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "8_A_");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("region",QueryOper
ator.LIKE,
  "%lRegion");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

NotNode notNode =
QueryNodeFactory.createNotNode(andNodeRight);

OrNode orNode = QueryNodeFactory.createOrNode(notNode,
rightNode);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Devuelve BRG3
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados.

```
// Prop AND (Prop AND (Prop AND Prop))
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");
```

```
AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

PropertyIsDefinedQueryNode node2 =
QueryNodeFactory.createPropertyIsDefinedQueryNode("ID_cert4
5");

AndNode andNode = QueryNodeFactory.createAndNode(node2,
andNodeLeft);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);
// Devuelve BRG2
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados.

```
// (Prop AND (Prop AND Prop)) AND Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",QueryOper
ator.LIKE,
"___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
QueryOperator.LIKE,
"7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
QueryOperator.LIKE,
"%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_app45",QueryOp
erator.LIKE, "GH_");

AndNode andNode =
QueryNodeFactory.createAndNode(andNodeLeft, leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG4
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados y un nodo NOT:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados y un nodo NOT.

```
// Prop AND (Prop AND (Prop AND NOT Prop))
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "%lRegion");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,notNode);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
  QueryOperator.LIKE,
  "AB_");

AndNode andNode = QueryNodeFactory.createAndNode(leftNode,
andNodeLeft);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG2
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores AND anidados.

```
// (Prop AND (Prop AND Prop)) AND Prop - Devuelve un valor vacío
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

AndNode andNodeRight =
QueryNodeFactory.createAndNode(leftNode2,rightNode2);

AndNode andNodeLeft =
QueryNodeFactory.createAndNode(rightNode,andNodeRight);

QueryNode leftNode =
```

```
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
  QueryOperator.LIKE,
  "GH_");

AndNode andNode =
QueryNodeFactory.createAndNode(andNodeLeft, leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

//No devuelve ningún BRG
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados.

```
// (Prop OR (Prop OR Prop)) OR Prop

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(leftNode2,rightNode2);

OrNode orNodeLeft =
QueryNodeFactory.createOrNode(rightNode,orNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
  QueryOperator.LIKE,
  "GH_");

OrNode orNode = QueryNodeFactory.createOrNode(orNodeLeft,
leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Devuelve BRG1
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados.

```
// (Prop OR (Prop OR NOT Prop)) OR Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "___thRegion");
```

```
QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(leftNode2,notNode);

OrNode orNodeLeft =
QueryNodeFactory.createOrNode(rightNode,orNodeRight);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("ID_cert45",
  QueryOperator.LIKE,
  "GH_");

OrNode orNode = QueryNodeFactory.createOrNode(orNodeLeft,
leftNode);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

// Devuelve BRG3
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados y un nodo NOT:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR y un nodo NOT.

```
// Prop OR NOT(Prop OR Prop)
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "___thRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode(
  "organization",
  QueryOperator.LIKE,
  "7%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
  "department",
  QueryOperator.LIKE,
  "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(rightNode2,
  rightNode);

NotNode notNode =
QueryNodeFactory.createNotNode(orNodeRight);

OrNode orNodeLeft = QueryNodeFactory.createOrNode(leftNode,
notNode);
```

```
brgList =
BusinessRuleManager.getBRGsByProperties(orNodeLeft, 0, 0);

// Devuelve BRG3
```

Consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados y un nodo NOT:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por varias propiedades combinadas con operadores OR anidados y un nodo NOT.

```
// NOT(Prop OR Prop) OR Prop
QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "%lRegion");

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode(
  "organization",
  QueryOperator.LIKE,
  "7%");

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode(
  "department",
  QueryOperator.LIKE,
  "%ing");

OrNode orNodeRight =
QueryNodeFactory.createOrNode(rightNode2,rightNode);

NotNode notNode =
QueryNodeFactory.createNotNode(orNodeRight);

OrNode orNodeLeft =
QueryNodeFactory.createOrNode(notNode,leftNode);

brgList =
BusinessRuleManager.getBRGsByProperties(orNodeLeft, 0, 0);

// Devuelve BRG2 y BRG4
```

Consulta de grupos de normas empresariales por una lista de nodos combinados con un operador AND:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por una lista de nodos combinados con un operador AND

```
// Lista AND
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "%thRegion");

list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7%");
```

```
list.add(rightNode2);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

list.add(leftNode);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "7H%");

list.add(leftNode2);

AndNode andNode = QueryNodeFactory.createAndNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG4
```

Consulta de grupos de normas empresariales por una lista de nodos y un nodo NOT combinados con un operador AND:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por una lista de nodos y un nodo NOT combinados con un operador AND

```
// Lista AND con notNode
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "%thRegion");

list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "8%");

NotNode notNode =

QueryNodeFactory.createNotNode(rightNode2);

list.add(notNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

list.add(leftNode);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",

list.add(leftNode2);

AndNode andNode = QueryNodeFactory.createAndNode(list);
```

```
brgList = BusinessRuleManager.getBRGsByProperties(andNode,
0, 0);

// Devuelve BRG4
```

Consulta de grupos de normas empresariales por una lista de nodos combinados con un operador OR:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por una lista de nodos combinados con un operador OR

```
// Lista OR
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "%thRegion");

list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "8%");

list.add(rightNode2);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

list.add(leftNode);

OrNode orNode = QueryNodeFactory.createOrNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

//Devuelve BRG3
```

Consulta de grupos de normas empresariales por una lista de nodos y un nodo Not combinados con un operador OR:

A continuación figura un ejemplo de una consulta de grupos de normas empresariales por una lista de nodos y un nodo Not combinados con un operador OR

```
// Lista OR con nodo Not
List<QueryNode> list = new ArrayList<QueryNode>();

QueryNode rightNode =
QueryNodeFactory.createPropertyQueryNode("region",
  QueryOperator.LIKE,
  "%thRegion");

list.add(rightNode);

QueryNode rightNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "8%");

NotNode notNode =
QueryNodeFactory.createNotNode(rightNode2);
```

```
list.add(notNode);

QueryNode leftNode =
QueryNodeFactory.createPropertyQueryNode("department",
  QueryOperator.LIKE,
  "%ing");

list.add(leftNode);

QueryNode leftNode2 =
QueryNodeFactory.createPropertyQueryNode("organization",
  QueryOperator.LIKE,
  "8%");

list.add(leftNode2);

OrNode orNode = QueryNodeFactory.createOrNode(list);

brgList = BusinessRuleManager.getBRGsByProperties(orNode,
0, 0);

//Devuelve BRG1, BRG2, BRG3 y BRG4
```

Clases de operaciones comunes

Esta sección contiene clases adicionales que se han utilizado para simplificar las operaciones comunes.

Clase Formatter

Esta clase proporciona métodos distintos para ayudar a la visualización de los distintos ejemplos. Añade códigos HTML distintos para aplicar formato a la salida.

```
package com.ibm.websphere.sample.brules.mgmt;

public class Formatter {

private StringBuffer buffer;

public Formatter()
{
  buffer = new StringBuffer();
}

public void println(Object o)
{
  buffer.append(o);
  buffer.append("<br>\n");
  }

  public void print(Object o)
  {
    buffer.append(o);
  }

  public void printlnBold(Object o)
  {
    buffer.append("<b>");
    buffer.append(o);
    buffer.append("</b><brbr>\n");
  }

  public void printBold(Object o)
  {
    buffer.append("<b>");
    buffer.append(o);
    buffer.append("</b>");
```

```
  }

  public String toString()
  {
    return buffer.toString();
  }

  public void clear()
  {
    buffer = new StringBuffer();
  }
}
```

Clase RuleArtifactUtility

Esta clase de programa de utilidad tiene dos métodos públicos. El primer método público es para imprimir una tabla de decisiones. Este método utiliza un método privado que utiliza la recurrencia para imprimir las condiciones y acciones de la tabla de decisiones. El segundo método público es para imprimir un conjunto de normas.

```
package com.ibm.websphere.sample.brules.mgmt;

import java.util.Iterator;
import java.util.List;

import com.ibm.wbiserver.brules.mgmt.BusinessRule;
import com.ibm.wbiserver.brules.mgmt.Parameter;
import com.ibm.wbiserver.brules.mgmt.ParameterValue;
import com.ibm.wbiserver.brules.mgmt.RuleTemplate;
import com.ibm.wbiserver.brules.mgmt.Template;
import com.ibm.wbiserver.brules.mgmt.dtable.ActionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.CaseEdge;
import com.ibm.wbiserver.brules.mgmt.dtable.ConditionNode;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTable;
import com.ibm.wbiserver.brules.mgmt.dtable.DecisionTableRule;
import
com.ibm.wbiserver.brules.mgmt.dtable.DecisionTableTemplateInstanceRule;
import com.ibm.wbiserver.brules.mgmt.dtable.TemplateInstanceExpression;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeAction;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeActionTermDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeBlock;
import
com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionTermDefinition;
import
com.ibm.wbiserver.brules.mgmt.dtable.TreeConditionValueDefinition;
import com.ibm.wbiserver.brules.mgmt.dtable.TreeNode;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleBlock;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSet;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRule;
import com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetRuleTemplate;
import
com.ibm.wbiserver.brules.mgmt.ruleset.RuleSetTemplateInstanceRule;

public class RuleArtifactUtility
{
  static Formatter out = new Formatter();

  /*
  Método para imprimir una tabla de decisiones con las condiciones y
  acciones imprimidas en formato tabular HTML. Las condiciones
  y acciones se imprimen con un método individual que
  funciona de forma recursiva a través de los límites de caso de las
  tablas de decisiones.
  */

  public static String printDecisionTable(BusinessRule
```

```
ruleArtifact)
{
  out.clear();
  out.printlnBold("Tabla de decisiones");
  DecisionTable decisionTable = (DecisionTable)
  ruleArtifact;
  out.println("Nombre: " +
  decisionTable.getName());
  out.println("Espacio de nombres: " +
  decisionTable.getTargetNameSpace());

  // Produce como salida la norma init de la tabla de decisiones
  antes de
  // trabajar en la tabla de condiciones y
  acciones
  DecisionTableRule initRule =
  decisionTable.getInitRule();
  if (initRule != null)
  {
    out.printBold("Norma init: ");
    out.println(initRule.getName());
    out.println("Nombre de visualización: " +
    initRule.getDisplayName());
    out.println("Descripción: " +
    initRule.getDescription());
    // La presentación de usuario ampliada
    llenará automáticamente la
    // presentación con los valores
    de parámetro y se puede utilizar para
    // mostrar si la norma init se ha
    definido con una plantilla. Si no
    // se ha definido ninguna plantilla la
    presentación de usario ampliada
    // es la misma que la presentación
    regular.
    out.println("Presentación de usuario
    ampliada: "
      +
    initRule.getExpandedUse
    rPresentation());
    // La presentación de usuario regular
    tendrá marcadores de posición en la
    // serie donde el
    // parámetro se puede sustituir si
    se ha definido la norma init con una
    // plantilla
    // Si no se ha definido la norma con
    una plantilla, la presentación de
    // usuario sólo será
    // una serie sin los
    marcadores de posición. Los marcadores de posición
    tienen un
    // formato de {n} donde
    // n es el índice (de base cero) del
    parámetro de la plantilla. Este
    // valor
    // se puede utilizar para crear una
    interfaz para la edición donde
    hay
    // campos con
    // los valores de parámetro disponibles
    para la edición
    out.println("Presentación de usuario: " +
    initRule.getUserPresentation());
    // Las normas init se pueden definir con
    o sin una plantilla
    // Compruebe para asegurarse de que
```

```
        se ha utilizado una plantilla antes de intentar
        // acceder a los parámetros
        if (initRule instanceof
        DecisionTableTemplateInstanceRule)
        {
          DecisionTableTemplateIn
          stanceRule
          templateInstance =
          (DecisionTableTemplateI
          nstanceRule) initRule;

          RuleTemplate template =
          templateInstance.getRul
          eTemplate();

          List<Parameter>
          parameters =
          template.getParameters(
          );
          Iterator<Parameter>
          paramIterator =
          parameters.iterator();

          Parameter parameter =
          null;

          while
          (paramIterator.hasNext(
          )) {
          parameter =
          paramIterator.next();

           out.println("Parameter
          Name: " +
          parameter.getName());
          out.println("Parameter
          Value: "
          +
          templateInstance.getPar
          ameterValue(parameter
          .getName()));
          }
        }
      }
      // Para el resto de la tabla de decisiones, comience en
      el raíz y de forma
      // recursiva trabaje en los distintos límites
      de caso y
      // acciones
      TreeBlock treeBlock =
      decisionTable.getTreeBlock();
      TreeNode treeNode = treeBlock.getRootNode();

      printDecisionTableConditionsAndActions(treeNode
      , 0);
      out.println("");
      return out.toString();
    }
    /*Método para trabajar de forma recursiva por los límites de caso e imprimir
    las condiciones y acciones.
  */
    static private void printDecisionTableConditionsAndActions(
      TreeNode treeNode, int indent)
    {
      out.print("<table border=\"1\">");
      if (treeNode instanceof ConditionNode)
      {
```

```
// Obtener los límites de caso del
nodo del árbol actual
// y para cada límite de caso imprimir
las condiciones
ConditionNode conditionNode =
(ConditionNode) treeNode;

List<CaseEdge> caseEdges =
conditionNode.getCaseEdges();
Iterator<CaseEdge> caseEdgeIterator
= caseEdges.iterator();

CaseEdge caseEdge = null;

while (caseEdgeIterator.hasNext())
{
  out.print("<tr>");
  // Si se trata del inicio
  de las condiciones del
  // nodo de condición,
  imprimir el término
  if (indent == 0)
  {
  out.print("<td>");

  TreeConditionTermDefinition
  termDefinition =
  conditionNode
  .getTermDefinition();

  out.print(termDefinitio
  n.getUserPresentation()
  );
  out.print("</td>");
  indent++;
  } else {
  // Después de que se ha
  imprimido el término de la condición
  de un
  // límite de caso omitir para
  el resto de los límites
  de caso
  out.print("<td></td>");
  }

  caseEdge =
  caseEdgeIterator.next()
  ;

  out.print("<td>");

  // Comprobar si una plantilla
  ha definido un
  caseEdge
  if
  (caseEdge.getValueDefin
  ition() != null)
  {
  TemplateInstanceExpress
  ion templateInstance =
  caseEdge
  .getValueTemplateInstan
  ce();

  out.println(templateIns
  tance.getExpandedUserPr
  esentation());
```

```
TreeConditionValueDefin
ition valueDef =
caseEdge
.getValueDefinition();

out.println(valueDef.ge
tUserPresentation());

Template template =
templateInstance.getTem
plate();

// Obtener los parámetros
de la definición de
plantilla e
// imprimir los
nombres de parámetro y
los valores
List<Parameter>
parameters =
template.getParameters(
);
Iterator<Parameter>
paramIterator =
parameters.iterator();

List<ParameterValue>
parameterValues =
templateInstance
.getParameterValues();
Iterator<ParameterValue
> paramValues =
parameterValues
.iterator();

Parameter parameter =
null;
ParameterValue
parameterValue = null;

while
(paramIterator.hasNext(
) &&
paramValues.hasNext())
{
parameter =
paramIterator.next();
parameterValue =
paramValues.next();

out.println("Parameter
Name: " +
parameter.getName());
out.println("Parameter
Value: "
  +
parameterValue.getValue
());
}
}

out.print("</td><td>");
// Imprimir el nodo hijo
de caseEdge
printDecisionTableCondi
tionsAndActions(caseEdg
```

```
  e.getChildNode(),
  0);

  out.print("</td></tr>")
  ;
  }


  // Añadir una condición Otherwise si
  existe
  TreeNode otherwise =
  conditionNode.getOtherwiseCase();

  if (otherwise != null)
  {
    out.print("<tr><td></td>
    <td>Otherwise</td><td>
    ");
    // Print the Otherwise
    ConditionNode
    printDecisionTableCondi
    tionsAndActions(otherwi
    se, 0);
    out.print("</td></td>")
    ;
  }
  out.print("</table>");
} else {
  // Se ha encontrado ActionNode y
  es necesario que una lógica distinta
  // imprima las TreeActions
  ActionNode actionNode =
  (ActionNode) treeNode;
  List<TreeAction> treeActions =
  actionNode.getTreeActions();

  Iterator<TreeAction>
  treeActionIterator =
  treeActions.iterator();

  TreeAction treeAction = null;

  // ActionNode puede contener
  varias TreeActions para
  // imprimir
  while
  (treeActionIterator.hasNext())
  {
    out.print("<tr>");
    treeAction =
    treeActionIterator.next
    ();

    TreeActionTermDefinitio
    n treeActionTerm =
    treeAction
    .getTermDefinition();

    if (indent == 0) {
    out.print("<td>");
    out.print(treeActionTer
    m.getUserPresentation()
    );
    out.print("</td>");
    }
    out.print("<td>");
    TemplateInstanceExpress
```

```
ion templateInstance =
treeAction
.getValueTemplateInstan
ce();

// Comprobar que se ha
especificado una plantilla
para
// TreeAction
antes de trabajar con el
// nombre de parámetro y
los valores
if (templateInstance !=
null) {
out.println(templateIns
tance.getExpandedUserPr
esentation());

Template template =
templateInstance.getTem
plate();

List<Parameter>
parameters =
template.getParameters(
);

Iterator<Parameter>
paramIterator =
parameters.iterator();

List<ParameterValue>
parameterValues =
templateInstance
.getParameterValues();
Iterator<ParameterValue
> paramValues =
parameterValues
.iterator();

Parameter parameter =
null;
ParameterValue
parameterValue = null;

while
(paramIterator.hasNext(
) &&
paramValues.hasNext())
{
{parameter =
paramIterator.next();
parameterValue =
paramValues.next();

out.println(" Parameter
Name: " +
parameter.getName());
out.println(" Parameter
Value: "
+
parameterValue.getValue
());

}
} else
{
```

```
        // Si no se ha utilizado una
        plantilla, el único elemento
        que está
        // disponible es
        UserPresentation si
        se ha
        // especificado al
        crearse la norma
        out.print(treeAction.ge
        tValueUserPresentation(
        ));
        }

        out.print("</td></tr>")
        ;
      }
      out.print("</table>");
    }
  }
  /*
    Método para imprimir un conjunto de normas
  */
public static String printRuleSet(BusinessRule
ruleArtifact)
{
   out.clear();
   out.printlnBold("Conjunto de normas");
   RuleSet ruleSet = (RuleSet) ruleArtifact;
   out.println("Nombre: " + ruleSet.getName());
   out.println("Espacio de nombres: " +
   ruleSet.getTargetNameSpace());

   // Las normas de un conjunto de normas están
   contenidas en un bloque de normas
   RuleBlock ruleBlock =
   ruleSet.getFirstRuleBlock();

   Iterator<RuleSetRule> ruleIterator =
   ruleBlock.iterator();

   RuleSetRule rule = null;

   // Recorrer en interación las normas del bloque de normas.
   while (ruleIterator.hasNext())
   {
     rule = ruleIterator.next();
     out.printBold("Norma: ");
     out.println(rule.getName());
     out.println("Nombre de visualización: " +
     rule.getDisplayName());
     out.println("Descripción: " +
     rule.getDescription());
     // La presentación de usuario ampliada
     llenará automáticamente la
     // presentación con los valores
     de parámetro y se puede utilizar para
     // la visualización si se ha definido la norma
     con una plantilla. Si no
     // se ha definido ninguna plantilla la
     presentación de usario ampliada
     // es la misma que la presentación
     regular.
     out.println("Presentación de usuario
     ampliada: "
       +
     rule.getExpandedUserPre
     sentation());
```

```
// La presentación de usuario regular
tendrá marcadores de posición en la
// serie donde el parámetro se puede
sustituir si la norma se
// ha definido con una plantilla. Si
la norma no se ha definido con
// una plantilla, la presentación de
usuario sólo será una serie
// sin marcadores de posición. Los
marcadores de posición están en el formato {n}
// donde n es el índice (de base cero)
del parámetro de la
// plantilla. Este valor se puede utilizar
para crear una interfaz para
// la edición donde hay campos
con los valores de parámetro
// disponibles para la edición
out.println("Presentación de usuario: " +
rule.getUserPresentation());

// Comprobar si la norma se ha definido
con una plantilla
if (rule instanceof
RuleSetTemplateInstanceRule) {
  RuleSetTemplateInstance
  Rule templateInstance =
  (RuleSetTemplateInstanc
  eRule) rule;

  RuleSetRuleTemplate
  template =
  templateInstance
  .getRuleSetRuleTemplate
  ();

  List<Parameter>
  parameters =
  template.getParameters(
  );
  Iterator<Parameter>
  paramIterator =
  parameters.iterator();

  Parameter parameter =
  null;

  // Recuperar todos los
  parámetros y generar como salida
  el nombre y el valor
  while
  (paramIterator.hasNext(
  ))
  {
  parameter =
  paramIterator.next();

  out.println("Parameter
  Name: " +
  parameter.getName());
  out.println("Parameter
  Value: "
    +
  templateInstance.getPar
  ameterValue(
  parameter.getName()).ge
  tValue());
  }
```

```
        }
      }
      out.println("");
      return out.toString();
    }
}
```

Programación de widget

Business Space proporciona widgets que están habilitados para diversos productos de gestión de procesos de negocio de IBM. Sin embargo, también puede crear sus propios widgets, posiblemente integrándolos con los de IBM. Las siguientes referencias explican por qué debe crear los widgets y cómo los debe crear y proporciona referencias a las API que puede utilizar para crear sus propios widgets.

En el centro de información de Business Space, consulte estos temas:

- Visión general del desarrollo de widgets
- Guía de programación de widget

Capítulo 4. Desarrollo de aplicaciones cliente para procesos de empresa y tareas

Puede utilizar una herramienta de creación de modelos para generar y desplegar procesos de empresa y tareas. Se interactúa con estos procesos y tareas durante la ejecución; por ejemplo, se inicia un proceso o las tareas se reclaman y se completan. Puede utilizar Business Process Choreographer Explorer para interactuar con procesos y tareas, o utilizar las API de Business Process Choreographer para desarrollar clientes personalizados para estas interacciones.

Acerca de esta tarea

Estos clientes pueden ser clientes EJB (Enterprise JavaBeans), los clientes de servicios Web o los clientes Web que utilizan los componentes JSF (JavaServer Faces) de Business Process Choreographer Explorer. Business Process Choreographer proporciona las API de EJB (Enterprise JavaBeans) e interfaces para los servicios Web para que desarrolle estos clientes. Cualquier aplicación puede acceder a la API EJB mediante cualquier aplicación Java. Se puede acceder a las interfaces de servicios Web desde cualquier entorno Java o desde cualquier entorno Microsoft .Net.

Comparación de las interfaces de programación para interactuar con procesos empresariales y tareas de usuario

Las interfaces de programación genéricas de EJB (Enterprise JavaBeans), servicio Web, JMS (Java Message Service) y REST (servicios de transferencia de estado representativo) están disponibles para crear aplicaciones cliente que interactúan con procesos empresariales y tareas de usuario. Cada una de estas interfaces tiene características diferentes.

La interfaz de programación que elija depende de varios factores, que incluyen la funcionalidad que debe proporcionar la aplicación cliente, de si tiene una infraestructura de cliente de usuario final existente, de si desea manejar los flujos de trabajo de usuarios. Para ayudarle a decidir qué interfaz utilizar, en la tabla siguiente se comparan las características de las interfaces de programación EJB, servicio Web, JMS y REST.

	Interfaz EJB	**Interfaz de servicio Web**	**Interfaz de mensaje JMS**	**Interfaz REST**
Funcionalidad	Esta interfaz está disponible tanto para los procesos empresariales, como para las tareas de usuario. Utilice esta interfaz para crear clientes que trabajen normalmente con procesos y tareas.	Esta interfaz está disponible tanto para los procesos empresariales, como para las tareas de usuario. Utilice esta interfaz para crear clientes para un conjunto conocido de procesos y tareas.	Esta interfaz está disponible sólo para los procesos empresariales. Utilice esta interfaz para crear clientes de mensajería para un conjunto conocido de procesos.	Esta interfaz está disponible tanto para los procesos empresariales, como para las tareas de usuario. Utilice esta interfaz para crear clientes de estilo Web 2.0 para un conjunto de procesos y tareas conocidos.

	Interfaz EJB	Interfaz de servicio Web	Interfaz de mensaje JMS	Interfaz REST
Manejo de datos	Soporta la carga de artefactos remotos de esquemas para acceder a metadatos de objetos de empresa. Si la aplicación cliente EJB se ejecuta en la misma célula que el WebSphere Process Server al que se conecta, los esquemas que se necesitan para los objetos de empresa de los procesos y las tareas no tienen que estar disponibles en el cliente, se pueden cargar desde el servidor utilizando el Cargador de artefactos remotos (RAL). RAL se puede utilizar también entre células si la aplicación cliente se ejecuta en una instalación de servidor completa deWebSphere Process Server. Sin embargo, RAL no se puede utilizar en una configuración de célula cruzada donde la aplicación cliente se ejecuta en una instalación de cliente WebSphere Process Server.	Los artefactos de esquema para los datos de entrada, los datos de salida y las variables deben estar disponibles en un formato apropiado en el cliente.	Los artefactos de esquema para los datos de entrada, los datos de salida y las variables deben estar disponibles en un formato apropiado en el cliente.	Los artefactos de esquema para los datos de entrada, los datos de salida y las variables deben estar disponibles en un formato apropiado en el cliente.
Entorno de cliente	Una instalación de WebSphere Process Server o una instalación de cliente de WebSphere Process Server.	Cualquier entorno de ejecución que soporte las llamadas de servicio Web, incluidos los entornos de Microsoft .NET.	Cualquier entorno de ejecución que soporte los clientes JMS, incluidos los módulos SCA que utilizan las importaciones JMS SCA.	Cualquier entorno en tiempo de ejecución que admita clientes REST.
Seguridad	Seguridad de Java Platform, Enterprise Edition (Java EE).	Seguridad de servicios Web.	Seguridad de Java Platform, Enterprise Edition (Java EE) para la instalación de WebSphere Process Server. También puede proteger las colas donde la aplicación cliente JMS pone los mensajes de API, por ejemplo utilizando los mecanismos de seguridad de WebSphere MQ.	La aplicación cliente que llame a métodos REST debe utilizar un mecanismo de autenticación HTTP apropiado.

Una operación puede ser expuesta por varios protocolos. Observe las siguientes consideraciones generales si utiliza la misma operación en distintos protocolos.

- En las interfaces de servicio Web y REST, todos los identificadores de objetos, como por ejemplo PIID, AIID y TKIID, se representan mediante un tipo de serie. Sólo la interfaz de la API EJB espera un ID de objeto de tipo seguro.
- La sobrecarga de la operación sólo se utiliza para métodos EJB y no para operaciones WSDL. En algunos casos, existen varias operaciones WSDL; en

otros, sólo existe una operación WSDL que permite todas las variaciones de parámetros por omisión (`minOccurs="0"`) o valores nulos (`nillable="true"`).

- En algunos métodos EJB, los espacios de nombres XML y los nombres locales se pasan como parámetros distintos. La mayoría de las operaciones WSDL utilizan el tipo de esquema XML QName para pasar estos parámetros.
- Las interacciones asíncronas con operaciones de respuesta de solicitud WSDL de larga ejecución, como por ejemplo la operación callWithReplyContext en la interfaz EJB o la operación callAsync en la interfaz WSDL, se representan mediante la operación call en la interfaz JMS.
- La interfaz EJB devuelve un conjunto de objetos de la API, que exponen los métodos getter y setter para los campos contenidos. Las interfaces de servicios Web y REST devuelven documentos de tipo complejo (XML o JSON) al cliente.
- Algunos servicios de Human Task Manager que operan en tareas de usuario también están disponibles como servicios de Business Flow Manager que operan en actividades que invocan una tarea de usuario.

Tareas relacionadas

Desarrollo de aplicaciones de cliente EJB
Las API de EJB proporcionan un conjunto de métodos genéricos para desarrollar aplicaciones cliente EJB para trabajar con los procesos de empresa y tareas de usuario instalados en WebSphere Process Server.

Desarrollo de aplicaciones cliente de la API de servicio Web
Puede desarrollar aplicaciones cliente que acceden a las aplicaciones de proceso empresarial y las aplicaciones de tareas de usuario mediante las API de servicios Web de Business Process Choreographer. El proceso de desarrollo de aplicaciones cliente consta de varios pasos obligatorios y opcionales, incluyendo la generación de un proxy de servicio Web y la adición de políticas de seguridad y transacción a la aplicación cliente.

Desarrollo de aplicaciones de cliente JMS
Puede desarrollar aplicaciones cliente que acceden a las aplicaciones de proceso empresarialde manera asíncrona mediante la API de JMS (Java Messaging Service).

Consultas sobre procesos empresariales y datos de tarea

Los procesos empresariales de larga ejecución y las tareas de usuario se almacenan de manera persistente en la base de datos y las consultas pueden acceder a ellos. También se puede acceder a los datos de plantilla de las plantillas de procesos empresariales yde tareas de usuario utilizando una interfaz de consulta.

Las interfaces de consulta de EJB, la API de consulta y la API de tabla de consulta están disponibles con Business Process Choreographer.

En función de los clientes que accedan a los datos relacionados con tareas o procesos, una o varias de estas interfaces puede ser la opción adecuada. Las API de REST y de servicios Web están disponibles en Business Process Choreographer para consultar datos de lista de tareas y procesos. Sin embargo, para consultas de lista de tareas o de lista de procesos de gran volumen, utilice la API de tabla de consulta de EJB o la API de tabla de consulta de REST de Business Process Choreographer por razones de rendimiento.

Comparación de las interfaces de programación para recuperar datos de procesos y tareas

Business Process Choreographer proporciona una API de tabla de consulta y una API de consulta para recuperar datos de procesos y tareas. Cada una de estas interfaces tiene características diferentes.

La interfaz de consulta que seleccione dependerá de diversos factores, que incluyen la funcionalidad que deba proporcionar la aplicación cliente, si tiene una infraestructura de cliente de usuario final existente y de las consideraciones de rendimiento. Para ayudarle a decidir qué interfaz utilizar, en la tabla siguiente se comparan las características de las interfaces de programación de tabla de consulta y de consulta.

Característica	API de tabla de consulta	API de consulta
Disponibilidad	La API de tabla de consulta está disponible para la interfaz EJB de Business Flow Manager y la interfaz de programación de REST.	La API de consulta está disponible para las interfaces de programación de REST, EJB, servicio Web y JMS.
Métodos para la recuperación de contenido	La API proporciona los métodos siguientes: • queryEntities • queryEntityCount • queryRows • queryRowCount	La API proporciona los métodos siguientes: • query • queryAll • queryProcessTemplates • queryTaskTemplates
Métodos para la recuperación de metadatos	La API proporciona los métodos siguientes: • getQueryTableMetaData • findQueryTableMetaData	La API proporciona los métodos siguientes: • QueryResultSet.getColumnType
Nombre de tabla de consulta	Especifica la tabla de consulta en la que se ejecuta la API de tabla de consulta. Sólo se puede consultar una única tabla de consulta cada vez. Por ejemplo, `queryEntities("CUST.TASKS", ...)`.	La cláusula SELECT especifica las columnas y vistas de base de datos predefinidas en las que se ejecuta la consulta. Esta especificación es similar a una cláusula select de SQL. Por ejemplo, `query("TASK.TKIID, TASK.STATE, WORK_ITEM.REASON", ...)`.
Cláusula SELECT y atributos seleccionados	Utilice las opciones de filtro de la API de tabla de consulta para especificar los atributos que va a devolver la consulta. Debido a que la consulta se ejecuta contra una sola tabla de consulta, los atributos se identifican de forma exclusiva por sus nombres.	Utilice la cláusula SELECT para especificar atributos. La sintaxis del nombre de atributo es: *nombre_vista.nombre_atributo*. Por ejemplo, para buscar estados de tarea, especifique TASK.STATE en la consulta.
Cláusula WHERE y filtros	Utilice la propiedad queryCondition en la API de tabla de consulta para filtrar más el resultado de consultas. Las tablas de consulta proporcionan contenido filtrado previamente si se han especificado filtros de tabla de consulta primaria, filtros de autorización o filtros de tabla de consulta en la definición de tabla de consulta.	Utilice la cláusula WHERE para filtrar el resultado de la consulta.
Cláusula WHERE y criterios de selección	La cláusula WHERE de la API de consulta no es necesaria en este formato en la API de tabla de consulta. Utilice la propiedad queryCondition en la API de tabla de consulta para realizar filtrado adicional. Los criterios de selección de la definición de tabla de consulta seleccionan una propiedad determinada de la tabla de consulta conectada. Esto se consigue, además de con el filtrado, mediante la cláusula WHERE de la API de consulta.	Los criterios de selección no están disponibles para la API de consulta. Sin embargo, los criterios de selección son similares a la parte de la cláusula WHERE que define, por ejemplo, el nombre o entorno local de QUERY_PROPERTY, TASK_CPROP o TASK_DESC. Por ejemplo, una cláusula WHERE de `QUERY_PROPERTY.NAME='xyz'` equivale a especificar `NAME='xyz'` como un criterio de selección en la definición de tabla de consulta para la tabla de consulta conectada QUERY_PROPERTY.

Característica	API de tabla de consulta	API de consulta
Elementos de trabajo y autorización	Utilice la tabla de consulta WORK_ITEM para acceder a los elementos de trabajo. Puede personalizar la utilización de los elementos de trabajo en la definición de tabla de consulta cuando se desarrolla la tabla de consulta y en la API de tabla de consulta, utilizando el objeto AuthorizationOptions o el objeto AdminAuthorizationOptions. Por ejemplo, para excluir elementos de trabajo Todos al consultar la tabla de consulta TASK, especifique una propiedad de queryCondition `WI.EVERYBODY=0` o especifique `setUseEverbody(Boolean.FALSE)` en la propiedad AuthorizationOptions.	Utilice la vista WORK_ITEM para acceder a los elementos de trabajo. Se consideran los cuatro tipos de elementos de trabajo para el resultado de la consulta: elementos de trabajo Todos, Persona, Grupo y Heredado. Para filtrar los elementos de trabajo para un tipo de elemento de trabajo específico, personalice la cláusula WHERE. Por ejemplo, para excluir los elementos de trabajo Todos, especifique `WORK_ITEM.EVERYBODY=0` en la cláusula WHERE.
Parámetros	Puede utilizar parámetros en filtros y criterios de selección para tablas de consulta compuestas.	Los parámetros no están disponibles para la API de consulta a menos que se utilicen consultas almacenadas.
Consultas almacenadas y tablas de consulta	La diferencia entre una consulta almacenada y una tabla de consulta es que las consultas almacenadas se definen para una consulta determinada, mientras que una tabla de consulta se define para un conjunto de consultas determinado. Por ejemplo, la definición de tabla de consulta no permite la especificación de una cláusula order-by ya que esta información normalmente está disponible sólo cuando se ejecuta la consulta.	Puede utilizar consultas almacenadas para ejecutar una consulta que contenga un conjunto de opciones predefinido.
Vistas materializadas	Las vistas materializadas no están disponibles para la API de tabla de consulta.	Las vistas materializadas utilizan tecnologías de base de datos para proporcionar mejoras de rendimiento para las consultas.
Tablas personalizadas	Las tablas de consulta suplementarias ofrecen la misma funcionalidad que las tablas personalizadas.	Las tablas personalizadas se utilizan para incluir datos en consultas externos al esquema de base de datos de Business Process Choreographer.
queryAll y opciones de autorización	La funcionalidad queryAll la proporciona el objeto AdminAuthorizationOptions, que se puede pasar a la API de tabla de consulta en lugar del objeto AuthorizationOptions. El emisor de la llamada debe estar en BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor o TaskSystemMonitor.	Método queryAll que pueden utilizar los usuarios que tienen el rol BPESystemAdministrator de Java EE para devolver todos los objetos en el resultado de consulta sin restricciones de elementos de trabajo para un usuario o grupo determinado.
Internacionalización	Para atributos de tablas de consulta y para la tabla de consulta, cuando se utilizan tablas de consulta hay disponibles descripciones y nombres de visualización localizados.	Se devuelven los nombres de las columnas de las vistas seleccionadas, como aparecen en la base de datos o como se especifican en la cláusula select.

Tablas de consulta en Business Process Choreographer

Las tablas de consulta dan soporte a consultas de lista de tareas y procesos en datos contenidos en el esquema de base de datos de Business Process Choreographer. Esto incluye datos de tareas de usuario y datos de procesos empresariales gestionados por Business Process Choreographer, así como datos empresariales externos. Las tablas de consulta proporcionan una abstracción de los datos de Business Process Choreographer que pueden utilizar las aplicaciones cliente. De esta forma, las aplicaciones cliente pasan a ser independientes de la propia implementación de la tabla de consulta. Las definiciones de tabla de consulta se despliegan en contenedores de Business Process Choreographer y son accesibles utilizando la API de tabla de consulta.

Existen tres tipos de tablas de consulta:

- Tablas de consulta predefinidas

- Tablas de consulta suplementarias
- Tablas de consulta compuestas

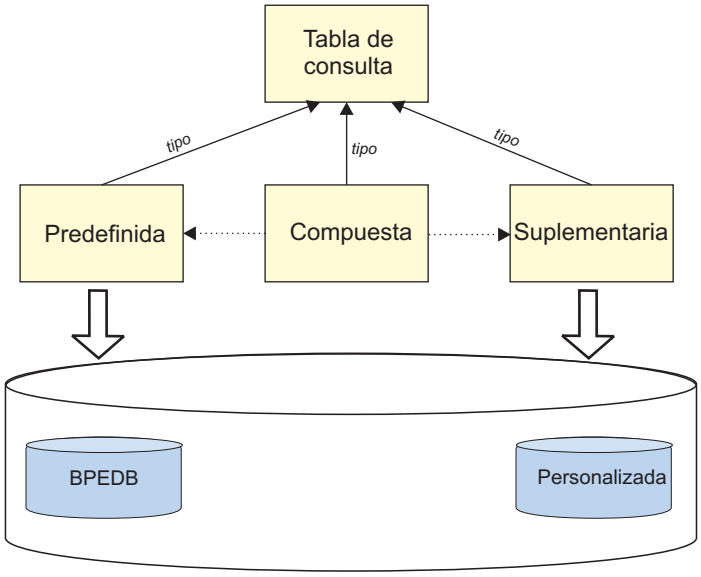


Figura 72. Tablas de consulta en Business Process Choreographer

Las tablas de consulta se representan utilizando modelos similares en el tiempo de ejecución de tabla de consultas y puede utilizar la API de tabla de consultas para consultarlas. Mientras que las tablas de consulta predefinidas y suplementarias apuntan directamente a tablas o vistas de la base de datos, las tablas de consulta compuestas componen partes de estos datos, lo que se representa en una única tabla de consulta.

Las tablas de consulta amplían las vistas de base de datos predefinidas y las interfaces de consulta existentes de Business Process Choreographer y:

- Están optimizadas para ejecutar consultas de lista de tareas y procesos, utilizando patrones de acceso optimizados de rendimiento.
- Simplifican y unifican el acceso a la información necesaria.
- Permiten la configuración precisa de las opciones de autorización y filtro.

Puede personalizar las tablas de consulta, por ejemplo, puede configurar una tabla de consulta para que sólo contenga las tareas y las instancias de proceso que son pertinentes en un caso determinado. También se recomienda utilizar tablas de consulta cuando el rendimiento sea una cuestión importante, por ejemplo con consultas de listas de procesos y listas de tareas de gran volumen.

Se proporciona Query Table Builder como un plug-in de Eclipse para:

- Desarrollar tablas de consulta compuestas y suplementarias
- Importar y exportar definiciones de tabla de consulta en formato XML

Puede descargar Query Table Builder en el sitio de SupportPacs de WebSphere Business Process Management. Busque la sección PA71 WebSphere Process Server - Query Table Builder. Para acceder al enlace, consulte la sección de referencias relacionadas de este tema.

Tablas de consulta predefinidas

Las tablas de consulta predefinidas proporcionan acceso a los datos de la base de datos de Business Process Choreographer. Son la representación de las tablas de consulta de las vistas de base de datos de Business Process Choreographer predefinidas correspondientes, como por ejemplo la vista TASK o la vista PROCESS_INSTANCE. Estas tablas de consulta predefinidas mejoran la funcionalidad y el rendimiento de las vistas de base de datos predefinidas porque están optimizadas para ejecutar consultas de lista de tareas y procesos.

Las tablas de consulta predefinidas se pueden consultar directamente utilizando la API de tabla de consulta. Cuando accede a las tablas utilizando la API de tabla de consulta, dispone de más opciones para la configuración que cuando utiliza la API de consulta.

Propiedades

Las tablas de consulta predefinidas tienen las propiedades siguientes:

Tabla 25. Propiedades de tablas de consulta predefinidas

Propiedad	Descripción
Nombre	El nombre de tabla de consulta es el nombre de una de las vistas de base de datos predefinidas, en mayúsculas, por ejemplo TASK.
Atributos	Los atributos de las tablas de consulta predefinidas definen la información que estará disponible para las consultas. Estos atributos son los nombres de columnas, en mayúsculas, especificados por las vistas de base de datos predefinida. Los atributos se definen con un nombre y un tipo. El tipo es uno de los siguientes: • **Booleano**: un valor booleano • **Decimal**: un número de punto flotante • **ID**: un ID de objeto, por ejemplo TKIID de la tabla de consulta TASK • **Número**: un entero, corto o largo • **Serie**: una serie • **Indicación de fecha y hora**: una indicación de fecha y hora

Tabla 25. Propiedades de tablas de consulta predefinidas (continuación)

Propiedad	Descripción
Autorización	Las tablas de consulta predefinidas utilizan autorización basada en instancia o autorización basada en rol. • Las tablas de consulta predefinidas con datos de instancia requieren autorización basada en instancia. Esto significa que sólo se devolverán los objetos con un elemento de trabajo para el usuario que realiza la consulta. Sin embargo, utilizando el objeto AdminAuthorizationOptions, esta verificación se puede reducir a una verificación de la existencia de un elemento de trabajo de cualquier usuario. El usuario debe tener el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager, para esas consultas. • Las tablas de consulta predefinidas con datos de plantilla necesitan autorización basada en rol, lo que significa que sólo pueden acceder al contenido de esas tablas de consulta los usuarios en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager.

Tablas de consulta predefinidas con datos de instancia

En la tabla siguiente se muestran las tablas de consulta predefinidas que contienen datos de instancia. Estas tablas de consulta:

- Se pueden utilizar como la consulta primaria de una tabla de consulta compuesta.
- Utilizan autorización basada en instancia si la consulta se realiza directamente. Esto se consigue con una unión (SQL-) con la vista que almacena la información de autorización, es decir, la tabla de consulta o vista WORK_ITEM predefinida.
- Contienen datos de instancia, por ejemplo los datos de instancias de tareas o de instancias de procesos.

Tabla 26. Tablas de consulta predefinidas que contienen datos de instancias

Datos de instancia	Nombre de tabla de consulta
Información sobre actividades de una instancia de proceso.	ACTIVITY
	ACTIVITY_ATTRIBUTE
	ACTIVITY_SERVICE
Información sobre escaladas que pertenecen a tareas de usuario.	ESCALATION
	ESCALATION_CPROP
	ESCALATION_DESC
Información sobre instancias de procesos.	PROCESS_ATTRIBUTE
	PROCESS_INSTANCE
	QUERY_PROPERTY
Información sobre tareas de usuario.	TASK
	TASK_CPROP
	TASK_DESC

La tabla de consulta WORK_ITEM también contiene datos de instancia, pero no está disponible como la tabla de consulta primaria ni como una tabla de consulta conectada. La información de elementos de trabajo está disponible implícitamente al consultar las tablas de consulta que utilizan autorización basada en instancia. Es decir, los atributos de la tabla de consulta WORK_ITEM se pueden utilizar al consultar una tabla de consulta con autorización basada en instancia, incluso aunque los atributos no los especifique explícitamente la tabla de consulta.

Tablas de consulta predefinidas con datos de plantilla

Las tablas de consulta predefinidas con datos de plantilla requieren autorización basada en rol. Sólo las pueden consultar los administradores utilizando el objeto AdminAuthorizationOptions.

En la tabla siguiente se muestran las tablas de consulta predefinidas que contienen datos de plantilla. Estas tablas de consulta:

- Se pueden utilizar como la tabla de consulta primaria de una tabla de consulta compuesta.
- Utilizan autorización basada en rol si la consulta se realiza directamente. Esto significa que el emisor de la llamada que utiliza el método de consulta de API debe estar en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager, y se debe utilizar AdminAuthorizationOptions.
- Contienen datos de plantilla, por ejemplo, los datos de plantilla de plantillas de tareas o de plantillas de procesos.

Tabla 27. Tablas de consulta predefinidas que contienen datos de plantilla

Datos de plantilla	**Nombre de tabla de consulta**
Información sobre componentes de la aplicación.	APPLICATION_COMP
Información sobre plantillas de escaladas.	ESC_TEMPL
	ESC_TEMPL_CPROP
	ESC_TEMPL_DESC
Información sobre plantillas de procesos.	PROCESS_TEMPLATE
	PROCESS_TEMPL_ATTR
Información sobre plantillas de tareas.	TASK_TEMPL
	TASK_TEMPL_CPROP
	TASK_TEMPL_DESC

Conceptos relacionados

"Tablas de consulta suplementarias"
Las tablas de consulta suplementarias en Business Process Choreographer exponen a la API de tabla de consulta datos empresariales que no gestiona Business Process Choreographer. Con tablas de consulta suplementarias, estos datos externos se pueden utilizar con datos de tablas de consulta predefinidas al recuperar información de instancias de procesos empresariales o información de tareas de usuario.

"Tablas de consulta compuestas" en la página 326
Las tablas de consulta compuestas de Business Process Choreographer no tienen una representación específica de los datos de la base de datos; constan de una combinación de datos de tablas de consulta predefinidas y suplementarias relacionadas. Utilice una tabla de consulta compuesta para recuperar la información para una lista de instancias de procesos o una lista de tareas, como por ejemplo Mis tareas a realizar.

"Desarrollo de tablas de consulta" en la página 334
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante Query Table Builder. Las tablas de consulta predefinidas no se pueden desarrollar ni desplegar. Están disponibles cuando se instala Business Process Choreographer y proporcionan una vista simple de los artefactos del esquema de base de datos de Business Process Choreographer.

"Consultas de tabla de consulta" en la página 354
Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta, que está disponible en la API de REST y de EJB de Business Flow Manager.

"Autorización para tablas de consulta" en la página 343
Puede utilizar la autorización basada en instancia, la autorización basada en rol o ninguna autorización al ejecutar consultas en tablas de consulta.

Tablas de consulta suplementarias

Las tablas de consulta suplementarias en Business Process Choreographer exponen a la API de tabla de consulta datos empresariales que no gestiona Business Process Choreographer. Con tablas de consulta suplementarias, estos datos externos se pueden utilizar con datos de tablas de consulta predefinidas al recuperar información de instancias de procesos empresariales o información de tareas de usuario.

Las tablas de consulta suplementarias se relacionan con tablas de base de datos o vistas de base de datos en la base de datos de Business Process Choreographer. Son tablas de consulta que contienen datos empresariales mantenidos por las aplicaciones de cliente. Las tablas de consultas suplementarias proporcionan información en una tabla de consulta compuesta además de la información contenida en una tabla de consulta predefinida.

Las tablas de consulta suplementarias tienen las propiedades siguientes:

Tabla 28. Propiedades de las tablas de consulta suplementarias

Propiedad	Descripción
Nombre	El nombre de tabla de consulta debe ser exclusivo en una instalación de Business Process Choreographer. Cuando se ejecuta la consulta, este nombre se utiliza para identificar la tabla de consulta que se consulta. Una tabla de consulta se identifica de forma exclusiva utilizando su nombre, que se define como *prefijo.nombre*. La longitud máxima de *prefijo.nombre* es de 28 caracteres. El prefijo debe ser diferente del prefijo reservado `'IBM'`, por ejemplo `'COMPANY.BUS_DATA'`. No utilice un dígito al final del nombre de tabla. Si una tabla se utiliza varias veces dentro de una consulta, el nombre de la tabla se amplía con un número de 0 a 9. Por ejemplo CUSTOM_VIEW0, CUSTOM_VIEW1, etc. Si ya hay un dígito al final del nombre de tabla, Business Process Choreographer elimina este dígito, lo que produce una excepción QueryUnknownTableException.
Nombre de la base de datos	Nombre de la tabla o vista relacionada en la base de datos. Sólo se pueden utilizar letras en mayúsculas.
Esquema de base de datos	Esquema de la tabla o vista relacionada en la base de datos. Sólo se pueden utilizar letras en mayúsculas. El esquema de base de datos debe ser distinto del esquema de base de datos de la base de datos de Business Process Choreographer. Sin embargo, la tabla o vista debe ser accesible con el mismo origen de datos JDBC que se utiliza para acceder a la base de datos de Business Process Choreographer.
Atributos	Los atributos de tablas de consulta suplementarias definen la información que estará disponible para las consultas. Estos atributos deben coincidir con el nombre relacionado de las columnas en la tabla o vista de base de datos relacionada. Los atributos se definen con un nombre y un tipo. El nombre se define en mayúsculas. El tipo es uno de los siguientes: • **Booleano**: un valor booleano • **Decimal**: un número de punto flotante • **ID**: un ID de objeto de 16 bytes de longitud, como por ejemplo TKIID de la tabla de consulta TASK • **Número**: un entero, corto o largo • **Serie**: una serie • **Indicación de fecha y hora**: una indicación de fecha y hora
Unión	Las uniones se deben definir en tablas de consulta suplementarias si están conectadas en tablas de consulta compuestas. Una unión define los atributos que se utilizan para correlacionar información de la tabla de consulta suplementaria con la información de la tabla de consulta primaria. Cuando se define una unión, el atributo de origen y el atributo de destino deben ser del mismo tipo.
Autorización	No se especifica ninguna autorización para tablas de consulta suplementarias; por lo tanto, todos los usuarios autenticados pueden ver el contenido.

Conceptos relacionados

"Tablas de consulta predefinidas" en la página 321
Las tablas de consulta predefinidas proporcionan acceso a los datos de la base de datos de Business Process Choreographer. Son la representación de las tablas de consulta de las vistas de base de datos de Business Process Choreographer predefinidas correspondientes, como por ejemplo la vista TASK o la vista PROCESS_INSTANCE. Estas tablas de consulta predefinidas mejoran la funcionalidad y el rendimiento de las vistas de base de datos predefinidas porque están optimizadas para ejecutar consultas de lista de tareas y procesos.

"Tablas de consulta compuestas"
Las tablas de consulta compuestas de Business Process Choreographer no tienen una representación específica de los datos de la base de datos; constan de una combinación de datos de tablas de consulta predefinidas y suplementarias relacionadas. Utilice una tabla de consulta compuesta para recuperar la información para una lista de instancias de procesos o una lista de tareas, como por ejemplo Mis tareas a realizar.

"Desarrollo de tablas de consulta" en la página 334
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante Query Table Builder. Las tablas de consulta predefinidas no se pueden desarrollar ni desplegar. Están disponibles cuando se instala Business Process Choreographer y proporcionan una vista simple de los artefactos del esquema de base de datos de Business Process Choreographer.

"Consultas de tabla de consulta" en la página 354
Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta, que está disponible en la API de REST y de EJB de Business Flow Manager.

"Autorización para tablas de consulta" en la página 343
Puede utilizar la autorización basada en instancia, la autorización basada en rol o ninguna autorización al ejecutar consultas en tablas de consulta.

Tablas de consulta compuestas

Las tablas de consulta compuestas de Business Process Choreographer no tienen una representación específica de los datos de la base de datos; constan de una combinación de datos de tablas de consulta predefinidas y suplementarias relacionadas. Utilice una tabla de consulta compuesta para recuperar la información para una lista de instancias de procesos o una lista de tareas, como por ejemplo Mis tareas a realizar.

Las tablas de consulta compuestas están diseñadas por desarrolladores de clientes y permiten una configuración precisa de filtros y de opciones de autorización para el acceso a datos optimizado cuando se ejecuta la consulta. Se realizan con SQL, que se ha optimizado para las consultas de lista de tareas y procesos.

Se recomienda utilizar tablas de consulta compuestas en escenarios de producción en lugar de las API de consulta de Business Process Choreographer estándares, porque las tablas de consulta compuestas proporcionan una abstracción de la implementación real de la consulta y de este modo permiten optimizaciones de consulta.

Además, en el tiempo de ejecución puede cambiar las tablas de consulta compuestas sin volver a desplegar el cliente que accede a la tabla de consulta.

En la figura siguiente se proporciona una visión general del contenido de las tablas de consulta compuestas:

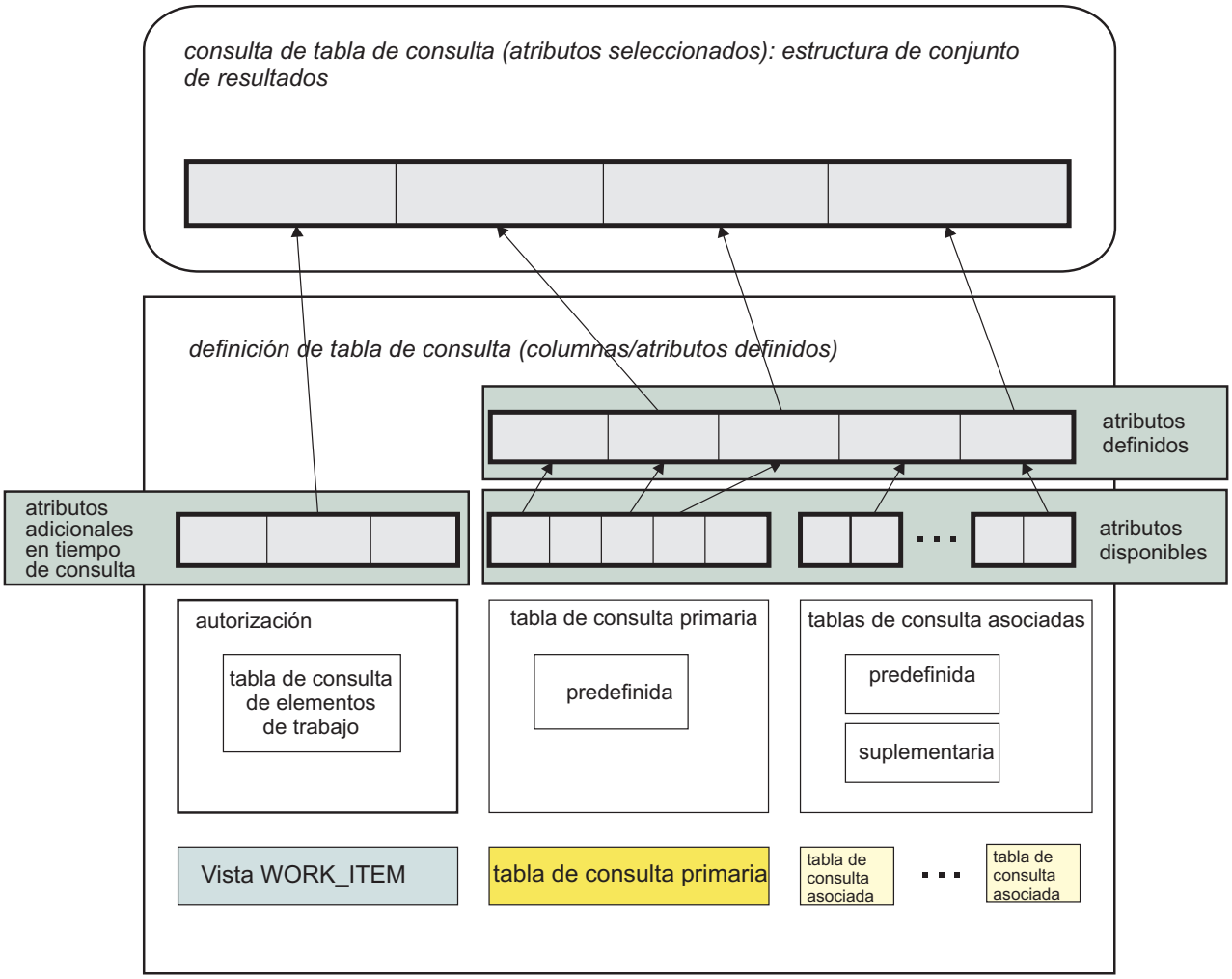


Figura 73. Contenido de tabla de consulta compuesta

Todas las tablas de consulta compuestas se definen con una tabla de consulta primaria y cero o más tablas de consulta conectadas.

Tablas de consulta primarias:

- Constituyen la información principal contenida en una tabla de consulta compuesta.
- Deben ser una de las tablas de consulta predefinidas.
- Identifican de forma exclusiva cada uno de los objetos de la tabla de consulta compuesta mediante la clave primaria. Por ejemplo, para la tabla de consulta predefinida TASK, es el ID de tarea TKIID.
- Autorizan el contenido de una tabla de consulta utilizando elementos de trabajo que están contenidos en la tabla de consulta WORK_ITEM, si se utiliza autorización basada en instancia.
- Determinan la lista de objetos que se devuelven como filas de una tabla cuando se consulta la tabla de consulta compuesta.

Tablas de consulta conectadas:

- Pueden ser tablas de consulta predefinidas y tablas de consulta suplementarias, que ya están desplegadas en el sistema.
- Están disponibles para proporcionar información adicional a la información que proporciona la tabla de consulta primaria. Por ejemplo, si TASK es la tabla de consulta primaria, la descripción de la tarea proporcionada en la tabla de consulta TASK_DESC se puede añadir al contenido de la tabla de consulta compuesta.

Normalmente, la tabla de consulta primaria se selecciona en función de la finalidad de la tabla de consulta compuesta.

- Si la tabla de consulta compuesta describe una lista de tareas, la tabla de consulta TASK es la tabla de consulta primaria.
- Si la tabla de consulta compuesta describe una lista de procesos, la tabla de consulta PROCESS_INSTANCE es la tabla de consulta primaria.
- Se recuperan listas de actividades utilizando la tabla de consulta primaria ACTIVITY.
- Se recuperan listas de escaladas de tareas de usuario utilizando la tabla de consulta primaria ESCALATION.

Relación entre tablas de consulta primarias y conectadas

La tabla de consulta conectada y la tabla de consulta primaria deben tener una relación de uno a uno o de uno a cero. Si se viola la relación de uno a uno o de cero a uno, se produce una excepción de tiempo de ejecución cuando se ejecuta la consulta.

Las tablas de consulta primarias y las tablas de consulta conectadas se correlacionan utilizando un atributo de unión que está definido en la tabla de consulta conectada. Este atributo de unión no se puede modificar para tablas de consulta predefinidas, ya que describe la relación entre los datos de las distintas tablas de consulta de Business Process Choreographer. El atributo de unión suele ser suficiente para mantener la relación de uno a uno o de uno a cerdo. Por ejemplo, el atributo CONTAINMENT_CTX_ID se utiliza en la tabla de consulta TASK para conectar la información de instancia de proceso relacionada identificada por el atributo PIID en la tabla de consulta PROCESS_INSTANCE.

Cuando existe una relación de uno a muchos, debe especificar un criterio adicional, conocido como criterio de selección, en Query Table Builder, al definir la tabla de consulta. Por ejemplo, podría ser `"LOCALE='en_US'"`. Una tarea puede tener varias descripciones que se identifican utilizando distintos entornos locales para una única tarea.

Ejemplo 1:

En la figura siguiente se proporciona una visualización de ejemplo de los criterios de selección que se especifican en las tablas de consulta conectadas:

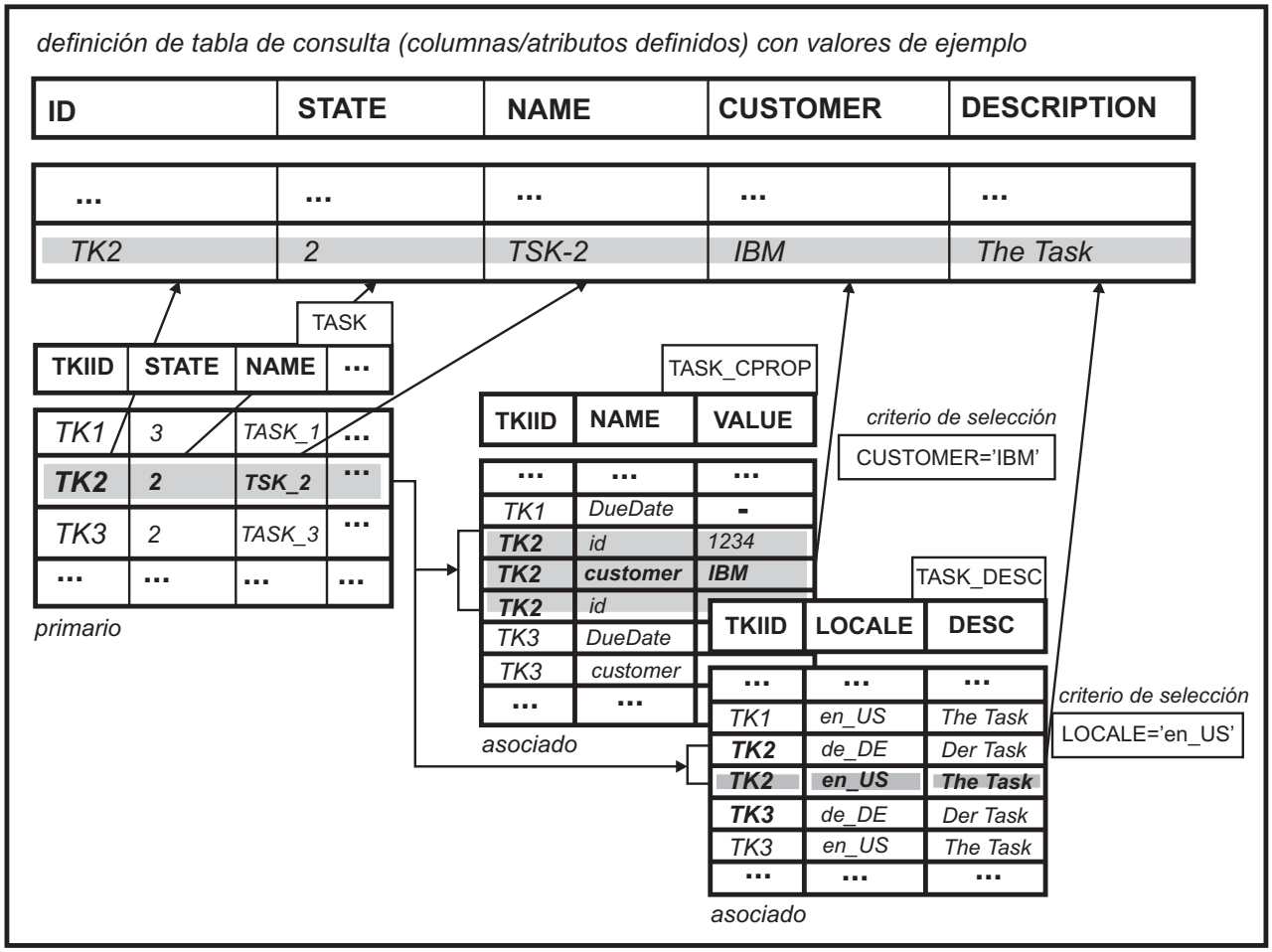


Figura 74. Tabla de consulta compuesta con criterios de selección

La tabla de consulta compuesta contiene los atributos ID, STATE, NAME, CUSTOMER y DESCRIPTION.

- ID, STATE y NAME los proporciona la tabla de consulta primaria TASK.
- CUSTOMER es una propiedad personalizada de TASK. Las propiedades personalizadas se almacenan en la tabla de consulta TASK_CPROP. Para una tarea determinada, una propiedad personalizada se identifica de forma exclusiva utilizando su nombre. Esto se refleja en el criterio de selección `"CUSTOMER='IBM'"`.
- DESCRIPTION es la descripción de la tarea, que se almacena en la tabla de consulta TASK_DESC. Para cada instancia de tarea, la descripción de tarea para una tarea determinada se identifica de forma exclusiva mediante su entorno local. Esto se refleja en el criterio de selección `"LOCALE='en_US'"`.

Ejemplo 2:

El foco de este ejemplo se encuentra en la relación entre las tablas de consulta primarias y conectadas, utilizando TASK como la tabla de consulta primaria y TASK_DESC como la tabla de consulta conectada. Cuando defina la tabla de consulta compuesta, deberá especificar el atributo LOCALE de la tabla de consulta TASK_DESC para asegurar que existe una relación de uno a uno o de uno a cero entre la tabla de consulta primaria y la tabla de consulta conectada. En la tabla se muestra contenido de ejemplo de una tabla de consulta compuesta con un criterio de selección válido para la tabla de consulta TASK_DESC conectada.

Tabla 29. Contenido válido de una tabla de consulta compuesta

Información de la tabla de consulta primaria TASK	**Información de la tabla de consulta conectada TASK_DESC**	
NAME	LOCALE	DESCRIPTION
task_one	en_US	This is a description.
task_two	en_US	This is a description.
...	...	...

En la tabla siguiente se muestra contenido hipotético no válido (en **negrita**) si el criterio de selección se establece incorrectamente, lo que significa que se viola la relación de uno a uno o de uno a cero.

Tabla 30. Contenido no válido de una tabla de consulta compuesta

Información de TASK (tabla de consulta primaria)	**Información de TASK_DESC (tabla de consulta primaria)**	
NAME	LOCALE	DESCRIPTION
task_one	en_US	This is a description.
task_one	**de_DE**	**Das ist eine Beschreibung.**
...	...	...

Propiedades

Las tablas de consulta compuestas tienen las propiedades siguientes:

Tabla 31. Propiedades de las tablas de consulta compuestas

Propiedad	**Descripción**
Nombre	El nombre de tabla de consulta debe ser exclusivo en una instalación de Business Process Choreographer. Cuando se ejecuta la consulta, este nombre de tabla de consulta se utiliza para identificar la tabla de consulta que se consulta. Una tabla de consulta se identifica de forma exclusiva utilizando su nombre, que se define como *prefijo.nombre* para tablas de consulta compuestas. La longitud máxima de *prefijo.nombre* es de 28 caracteres. El prefijo debe ser diferente del prefijo reservado 'IBM', por ejemplo 'COMPANY.TODO_TASK_LIST". No utilice un dígito al final del nombre de tabla. Si una tabla se utiliza varias veces dentro de una consulta, el nombre de la tabla se amplía con un número de 0 a 9. Por ejemplo CUSTOM_VIEW0, CUSTOM_VIEW1, etc. Si ya hay un dígito al final del nombre de tabla, Business Process Choreographer elimina este dígito, lo que produce una excepción QueryUnknownTableException.

Tabla 31. Propiedades de las tablas de consulta compuestas (continuación)

Propiedad	Descripción
Atributos	Los atributos de las tablas de consulta compuestas definen la información que estará disponible para las consultas. Los atributos se definen con un nombre, en mayúsculas. El tipo se hereda del atributo al que se hace referencia, que es uno de los siguientes: • **Booleano**: un valor booleano • **Decimal**: un número de punto flotante • **ID**: un ID de objeto, por ejemplo TKIID de la tabla de consulta TASK • **Número**: un entero, corto o largo • **Serie**: una serie • **Indicación de fecha y hora**: una indicación de fecha y hora Los atributos de las tablas de consulta compuestas se definen utilizando una referencia a atributos de la tabla de consulta primaria o las tablas de consulta conectadas. Los atributos de las tablas de consulta compuestas heredan los tipos y las constantes de los atributos a los que se hace referencia. Además de los atributos que forman parte de la definición de tabla de consulta, durante la ejecución se puede consultar la información de los elementos de trabajo. Esto es posible si la tabla de consulta primaria contiene datos de instancia, como por ejemplo TASK o PROCESS_INSTANCE, y si se utiliza la autorización basada en instancia en la tabla de consulta compuesta. Por ejemplo, se puede definir la consulta para devolver sólo tareas de usuario de las que el usuario es un propietario potencial.

Tabla 31. Propiedades de las tablas de consulta compuestas (continuación)

Propiedad	Descripción
Autorización	Cada tabla de consulta compuesta define si se utiliza autorización basada en instancia, basada en rol o ninguna cuando se ejecutan consultas en ella. Si se define autorización basada en instancia, sólo se devolverán los objetos con un elemento de trabajo para el usuario que realiza la consulta. Sin embargo, utilizando AdminAuthorizationOptions esta verificación se puede reducir a una verificación de la existencia de un elemento de trabajo de cualquier usuario. El usuario debe estar en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager, para esas consultas y se debe pasar AdminAuthorizationOptions a la API de tabla de consulta. Si se define la autorización basada en rol, el usuario debe estar en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager, para las consultas y se debe pasar AdminAuthorizationOptions a la API de tabla de consulta. Si no se define ninguna autorización, la consulta se ejecuta sin las comprobaciones de la existencia de elementos de trabajo de los objetos relacionados en la tabla de consulta. Todos los usuarios autenticados pueden ver el contenido de la tabla de consulta. Se puede definir autorización basada en instancia si la tabla de consulta primaria contiene datos de instancia; se puede definir autorización basada en rol si la tabla de consulta primaria contiene datos de plantilla. Se puede definir ninguna autorización en las tablas de consulta compuestas independientemente de la tabla de consulta primaria que se utilice.

Filtros

Se utilizan filtros para limitar el número de objetos, o filas, contenidos en una tabla de consulta compuesta.

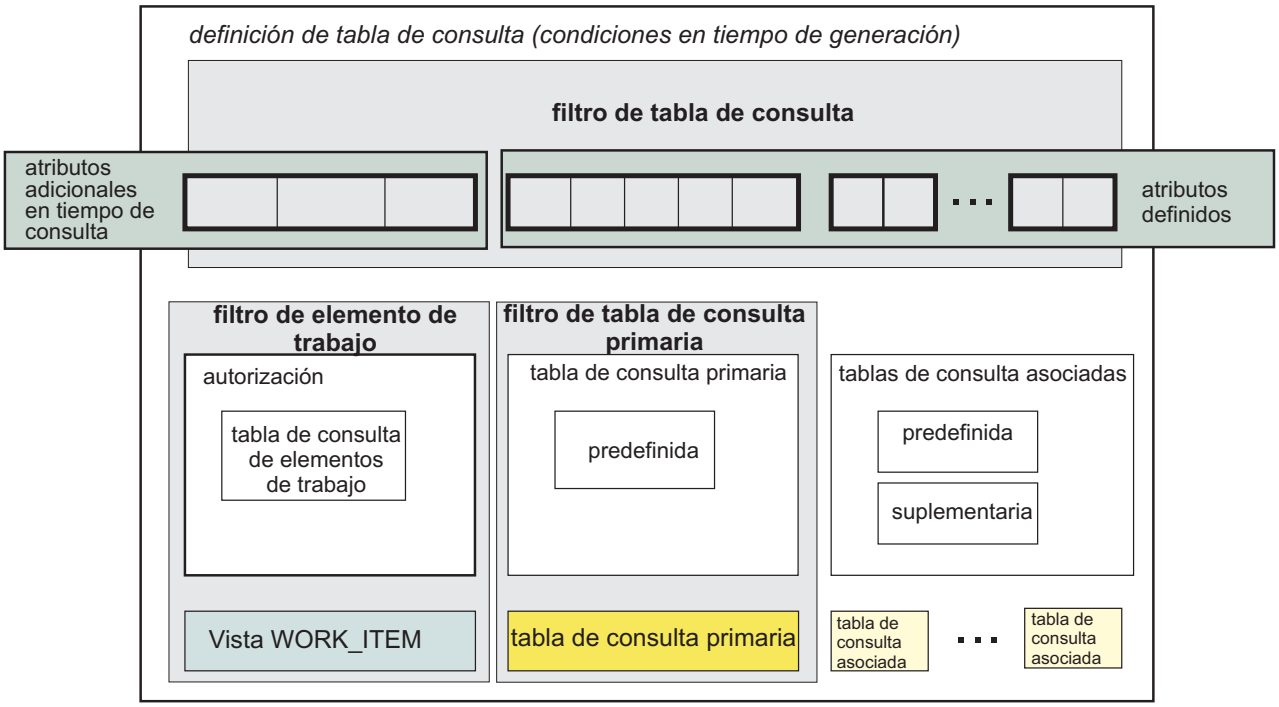


Figura 75. Filtros en tablas de consulta compuestas

Se pueden definir filtros en tablas de consulta compuestas durante el desarrollo de:

- La tabla de consulta primaria, como filtro de tabla de consulta primaria.
- La tabla de consulta WORK_ITEM disponible implícitamente que es responsable de la autorización si la tabla de consulta primaria contiene datos de instancia. Este filtro se denomina filtro de autorización y está disponible sólo si la tabla de consulta compuesta está configurada para utilizar autorización basada en instancia.
- La tabla de consulta compuesta, como filtro de tabla de consulta.

Se definen los filtros durante el desarrollo de la tabla de consulta. Por ejemplo, una tabla de consulta compuesta con la tabla de consulta primaria TASK se puede filtrar en tareas que están en estado preparado (`"STATE=STATE_READY"` como filtro de tabla de consulta primaria).

Autorización

La autorización para acceder al contenido de una tabla de consulta compuesta con una tabla de consulta primaria es similar a la autorización que se utiliza para acceder a la tabla de consulta primaria. La diferencia es que las tablas de consulta compuestas se pueden configurar para que sean más restrictivas.

- Si se configura la utilización de autorización basada en instancia, los datos contenidos en la tabla de consulta primaria se verifican para ver si existen elementos de trabajo en la tabla de consulta WORK_ITEM. Esta verificación se realiza respecto a la tabla de consulta primaria. Para la verificación se utilizan elementos de trabajo Todos, de persona, de grupo y heredados, en función de la configuración de la tabla de consulta compuesta. Si se especifican elementos de trabajo Heredados, los objetos que tienen una instancia de proceso como padre, como por ejemplo una tarea de usuario participante, con un elemento de trabajo

Todos, Persona o Grupo configurado, están contenidos en la tabla de consulta compuesta. Normalmente, los elementos de trabajo de tipo Heredado resultan útiles solo a los administradores.

- Las tablas de consulta compuestas con una tabla de consulta primaria que contiene datos de plantilla no se deben establecer para utilizar autorización basada en instancia. Si se utiliza la autorización basada en rol, las consultas sólo las pueden ejecutar usuarios que estén en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager, y se debe utilizar el objeto AdminAuthorizationOptions.

Conceptos relacionados

"Tablas de consulta predefinidas" en la página 321
Las tablas de consulta predefinidas proporcionan acceso a los datos de la base de datos de Business Process Choreographer. Son la representación de las tablas de consulta de las vistas de base de datos de Business Process Choreographer predefinidas correspondientes, como por ejemplo la vista TASK o la vista PROCESS_INSTANCE. Estas tablas de consulta predefinidas mejoran la funcionalidad y el rendimiento de las vistas de base de datos predefinidas porque están optimizadas para ejecutar consultas de lista de tareas y procesos.

"Tablas de consulta suplementarias" en la página 324
Las tablas de consulta suplementarias en Business Process Choreographer exponen a la API de tabla de consulta datos empresariales que no gestiona Business Process Choreographer. Con tablas de consulta suplementarias, estos datos externos se pueden utilizar con datos de tablas de consulta predefinidas al recuperar información de instancias de procesos empresariales o información de tareas de usuario.

Desarrollo de tablas de consulta

Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante Query Table Builder. Las tablas de consulta predefinidas no se pueden desarrollar ni desplegar. Están disponibles cuando se instala Business Process Choreographer y proporcionan una vista simple de los artefactos del esquema de base de datos de Business Process Choreographer.

Query Table Builder está disponible como un plug-in de Eclipse y se puede descargar del sitio de SupportPacs de WebSphere Business Process Management. Busque la sección PA71 WebSphere Process Server - Query Table Builder. Para acceder al enlace, consulte la sección de referencias relacionadas de este tema.

Las tablas de consulta afectan la manera en que las aplicaciones de desarrollan y despliegan. En los pasos siguientes se describen los roles implicados al diseñar y desarrollar una aplicación de Business Process Choreographer que utilice tablas de consulta.

Tabla 32. Pasos del desarrollo de una tabla de consulta

Paso	Quién	Descripción
1. Análisis	Analista empresarial, desarrollador de cliente	Analizar qué tablas de consulta son necesarias en la aplicación cliente. Las preguntas que se deben responder son las siguientes: • ¿Cuántas listas de tareas o procesos se proporcionan al usuario? ¿Existen listas de tareas o procesos que puedan compartir la misma tabla de consulta? • ¿Qué clase de autorización se utiliza? ¿Autorización basada en instancia, autorización basada en rol o ninguna? • ¿Existen otras tablas de consulta ya definidas en el sistema que se puedan reutilizar? • ¿Deben las tablas de consulta proporcionar el contenido en varios idiomas? Si es así, los criterios de selección de las tablas de consulta conectadas deben ser `LOCALE=$LOCALE`.
2. Desarrollo de las tablas de consulta	Desarrollador de cliente, analista empresarial	Desarrollar las tablas de consulta que se utilizan en la aplicación cliente. Intente especificar la definición de las tablas de consulta de forma que se consiga el mejor rendimiento con las consultas de tabla de consulta.
3. Despliegue de las tablas de consulta	Administrador	Para poder utilizarlas, las tablas de consulta se deben desplegar en el entorno de ejecución. Este paso se realiza utilizando el mandato wsadmin de manageQueryTable.py.
4. Consultas de las tablas de consulta	Desarrollador de cliente	La ejecución de consultas contra tablas de consulta es el último paso del desarrollo de las tablas de consulta. El desarrollador de cliente debe conocer el nombre de la tabla de consulta y sus atributos.

A continuación aparece un código de ejemplo que utiliza la API de tabla de consulta para consultar una tabla de consulta. Los ejemplos 1 y 2 se proporcionan para consultar la tabla de consulta predefinida TASK por motivos de simplicidad. Los ejemplos 3 y 4 consultan una tabla de consulta compuesta, que se supone que se ha desplegado en el sistema. En el desarrollo de aplicaciones, deberá utilizar tablas de consulta compuestas en lugar de consultar directamente las tablas de consulta predefinidas.

Ejemplo 1

```
// obtener el contexto de denominación y buscar el inicio de EJB
// de Business Flow Manager; tenga en cuenta que el inicio de EJB de Business Flow
// Manager se debe almacenar en la antememoria por razones de
// rendimiento; además, se supone que hay una referencia de Enterprise JavaBeans
// en el Enterprise JavaBeans local de Business Flow Manager
Context ctx = new InitialContext();
LocalBusinessFlowManagerHome home =
 (LocalBusinessFlowManagerHome)
 ctx.lookup("java:comp/env/ejb/BFM");
```

```
// si se utiliza Enterprise JavaBeans de Human Task Manager, realice lo siguiente:
// LocalHumanTaskManagerHome home =
//   (LocalHumanTaskManagerHome) ctx.lookup("java:comp/env/ejb/HTM");
// suponiendo que se ha definido una referencia de EJB al
// EJB de Human Task Manager

// crear el módulo de programa del lado del cliente de Business Flow Manager
LocalBusinessFlowManager bfm = home.create();
// si se utiliza el EJB de Human Task Manager , realice lo siguiente:
// LocalHumanTaskManager htm = home.create();
// tenga en cuenta que Enterprise JavaBeans de Human Task Manager proporciona los
// mismos métodos que los Enterprise JavaBeans de Business Flow Manager
// ************************************************
// ****************** ejemplo 1 *******************
// ************************************************

// ejecutar una consulta contra la tabla de consulta predefinida TASK;
// esto está relacionado con una lista de tareas de tipo Tareas a realizar simple
EntityResultSet ers = null;
ers = bfm.queryEntities("TASK", null, null, null);

// imprimir el resultado en STDOUT
EntityInfo entityInfo = ers.getEntityInfo();
List attList = entityInfo.getAttributeInfo();
int attSize = attList.size();

Iterator iter = ers.getEntities().iterator();
while( iter.hasNext() ) {
 System.out.print("Entity: ");
 Entity entity = (Entity) iter.next();
 for (int i = attSize - 1; i >= 0; i--) {
  AttributeInfo ai = (AttributeInfo) attList.get(i);
  System.out.print(
    entity.getAttributeValue(ai.getName()));
 }
 System.out.println();
}
```

Ejemplo 2

```
// ************************************************
// ****************** ejemplo 2 *******************
// ************************************************

// el mismo ejemplo que el ejemplo 1, pero usando el enfoque de consulta
// basado en fila
RowResultSet rrs = null;
rrs = bfm.queryRows("TASK", null, null, null);

attList = rrs.getAttributeInfo();
attSize = attList.size();

// imprimir el resultado en STDOUT
while (rrs.next()) {
 System.out.print("Row: ");
 for (int i = attSize - 1; i >= 0; i--) {
  AttributeInfo ai = (AttributeInfo) attList.get(i);
  System.out.print(
    rrs.getAttributeValue(ai.getName()));
 }
 System.out.println();
}
```

Ejemplo 3

```
   // ************************************************
   // ****************** ejemplo 3 *******************
   // ************************************************

   // ejecutar una consulta contra una tabla de consulta compuesta
   // que se haya desplegado antes en el sistema;
   // se supone que el nombre es COMPANY.TASK_LIST
   ers = bfm.queryEntities(
     "COMPANY.TASK_LIST", null, null, null);
^
   // imprimir el resultado en STDOUT ...
```

Ejemplo 4

```
   // ************************************************
   // ****************** ejemplo 4 *******************
   // ************************************************

   // consultar contra la misma tabla de consulta que en el ejemplo 3,
   // pero con opciones personalizadas
   FilterOptions fo = new FilterOptions();

   // devolver sólo objetos que estén en estado preparado
   fo.setQueryCondition("STATE=STATE_READY");

   // ordenar por el ID de objeto
   fo.setSortAttributes("ID");

   // limitar el número de entidades a 50
   fo.setThreshold(50);

   // obtener sólo un subconjunto de los atributos definidos
   // en la tabla de consulta
   fo.setSelectedAttributes("ID, STATE, DESCRIPTION");

   AuthorizationOptions ao = new AuthorizationOptions();

   // no devolver objetos que puedan ver todos
   // los usuarios
   ao.setEverybodyUsed(Boolean.FALSE);

   ers = bfm.queryEntities(
     "COMPANY.TASK_LIST", fo, ao, null);

   // imprimir el resultado en STDOUT ...
```

Conceptos relacionados

"Consultas de tabla de consulta" en la página 354
Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta, que está disponible en la API de REST y de EJB de Business Flow Manager.

"Filtros y criterios de selección de tablas de consulta"
Los filtros y los criterios de selección se definen durante el desarrollo de la tabla de consulta utilizando Query Table Builder, que utiliza una sintaxis similar a las cláusulas WHERE de SQL. Utilice estos filtros y criterios de selección claramente definidos para especificar las condiciones que se basan en los atributos de las tablas de consulta.

"Tablas de consulta predefinidas" en la página 321
Las tablas de consulta predefinidas proporcionan acceso a los datos de la base de datos de Business Process Choreographer. Son la representación de las tablas de consulta de las vistas de base de datos de Business Process Choreographer predefinidas correspondientes, como por ejemplo la vista TASK o la vista PROCESS_INSTANCE. Estas tablas de consulta predefinidas mejoran la funcionalidad y el rendimiento de las vistas de base de datos predefinidas porque están optimizadas para ejecutar consultas de lista de tareas y procesos.

"Tablas de consulta suplementarias" en la página 324
Las tablas de consulta suplementarias en Business Process Choreographer exponen a la API de tabla de consulta datos empresariales que no gestiona Business Process Choreographer. Con tablas de consulta suplementarias, estos datos externos se pueden utilizar con datos de tablas de consulta predefinidas al recuperar información de instancias de procesos empresariales o información de tareas de usuario.

"Tablas de consulta compuestas" en la página 326
Las tablas de consulta compuestas de Business Process Choreographer no tienen una representación específica de los datos de la base de datos; constan de una combinación de datos de tablas de consulta predefinidas y suplementarias relacionadas. Utilice una tabla de consulta compuesta para recuperar la información para una lista de instancias de procesos o una lista de tareas, como por ejemplo Mis tareas a realizar.

Filtros y criterios de selección de tablas de consulta

Los filtros y los criterios de selección se definen durante el desarrollo de la tabla de consulta utilizando Query Table Builder, que utiliza una sintaxis similar a las cláusulas WHERE de SQL. Utilice estos filtros y criterios de selección claramente definidos para especificar las condiciones que se basan en los atributos de las tablas de consulta.

Para obtener información sobre cómo instalar Query Table Builder, consulte el sitio de SupportPacs de WebSphere Business Process Management. Busque la sección PA71 WebSphere Process Server - Query Table Builder. Para acceder al enlace, consulte la sección de referencias relacionadas de este tema.

Atributos

Los atributos utilizados en una expresión hacen referencia a los atributos de tabla de consulta. En función de la ubicación de la expresión, están disponibles distintos atributos. Para el desarrollador de cliente, los filtros de consulta pasados a la API de tabla de consulta son la única ubicación donde se pueden utilizar expresiones. Para desarrolladores de tablas de consulta compuestas, existen otras varias ubicaciones donde se pueden utilizar expresiones. En la tabla siguiente se describen los atributos disponibles en las distintas ubicaciones.

Tabla 33. Atributos para expresiones de tabla de consulta

Dónde	Expresión	Atributos disponibles
API de tabla de consulta	Filtro de consulta	• Todos los atributos definidos en la tabla de consulta. • Si se utiliza autorización basada en instancia, todos los atributos definidos en las tablas de consulta WORK_ITEM, con el prefijo `'WI.'` . Ejemplos: • `STATE=STATE_READY`, si la tabla de consulta contiene un atributo STATE y si se ha definido una constante STATE_READY para este atributo • `STATE=STATE_READY AND WI.REASON=REASON_POTENTIAL_OWNER`, si la tabla de consulta contiene un atributo STATE y la tabla de consulta utiliza autorización basada en instancia
Tabla de consulta compuesta	Filtro de tabla de consulta	
	Filtro de tabla de consulta primaria	• Todos los atributos definidos para la tabla de consulta primaria. Ejemplo: • `STATE=STATE_READY`, si la tabla de consulta contiene un atributo STATE y se ha definido una constante STATE_READY para este atributo
	Filtro de autorización	• Todos los atributos definidos en la tabla de consulta predefinida WORK_ITEM, con el prefijo `'WI.'` . Ejemplo: • `WI.REASON=REASON_POTENTIAL_OWNER`
	Criterio de selección	• Todos los atributos definidos en la tabla de consulta conectada relacionada. Ejemplo: • `LOCALE='en_US'`, si la tabla de consulta conectada contiene un atributo LOCALE, como por ejemplo la tabla de consulta TASK_DESC

En la figura siguiente se muestran las diversas ubicaciones de filtros y criterios de selección en expresiones, y se incluyen ejemplos:

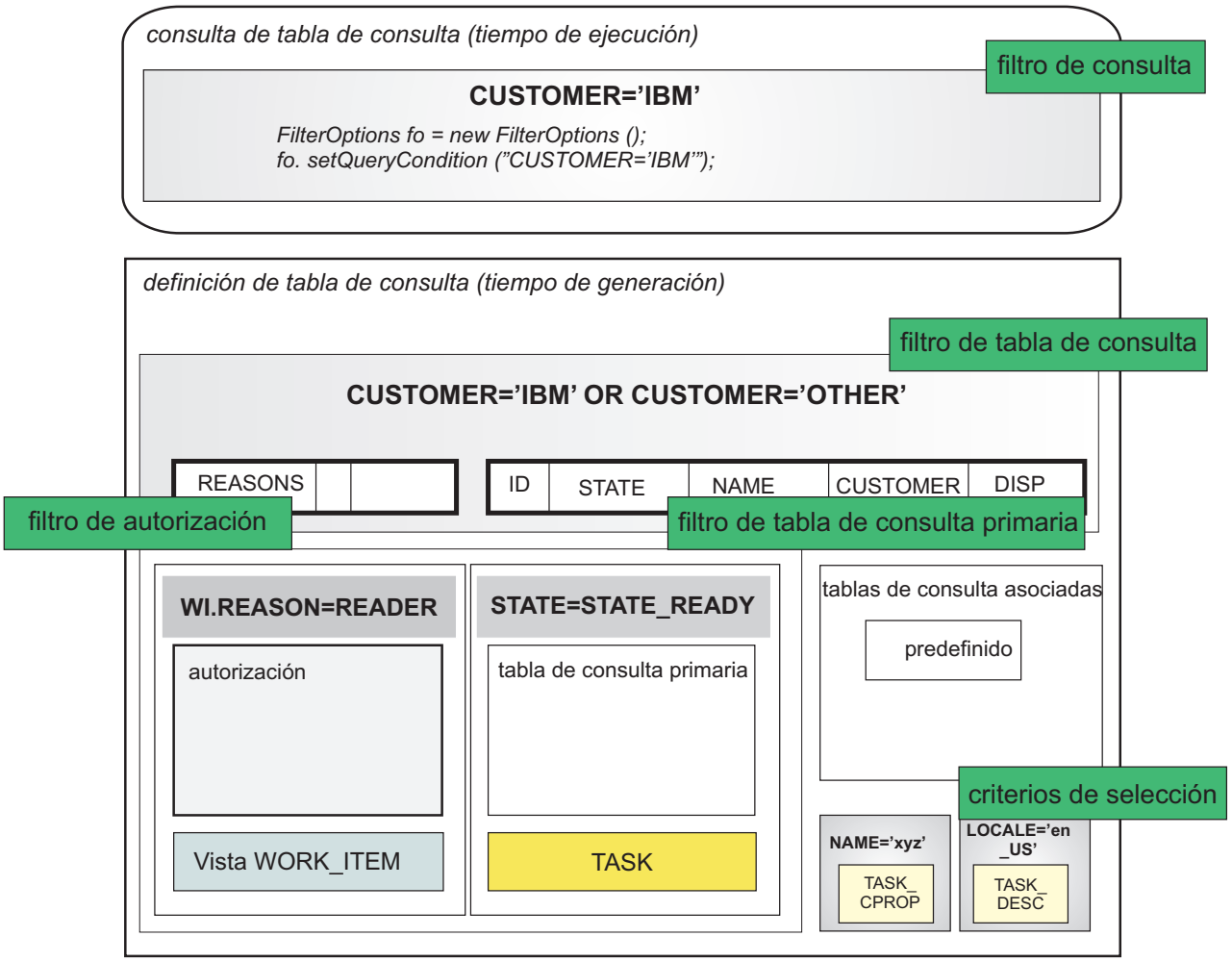


Figura 76. Filtros y criterios de selección en expresiones

Expresiones

Las expresiones tienen la sintaxis siguiente:

```
expression  ::=  atributo op_binario valor |
                 atributo op_unario |
                 atributo op_lista lista |
                 (expresión) |
                 expresión AND expresión |
                 expresión> OR expresión
```

Se aplican las reglas siguientes:

- `AND` tiene prioridad sobre `OR`. Las subexpresiones se conectan utilizando `AND` y `OR`.
- Se pueden utilizar delimitadores para agrupar expresiones y deben estar equilibrados.

Ejemplos:

- `STATE = STATE_READY`
- `NAME IS NOT NULL`
- `STATE IN (2, 5, STATE_FINISHED)`
- `((PRIORITY=1) OR (WI.REASON=2)) AND (STATE=2)`

Una expresión se ejecuta en un ámbito determinado que determina los atributos que son válidos para la expresión. Los criterios de selección, o los filtros de consulta, se ejecutan en el ámbito de la tabla de consulta en la que se ejecuta la consulta.

La tabla siguiente es para una consulta que se ejecuta en la tabla de consulta TASK predefinida:

```
'(STATE=STATE_READY AND WI.REASON=REASON_POTENTIAL_OWNER)
OR (WI.REASON=REASON_OWNER)'
```

Operadores binarios

Están disponibles los siguientes operadores binarios:

op_binario ::= = | < | > | <> | <= | >= | LIKE | NOT LIKE

Se aplican las reglas siguientes:

- El operando del lado izquierdo de un operador binario debe hacer referencia a un atributo de una tabla de consulta.
- El operando del lado derecho de un operador binario debe ser un valor literal, un valor de constante o un parámetro.
- Los operadores `LIKE` y `NOT LIKE` son los únicos válidos para atributos de tipo de atributo `STRING`.
- El operando del lado izquierdo y el operando del lado derecho deben ser tipos de atributo compatibles.
- Los parámetros de usuario deben ser compatibles con el tipo de atributo del atributo del lado izquierdo.

Ejemplos:

- `STATE > 2`
- `NAME LIKE 'start%'`
- `STATE <> PARAM(elEstado)`

Operadores unarios

Están disponibles los siguientes operadores unarios:

op_unario ::= IS NULL | IS NOT NULL

Se aplican las reglas siguientes:

- El operando del lado izquierdo de un operador binario debe hacer referencia a un atributo de una tabla de consulta. Los atributos válidos dependen de la ubicación del filtro o criterio de selección.
- Se comprueban todos los atributos para ver si tienen valores nulos, por ejemplo: `CUSTOMER IS NOT NULL`.

Ejemplo:

```
DESCRIPTION IS NOT NULL
```

Operadores de lista

Están disponibles los operadores de lista siguientes:

op_lista ::= IN | NOT IN

Se aplican las reglas siguientes:

- El lado derecho de un operador de lista no se debe sustituir por un parámetro de usuario.
- Se pueden utilizar parámetros de usuario dentro de la lista del operando del lado derecho.

Ejemplo:

```
STATE IN (STATE_READY, STATE_RUNNING, PARAM(st), 1)
```

Las listas se representan de la forma siguiente:

lista ::= *valor* [, *lista*]

Se aplican las reglas siguientes:

- El lado derecho de un operador de lista no se debe sustituir por un parámetro de usuario.
- Se pueden utilizar parámetros de usuario dentro de la lista del operando del lado derecho.

Ejemplos:

- `(2, 5, 8)`
- `(STATE_READY, STATE_CLAIMED)`

Valores

En expresiones, un valor es uno de los siguientes:

- **Constante**: un valor de constante, que se define para el atributo de una tabla de consulta predefinida. Por ejemplo, `STATE_READY` se define para el atributo STATE de la tabla de consulta TASK.
- **Literal**: cualquier valor codificado por el software.
- **Parámetro**: se sustituye un parámetro cuando la consulta se ejecuta con un valor específico.

Hay **constantes** disponibles para algunos atributos de tablas de consulta predefinidas. Para obtener información sobre las constantes que están disponibles en los atributos de las tablas de consulta predefinidas, consulte la información sobre las vistas predefinidas. Sólo las constantes que definen valores de entero se exponen con las tablas de consulta. Además, en lugar de constantes, se pueden utilizar valores literales relacionados o parámetros.

Ejemplos:

- `STATE_READY` en el atributo STATE de la tabla de consulta TASK se puede utilizar en un filtro para comprobar si la tarea se encuentra en estado preparado.
- `REASON_POTENTIAL_OWNER` en el atributo REASON de la tabla de consulta WORK_ITEM se puede utilizar en un filtro para comprobar si el usuario que ejecuta la consulta contra una tabla de consulta es un propietario potencial.
- El filtro de consulta `STATE=STATE_READY` es el mismo que `STATE=2`, si la consulta se ejecuta en la tabla de consulta TASK.

También se pueden utilizar **literales** en las expresiones. Se debe utilizar una sintaxis especial para las indicaciones de fecha y hora y los ID.

Ejemplos:

- `STATE=1`

- `NAME='elNombre'`
- `CREATED > TS ('2008-11-26 T12:00:00')`
- `TKTID=ID('_TKT:801a011e.9d57c52.ab886df6.1fcc0000')`

Los **parámetros** en las expresiones permiten la capacidad de dinamización de tablas de consulta compuestas. Existen parámetros de usuario y parámetros del sistema:

- Los parámetros de usuario se especifican utilizando PARAM (*nombre*). Este parámetro se debe proporcionar cuando se ejecuta la consulta. Se pasa como una instancia de la clase com.ibm.bpe.api.Parameter a la API de tabla de consulta.
- Los parámetros del sistema son parámetros que proporciona el entorno de ejecución de tabla de consulta, sin que se especifiquen cuando se ejecuta la consulta. Están disponibles los parámetros del sistema `$USER` y `$LOCALE`.
  - `$USER`, que es una serie, contiene el valor del usuario que ejecuta la consulta.
  - `$LOCALE`, que es una serie, contiene el valor del entorno local que se utiliza cuando se ejecuta la consulta. Un ejemplo de valor de `$LOCALE` es `'en_US'`.

Puede especificar un parámetro en los criterios de selección de un tabla de consulta conectada que seleccionada en un entorno local específico. Por ejemplo, si la tabla de consulta primaria es TASK en una tabla de consulta compuesta y una tabla de consulta conectada es TASK_DESC. Los siguientes son ejemplos de parámetros:

- `STATE=PARAM(elEstado)`
- `LOCALE=$LOCALE`
- `OWNER=$USER`

Conceptos relacionados

"Desarrollo de tablas de consulta" en la página 334
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante Query Table Builder. Las tablas de consulta predefinidas no se pueden desarrollar ni desplegar. Están disponibles cuando se instala Business Process Choreographer y proporcionan una vista simple de los artefactos del esquema de base de datos de Business Process Choreographer.

"Consultas de tabla de consulta" en la página 354
Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta, que está disponible en la API de REST y de EJB de Business Flow Manager.

Tareas relacionadas

"Creación de tablas de consulta para Business Process Choreographer Explorer" en la página 376
Puede utilizar tablas de consulta en lugar de la API query EJB para mejorar el rendimiento de Business Process Choreographer Explorer. Para crear las tablas de consulta, utilice el generador de tablas de consulta.

Autorización para tablas de consulta

Puede utilizar la autorización basada en instancia, la autorización basada en rol o ninguna autorización al ejecutar consultas en tablas de consulta.

El tipo de autorización se define en la tabla de consulta.

- La autorización basada en instancia indica que los objetos en la tabla de consulta se autorizan utilizando un elemento de trabajo. Esto se realiza verificando si existe un elemento de trabajo adecuado.

- La autorización basada en rol se basa en roles de Java EE. Indica que el emisor de la llamada que utiliza el método de consulta de API debe estar en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager, para ver el contenido de la tabla de consulta. Está disponible para tablas de consulta predefinidas con datos de plantilla y para tablas de consulta compuestas con una tabla de consulta primaria que contenga datos de plantilla. Los objetos de estas tablas de consulta no tienen elementos de trabajo relacionados.
- Cuando no se especifica ninguna autorización, todos los usuarios autenticados pueden ver todo el contenido de la tabla de consulta, una vez que se han aplicado los filtros.

El tipo de autorización en tablas de consulta predefinidas y el tipo de autorización que se puede configurar en tablas de consulta compuestas y suplementarias se describe brevemente en la tabla siguiente.

Tabla 34. Tipos de autorización para tablas de consulta

Tabla de consulta	**Autorización basada en instancia**	**Autorización basada en rol**	**Ninguna autorización**
Predefinida	Necesaria para tablas de consulta predefinidas con datos de instancia.	Necesaria para tablas de consulta predefinidas con datos de plantilla.	N/A
Compuesta	Se puede desactivar, lo que significa que no se utiliza ninguna autorización y que se alteran temporalmente las restricciones de seguridad. Es decir, cada usuario autenticado puede utilizar la tabla de consulta para recuperar datos, independientemente de si está autorizado para los objetos respectivos. Las tablas de consulta compuestas con una tabla de consulta primaria que contiene datos de plantilla no se deben establecer para utilizar autorización basada en instancia.	Se puede desactivar, por ejemplo para tablas de consulta compuestas con una tabla de consulta primaria que contiene datos de plantilla. Esto significa que no se utiliza ninguna autorización y que se alteran temporalmente las restricciones de seguridad. Es decir, cada usuario autenticado puede utilizar la tabla de consulta para recuperar datos, independientemente de si está autorizado para los objetos respectivos. Las tablas de consulta compuestas con una tabla de consulta primaria que contiene datos de instancia no se deben establecer para utilizar la autenticación basada en rol.	Todos los usuarios autenticados pueden ver todo el contenido de la tabla de consulta, una vez que se han aplicado los filtros.
Suplementaria	Las tablas de consulta suplementarias no se deben establecer para utilizar autorización basada en instancia ya que no las gestiona Business Process Choreographer y por lo tanto no dispone de información de autorización para el contenido de estas tablas.	Las tablas de consulta suplementarias no se deben establecer para utilizar autorización basada en rol.	Todos los usuarios autenticados pueden ver todo el contenido de la tabla de consulta, una vez que se han aplicado los filtros.

En la figura siguiente se proporciona una visión general de las opciones disponibles para los tipos de autorización, en función del tipo de tabla de consulta. Además, describe brevemente los distintos comportamientos y la API de tabla de consulta así como sus opciones de autorización.

Figura 77. Autorización basada en instancia para tablas de consulta

*) Si se establece onBehalfUser, se aplica (A)

La autorización basada en instancia para objetos en el resultado de consulta que utiliza elementos de trabajo depende del parámetro de autorización que se pasa a la API de tabla de consulta y del valor del distintivo de autorización basada en instancia de la tabla de consulta.

- (A) Las consultas en tablas de consulta predefinidas o compuestas que utilizan el objeto AuthorizationOptions devuelven entidades que se correlacionan con un elemento de trabajo relacionado para este usuario específico. También es este el caso si se utiliza el objeto AdminAuthorizationOptions y se establece onBehalfUser. Los clientes estándares que presentan listas de tareas o procesos a los usuarios normalmente utilizan esta combinación de tablas de consulta o parámetros de la API de tabla de consulta.
- (B) El contenido completo de una tabla de consulta consta de las entidades que tienen un elemento de trabajo relacionado, tal como se ha configurado con la autorización basada en instancia de la tabla de consulta. La autorización basada en instancia considera cuatro tipos de elementos de trabajo: todos, persona, grupo y heredado. El emisor de la llamada que utiliza el método de consulta de API debe estar en el rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager. Esta combinación de tablas de consulta y parámetros de la API de tabla de consulta está destinada a ser utilizada en situaciones de administración donde se deba mostrar la lista completa de tareas o procesos disponibles, o realizar búsquedas en ella.
- (C) Las consultas de tablas de consulta que no utilizan la autorización basada en instancia o basada en rol devuelven el mismo resultado si se pasa AdminAuthorizationOptions o AuthorizationOptions en la API de tabla de consulta. Está disponible para tablas de consulta compuestas y suplementarias. Puesto que no existe ninguna comprobación de los elementos de trabajo o roles de Java EE, todos los usuarios autenticados ven el contenido completo. Los clientes que no deseen restringir la visibilidad de los objetos aplicando restricciones de autorización basada en instancia o basada en rol que proporciona Business Process Choreographer pueden desactivar las comprobaciones de autorización al desarrollar definiciones de tabla de consulta. Sin embargo, cuando se utiliza reclamar y completar, los usuarios deben tener elementos de trabajo relacionados.
- (D) Se puede acceder a los datos de plantilla en tablas de consulta predefinidas o tablas de consulta compuestas con autorización basada en rol sólo con autorización basada en rol. Esto requiere que el emisor de la llamada que utiliza el método de consulta de API esté en rol BPESystemAdministrator de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator de Java EE, si se utiliza el EJB de Human Task Manager. La API de tabla de consulta se puede utilizar para acceder a la información de plantilla en lugar de la API de consulta.

Elementos de trabajo y autorización basada en instancia

La autorización basada en instancia proporcionada por Business Process Choreographer se basa en los elementos de trabajo. Cada elemento de trabajo describe quién tiene qué derechos en qué objeto. Se puede acceder a esta información utilizando la tabla de consulta WORK_ITEM, si se utiliza autorización basada en instancia.

En la tabla se describen los distintos tipos de elementos de trabajo que se consideran si se utiliza autorización basada en instancia cuando se ejecuta una consulta contra una tabla de consulta:

Tabla 35. Tipos de elemento de trabajo

Tipo de elemento de trabajo	Descripción
Todos	Permite que todos los usuarios accedan a un objeto específico, por ejemplo una instancia de proceso o una tarea. En este caso, el atributo EVERYBODY del elemento de trabajo relacionado se establece en TRUE.
Persona	Elementos de trabajo que se crean para usuarios determinados. El atributo OWNER_ID del elemento de trabajo relacionado se establece en un usuario específico. Pueden existir varios elementos de trabajo que difieren en el atributo OWNER_ID para un objeto, como por ejemplo una tarea.
Grupo	Elementos de trabajo que se crean para usuarios de un grupo determinado. El atributo GROUP_NAME del elemento de trabajo relacionado se establece en un grupo específico.
Heredado	También se permite a los lectores y administradores de instancias de proceso heredar el acceso a las tareas de usuario que pertenecen a estas instancias de proceso, incluyendo las escaladas. Las comprobaciones de la existencia de un elemento de trabajo heredado en consultas de tareas se realizan con uniones de SQL complejas en el tiempo de ejecución, lo que afecta al rendimiento.

Business Process Choreographer crea elementos de trabajo en distintas situaciones. Por ejemplo, durante la creación de una tarea, se crean elementos de trabajo para los distintos roles, como por ejemplo el lector y el propietario potencial, si se han especificado criterios relacionados de asignación de personas.

En la tabla siguiente se describen los tipos de elementos de trabajo que se crean, en función de los criterios de asignación de personas que se han definido, si se utiliza autorización basada en instancia cuando se ejecuta la consulta en una tabla de consulta. Los elementos de trabajo heredados no aparecen en la tabla ya que reflejan una relación que no se ha modelado explícitamente durante el desarrollo de la aplicación de proceso.

Tabla 36. Elementos de trabajo y criterios de asignación de personas

Tipo de elemento de trabajo	Criterios relacionados de asignación de personas
Todos	Everybody
Persona	Todos los criterios de asignación de personas a excepción de los verbos *Nobody*, *Everybody* y *Group*
Grupo	Group

Filtro de autorización en tablas de consulta compuestas

En tablas de consulta compuestas, puede especificar un filtro de autorización si se utiliza la autorización basada en instancia. Este filtro restringe los elementos de trabajo que se utilizan para la autorización, en función de determinados atributos de los elementos de trabajo. Por ejemplo, el filtro de autorización `"WI.REASON=REASON_POTENTIAL_OWNER"` en una tabla de consulta compuesta con la tabla de consulta primaria TASK restringe las tareas que se pueden devolver cuando una persona ejecuta una consulta. El resultado contiene sólo las tareas que representan una tarea a realizar para esa persona, es decir, el resultado se restringe

a las tareas que la persona está autorizada a reclamar. Este filtro también se puede especificar como filtro de tabla de consulta o como filtro de consulta, pero por razones de rendimiento de las consultas, es beneficioso especificar este filtro como filtro de autorización.

Conceptos relacionados

"Tablas de consulta predefinidas" en la página 321
Las tablas de consulta predefinidas proporcionan acceso a los datos de la base de datos de Business Process Choreographer. Son la representación de las tablas de consulta de las vistas de base de datos de Business Process Choreographer predefinidas correspondientes, como por ejemplo la vista TASK o la vista PROCESS_INSTANCE. Estas tablas de consulta predefinidas mejoran la funcionalidad y el rendimiento de las vistas de base de datos predefinidas porque están optimizadas para ejecutar consultas de lista de tareas y procesos.

"Tablas de consulta suplementarias" en la página 324
Las tablas de consulta suplementarias en Business Process Choreographer exponen a la API de tabla de consulta datos empresariales que no gestiona Business Process Choreographer. Con tablas de consulta suplementarias, estos datos externos se pueden utilizar con datos de tablas de consulta predefinidas al recuperar información de instancias de procesos empresariales o información de tareas de usuario.

"Tablas de consulta compuestas" en la página 326
Las tablas de consulta compuestas de Business Process Choreographer no tienen una representación específica de los datos de la base de datos; constan de una combinación de datos de tablas de consulta predefinidas y suplementarias relacionadas. Utilice una tabla de consulta compuesta para recuperar la información para una lista de instancias de procesos o una lista de tareas, como por ejemplo Mis tareas a realizar.

"Opciones de autorización para la API de tabla de consulta" en la página 360
Cuando ejecuta una consulta en una tabla de consulta en Business Process Choreographer, puede pasar opciones de autorización como parámetros de entrada a los métodos de la API de tabla de consulta.

Tareas relacionadas

"Creación de tablas de consulta para Business Process Choreographer Explorer" en la página 376
Puede utilizar tablas de consulta en lugar de la API query EJB para mejorar el rendimiento de Business Process Choreographer Explorer. Para crear las tablas de consulta, utilice el generador de tablas de consulta.

Tipos de atributo para tablas de consulta

Los tipos de atributo son necesarios en Business Process Choreographer cuando se definen tablas de consulta, cuando se utilizan valores literales en las consultas y cuando se accede a los valores de un resultado de consulta. Hay reglas y correlaciones disponibles para cada uno de los tipos de atributo.

Para definir el tipo de un atributo de una tabla de consulta se utiliza un subconjunto de los tipos que están disponibles en el lenguaje de programación Java y de bases de datos. Los tipos de atributo son una abstracción del tipo de base de datos o tipo Java específico. Para tablas de consulta suplementarias, debe utilizar un tipo de base de datos válido a la correlación de tipos de atributo.

En la tabla siguiente se describen los tipos de atributo:

Tabla 37. Tipos de atributo

Tipo de atributo	Descripción
ID	El ID que se utiliza para identificar una tarea de usuario (TKIID), una instancia de proceso (PIID) u otros objetos. Por ejemplo, se utilizan ID para reclamar o completar una tarea de usuario determinada, que se identifica con el TKIID especificado.
STRING	Las descripciones de tarea o propiedades de consulta se pueden representar como serie.
NÚMERO	Se utilizan números para los atributos, como por ejemplo la prioridad de una tarea.
INDICACIÓN DE FECHA Y HORA	Las indicaciones de fecha y hora describen un punto en el tiempo, como por ejemplo la hora a la que se creó una tarea de usuario, o a la que finalizó una instancia de proceso.
DECIMAL	Se pueden utilizar decimales como tipo de propiedades de consulta, por ejemplo al definir una propiedad de consulta con una variable de tipo XSD doble.
BOOLEAN	Los booleanos pueden tener uno de dos valores, true o false. Por ejemplo, las tareas de usuario proporcionan un atributo, autoClaim, que identifica si la tarea se reclama automáticamente si sólo existe un único usuario como propietario potencial de esta tarea.

Correlación entre tipos de base de datos y tipos de atributo:

Utilice los tipos de atributo para definir tablas de consulta en Business Process Choreographer, cuando ejecute consultas en tablas de consulta y para acceder a valores de un resultado de consulta.

En la tabla siguiente se describen los tipos de base de datos y su correlación con los tipos de atributo:

Tabla 38. Correlación entre tipos de base de datos y tipos de atributo

Tipo de base de datos	Tipo de atributo
Un tipo binario con 16 bytes. Es el tipo utilizado para ID como por ejemplo TKIID en TASK de las tablas de Business Process Choreographer.	ID
Un tipo basado en caracteres. La longitud depende de la columna en la tabla de base de datos a la que hace referencia el atributo de la tabla de consulta.	STRING
Un tipo de base de datos de entero, como por ejemplo entero, corto o largo.	NÚMERO
Un tipo de base de datos de indicación de fecha y hora.	INDICACIÓN DE FECHA Y HORA
Un tipo de decimal, como por ejemplo flotante o doble.	DECIMAL
Un tipo convertible a un valor booleano, como por ejemplo un número. 1 se interpreta como *true* y todos los demás números como *false*.	BOOLEAN

Las tablas de consulta suplementarias normalmente hacen referencia a tablas y vistas de base de datos existentes, de tal forma que la creación de vista o tabla no es necesaria.

Ejemplo

Considere una tabla en un entorno DB2, CUSTOM.ADDITIONAL_INFO, que se representará en Business Process Choreographer como una tabla de consulta suplementaria. La siguiente sentencia SQL crea la tabla de base de datos:

```
CREATE TABLE CUSTOM.ADDITIONAL_INFO
(
  PIID       CHAR(16) FOR BIT DATA,
  INFO       VARCHAR(220),
  COUNT INTEGER);
```

Se utiliza la siguiente correlación entre tipos de columna de base de datos y tipos de atributo de tabla de consulta para una tabla de consulta suplementaria para la tabla CUSTOM.ADDITIONAL_INFO.

Tabla 39. Ejemplo de correlación entre tipos de base de datos y tipos de atributo

Columna de base de datos y tipo	Atributo y tipo de tabla de consulta
PIID CHAR(16) FOR BIT DATA	PIID (ID)
INFO VARCHAR(220)	INFO (STRING)
COUNT INTEGER	COUNT (NUMBER)

Correlación entre tipos de atributo y representaciones literales:

Se utilizan tipos de atributo cuando se definen tablas de consulta en Business Process Choreographer, cuando se ejecutan consultas en las tablas de consulta y cuando se accede a los valores de un resultado de consulta. Utilice este tema para obtener información sobre la correlación entre tipos de atributo y representaciones literales.

Los valores literales se pueden utilizar en expresiones para definir los criterios de filtro y selección, como por ejemplo en filtros de tablas de consulta compuestas y en filtros que se pasen a la API de tabla de consulta.

En la tabla siguiente se describen los tipos de atributo y su correlación con los valores literales. Los marcadores se marcan en *cursiva*. Tenga en cuenta que los tipos de atributo ID e INDICACIÓN DE FECHA Y HORA, que se pueden pasar a la API de tabla de consulta, utilizan una sintaxis especial, que también utiliza la API de consulta.

Tabla 40. Correlación entre tipos de atributo y valores literales

Tipo de atributo	Sintaxis y uso como valor literal en expresiones
ID	`ID ('representación de serie de un ID' )`
	Cuando se desarrollan aplicaciones cliente, los ID se representan como una serie o como una instancia de la interfaz com.ibm.bpe.api.OID. La representación de serie se puede obtener de una instancia de la interfaz com.ibm.bpe.api.OID utilizando el método toString. La serie debe estar entre comillas.
STRING	`'la serie'`
	La serie se debe colocar entre comillas.

Tabla 40. Correlación entre tipos de atributo y valores literales (continuación)

Tipo de atributo	Sintaxis y uso como valor literal en expresiones
NÚMERO	`número`
	Número como texto, y sin comillas. Se definen constantes para algunos atributos de número en tablas de consulta predefinidas, y éstas se pueden utilizar.
INDICACIÓN DE FECHA Y HORA	`TS ('AAAA-MM-DDThh:mm:ss')`
	La indicación de fecha y hora se debe especificar como: • `AAAA` es el año de 4 dígitos • `MM` es el mes de 2 dígitos del año • `DD` es el día de 2 dígitos del mes • `hh` es la hora de 2 dígitos del día (24 horas) • `mm` son los minutos de 2 dígitos de la hora • `ss` son los segundos de 2 dígitos del minuto. La indicación de fecha y hora se define en el huso horario del usuario.
DECIMAL	`número.fracción`
	Número decimal como texto y sin comillas; la parte `.fracción` es opcional.
BOOLEAN	`true`, `false`
	Valor booleano como texto.

Ejemplo

- `filterOptions.setQueryCondition("STATE=2");`
- `filterOptions.setQueryCondition("STATE=STATE_READY");`
- `criterio de selección en una tabla de consulta conectada TASK_DESC: "LOCALE='en_US'"`
- `filterOptions.setQueryCondition( "PTID=ID('_PT:8001011e.1dee8e51.247d6df6.29a60000')");`

Correlación entre tipos de atributo y parámetros:

Utilice los tipos de atributo al definir tablas de consulta en Business Process Choreographer, cuando ejecute consultas en las tablas de consulta y para acceder a valores de un resultado de consulta.

En la tabla siguiente se describen los tipos de atributo y su correlación con los valores de parámetro que se pueden utilizar en expresiones para definir los criterios de filtro y de selección, como por ejemplo en filtros de tablas de consulta compuestas y en filtros pasados a la API de tabla de consulta.

Tabla 41. Correlación entre tipos de atributo y valores de los parámetros de usuario

Tipo de atributo	Uso como valor de parámetro en expresiones
ID	`PARAM(nombre)` Cuando se desarrollan aplicaciones cliente, los ID se representan como una serie o como una instancia de la interfaz com.ibm.bpe.api.OID. Como parámetro, ambas representaciones son válidas. También se puede utilizar una matriz de bytes que refleje un OID válido (byte).
SERIE	`PARAM(nombre)` Representación de serie del objeto que pasa a la API de tabla de consulta durante la ejecución el método toString.
NÚMERO	`PARAM(nombre)` Una representación java.lang.Long, java.lang.Integer, java.lang.Short o java.lang.String del número se pasa a la API de tabla de consulta. También se pueden pasar nombres de las constantes, tal como se definen en algunos atributos de tablas de consulta predefinidas.
INDICACIÓN DE FECHA Y HORA	`PARAM(nombre)` Son válidas las representaciones siguientes: • Una representación java.lang.String de la indicación de fecha y hora • Instancias de com.ibm.bpe.api.UTCDate • Instancias de java.util.Calendar
DECIMAL	`PARAM(nombre)` Una representación java.lang.Long, java.lang.Integer, java.lang.Short, java.lang.Double, java.lang.Float o java.lang.String del decimal se pasa a la API de tabla de consulta.
BOOLEANO	`PARAM(nombre)` Los valores válidos son: • Una representación java.lang.String del booleano • java.lang.Short, java.lang.Integer, java.lang.Long con los valores adecuados; `0` (para false) o `1` (para true) • Un objeto `java.lang.Boolean`

Ejemplo

```
...
// este ejemplo muestra una consulta contra una tabla de consulta compuesta
// COMP.TASKS con un parámetro "customer"
java.util.List params = new java.util.ArrayList();

list.add(new com.ibm.bpe.api.Parameter("customer", "IBM");
// se pueden utilizar los Enterprise JavaBeans de Business Flow Manager o los
// Enterprise JavaBeans de Human Task Manager para acceder a las tablas de consulta
service.bfm.queryEntities("COMP.TASKS", null, null, params);
...
```

Correlación entre tipos de atributo y tipos de objeto Java:

Se utilizan tipos de atributo cuando se definen tablas de consulta en Business Process Choreographer, cuando se ejecutan consultas en las tablas de consulta y cuando se accede a los valores de un resultado de consulta. Utilice este tema para obtener información sobre la correlación entre tipos de atributo y tipos de objeto Java.

En la tabla siguiente se describen los tipos de atributo y su correlación con tipos de objeto Java en los conjuntos de resultados de consulta.

Tabla 42. Correlación entre tipos de atributo y tipos de objeto Java

Tipo de atributo	Tipo de objeto Java relacionado
ID	com.ibm.bpe.api.OID
SERIE	java.lang.String
NÚMERO	java.lang.Long
INDICACIÓN DE FECHA Y HORA	java.util.Calendar
DECIMAL	java.lang.Double
BOOLEANO	java.lang.Boolean

Ejemplo

```
...
// el ejemplo siguiente muestra una consulta en una tabla de consulta compuesta
// COMP.TA; el atributo "STATE" es de tipo de atributo NÚMERO
...
// ejecutar la consulta
// se pueden utilizar los Enterprise JavaBeans de Business Flow Manager o los
// Enterprise JavaBeans de Human Task Manager para acceder a las tablas de consulta
EntityResultSet rs = bfm.queryEntities("COMP.TA",null,null,params);

// obtener las entidades e iterar sobre ella
List entities = rs.getEntities();
for (int i = 0 ; i < entities.size(); i++) {

      // trabajar en una entidad determinada
      Entity en = (Entity) entities.get(i);

      // tenga en cuenta que el código siguiente se podría escribir
      // de forma más generalizada utilizando los objetos de información
      // de atributos contenidos en ei.getAttributeInfo()

      // obtener atributo STATE
      Long state = (Long) en.getAttributeValue("STATE");
      ...
}
...
```

Compatibilidad de tipo de atributo:

Utilice los tipos de atributo al definir tablas de consulta en Business Process Choreographer, cuando ejecute consultas en las tablas de consulta y para acceder a valores de un resultado de consulta.

En la tabla siguiente se muestran los tipos de atributo y sus tipos de atributo compatibles, que se pueden utilizar para definir filtros y criterios de selección en las tablas de consulta. Los tipos de atributo compatibles aparecen con la marca **X**.

Tabla 43. Compatibilidad de tipo de atributo

Tipo de atributo	ID	SERIE	NÚMERO	INDICACIÓN DE FECHA Y HORA	DECIMAL	BOOLEANO
ID	X					
SERIE		X				
NÚMERO			X		X	
INDICACIÓN DE FECHA Y HORA				X		
DECIMAL			X		X	
BOOLEANO						X

En las expresiones de tablas de consulta que especifican criterios de filtro y condición, los tipos de atributo o valores que se comparan deben ser compatibles. Por ejemplo, `WI.OWNER_ID=1` es un filtro no válido porque el operando del lado izquierdo es de tipo SERIE y el operando del lado derecho es de tipo NÚMERO.

Consultas de tabla de consulta

Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta, que está disponible en la API de REST y de EJB de Business Flow Manager.

Una consulta se ejecuta sólo en una única tabla de consulta. Se utilizan los métodos de la API basados en entidades y los métodos de la API basados en filas para recuperar contenido de las tablas de consulta. Los parámetros de entrada se pasan a los métodos de la API de tabla de consulta.

Conceptos relacionados

"Desarrollo de tablas de consulta" en la página 334
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante Query Table Builder. Las tablas de consulta predefinidas no se pueden desarrollar ni desplegar. Están disponibles cuando se instala Business Process Choreographer y proporcionan una vista simple de los artefactos del esquema de base de datos de Business Process Choreographer.

"Tablas de consulta predefinidas" en la página 321
Las tablas de consulta predefinidas proporcionan acceso a los datos de la base de datos de Business Process Choreographer. Son la representación de las tablas de consulta de las vistas de base de datos de Business Process Choreographer predefinidas correspondientes, como por ejemplo la vista TASK o la vista PROCESS_INSTANCE. Estas tablas de consulta predefinidas mejoran la funcionalidad y el rendimiento de las vistas de base de datos predefinidas porque están optimizadas para ejecutar consultas de lista de tareas y procesos.

"Tablas de consulta suplementarias" en la página 324
Las tablas de consulta suplementarias en Business Process Choreographer exponen a la API de tabla de consulta datos empresariales que no gestiona Business Process Choreographer. Con tablas de consulta suplementarias, estos datos externos se pueden utilizar con datos de tablas de consulta predefinidas al recuperar información de instancias de procesos empresariales o información de tareas de usuario.

"Tablas de consulta compuestas" en la página 326
Las tablas de consulta compuestas de Business Process Choreographer no tienen una representación específica de los datos de la base de datos; constan de una combinación de datos de tablas de consulta predefinidas y suplementarias relacionadas. Utilice una tabla de consulta compuesta para recuperar la información para una lista de instancias de procesos o una lista de tareas, como por ejemplo Mis tareas a realizar.

"Filtros y criterios de selección de tablas de consulta" en la página 338
Los filtros y los criterios de selección se definen durante el desarrollo de la tabla de consulta utilizando Query Table Builder, que utiliza una sintaxis similar a las cláusulas WHERE de SQL. Utilice estos filtros y criterios de selección claramente definidos para especificar las condiciones que se basan en los atributos de las tablas de consulta.

Métodos de la API de tabla de consulta:

Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta. Hay disponibles métodos de la API basados en entidades y los métodos de la API basados en filas para recuperar el contenido de las tablas de consulta.

Se proporcionan los métodos basados en entidades y los métodos basados en filas siguientes para ejecutar consultas en las tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta:

Tabla 44. Métodos para consultas ejecutadas en tablas de consulta

Finalidad	Métodos
Contenido de la consulta	• queryEntities • queryRows Ambos métodos devuelven contenido de la tabla de consulta. El método queryEntities devuelve contenido basado en entidades y queryRows devuelve contenido basado en filas.
Consulta el número de objetos	• queryEntityCount • queryRowCount Ambos métodos devuelven el número de objetos en la tabla de consulta, mientras que el número real puede depender de si se realiza un enfoque basado en entidades o uno basado en filas.

Las consultas basadas en entidades, utilizando el método queryEntities y el método queryEntityCount, asumen que una tabla de consulta contiene exclusivamente entidades identificables, tal como define la clave primaria de la tabla de consulta primaria.

Las consultas basadas en filas, utilizando el método queryRows y el método queryRowCount, devuelven un conjunto de resultados como JDBC, que está basado en filas, y proporciona los métodos first y next para la navegación en este. El conjunto de resultados que se devuelve al ejecutar una consulta en una tabla de consulta utilizando la API de tabla de consulta se puede comparar con QueryResultSet que devuelve la API de consulta. En general, el número de filas es mayor que el número de entidades que contiene una tabla de consulta. La misma entidad, por ejemplo, una tarea de usuario identificada por su ID de tarea, como por ejemplo TKIID, puede aparecer varias veces en el conjunto de resultados de fila.

Una instancia específica contenida en cualquier tabla de consulta predefinida existe sólo una vez en un entorno de Business Process Choreographer. Ejemplos de instancias son las tareas de usuario y los procesos empresariales. Estas instancias se identifican de forma exclusiva utilizando un ID o conjunto de ID. Éste es el TKIID para instancias de tareas de usuario y el PIID para instancias de proceso.

Las tablas de consulta compuestas se componen de una tabla de consulta primaria y de cero o más tablas de consulta conectadas. Los objetos contenidos en las tablas de consulta compuestas se identifican de forma exclusiva mediante el ID exclusivo de los objetos contenidos en la tabla de consulta primaria. La tabla de consulta primaria de una tabla de consulta compuesta determina su tipo de entidad. Por ejemplo, una tabla de consulta compuesta con la tabla de consulta primaria TASK contiene entidades del tipo TASK. La relación de uno a uno o de uno a cero entre las tablas de consulta primaria y conectadas asegura que las tablas de consulta conectadas no darán como resultado entidades duplicadas.

Las consultas basadas en entidades explotan las entidades identificables de forma exclusiva de una tabla de consulta, como define la clave primaria de la tabla de consulta primaria. Un programador de aplicaciones cliente para interfaces de usuario normalmente está interesado en instancias exclusivas sin duplicados, por

ejemplo, para visualizar una tarea de usuario una vez sólo en la interfaz de usuario. Se devuelven instancias exclusivas si se utiliza la API de tabla de consulta basada en entidades.

Las consultas basadas en filas pueden devolver filas duplicadas de la tabla de consulta primaria si se utiliza la autorización basada en instancia.

- La información de la tabla de consulta WORK_ITEM se recupera con la consulta. Por ejemplo, si se recupera el atributo `WI.REASON` además de los atributos definidos en la tabla de consulta, múltiples filas se califican para el resultado. Esto se debe a que existen varias razones por las que un usuario puede acceder a una entidad como, por ejemplo, una instancia de proceso o tarea.
- Se utiliza autorización basada en instancia y no se especifica distinct. Aunque no se recupera información de elementos de trabajo, se pueden devolver varias filas si se utiliza la autorización basada en instancia.

Si se utiliza la API de tabla de consulta basada en entidades:

- Las consultas basadas en entidades se ejecutan siempre con el operador distinct de SQL.
- Las consultas basadas en entidades devuelven un resultado que permite valores de matriz para información relacionada con elementos de trabajo.

Parámetros de la API de tabla de consulta:

Puede utilizar métodos de la API de tabla de consulta para recuperar contenido al ejecutar consultas en una tabla de consulta en Business Process Choreographer.

Se pasan los siguientes parámetros de entrada a los métodos de la API de tabla de consulta:

Tabla 45. Parámetros de la API de tabla de consulta

Parámetro	Opcional	Tipo y descripción
Nombre de tabla de consulta	No	java.lang.String
		Nombre exclusivo de la tabla de consulta.
Opciones de filtro	Sí	com.ibm.bpe.api.FilterOptions, si se utiliza el Enterprise JavaBeans de Business Flow Manager, o com.ibm.task.api.FilterOptions, si se utiliza Enterprise JavaBeans de Human Task Manager.
		Opciones que se pueden utilizar para definir la consulta. Por ejemplo, se establece un umbral de consulta en este parámetro para limitar el número de resultados devueltos.
Opciones de autorización	Sí	com.ibm.bpe.api.AuthorizationOptions o com.ibm.bpe.api.AdminAuthorizationOptions si se utiliza Enterprise JavaBeans de Business Flow Manager. com.ibm.task.api.AuthorizationOptions o com.ibm.task.api.AdminAuthorizationOptions si se utiliza Enterprise JavaBeans de Human Task Manager.
		Se puede restringir aún más la autorización si se utiliza la autorización basada en instancia. Para tablas de consulta que requieren autorización basada en rol, se debe pasar una instancia de AdminAuthorizationOptions.

Tabla 45. Parámetros de la API de tabla de consulta (continuación)

Parámetro	Opcional	Tipo y descripción
Parámetros	Sí	java.util.List de com.ibm.bpe.api.Parameter, si se utiliza Enterprise JavaBeans de Business Flow Manager, o com.ibm.task.api.Parameter, si se utiliza Enterprise JavaBeans de Human Task Manager.
		Se utiliza este parámetro para pasar parámetros de usuario, que se especifican en un criterio de selección o filtro en una tabla de consulta compuesta.

Una consulta se ejecuta solamente en una tabla de consulta específica. La relación entre varias tablas de consulta se define con tablas de consulta compuestas. En términos de la API de consulta (diferenciada de la API de tabla de consulta), esto corresponde a vistas de base de datos.

Especifique filtros y criterios de selección en expresiones durante el desarrollo de tabla de consulta utilizando Query Table Builder. Para obtener más información, consulte en el centro de información el tema que trata sobre las tablas de consulta compuestas y el tema que trata sobre los criterios de filtro y búsqueda de las tablas de consulta. Para obtener información sobre Query Table Builder, consulte el sitio de SupportPacs de WebSphere Business Process Management. Busque la sección PA71 WebSphere Process Server - Query Table Builder. Para acceder al enlace, consulte la sección de referencias relacionadas de este tema.

Nombre de tabla de consulta:

Cuando ejecuta una consulta en una tabla de consulta de Business Process Choreographer, el nombre de la tabla de consulta se pasa como parámetro de entrada a los métodos de la API de tabla de consulta.

El nombre de la tabla de consulta es el nombre de la tabla de consulta en la que se ejecuta la consulta.

- Para las tablas de consulta predefinidas, es el nombre de la tabla de consulta predefinida.
- Para las tablas de consulta compuestas y suplementarias, es el nombre de la tabla de consulta respectiva que se especifica al crear el modelo de la tabla de consulta. El nombre de la tabla de consulta compuesta o suplementaria sigue el convenio de denominación *prefijo.nombre*, donde *prefijo* no puede ser `'IBM'`.

Tanto el nombre de la tabla de consulta como el prefijo deben estar en mayúsculas. La longitud máxima del nombre de la tabla de consulta es de 28 caracteres.

Opciones de filtro para las tablas de consultas:

Cuando ejecuta una consulta en una tabla de consulta en Business Process Choreographer, las opciones de filtro se pueden pasar como parámetros de entrada a los métodos de la API de tabla de consulta.

Se puede pasar a la API de tabla de consulta una instancia de la clase com.ibm.bpe.api.FilterOptions, si se utiliza Enterprise JavaBeans de Business Flow Manager, o una instancia de com.ibm.task.api.FilterOptions, si se utiliza Enterprise JavaBeans de Human Task Manager. Las opciones de filtro permiten una configuración de la consulta utilizando:

- Un umbral y desplazamiento (skipCount)
- Atributos de ordenación (similares a la cláusula ORDER BY de una consulta de SQL)
- Un filtro de consultas proporcionado por el usuario
- El conjunto de atributos devuelto, incluida la información de elementos de trabajo
- Otros

El conjunto de resultados que se puede obtener de una tabla de consulta se especifica mediante la definición de la tabla de consulta. Sin embargo, es posible que desee especificar opciones adicionales cuando se ejecute la consulta. La tabla siguiente describe las opciones que se pueden especificar como opciones de filtro utilizando el objeto FilterOptions.

Tabla 46. Parámetros de la API de tabla de consulta: Opciones de filtro

Opción	Tipo	Descripción
Atributos seleccionados	java.lang.String	• Una lista de atributos separados por comas de la tabla de consulta que se debe devolver en el conjunto de resultados. • Si se utiliza una autorización basada en instancias, la información de elemento de trabajo se puede recuperar especificando los atributos de la tabla de consultas WORK_ITEM, con el prefijo `WI.`, por ejemplo, `WI.REASON`. • Si se especifica nulo, se devolverán todos los atributos de la tabla de consulta, sin la información de los elementos de trabajo.
Atributos de ordenación	java.lang.String	Una lista de atributos separados por comas de la tabla de consulta, opcionalmente seguida por ASC o DESC, para orden ascendente y descendente, respectivamente. Esta lista es similar a la cláusula de SQL ORDER BY: *atributosOrdenación* ::= *atributo* [ASC|DESC] [, *atributosOrdenación*]. Si no se especifica ASC o DESC, se asume ASC. La ordenación se produce en la secuencia de los atributos de ordenación. Este ejemplo clasifica las tareas de la tabla de consulta TASK en orden ascendente por estado y dentro de los grupos del mismo estado (STATE) por nombre (NAME), en orden ascendente: `"STATE DESC, NAME ASC".`
Umbral	java.lang.Integer	Define el máximo: • Número de filas devueltas si se utiliza queryRows. • Número de entidades devuelvas si se utiliza queryEntities. El número real de entidades disponibles en la tabla de consulta respectiva puede superar el número de umbral de entidades para la consulta incluso si el conjunto de resultados de la entidad no contiene entidades como el número de umbral. Esto se debe a razones técnicas si se selecciona la información de elementos de trabajo. • Recuento devuelto si se utiliza queryRowCount o queryEntityCount. El valor por omisión es nulo, lo que significa que no se establece ningún umbral.

Tabla 46. Parámetros de la API de tabla de consulta: Opciones de filtro (continuación)

Opción	Tipo	Descripción
Recuento de saltos	java.lang.Integer	Define el número de filas (consultas basadas en fila) o el número de entidades (consultas basadas en entidad) que se saltan. De la misma forma que con el parámetro de umbral, skipCount puede no ser preciso para consultas basadas en entidad. Se utiliza el recuento de saltos para permitir la paginación de un conjunto de resultados grande. El valor por omisión es nulo, lo que significa que no se establece ningún skipCount.
Huso horario	java.util.TimeZone	El huso horario que se utiliza al convertir indicaciones de fecha y hora. Un ejemplo es CREATED en la tabla de consulta predefinida TASK. Si no se especifica (nulo), se utilizará el huso horario del servidor.
Entorno local	java.util.Locale	Se utiliza el entorno local para calcular el valor del parámetro del sistema `$LOCALE`. Un uso de ejemplo de `$LOCALE` en un criterio de selección es: `'LOCALE=$LOCALE'`.
Filas Distinct	java.lang.Boolean	Se utiliza sólo para consultas basadas en fila. Si se establece en true, las consultas basadas en fila devuelven filas Distinct. Esto no implica que se devuelvan filas exclusivas debido a la posible multiplicidad de información de elementos de trabajo.
Condición de consulta	java.lang.String	Esta opción realiza un filtrado adicional en el conjunto de resultados. Se puede hacer referencia a todos los atributos definidos para la tabla de consultas. Si la autorización es necesaria para la tabla de consultas, también se puede hacer referencia a las columnas definidas para la tabla de consultas WORK_ITEM utilizando el prefijo `WI`, por ejemplo, `WI.REASON=REASON_POTENTIAL_OWNER`.

Opciones de autorización para la API de tabla de consulta:

Cuando ejecuta una consulta en una tabla de consulta en Business Process Choreographer, puede pasar opciones de autorización como parámetros de entrada a los métodos de la API de tabla de consulta.

Utilice una instancia de la clase com.ibm.bpe.api.AuthorizationOptions o com.ibm.bpe.api.AdminAuthorizationOptions, si se utiliza el EJB de Business Flow Manager EJB, o una instancia de la clase com.ibm.task.api.AuthorizationOptions o la clase com.ibm.task.api.AdminAuthorizationOptions, si se utiliza el EJB de Human Task Manager, para especificar opciones de autorización adicionales cuando se ejecuta la consulta.

Si se utiliza la autorización basada en instancia, las instancias de la clase AuthorizationOptions permiten la especificación del tipo de elementos de trabajo utilizados para identificar instancias elegibles devueltas por la consulta.

Se puede pasar una instancia de la clase AuthorizationOptions a la API de tabla de consulta si la consulta se ejecuta en una tabla de consulta predefinida que contiene datos de instancia. También se puede pasar si la consulta se ejecuta en una tabla de consulta compuesta con una tabla de consulta primaria que contiene datos de instancia y está configurada la utilización de la autorización basada en instancia. Si la consulta se ejecuta en una tabla de consulta predefinida con datos de plantilla o en una tabla de consulta compuesta con la autorización basada en rol configurada, se genera una excepción `com.ibm.bpe.api.EngineNotAuthorizedException`, si se utiliza el EJB de Business Flow Manager, o se genera

com.ibm.task.api.NotAuthorizedException, si se utiliza el EJB de Human Task Manager. En todos los demás casos, las opciones de autorización pasadas a la API de tabla de consulta se ignoran.

Las tablas de consulta compuestas pueden restringir los tipos de elementos de trabajo que se consideran al identificar objetos (o entidades) contenidas en ellas. Por ejemplo, si las opciones de autorización que se pasan a la API de tabla de consulta se configuran para utilizar elementos de trabajo Todos, esto sólo se tiene en cuenta si está definida la utilización de elementos de trabajo Todos en la definición de la tabla de consulta compuesta. Como regla simple, un tipo de elemento de trabajo que no se especifica para ser considerado en la definición de tabla de consulta no se puede sobrescribir para ser considerado por la API de tabla de consulta, pero un tipo de elemento de trabajo que se especifica para ser considerado en la definición de tabla de consulta se puede sobrescribir para no ser utilizado. Además, el tipo de autorización de una tabla de consulta predefinida o compuesta no se puede sobrescribir por la API de tabla de consulta.

En función del tipo de tabla de consulta que se consulte, se aplican distintos valores por omisión de opciones de autorización si el objeto de autorización no se especifica o si los atributos relacionados (todos, individual, grupo o heredado) se establecen en nulo, que es el valor por omisión.

En la tabla siguiente se muestran los valores por omisión de las opciones de autorización para la autorización basada en instancia para el tipo de tabla de consulta y tipo de elemento de trabajo utilizados.

Tabla 47. Parámetros de la API de tabla de consulta: valores por omisión de las opciones de autorización para la autorización basada en instancia

Tipo de tabla de consulta	Elemento de trabajo Todos	Elemento de trabajo Persona	Elemento de trabajo Grupo	Elemento de trabajo Heredado
Predefinida con datos de instancia	TRUE	TRUE	TRUE	FALSE
Predefinida con datos de plantilla	N/A	N/A	N/A	N/A
Compuesta con una tabla de consulta primaria con datos de instancia	TRUE	TRUE	TRUE	TRUE
Compuesta con una tabla de consulta primaria con datos de plantilla	N/A	N/A	N/A	N/A
Suplementaria	N/A	N/A	N/A	N/A

N/A significa que no se utiliza la autorización basada en instancia y, por lo tanto, cualquier valor en el objeto de autorización respecto a los elementos de trabajo se ignorará.

Si se especifica TRUE, la consulta resultante sólo considerará el tipo de elemento de trabajo específico si la tabla de consulta está definida para utilizar este tipo de elemento de trabajo. Es true para todas las tablas de consulta predefinidas con datos de instancia, pero es posible que no sea true para una tabla de consulta compuesta. Para el elemento de trabajo Grupo, este último debe estar habilitado en el contenedor de tareas de usuario. Un ejemplo del elemento de trabajo heredado establecido en TRUE es que el administrador de una instancia de proceso pueda ver las instancias de tareas de usuario participantes que se crean para esa instancia de proceso.

Especifique una instancia de la clase AdminAuthorizationOptions en lugar de una instancia de la clase AuthorizationOptions si:

- Una consulta se ejecuta en una tabla de consulta con autorización basada en rol. Las tablas de consulta predefinidas con datos de plantilla requieren la autorización basada en rol y las tablas de consulta compuesta con una tabla de consulta primaria con datos de plantilla se pueden configurar para requerir la autorización basada en rol.
- Una consulta se ejecuta en una tabla de consulta con datos de instancia o en una tabla de consulta compuesta con una tabla de consulta primaria que contiene datos de instancias. Debe devolver el contenido de esa tabla de consulta, independientemente de las restricciones debidas a la autorización para un usuario determinado. Este comportamiento es equivalente a la utilización del método queryAll en la API de consulta (diferenciado de la API de tabla de consulta).
- Una consulta se debe ejecutar en nombre de otro usuario.

En la tabla siguiente se describe cómo se consiguen los diversos comportamientos mencionados anteriormente:

Tabla 48. Parámetros de la API de tabla de consulta: AdminAuthorizationOptions

Situación	**Descripción**
onBehalfUser establecido en nulo	• Si la consulta se ejecuta en una tabla de consulta con autorización basada en rol, se devuelve todo el contenido de esa tabla de consulta. • Si la consulta se ejecuta en una tabla de consulta que utiliza autorización basada en instancia, los objetos específicos contenidos en la tabla de consulta no se comprueban para ver si existen elementos de trabajo para un usuario determinado. Se devuelven todos los objetos contenidos en la tabla de consulta.
onBehalfUser establecido en un usuario determinado	La consulta se ejecuta con la autorización del usuario especificado y los objetos de la tabla de consulta se comprueban contra los elementos de trabajo para este usuario, si se utiliza la autorización basada en instancia.

Si especifica AdminAuthorizationOptions, el emisor de la llamada debe estar en rol BPESystemAdministrator o BPESystemMonitor de Java EE, si se utiliza el EJB de Business Flow Manager, o en el rol TaskSystemAdministrator o TaskSystemMonitor de Java EE, si se utiliza el EJB de Human Task Manager.

Conceptos relacionados

"Autorización para tablas de consulta" en la página 343
Puede utilizar la autorización basada en instancia, la autorización basada en rol o ninguna autorización al ejecutar consultas en tablas de consulta.

Parámetros:

Cuando ejecuta una consulta en una tabla de consulta en Business Process Choreographer, puede pasar parámetros de usuario como parámetros de entrada a los métodos de la API de tabla de consulta. En las definiciones de tabla de consulta, puede especificar parámetros en filtros en la tabla de consulta primaria, en la autorización y en la tabla de consulta. También se pueden especificar parámetros en los criterios de selección en las tablas de consulta conectadas.

Los parámetros del sistema, `$USER` y `$LOCALE`, se sustituyen durante la ejecución en los filtros y criterios de selección, y no es necesario pasarlos a la API de tabla de consulta. El valor de entrada para el cálculo del parámetro del sistema `$LOCALE` se proporciona estableciendo el entorno local en las opciones de filtro.

Se deben pasar los parámetros de usuario en la API de tabla de consulta cuando se ejecuta la consulta. Esto se lleva a cabo pasando una lista de instancias de la clase com.ibm.bpe.api.Parameter, si se utiliza el EJB de Business Flow Manager, o una instancia de la clase com.ibm.task.api.Parameter, si se utiliza el EJB de Human Task Manager.

Se deben especificar las propiedades siguientes en un objeto de parámetro:

Tabla 49. Parámetros de usuario para la API de tabla de consulta

Propiedad	Descripción
Nombre	Nombre del parámetro tal y como se utiliza en la definición de la tabla de consulta. El nombre distingue entre mayúsculas y minúsculas.
Valor	Valor del parámetro. El tipo del parámetro debe ser compatible con el tipo del operando del lado izquierdo de todos los filtros y criterios de selección donde se utiliza este parámetro. Las constantes que se definen en algunos atributos de las tablas de consulta predefinidas se pueden pasar como serie, por ejemplo, `STATE_READY`.

Ejemplo

```
// ejecutar una consulta contra una tabla de consulta compuesta
// CUST.CPM con el filtro de tabla de consulta primaria
// establecido en 'STATE=PARAM(elEstado)'
EntityResultSet ers = null;
List parameterList = new ArrayList();
parameterList.add(new Parameter
("elEstado", new Integer(2)));

// ejecutar la consulta;
// se puede utilizar el EJB de Business Flow Manager o el
// EJB de Human Task Manager para acceder a las tablas de consulta
ers = bfm.queryEntities
("CUST.CPM", null, null, parameterList);

// trabajar en el conjunto de resultados
// ...
```

Resultados de consultas de tabla de consulta:

Puede utilizar métodos de la API de tabla de consulta cuando ejecute consultas en una tabla de consulta en Business Process Choreographer. El resultado de una consulta de método queryEntityCount o queryRowCount es un número. Los métodos queryEntities y queryRows devuelven conjuntos de resultados.

EntityResultSet

El método queryEntities devuelve una instancia de la clase com.ibm.bpe.api.EntityResultSet si se utiliza Enterprise JavaBeans de Business Flow Manager. El método queryEntities devuelve una instancia de la clase com.ibm.task.api.EntityResultSet si se utiliza Enterprise JavaBeans de Human Task Manager. Un conjunto de resultados de entidad tiene las propiedades siguientes:

Tabla 50. Propiedades de conjunto de resultados de entidad de una entidad de la API de tabla de consulta

Propiedad	Descripción
queryTableName	Nombre de la tabla de consulta en la que se ha ejecutado la consulta.
entityTypeName	• Si la consulta se ha ejecutado en una tabla de consulta compuesta, es el nombre de la tabla de consulta primaria. • Si la consulta se ha ejecutado en una tabla de consulta predefinida o en una tabla de consulta suplementaria, es el nombre de la tabla de consulta, es decir, el mismo valor que la propiedad *queryTableName*.
EntityInfo	Esta propiedad contiene la metainformación de las entidades contenidas en el conjunto de resultados de entidad. En este objeto se puede recuperar una lista java.util.List de objetos com.ibm.bpe.api.AttributeInfo, si se utiliza el EJB de Business Flow Manager, o una lista de objetos com.ibm.task.api.AttributeInfo, si se utiliza el EJB de Human Task Manager. Esta lista contiene los nombres de atributo y tipos de atributo de la información contenida en las entidades de este conjunto de resultados. También contiene metainformación sobre los atributos que constituyen la clave para estas entidades.
entities	Una lista java.util.List de objetos com.ibm.bpe.api.Entity, si se utilzia el EJB de Business Flow Manager, o una lista de objetos com.ibm.task.api.Entity, si se utiliza Human Task Manager.
locale	El entorno local que se calcula para el parámetro del sistema `$LOCALE`.

Las instancias de la clase Entity contienen la información que se recupera de la consulta de tabla de consulta. Una entidad representa un objeto identificable de forma exclusiva como por ejemplo una tarea, una instancia de proceso, una actividad o una escalada. Están disponibles las propiedades siguientes para las entidades:

Tabla 51. Propiedades de entidad de una entidad de la API de tabla de consulta

Propiedad	Descripción
EntityInfo	El objeto EntityInfo que también está contenido en el conjunto de resultados de entidad. En este objeto se puede recuperar una lista java.util.List de objetos com.ibm.bpe.api.AttributeInfo, si se utiliza el EJB de Business Flow Manager, o una lista de objetos com.ibm.task.api.AttributeInfo, si se utiliza el EJB de Human Task Manager. Esta lista contiene los nombres de atributo y tipos de atributo de la información contenida en las entidades de este conjunto de resultados. También contiene metainformación sobre los atributos que constituyen la clave para estas entidades.
attributeValue (*nombreAtributo*)	El valor del atributo especificado que se recupera para esta entidad. El tipo está contenido en el objeto AttributeInfo relacionado de este atributo.
attributeValuesOfArray (*nombreAtributo*)	Una matriz de valores. Utilice esta propiedad si la propiedad de información de atributo *array* está establecida en true, lo que es actualmente el caso sólo si el atributo hace referencia a la información de elementos de trabajo.

El número de entidades del conjunto de resultados de entidad se recupera utilizando el método size en la lista de entidades.

Ejemplo: API de tabla de consulta basada en entidad:

```
...
// el ejemplo siguiente muestra una consulta contra la
// tabla de consulta predefinida TASK, utilizando la API basada en entidad

...
// ejecutar la consulta
// el servicio es un objeto (Local)BusinessFlowManager o un
// objeto (Local)HumanTaskManager
EntityResultSet rs = service.queryEntities("TASK", null, null, null);

// obtener la metainformación de entidades
EntityInfo ei = rs.getEntityInfo();
List atts = ei.getAttributeInfo();

// obtener las entidades e iterar sobre ella
Iterator entitiesIter = rs.getEntities().iterator();
while (entitiesIter.hasNext()) {

      // trabajar en una entidad determinada
      Entity en = (Entity) entitiesIter.next();

      for (int i = 0; i < atts.size(); i++) {
            AttributeInfo ai = (AttributeInfo) atts.get(i);
            Serializable value = en.getAttributeValue(ai.getName()) ;

            // procesar...
      }
}
...
```

RowResultSet

El método queryRows devuelve una instancia de la clase com.ibm.bpe.api.RowResultSet si se utiliza Enterprise JavaBeans de Business Flow Manager. El método queryRows devuelve una instancia de la clase

com.ibm.task.api.RowResultSet si se utiliza Enterprise JavaBeans de Human Task Manager. Este tipo de conjunto de resultados es similar a un conjunto de resultados JDBC. Un conjunto de resultados de fila tiene las propiedades siguientes:

Tabla 52. Propiedades de conjunto de resultados de fila de una fila de la API de tabla de consulta

Propiedad	Descripción
primaryQueryTableName	• Si la consulta se ha ejecutado en una tabla de consulta compuesta, es el nombre de la tabla de consulta primaria. • Si la consulta se ha ejecutado en una tabla de consulta predefinida o en una tabla de consulta suplementaria, es el nombre de la tabla de consulta, es decir, el mismo valor que la propiedad *queryTableName*.
attributeInfo	Esta propiedad contiene una lista de objetos com.ibm.bpe.api.AttributeInfo, si se utiliza Enterprise JavaBeans de Business Flow Manager, o una lista de objetos com.ibm.task.api.AttributeInfo, si se utiliza Enterprise JavaBeans de Human Task Manager. Describen la metainformación para este conjunto de resultados. Los objetos AttributeInfo contienen los nombres de atributo y tipos de atributo de la información. No se incluyen los metadatos sobre claves ya que los conjuntos de resultados de fila no contienen ninguna clave.
attributeValue	El valor del atributo especificado que se ha recuperado para esta fila. El tipo está contenido en el objeto AttributeInfo relacionado de este atributo.
next, first, last, previous	Estos métodos se utilizan para desplazarse por el conjunto de resultados de fila. Compare su uso con los iteradores, enumeraciones o conjuntos de resultados JDBC.

El número de filas del conjunto de resultados de fila se recupera utilizando el método size en la lista de filas.

Ejemplo: API de tabla de consulta basada en fila

```
...
// el ejemplo siguiente muestra una consulta contra la
// tabla de consulta predefinida TASK, utilizando la API basada en entidad
...
// ejecutar la consulta
// el servicio es un objeto (Local)BusinessFlowManager o un
// objeto (Local)HumanTaskManager
RowResultSet rs = service.queryRows("TASK", null, null, null);

// obtener la metainformación de entidades
List atts = rs.getAttributeInfo();

// obtener las entidades e iterar sobre ella
while (rs.next()) {

      // trabajar en una fila determinada
      for (int i = 0; i < atts.size(); i++) {
            AttributeInfo ai = (AttributeInfo) atts.get(i);
            Serializable value = rs.getAttributeValue(ai.getName()) ;

            // procesar...
      }
}
...
```

Consultas de tabla de consulta para recuperación de metadatos

Las consultas se ejecutan en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta. Hay disponibles métodos para recuperar metadatos de las tablas de consulta.

Se proporcionan los métodos siguientes para recuperar metadatos al ejecutar consultas en tablas de consulta en Business Process Choreographer utilizando la API de tabla de consulta:

Tabla 53. Métodos para la recuperación de metadatos de tablas de consultas

Finalidad	Método
Devolver los metadatos de una tabla de consulta específica	getQueryTableMetaData
Devolver una lista de metadatos de tabla de consulta con propiedades específicas	findQueryTableMetaData
Devolver contenido de una tabla de consulta, basada en entidades, y un subconjunto de los metadatos para los atributos seleccionados	queryEntities
Devolver contenido de una tabla de consulta, basada en filas, y un subconjunto de los metadatos para los atributos seleccionados	queryRows

Los metadatos de las tablas de consulta constan de datos que están relacionados con la estructura y de datos que están relacionados con la internacionalización.

En la tabla siguiente se muestran los metadatos que están relacionados con la estructura de una tabla de consulta.

Tabla 54. Metadatos relacionados con la estructura de la tabla de consulta

Metadatos	Descripción	Devueltos por getQuery-TableMetaData	Devueltos por findQuery-TableMetaData	Devueltos por queryEntities	Devueltos por queryRows
Nombre de tabla de consulta	Nombre de la tabla de consulta	Sí	Sí	Sí	Sí
Nombre de la tabla de consulta primaria	Para tablas de consulta predefinidas y suplementarias, nombre de la tabla de consulta; para tablas de consulta compuestas, nombre de la tabla de consulta primaria	Sí	Sí	Sí	Sí
Tipo	Tipo de la tabla de consulta: predefinida, compuesta o suplementaria	Sí	Sí	No	No
Autorización	Autorización definida en la tabla de consulta: • Uso de elementos de trabajo • Autorización basada en instancia, basada en rol o ninguna	Sí	Sí	No	No

Tabla 54. Metadatos relacionados con la estructura de la tabla de consulta (continuación)

Metadatos	Descripción	Devueltos por getQuery-TableMetaData	Devueltos por findQuery-TableMetaData	Devueltos por queryEntities	Devueltos por queryRows
Atributos definidos	Metadatos de los atributos definidos en la tabla de consulta	Sí	Sí	No. Se devuelven los metadatos de los atributos seleccionados.	No. Se devuelven los metadatos de los atributos seleccionados.
Atributos de clave	Atributos de clave de la tabla de consulta	Sí	Sí	Sí	No. No aplicable a consultas basadas en fila.

En la tabla siguiente se muestran los metadatos que están relacionados con la internacionalización de una tabla de consulta.

Tabla 55. Metadatos relacionados con la internacionalización de la tabla de consulta

Metadatos	Descripción	Devueltos por getQuery-TableMetaData	Devueltos por findQuery-TableMetaData	Devueltos por queryEntities	Devueltos por queryRows
locales[]	Entornos locales para los que se definen nombres de visualización y descripciones de la tabla de consulta y los atributos.	Sí	Sí	No	No
Entorno local	Valor del parámetro del sistema `$LOCALE` que se obtiene del entorno local que se pasa a la API.	Sí	Sí	Sí	Sí
Nombre de visualización y descripción de la tabla de consulta	Nombres de visualización y descripciones para la tabla de consulta, que se proporcionan para todos los entornos locales definidos.	Sí	Sí	No	No
Nombres de visualización y descripciones de los atributos	Nombres de visualización y descripciones de los atributos, que se proporcionan para todos los entornos locales definidos.	Sí	Sí	No	No

Todos los métodos de la API de tabla de consulta de EJB que devuelven metadatos de tabla de consulta aceptan un parámetro de entorno local, como por ejemplo `FilterOptions.setLocale` y `MetaDataOptions.setLocale`. Este parámetro se debe establecer en el entorno local de Java que utiliza el cliente para presentar la información al usuario. Este parámetro de entorno local se utiliza para calcular el valor del parámetro del sistema `$LOCALE`, que se puede utilizar en filtros y criterios de selección. El entorno local que se devuelve contiene el propio entorno Java que se utiliza para `$LOCALE`.

Si se recuperan los nombres de visualización y las descripciones de una tabla de consulta específica, pase getLocale a los métodos relacionados para obtener los nombres de visualización y descripciones en el mismo entorno local que las

descripciones de las tareas. Por ejemplo, estas descripciones se adjuntan utilizando un criterio de selección de 'LOCALE=$LOCALE'.

Ejemplo

```
// el ejemplo siguiente muestra cómo se pueden recuperar metadatos
// para una tabla de consulta compuesta específica

...
// ejecutar la consulta
MetaDataOptions mdo = new MetaDataOptions("TASK", null, false, new Locale("en_US"));
List list = bfm.findQueryTableMetaData(mdo);

// obtener los metadatos de una tabla de consulta específica
// use bfm.getQueryTableMetaData(...)

// iterar a través de la lista de tablas de consulta que tienen TASK como tabla
// de consulta primaria
// => se devuelve como mínimo una tabla de consulta: la tabla de consulta
// predefinida TASK

Iterator iter = list.iterator();
while( iter.hasNext() ) {
 QueryTableMetaData md = (QueryTableMetaData) iter.next();
 Locale effectiveLocale = md.getLocale();
 String queryTableDisplayName = md.getDisplayName(effectiveLocale);
 System.out.println("found query table: " + queryTableDisplayName);
 List attributesList = md.getAttributeMetaData();
 Iterator attrIter = attributesList.iterator();
 while (attrIter.hasNext()) {
  AttributeMetaData amd = (AttributeMetaData) attrIter.next();
  String attributeDisplayName = amd.getDisplayName(effectiveLocale);
  System.out.println("\tattribute:" + attributeDisplayName);
 }
}
```

Entorno local más coincidente

Al especificar las condiciones en una tabla de consulta conectada, el uso del valor $LOCALE puede devolver resultados imprevistos si el entorno local especificado no coincide exactamente con los metadatos. Por ejemplo, si se pasa un entorno local es_ES con una consulta sobre una tabla de consulta que tenga metadatos que especifiquen el idioma como es, el resultado devuelto será null.

Para evitar estos casos, puede utilizar LOCALE=$LOCALE_BEST_MATCH, que aplica un algoritmo de mejor coincidencia para calcular el entorno local real utilizado en la consulta. Por ejemplo, una consulta con el entorno local es_ES sobre una tabla de consulta en el idioma es se realiza utilizando es.

No puede especificar ningún otro operador lógico o de comparación en la condición LOCALE=$LOCALE_BEST_MATCH. Sólo puede utilizar la condición de entorno local de mejor coincidencia en tablas de consulta conectadas, especificándola como condición sobre otros resultados de consulta en un error.

Internacionalización para metadatos de tabla de consulta

Se da soporte a la internacionalización de metadatos de tabla de consulta.

Se pueden proporcionar nombres de visualización y descripciones de las tablas de consulta compuestas en distintos entornos locales. Por ejemplo, una tabla de consulta compuesta puede definir un nombre de visualización para la tabla de consulta en el entorno local en_US, el entorno local de y el entorno local por omisión. Esto se realiza cuando la tabla de consulta se desarrolla utilizando Query

Table Builder. Para desplegar tablas de consulta con descripciones y nombres de visualización localizados, se debe utilizar la opción `-deploy jarFile` cuando la tabla de consulta se despliega en el contenedor de Business Process Choreographer.

En términos de manejo de entornos locales, el comportamiento de los métodos de la API de tabla de consulta, queryEntities y queryRows, y los métodos de metadatos de la API de tabla de consulta, getQueryTableMetaData y findQueryTableMetaData, es similar al proporcionado por los paquetes de recursos Java.

Para que los nombres de visualización y las descripciones de los metadatos de la tabla de consulta sean coherentes con el contenido de la tabla de consulta, el valor del parámetro del sistema `$LOCALE` depende de los entornos locales para los que se especifiquen los nombres de visualización y las descripciones en la tabla de consulta.

Ejemplo

Considere el siguiente escenario de un cliente que visualiza listas de tareas o listas de procesos y crea una solicitud para consultar una tabla de consulta.

- El cliente no ha especificado el entorno local que utiliza para presentar información al usuario. Es probable que la aplicación no esté habilitada para distintos idiomas.
  - Se especifica un entorno local por omisión en la tabla de consulta para nombres de visualización y descripciones. Este es el caso para todas las tablas de consulta compuestas o suplementarias que se crean con la versión actual de Query Table Builder. Por lo tanto, el valor de `$LOCALE` se establece en `default`.
  - La tabla de consulta no especifica nombres de visualización o descripciones en la tabla de consulta para el entorno local por omisión. Este es el caso para todas las tablas de consulta predefinidas y para todas las tablas de consulta que se despliegan utilizando la opción `-deploy qtdFile`. El valor de `$LOCALE` se basa en el método de paquete de recursos Java.
- El cliente especificado en el entorno local que se utiliza para presentar la información al usuario. Por ejemplo, este es el caso cuando se utiliza la API de REST para tablas de consulta.
  - Los nombres de visualización y las descripciones se especifican en la tabla de consulta. El método del paquete de recursos Java se utiliza para calcular el valor de `$LOCALE`, en función del entorno local que pasa el cliente.
  - No se especifican nombres de visualización ni descripciones en la tabla de consulta. El valor de `$LOCALE` se establece en el valor que pasa el cliente.

Entorno local más coincidente

Al especificar las condiciones en una tabla de consulta conectada, el uso del valor `$LOCALE` puede devolver resultados imprevistos si el entorno local especificado no coincide exactamente con los metadatos. Por ejemplo, si se pasa un entorno local `es_ES` con una consulta sobre una tabla de consulta que tenga metadatos que especifiquen el idioma como `es`, el resultado devuelto será `null`.

Para evitar estos casos, puede utilizar `LOCALE=$LOCALE_BEST_MATCH`, que aplica un algoritmo de mejor coincidencia para calcular el entorno local real utilizado en la consulta. Por ejemplo, una consulta con el entorno local `es_ES` sobre una tabla de consulta en el idioma `es` se realiza utilizando `es`.

No puede especificar ningún otro operador lógico o de comparación en la condición LOCALE=$LOCALE_BEST_MATCH. Sólo puede utilizar la condición de entorno local de mejor coincidencia en tablas de consulta conectadas, especificándola como condición sobre otros resultados de consulta en un error.

Tareas relacionadas

"Creación de tablas de consulta para Business Process Choreographer Explorer" en la página 376
Puede utilizar tablas de consulta en lugar de la API query EJB para mejorar el rendimiento de Business Process Choreographer Explorer. Para crear las tablas de consulta, utilice el generador de tablas de consulta.

Tablas de consulta y rendimiento de consulta

Las tablas de consulta presentan un modelo de programación limpio para desarrollar aplicaciones cliente que recuperan listas de tareas de usuario y procesos empresariales en Business Process Choreographer. Si se utilizan tablas de consulta, mejora el rendimiento. Se proporciona información sobre los parámetros de API de tabla de consulta y otros factores que afectan al rendimiento.

Los tiempos de respuesta de consulta de las tablas de consulta dependen principalmente de las opciones de autorización, los filtros, y los criterios de selección que se utilicen. A continuación se detallan algunos consejos generales de rendimiento que se deben tener en cuenta.

- Las opciones de autorización tienen un impacto considerable en el rendimiento. Habilite la autorización utilizando el menor número de opciones posible como, por ejemplo, elementos individuales y de trabajo de grupo. Evite utilizar elementos de trabajo heredados. Las opciones de autorización pueden restringirse aún más cuando se ejecuta la consulta. Además, si no es necesario, especifique que no se requiere la autorización que utiliza los elementos de trabajo.
- Si esta autorización fuera necesaria, especifique un filtro de autorización. Por ejemplo, para permitir sólo objetos en la tabla de consulta que tenga un elemento de trabajo de propietario potencial, utilice WI.REASON=REASON_POTENTIAL_OWNER.
- El filtrado de la tabla de consulta primaria es eficiente, por ejemplo, para permitir sólo tareas en estado preparado en la tabla de consulta donde TASK sea la tabla de consulta primaria.
- Los filtros en la tabla de consulta, así como también los filtros de consulta, que son los filtros que se pasan cuando se ejecuta la consulta, resultan menos eficientes como filtros primarios, en términos de rendimiento.
- Evite, donde sea posible, utilizar parámetros en los filtros y los criterios de selección.
- Evite utilizar los operadores LIKE en los filtros y los criterios de selección.

Definición de tabla de consulta compuesta

En la tabla siguiente se proporciona información acerca del efecto sobre el rendimiento de consulta de las opciones que se definen en las tablas de consulta compuestas. También se proporciona información sobre otros temas relacionados con las definiciones de tabla de consulta compuesta. El efecto que se proporciona en la columna Efecto sobre el rendimiento es un efecto promedio sobre el rendimiento y las observaciones reales del efecto pueden variar.

Tabla 56. Efecto sobre el rendimiento de consulta de las opciones de la tabla de consulta compuesta

Objeto o tema	Efecto sobre el rendimiento	Descripción
Filtro de tabla de consulta	Negativo	Los filtros de las tablas de consulta son los filtros con el efecto más negativo sobre el rendimiento de consulta. Estos filtros normalmente no pueden utilizar índices definidos en la base de datos.
Filtro de tabla de consulta primaria	Positivo	Un filtro en la tabla de consulta primaria proporciona filtrado de alto rendimiento en una etapa muy temprana del cálculo del conjunto de resultados de la consulta. Se sugiere restringir el contenido de la tabla de consulta utilizando un filtro de tabla de consulta primaria.
Filtro de autorización	Positivo	Un filtro de autorización puede mejorar el rendimiento de la consulta, de la misma forma en la que el filtro de tabla de consulta primaria lo mejora. Si es posible, se debe aplicar un filtro de autorización. Por ejemplo, si no se deben considerar elementos de trabajo de lector, especifique `WI.REASON=REASON_READER`.
Criterios de selección	Ninguno	Algunas relaciones entre tabla de consulta primaria y tabla de consulta conectada requieren la definición de un criterio de selección a fin de cumplir la relación de uno a uno o de uno a cero. Un criterio de selección normalmente tiene poco efecto sobre el rendimiento ya que se evalúa sólo para un pequeño número de filas.
Parámetros	Ninguno	Actualmente, la utilización de parámetros en las tablas de consulta no tiene ningún efecto negativo sobre el rendimiento. Sin embargo, se deben utilizar parámetros sólo si es necesario.
Autorización basada en instancia	Negativo	Si se utiliza autorización basada en instancia, cada objeto de la tabla de consulta se debe comprobar respecto a la existencia de un elemento de trabajo. Los elementos de trabajo se representan como entradas en la tabla de consulta WORK_ITEM. Esta verificación afecta al rendimiento.
Autorización basada en instancia: • todos • personas • grupos • heredados	Negativo	Cada tipo de elemento de trabajo que se especifica para utilizar en la tabla de consulta tiene un efecto sobre el rendimiento. Las aplicaciones con consultas de gran volumen sólo deben utilizar elementos de trabajo Persona y de grupo, o sólo uno de ellos. Los elementos de trabajo heredados normalmente no son necesarios, específicamente al definir listas de tareas que devuelven tareas de usuario que representan tareas a realizar. Se deben utilizar sólo cuando sea obvio que se necesitan, por ejemplo, para devolver listas de tareas que pertenecen a un proceso empresarial donde es posible que una persona tenga acceso de lectura basado en la autorización para el proceso empresarial adjunto.
Autorización basada en rol o ninguna autorización	Ninguno	Si se utiliza autorización basada en rol o ninguna autorización, las comprobaciones respecto a elementos de trabajo no se realizan.

Tabla 56. Efecto sobre el rendimiento de consulta de las opciones de la tabla de consulta compuesta (continuación)

Objeto o tema	Efecto sobre el rendimiento	Descripción
Número de atributos definidos	Ninguno actualmente	Actualmente, el número de atributos contenidos en una tabla de consulta no tiene ningún efecto sobre el rendimiento. Sin embargo, sólo estos atributos necesarios deben formar parte de una tabla de consulta.

API de tabla de consulta

En la tabla siguiente se proporciona información acerca del efecto sobre el rendimiento de consulta de las opciones que se especifican en la API de tabla de consulta. El efecto que se proporciona en la columna Efecto sobre el rendimiento es un efecto promedio sobre el rendimiento y las observaciones reales del efecto pueden variar.

Tabla 57. Efecto sobre el rendimiento de consulta de las opciones de la API de tabla de consulta

Opción	Efecto sobre el rendimiento	Descripción
Atributos seleccionados	Negativo (menos es mejor)	El número de atributos que se seleccionan al ejecutar una consulta en una tabla de consulta afecta al número que la base de datos y el entorno de ejecución de la tabla de consulta de Business Process Choreographer deben procesar. Además, para tablas de consulta compuestas, es necesario recuperar la información de las tablas de consulta conectadas sólo si éstas se especifican en los atributos seleccionados o se hace referencia a ellas en el filtro de tabla de consulta o en el filtro de consulta.
Filtro de consulta	Negativo	Si se especifica, el filtro de consulta actualmente tiene el mismo efecto sobre el rendimiento que el filtro de tabla de consulta. Sin embargo, es recomendable especificar los filtros en las tablas de consulta en lugar de pasarlos a la API de tabla de consulta.
Atributos de ordenación	Negativo	La ordenación de los conjuntos de resultados de consulta es una operación costosa y si se utiliza la ordenación se restringen las optimizaciones de base de datos. Si no es necesaria, se debe evitar la ordenación. Sin embargo, la mayoría de las aplicaciones requieren ordenación.
Umbral	Positivo	La especificación de un umbral puede mejorar considerablemente el rendimiento de las consultas. Es recomendable especificar siempre un umbral.
Recuento de saltos	Negativo	Saltar un número determinado de objetos en el conjunto de resultados de consulta es costoso y sólo se debe realizar si es necesario, por ejemplo, cuando se pagina un resultado de consulta.
Huso horario	Ninguno	El valor de huso horario no tiene ningún efecto sobre el rendimiento.
Entorno local	Ninguno	El valor del entorno local no tiene ningún efecto sobre el rendimiento.

Tabla 57. Efecto sobre el rendimiento de consulta de las opciones de la API de tabla de consulta (continuación)

Opción	Efecto sobre el rendimiento	Descripción
Filas Distinct	Negativo	La utilización de distinct en consultas tiene cierto efecto sobre el rendimiento, pero es posible que sea necesario a fin de recuperar filas no duplicadas. Esta opción afecta sólo a las consultas basadas en fila y se ignora en caso contrario.
Consultas de recuento	Positivo	Si sólo es necesario el número total de entidades o el número de filas de una consulta específica, es decir, no es necesario el contenido de todas las entradas de la tabla de consulta, se debe utilizar el método queryEntityCount o queryRowCount. El entorno de ejecución de Business Process Choreographer puede aplicar optimizaciones que son válidas sólo para consultas de recuento.

Otras consideraciones

Otros factores que se deben considerar respecto al rendimiento son los siguientes:

Tabla 58. Rendimiento de tabla de consulta: Otras consideraciones

Elemento	Descripción
Número de tablas de consulta en el sistema	El número de tablas de consulta que se despliegan en un contenedor de Business Process Choreographer no afecta al rendimiento de las consultas de tabla de consulta. Además, actualmente no afecta a la navegación de las instancias de proceso empresarial, ni tiene ningún efecto sobre las operaciones de completar o reclamar sobre tareas de usuario. Debido a la capacidad de mantenimiento, mantenga el número de tablas de consulta en un nivel razonable. Normalmente, una tabla de consulta representa una lista de tareas o una lista de procesos que se visualiza en la interfaz de usuario.

Tabla 58. Rendimiento de tabla de consulta: Otras consideraciones (continuación)

Elemento	Descripción
Ajuste de base de datos	Aunque se utiliza SQL optimizado para acceder al contenido de una tabla de consulta, sigue siendo necesario implementar los ajustes de base de datos en una base de datos de Business Process Choreographer: • La memoria de base de datos se debe establecer en un máximo, teniendo en cuenta otros procesos que estén en ejecución en el servidor de bases de datos, así como las restricciones de hardware. • Las estadísticas de la base de datos deben estar actualizadas y se deben actualizar periódicamente. Normalmente, estos procedimientos ya se implementan en topologías grandes. Por ejemplo, recopile estadísticas de base de datos para el optimizador una vez por semana a fin de que reflejen los cambios de los datos de la base de datos. • Los sistemas de bases de datos proporcionan herramientas para reorganizar (o desfragmentar) los contenedores de datos. El diseño físico de los datos de una base de datos puede también afectar el rendimiento de consulta y las vías de acceso de las consultas. • Los índices óptimos son la clave para un buen rendimiento de consulta. Business Process Choreographer se proporciona con índices predefinidos que están optimizados para la navegación de procesos y el rendimiento de consulta de los escenarios típicos. En entornos personalizados, es posible que sean necesarios índices adicionales a fin de dar soporte a consultas de lista de procesos o de tareas de gran volumen. Utilice las herramientas que proporciona la base de datos a fin de dar soporte a las consultas que se ejecutan en una tabla de consulta.

Creación de tablas de consulta para Business Space

En Query Table Builder, puede utilizar la definición de tabla de consulta compuesta que tiene propiedades predefinidas para crear tablas de consulta para Business Space.

Antes de empezar

Query Table Builder está disponible como un plug-in de Eclipse y se puede descargar del sitio de SupportPacs de WebSphere Business Process Management. Busque la sección PA71 WebSphere Process Server - Query Table Builder. Para acceder al enlace, consulte la sección de referencias relacionadas de este tema.

Procedimiento

1. En Query Table Builder, pulse el botón derecho del ratón en el proyecto y, a continuación, seleccione **Nuevo → Definición de consulta compuesta para Business Space**. Siga las instrucciones del asistente para crear una definición de tabla de consulta. La nueva definición de tabla de consulta consta de las propiedades predefinidas. Si es necesario, añada más propiedades a la definición de tabla de consulta y despliegue el archivo de definición de tabla de consulta en WebSphere Process Server.

   Nota: Los nombres a los que ha dado las propiedades en Query Table Builder se utilizan como los nombres de las propiedades de tarea en Business Space para Choreographer.

2. Después de haber creado los archivos de definición de tabla de consulta y de haberlos desplegado, puede configurarlos en Business Space. Por ejemplo, si ha desplegado un archivo de definición de tabla de consulta para el widget Lista de tareas:
   a. Abra el menú de widget y seleccione **Configurar** y, a continuación, el separador **Contenido**.
   b. En el separador **Contenido**, abra la lista desplegable **Seleccionar lista de tareas a visualizar** para visualizar las listas que puede dejar disponibles para el usuario del widget. Seleccione **Añadir listas de tareas**. La definición de tabla de consulta que ha desplegado debe estar disponible en esta lista para seleccionarla.

   Si la definición de tabla de consulta no está disponible, necesitará volver a Query Table Builder y comprobar si el archivo de definición se ha definido y desplegado correctamente.

Creación de tablas de consulta para Business Process Choreographer Explorer

Puede utilizar tablas de consulta en lugar de la API query EJB para mejorar el rendimiento de Business Process Choreographer Explorer. Para crear las tablas de consulta, utilice el generador de tablas de consulta.

Antes de empezar

El generador de tablas de consulta está disponible como un plug-in Eclipse y se puede descargar desde el sitio de SupportPacs de WebSphere Business Process Management. Busque la sección PA71 WebSphere Process Server - Query Table Builder. Para acceder al enlace, consulte la sección de referencias relacionadas de este tema.

Procedimiento

1. En el generador de tablas de consulta, pulse con el botón secundario del ratón el proyecto, seleccione **Nuevo → Definición de consulta compuesta para Business Space**. Esta opción garantiza que seleccionen previamente todas las columnas necesarias para Business Process Choreographer Explorer.
2. Siga las instrucciones del asistente para crear una definición de la tabla de consultas. Si es necesario, añada más propiedades a la definición de la tabla de consultas. Considere los aspectos siguientes cuando defina la tabla de consultas:

   Criterios de filtros
   Cuando cree vistas en Business Process Choreographer Explorer basándose en tablas de consultas, no puede especificar filtros ni variables adicionales para los criterios de búsqueda. Debe especificar estos criterios de filtro y los parámetros para las variables cuando cree la tabla de consultas.

   Puede utilizar una tabla de consultas para más de una vista en Business Process Choreographer Explorer utilizando parámetros en la definición de la tabla de consultas. Si desea más flexibilidad, también puede especificar si los valores predeterminados de los parámetros se pueden sobrescribir cuando se ejecuta la consulta de la vista personalizada.

   Autorización
   Cuando cree vistas en Business Process Choreographer Explorer basadas en tablas de consulta, no puede filtrar los criterios de búsqueda basados en el rol de usuario. Debe establecer los criterios de filtro para

los roles de usuario cuando defina la tabla de consultas. Para las tablas de consultas primarias basadas en la información de plantilla, utilice la autorización basada en instancias como el tipo de autorización y no la autorización basada en rol. Para las tablas de consultas primarias basadas en la información de instancia, especifique el filtro apropiado de autorización basado en instancias.

Internacionalización

Cuando defina propiedades en el generador de tablas de consultas, también puede especificar los nombres y las descripciones de estas propiedades en idiomas diferentes. Cuando se ejecute la consulta para la vista personalizada, Business Process Choreographer Explorer utiliza la traducción apropiada para el valor de idioma del navegador.

Nombre de visualización y descripción para la definición de tabla de consultas

En el generador de tablas de consultas, puede proporcionar un nombre de visualización y una descripción para todos los idiomas soportados por la vista.

Nombre de visualización y descripción para las columnas

Durante el tiempo de ejecución, Business Process Choreographer Explorer recupera los nombres de columna internacionalizados apropiados visualizados en una lista de resultados. Para las columnas que proceden de la tabla de consultas primarias como, por ejemplo, PIID, Business Process Choreographer Explorer utiliza la traducción que ya está disponible para todos los idiomas soportados.

Para las columnas que proceden de una tabla de consultas adjuntas como, por ejemplo, QUERY_PROPERTY, debe proporcionar nombres de visualización y descripciones en el generador de tablas de consultas en todos los idiomas soportados por la empresa.

Nombres de tarea y descripción

Si ha internacionalizado nombres de tarea y descripciones en WebSphere Integration Developer, se visualizan en Business Process Choreographer Explorer de acuerdo con los valores de idioma y país del navegador. Si los valores del navegador no coinciden con los valores definidos en el modelo de proceso, se utiliza la traducción del idioma predeterminado.

Criterios de clasificación

Cuando defina criterios de clasificación para una definición de tabla de consultas, debe ser consciente de que varias propiedades, por ejemplo, el estado de proceso, se almacenan como valores enteros, mientras que Business Process Choreographer Explorer los visualiza como series traducidas en la lista resultante. Esto podría generar resultados inesperados de clarificación en algunos idiomas.

La nueva definición de tabla de consultas consta de las propiedades predefinidas y de las propiedades adicionales que defina.

Qué hacer a continuación

Despliegue y pruebe la definición de la consulta en el generador de tablas de consultas en un servidor de aplicaciones. Si este es el servidor al que está

conectado Business Process Choreographer Explorer, ahora puede utilizar la tabla de consultas cuando personalice Business Process Choreographer Explorer para su propio uso, o para distintos grupos de usuarios. Si Business Process Choreographer Explorer está conectado a un servidor diferente, debe volver a desplegar la tabla de consultas en el servidor apropiado antes de poder utilizarlo para crear vistas personalizadas.

Conceptos relacionados

"Autorización para tablas de consulta" en la página 343
Puede utilizar la autorización basada en instancia, la autorización basada en rol o ninguna autorización al ejecutar consultas en tablas de consulta.

"Filtros y criterios de selección de tablas de consulta" en la página 338
Los filtros y los criterios de selección se definen durante el desarrollo de la tabla de consulta utilizando Query Table Builder, que utiliza una sintaxis similar a las cláusulas WHERE de SQL. Utilice estos filtros y criterios de selección claramente definidos para especificar las condiciones que se basan en los atributos de las tablas de consulta.

"Internacionalización para metadatos de tabla de consulta" en la página 369
Se da soporte a la internacionalización de metadatos de tabla de consulta.

API de consulta EB de Business Process Choreographer

Utilice el método query o el método queryAll de la API de servicio para recuperar la información almacenada acerca de los procesos empresariales y tareas.

Todos los usuarios pueden llamar al método query y éste devuelve las propiedades de los objetos para los que existen elementos de trabajo. Sólo pueden llamar al método queryAll los usuarios que tienen uno de los siguientes roles de Java EE: BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor o TaskSystemMonitor. Este método devuelve las propiedades de todos los objetos almacenados en la base de datos.

Todas las consultas de la API se correlacionan con consultas SQL. La forma de la consulta SQL generada depende de los aspectos siguientes:

- La consulta ha sido invocada por alguien que tiene uno de los roles de Java EE.
- Los objetos que se consultan. Se proporcionan vistas de bases de datos predefinidas para que se consulten las propiedades de objeto.
- La inserción de una cláusula, condiciones de unión y condiciones específicas del usuario para el control de accesos.

Puede incluir propiedades personalizadas y propiedades de variables en consultas. Si incluye varias propiedades personalizadas o propiedades de variables en la consulta, se producen uniones automáticas en la tabla de base de datos correspondiente. En función del sistema de base de datos, estas llamadas a `query()` podrían tener implicaciones en el rendimiento.

También puede almacenar consultas en la base de datos de Business Process Choreographer utilizando el método createStoredQuery. Proporcione el criterio de consulta cuando defina la consulta almacenada. El criterio se aplica dinámicamente cuando se ejecuta la consulta almacenada, esto es, los datos se ensamblan durante la ejecución. Si la consulta almacenada contiene parámetros, estos se resuelven también cuando se ejecuta la consulta.

Para obtener más información sobre las API de Business Process Choreographer, consulte el Javadoc en el paquete com.ibm.bpe.api para los métodos relativos a procesos y el paquete com.ibm.task.api para los métodos relativos a tareas.

Sintaxis del método query de la API

La sintaxis de las consultas de la API de Business Process Choreographer es similar a la de las consultas SQL. Una consulta puede incluir una cláusula select, una cláusula where, una cláusula order-by, un parámetro skip-tuples, un parámetro threshold y un parámetro time-zone.

La sintaxis de la consulta depende del tipo de objeto. En la tabla siguiente se muestra la sintaxis de cada uno de los distintos tipos de objeto.

Tabla 59. Sintaxis de consulta para diferentes tipos de objeto

Objeto	Sintaxis
Plantilla de proceso	`ProcessTemplateData[] queryProcessTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);`
Plantilla de tarea	`TaskTemplate[] queryTaskTemplates (java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer threshold, java.util.TimeZone timezone);`
Datos de proceso de llamada y datos relacionados con tareas	`QueryResultSet query (java.lang.String selectClause, java.lang.String whereClause, java.lang.String orderByClause, java.lang.Integer skipTuples java.lang.Integer threshold, java.util.TimeZone timezone);`

Cláusula select:

La cláusula select de la función de consulta identifica las propiedades de objeto que una consulta ha de devolver.

La cláusula select describe el resultado de la consulta. Especifica una lista de los nombres que identifican las propiedades del objeto (las columnas del resultado) que se han de devolver. Su sintaxis es parecida a la de una cláusula SELECT de SQL; utiliza comas para separar las partes de la cláusula. Cada parte de la cláusula debe especificar una columna de una de las vistas predefinidas. Las columnas deben estar totalmente especificadas por nombre de vista y nombre de columna. Las columnas devueltas en el objeto QueryResultSet aparecerán con el mismo orden que las columnas especificadas en la cláusula select.

La cláusula select no tiene soporte para las funciones de agregación de SQL, como AVG(), SUM(), MIN() o MAX().

Para seleccionar las propiedades de varios pares de nombre y valor como, por ejemplo, las propiedades personalizadas y las propiedades de variables que se pueden consultar, añada un contador de un dígito al nombre de vista. Este contador puede tomar los valores de 1 a 9.

Ejemplos de cláusulas select

- "WORK_ITEM.OBJECT_TYPE, WORK_ITEM.REASON"

  Obtiene los tipos de objeto de los objetos asociados y las razones de asignación de los elementos de trabajo.
- "DISTINCT WORK_ITEM.OBJECT_ID"

Obtiene todos los ID de objetos, sin duplicados, para los que el llamante tiene un elemento de trabajo.

- "ACTIVITY.TEMPLATE_NAME, WORK_ITEM.REASON"

  Obtiene los nombres de las actividades para las que el llamante tiene elementos de trabajo y los motivos de asignación.
- "ACTIVITY.STATE, PROCESS_INSTANCE.STARTER"

  Obtiene los estados de las actividades y los iniciadores de sus instancias de proceso asociadas.
- "DISTINCT TASK.TKIID, TASK.NAME"

  Obtiene todos los ID y nombres de tareas, sin duplicados, para los que el llamante tiene un elemento de trabajo.
- "TASK_CPROP1.STRING_VALUE, TASK_CPROP2.STRING_VALUE"

  Obtiene los valores de las propiedades personalizadas que se especifican con más detalle en la cláusula where.
- "QUERY_PROPERTY1.STRING_VALUE, QUERY_PROPERTY2.INT_VALUE

  Obtiene los valores de las propiedades de variables que se pueden consultar. Estas partes se especifican más en la cláusula where.
- "COUNT(DISTINCT TASK.TKIID)"

  Cuenta el número de elementos de trabajo para las taeras exclusivas que satisfacen la cláusula where.

Cláusula where:

La cláusula where de la función de consulta describe los criterios de filtro que se aplicarán al dominio de consulta.

La sintaxis de una cláusula where es parecida a la de una cláusula WHERE de SQL. No es necesario añadir explícitamente una cláusula from de SQL o predicados de unión a la cláusula where de la API, estas construcciones se añaden automáticamenet cuando se ejecuta la consulta. Si no quiere aplicar criterios de filtro, debe especificar `null` para la cláusula where.

La sintaxis de la cláusula where tiene soporte para:

- Palabras clave: AND, OR, NOT
- Operadores de comparación: =, <=, <, <>, >,>=, LIKE

  La operación LIKE admite los caracteres comodín definidos para la base de datos consultada.
- Operación de definición: IN

También se aplican las normas siguientes:

- Especifique las constantes de ID de objeto como `ID('string-rep-of-oid')`.
- Especifique las constantes binarias como `BIN('UTF-8 string')`.
- Utilice constantes simbólicas en lugar de enumeraciones de enteros. Por ejemplo, en vez de especificar una expresión de estado de actividad `ACTIVITY.STATE=2`, especifique `ACTIVITY.STATE=ACTIVITY.STATE.STATE_READY`.
- Si el valor de la propiedad en la sentencia de comparación contiene apóstrofos ('), doble las comillas; por ejemplo, `"TASK_CPROP.STRING_VALUE='d''automatisation'"`.
- Consulte las propiedades de varios pares de nombre y valor como, por ejemplo, las propiedades personalizadas, añadiendo un sufijo de un dígito al nombre de vista. Por ejemplo: `"TASK_CPROP1.NAME='prop1' AND "TASK_CPROP2.NAME='prop2'"`

- Especifique las constantes de indicación de la hora como `TS('yyyy-mm-ddThh:mm:ss')`. para consultar la fecha actual, especifique `CURRENT_DATE` como la indicación de la hora.

  Debe especificar como mínimo un valor de fecha o de hora en la indicación de la hora:

  – Si especifica solamente una fecha, el valor de hora se establece en cero.
  – Si especifica solamente una hora, la fecha se establece en la fecha actual.
  – Si especifica una fecha, el año debe constar de cuatro dígitos. Los valores de mes y día son opcionales. Si faltan los valores de mes y día se establecen en 01. Por ejemplo, `TS('2003')` es igual que `TS('2003-01-01T00:00:00')`.
  – Si especifica una hora, estos valores se expresan en el sistema de 24 horas. Por ejemplo, si la fecha actual es 1 de Enero de 2003, `TS('T16:04')` o `TS('16:04')` es igual que `TS('2003-01-01T16:04:00')`.

Ejemplos de cláusulas where

- Comparación de un ID de objeto con un ID existente

  ```
  "WORK_ITEM.WIID = ID('_WI:800c00ed.df8d7e7c.feffff80.38')"
  ```

  Este tipo de cláusula where suele crearse de forma dinámica con un ID de objeto existente a partir de una llamada anterior. Si este ID de objeto se guarda en una variable *wiid1*, la cláusula podría construirse como:

  ```
  "WORK_ITEM.WIID = ID('" + wiid1.toString() + "')"
  ```

- Utilización de las indicaciones de la hora

  ```
  "ACTIVITY.STARTED >= TS('2002-06-1T16.00.00')"
  ```

- Uso de constantes simbólicas

  ```
  "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER"
  ```

- Uso de los valores booleanos true y false

  ```
  "ACTIVITY.BUSINESS_RELEVANCE = TRUE"
  ```

- Uso de propiedades personalizadas

  ```
  "TASK_CPROP1.NAME = 'prop1' AND " TASK_CPROP1.STRING_VALUE = 'v1' AND
   TASK_CPROP2.NAME = 'prop2' AND " TASK_CPROP2.STRING_VALUE = 'v2'"
  ```

Cláusula order-by:

La cláusula order-by de la función de consulta especifica los criterios de clasificación del conjunto de resultados de la consulta.

Puede especificar una lista de columnas desde las vistas mediante las cuáles se clasifica el resultado. Estas columnas deben estar plenamente cualificadas por el nombre de la vista y de la columna.

La sintaxis de la cláusula order-by es similar a la de una cláusula order-by de SQL; utilice comas para separar cada componente de la cláusula. También puede especificar `ASC` para clasificar las columnas en orden ascendente y `DESC` para clasificarlas en orden descendente. Si no quiere clasificar el conjunto de resultados de consulta, debe especificar `null` para la cláusula order-by.

Se aplican criterios de clasificación en el servidor, es decir, se utiliza el entorno local del servidor para realizar la clasificación. Si especifica más de una columna, el conjunto de resultados de la consulta se clasifica por los valores de la primera columna, luego por los valores de la segunda columna y así sucesivamente. No puede especificar las columnas en la cláusula order-by por posición como puede en una consulta SQL.

Ejemplos de cláusulas order-by

- "PROCESS_TEMPLATE.NAME"

  Clasifica el resultado de la consulta en orden alfabético por el nombre de la plantilla de proceso.
- "PROCESS_INSTANCE.CREATED, PROCESS_INSTANCE.NAME DESC"

  Clasifica el resultado de la consulta por la fecha de creación y, para una fecha determinada, clasifica el resultado en orden alfabético inverso por el nombre de instancia de proceso.
- "ACTIVITY.OWNER, ACTIVITY.TEMPLATE_NAME, ACTIVITY.STATE"

  Clasifica el resultado de la consulta por el propietario de la actividad, luego por nombre de plantilla de la actividad y luego por el estado de la actividad.

Parámetro skip-tuples:

El parámetro skip-tuples especifica el número de tuples de tipo query-result-set desde el principio del conjunto de resultados de la consulta que se deben pasar por alto y no devolverse al llamante en el conjunto de resultados de la consulta.

Utilice este parámetro con el parámetro threshold para implementar la paginación de una aplicación de cliente, por ejemplo, para recuperar los primeros 20 elementos, a continuación, los siguientes 20 elementos, etc.

Si este parámetro se establece en `null` y no se establece el parámetro threshold, se devuelven todos los tuples cualificados.

Ejemplo de parámetro skip-tuples

- new Integer(5)

  Especifica que no deben devolverse los cinco primeros tuples calificados.

Parámetro threshold:

El parámetro de umbral (threshold) de la función de consulta restringe el número de objetos devueltos del servidor al cliente en el conjunto de resultados de consulta.

Debido a que los conjuntos de resultados de la consulta en los escenarios de producción pueden contener miles o incluso millones de elementos, especifique un valor para el parámetro threshold. Si establece el parámetro threshold en consecuencia, la consulta de base de datos es más rápida y es necesario transferir menos datos del servidor al cliente. El parámetro threshold puede ser útil, por ejemplo, en una interfaz gráfica de usuario en la que solamente deben visualizarse un número reducido de elementos.

Si este parámetro se establece en `null` y no se establece el parámetro skip-tuples, se devuelven todos los objetos cualificados.

Ejemplo de un parámetro threshold

- new Integer(50)

  Especifica que han de devolverse 50 tuples cualificados.

Parámetro timezone:

El parámetro time-zone de la función de consulta define el huso horario para las constantes de indicación de la hora de la consulta.

Los husos horarios del cliente que inicia la consulta y el servidor que la procesa, pueden ser distintos. Utilice el parámetro de huso horario, time-zone, para especificar el huso horario de las constantes de indicación de la hora que se utilizan, por ejemplo, en la cláusula where para especificar la hora local. Las fechas devueltas en el conjunto de resultados de la consulta tienen el mismo huso horario que las especificadas en la consulta.

Si el valor del parámetro se establece en `null`, se presupone que las constantes de timestamp tienen el formato UTC (Coordinated Universal Time).

Ejemplos de parámetros de huso horario

- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
                 (String)null,
                 (Integer)null,
                 java.util.TimeZone.getDefault() );
  ```

  Devuelve los ID de objeto de las actividades que se iniciaron después de las 17:40, hora local, el 1 de enero de 2005.
- ```
  process.query("ACTIVITY.AIID",
                "ACTIVITY.STARTED > TS('2005-01-01T17:40')",
                 (String)null, (Integer)null, (TimeZone)null);
  ```

  Devuelve los ID de objeto de las actividades que se iniciaron después de las 17:40, hora UTC, el 1 de enero de 2005. Esta especificación es, por ejemplo, 6 horas antes en la hora estándar del Este.

Filtro de los datos con variables en las consultas:

El resultado de una consulta devuelve los objetos que cumplen los criterios de la consulta. Quizá desee filtrar estos resultados según los valores de las variables.

Acerca de esta tarea

Puede definir las variables que un proceso utiliza durante la ejecución en el modelo de proceso. Para estas variables, puede declarar qué partes se pueden consultar.

Por ejemplo, Juan Torres, llama al número de servicio de su compañía aseguradora para averiguar el progreso de la reclamación del seguro de su coche dañado. El administrador de reclamaciones utiliza el identificador de cliente para encontrar la reclamación.

Procedimiento

1. Opcional: Enumere las propiedades de las variables de un proceso que se pueden consultar.

   Utilice el identificador de plantilla de proceso para identificar el proceso. Puede omitir este paso si sabe qué variables se pueden consultar.

   ```
   List variableProperties = process.getQueryProperties(ptid);
   for (int i = 0; i < variableProperties.size(); i++)
   {
      QueryProperty queryData = (QueryProperty)variableProperties.get(i);
      String variableName = queryData.getVariableName();
      String name         = queryData.getName();
      int mappedType      = queryData.getMappedType();
      ...
   }
   ```
2. Enumere las instancias de proceso con variables que cumplen los criterios de filtro.

Para este proceso, el identificador de cliente se crea como parte de la variable customerClaim que se puede consultar. Por lo tanto, puede utilizar el identificador de cliente para encontrar la reclamación.

```
QueryResultSet result = process.query
    ("PROCESS_INSTANCE.NAME, QUERY_PROPERTY.STRING_VALUE",
     "QUERY_PROPERTY.VARIABLE_NAME = 'customerClaim' AND " +
     "QUERY_PROPERTY.NAME = 'customerID' AND " +
     "QUERY_PROPERTY.STRING_VALUE like 'Torres%'",
      (String)null, (Integer)null,
      (Integer)null, (TimeZone)null );
```

Esta acción devuelve un conjunto de resultados de consulta que contiene los nombres de instancia de proceso y los valores de los identificadores de clientes cuyos identificadores empiezan con Torres.

Resultados de la consulta:

Un conjunto de resultados de consulta de una consulta de la API de Business Process Choreographer.

Los elementos del conjunto de resultados son propiedades de los objetos que satisfacen la cláusula where proporcionada por el emisor de la llamada y que este tiene autorización para ver. Puede leer elementos de lectura de forma relativa utilizando el método de API next de modo absoluto utilizando los métodos first y last. Como el cursor implícito de un conjunto de resultados de consulta está inicialmente posicionado antes del primer elemento, debe llamar a los métodos first o next antes de leer un elemento. Puede utilizar el método size para determinar el número de elementos del conjunto.

Un elemento de un conjunto de resultados de consulta comprende los atributos seleccionados de los elementos de trabajo y sus objetos referenciados asociados, como instancias de actividad y de proceso. El primer atributo (columna) de un elemento QueryResultSet especifica el valor del primer atributo especificado en la cláusula select de la petición de consulta. El segundo atributo (columna) de un elemento QueryResultSet especifica el valor del segundo atributo especificado en la cláusula select de la petición de consulta, y así sucesivamente.

Se pueden recuperar los valores de los atributos mediante la invocación de un método que es compatible con el tipo de atributo y mediante la especificación del índice de columna adecuado. La numeración de los índices de columna comienza por 1.

Tipo de atributo	**Método**
String	getString
OID	getOID
Timestamp	getTimestamp getString getTimestampAsLong
Integer	getInteger getShort getLong getString getBoolean

Tipo de atributo	Método
Boolean	getBoolean getShort getInteger getLong getString
byte[]	getBinary

Ejemplo:

Se ejecuta la consulta siguiente:

```
QueryResultSet resultSet = process.query("ACTIVITY.STARTED,
                                         ACTIVITY.TEMPLATE_NAME AS NAME,
                                         WORK_ITEM.WIID, WORK_ITEM.REASON",
                                         (String)null, (String)null,
                                         (Integer)null, (TimeZone)null);
```

El conjunto de resultados de consulta devuelto tiene cuatro columnas:
- La columna 1 es una indicación de la hora
- La columna 2 es una serie
- La columna 3 es un ID de objeto
- La columna 4 es un entero

Puede utilizar los métodos siguientes para recuperar los valores de atributo:

```
while (resultSet.next())
{
 java.util.Calendar activityStarted = resultSet.getTimestamp(1);
 String templateName = resultSet.getString(2);
 WIID wiid = (WIID) resultSet.getOID(3);
 Integer reason = resultSet.getInteger(4);
}
```

Puede utilizar los nombres de visualización del conjunto de resultados, por ejemplo, como cabeceras para imprimir una tabla. Son los nombres de columna de la vista o el nombre definido mediante la cláusula AS en la consulta. Puede utilizar el método siguiente para recuperar los nombres de visualización del ejemplo:

```
resultSet.getColumnDisplayName(1) returns "STARTED"
resultSet.getColumnDisplayName(2) returns "NAME"
resultSet.getColumnDisplayName(3) returns "WIID"
resultSet.getColumnDisplayName(4) returns "REASON"
```

Condiciones de acceso específicas del usuario

Las condiciones de acceso específicas del usuario se añaden cuando se genera la declaración de SQL SELECT desde la consulta de API. Estas condiciones garantizan que solo estos objetos se devuelven al llamante para satisfacer la condición especificada por el mismo y para la que está autorizado.

La condición de acceso que se añade depende de si el usuario es un administrador del sistema.

Consultas invocadas por usuarios que no sean administradores del sistema

La cláusula de SQL WHERE generada combina la cláusula where de la API con una condición de control de acceso que sea específica del usuario. La consulta recupera solo los objetos a los que el usuario está autorizado a acceder, es decir,

solo aquellos objetos para los que el usuario tiene un elemento de trabajo. Un elemento de trabajo representa la asignación de un usuario o grupo de usuarios a un rol de autorización de un objeto empresarial, como una tarea o proceso. Si, por ejemplo, el usuario Juan Torres es miembro del rol de propietarios potenciales de una tarea determinada, existe un objeto de elemento de trabajo que representa esta relación.

Por ejemplo, si un usuario, que no sea un administrador del sistema, consulta tareas, se añade la siguiente condición de acceso a la cláusula WHERE si no se han habilitado elementos de trabajo de grupo:

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'user'
      OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

Por tanto, si Juan Torres desea obtener una lista de tareas para las que es el propietario potencial, la cláusula where de la API podría tener este aspecto:

```
"WORK_ITEM.REASON == WORK_ITEM.REASON.REASON_POTENTIAL_OWNER"
```

Esta cláusula where de la API resulta en la siguiente condición de acceso en la declaración SQL:

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
AND ( WI.OWNER_ID = 'JuanTorres'
      OR WI.OWNER_ID = null AND WI.EVERYBODY = true)
AND WI.REASON = 1
```

Esto también significa que si Juan Torres desea ver las actividades y tareas para las que él es un lector de procesos o un administrador de procesos y para las que no tiene un elemento de trabajo, debe añadirse una propiedad de la vista PROCESS_INSTANCE a la cláusula select, where u order-by de la consulta, por ejemplo, PROCESS_INSTANCE.PIID.

Si se habilitan los elementos de trabajo de grupo, se añade una condición de acceso adicional a la cláusula WHERE que permite que un usuario acceda a objetos para los que el grupo tiene acceso.

Consultas invocadas por administradores del sistema

Los administradores del sistema pueden invocar el método query para recuperar objetos que tengan elementos de trabajo asociados. En este caso, se añade una unión con la vista WORK_ITEM a la consulta SQL generada, pero no se añade una condición de control de acceso para WORK_ITEM.OWNER_ID.

En este caso, la consulta SQL para tareas contiene lo siguiente:

```
FROM TASK TA, WORK_ITEM WI
WHERE WI.OBJECT_ID = TA.TKIID
```

Consultas queryAll

Este tipo de consulta solo se puede invocar por parte de los administradores del sistema o supervisores del sistema. No se añade ninguna condición para el control de acceso ni una unión a la vista WORK_ITEM. Este tipo de consulta devuelve todos los datos para todos los objetos.

Ejemplos de los métodos query y queryAll

Estos ejemplos muestran la sintaxis de varias consultas típicas de API y las declaraciones SQL asociadas que se generan al procesar la consulta.

Ejemplo: consulta de tareas en estado preparado:

Este ejemplo muestra cómo utilizar el método query para recuperar tareas con las que puede trabajar el usuario que ha iniciado la sesión.

Juan Torres desea obtener una lista de las tareas que se le han asignado. Para que un usuario pueda trabajar en una tarea, esta debe estar en estado preparado. El usuario que ha iniciado la sesión también debe tener un elemento de trabajo de propietario potencial para la tarea. El fragmento de código siguiente muestra la llamda al método query para esta consulta:

```
query( "DISTINCT TASK.TKIID",
           "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
             AND " +
           "TASK.STATE = TASK.STATE.STATE_READY AND " +
           "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
            (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Se adoptan las acciones siguientes cuando se genera la declaración SELECT:

- Se añade una condición para el control de acceso a la cláusula where. En este ejemplo se asume que los elementos de trabajo de grupo no están habilitados.
- Las constantes, como TASK.STATE.STATE_READY, se sustituyen por sus valores numéricos.
- Se añade una cláusula FROM y condiciones de unión.

El fragmento de código siguiente muestra la sentencia SQL que se genera a partir de la consulta de API:

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE WI.OBJECT_ID = TA.TKIID
   AND    TA.KIND IN ( 101, 105 )
   AND    TA.STATE = 2
   AND    WI.REASON = 1
   AND  ( WI.OWNER_ID = 'JohnSmith' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

Para restringir la consulta de API a tareas para un proceso determinado, por ejemplo, sampleProcess, la consulta tiene el aspecto siguiente:

```
query( "DISTINCT TASK.TKIID",
           "PROCESS_TEMPLATE.NAME = 'sampleProcess' AND "+
           "TASK.KIND IN ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING )
             AND " +
           "TASK.STATE = TASK.STATE.STATE_READY AND " +
           "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
            (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Ejemplo: consulta de tareas en estado reclamado:

En este ejemplo se muestra cómo utilizar el método query para recuperar las tareas que el usuario conectado ha reclamado.

El usuario, John Smith, desea buscar las tareas que ha reclamado y que todavía están es el estado reclamadas. La condición que especifica "reclamadas por John Smith" es TASK.OWNER = 'JohnSmith'. En el fragmento de código siguiente se muestra la llamada al método query de la consulta:

```
query( "DISTINCT TASK.TKIID",
           "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
           "TASK.OWNER = 'JohnSmith'",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

En el fragmento de código siguiente se muestra la sentencia SQL que se genera de la consulta de la API:

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE WI.OBJECT_ID = TA.TKIID
   AND    TA.STATE = 8
   TA.OWNER = 'JohnSmith'
   AND  ( WI.OWNER_ID = 'JuanTorres' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

Cuando se reclama una tarea, se crean elementos de trabajo para el propietario de la tarea. De manera que, un modo alternativo de formar la consulta para las tareas reclamadas de John Smith es añadir la condición siguiente a la consulta en lugar de utilizar `TASK.OWNER = 'JohnSmith'`:

```
WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER
```

La consulta entonces se parecerá al siguiente fragmento de código:

```
query( "DISTINCT TASK.TKIID",
           "TASK.STATE = TASK.STATE.STATE_CLAIMED AND " +
           "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_OWNER",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Se adoptan las acciones siguientes cuando se genera la declaración SELECT:

- Se añade una condición para el control de acceso a la cláusula where. En este ejemplo se asume que los elementos de trabajo de grupo no están habilitados.
- Las constantes, como `TASK.STATE.STATE_READY`, se sustituyen por sus valores numéricos.
- Se añade una cláusula FROM y condiciones de unión.

En el fragmento de código siguiente se muestra la sentencia SQL que se genera de la consulta de la API:

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE WI.OBJECT_ID = TA.TKIID
   AND    TA.STATE = 8
   AND    WI.REASON = 4
   AND  ( WI.OWNER_ID = 'JuanTorres' OR WI.OWNER_ID = null AND WI.EVERYBODY = true )
```

John está a punto de irse de vacaciones de modo que la jefe de su equipo, Anne Grant, desea verificar su carga de trabajo actual. Anne tiene derechos de administrador del sistema. La consulta que ella invoca es la misma que la que ha invocado John. No obstante, la sentencia SQL que se genera es distinta porque Anne es administradora. En el fragmento de código siguiente se muestra la sentencia SQL generada:

```
SELECT DISTINCT TASK.TKIID
   FROM   TASK TA, WORK_ITEM WI,
   WHERE  TA.TKIID = WI.OBJECT_ID =
   AND    TA.STATE = 8
   AND   TA.OWNER = 'JuanTorres')
```

Dado que Anne es administradora, no se añade una condición de control de accesos a la cláusula WHERE.

Ejemplo: consulta de escaladas:

Este ejemplo muestra cómo utilizar el método query para recuperar escaladas para el usuario que ha iniciado la sesión.

Cuando se escala una tarea, se crea un elemento de trabajo del destinatario de escalada. El usuario, Mary Jones, desea ver una lista de las tareas que se le han escalado. El fragmento de código siguiente muestra la llamda al método query para la consulta:

```
query( "DISTINCT ESCALATION.ESIID, ESCALATION.TKIID",
         "WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_ESCALATION_RECEIVER",
           (String)null, (String)null, (Integer)null, (TimeZone)null )
```

Se adoptan las acciones siguientes cuando se genera la declaración SELECT:

- Se añade una condición para el control de acceso a la cláusula where. En este ejemplo se asume que los elementos de trabajo de grupo no están habilitados.
- Las constantes, como TASK.STATE.STATE_READY, se sustituyen por sus valores numéricos.
- Se añade una cláusula FROM y condiciones de unión.

El fragmento de código siguiente muestra la sentencia SQL que se genera a partir de la consulta de API:

```
SELECT DISTINCT ESCALATION.ESIID, ESCALATION.TKIID
  FROM   ESCALATION ESC, WORK_ITEM WI
  WHERE  ESC.ESIID = WI.OBJECT_ID
  AND    WI.REASON  = 10
  AND
 ( WI.OWNER_ID = 'MaryJones' OR WI.OWNER_ID = null AND  WI.EVERYBODY = true )
```

Ejemplo: uso del método queryAll:

Este ejemplo muestra cómo utilizar el método queryAll para recuperar todas las actividades que pertenecen a una plantilla de proceso.

El método queryAll sólo está disponible para usuarios con derechos de administrador del sistema o de supervisor del sistema. El fragmento de código siguiente muestra la llamada al método queryAll para que la consulta recupere todas las actividades que pertenecen a la plantilla de proceso, sampleProcess:

```
queryAll( "DISTINCT ACTIVITY.AIID",
       "PROCESS_TEMPLATE.NAME = 'sampleProcess'",
       (String)null, (String)null, (Integer)null, (TimeZone)null )
```

El fragmento de código siguiente muestra la consulta SQL que se genera a partir de la consulta de la API:

```
SELECT DISTINCT ACTIVITY.AIID
  FROM   ACTIVITY AI, PROCESS_TEMPLATE PT
  WHERE  AI.PTID = PT.PTID
  AND    PT.NAME = 'sampleProcess'
```

Dado que un administrador es quien invoca la llamada, no se añade una condición de control de acceso a la declaración SQL generada. Tampoco se añade una unión con la vista WORK_ITEM. Esto significa que la consulta recupera todas las actividades para la plantilla de proceso, incluidas las actividades sin elementos de trabajo.

Ejemplo: inclusión de propiedades de consulta en una consulta:

Este ejemplo muestra cómo utilizar el método query para recuperar tareas que pertenecen a un proceso de empresa. El proceso tiene propiedades de consulta definidas para dicho proceso que se pueden incluir en la búsqueda.

Por ejemplo, si desea buscar todas las tareas de usuario en estado preparado que pertenecen a un proceso de empresa. El proceso tiene una propiedad de consulta, **customerID**, con el valor CID_12345 y un espacio de nombres. El fragmento de código siguiente muestra la llamda al método query para la consulta:

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                   PROCESS_INSTANCE.NAME",
          " QUERY_PROPERTY.NAME = 'customerID' AND " +
          " QUERY_PROPERTY.STRING_VALUE = 'CID_12345' AND " +
          " QUERY_PROPERTY.NAMESPACE =
             'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Si ahora desea añadir una segunda propiedad de consulta a la consulta, por ejemplo **Priority**, con un espacio de nombres determinado, la llamada de método query para la consulta tiene el aspecto siguiente:

```
query ( " DISTINCT TASK.TKIID, TASK_TEMPL.NAME, TASK.STATE,
                   PROCESS_INSTANCE.NAME",
          " QUERY_PROPERTY1.NAME = 'customerID' AND " +
          " QUERY_PROPERTY1.STRING_VALUE = 'CID_12345' AND " +
          " QUERY_PROPERTY1.NAMESPACE =
                 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
          " QUERY_PROPERTY2.NAME = 'Priority' AND " +
          " QUERY_PROPERTY2.NAMESPACE =
                 'http://www.ibm.com/xmlns/prod/websphere/mqwf/bpel/' AND " +
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Si añade más de una propiedad de consulta a la consulta, debe numerar las propiedades que añada, tal como se muestra en el fragmento de código. No obstante, la consulta de propiedades personalizadas afecta al rendimiento; el rendimiento disminuye a mayor número de propiedades personalizadas en la consulta.

Ejemplo: inclusión de propiedades personalizadas en una consulta:

Este ejemplo muestra cómo utilizar el método query para recuperar tareas que tienen propiedades personalizadas.

Por ejemplo, si desea buscar todas las tareas de usuario en estado preparado que tienen una propiedad personalizada, **customerID**, con el valor CID_12345. El fragmento de código siguiente muestra la llamda al método query para la consulta:

```
query ( " DISTINCT TASK.TKIID ",
          " TASK_CPROP.NAME = 'customerID' AND " +
          " TASK_CPROP.STRING_VALUE = 'CID_12345' AND " +
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

Si ahora desea recuperar las tareas y sus propiedades personalizadas, la llamada de método query para la consulta tiene el aspecto siguiente:

```
query ( " DISTINCT TASK.TKIID, TASK_CPROP.NAME, TASK_CPROP.STRING_VALUE",
          " TASK.KIND IN
             ( TASK.KIND.KIND_HUMAN, TASK.KIND.KIND_PARTICIPATING ) AND " +
          " TASK.STATE = TASK.STATE.STATE_READY ",
          (String)null, (String)null, (Integer)null, (TimeZone)null );
```

La sentencia SQL que se genera a partir de esta consulta de API se muestra en el siguiente fragmento de código:

```
SELECT DISTINCT TA.TKIID , TACP.NAME , TACP.STRING_VALUE
   FROM  TASK TA LEFT JOIN TASK_CPROP TACP ON (TA.TKIID = TACP.TKIID),
         WORK_ITEM WI
   WHERE WI.OBJECT_ID = TA.TKIID
   AND    TA.KIND IN ( 101, 105 )
   AND    TA.STATE = 2
   AND  (WI.OWNER_ID = 'JuanTorres' OR WI.OWNER_ID IS NULL AND WI.EVERYBODY = 1 )
```

Esta declaración SQL contiene una unión exterior entre la vista TASK y la vista TASK_CPROP. Esto significa que las tareas que satisfacen la cláusula WHERE se recuperan incluso si no tienen ninguna propiedad personalizada.

Gestión de consultas almacenadas

Las consultas almacenadas proporcionan un modo de guardar las consultas que se ejecutan con frecuencia. La consulta almacenada puede ser una consulta que está disponible para todos los usuarios (consulta pública) o una consulta que pertenece a un usuario específico (consulta privada).

Acerca de esta tarea

Una consulta almacenada es una consulta que se almacena en la base de datos y se identifica con un nombre. Una consulta almacenada privada y pública puede tener el mismo nombre; las consultas privadas almacenadas de distintos propietarios también pueden tener el mismo nombre.

Puede tener consultas almacenadas para objetos de proceso de empresa, objetos de tarea, o una combinación de estos dos tipos de objetos.

Conceptos relacionados

“Parámetros de las consultas almacenadas”
Una consulta almacenada es una consulta que se almacena en la base de datos y se identifica con un nombre. Los tuples de calificación se ensamblan dinámicamente cuando se ejecuta la consulta. Para que las consultas almacenadas se puedan volver a utilizar, puede utilizar los parámetros de la definición de consulta que se resuelven durante la ejecución.

Parámetros de las consultas almacenadas:

Una consulta almacenada es una consulta que se almacena en la base de datos y se identifica con un nombre. Los tuples de calificación se ensamblan dinámicamente cuando se ejecuta la consulta. Para que las consultas almacenadas se puedan volver a utilizar, puede utilizar los parámetros de la definición de consulta que se resuelven durante la ejecución.

Por ejemplo, supongamos que ha definido propiedades personalizadas para almacenar nombres de cliente. Puede definir las consultas para devolver las tareas que se asocian a un cliente determinado, ACME Co. Para consultar esta información, la cláusula where de la consulta sería similar al ejemplo siguiente:

```
String whereClause =
  "TASK.STATE = TASK.STATE.STATE_READY
  AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
  AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = 'ACME Co.'";
```

Para hacer que esta consulta sea reutilizable de manera que pueda buscar también el cliente, BCME Ltd, puede utilizar parámetros para los valores de la propiedad personalizada. Si añade parámetros a la consulta de tarea, podría ser similar al ejemplo siguiente:

```
String whereClause =
  "TASK.STATE = TASK.STATE.STATE_READY
  AND WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER
  AND TASK_CPROP.NAME = 'company' AND TASK_CPROP.STRING_VALUE = '@param1'";
```

El parámetro `@param1` se resuelve en tiempo de ejecución de la lista de parámetros que se pasa al método query. Se aplican las siguientes normas para el uso de parámetros en las consultas:

- Los parámetros sólo se pueden utilizar en la cláusula where.
- Los parámetros son de tipo string.
- Los parámetros se sustituyen durante la ejecución utilizando la sustitución de la serie. Si necesita caracteres especiales debe especificarlos en la cláusula where o pasarlos durante la ejecución como parte del parámetro.
- Los nombres de parámetro constan de la serie `@param` concatenada con un número entero. El número inferior es 1, que señala al primer elemento de la lista de parámetros que se pasa a la API de consulta durante la ejecución.
- Se puede utilizar un parámetro varias veces en una cláusula where. Todas las apariciones del parámetro se sustituyen por el mismo valor.

Tareas relacionadas

"Gestión de consultas almacenadas" en la página 391
Las consultas almacenadas proporcionan un modo de guardar las consultas que se ejecutan con frecuencia. La consulta almacenada puede ser una consulta que está disponible para todos los usuarios (consulta pública) o una consulta que pertenece a un usuario específico (consulta privada).

Gestión de consultas almacenadas públicas:

Las consultas almacenadas públicas las crea el administrador del sistema. Estas consultas están disponibles para todos los usuarios.

Acerca de esta tarea

Como administrador del sistema, puede crear, ver y suprimir consultas almacenadas públicas. Si no especifica un ID de usuario en la llamada de la API, se presupone que la consulta almacenada es una consulta almacenada pública.

Procedimiento

1. Cree una consulta almacenada pública.

   Por ejemplo, el fragmento de código siguiente crea una consulta almacenada para las instancias de proceso y lo guarda con el nombre CustomerOrdersStartingWithA.

   ```
   process.createStoredQuery("CustomerOrdersStartingWithA",
               "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
               "PROCESS_INSTANCE.NAME LIKE 'A%'",
               "PROCESS_INSTANCE.NAME",
                (Integer)null, (TimeZone)null);
   ```

El resultado de la consulta almacenado es una lista ordenada de todos los nombres de instancias de proceso que comienzan por la letra A y sus ID (PIID) de instancia de proceso asociados.

2. Ejecutar la consulta definida mediante la consulta almacenada.

```
QueryResultSet result = process.query("CustomerOrdersStartingWithA",
               new Integer(0), null);
```

Esta acción devolverá los objetos que cumplan los criterios. En este caso, todos los pedidos de cliente que empiezan por A.

3. Enumere los nombres de las consultas almacenadas públicas disponibles.

En el fragmento de código siguiente se muestra cómo limitar la lista de consultas devueltas a sólo las consultas públicas.

```
String[] storedQuery = process.getStoredQueryNames(StoredQueryData.KIND_PUBLIC);
```

4. Opcional: Compruebe la consulta definida por una consulta almacenada específica.

Una consulta privada almacenada puede tener el mismo nombre que una consulta pública almacenada. Si estos nombres son iguales, se devolverá la consulta almacenada privada. En el fragmento de código siguiente se muestra cómo devolver sólo la consulta pública con el nombre especificado. Si utiliza la API de Human Task Manager para recuperar información sobre una consulta almacenada, utilice StoredQuery para el objeto devuelto, en lugar de StoredQueryData.

```
StoredQueryData storedQuery = process.getStoredQuery
   (StoredQueryData.KIND_PUBLIC, "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

5. Suprima una consulta almacenada pública.

El siguiente fragmento de código muestra cómo suprimir la consulta almacenada que creó en el paso 1.

```
process.deleteStoredQuery("CustomerOrdersStartingWithA");
```

Gestión de consultas almacenadas privadas de otros usuarios:

Cualquier usuario puede crear consultas privadas. Estas consultas sólo están disponibles para el propietario de la consulta y el administrador del sistema.

Acerca de esta tarea

Como administrador del sistema, puede gestionar las consultas almacenadas privadas que pertenecen a un usuario determinado.

Procedimiento

1. Cree una consulta almacenada privada para el ID de usuario Smith.

Por ejemplo, el fragmento de código siguiente crea una consulta almacenada para instancias de proceso y la guarda con el nombre CustomerOrdersStartingWithA para el ID de usuario Smith.

```
process.createStoredQuery("Smith", "CustomerOrdersStartingWithA",
             "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
             "PROCESS_INSTANCE.NAME LIKE 'A%'",
             "PROCESS_INSTANCE.NAME",
              (Integer)null, (TimeZone)null,
              (List)null, (String)null);
```

El resultado de la consulta almacenado es una lista ordenada de todos los nombres de instancias de proceso que comienzan por la letra A y sus ID (PIID) de instancia de proceso asociados.

2. Ejecutar la consulta definida mediante la consulta almacenada.

```
QueryResultSet result = process.query
                        ("Smith", "CustomerOrdersStartingWithA",
                          (Integer)null, (Integer)null, (List)null);
new Integer(0));
```

Esta acción devolverá los objetos que cumplan los criterios. En este caso, todos los pedidos de cliente que empiezan por A.

3. Obtenga una lista de los nombres de las consultas privadas que pertenecen a un usuario determinado.

Por ejemplo, el fragmento de código siguiente muestra cómo obtener una lista de consultas privadas que pertenecen al usuario Smith.

```
String[] storedQuery = process.getStoredQueryNames("Smith");
```

4. Visualice los detalles de una consulta determinada.

En el fragmento de código siguiente se muestra cómo visualizar los detalles de la consulta CustomerOrdersStartingWithA cuyo propietario es el usuario Smith.

```
StoredQueryData storedQuery = process.getStoredQuery
   ("Smith", "CustomerOrdersStartingWithA");
String selectClause = storedQuery.getSelectClause();
String whereClause = storedQuery.getWhereClause();
String orderByClause = storedQuery.getOrderByClause();
Integer threshold = storedQuery.getThreshold();
String owner = storedQuery.getOwner();
```

Si utiliza la API de Human Task Manager para recuperar información sobre una consulta almacenada, utilice StoredQuery para el objeto devuelto, en lugar de StoredQueryData.

5. Suprimir una consulta almacenada privada.

En el fragmento de código siguiente se muestra cómo suprimir una consulta privada cuyo propietario es el usuario Smith.

```
process.deleteStoredQuery("Smith", "CustomerOrdersStartingWithA");
```

Cómo trabajar con las consultas almacenadas privadas:

Si usted no es el administrador del sistema, puede crear, ejecutar y suprimir sus propias consultas almacenadas privadas. También puede utilizar las consultas almacenadas públicas que el administrador del sistema ha creado.

Procedimiento

1. Cree una consulta almacenada privada.

Por ejemplo, el fragmento de código siguiente crea una consulta almacenada para instancias de proceso y la guarda con un nombre específico. Si no se especifica un ID de usuario, se presupone que la consulta almacenada es una consulta almacenada privada para el usuario que ha iniciado la sesión.

```
process.createStoredQuery("CustomerOrdersStartingWithA",
             "DISTINCT PROCESS_INSTANCE.PIID, PROCESS_INSTANCE.NAME",
             "PROCESS_INSTANCE.NAME LIKE 'A%'",
             "PROCESS_INSTANCE.NAME",
              (Integer)null, (TimeZone)null);
```

Esta consulta devuelve una lista ordenada de todos los nombres de instancias de proceso que comienzan por la letra A y sus ID (PIID) de instancia de proceso asociados.

2. Ejecutar la consulta definida mediante la consulta almacenada.

```
QueryResultSet result = process.query("CustomerOrdersStartingWithA",
                new Integer(0));
```

Esta acción devolverá los objetos que cumplan los criterios. En este caso, todos los pedidos de cliente que empiezan por A.

3. Obtenga una lista de los nombres de las consultas almacenadas a las que el usuario que ha iniciado la sesión tiene acceso.

   En el fragmento de código siguiente se muestra cómo obtener las consultas almacenadas públicas y privadas a las que tiene acceso el usuario.

   ```
   String[] storedQuery = process.getStoredQueryNames();
   ```

4. Visualice los detalles de una consulta determinada.

   En el fragmento de código siguiente se muestra cómo visualizar los detalles de la consulta CustomerOrdersStartingWithA cuyo propietario es el usuario Smith.

   ```
   StoredQueryData storedQuery = process.getStoredQuery
      ("CustomerOrdersStartingWithA");
   String selectClause = storedQuery.getSelectClause();
   String whereClause = storedQuery.getWhereClause();
   String orderByClause = storedQuery.getOrderByClause();
   Integer threshold = storedQuery.getThreshold();
   String owner = storedQuery.getOwner();
   ```

   Si utiliza la API de Human Task Manager para recuperar información sobre una consulta almacenada, utilice StoredQuery para el objeto devuelto, en lugar de StoredQueryData.

5. Suprimir una consulta almacenada privada.

   El fragmento de código siguiente muestra cómo se suprime una consulta almacenada privada.

   ```
   process.deleteStoredQuery("CustomerOrdersStartingWithA");
   ```

Desarrollo de aplicaciones de cliente EJB para los procesos empresariales y tareas de usuario

Las API de EJB proporcionan un conjunto de métodos genéricos para desarrollar aplicaciones cliente EJB para trabajar con los procesos de empresa y tareas de usuario instalados en WebSphere Process Server.

Acerca de esta tarea

Con estas API EJB (Enterprise JavaBeans), puede crear aplicaciones de cliente para realizar lo siguiente:

- Gestionar el ciclo de vida de los procesos y tareas desde iniciarlas a suprimirlas cuando finalizan
- Reparar actividades y procesos
- Gestionar y distribuir la carga de trabajo en todos los miembros de un grupo de trabajo

Las API de EJB se proporcionan como dos enterprise bean de sesión sin estado.

- La interfaz BusinessFlowManagerService proporciona los métodos para las aplicaciones de proceso de empresa
- La interfaz HumanTaskManagerService proporciona los métodos para las aplicaciones basadas en tareas.

Para obtener más información sobre las API de EJB, consulte el Javadoc de los paquetes com.ibm.bpe.api y com.ibm.task.api.

En los pasos siguientes se proporciona una visión general de las acciones que es necesario realizar para desarrollar una aplicación de cliente EJB.

Procedimiento

1. Decida sobre la funcionalidad que la aplicación va a proporcionar.
2. Determine los beans de sesión que va a utilizar.

   Según los escenarios que desee implementar con la aplicación, puede utilizar uno de los beans de sesión o ambos.
3. Determine las autorizaciones que necesitan los usuarios de la aplicación.

   A los usuarios de la aplicación se les debe asignar los roles de autorización adecuados para llamar a los métodos que se incluyan en la aplicación y ver los objetos y los atributos de estos objetos que estos métodos devuelvan. Cuando se crea una instancia del bean de sesión adecuado, WebSphere Application Server asocia un contexto a la instancia. El contexto contiene información acerca del ID del principal del proceso que efectúa la llamada, la lista de miembros del grupo y los roles. Esta información se utiliza para comprobar la autorización del llamante para cada llamada.

   El Javadoc contiene información de autorización de todos los métodos.
4. Decida cómo representar la aplicación.

   Se puede llamar a las API de EJB de forma local o remota.
5. Desarrolle la aplicación.
   a. Acceda a la API de EJB.
   b. Utilice la API de EJB para interactuar con procesos o tareas.
      - Consulte los datos.
      - Trabaje con los datos.

Conceptos relacionados

Modalidad alternativa de autorización de administración

Referencia relacionada

"Interfaz BusinessFlowManagerService" en la página 425
La interfaz BusinessFlowManagerService expone funciones de proceso empresarial que una aplicación cliente puede llamar.

"Interfaz HumanTaskManagerService" en la página 444
La interfaz HumanTaskManagerService expone funciones relativas a tareas que un cliente local o remoto puede llamar.

Acceso a las API de EJB

Las API de EJB (Enterprise JavaBeans) se proporcionan como dos enterprise beans de sesión sin estado. Las aplicaciones de procesos de empresa y aplicaciones de tareas acceden al enterprise bean de sesión adecuado mediante la interfaz inicial del bean.

Acerca de esta tarea

La interfaz BusinessFlowManagerService proporciona los métodos para las aplicaciones de proceso de empresa y la interfaz HumanTaskManagerService proporciona los métodos para aplicaciones basadas en tareas. La aplicación puede ser cualquier aplicación Java, incluida otra aplicación EJB (Enterprise JavaBeans).

Acceso a la interfaz remota del bean de sesión

Una aplicación cliente EJB para procesos empresariales o tareas de usuario accede a la interfaz remota del bean de sesión a través de la interfaz inicial remota del bean.

Acerca de esta tarea

El bean de sesión puede ser el bean de sesión de BusinessFlowManager para las aplicaciones de proceso o el bean de sesión HumanTaskManager para las aplicaciones de tareas.

Procedimiento

1. Añada una referencia a la interfaz remota del bean de sesión para el descriptor de despliegue de la aplicación. Añada la referencia a uno de los archivos siguientes:
   - El archivo `application-client.xml`, para una aplicación cliente de Java Platform, Enterprise Edition (Java EE)
   - El archivo `web.xml` para una aplicación Web
   - El archivo `ejb-jar.xml` para una aplicación EJB (Enterprise JavaBeans)

   La referencia a la interfaz inicial remota para aplicaciones de proceso se muestra en el ejemplo siguiente:

   ```
   <ejb-ref>
    <ejb-ref-name>ejb/BusinessFlowManagerHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
    <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
   </ejb-ref>
   ```

   La referencia a la interfaz inicial remota para aplicaciones de tarea se muestra en el ejemplo siguiente:

   ```
   <ejb-ref>
    <ejb-ref-name>ejb/HumanTaskManagerHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <home>com.ibm.task.api.HumanTaskManagerHome</home>
    <remote>com.ibm.task.api.HumanTaskManager</remote>
   </ejb-ref>
   ```

   Si utiliza WebSphere Integration Developer para añadir la referencia EJB al descriptor de despliegue, el enlace para la referencia EJB se crea automáticamente cuando se despliega la aplicación. Para obtener más información sobre cómo añadir referencias EJB, consulte la documentación de WebSphere Integration Developer.
2. Decida cómo va a proporcionar definiciones de objetos de empresa.

   Para trabajar con objetos de empresa en una aplicación de cliente remoto, debe tener acceso a los esquemas correspondientes para los objetos de empresa (archivos XSD o WSDL) utilizados para interactuar con un proceso o una tarea. El acceso a dichos archivos se puede proporcionan de una las formas siguientes:
   - Si la aplicación cliente no se ejecuta en un entorno gestionado Java EE, empaquete los archivos con el archivo EAR de la aplicación cliente.
   - Si la aplicación cliente es una aplicación Web o un cliente EJB en un entorno Java EE gestionado, empaquete los archivos con el archivo EAR de la aplicación cliente o aproveche la carga de artefactos remota.
     a. Utilice la API EJB de Business Process Choreographer createMessage y los métodos ClientObjectWrapper.getObject para cargar las definiciones de objeto empresarial remoto desde la aplicación correspondiente en el servidor de forma transparente.

b. Utilice la API de Service Data Object Programming para crear o leer un objeto de empresa como parte de un objeto de empresa del que ya se ha creado una instancia. Realice esta acción utilizando los métodos commonj.sdo.DataObject.createDataObject o getDataObject en la interfaz DataObject.

c. Cuando desee crear un objeto empresarial como valor para una propiedad del objeto empresarial que se ha escrito utilizando el esquema XML any o anyType, utilice los servicios de objeto empresarial para crear o leer el objeto empresarial. Para hacerlo, debe establecer el contexto del cargador de artefactos remotos para indicar la aplicación desde la cual se cargarán los esquemas. A continuación, puede utilizar los servicios de objeto empresarial apropiados.

   Por ejemplo, cree un objeto empresarial, donde "ApplicationName" es el nombre de la aplicación que contiene las definiciones de objeto empresarial.

```
BOFactory bofactory = (BOFactory) new
   ServiceManager().locateService("com/ibm/websphere/bo/BOFactory");

com.ibm.wsspi.al.ALContext.setContext
   ("RALTemplateName", "ApplicationName");
try {
     DataObject dataObject = bofactory.create("uriName", "typeName" );
   } finally {
     com.ibm.wsspi.al.ALContext.unset();
   }
```

   Por ejemplo, lea la entrada XML, donde "ApplicationName" es el nombre de la aplicación que contiene las definiciones del objeto de empresa.

```
BOXMLSerializer serializerService =
  (BOXMLSerializer) new ServiceManager().locateService
    ("com/ibm/websphere/bo/BOXMLSerializer");
ByteArrayInputStream input = new ByteArrayInputStream(<?xml?>..);

com.ibm.wsspi.al.ALContext.setContext
          ("RALTemplateName", "ApplicationName");
try {
     BOXMLDocument document = serializerService.readXMLDocument(input);
     DataObject dataObject = document.getDataObject();
   } finally {
     com.ibm.wsspi.al.ALContext.unset();
   }
```

3. Busque la interfaz inicial remota del bean de sesión en la interfaz Java Naming and Directory Interface (JNDI).

   El ejemplo siguiente muestra este paso para una aplicación de proceso:

```
// Obtener el contexto inicial por omisión de JNDI
InitialContext initialContext = new InitialContext();

// Buscar la interfaz inicial remota del bean BusinessFlowManager
Object result =
       initialContext.lookup("java:comp/env/ejb/BusinessFlowManagerHome");

// Convertir el resultado de búsqueda al tipo adecuado
   BusinessFlowManagerHome processHome =
           (BusinessFlowManagerHome)javax.rmi.PortableRemoteObject.narrow
           (result,BusinessFlowManagerHome.class);
```

   La interfaz inicial remota del bean de sesión contiene un método create para objetos EJB. El método devuelve la interfaz remote del bean de sesión.

4. Acceda a la interfaz remota del bean de sesión.

   El ejemplo siguiente muestra este paso para una aplicación de proceso:

```
BusinessFlowManager process = processHome.create();
```

El acceso al bean de sesión no garantiza que el proceso que efectúa la llamada pueda realizar todas las acciones proporcionadas por el bean, además, dicho proceso debe tener autorización para estas acciones. Cuando se crea una instancia del bean de sesión, se asocia un contexto a la instancia del bean de sesión. El contexto contiene el ID principal del emisor de la llamada, la lista de miembros de grupo e indica si el emisor de la llamada tiene uno de los roles Java EE de Business Process Choreographer. El contexto se utiliza para comprobar la autorización para cada llamada del que la realiza, incluso cuando no se ha establecido la seguridad administrativa. Si la seguridad administrativa no se ha establecido, el ID principal del que llama tiene el valor `UNAUTHENTICATED`.

5. Llame a las funciones de empresa expuestas por la interfaz de servicio.

   El ejemplo siguiente muestra este paso para una aplicación de proceso:

   ```
   process.initiate("MyProcessModel",input);
   ```

   Las llamadas procedentes de las aplicaciones se ejecutarán como transacciones. Se establece y finaliza una transacción de alguna de las formas siguientes:
   - Automáticamente, por parte de WebSphere Application Server (el descriptor de despliegue especifica TX_REQUIRED).
   - Explícitamente, por parte de la aplicación. Puede empaquetar las llamadas de aplicación en una transacción:

     ```
     // Obtener interfaz de transacción de usuario
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("java:comp/UserTransaction");

     // Empezar una transacción
     transaction.begin();

     // Llamadas de aplicaciones ...

     // Cuando se devuelva de forma satisfactoria, confirmar la transacción
     transaction.commit();
     ```

   Consejo: Para impedir conflictos de bloqueo en la base de datos, procure no ejecutar las sentencias similares a las siguientes en paralelo:

   ```
   // Obtener interfaz de transacción de usuario
   UserTransaction transaction=
          (UserTransaction)initialContext.lookup("java:comp/UserTransaction");

   transaction.begin();

   //leer la instancia de actividad
   process.getActivityInstance(aiid);
   //reclamar la instancia de actividad
   process.claim(aiid);

   transaction.commit();
   ```

   El método getActivityInstance y otras operaciones de lectura establecen un bloqueo de lectura. En este ejemplo, se actualiza un bloqueo de lectura en la instancia de actividad a un bloqueo de actualización en la instancia de actividad. Esto puede producir un punto muerto en la base de datos cuando estas transacciones se ejecutan en paralelo.

Ejemplo

Éste es un ejemplo de cómo los pasos 3 a 5 pueden buscar una aplicación de tareas.

```
// Obtener el contexto inicial por omisión de JNDI
InitialContext initialContext = new InitialContext();

// Buscar la interfaz inicial remota del bean HumanTaskManager
Object result =
         initialContext.lookup("java:comp/env/ejb/HumanTaskManagerHome");

// Convertir el resultado de búsqueda al tipo adecuado
HumanTaskManagerHome taskHome =
           (HumanTaskManagerHome)javax.rmi.PortableRemoteObject.narrow
           (result,HumanTaskManagerHome.class);

...
//Acceder a la interfaz remota del bean de sesión.
HumanTaskManager task = taskHome.create();

...
//Llamar a las funciones de empresa expuestas por la interfaz de servicio
task.callTask(tkiid,input);
```

Acceso a la interfaz local del bean de sesión

Una aplicación cliente EJB para procesos empresariales o tareas de usuario accede a la interfaz local del bean de sesión a través de la interfaz inicial local del bean.

Acerca de esta tarea

El bean de sesión puede ser el bean de sesión de BusinessFlowManager para las aplicaciones de proceso o el bean de sesión HumanTaskManager para las aplicaciones de tareas de usuario.

Procedimiento

1. Añada una referencia a la interfaz local del bean de sesión para el descriptor de despliegue de la aplicación. Añada la referencia a uno de los archivos siguientes:
   - El archivo `application-client.xml`, para una aplicación cliente de Java Platform, Enterprise Edition (Java EE)
   - El archivo `web.xml` para una aplicación Web
   - El archivo `ejb-jar.xml` para una aplicación EJB (Enterprise JavaBeans)

   La referencia a la interfaz inicial local para aplicaciones de proceso se muestra en el ejemplo siguiente:

   ```
   <ejb-local-ref>
    <ejb-ref-name>ejb/LocalBusinessFlowManagerHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
    <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
   </ejb-local-ref>
   ```

   La referencia a la interfaz inicial local para aplicaciones de tarea se muestra en el ejemplo siguiente:

   ```
   <ejb-local-ref>
    <ejb-ref-name>ejb/LocalHumanTaskManagerHome</ejb-ref-name>
    <ejb-ref-type>Session</ejb-ref-type>
    <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
    <local>com.ibm.task.api.LocalHumanTaskManager</local>
   </ejb-local-ref>
   ```

   Si utiliza WebSphere Integration Developer para añadir la referencia EJB al descriptor de despliegue, el enlace para la referencia EJB se crea automáticamente cuando se despliega la aplicación. Para obtener más información sobre cómo añadir referencias EJB, consulte la documentación de WebSphere Integration Developer.

2. Busque la interfaz inicial local del bean de sesión en la interfaz Java Naming and Directory Interface (JNDI).

   El ejemplo siguiente muestra este paso para una aplicación de proceso:

   ```
   // Obtener el contexto inicial por omisión de JNDI
   InitialContext initialContext = new InitialContext();

   // Buscar la interfaz inicial local del bean BusinessFlowManager

    LocalBusinessFlowManagerHome processHome =
         (LocalBusinessFlowManagerHome)initialContext.lookup
         ("java:comp/env/ejb/LocalBusinessFlowManagerHome");
   ```

   La interfaz inicial local del bean de sesión contiene un método create para objetos EJB. El método devuelve la interfaz local del bean de sesión.

3. Acceda a la interfaz local del bean de sesión.

   El ejemplo siguiente muestra este paso para una aplicación de proceso:

   ```
   LocalBusinessFlowManager process = processHome.create();
   ```

   El acceso al bean de sesión no garantiza que el proceso que efectúa la llamada pueda realizar todas las acciones proporcionadas por el bean, además, dicho proceso debe tener autorización para estas acciones. Cuando se crea una instancia del bean de sesión, se asocia un contexto a la instancia del bean de sesión. El contexto contiene el ID principal del emisor de la llamada, la lista de miembros de grupo e indica si el emisor de la llamada tiene uno de los roles Java EE de Business Process Choreographer. El contexto se utiliza para comprobar la autorización para cada llamada del que la realiza, incluso cuando no se ha establecido la seguridad administrativa. Si la seguridad administrativa no se ha establecido, el ID principal del que llama tiene el valor `UNAUTHENTICATED`.

4. Llame a las funciones de empresa expuestas por la interfaz de servicio.

   El ejemplo siguiente muestra este paso para una aplicación de proceso:

   ```
   process.initiate("MyProcessModel",input);
   ```

   Las llamadas procedentes de las aplicaciones se ejecutarán como transacciones. Se establece y finaliza una transacción de alguna de las formas siguientes:
   - Automáticamente, por parte de WebSphere Application Server (el descriptor de despliegue especifica TX_REQUIRED).
   - Explícitamente, por parte de la aplicación. Puede empaquetar las llamadas de aplicación en una transacción:

     ```
     // Obtener interfaz de transacción de usuario
     UserTransaction transaction=
            (UserTransaction)initialContext.lookup("java:comp/UserTransaction");

     // Iniciar una transacción
     transaction.begin();

     // Llamadas de aplicaciones ...

     // Cuando se devuelva de forma satisfactoria, confirmar la transacción
     transaction.commit();
     ```

   Consejo: Para impedir los puntos muertos en la base de datos, procure no ejecutar las sentencias similares a las siguientes en paralelo:

   ```
   // Obtener interfaz de transacción de usuario
   UserTransaction transaction=
          (UserTransaction)initialContext.lookup("java:comp/UserTransaction");

   transaction.begin();
   ```

```
//leer la instancia de actividad
process.getActivityInstance(aiid);
//reclamar la instancia de actividad
process.claim(aiid);

transaction.commit();
```

El método getActivityInstance y otras operaciones de lectura establecen un bloqueo de lectura. En este ejemplo, se actualiza un bloqueo de lectura en la instancia de actividad a un bloqueo de actualización en la instancia de actividad. Esto puede producir un punto muerto en la base de datos cuando estas transacciones se ejecutan en paralelo

Ejemplo

Éste es un ejemplo de cómo los pasos 2 a 4 pueden buscar una aplicación de tareas.

```
// Obtener el contexto inicial por omisión de JNDI
InitialContext initialContext = new InitialContext();

//Buscar la interfaz inicial local del bean HumanTaskManager
LocalHumanTaskManagerHome taskHome =
        (LocalHumanTaskManagerHome)initialContext.lookup
        ("java:comp/env/ejb/LocalHumanTaskManagerHome");

...
//Acceder a la interfaz local del bean de sesión
LocalHumanTaskManager task = taskHome.create();

...
//Llamar a las funciones de empresa expuestas por la interfaz de servicio
task.callTask(tkiid,input);
```

Desarrollo de aplicaciones para procesos de empresa

Un proceso empresariales un conjunto de actividades relacionadas con la empresa que se invocan en una secuencia específica para alcanzar un objetivo de empresa. Se proporcionan ejemplos que muestran cómo se pueden desarrollar aplicaciones para acciones típicas en procesos.

Acerca de esta tarea

Un proceso empresarialpuede ser un microflujo o un proceso de larga ejecución:

- Los microflujos son procesos de empresa de corta ejecución que se ejecutan de forma síncrona. Después de muy poco tiempo, el resultado se devuelve al llamante.
- Los procesos interrumpibles de larga ejecución se ejecutan como una secuencia de actividades encadenadas. El uso de determinadas construcciones en un proceso crea interrupciones en el flujo del proceso, por ejemplo, cuando se invoca una tarea de usuario, se invoca un servicio utilizando un enlace síncrono o se utilizan actividades dirigidas por temporizador.

  Generalmente se navega de forma asíncrona por las ramas paralelas del proceso, esto es, las actividades de las ramas paralelas se ejecutan de forma simultánea. Según el tipo y el valor de transacción de la actividad, puede ejecutarse una actividad en su propia transacción.

Roles necesarios para las acciones en instancias de proceso

Acceder a la interfaz BusinessFlowManager no garantiza que el emisor de la llamada pueda realizar todas las acciones de un proceso. El emisor de la llamada debe iniciar la sesión en la aplicación cliente con un rol que tenga autorización para realizar la acción.

En la tabla siguiente se muestran las acciones en una instancia de proceso que un rol específico puede realizar.

Acción	**Rol principal del emisor de la llamada**		
	Lector	**Iniciador**	**Administrador**
createMessage	x	x	x
createWorkItem			x
delete			x
deleteWorkItem			x
forceTerminate			x
getActiveEventHandlers	x		x
getActivityInstance	x		x
getAllActivities	x		x
getAllWorkItems	x		x
getClientUISettings	x	x	x
getCustomProperties	x	x	x
getCustomProperty	x	x	x
getCustomPropertyNames	x	x	x
getFaultMessage	x	x	x
getInputClientUISettings	x	x	x
getInputMessage	x	x	x
getOutputClientUISettings	x	x	x
getOutputMessage	x	x	x
getProcessInstance	x	x	x
getVariable	x	x	x
getWaitingActivities	x	x	x
getWorkItems	x		x
restart			x
resume			x
setCustomProperty		x	x
setVariable			x
suspender			x
transferWorkItem			x

Nota: Si la administración del proceso está limitada a los administradores del sistema, la administración basada en instancias está inhabilitada. Esto significa que las acciones administrativas en procesos, ámbitos y actividades están limitadas a los usuarios del rol BPESystemAdministrator. Además, sólo los usuarios de los roles BPESystemAdministrator o BPESystemMonitor pueden leer, visualizar y

supervisar una instancia o partes de un proceso. Si desea más información sobre esta modalidad de administración, consulte doc/bpc/cnot_instance_based_admin.dita.

Roles necesarios para acciones en actividades de procesos empresariales

Acceder a la interfaz BusinessFlowManager no garantiza que el emisor de la llamada pueda realizar todas las acciones de una actividad. El emisor de la llamada debe iniciar la sesión en la aplicación cliente con un rol que tenga autorización para realizar la acción.

En la tabla siguiente se muestran las acciones en una instancia de actividad que un rol específico puede realizar.

Acción	**Rol principal del emisor de la llamada**				
	Lector	**Editor**	**Propietario potencial**	**Propietario**	**Administrador**
cancelClaim				x	x
claim			x		x
complete				x	x
createMessage	x	x	x	x	x
createWorkItem					x
deleteWorkItem					x
forceComplete					x
forceRetry					x
getActivityInstance	x	x	x	x	x
getAllWorkItems	x	x	x	x	x
getClientUISettings	x	x	x	x	x
getCustomProperties	x	x	x	x	x
getCustomProperty	x	x	x	x	x
getCustomPropertyNames	x	x	x	x	x
getFaultMessage	x	x	x	x	x
getFaultNames	x	x	x	x	x
getInputMessage	x	x	x	x	x
getOutputMessage	x	x	x	x	x
getVariable	x	x	x	x	x
getVariableNames	x	x	x	x	x
getInputVariableNames	x	x	x	x	x
getOutputVariableNames	x	x	x	x	x
getWorkItems	x	x	x	x	x
setCustomProperty		x		x	x
setFaultMessage		x		x	x
setOutputMessage		x		x	x
setVariable					x

Acción	Rol principal del emisor de la llamada				
	Lector	Editor	Propietario potencial	Propietario	Administrador
transferWorkItem				x Sólo para propietarios potenciales o administra-dores	x

Nota: Si la administración del proceso está limitada a los administradores del sistema, la administración basada en instancias está inhabilitada. Esto significa que las acciones administrativas en procesos, ámbitos y actividades están limitadas a los usuarios del rol BPESystemAdministrator. Además, sólo los usuarios de los roles BPESystemAdministrator o BPESystemMonitor pueden leer, visualizar y supervisar una instancia o partes de un proceso. Si desea más información sobre esta modalidad de administración, consulte doc/bpc/cnot_instance_based_admin.dita.

Gestión del ciclo de vida de un proceso de empresa

Una instancia de proceso pasa a existir cuando se invoca un método de la API de Business Process Choreographer que puede iniciar un proceso. La navegación de la instancia de proceso continúa hasta que todas sus actividades están en un estado final. Se pueden llevar a cabo varias acciones en la instancia de proceso para gestionar su ciclo de vida.

Acerca de esta tarea

Se proporcionan ejemplos que muestran cómo puede desarrollar aplicaciones para las siguientes acciones típicas de ciclo de vida en los procesos.

Inicio de procesos de empresa:

La manera en que se inicia un proceso de empresa depende de si el proceso es un microflujo o un proceso de larga ejecución. El servicio que inicia el proceso también es importante para la manera en que se inicia un proceso; el proceso puede tener un servicio inicial exclusivo o varios servicios iniciales.

Acerca de esta tarea

Se proporcionan ejemplos que muestran cómo puede desarrollar aplicaciones de casos típicos para iniciar microflujos y procesos de larga ejecución.

Ejecución de un microflujo que contiene un servicio de arranque exclusivo:

Se puede iniciar un microflujo mediante una actividad de recepción o de obtención. El servicio de arranque es exclusivo si el microflujo se inicia con una actividad de recepción o cuando la actividad de obtención únicamente tiene una definición onMessage.

Acerca de esta tarea

Si el microflujo implementa una operación de petición y respuesta, es decir, el proceso contiene una respuesta, puede utilizar el método call para ejecutar el proceso pasando el nombre de plantilla de proceso como parámetro en la llamada.

Si el microflujo es una operación unidireccional, utilice el método sendMessage para ejecutar el proceso. Este método no está cubierto en este ejemplo.

Procedimiento

1. Opcional: Liste las plantillas de proceso para encontrar el nombre del proceso que desea ejecutar.

   Este paso es opcional si ya sabe el nombre del proceso.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.EXECUTION_MODE =
          PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_MICROFLOW",
    "PROCESS_TEMPLATE.NAME",
     new Integer(50),
     (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas clasificadas que se pueden iniciar mediante el método call.

2. Iniciar el proceso con un mensaje de entrada del tipo adecuado.

   Cuando cree el mensaje, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje.

   ```
   ProcessTemplateData template = processTemplates[0];
   //crear un mensaje sólo para la actividad de recepción inicial
   ClientObjectWrapper input = process.createMessage
                              (template.getID(),
                               template.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //establecer las series del mensaje, por ejemplo, un nombre de cliente
     myMessage.setString("CustomerName", "Smith");
   }

   //ejecutar el proceso
   ClientObjectWrapper output = process.call(template.getName(), input);
   DataObject myOutput = null;
   if ( output.getObject() != null && output.getObject() instanceof DataObject )
   {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
   }
   ```

   Esta acción crea una instancia de la plantilla de proceso, CustomerTemplate, y pasa algunos datos al cliente. La operación sólo devuelve resultados cuando el proceso se ha completado. Se devuelve el resultado del proceso, OrderNo, al proceso que efectúa la llamada.

Ejecución de un microflujo que contiene un servicio de arranque no exclusivo:

Se puede iniciar un microflujo mediante una actividad de recepción o de obtención. El servicio de arranque no es exclusivo si el microflujo se inicia con una actividad de obtención que tenga varias definiciones onMessage.

Acerca de esta tarea

Si el microflujo implementa una operación de petición y respuesta, es decir, el proceso contiene una respuesta, puede utilizar el método call para ejecutar el proceso pasando el ID del servicio inicial en la llamada.

Si el microflujo es una operación unidireccional, utilice el método sendMessage para ejecutar el proceso. Este método no está cubierto en este ejemplo.

Procedimiento

1. Opcional: Liste las plantillas de proceso para encontrar el nombre del proceso que desea ejecutar.

   Este paso es opcional si ya sabe el nombre del proceso.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ("PROCESS_TEMPLATE.EXECUTION_MODE =
         PROCESS_TEMPLATE.EXECUTION_MODE.EXCECUTION_MODE_MICROFLOW",
    "PROCESS_TEMPLATE.NAME",
     new Integer(50),
     (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas clasificadas que se pueden iniciar como microflujos.

2. Determinar el servicio de arranque al que se va a llamar.

   Este ejemplo utiliza la primera plantilla que se encuentra.

   ```
   ProcessTemplateData template = processTemplates[0];
   ActivityServiceTemplateData[] startActivities =
           process.getStartActivities(template.getID());
   ```

3. Iniciar el proceso con un mensaje de entrada del tipo adecuado.

   Cuando cree el mensaje, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje.

   ```
   ActivityServiceTemplateData activity = startActivities[0];
   //crear un mensaje para el servicio que se va a llamar
   ClientObjectWrapper input =
         process.createMessage(activity.getServiceTemplateID(),
                               activity.getActivityTemplateID(),
                               activity.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //establecer las series del mensaje, por ejemplo, un nombre de cliente
     myMessage.setString("CustomerName", "Smith");
   }
   //ejecutar el proceso
   ClientObjectWrapper output = process.call(activity.getServiceTemplateID(),
                                             activity.getActivityTemplateID(),
                                             input);
   //comprobar la salida del proceso, por ejemplo, un número de pedido
   DataObject myOutput = null;
   if ( output.getObject() != null && output.getObject() instanceof DataObject )
   {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
   }
   ```

   Esta acción crea una instancia de la plantilla de proceso, CustomerTemplate, y pasa algunos datos al cliente. La operación sólo devuelve resultados cuando el proceso se ha completado. Se devuelve el resultado del proceso, OrderNo, al proceso que efectúa la llamada.

Inicio de un proceso de larga ejecución que contiene un servicio de arranque exclusivo:

Si el servicio de arranque es exclusivo, puede utilizar el método initiate y pasar el nombre de la plantilla de proceso como un parámetro. Este es el caso, cuando el proceso de larga ejecución se inicia con una sola actividad de recepción o de obtención y cuando la actividad de obtención individual solamente tiene una definición onMessage.

Procedimiento

1. Opcional: Listar las plantillas de proceso para encontrar el nombre del proceso que desea iniciar.

   Este paso es opcional si ya sabe el nombre del proceso.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
     ("PROCESS_TEMPLATE.EXECUTION_MODE =
          PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_LONG_RUNNING",
      "PROCESS_TEMPLATE.NAME",
       new Integer(50),
       (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas clasificadas que se pueden iniciar mediante el método initiate.

2. Iniciar el proceso con un mensaje de entrada del tipo adecuado.

   Cuando cree el mensaje, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje. Si especifica un nombre de instancia de proceso, no debe comenzar con el carácter de subrayado. Si no se ha especificado un nombre de instancia de proceso, se utiliza como nombre el ID de instancia de proceso (PIID) con formato de serie.

   ```
   ProcessTemplateData template = processTemplates[0];
   //crear un mensaje sólo para la actividad de recepción inicial
   ClientObjectWrapper input = process.createMessage
                               (template.getID(),
                               template.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //establecer las series del mensaje, por ejemplo, un nombre de cliente
     myMessage.setString("CustomerName", "Smith");
   }
   //iniciar el proceso
   PIID piid = process.initiate(template.getName(), "CustomerOrder", input);
   ```

   Esta acción crea una instancia, CustomerOrder, y pasa algunos datos al cliente. Cuando se inicia el proceso, la operación devuelve el ID de objeto de la nueva instancia de proceso al proceso que efectúa la llamada.

   Se establece el iniciador de la instancia del proceso en el proceso que efectúa la llamada de la petición. Esta persona recibe un elemento de trabajo para la instancia de proceso. Se determinan los administradores de procesos, los lectores y los editores de la instancia de proceso y reciben elementos de trabajo para dicha instancia de proceso. Se determinan las instancias de las actividades que siguen. Se inician automáticamente o, si son actividades de tareas de usuario, de recepción o de obtención, se crean elementos de trabajo para los posibles propietarios.

Inicio de un proceso de larga ejecución que contiene un servicio de arranque que no es exclusivo:

Un proceso de larga ejecución se puede iniciar mediante varias actividades de recepción o de obtención iniciales. Puede utilizar el método initiate para iniciar el proceso. Si el servicio de inicio no es exclusivo, por ejemplo, si el proceso se inicia con varias actividades de recepción o de obtención, o una actividad de obtencion que tiene varias definiciones onMessage, debe identificar el servicio al que se ha de llamar.

Procedimiento

1. Opcional: Listar las plantillas de proceso para encontrar el nombre del proceso que desea iniciar.

   Este paso es opcional si ya sabe el nombre del proceso.

   ```
   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
     ("PROCESS_TEMPLATE.EXECUTION_MODE =
          PROCESS_TEMPLATE.EXECUTION_MODE.EXECUTION_MODE_LONG_RUNNING",
      "PROCESS_TEMPLATE.NAME",
       new Integer(50),
       (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas clasificadas que se pueden iniciar como procesos de larga ejecución.

2. Determinar el servicio de arranque al que se va a llamar.

   ```
   ProcessTemplateData template = processTemplates[0];
   ActivityServiceTemplateData[] startActivities =
           process.getStartActivities(template.getID());
   ```

3. Iniciar el proceso con un mensaje de entrada del tipo adecuado.

   Cuando cree el mensaje, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje. Si especifica un nombre de instancia de proceso, no debe comenzar con el carácter de subrayado. Si no se ha especificado un nombre de instancia de proceso, se utiliza como nombre el ID de instancia de proceso (PIID) con formato de serie.

   ```
   ActivityServiceTemplateData activity = startActivities[0];
   //crear un mensaje para el servicio que se va a llamar
   ClientObjectWrapper input = process.createMessage
                               (activity.getServiceTemplateID(),
                                activity.getActivityTemplateID(),
                                activity.getInputMessageTypeName());
   DataObject myMessage = null;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //establecer las series del mensaje, por ejemplo, un nombre de cliente
     myMessage.setString("CustomerName", "Smith");
   }
   //iniciar el proceso
   PIID piid = process.sendMessage(activity.getServiceTemplateID(),
                                activity.getActivityTemplateID(),
                                input);
   ```

   Esta acción crea una instancia y pasa algunos datos al cliente. Cuando se inicia el proceso, la operación devuelve el ID de objeto de la nueva instancia de proceso al proceso que efectúa la llamada.

   Se establece el iniciador de la instancia del proceso en el proceso que efectúa la llamada de la petición y recibe un elemento de trabajo para la instancia del proceso. Se determinan los administradores de procesos, los lectores y los editores de la instancia de proceso y reciben elementos de trabajo para dicha instancia de proceso. Se determinan las instancias de las actividades que

siguen. Se inician automáticamente o, si son actividades de tareas de usuario, de recepción o de obtención, se crean elementos de trabajo para los posibles propietarios.

Suspensión y reanudación de un proceso empresarial:

Puede suspender instancias de proceso de nivel superior de larga ejecución mientras se están ejecutando y reanudarlas de nuevo para completarlas.

Antes de empezar

Acerca de esta tarea

Por ejemplo, quizá desee suspender una instancia de proceso, de manera que pueda configurar el acceso a un sistema de programa de fondo que se utiliza posteriormente en el proceso. Cuando se cumplan los prerrequisitos del proceso, puede reanudar la instancia de proceso. También es posible que desee suspender un proceso para solucionar un problema que está haciendo que la instancia de proceso falle y luego volver a reanudarlo cuando se soluciona el problema.

Para suspender una instancia de proceso, debe estar en el estado de ejecución o anómalo. El interlocutor debe ser un administrador de procesos o un administrador del sistema. Sin embargo, si Business Flow Manager está utilizando la modalidad alternativa de autorización de administración de proceso, que restringe la administración del proceso a los administraciones del sistema, sólo los interlocutores del rol BPESystemAdministrator pueden realizar esta acción.

Procedimiento

1. Obtenga el proceso en ejecución, CustomerOrder, que desea suspender.

   ```
   ProcessInstanceData processInstance =
                       process.getProcessInstance("CustomerOrder");
   ```

2. Suspenda la instancia de proceso.

   ```
   PIID piid = processInstance.getID();
   process.suspend( piid );
   ```

   Esta acción suspende la instancia de proceso de nivel superior especificada. La instancia de proceso se pone en estado suspendido. En este estado, las actividades que están iniciadas aún se pueden finalizar pero no se activan actividades nuevas. Los subprocesos con el atributo autonomy establecido en el valor `child` también se suspenden si están en el estado de en ejecución, anómalo, terminando o compensándose. Las tareas incorporadas y las tareas autónomas que están asociadas con esta instancia de proceso no se suspenden.

3. Reanude la instancia de proceso.

   ```
   process.resume( piid );
   ```

   Esta acción pone la instancia de proceso y sus subprocesos en los estados que tenían antes de suspenderse.

Reinicio de un proceso empresarial:

Puede reiniciar una instancia de proceso que esté en estado finalizado, terminado, anómalo o compensado.

Acerca de esta tarea

Reiniciar una instancia de proceso es similar a iniciar una instancia de proceso por primera vez. Sin embargo, cuando se reinicia una instancia de proceso, se conoce el ID de instancia de proceso y el mensaje de entrada de la instancia queda disponible.

Si el proceso tiene más de una actividad de recepción o de obtención (también conocida como una actividad de recepción y elección) que pueda crear la instancia de proceso, todos los mensajes que pertenecen a estas actividades se utilizan para reiniciar la instancia de proceso. Si cualquiera de estas actividades implementa una operación de petición-respuesta, la respuesta se vuelve a enviar cuando se navegue por la actividad de respuesta asociada.

El interlocutor debe ser un administrador de procesos o un administrador del sistema. Sin embargo, si Business Flow Manager está utilizando la modalidad alternativa de autorización de administración de proceso, que restringe la administración del proceso a los administraciones del sistema, sólo los interlocutores del rol BPESystemAdministrator pueden realizar esta acción.

Procedimiento

1. Obtenga el proceso que desea reiniciar.

   ```
   ProcessInstanceData processInstance =
                       process.getProcessInstance("CustomerOrder");
   ```

2. Reinicie la instancia de proceso.

   ```
   PIID piid = processInstance.getID();
   process.restart( piid );
   ```

   Esta acción reinicia la instancia de proceso especificada.

Terminar una instancia de proceso:

A veces, una instancia de proceso de nivel superior en un estado no recuperable se debe terminar.

Acerca de esta tarea

Para realizar esta acción, el emisor debe ser un administrador de procesos o un administrador del sistema. Sin embargo, si Business Flow Manager está utilizando la modalidad alternativa de autorización de administración de proceso, que restringe la administración del proceso a los administraciones del sistema, sólo los interlocutores del rol BPESystemAdministrator pueden realizar esta acción.

Dado que una instancia de proceso se termina inmediatamente, sin esperar a subprocesos o actividades pendientes, sólo debe terminar esta acción en situaciones excepcionales.

Procedimiento

1. Recupere la instancia de proceso que se ha de finalizar.

   ```
   ProcessInstanceData processInstance =
          process.getProcessInstance("CustomerOrder");
   ```

2. Finalizar la instancia de proceso.

   Si termina una instancia de proceso, puede terminar la instancia de proceso con o sin compensación.

   Para finalizar la instancia de proceso sin compensación:

```
PIID piid = processInstance.getID();
process.forceTerminate(piid, CompensationBehaviour.INVOKE_COMPENSATION);
```

Para finalizar la instancia de proceso sin compensación:

```
PIID piid = processInstance.getID();
process.forceTerminate(piid);
```

Si termina la instancia de proceso con compensación, la compensación del proceso se ejecuta como si se hubiera producido una anomalía en el ámbito de nivel superior. Si termina la instancia de proceso sin compensación, termina la instancia de proceso inmediatamente sin esperar a que las actividades, las tareas a realizar o las tareas de invocación en línea finalicen normalmente.

Las aplicaciones que inicia el proceso y las tareas autónomas que están relacionadas con el proceso no se terminan mediante la petición de forzar terminación. Si se han de terminar estas aplicaciones, debe añadir sentencias a la aplicación del proceso que termina explícitamente las aplicaciones iniciadas por el proceso.

Supresión de instancias de proceso:

Las instancias de proceso completadas se suprimen automáticamente de la base de datos de Business Process Choreographer si está establecida la propiedad correspondiente para la plantilla de proceso del modelo de proceso. Quizá desee conservar instancias de proceso en la base de datos, por ejemplo, para consultar datos de instancias de proceso que no se graban en las anotaciones cronológicas de auditoría. No obstante, los datos de la instancia de proceso almacenados no sólo afectan el espacio de disco y el rendimiento sino que también impiden que se creen instancias de proceso utilicen los mismos valores del conjunto de correlación. Por lo tanto, regularmente debe eliminar los datos de la instancia de proceso de la base de datos.

Acerca de esta tarea

Para suprimir una instancia de proceso, necesita derechos de administrador de procesos y la instancia de proceso debe ser una instancia de proceso de nivel superior.

En el ejemplo siguiente se muestra cómo suprimir todas las instancias de proceso finalizadas.

Procedimiento

1. Listar las instancias de proceso que han finalizado.

```
QueryResultSet result =
     process.query("DISTINCT PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                           PROCESS_INSTANCE.STATE.STATE_FINISHED",
                    (String)null, (Integer)null, (TimeZone)null);
```

   Esta acción devuelve un conjunto de resultados de consulta que lista las instancias de procesos finalizadas.

2. Suprima las instancias de proceso que hayan finalizado.

```
while (result.next() )
{
   PIID piid = (PIID) result.getOID(1);
   process.delete(piid);
}
```

   Esta acción suprime la instancia de proceso seleccionada y sus tareas en línea de la base de datos.

Proceso de actividades de tareas de usuario

Las actividades de tareas de usuario en procesos de empresa se asignan a distintas personas de la organización mediante elementos de trabajo. Cuando se inicia un proceso, se crean elementos de trabajo para los propietarios potenciales.

Acerca de esta tarea

Cuando se activa una actividad de tarea de usuario, se crean una instancia de actividad y una tarea a realizar asociada. El manejo de la actividad de tarea de usuario y la gestión de elementos de trabajo se delega al Gestor de tareas de usuario. Cualquier cambio de estado de la instancia de actividad se refleja en la instancia de tarea y viceversa.

Un propietario potencial reclama la actividad. Esta persona se encarga de proporcionar la información relevante y completar la actividad.

Procedimiento

1. Enumere las actividades que pertenecen a una persona que ha iniciado la sesión y que están preparadas para utilizarse:

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_READY AND
                    ACTIVITY.KIND = ACTIVITY.KIND.KIND_STAFF AND
                    WORK_ITEM.REASON =
                         WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                    (String)null, (Integer)null, (TimeZone)null);
```

   Esta acción devuelve un conjunto de resultados de consulta que contiene las actividades con las que puede trabajar la persona que ha iniciado la sesión.

2. Reclame la actividad en la que se va a trabajar:

```
if (result.size() > 0)
{
 result.first();
 AIID aiid = (AIID) result.getOID(1);
 ClientObjectWrapper input = process.claim(aiid);
 DataObject activityInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    activityInput = (DataObject)input.getObject();
    // leer los valores
    ...
  }
}
```

   Cuando se reclama la actividad, se devuelve el mensaje de entrada de la actividad.

3. Cuando haya acabado el trabajo en la actividad, finalice la actividad. La actividad puede completarse satisfactoriamente o con un mensaje de error. Si la actividad se realiza satisfactoriamente se pasa un mensaje de salida. Si no se realiza satisfactoriamente la actividad, se pone en el estado con anomalía o detenida y se pasa un mensaje de error. Deberá crear los mensajes adecuados para estas acciones. Cuando cree el mensaje, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje.
   a. Para completar la actividad correctamente, cree un mensaje de salida.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
```

```
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //establecer las partes del mensaje, por ejemplo, un número de pedido
  myMessage.setInt("OrderNo", 4711);
}

//completar la actividad
process.complete(aiid, output);
```

Esta acción establece un mensaje de salida que contiene el número de pedido.

b. Para completar la actividad cuando se produce un error, cree un mensaje de error.

```
//recuperar los errores diseñados para la actividad de tarea de usuario
List faultNames = process.getFaultNames(aiid);

//crear un mensaje del tipo adecuado
ClientObjectWrapper myFault =
      process.createMessage(aiid, faultNames.get(0));

// establecer las partes del mensaje de error, por ejemplo, un número
// de error
DataObject myMessage = null ;
if ( myFault.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)myFault.getObject();
  //establecer las partes del mensaje, por ejemplo, un nombre de cliente
  myMessage.setInt("error",1304);
}

process.complete(aiid, myFault,(String)faultNames.get(0) );
```

Esta acción establece la actividad en el estado con anomalía o detenida. Si el parámetro **continueOnError** de la actividad del modelo de proceso se establece en true, se pone la actividad en el estado con anomalía y continúa la navegación. Si el parámetro **continueOnError** se establece en false y el error no se captura en el ámbito que lo rodea, la actividad se pasa al estado detenido. En este estado se puede reparar la actividad utilizando force complete o force retry.

Conceptos relacionados

Comportamiento de continuar en error de las actividades y procesos empresariales
Cuando define un proceso empresarial, puede especificar lo que le sucede a un proceso si se genera un error imprevisto y no se ha definido un manejador de errores para ese error. Puede utilizar el valor **Continuar tras error** cuando defina el proceso para especificar que se ha detener donde se produce el error.

Proceso del flujo de trabajo de un solo usuario

Algunos flujos de trabajo sólo los realiza un usuario, por ejemplo pedir libros de una librería en línea. Este tipo de flujo de trabajo no tiene rutas paralelas. Las API initiateAndClaimFirst y completeAndClaimSuccessor soportan el proceso de este tipo de flujo de trabajo. Este ejemplo muestra la implementación de un único flujo de trabajo de persona utilizando un flujo de página del lado del cliente.

Acerca de esta tarea

También se hace referencia a un flujo de trabajo singular de persona como *flujo de página* o *flujo de pantalla*. Existen dos tipos de flujos de página:

- Los flujos de página del lado del cliente, donde la navegación entre las distintas páginas se realiza mediante la tecnología del lado del cliente como, por ejemplo, un formulario Lotus Forms de varias páginas.
- Los flujos de página del lado del servidor se realizan utilizando un proceso empresarial y un conjunto de tareas humanas que se han modelado de forma que las siguientes tareas se asignan a la misma persona.

Los flujos de página del lado del servidor son más potentes que los flujos de página del lado del cliente, pero consumen más recursos de servidor para procesarlos. Por lo tanto, considere utilizar el tipo de flujo de trabajo básicamente en las siguientes situaciones:

- Debe invocar servicios entre los pasos llevados a cabo en una interfaz de usuario, por ejemplo, para recuperar o actualizar datos.
- Tiene requisitos de auditoría que requieren que los sucesos CEI se escriban después de que se complete una interacción de interfaz de usuario.

Un ejemplo típico de un flujo de trabajo único de persona es el proceso de pedidos en una librería en línea, donde el comprador completa una secuencia de acciones para pedir un libro. Esta secuencia de acciones se puede implementar como una serie de actividades de tareas de usuario (tareas a realizar). Si el comprador decide pedir varios libros, esto equivale a iniciar un proceso de pedido y a reclamar la siguiente actividad de tarea de usuario.

La API initiateAndClaimFirst inicia el flujo de páginas, es decir, inicia el proceso especificado y reclama la primera actividad de tarea de usuario en la secuencia de actividades. Devuelve información sobre la actividad reclamada, incluido el mensaje de entrada sobre el que se va a actuar.

La API completeAndClaimSuccessor completa entonces la actividad de tarea de usuario y reclama la siguiente de la misma instancia de proceso para la persona que ha iniciado la sesión. Devuelve información sobre la siguiente actividad reclamada, incluido el mensaje de entrada sobre el que se va a actuar. Dado que la actividad siguiente está disponible dentro de la misma transacción de la actividad que se ha completado, el comportamiento transaccional de todas las actividades de tareas de usuario del modelo de proceso se debe establecer en `participates`.

Compare este ejemplo que utiliza la API de Business Flow Manager y la API de Human Task Manager.

Procedimiento

1. Inicie el proceso de pedido de libros y reclame la primera actividad en la secuencia de actividades. Iniciar el proceso con un mensaje de entrada del tipo adecuado. Cuando cree el mensaje, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje. Si especifica un nombre de instancia de proceso, no debe empezar con un carácter de subrayado. Si no se ha especificado un nombre de instancia de proceso, se utiliza como nombre el ID de instancia de proceso (PIID) en formato de serie.
   a. Recupere la plantilla de proceso para crear un mensaje de entrada del tipo apropiado.

```
ProcessTemplateData template = process.getProcessTemplate("CustomerOrder");
ClientObjectWrapper input = process.createMessage(template.getID(),
                            template.getInputMessageTypeName());
DataObject myMessage = null;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
```

```
  myMessage = (DataObject)input.getObject();
  //establecer las series del mensaje, por ejemplo, un nombre de cliente
  myMessage.setString("CustomerName", "Smith");
}
```

b. Inicie el proceso y reclame la primera actividad de tarea de usuario.

```
InitiateAndClaimFirstResult result =
  process.initiateAndClaimFirst("CustomerOrder", "MyOrderProcess", input);
AIID aiid = result.getAIID();
ClientObjectWrapper input = result.getInputMessage();
```

Cuando se reclama la primera actividad, se devuelven el mensaje de entrada y el ID de la actividad reclamada.

2. Cuando finalice el trabajo de la actividad, complete la actividad y relame la siguiente actividad.

   Para completar esta actividad, se pasa un mensaje de salida. Cuando cree el mensaje de salida, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //establecer las partes del mensaje, por ejemplo, un número de pedido
  myMessage.setInt("OrderNo", 4711);
}

//complete la actividad y reclame la siguiente
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

   Esta acción establece un mensaje de salida que contiene el número de pedido y reclama la siguiente actividad de la secuencia. Si se establece `AutoClaim` para las actividades de sucesor y hay varias vías de acceso que se pueden seguir, se reclaman todas las actividades de sucesor y se devuelve una actividad aleatoria como la actividad siguiente. Si no hay más actividades de sucesor que se puedan asignar a este usuario, se devuelve `Null`.

   Si el proceso contiene vías de acceso paralelas que se pueden seguir y estas vías de acceso contienen actividades de tareas de usuario para las que el usuario conectado es un propietario potencial de más de una de estas actividades, se reclama automáticamente una actividad aleatoria y se devuelve como la actividad siguiente.

3. Trabaje en la siguiente actividad.

```
String name = successor.getActivityName();

ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // leer los valores
  ...
}

aiid = successor.getAIID();
```

4. Continúe con el paso 2 para completar la actividad.

Tareas relacionadas

“Proceso del flujo de trabajo de una sola persona que incluye tareas de usuario” en la página 449
Algunos flujos de trabajo sólo los realiza un usuario, por ejemplo pedir libros de una librería en línea. Este ejemplo muestra cómo implementar un flujo de trabajo único de persona utilizando un flujo de página del lado del servidor. Se utiliza Business Flow Manager y las API de Human Task Manager para procesar el flujo de trabajo.

Envío de un mensaje a una actividad en espera

Puede utilizar las actividades de mensajes de entrada (actividades de recepción, onMessage en actividades de captación, onEvent en manejadores de sucesos) para sincronizar un proceso en ejecución con sucesos del "mundo exterior". Por ejemplo, un suceso de este tipo puede ser cuando se recibe un correo electrónico de un cliente como respuesta a una petición de información.

Acerca de esta tarea

Para enviar el mensaje a la actividad puede utilizar las tareas que la originan.

Procedimiento

1. Liste las plantillas de servicios de actividades que están a la espera de un mensaje del usuario conectado en una instancia de proceso con un ID de instancia de proceso específico.

   ```
   ActivityServiceTemplateData[] services = process.getWaitingActivities(piid);
   ```

2. Envíe un mensaje al primer servicio en espera.

   Se presupone que el primer servicio es el que desea servir. El llamante debe ser el iniciador potencial de la actividad que recibe el mensaje o un administrador de la instancia de proceso.

   ```
   VTID vtid = services[0].getServiceTemplateID();
   ATID atid = services[0].getActivityTemplateID();
   String inputType = services[0].getInputMessageTypeName();

   // crear un mensaje para el servicio que se va a llamar
     ClientObjectWrapper message =
           process.createMessage(vtid,atid,inputMessageTypeName);
     DataObject myMessage = null;
     if ( message.getObject()!=
           null && message.getObject() instanceof DataObject )
     {
       myMessage = (DataObject)message.getObject();
       // establecer las series del mensaje, por ejemplo, se va a realizar
       // un pedido de chocolate
       myMessage.setString("Order", "chocolate");
     }

     // enviar el mensaje a la actividad que está en espera
     process.sendMessage(vtid, atid, message);
   }
   ```

   Esta acción envía el mensaje especificado al servicio de actividades en espera y se pasarán algunos datos del pedido.

   También puede especificar el ID de instancia de proceso para asegurarse de que se envía el mensaje a la instancia de proceso especificada. Si no se especifica el ID de instancia de proceso, se envía el mensaje al servicio de actividades y a la instancia de proceso que identifican los valores de correlación del mensaje. Si se especifica el ID de instancia de proceso, se comprueba la instancia de proceso

que se ha encontrado utilizando los valores de correlación para asegurarse de que tiene el ID de instancia de proceso especificado.

Manejo de sucesos

Un proceso de empresa completo y cada uno de sus ámbitos puede asociarse con manejadores de sucesos que se invocan si se produce el suceso asociado. Los manejadores de sucesos son similares para recibir o seleccionar actividades en lo referente a que un proceso puede proporcionar operaciones de servicios Web mediante manejadores de sucesos.

Acerca de esta tarea

Puede invocar un manejador de sucesos cualquier número de veces mientras se ejecute el ámbito correspondiente. Además, varias instancias de un manejador de sucesos pueden activarse de forma simultánea.

El siguiente fragmento de código muestra cómo obtener los manejadores de sucesos activos para una instancia de proceso determinada y cómo enviar un mensaje de entrada.

Procedimiento

1. Determine los datos del ID de instancia de proceso y liste los manejadores de sucesos activos para el proceso.

```
ProcessInstanceData processInstance =
       process.getProcessInstance( "CustomerOrder2711");
EventHandlerTemplateData[] events = process.getActiveEventHandlers(
                                        processInstance.getID() );
```

2. Envíe el mensaje de entrada.

   Este ejemplo utiliza el primer manejador de sucesos que se encuentra.

```
EventHandlerTemplateData event = null;
if ( events.length > 0 )
{
    event = events[0];

    // crear un mensaje para el servicio que se va a llamar
    ClientObjectWrapper input = process.createMessage(
    event.getID(), event.getInputMessageTypeName());

    if (input.getObject() != null && input.getObject() instanceof DataObject )
    {
       DataObject inputMessage = (DataObject)input.getObject();
       // establecer contenido del mensaje, por ejemplo, un nombre de
       // cliente y número de pedido
       inputMessage.setString("CustomerName", "Smith");
       inputMessage.setString("OrderNo", "2711");

        // enviar el mensaje
       process.sendMessage( event.getProcessTemplateName(),
                            event.getPortTypeNamespace(),
                            event.getPortTypeName(),
                            event.getOperationName(),
     input );
     }
 }
```

   Esta acción envía el mensaje especificado al manejador de sucesos activo para el proceso.

Análisis de los resultados de un proceso

Un proceso puede exponer las operaciones de servicios Web que están diseñadas como operaciones unidireccionales o de petición y respuestas de tipo WSDL (Web Services Description Language). Los resultados de procesos de larga duración con interfaces unidireccionales no pueden recuperarse mediante el método getOutputMessage porque el proceso no tiene salida. En cambio, sin embargo, puede consultar el contenido de las variables.

Acerca de esta tarea

Los resultados del proceso sólo se almacenan en la base de datos si la plantilla de proceso de la que se ha derivado la instancia de proceso no especifica que el mensaje de salida se ha de suprimir automáticamente.

Procedimiento

Analizar los resultados del proceso, por ejemplo, comprobar el número de pedido.

```
QueryResultSet result = process.query
                       ("PROCESS_INSTANCE.PIID",
                        "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                         PROCESS_INSTANCE.STATE =
                                 PROCESS_INSTANCE.STATE.STATE_FINISHED",
                         (String)null, (Integer)null, (TimeZone)null); if (result.size() > 0)
{
  result.first();
  PIID piid = (PIID) result.getOID(1);
  ClientObjectWrapper output = process.getOutputMessage(piid);
  DataObject myOutput = null;
  if ( output.getObject() != null && output.getObject() instanceof DataObject )
  {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
  }
}
```

Reparación de actividades

Un proceso de larga ejecución puede contener actividades que también son de larga ejecución. Estas actividades pueden encontrar errores no descubiertos y pasar al estado detenido. Una actividad que está en estado de en ejecución también podría parecer que no está respondiendo. En los dos casos, un administrador de procesos puede actuar sobre la actividad de distintos modos de manera que pueda continuar la navegación del proceso.

Acerca de esta tarea

La API de Business Process Choreographer proporciona los métodos forceRetry y forceComplete para reparar las actividades. Se proporcionan ejemplos que muestran cómo se pueden añadir acciones de reparación de actividades a las aplicaciones del usuario.

Forzar la finalización de una actividad:

A veces, las actividades de procesos de larga ejecución pueden encontrar errores. Si estos errores no son captados por el manejador de errores en el ámbito circundante y si la plantilla de la actividad asociada especifica que la actividad se detenga cuando se produzca un error, la actividad se pasará al estado de detenida para que se pueda reparar. En este estado, puede forzar la finalización de la actividad.

Acerca de esta tarea

También puede forzar la finalización de actividades en estado de ejecución si, por ejemplo, una actividad no responde.

Existen requisitos adicionales para ciertos tipos de actividades.

Actividades de las tareas de usuario
Puede pasar parámetros a la llamada force-complete, como el mensaje que debería haberse enviado o el error que debería haberse producido.

Actividades de script
No puede pasar parámetros en la llamada force-complete. No obstante, debe establecer las variables que se han de reparar.

Invocar actividades
También puede forzar actividades de invocación completas que llaman a un servicio asíncrono que no sea un subproceso si estas actividades están en estado de ejecución. Puede que desee hacer esto, por ejemplo, si se llama al servicio asíncrono y éste no responde.

Procedimiento

1. Listar las actividades detenidas en estado detenido.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                    (String)null, (Integer)null, (TimeZone)null);
```

   Esta acción devuelve las actividades detenidas para la instancia de proceso CustomerOrder.

2. Completar la actividad, por ejemplo, una actividad de tarea de usuario detenida.

   En este ejemplo, se pasa un mensaje de salida.

```
if (result.size() > 0)
{
   result.first();
   AIID aiid = (AIID) result.getOID(1);
   ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper output =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  DataObject myMessage = null;
  if ( output.getObject()!= null && output.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)output.getObject();
     //establecer las partes del mensaje, por ejemplo, un número de pedido
     myMessage.setInt("OrderNo", 4711);
   }

   boolean continueOnError = true;
   process.forceComplete(aiid, output, continueOnError);
}
```

   Esta acción completa la actividad. Si se produce un error, el parámetro **continueOnError** determina la acción que se va a llevar a cabo si se proporciona un error con la petición forceComplete.

   En el ejemplo, el valor de **continueOnError** es true. Este valor significa que si se proporciona un error, la actividad se coloca en estado de error. El error se propaga a los ámbitos que circundan la actividad hasta que se maneja o hasta que se alcanza el ámbito del proceso. A continuación, el proceso se pone en estado de ejecución errónea hasta que finalmente pasa a estado erróneo.

Conceptos relacionados

 Comportamiento de continuar en error de las actividades y procesos empresariales
Cuando define un proceso empresarial, puede especificar lo que le sucede a un proceso si se genera un error imprevisto y no se ha definido un manejador de errores para ese error. Puede utilizar el valor **Continuar tras error** cuando defina el proceso para especificar que se ha detener donde se produce el error.

Reintento de la ejecución de una actividad detenida:

Si durante una actividad de un proceso de larga ejecución se produce un error no capturado en el ámbito que la circunda y si la plantilla de la actividad asociada especifica que la actividad se detenga cuando se produzca un error, la actividad se pasa al estado de detenida para que se pueda reparar. Puede intentar la ejecución de la actividad otra vez.

Acerca de esta tarea

Puede establecer las variables que utiliza la actividad. A excepción de las actividades de script, también puede pasar parámetros en la llamada force-retry como, por ejemplo, el mensaje que esperaba la actividad.

Procedimiento

1. Listar las actividades detenidas.

```
QueryResultSet result =
      process.query("DISTINCT ACTIVITY.AIID",
                    "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                     PROCESS_INSTANCE.NAME='CustomerOrder'",
                     (String)null, (Integer)null, (TimeZone)null);
```

   Esta acción devuelve las actividades detenidas para la instancia de proceso CustomerOrder.

2. Reintentar la ejecución de la actividad, por ejemplo, una actividad de tareas de usuario detenida.

```
if (result.size() > 0)
{
  result.first();
     AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);
  ClientObjectWrapper input =
        process.createMessage(aiid, activity.getOutputMessageTypeName());
  DataObject myMessage = null;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
    {
      myMessage = (DataObject)input.getObject();
      // establecer las series del mensaje, por ejemplo, se va a realizar un
      // pedido de chocolate
      myMessage.setString("OrderNo", "chocolate");
    }

   boolean continueOnError = true;
   process.forceRetry(aiid, input, continueOnError);
}
```

   Esta acción reintenta la actividad. Si se produce un error, el parámetro **continueOnError** determina la acción que se realizará si se produce un error durante el proceso de la petición forceRetry.

   En el ejemplo, el valor de **continueOnError** es true. Esto significa que si se produce un error durante el proceso de la petición forceRetry, la actividad se

pasa a estado de ejecución errónea. El error se propaga a los ámbitos que circundan la actividad hasta que se maneja o hasta que se alcanza el ámbito del proceso. A continuación, el proceso se pone en estado de ejecución errónea y se ejecuta un manejador de errores en el nivel de proceso antes de que el estado de proceso termine en el estado erróneo.

Conceptos relacionados

 Comportamiento de continuar en error de las actividades y procesos empresariales
Cuando define un proceso empresarial, puede especificar lo que le sucede a un proceso si se genera un error imprevisto y no se ha definido un manejador de errores para ese error. Puede utilizar el valor **Continuar tras error** cuando defina el proceso para especificar que se ha detener donde se produce el error.

Reparación de actividades que se han detenido porque ha fallado una unión, un bucle o una evaluación de contador:

Las actividades se pueden detener porque se ha producido una excepción cuando se evaluaba una condición de unión o bucle o un valor de contador forEach. El administrador decide no volver a intentar la ejecución de la actividad, por ejemplo porque la evaluación puede fallar otra vez. En tales casos, se pueden proporcionar los valores correctos para la expresión utilizando la API EJB de Business Process Choreographer para que la navegación del proceso pueda continuar.

Acerca de esta tarea

Puede establecer el valor de una condición de unión para cualquier tipo de actividad, el valor de una condición de bucle de una actividad while o repeat-until. También puede establecer los valores de los contadores iniciales y finales y el número máximo de ramas completadas para una actividad forEach. El valor que establece para las ramas completadas depende de la definición de la actividad forEach del modelo de proceso. Si se especifica una condición de salida anticipada en el modelo, establezca un valor para el máximo de ramas completadas. Si no se especifica una condición de salida anticipada, establezca el valor del máximo de ramas completadas en `null`.

El ejemplo siguiente muestra cómo establecer el valor de una condición de bucle.

Procedimiento

1. Liste las actividades que se han detenido porque ha fallado la evaluación de una condición de bucle.

```
QueryResultSet result = process.query(
  "DISTINCT ACTIVITY.AIID",
  "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
   ACTIVITY.STOP_REASON = ACTIVITY.STOP_REASON.STOP_REASON_IMPLEMENTATION_FAILED AND
   (ACTIVITY.KIND = ACTIVITY.KIND.KIND_WHILE OR
      ACTIVITY.KIND = ACTIVITY.KIND.KIND_REPEAT_UNTIL) AND
   PROCESS_INSTANCE.NAME='CustomerOrder'",
   (String)null, (Integer)null, (TimeZone)null );
```

   De manera similar, puede listar las actividades que se han detenido porque ha fallado la evaluación de una condición de unión o un contador forEach.

   - Para una condición de unión anómala, utilice la expresión siguiente:

     ```
     ACTIVITY.STOP_REASON.STOP_REASON_ACTIVATION_FAILED
     ```

   - Para un contador forEach anómalo, utilice la expresión siguiente:

```
ACTIVITY.STOP_REASON.STOP_REASON_IMPLEMENTATION_FAILED AND
(ACTIVITY.KIND = ACTIVITY.KIND.KIND_FOR_EACH_SERIAL OR
   ACTIVITY.KIND = ACTIVITY.KIND.KIND_FOR_EACH_PARALLEL)
```

Esta acción devuelve las actividades para la instancia de proceso CustomerOrder que se ha detenido porque ha fallado la evaluación de una condición de bucle.

2. Proporcione el valor de la condición de bucle, por ejemplo true.

```
if (result.size() > 0)
{
  result.first();
  AIID aiid = (AIID) result.getOID(1);

  process.forceLoopCondition(aiid, true);
}
```

Esta acción establece en true el valor de la condición de bucle para la actividad y la navegación de la instancia de proceso continúa.

De manera similar, puede establecer el valor de una condición de unión (`process.forceJoinCondition(aiid, true);`) o los valores de los contadores de actividad forEach (`process.forceForEachCounterValues(aiid, 1, 5, new Integer(2));`).

Actualización de conjuntos de correlaciones asociados con actividades detenidas:

Los conjuntos de correlaciones se utilizan para soportar la colaboración con estado entre servicios Web. En tales casos, se pueden proporcionar los valores correctos para la expresión utilizando la API EJB de Business Process Choreographer para que la navegación del proceso pueda continuar.

Acerca de esta tarea

Una actividad que está en un estado detenido puede necesitar una actualización del conjunto de correlaciones asociado por una de las razones siguientes:

- Se ha producido una excepción al evaluar el conjunto de correlaciones. El conjunto de correlaciones se debe inicializar pero ya está inicializado.
- Se ha producido una excepción al evaluar el conjunto de correlaciones. El conjunto de correlaciones no se debe inicializar pero los valores no se han establecido. Esto se puede producir, por ejemplo porque se ha omitido una actividad de inicialización.
- Es necesario volver a intentar la actividad. Si la actividad inicializa el conjunto de correlaciones, se puede eliminar la inicialización del mismo o cambiarlo antes de que se llame al método forceRetry.
- Es necesario completar la actividad. Si la actividad inicializa el conjunto de correlaciones, se puede eliminar la inicialización del mismo o cambiarlo antes de que se llame al método forceComplete.

Puede recuperar las instancias de conjunto de correlación de una instancia de proceso o actividad. El ejemplo siguiente muestra cómo inicializar instancias de conjuntos de correlaciones o eliminar la inicialización de dichas instancias.

Procedimiento

1. Listar las actividades detenidas en estado detenido.

```
QueryResultSet result =
     process.query("DISTINCT ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    PROCESS_INSTANCE.NAME='CustomerOrder'",
                    (String)null, (Integer)null, (TimeZone)null);
```

Esta acción devuelve las actividades detenidas para la instancia de proceso CustomerOrder.

2. Recupere las instancias de conjunto de correlaciones que están definidas para la actividad.

```
AIID aiid = null;

List correlationSet = null;

if (result.size() > 0)
{
   result.first();
   AIID aiid = (AIID) result.getOID(1);

   ActivityInstanceData activity = process.getActivityInstance(aiid);

   correlationSet = process.getCorrelationSetInstances
                    (aiid, activity.getInputMessageTypeName());
}
```

3. Elimine la inicialización del conjunto de correlaciones, por ejemplo MyCorrelationSet.

```
for ( int i=0; i<correlationSet.size(); i++ )
{
  CorrelationSetInstanceData correlationSetInstance =
            (CorrelationSetInstanceData)correlationSet.get(i);

  if ( correlationSetInstance.isInitialized() &&
       correlationSetInstance.getCorrelationSetName().equals("MyCorrelationSet") )
  {
    process.uninitializeCorrelationSet
       ( activity.getProcessInstanceID(), correlSetInstance.getCorrelationSetName() );
  }
}
```

Esta acción elimina la inicialización del conjunto de correlaciones MyCorrelationSet.

4. Inicialice el conjunto de correlaciones, por ejemplo MyCorrelationSet. En este ejemplo, se establece una propiedad con valor de serie del conjunto de correlaciones.

```
for ( int i=0; i<correlationSet.size(); i++ )
{
  CorrelationSetInstanceData correlationSetInstance =
            (CorrelationSetInstanceData)correlationSet.get(i);

  if ( correlationSetInstance.getCorrelationSetName().equals("MyCorrelationSet") )
  {
    List correlationSetProperties =
                    correlationSetInstance.getCorrelationSetProperties();
    for ( int j=0; j<correlationSetProperties.size(); j++ )
    {
      CorrelationPropertyInstanceData property =
                (CorrelationPropertyInstanceData)correlationSetProperties.get(j);

      if ( property.getPropertyName().equals("MyProperty") )
      {
        property.setValue("NewValue");

        process.initializeCorrelationSet
           ( activity.getProcessInstanceID(), correlationSetInstance );
      }
    }
  }
}
```

Esta acción inicializa la propiedad con valor de serie MyProperty en el conjunto de correlaciones MyCorrelationSet.

Interfaz BusinessFlowManagerService

La interfaz BusinessFlowManagerService expone funciones de proceso empresarial que una aplicación cliente puede llamar.

Los métodos que la interfaz BusinessFlowManagerService puede llamar dependen del estado del proceso o la actividad y la autorización de la persona que utilice la aplicación que contiene el método. Los métodos principales para manejar objetos de proceso empresarial se listan aquí. Para obtener más información sobre estos y otros métodos que están disponibles en la interfaz BusinessFlowManagerService, consulte el Javadoc que se encuentra en el paquete com.ibm.bpe.api.

Plantillas de proceso

Una plantilla de proceso es un modelo de proceso versionado, desplegado e instalado que contiene la especificación de un proceso empresarial. Se puede crear la instancia e iniciarse emitiendo las solicitudes apropiadas, por ejemplo sendMessage(). La ejecución de la instancia de proceso la dirige automáticamente el servidor.

Tabla 60. Métodos API para plantillas de proceso

Método	Descripción
getProcessTemplate	Recupera la plantilla de proceso especificada.
queryProcessTemplates	Recupera plantillas de proceso que se almacenan en la base de datos.

Instancias de proceso

Los siguientes métodos API están relacionados con el inicio de instancias de proceso.

Tabla 61. Métodos API que están relacionados con el inicio de instancias de proceso

Método	Descripción
call	Crea y ejecuta un microflujo.
callWithReplyContext	Crea y ejecuta un microflujo con un servicio de arranque exclusivo o un proceso de larga ejecución con un servicio de arranque exclusivo a partir de la plantilla de proceso especificada. La llamada espera de forma asíncrona al resultado.
callWithUISettings	Crea y ejecuta un microflujo y devuelve el mensaje de salida y los valores de la interfaz de usuario (UI) de cliente.
initiate	Crea una instancia de proceso e inicia el proceso de la instancia de proceso. Utilice este método para procesos de larga ejecución. También puede utilizar este método para microflujos que desea activar y omitir.

Tabla 61. Métodos API que están relacionados con el inicio de instancias de proceso (continuación)

Método	Descripción
initiateAndSuspend	Crea una instancia de proceso pero suspende inmediatamente el proceso adicional de la instancia de proceso.
initiateAndClaimFirst	Crea una instancia de proceso y reclama la primera tarea de usuario incorporada.
sendMessage	Envía el mensaje especificado al servicio de actividad y la instancia de proceso especificados. Si no existe una instancia de proceso con los mismos valores de conjunto de correlaciones, se creará. El proceso puede tener servicios de arranque exclusivos y no exclusivos.
getStartActivities	Devuelve información sobre las actividades que pueden iniciar una instancia de proceso a partir de la plantilla de proceso especificada.
getActivityServiceTemplate	Recupera la plantilla de servicio de actividad especificada.

Tabla 62. Métodos API para controlar el ciclo de vida de las instancias de proceso

Método	Descripción
suspend	Suspende la ejecución de una instancia de proceso de nivel superior y larga ejecución que está en el estado de ejecución o anómalo.
resume	Reanuda la ejecución de una instancia de proceso de nivel superior y larga ejecución que está en el estado suspendido.
restart	Reinicia una instancia de proceso de nivel superior y larga ejecución en el estado finalizado, anómalo o terminado.
forceTerminate	Termina la instancia de proceso de nivel superior especificada, sus subprocesos con autonomía de hijo y sus actividades de ejecución, reclamadas o en espera.
delete	Suprime la instancia de proceso de nivel superior especificada y sus subprocesos con autonomía de hijo.
query	Recupera las propiedades de la base de datos que cumplen los criterios de búsqueda.
queryEntities	Utiliza tablas de consulta para recuperar de la base de datos las propiedades que coinciden con los criterios de búsqueda.
getWaitingActivities	Devuelve información sobre las actividades que están esperando un mensaje para que pueda continuar el proceso de estas actividades.

Tabla 62. Métodos API para controlar el ciclo de vida de las instancias de proceso (continuación)

Método	Descripción
migrate	Migra una instancia de proceso a la versión más reciente especificada del modelo de proceso.

Actividades

Para las actividades de invocación, puede especificar en el modelo de proceso que estas actividades continúan en situaciones de error. Si el distintivo continueOnError se establece en false y se produce un error no manejado, la actividad se coloca en estado detenido. A continuación, un administrador de proceso puede reparar la actividad. El distintivo continueOnError y las funciones de reparación asociadas pueden, por ejemplo, utilizarse en un proceso de larga ejecución en el que ocasionalmente falla una actividad de invocación pero el esfuerzo necesario para modelar la compensación y la gestión de errores es demasiado elevado.

Los métodos siguientes están disponibles para trabajar con actividades y repararlas.

Tabla 63. Métodos API para controlar el ciclo de vida de las instancias de actividad

Método	Descripción
claim	Reclama una instancia de actividad preparada para que un usuario trabaje en la actividad.
cancelClaim	Cancela la reclamación de la instancia de actividad.
complete	Completa la instancia de actividad.
completeAndClaimSuccessor	Completa la instancia de actividad y reclama la siguiente en la misma instancia de proceso para la persona que ha iniciado la sesión.
forceComplete	Fuerza la finalización de lo siguiente: • Una instancia de actividad que está en el estado ejecución o detenido. • Una actividad de tarea de usuario que está en el estado preparado o solicitado. • Una actividad de espera en el estado esperando.
forceRetry	Fuerza la repetición de lo siguiente: • Una instancia de actividad que está en el estado ejecución o detenido. • Una actividad de tarea de usuario que está en el estado preparado o solicitado.
forceNavigate, forceForEach, forceLoop, forceJoin	Estos métodos fuerzan la navegación de una actividad detenida.
skip	Omite el proceso de la actividad.
jump	Salta de una actividad a otra.

Tabla 63. Métodos API para controlar el ciclo de vida de las instancias de actividad (continuación)

Método	Descripción
query	Recupera las propiedades de la base de datos que cumplen los criterios de búsqueda.
queryEntities	Utiliza tablas de consulta para recuperar de la base de datos las propiedades que coinciden con los criterios de búsqueda.

Variables y propiedades personalizadas

La interfaz proporciona un método get y set para recuperar y establecer valores para variables. También puede asociar las propiedades con nombre con, y recuperar propiedades con nombre de, las instancias de proceso y actividad. Los nombres y valores de propiedad personalizados deben ser del tipo java.lang.String.

Tabla 64. Métodos API para variables y propiedades personalizadas

Método	Descripción
getVariable	Recupera la variable especificada.
setVariable	Establece la variable especificada.
getCustomProperty	Recupera la propiedad personalizada indicada de la instancia de actividad o proceso especificada.
getCustomProperties	Recupera las propiedades personalizadas de la actividad especificada o instancia de proceso.
getCustomPropertyNames	Recupera los nombres de las propiedades personalizadas de la instancia de actividad o proceso especificada.
setCustomProperty	Almacena valores específicos personalizados para la instancia de actividad o proceso especificada.

Desarrollo de aplicaciones para tareas de usuario

Una tarea consiste en los medios con los que los componentes invocan a usuarios como servicios o con los que los usuarios invocan servicios. Se proporcionan ejemplos de aplicaciones típicas para tareas de usuario.

Acerca de esta tarea

Para obtener más información sobre la API del Gestor de tareas de usuario, consulte el Javadoc en el paquete com.ibm.task.api.

Inicio de una tarea de invocación que invoca una interfaz síncrona

Una tarea de invocación está asociada con un componente SCA (Java Service Component Architecture). Cuando se inicia la tarea, invoca el componente SCA. Inicie una tarea de invocación de forma síncrona sólo si el componente SCA asociado puede invocarse de forma síncrona.

Acerca de esta tarea

Por ejemplo, este tipo de componente SCA puede implementarse como un microflujo o como una simple clase Java.

Este escenario crea una instancia de una plantilla de tarea y pasa algunos datos de cliente. La tarea permanece en estado de ejecución hasta que se devuelve la operación bidireccional. El resultado de la tarea, OrderNo, se devuelve al emisor de la llamada.

Procedimiento

1. Opcional: Liste las plantillas de tarea para encontrar el nombre de la tarea de invocación que desea ejecutar.

   Este paso es opcional si ya conoce el nombre de la tarea.

   ```
   TaskTemplate[] taskTemplates = task.queryTaskTemplates
   ("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
    "TASK_TEMPL.NAME",
     new Integer(50),
     (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas originadoras clasificadas.

2. Cree un mensaje de entrada del tipo adecuado.

   ```
   TaskTemplate template = taskTemplates[0];

   // crear un mensaje para la tarea seleccionada
   ClientObjectWrapper input = task.createInputMessage( template.getID());
   DataObject myMessage = null ;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //establecer las partes del mensaje, por ejemplo, un nombre de cliente
     myMessage.setString("CustomerName", "Smith");
   }
   ```

3. Cree la tarea y ejecútela de forma síncrona.

   Para que una tarea se ejecute de forma síncrona, debe ser una operación bidireccional. El ejemplo utiliza el método createAndCallTask para crear y ejecutar la tarea.

   ```
   ClientObjectWrapper output = task.createAndCallTask( template.getName(),
                                                        template.getNamespace(),
                                                        input);
   ```

4. Analice el resultado de la tarea.

   ```
   DataObject myOutput = null;
   if ( output.getObject() != null && output.getObject() instanceof DataObject )
   {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
   }
   ```

Inicio de una tarea de invocación que invoca una interfaz asíncrona

Una tarea de invocación está asociada con un componente SCA (Java Service Component Architecture). Cuando se inicia la tarea, invoca el componente SCA. Inicie una tarea de invocación de forma asíncrona sólo si el componente SCA asociado puede invocarse de forma asíncrona.

Acerca de esta tarea

Por ejemplo, este tipo de componente SCA puede implementarse como un proceso de larga ejecución o una operación de una dirección.

Este escenario crea una instancia de una plantilla de tarea y pasa algunos datos de cliente.

Procedimiento

1. Opcional: Liste las plantillas de tarea para encontrar el nombre de la tarea de invocación que desea ejecutar.

   Este paso es opcional si ya conoce el nombre de la tarea.

   ```
   TaskTemplate[] taskTemplates = task.queryTaskTemplates
     ("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_ORIGINATING",
      "TASK_TEMPL.NAME",
       new Integer(50),
       (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas originadoras clasificadas.

2. Cree un mensaje de entrada del tipo adecuado.

   ```
   TaskTemplate template = taskTemplates[0];

   // crear un mensaje para la tarea seleccionada
   ClientObjectWrapper input = task.createInputMessage( template.getID());
   DataObject myMessage = null ;
   if ( input.getObject()!= null && input.getObject() instanceof DataObject )
   {
     myMessage = (DataObject)input.getObject();
     //establecer las partes del mensaje, por ejemplo, un nombre de cliente
     myMessage.setString("CustomerName", "Torres");
   }
   ```

3. Cree la tarea y ejecútela de forma asíncrona.

   El ejemplo utiliza el método createAndStartTask para crear y ejecutar la tarea.

   ```
   task.createAndStartTask( template.getName(),
                            template.getNamespace(),
                            input,
                            (ReplyHandlerWrapper)null);
   ```

Creación e inicio de una instancia de tarea

En este escenario se muestra cómo crear una instancia de una plantilla de tarea que define una tarea de colaboración (también conocida como *tarea de usuario* de la API) e inicia la instancia de tarea.

Procedimiento

1. Opcional: Liste las plantillas de tarea para buscar el ID de plantilla de tarea (TKTID) correspondiente a la tarea de colaboración que desea ejecutar.

   Este paso es opcional si ya conoce el ID de plantilla de tarea.

   ```
   TaskTemplate[] taskTemplates = task.queryTaskTemplates
   ("TASK_TEMPL.KIND=TASK_TEMPL.KIND.KIND_HUMAN",
    "TASK_TEMPL.NAME",
     new Integer(50),
     (TimeZone)null);
   ```

   El resultado se clasifica por nombre. La consulta devuelve una matriz que contiene las 50 primeras plantillas de tareas clasificadas.

2. Cree un mensaje de entrada del tipo adecuado.

```
TaskTemplate template = taskTemplates[0];

// crear un mensaje para la tarea seleccionada
ClientObjectWrapper input = task.createInputMessage( template.getID());
DataObject myMessage = null ;
if ( input.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)input.getObject();
  //establecer las partes del mensaje, por ejemplo, un nombre de cliente
  myMessage.setString("CustomerName", "Smith");
}
```

3. Cree e inicie la tarea de colaboración; en este ejemplo no se especifica un manejador de respuestas.

   En el ejemplo se utiliza el método createAndStartTask para crear e iniciar la tarea.

   ```
   TKIID tkiid = task.createAndStartTask( template.getName(),
                                          template.getNamespace(),
                                          input,
                                          (ReplyHandlerWrapper)null);
   ```

   Se crearán elementos de trabajo de los usuarios a los que les interesa la instancia de tarea. Por ejemplo, un propietario potencial puede reclamar la nueva instancia de tarea.

4. Reclame la instancia de tarea.

   ```
   ClientObjectWrapper input2 = task.claim(tkiid);
   DataObject taskInput = null ;
   if ( input2.getObject()!= null && input2.getObject() instanceof DataObject )
   {
     taskInput = (DataObject)input2.getObject();
     // leer los valores
     ...
   }
   ```

   Cuando se reclama la instancia de tarea, se devuelve el mensaje de entrada de la tarea.

Proceso de tareas a realizar o de colaboración

Las tareas a realizar (también conocidas como *tareas participativas* en la API) o tareas de colaboración (también conocidas como *tareas de usuario* en la API) se asignan a varias personas de la organización mediante elementos de trabajo. Las tareas a realizar y sus elementos de trabajo asociados se crean, por ejemplo, cuando un proceso navega hacia una actividad de tareas de usuario.

Acerca de esta tarea

Uno de los propietarios potenciales reclama la tarea asociada con el elemento de trabajo. Esta persona es responsable de proporcionar la información relevante y completar la tarea.

Procedimiento

1. Liste las tareas pertenecientes a una persona que ha iniciado la sesión y que están preparadas para trabajar con ellas.

   ```
   QueryResultSet result =
        task.query("TASK.TKIID",
                   "TASK.STATE = TASK.STATE.STATE_READY AND
                   (TASK.KIND = TASK.KIND.KIND_PARTICIPATING OR
                    TASK.KIND = TASK.KIND.KIND_HUMAN)AND
                    WORK_ITEM.REASON =
                      WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                    (String)null, (Integer)null, (TimeZone)null);
   ```

Esta acción devuelve un conjunto de resultados de consulta que contiene las tareas con las que puede trabajar la persona que ha iniciado la sesión.

2. Reclame la tarea en la que se va a trabajar.

```
if (result.size() > 0)
{
  result.first();
  TKIID tkiid = (TKIID) result.getOID(1);
  ClientObjectWrapper input = task.claim(tkiid);
  DataObject taskInput = null ;
  if ( input.getObject()!= null && input.getObject() instanceof DataObject )
  {
    taskInput = (DataObject)input.getObject();
    // leer los valores
    ...
  }
}
```

Cuando se reclama la tarea, se devuelve el mensaje de entrada de la tarea.

3. Cuando haya acabado el trabajo en la tarea, complete la tarea.

La tarea se puede completar satisfactoriamente o con un mensaje de error. Si la tarea se realiza satisfactoriamente se pasa un mensaje de salida. Si la tarea no se realiza satisfactoriamente se pasa un mensaje de error. Deberá crear los mensajes adecuados para estas acciones.

a. Para completar la tarea correctamente, cree un mensaje de salida.

```
ClientObjectWrapper output =
      task.createOutputMessage(tkiid);
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //establecer las partes del mensaje, por ejemplo, un número de pedido
  myMessage.setInt("OrderNo", 4711);
}

//completar la tarea
task.complete(tkiid, output);
```

Esta acción establece un mensaje de salida que contiene el número de pedido. La tarea se coloca en el estado de finalizada.

b. Para completar la tarea cuando se produce un error, cree un mensaje de error.

```
//recuperar los errores diseñados para la tarea
List faultNames = task.getFaultNames(tkiid);

//crear un mensaje del tipo adecuado
ClientObjectWrapper myFault =
      task.createFaultMessage(tkiid, (String)faultNames.get(0));

// establecer las partes del mensaje de error, por ejemplo, un número
// de error
DataObject myMessage = null ;
if ( myFault.getObject()!= null && input.getObject() instanceof DataObject )
{
  myMessage = (DataObject)myFault.getObject();
  //establecer las partes del mensaje, por ejemplo, un nombre de cliente
  myMessage.setInt("error",1304);
}

task.complete(tkiid, (String)faultNames.get(0), myFault);
```

Esta acción establece un mensaje de error que contiene el código de error. La tarea se coloca en el estado de finalizada.

Conceptos relacionados

 Diagrama de transición de estado de tareas de colaboración
Las tareas de colaboración dan soporte a personas cuando trabajan para otras personas. Durante el ciclo de vida de una tarea de colaboración, determinadas interacciones sólo son posibles en determinados estados de tarea, y estas interacciones, a su vez, influyen en el estado de la tarea.

Suspensión y reanudación de instancias de tarea

Puede suspender las instancias de tareas de colaboración (también conocidas como *tareas de usuario* en la API) o instancias de tareas a realizar (también conocidas como *tareas participativas* en la API).

Antes de empezar

La instancia de tarea puede estar en el estado de preparada o reclamada. Se puede escalar. El llamante debe ser el propietario, el originador o el administrador de la instancia de tarea.

Acerca de esta tarea

Puede suspender una instancia de tarea cuando está en ejecución. Podría querer hacerlo, por ejemplo, de modo que pueda recabar información que es necesaria para completar la tarea. Cuando esté disponible la información, puede reanudar la instancia de tarea.

Procedimiento

1. Obtenga una lista de tareas reclamadas por el usuario conectado.

   ```
   QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                      "TASK.STATE = TASK.STATE.STATE_CLAIMED",
                                      (String)null,
                                      (Integer)null,
                                      (TimeZone)null);
   ```

   Esta acción devuelve un conjunto de resultados de consulta que contiene una lista de las tareas reclamadas por el usuario conectado.
2. Suspenda la instancia de tarea.

   ```
   if (result.size() > 0)
   {
     result.first();
     TKIID tkiid = (TKIID) result.getOID(1);
     task.suspend(tkiid);
   }
   ```

   Esta acción suspende la instancia de tarea especificada. Se pondrá la instancia de tarea en estado suspendido.
3. Reanude la instancia de proceso.

   ```
   task.resume( tkiid );
   ```

   Esta acción pone la instancia de tarea en el estado que tenía antes de que se suspendiera.

Análisis de los resultados de una tarea

Una tarea a realizar (también conocida como *tarea participativa* en la API) o una tarea de colaboración (también conocida como *tarea de usuario* en la API) se ejecuta asíncronamente. Si se especifica un manejador de respuestas cuando se inicia la tarea, se devuelve automáticamente el mensaje de salida cuando se completa la tarea. Si no se especifica un manejador de respuestas, el mensaje debe recuperarse explícitamente.

Acerca de esta tarea

Los resultados de la tarea sólo se almacenan en la base de datos si la plantilla de tarea de la que se ha derivado la instancia de tarea no especifica la supresión automática de las instancias de tarea derivadas.

Procedimiento

Analice los resultados de la tarea.
El ejemplo muestra cómo comprobar el número de pedido de una tarea completada satisfactoriamente.

```
QueryResultSet result = task.query("DISTINCT TASK.TKIID",
                                   "TASK.NAME = 'CustomerOrder' AND
                                    TASK.STATE = TASK.STATE.STATE_FINISHED",
                                    (String)null, (Integer)null, (TimeZone)null);
if (result.size() > 0)
{
  result.first();
  TKIID tkiid = (TKIID) result.getOID(1);
  ClientObjectWrapper output = task.getOutputMessage(tkiid);
  DataObject myOutput = null;
  if ( output.getObject() != null && output.getObject() instanceof DataObject)
  {
     myOutput  = (DataObject)output.getObject();
     int order = myOutput.getInt("OrderNo");
  }
}
```

Terminación de una instancia de tarea

A veces, es necesario que un usuario que tenga derechos de administrador de procesos, termine una instancia de tarea de la que se sabe que está en estado irrecuperable. Dado que la instancia de tarea se finaliza de forma inmediata, debe finalizar una instancia de tarea solamente en situaciones excepcionales.

Procedimiento

1. Recuperar la instancia de tarea que se ha de terminar.

   ```
   Task taskInstance = task.getTask(tkiid);
   ```

2. Terminar la instancia de tarea.

   ```
   TKIID tkiid = taskInstance.getID();
   task.terminate(tkiid);
   ```

   La instancia de tarea termina inmediatamente sin esperar a otras actividades pendientes.

Supresión de instancias de tarea

Las instancias de tareas sólo se suprimen automáticamente cuando finalizan si se ha especificado así en la plantilla de tarea asociada de donde se derivan dichas instancias. En este ejemplo se muestra cómo suprimir todas las instancias de tarea que han finalizado y no se han suprimido automáticamente.

Procedimiento

1. Liste las instancias de tareas que han finalizado.

   ```
   QueryResultSet result =
        task.query("DISTINCT TASK.TKIID",
                   "TASK.STATE = TASK.STATE.STATE_FINISHED",
                    (String)null, (Integer)null, (TimeZone)null);
   ```

   Esta acción devuelve un conjunto de resultados de consulta que lista las instancias de tarea finalizadas.

2. Suprima las instancias de tarea que hayan finalizado.

```
while (result.next() )
{
 TKIID tkiid = (TKIID) result.getOID(1);
 task.delete(tkiid);
}
```

Liberación de una tarea reclamada

Cuando un propietario potencial reclama una tarea, esta persona se encarga de completar la tarea. No obstante, a veces la tarea reclamada debe liberarse, de modo que otro propietario potencial pueda reclamarla.

Acerca de esta tarea

A veces, es necesario que un usuario que tenga derechos de administrador libere una tarea reclamada. Esto puede suceder, por ejemplo, cuando deba completarse una tarea pero el propietario de la tarea esté ausente. El propietario de la tarea también puede liberar una tarea reclamada.

Procedimiento

1. Liste las tareas reclamadas que son propiedad de una persona específica, por ejemplo, Smith.

   ```
   QueryResultSet result =
        task.query("DISTINCT TASK.TKIID",
                   "TASK.STATE = TASK.STATE.STATE_CLAIMED AND
                    TASK.OWNER = 'Smith'",
                    (String)null, (Integer)null, (TimeZone)null);
   ```

   Esta acción devuelve un conjunto de resultados de consulta que lista las tareas que ha reclamado la persona especificada, Smith.

2. Libere la tarea reclamada.

   ```
   if (result.size() > 0)
   {
     result.first();
     TKIID tkiid = (TKIID) result.getOID(1);
     task.cancelClaim(tkiid, true);
   }
   ```

   Esta acción devuelve la tarea a estado preparado para que uno de los demás propietarios potenciales pueda reclamarla. Se conservarán los datos de salida o de error establecidos por el propietario inicial.

Gestión de elementos de trabajo

Durante la vida de una instancia de actividad o una instancia de tarea, el conjunto de personas asociadas con el objeto puede cambiar, por ejemplo, porque una persona está de vacaciones, se han contratado otras personas o la carga de trabajo tiene que distribuirse de forma diferente. Para que puedan realizarse estos cambios, puede desarrollar aplicaciones para crear, suprimir o transferir elementos de trabajo.

Acerca de esta tarea

Un elemento de trabajo representa la asignación de un objeto a un usuario o a un grupo de usuarios para un motivo en particular. El objeto es habitualmente una instancia de actividad de tareas de usuario, una instancia de proceso o una instancia de tarea. Los motivos se derivan del rol que el usuario tenga para el objeto. Un objeto puede tener varios elementos de trabajo, porque un usuario puede tener distintos roles en asociación con el objeto, y se crea un elemento de

trabajo para cada uno de estos roles. Por ejemplo, una instancia de tarea a realizar puede tener un elemento de trabajo de administrador, lector, editor y propietario al mismo tiempo.

Las acciones que se pueden realizar para gestionar elementos de trabajo dependen del rol que tiene el usuario, por ejemplo, un administrador puede crear, suprimir y transferir elementos de trabajo, pero el propietario de tareas sólo puede transferir elementos de trabajo.

Procedimiento

- Crear un elemento de trabajo.

```
// consultar la instancia de tarea para la que se ha de especificar
// un administrador adicional
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='CustomerOrder'",
                                    (String)null, (Integer)null,
                                    (TimeZone)null);
if ( result.size() > 0 )
{
  result.first();
  // crear el elemento de trabajo
  task.createWorkItem((TKIID)(result.getOID(1)),
                        WorkItem.REASON_ADMINISTRATOR,"Smith");
}
```

  Esta acción crea un elemento de trabajo para el usuario Smith que tiene el rol de administrador.

- Eliminar un elemento de trabajo.

```
// consultar la instancia de tarea de la que se suprimirá un elemento de trabajo
QueryResultSet result = task.query("TASK.TKIID",
                                   "TASK.NAME='CustomerOrder'",
                                    (String)null, (Integer)null,
                                    (TimeZone)null);
if ( result.size() > 0 )
{
  result.first();
  // eliminar el elemento de trabajo
  task.deleteWorkItem((TKIID)(result.getOID(1)),
                        WorkItem.REASON_READER,"Smith");
}
```

  Esta acción suprime el elemento de trabajo para el usuario Smith que tiene el rol de lector.

- Transferir un elemento de trabajo.

```
// consultar la tarea que se ha de volver a planificar
QueryResultSet result =
     task.query("DISTINCT TASK.TKIID",
                "TASK.NAME='CustomerOrder' AND
                 TASK.STATE=TASK.STATE.STATE_READY AND
                 WORK_ITEM.REASON=WORK_ITEM.REASON.REASON_POTENTIAL_OWNER AND
                 WORK_ITEM.OWNER_ID='Miller'",
                 (String)null, (Integer)null, (TimeZone)null);
if ( result.size() > 0 )
{
  result.first();
  // transferir el elemento de trabajo del usuario Miller al usuario Smith,
  // para que Smith pueda trabajar en la tarea
  task.transferWorkItem((TKIID)(result.getOID(1)),
                         WorkItem.REASON_POTENTIAL_OWNER,"Miller","Smith");
}
```

  Esta acción transfiere el elemento de trabajo al usuario Smith para que pueda trabajar en el mismo.

Creación de plantillas de tarea e instancias de tarea durante la ejecución

Habitualmente debe utilizarse una herramienta de modelado como, por ejemplo, WebSphere Integration Developer para construir plantillas de tarea. A continuación, instale las plantillas de tarea en WebSphere Process Server y cree instancias a partir de estas plantillas, por ejemplo, mediante Business Process Choreographer Explorer. Sin embargo, también puede crear plantillas o instancias de tareas de usuario o participativas durante la ejecución.

Acerca de esta tarea

Es aconsejable realizar esta acción cuando, por ejemplo, la definición de tarea no está disponible al desplegar la aplicación, las tareas que forman parte de un flujo de trabajo no se conocen aún o se necesita una tarea para cubrir alguna colaboración ad-hoc entre un grupo de personas.

Puede modelar ad-hoc tareas a realizar o de colaboración creando instancias de la clase com.ibm.task.api.TaskModel y utilizándolas para crear una plantilla de tarea reutilizable o crear directamente una instancia de tarea de una sola ejecución. Para crear una instancia de la clase TaskModel, hay disponible un conjunto de métodos de fábrica en la clase de fábrica com.ibm.task.api.ClientTaskFactory. El modelado de tareas de usuario en tiempo de ejecución está basado en EMF (Eclipse Modeling Framework).

Procedimiento

1. Cree un org.eclipse.emf.ecore.resource.ResourceSet con el método de fábrica createResourceSet.
2. Opcional: Si tiene previsto utilizar tipos de mensaje complejos, puede definirlos con el org.eclipse.xsd.XSDFactory que puede obtener con el método de fábrica getXSDFactory(), o bien importar directamente un esquema XML con el método de fábrica loadXSDSchema.

   Para que los tipos complejos estén disponibles en WebSphere Process Server, despliéguelos como parte de una aplicación de empresa.
3. Cree o importe una definición WSDL (Web Services Definition Language) del tipo javax.wsdl.Definition.

   Puede crear una nueva definición de WSDL utilizando el método createWSDLDefinition. A continuación, puede añadirle un tipo de puerto y una operación. También puede importar directamente una definición WSDL existente mediante el método de fábrica loadWSDLDefinition.
4. Cree la definición de tarea mediante el método de fábrica createTTask.

   Si desea añadir o manipular más elementos de tarea compleja, utilice la clase com.ibm.wbit.tel.TaskFactory que puede recuperar mediante el método de fábrica getTaskFactory.
5. Cree el modelo de tarea mediante el método de fábrica createTaskModel y páselo al paquete de recursos creado en el paso 1, que agrega otros artefactos que se hayan creado mientras tanto.
6. Opcional: Valide el modelo mediante el método validate de TaskModel.

Resultados

Utilice uno de los métodos create de la API del EJB de Human Task Manager que tenga un parámetro **TaskModel** para crear una plantilla de tarea reutilizable, o bien una instancia de una sola ejecución.

Creación de tareas de tiempo de ejecución que utilizan tipos Java simples:

Este ejemplo crea una tarea de tiempo de ejecución que sólo utiliza tipos Java complejos en su interfaz, por ejemplo, un objeto Serie.

Acerca de esta tarea

El ejemplo sólo se ejecuta en el contexto de la aplicación de empresa que llama, para la que se cargan los recursos.

Procedimiento

1. Acceda a ClientTaskFactory y cree un conjunto de recursos que contenga las definiciones del nuevo modelo de tarea.

```
ClientTaskFactory factory = ClientTaskFactory.newInstance();
ResourceSet resourceSet = factory.createResourceSet();
```

2. Cree la definición WSDL y añada las descripciones de las operaciones.

```
// crear la interfaz WSDL
Definition definition = factory.createWSDLDefinition
         ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );

// crear un tipo de puerto
PortType portType = factory.createPortType( definition, "doItPT" );

// crear una operación; los mensajes de entrada y salida son de tipo Serie:
// no se especifica un mensaje de anomalía
Operation operation = factory.createOperation
        ( definition, portType, "doIt",
          new QName( "http://www.w3.org/2001/XMLSchema", "string" ),
          new QName( "http://www.w3.org/2001/XMLSchema", "string" ),
          (Map)null );
```

3. Cree el modelo EMF de la nueva tarea de usuario.

   Si crea una instancia de tarea, no se necesita una fecha de válido-desde (UTCDate).

```
TTask humanTask = factory.createTTask( resourceSet,
                                       TTaskKinds.HTASK_LITERAL,
                                       "TestTask",
                                       new UTCDate( "2005-01-01T00:00:00" ),
                                       "http://www.ibm.com/task/test/",
                                       portType,
                                       operation );
```

   Este paso inicializa las propiedades del modelo de tarea con valores por omisión.

4. Modifique las propiedades del modelo de tarea de usuario.

```
// utilizar los métodos del paquete com.ibm.wbit.tel, por ejemplo,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// recuperar la fábrica de tareas para crear o modificar elementos de
// tarea compuestos
TaskFactory taskFactory = factory.getTaskFactory();

// especificar valores de escalada
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// crear escalationReceiver y añadir verbo
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);
```

```
// crear escalada y añadir receptor de escalada
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Cree un modelo de tarea que contenga todas las definiciones de recurso

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

6. Valide el modelo de tarea y corrija los problemas de validación que se encuentren.

   ```
   ValidationProblem[] validationProblems = taskModel.validate();
   ```

7. Cree la instancia o plantilla de tarea de tiempo de ejecución.

   Utilice la interfaz HumanTaskManagerService para crear la instancia de tarea o la plantilla de tarea. Dado que la aplicación sólo utiliza tipos Java simples, no es necesario que especifique un nombre de aplicación.

   - El snippet siguiente crea una instancia de tarea:

     ```
     atask.createTask( taskModel, (String)null, "HTM" );
     ```

   - El snippet siguiente crea una plantilla de tarea:

     ```
     task.createTaskTemplate( taskModel, (String)null );
     ```

Resultados

Si se ha creado una instancia de tarea de tiempo de ejecución, ahora puede iniciarse. Si se ha creado una plantilla de tarea de tiempo de ejecución, ahora puede crear instancias de tarea a partir de la plantilla.

Creación de tareas de tiempo de ejecución que utilizan tipos complejos:

Este ejemplo crea una tarea de tiempo de ejecución que utiliza tipos complejos en su interfaz. Los tipos complejos ya están definidos, es decir, el sistema de archivos local en el cliente tiene archivos XSD que contienen la descripción de los tipos complejos.

Acerca de esta tarea

El ejemplo sólo se ejecuta en el contexto de la aplicación de empresa que llama, para la que se cargan los recursos.

Procedimiento

1. Acceda a ClientTaskFactory y cree un conjunto de recursos que contenga las definiciones del nuevo modelo de tarea.

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();
   ResourceSet resourceSet = factory.createResourceSet();
   ```

2. Añada las definiciones XSD de los tipos complejos al conjunto de recursos, de manera que estén disponibles cuando defina las operaciones.

   Los archivos están ubicados en una posición relativa a la del lugar donde se ejecuta el código.

   ```
   factory.loadXSDSchema( resourceSet, "InputBO.xsd" );
   factory.loadXSDSchema( resourceSet, "OutputBO.xsd" );
   ```

3. Cree la definición WSDL y añada las descripciones de las operaciones.

   ```
   // crear la interfaz WSDL
   Definition definition = factory.createWSDLDefinition
            ( resourceSet, new QName( "http://www.ibm.com/task/test/", "test" ) );

   // crear un tipo de puerto
   PortType portType = factory.createPortType( definition, "doItPT" );
   ```

```
// crear una operación; el mensaje de entrada es un InputBO y
// el mensaje de salida es un OutputBO;
// no se especifica un mensaje de anomalía
Operation operation = factory.createOperation
        ( definition, portType, "doIt",
          new QName( "http://Input", "InputBO" ),
          new QName( "http://Output", "OutputBO" ),
          (Map)null );
```

4. Cree el modelo EMF de la nueva tarea de usuario.

   Si crea una instancia de tarea, no se necesita una fecha de válido-desde (UTCDate).

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
                                          new UTCDate( "2005-01-01T00:00:00" ),
                                          "http://www.ibm.com/task/test/",
                                          portType,
                                          operation );
   ```

   Este paso inicializa las propiedades del modelo de tarea con valores por omisión.

5. Modifique las propiedades del modelo de tarea de usuario.

   ```
   // utilizar los métodos del paquete com.ibm.wbit.tel, por ejemplo,
   humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

   // recuperar la fábrica de tareas para crear o modificar elementos de
   // tarea compuestos
   TaskFactory taskFactory = factory.getTaskFactory();

   // especificar valores de escalada
   TVerb verb = taskFactory.createTVerb();
   verb.setName("John");

   // crear escalationReceiver y añadir verbo
   TEscalationReceiver escalationReceiver =
                       taskFactory.createTEscalationReceiver();
   escalationReceiver.setVerb(verb);

   // crear escalada y añadir receptor de escalada
   TEscalation escalation = taskFactory.createTEscalation();
   escalation.setEscalationReceiver(escalationReceiver);
   ```

6. Cree un modelo de tarea que contenga todas las definiciones de recurso

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

7. Valide el modelo de tarea y corrija los problemas de validación que se encuentren.

   ```
   ValidationProblem[] validationProblems = taskModel.validate();
   ```

8. Cree la instancia o plantilla de tarea de tiempo de ejecución.

   Utilice la interfaz HumanTaskManagerService para crear la instancia de tarea o la plantilla de tarea. Debe proporcionar un nombre de aplicación que contenga las definiciones de tipo de datos para que se pueda acceder a ellas. La aplicación debe contener también una tarea o proceso ficticio para que Business Process Choreographer cargue la aplicación.

   - El snippet siguiente crea una instancia de tarea:

     ```
     task.createTask( taskModel, "BOapplication", "HTM" );
     ```

   - El snippet siguiente crea una plantilla de tarea:

     ```
     task.createTaskTemplate( taskModel, "BOapplication" );
     ```

Resultados

Si se ha creado una instancia de tarea de tiempo de ejecución, ahora puede iniciarse. Si se ha creado una plantilla de tarea de tiempo de ejecución, ahora puede crear instancias de tarea a partir de la plantilla.

Creación de tareas de tiempo de ejecución que utilizan una interfaz existente:

Este ejemplo crea una tarea de tiempo de ejecución que utiliza una interfaz que ya está definida, es decir, el sistema de archivos local en el cliente tiene un archivo que contiene la descripción de la interfaz.

Acerca de esta tarea

El ejemplo sólo se ejecuta en el contexto de la aplicación de empresa que llama, para la que se cargan los recursos.

Procedimiento

1. Acceda a ClientTaskFactory y cree un conjunto de recursos que contenga las definiciones del nuevo modelo de tarea.

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();
   ResourceSet resourceSet = factory.createResourceSet();
   ```

2. Acceda a la definición de WSDL y a las descripciones de las operaciones.

   La descripción de la interfaz está ubicada en una posición relativa a la del lugar donde se ejecuta el código.

   ```
   Definition definition = factory.loadWSDLDefinition(
                             resourceSet, "interface.wsdl"  );
   PortType portType = definition.getPortType(
                        new QName( definition.getTargetNamespace(), "doItPT" ) );
   Operation operation = portType.getOperation
                               ("doIt", (String)null, (String)null);
   ```

3. Cree el modelo EMF de la nueva tarea de usuario.

   Si crea una instancia de tarea, no se necesita una fecha de válido-desde (UTCDate).

   ```
   TTask humanTask = factory.createTTask( resourceSet,
                                          TTaskKinds.HTASK_LITERAL,
                                          "TestTask",
                                          new UTCDate( "2005-01-01T00:00:00" ),
                                          "http://www.ibm.com/task/test/",
                                          portType,
                                          operation );
   ```

   Este paso inicializa las propiedades del modelo de tarea con valores por omisión.

4. Modifique las propiedades del modelo de tarea de usuario.

   ```
   // utilizar los métodos del paquete com.ibm.wbit.tel, por ejemplo,
   humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

   // recuperar la fábrica de tareas para crear o modificar elementos de
   // tarea compuestos
   TaskFactory taskFactory = factory.getTaskFactory();

   // especificar valores de escalada
   TVerb verb = taskFactory.createTVerb();
   verb.setName("John");

   // crear escalationReceiver y añadir verbo
   TEscalationReceiver escalationReceiver =
                       taskFactory.createTEscalationReceiver();
   ```

```
escalationReceiver.setVerb(verb);

// crear escalada y añadir receptor de escalada
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Cree un modelo de tarea que contenga todas las definiciones de recurso

   ```
   TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
   ```

6. Valide el modelo de tarea y corrija los problemas de validación que se encuentren.

   ```
   ValidationProblem[] validationProblems = taskModel.validate();
   ```

7. Cree la instancia o plantilla de tarea de tiempo de ejecución.

   Utilice la interfaz HumanTaskManagerService para crear la instancia de tarea o la plantilla de tarea. Debe proporcionar un nombre de aplicación que contenga las definiciones de tipo de datos para que se pueda acceder a ellas. La aplicación debe contener también una tarea o proceso ficticio para que Business Process Choreographer cargue la aplicación.

   - El snippet siguiente crea una instancia de tarea:

     ```
     task.createTask( taskModel, "BOapplication", "HTM" );
     ```

   - El snippet siguiente crea una plantilla de tarea:

     ```
     task.createTaskTemplate( taskModel, "BOapplication" );
     ```

Resultados

Si se ha creado una instancia de tarea de tiempo de ejecución, ahora puede iniciarse. Si se ha creado una plantilla de tarea de tiempo de ejecución, ahora puede crear instancias de tarea a partir de la plantilla.

Creación de tareas de tiempo de ejecución que utilizan una interfaz desde la aplicación que llama:

Este ejemplo crea una tarea de tiempo de ejecución que utiliza una interfaz que forma parte de la aplicación que llama. Por ejemplo, la tarea de ejecución se crea en un snippet Java de un proceso de empresa y utiliza una interfaz de la aplicación de proceso.

Acerca de esta tarea

El ejemplo sólo se ejecuta en el contexto de la aplicación de empresa que llama, para la que se cargan los recursos.

Procedimiento

1. Acceda a ClientTaskFactory y cree un conjunto de recursos que contenga las definiciones del nuevo modelo de tarea.

   ```
   ClientTaskFactory factory = ClientTaskFactory.newInstance();

   // especificar el cargador de clase de contexto para que se encuentren los
   // siguientes recursos
   ResourceSet resourceSet = factory.createResourceSet
                ( Thread.currentThread().getContextClassLoader() );
   ```

2. Acceda a la definición de WSDL y a las descripciones de las operaciones.

   Especifique la vía de acceso del archivo JAR de contenedor.

   ```
   Definition definition = factory.loadWSDLDefinition( resourceSet,
                             "com/ibm/workflow/metaflow/interface.wsdl" );
                        PortType portType = definition.getPortType(
   ```

```
                        new QName( definition.getTargetNamespace(), "doItPT" ) );
Operation operation = portType.getOperation
                                    ("doIt", (String)null, (String)null);
```

3. Cree el modelo EMF de la nueva tarea de usuario.

   Si crea una instancia de tarea, no se necesita una fecha de válido-desde (UTCDate).

```
TTask humanTask = factory.createTTask( resourceSet,
                                       TTaskKinds.HTASK_LITERAL,
                                       "TestTask",
                                       new UTCDate( "2005-01-01T00:00:00" ),
                                       "http://www.ibm.com/task/test/",
                                       portType,
                                       operation );
```

   Este paso inicializa las propiedades del modelo de tarea con valores por omisión.

4. Modifique las propiedades del modelo de tarea de usuario.

```
// utilizar los métodos del paquete com.ibm.wbit.tel, por ejemplo,
humanTask.setBusinessRelevance( TBoolean, YES_LITERAL );

// recuperar la fábrica de tareas para crear o modificar elementos de
// tarea compuestos
TaskFactory taskFactory = factory.getTaskFactory();

// especificar valores de escalada
TVerb verb = taskFactory.createTVerb();
verb.setName("John");

// crear escalationReceiver y añadir verbo
TEscalationReceiver escalationReceiver =
                    taskFactory.createTEscalationReceiver();
escalationReceiver.setVerb(verb);

// crear escalada y añadir receptor de escalada
TEscalation escalation = taskFactory.createTEscalation();
escalation.setEscalationReceiver(escalationReceiver);
```

5. Cree un modelo de tarea que contenga todas las definiciones de recurso

```
TaskModel taskModel = ClientTaskFactory.createTaskModel( resourceSet );
```

6. Valide el modelo de tarea y corrija los problemas de validación que se encuentren.

```
ValidationProblem[] validationProblems = taskModel.validate();
```

7. Cree la instancia o plantilla de tarea de tiempo de ejecución.

   Utilice la interfaz HumanTaskManagerService para crear la instancia de tarea o la plantilla de tarea. Debe proporcionar un nombre de aplicación que contenga las definiciones de tipo de datos para que se pueda acceder a ellas.

   - El snippet siguiente crea una instancia de tarea:

     ```
     task.createTask( taskModel, "WorkflowApplication", "HTM" );
     ```

   - El snippet siguiente crea una plantilla de tarea:

     ```
     task.createTaskTemplate( taskModel, "WorkflowApplication" );
     ```

Resultados

Si se ha creado una instancia de tarea de tiempo de ejecución, ahora puede iniciarse. Si se ha creado una plantilla de tarea de tiempo de ejecución, ahora puede crear instancias de tarea a partir de la plantilla.

Interfaz HumanTaskManagerService

La interfaz HumanTaskManagerService expone funciones relativas a tareas que un cliente local o remoto puede llamar.

Los métodos que pueden llamarse dependen del estado de la tarea y la autorización de la persona que utiliza la aplicación que contiene el método. Los métodos principales para manejar objetos de tarea se listan aquí. Para obtener más información sobre estos y otros métodos que están disponibles en la interfaz HumanTaskManagerService, consulte el Javadoc que se encuentra en el paquete com.ibm.task.api.

Plantillas de tarea

Para trabajar con plantillas de tareas dispone de los métodos siguientes.

Tabla 65. Métodos API para plantillas de tareas.

Método	Descripción
getTaskTemplate	Recupera la plantilla de tarea especificada.
createTask	Crea una instancia de tarea a partir de la plantilla de tarea especificada.
createAndCallTask	Crea y ejecuta una instancia de tarea a partir de la plantilla de tarea especificada y espere el resultado de manera síncrona.
createAndStartTask	Crea e inicia una instancia de tarea a partir de la plantilla de tarea especificada.
createAndStartTaskAsSubtask	Crea e inicia una instancia de tarea como subtarea de la tarea especificada.
createInputMessage	Crea un mensaje de entrada para la plantilla de tarea especificada. Por ejemplo, cree un mensaje que pueda utilizarse para iniciar una tarea.
queryTaskTemplates	Recupera plantillas de tarea que se almacenan en la base de datos.

Instancias de tareas

Para trabajar con instancias de tarea se dispone de los métodos siguientes.

Tabla 66. Métodos API para instancias de tareas.

Método	Descripción
getTask	Recupera una instancia de tarea; la instancia de tarea puede estar en cualquier estado.
callTask	Inicia una tarea de invocación de forma síncrona.
startTask	Inicia una tarea que ya se ha creado.
startTaskAsSubtask	Inicia una tarea como subtarea de la instancia de tarea.
suspend	Suspende la tarea a realizar o de colaboración.
resume	Reanuda la tarea a realizar o de colaboración.

Tabla 66. Métodos API para instancias de tareas. (continuación)

Método	Descripción
restart	Reinicia la instancia de tarea.
terminate	Termina la instancia de tarea especificada. Si se termina una tarea de invocación, esta acción no tiene ningún impacto en el servicio invocado.
delete	Suprime la instancia de tarea especificada.
claim	Reclama la tarea para el proceso.
actualización	Actualiza la instancia de tarea.
complete	Completa la instancia de tarea.
completeWithFollowOnTask	Completa la instancia de tarea e inicia una tarea de continuación.
cancelClaim	Libera una instancia de tarea reclamada de manera que pueda trabajar con ella otro propietario potencial.
createWorkItem	Crea un elemento de trabajo para la instancia de tarea.
transferWorkItem	Transfiere el elemento de trabajo a un propietario especificado.
deleteWorkItem	Suprime el elemento de trabajo.

Escaladas

Para trabajar con escaladas se dispone de los métodos siguientes.

Tabla 67. Métodos API para trabajar con escaladas

Método	Descripción
getEscalation	Recupera la instancia de escalada especificada.
triggerEscalation	Desencadena manualmente una escalada.

Propiedades personalizadas

Todas las tareas, plantillas de tareas y escaladas tienen sus propiedades personalizadas. La interfaz proporciona un método get y un método set para recuperar y establecer los valores para las propiedades personalizadas. También puede asociar las propiedades con nombre con, y recuperar propiedades con nombre de instancias de tarea. Los nombres y valores de propiedad personalizados deben ser del tipo java.lang.String. Los métodos siguientes son válidos para tareas, plantillas de tareas y escaladas.

Tabla 68. Métodos API para variables y propiedades personalizadas

Método	Descripción
getCustomProperty	Recupera la propiedad personalizada indicada de la instancia de tarea especificada.
getCustomProperties	Recupera las propiedades personalizadas para la instancia de tarea especificada.

Tabla 68. Métodos API para variables y propiedades personalizadas (continuación)

Método	Descripción
getCustomPropertyNames	Recupera los nombres de las propiedades personalizadas de la instancia de tarea.
setCustomProperty	Almacena valores específicos personalizados para la instancia de tarea especificada.

Desarrollo de aplicaciones para procesos empresariales y tareas de usuario

Las personas implicadas en la mayoría de los escenarios de procesos empresariales. Por ejemplo, un proceso empresarial requiere interacción con personas cuando se inicia o administra el proceso, o bien cuando se efectúan actividades de tareas de usuario. Para dar soporte a estos escenarios, debe utilizar la API de Business Flow Manager y la API del Gestor de tareas de usuario.

Acerca de esta tarea

Para implicar personas en escenarios de procesos empresariales, puede incluir los siguientes tipos de tareas en el proceso empresarial:

- Una tarea de invocación en línea (también conocida como *tarea de origen* en la API).

  Puede proporcionar una tarea de invocación para cada actividad de recepción, para cada elemento onMessage de una actividad de selección y para cada elemento onEvent de un manejador de sucesos. A continuación, esta tarea controla quién está autorizado para iniciar un proceso o comunicarse con una instancia de proceso en ejecución.
- Una tarea de administración.

  Puede proporcionar una tarea de administración para especificar quién está autorizado para administrar el proceso o para realizar operaciones administrativas sobre las actividades del proceso que contengan errores.
- Una tarea a realizar (también conocida como *tarea de particiáción* en la API).

  Una tarea a realizar implementa una actividad de tarea de usuario. Este tipo de actividad permite implicar a personas en el proceso.

Las actividades de tareas de usuario del proceso empresarial representan las tareas a realizar que la gente lleva a cabo en el escenario de proceso empresarial. Puede utilizar la API de Business Flow Manager y la API del Gestor de tareas de usuario para realizar estos escenarios:

- El proceso empresarial es el contenedor para todas las actividades que pertenecen al proceso, incluidas las actividades de tareas de usuario que se representan mediante tareas a realizar. Cuando se crea una instancia de proceso, se le asigna un ID de objeto (PIID) exclusivo.
- Cuando se activa una actividad de tarea de usuario durante la ejecución de la instancia de proceso, se crea una instancia de actividad que se identifica mediante su ID de objeto (AIID) exclusivo. Al mismo tiempo, también se crea una instancia de tarea a realizar en línea que se identifica mediante su ID de objeto (TKIID). La relación de la actividad de tarea de usuario con la instancia de tarea se lleva a cabo mediante los ID de objeto:
  - El ID de tarea a realizar de la instancia de actividad se establece en el TKIID de la tarea a realizar asociada.

- El ID de contexto de contenedor de la instancia de tarea se establece en el PIID de la instancia de proceso que contiene la instancia de actividad asociada.
- El ID de contexto padre de la instancia de tarea se establece en el AIID de la instancia de actividad asociada.

- La instancia de proceso gestiona los ciclos de vida de todas las instancias de tareas a realizar en línea. Cuando se suprime la instancia de proceso, también se suprimen las instancias de tarea. Por ejemplo, todas las tareas que tengan el ID de contexto de contenedor establecido en el PIID de la instancia de proceso se suprimen automáticamente.

Determinación de las plantillas o actividades de proceso que se pueden iniciar

Un proceso empresarial puede iniciarse invocando los métodos call, initiate, o sendMessage de la API de Business Flow Manager. Si el proceso sólo tiene una actividad de inicio, puede utilizar la firma del método que requiere un nombre de plantilla de proceso como parámetro. Si el proceso tiene más de una actividad de inicio, debe identificar explícitamente la actividad de inicio.

Acerca de esta tarea

Cuando se crea un modelo de un proceso empresarial, el creador del modelo puede determinar que sólo un subconjunto de usuarios puede crear una instancia de proceso desde la plantilla de proceso. Para realizar esto se asocia una tarea de invocación en línea a una actividad de inicio del proceso y se especifican restricciones de autorización de esa tarea. Sólo a las personas que son iniciadores potenciales o administradores de la tarea se les permite crear una instancia de la tarea y, por lo tanto, una instancia de la plantilla de proceso.

Si no se asocia una tarea de invocación en línea a la actividad de inicio o si no se especifican restricciones de autorización para la tarea, todos los usuarios pueden crear una instancia de proceso utilizando la actividad de inicio.

Un proceso puede tener más de una actividad de inicio, cada una de ellas con diferentes consultas de personas sobre administradores o iniciadores potenciales. Esto significa que un usuario puede estar autorizado para iniciar un proceso utilizando la actividad A pero no con la actividad B.

Procedimiento

1. Utilice la API de Business Flow Manager para crear una lista de las versiones actuales de plantillas de proceso que están en el estado iniciado.

   Consejo: El método queryProcessTemplates excluye sólo las plantillas de proceso que son parte de aplicaciones que aún no se han iniciado. De modo que, si utiliza este método sin filtrar los resultados, este último devuelve todas las versiones de las plantillas de proceso independientemente del estado en que estén.

   ```
   // indicación de fecha y hora actual en formato UTC, convertido a aaaa-mm-ddThh:mm:ss
   String now = (new UTCDate()).toXsdString();
   String whereClause = "PROCESS_TEMPLATE.STATE =
                         PROCESS_TEMPLATE.STATE.STATE_STARTED AND
                         PROCESS_TEMPLATE.VALID_FROM =
                         (SELECT MAX(VALID_FROM) FROM PROCESS_TEMPLATE
                          WHERE NAME=PROCESS_TEMPLATE.NAME AND
                          VALID_FROM <= TS('" + now + "'))";

   ProcessTemplateData[] processTemplates = process.queryProcessTemplates
   ```

```
                                        ( whereClause,
                                          "PROCESS_TEMPLATE.NAME",
                                          (Integer)null, (TimeZone)null);
```

Los resultados se clasifican por nombre de plantilla de proceso.

2. Cree la lista de plantillas de proceso y la de actividades de inicio para las que el usuario tiene autorización.

   La lista de plantillas de proceso contiene las plantillas de proceso que tienen una sola actividad de inicio. Estas actividades no están protegidas o el usuario conectado puede iniciarlas. De forma alternativa, quizá prefiera reunir las plantillas de proceso que al menos una de las actividades de inicio pueda iniciarlas.

   Consejo: Un administrador de proceso también puede iniciar una instancia de proceso. Sin embargo, si Business Flow Manager está utilizando la modalidad alternativa de autorización de administración de proceso, que restringe la administración del proceso a los administradores del sistema, sólo los usuarios del rol BPESystemAdministrator pueden realizar esta acción. Por lo tanto, para obtener una lista completa de plantillas, también deberá comprobar si el usuario que ha iniciado la sesión es un administrador.

   ```
   List authorizedProcessTemplates = new ArrayList();
   List authorizedActivityServiceTemplates = new ArrayList();
   ```

3. Determine las actividades de inicio para cada plantilla de proceso.

   ```
   for( int i=0; i<processTemplates.length; i++ )
   {
     ProcessTemplateData template = processTemplates[i];
     ActivityServiceTemplateData[] startActivities =
                    process.getStartActivities(template.getID());
   ```

4. Para cada actividad de inicio, recupere el ID de la plantilla de tarea de invocación en línea asociada.

   ```
   for( int j=0; j<startActivities.length; j++ )
   {
     ActivityServiceTemplateData activity = startActivities[j];
     TKTID tktid = activity.getTaskTemplateID();
   ```

   a. Si no existe una plantilla de tarea de invocación, esta actividad de inicio no protege la plantilla de proceso.

      En este caso, todos los usuarios pueden crear una instancia de proceso utilizando esta actividad de inicio.

      ```
      boolean isAuthorized = false;
             if ( tktid == null )
          {
            isAuthorized = true;
            authorizedActivityServiceTemplates.add(activity);
          }
      ```

   b. Si existe una plantilla de tarea de invocación, utilice la API de Human Task Manager para comprobar la autorización del usuario conectado.

      En el ejemplo, el usuario conectado es Smith. El usuario conectado debe ser un iniciador potencial de la tarea de invocación o el administrador.

      ```
      if ( tktid != null )
          {
            isAuthorized =
              task.isUserInRole
                  (tkid, "Torres", WorkItem.REASON_POTENTIAL_STARTER) ||
              task.isUserInRole(tktid, "Smith", WorkItem.REASON_ADMINISTRATOR);

            if ( isAuthorized )
      ```

```
        {
          authorizedActivityServiceTemplates.add(activity);
        }
      }
```

Si el usuario tiene el rol especificado, o si los criterios de asignación de personas del rol no se han especificado, el método isUserInRole devuelve el valor `true`.

5. Compruebe si el proceso se puede iniciar solamente con el nombre de la plantilla de proceso.

```
if ( isAuthorized && startActivities.length == 1 )
    {
      authorizedProcessTemplates.add(template);
    }
```

6. Finalice los bucles.

```
  } // fin del bucle de todas las plantillas de servicio de actividad
}   // fin del bucle de todas las plantillas de proceso
```

Proceso del flujo de trabajo de una sola persona que incluye tareas de usuario

Algunos flujos de trabajo sólo los realiza un usuario, por ejemplo pedir libros de una librería en línea. Este ejemplo muestra cómo implementar un flujo de trabajo único de persona utilizando un flujo de página del lado del servidor. Se utiliza Business Flow Manager y las API de Human Task Manager para procesar el flujo de trabajo.

Acerca de esta tarea

También se hace referencia a un flujo de trabajo singular de persona como *flujo de página* o *flujo de pantalla*. Existen dos tipos de flujos de página:

- Los flujos de página del lado del cliente, donde la navegación entre las distintas páginas se realiza mediante la tecnología del lado del cliente como, por ejemplo, un formulario Lotus Forms de varias páginas.
- Los flujos de página del lado del servidor se realizan utilizando un proceso empresarial y un conjunto de tareas humanas que se han modelado de forma que las siguientes tareas se asignan a la misma persona.

Los flujos de página del lado del servidor son más potentes que los flujos de página del lado del cliente, pero consumen más recursos de servidor para procesarlos. Por lo tanto, considere utilizar el tipo de flujo de trabajo básicamente en las siguientes situaciones:

- Debe invocar servicios entre los pasos llevados a cabo en una interfaz de usuario, por ejemplo, para recuperar o actualizar datos.
- Tiene requisitos de auditoría que requieren que los sucesos CEI se escriban después de que se complete una interacción de interfaz de usuario.

En una librería en línea, el comprador completa una secuencia de acciones para pedir un libro. Esta secuencia de acciones se puede implementar como una serie de actividades de tareas de usuario (tareas a realizar). Si el comprador decide pedir varios libros esto es equivalente a reclamar la siguiente actividad de tarea de usuario. Business Flow Manager mantiene la información sobre la secuencia de tareas, mientras que Human Task Manager mantiene las tareas en sí.

Compare este ejemplo con el que utiliza solamente la API de Business Flow Manager.

Procedimiento

1. Utilice la API de Business Flow Manager para obtener la instancia de proceso en la que desea trabajar.

   En este ejemplo, una instancia del proceso CustomerOrder.

   ```
   ProcessInstanceData processInstance =
      process.getProcessInstance("CustomerOrder");
   String piid = processInstance.getID().toString();
   ```

2. Utilice la API de Human Task Manager para consultar las tareas a realizar preparadas (kind participating, tipo participación) que son parte de la instancia de proceso especificada.

   Utilice el ID de contexto de contención de la tarea para especificar la instancia de proceso que la contiene. Para un flujo de trabajo de una sola persona, la consulta devuelve la tarea a realizar asociada a la primera actividad de tarea de usuario en la secuencia de actividades de tareas de usuario.

   ```
   //
    //Consultar la lista de tareas a realizar que el usuario conectado puede reclamar
    // para la instancia de proceso especificada
   //
   QueryResultSet result =
        task.query("DISTINCT TASK.TKIID",
                   "TASK.CONTAINMENT_CTX_ID = ID('" + piid + "') AND
                    TASK.STATE = TASK.STATE.STATE_READY AND
                    TASK.KIND = TASK.KIND.KIND_PARTICIPATING AND
                    WORK_ITEM.REASON = WORK_ITEM.REASON.REASON_POTENTIAL_OWNER",
                   (String)null, (Integer)null, (TimeZone)null);
   ```

3. Reclame la tarea a realizar que se devuelve.

   ```
   if (result.size() > 0)
   {
    result.first();
    TKIID tkiid = (TKIID) result.getOID(1);
    ClientObjectWrapper input = task.claim(tkiid);
    DataObject activityInput = null ;
     if ( input.getObject()!= null && input.getObject() instanceof DataObject )
     {
       taskInput = (DataObject)input.getObject();
       // leer los valores
       ...
     }
   }
   ```

   Cuando se reclama la tarea, se devuelve el mensaje de entrada de la tarea.

4. Determine la actividad de tarea de usuario asociada a la tarea a realizar.

   Puede utilizar uno de estos métodos para correlacionar las actividades con sus tareas.

   - El método task.getActivityID:

     ```
     AIID aiid = task.getActivityID(tkiid);
     ```

   - El ID de contexto padre es parte del objeto de tarea:

     ```
     AIID aiid = null;
     Task taskInstance = task.getTask(tkiid);

     OID oid = taskInstance.getParentContextID();
     if ( oid != null and oid instanceof AIID )
     {
       aiid = (AIID)oid;
     }
     ```

5. Cuando haya finalizado el trabajo de la tarea, utilice la API de Business Flow Manager para completar la tarea y su actividad de tarea de usuario asociada, y reclame la próxima actividad de tarea de usuario de la instancia de proceso.

Para completar la actividad de tarea de usuario, se pasa un mensaje de salida. Cuando cree el mensaje de salida, debe especificar el nombre del tipo de mensaje, para que contenga la definición del mensaje.

```
ActivityInstanceData activity = process.getActivityInstance(aiid);
ClientObjectWrapper output =
      process.createMessage(aiid, activity.getOutputMessageTypeName());
DataObject myMessage = null ;
if ( output.getObject()!= null && output.getObject() instanceof DataObject )
{
  myMessage = (DataObject)output.getObject();
  //establecer las partes del mensaje, por ejemplo, un número de pedido
  myMessage.setInt("OrderNo", 4711);
}

//completar la actividad de tarea de usuario y su tarea a realizar asociada
// y reclamar la siguiente actividad de tarea de usuario
CompleteAndClaimSuccessorResult successor =
   process.completeAndClaimSuccessor(aiid, output);
```

Esta acción establece un mensaje de salida que contiene el número de pedido y reclama la siguiente actividad de la secuencia. Si se establece `AutoClaim` para las actividades de sucesor y hay varias vías de acceso que se pueden seguir, se reclaman todas las actividades de sucesor y se devuelve una actividad aleatoria como la actividad siguiente. Si no hay más actividades de sucesor que se puedan asignar a este usuario, se devuelve `Null`.

Si el proceso contiene vías de acceso paralelas que se pueden seguir y estas vías de acceso contienen actividades de tareas de usuario para las que el usuario conectado es un propietario potencial de más de una de estas actividades, se reclama automáticamente una actividad aleatoria y se devuelve como la actividad siguiente.

6. Trabaje en la siguiente tarea de usuario.

```
ClientObjectWrapper nextInput = successor.getInputMessage();
if ( nextInput.getObject()!=
               null && nextInput.getObject() instanceof DataObject )
{
  activityInput = (DataObject)input.getObject();
  // leer los valores
  ...
}

aiid = successor.getAIID();
```

7. Continúe con el paso 5 para completar la actividad de tarea de usuario y para recuperar la siguiente actividad de tarea de usuario.

Tareas relacionadas

"Proceso del flujo de trabajo de un solo usuario" en la página 414
Algunos flujos de trabajo sólo los realiza un usuario, por ejemplo pedir libros de una librería en línea. Este tipo de flujo de trabajo no tiene rutas paralelas. Las API initiateAndClaimFirst y completeAndClaimSuccessor soportan el proceso de este tipo de flujo de trabajo. Este ejemplo muestra la implementación de un único flujo de trabajo de persona utilizando un flujo de página del lado del cliente.

Manejo de excepciones y errores

Es posible que un proceso BPEL encuentre un error en puntos diferentes del proceso.

Acerca de esta tarea

Los errores BPEL (Business Process Execution Language) se originan a partir de:

- Invocaciones de servicios Web, errores WSDL (Web Services Description Language)
- Actividades de generación
- Los errores estándar BPEL que Business Process Choreographer reconoce

Existen mecanismos para gestionar estos errores. Utilice uno de estos mecanismos para gestionar los errores generados por una instancia de proceso:

- Ceda el control a los manejadores de errores correspondientes
- Compense el trabajo anterior del proceso
- Detenga el proceso y permita que alguien repare la situación (force-retry, force-complete)

Un proceso BPEL también devuelve errores al que invoca una operación proporcionada por el proceso. Puede diseñar el error en el proceso como una actividad de respuesta con un nombre de error y datos de error. Estos errores se devuelven al que invoca la API como excepciones comprobadas.

Si un proceso BPEL no maneja un error BPEL o si se produce una excepción de la API, se devuelve una excepción de tiempo de ejecución al que invoca a la API. Un ejemplo de una excepción de la API es cuando no existe el modelo de proceso del que se va a crear una instancia.

En las tareas siguientes se describe cómo manejar los errores y excepciones

Manejo de excepciones de la API EJB de Business Process Choreographer

Si un método de la interfaz BusinessFlowManagerService o HumanTaskManagerService no se completa correctamente, se genera una excepción que indica la causa del error. Puede manejar esta excepción específicamente para proporcionar alguna directriz al emisor de la llamada

Acerca de esta tarea

No obstante, en la práctica general se maneja únicamente un subconjunto de las excepciones específicamente y para el resto de las excepciones potencias se proporcionan directrices generales. Todas las excepciones específicas se heredan de una excepción ProcessException o TaskException genérica. Capture excepciones genéricas que finalicen con la sentencia `catch(ProcessException)` o `catch(TaskException)`. Esta sentencia le ayuda a asegurarse la compatibilidad con versiones posteriores de su programa de aplicación ya que tiene en cuenta todas las demás excepciones que se pueden producir.

Comprobación del error establecido para una actividad de tarea de usuario

Cuando se procesa una actividad de tarea de usuario, se puede completar correctamente. En este caso, puede pasar un mensaje de salida. Si la actividad de tarea de usuario no se completa correctamente, puede pasar un mensaje de error.

Acerca de esta tarea

Puede leer el mensaje de error para determinar la causa del error.

Procedimiento

1. Liste las actividades de tarea que están en estado erróneo o detenido.

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "(ACTIVITY.STATE = ACTIVITY.STATE.STATE_FAILED OR
                     ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED) AND
                     ACTIVITY.KIND=ACTIVITY.KIND.KIND_STAFF",
                     (String)null, (Integer)null, (TimeZone)null);
```

Esta acción devuelve un conjunto de resultados de la consulta que contiene las actividades erróneas o detenidas.

2. Leer el nombre del error.

```
if (result.size() > 0)
{
  result.first();
     AIID aiid = (AIID) result.getOID(1);
   ClientObjectWrapper faultMessage = process.getFaultMessage(aiid);
  DataObject fault = null ;
  if ( faultMessage.getObject() !=
           null && faultMessage.getObject() instanceof DataObject )
  {
    fault = (DataObject)faultMessage.getObject();
    Type type = fault.getType();
    String name = type.getName();
    String uri = type.getURI();
  }
}
```

Esto devuelve el nombre del error. También puede analizar la excepción no manejada para una actividad detenida en lugar de recuperar el nombre del error.

Comprobar si se ha producido un error para una actividad de invocación detenida

En un proceso bien diseñado, generalmente las excepciones y los errores los manejan los manejadores de errores. Puede recuperar la información acerca de la excepción o del error que se ha producido para una actividad de proceso desde la instancia de actividad.

Acerca de esta tarea

Si una actividad causa la aparición de un error, el tipo de error determina las acciones que puede emprender para reparar la actividad.

Procedimiento

1. Listar las actividades de tareas de usuario que están en estado detenido.

```
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "ACTIVITY.STATE = ACTIVITY.STATE.STATE_STOPPED AND
                    ACTIVITY.KIND=ACTIVITY.KIND.KIND_INVOKE",
                    (String)null, (Integer)null, (TimeZone)null);
```

Esta acción devuelve un conjunto de resultados de consulta que contiene actividades de invocación detenidas.

2. Leer el nombre del error.

```
if (result.size() > 0)
{
  result.first();
     AIID aiid = (AIID) result.getOID(1);
  ActivityInstanceData activity = process.getActivityInstance(aiid);

  ProcessException excp = activity.getUnhandledException();
  if ( excp instanceof ApplicationFaultException )
```

```
  {
   ApplicationFaultException fault = (ApplicationFaultException)excp;
   String faultName = fault.getFaultName();
  }
}
```

Comprobación de la excepción o del error no manejado que se ha producido para una instancia de proceso anómala.

En un proceso bien diseñado, generalmente las excepciones y los errores los maneja un manejador de errores. Si el proceso implementa una operación bidireccional, puede recuperar la información acerca de un error o de una excepción manejada a partir de la propiedad de nombre de error del objeto de la instancia de proceso. Para los errores, también puede recuperar el mensaje de error correspondiente utilizando la API getFaultMessage.

Acerca de esta tarea

Si falla una instancia de proceso debido a una excepción que no está manejada por ningún manejador de errores, puede recuperar la información acerca de la excepción no manejada desde el objeto de la instancia de proceso. Por el contrario, si un manejador de errores captura una excepción, entonces la información acerca del error no está disponible. No obstante, puede recuperar el nombre del error y el mensaje y devolverlo al emisor de la llamada utilizando una excepción FaultReplyException.

Procedimiento

1. Lista las instancias de proceso que están en estado anómalo.

```
QueryResultSet result =
     process.query("PROCESS_INSTANCE.PIID",
                   "PROCESS_INSTANCE.STATE =
                       PROCESS_INSTANCE.STATE.STATE_FAILED",
                    (String)null, (Integer)null, (TimeZone)null);
```

   Esta acción devuelve un conjunto de resultados de consulta que contienen las instancias de proceso anómalas.

2. Lea la información para la excepción no manejada.

```
if (result.size() > 0)
{
  result.first();
  PIID piid = (PIID) result.getOID(1);
  ProcessInstanceData pInstance = process.getProcessInstance(piid);

  ProcessException excp = pInstance.getUnhandledException();
  if ( excp instanceof RuntimeFaultException )
  {
   RuntimeFaultException xcp = (RuntimeFaultException)excp;
   Throwable cause = xcp.getRootCause();
  }
  else if ( excp instanceof StandardFaultException )
  {
   StandardFaultException xcp = (StandardFaultException)excp;
   String faultName = xcp.getFaultName();
  }
  else if ( excp instanceof ApplicationFaultException )
  {
   ApplicationFaultException xcp = (ApplicationFaultException)excp;
   String faultName = xcp.getFaultName();
  }
}
```

Resultados

Utilice esta información para buscar el nombre del error o la causa raíz del problema.

Desarrollo de aplicaciones cliente de API de servicios Web para procesos empresariales y tareas de usuario

Puede desarrollar aplicaciones cliente que acceden a las aplicaciones de proceso empresarial y las aplicaciones de tareas de usuario mediante las API de servicios Web de Business Process Choreographer. El proceso de desarrollo de aplicaciones cliente consta de varios pasos obligatorios y opcionales, incluyendo la generación de un proxy de servicio Web y la adición de políticas de seguridad y transacción a la aplicación cliente.

Acerca de esta tarea

A partir de la Versión 7, la API de servicios Web basada en JAX-WS sustituye a la API de servicios Web de Business Process Choreographer basada en JAX-RPC de la Versión 6 (que se publicó por primera vez en el release 6.0.2). La API de servicios Web de Business Process Choreographer basada en JAX-RPC quedará en desuso, de modo que se deberán implementar nuevas aplicaciones cliente de servicio Web utilizando la API basada en JAX-WS.

Nota: La API JMS (Java Message Service) de Business Process Choreographer sigue utilizando las definiciones de esquema WSDL y XML para la Versión 6.

Puede desarrollar aplicaciones cliente en cualquier entorno de cliente de servicios Web. Los pasos siguientes proporcionan una visión general de las acciones que necesita realizar para desarrollar una aplicación de este tipo.

Procedimiento

1. Decida qué API de servicios Web necesita utilizar su aplicación de cliente: la API de Business Flow Manager, la API de Human Task Manager o ambas.
2. Exporte los archivos necesarios del entorno WebSphere Process Server.
3. En el entorno de desarrollo de aplicaciones cliente, genere un *proxy de servicios Web* utilizando los artefactos exportados.
4. Desarrolle el código de la aplicación cliente.
5. Añada las políticas de seguridad o transacción necesarias a la aplicación cliente.

Componentes de servicio Web y secuencia de control

En aplicaciones de servicios Web, un número de componentes del lado del cliente y del lado del servidor participan en la secuencia de control que representa una solicitud de servicio Web y una respuesta.

A continuación se muestra una secuencia habitual de control.

1. En el cliente:
   a. Una aplicación cliente (proporcionada por el usuario) emite una petición de un servicio Web.
   b. Un proxy de servicio Web (también proporcionado por el usuario, pero que se puede generar automáticamente utilizando programas de utilidad del

lado de cliente) incluye la solicitud de servicio en un sobre de solicitud SOAP y reenvía la solicitud a un URL definido como punto final del servicio Web.

2. La red transmite la petición al punto final del servicio Web utilizando HTTP o HTTPS.
3. En el servidor:
   a. La API de servicios Web genérica recibe y decodifica la petición.
   b. La petición se maneja directamente mediante el componente Business Flow Manager o Human Task Manager o se dirige al proceso empresarial o tarea de usuario especificado.
   c. Los datos devueltos se incluyen en un sobre de respuesta SOAP.
4. La red transmite la respuesta al entorno del extremo del cliente utilizando HTTP o HTTPS.
5. De nuevo en el cliente:
   a. La infraestructura de desarrollo en el cliente desenvuelve el sobre de respuesta SOAP.
   b. El proxy de servicio Web extrae los datos de la respuesta SOAP y los pasa a la aplicación cliente.
   c. La aplicación cliente procesa los datos devueltos como corresponda.

Ejemplo

A continuación se proporciona una posible descripción de una aplicación cliente que accede a la API de servicios Web de Human Task Manager para procesar una tarea a realizar:

1. La aplicación cliente emite una llamada de servicio Web query a WebSphere Process Server solicitando una lista de tareas a realizar en la que un usuario debe trabajar.
2. WebSphere Process Server devuelve la lista de tareas a realizar.
3. Entonces la aplicación cliente emite una llamada de servicio Web claim para reclamar una de las tareas a realizar.
4. WebSphere Process Server devuelve el mensaje de entrada para la tarea.
5. La aplicación cliente emite una llamada de servicio Web a complete para completar la tarea con un mensaje de salida o de error.

Requisitos de API de servicio Web para procesos empresariales y tareas de usuario

Los procesos empresariales y las tareas de usuario desarrolladas con WebSphere Integration Developer para que se ejecuten en Business Process Choreographer deben ajustarse a normas específicas para que se pueda acceder a los mismos desde las API de servicios Web.

Los requisitos son:

- Las interfaces de los procesos empresariales y las tareas de usuario se deben definir utilizando el estilo "documento/literal con envoltura" definido en la API Java para la especificación de servicios Web basados en XML (JAX-WS 2.0). Éste es el estilo por omisión para todos los procesos empresariales y las tareas de usuario desarrollados con WebSphere Integration Developer.
- No utilice el atributo maxOccurs en los elementos de parámetro de las operaciones o asegúrese de que el valor de este atributo está establecido en el valor por omisión, `maxOccurs="1"`.

- Los mensajes de error mostrados por los procesos empresariales y las tareas de usuario para las operaciones de servicio Web deben constar de una parte de mensaje WSDL individual definida con un elemento de esquema XML. Por ejemplo:

  ```
  <wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
  ```

Información relacionada

Página de descargas de API Java para servicios Web basados en XML (JAX-WS 2.0)

¿Qué estilo de WSDL debe utilizar?

API de servicios web de Business Process Choreographer basadas en JAX-WS

A partir de la Versión 7, la API de servicios web basada en JAX-WS sustituye a la API de servicios web de Business Process Choreographer basada en JAX-RPC de la Versión 6 (que se publicó por primera vez en el release 6.0.2). Se proporcionan dos interfaces de servicios web de Business Process Choreographer, una para procesos empresariales y otra para tareas de usuario, cada una con sus propios artefactos de archivos y espacios de nombres de definición XML.

La tabla siguiente proporciona una visión general de los artefactos de archivo y espacios de nombres de definición XML para los servicios web basados en JAX-WS.

Tabla 69. Artefactos de archivo y espacios de nombres de definición XML para los servicios web basados en JAX-WS

Interfaz de servicios web de Business Process Choreographer	Artefacto de archivo de servicios web de JAX-WS	Espacio de nombres XML de servicios web de JAX-WS
Servicio Web de Business Flow Manager	BFMJAXWSService.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0/Binding
Interfaz de servicio web de Business Flow Manager	BFMJAXWSInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0
Tipos de datos de servicio web de Business Flow Manager	BFMDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/business-process/types/7.0
Servicio Web de devolución de llamada de Business Flow Manager	BFMJAXWSCallbackService.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/callback-services/7.0/Binding
Interfaz de servicio web de devolución de llamada de Business Flow Manager	BFMJAXWSCallbackInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/business-process/callback-services/7.0
Servicio Web de Human Task Manager	HTMJAXWSService.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/services/7.0/Binding
Interfaz de servicio web de Human Task Manager	HTMJAXWSInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/services/7.0
Tipos de datos de servicio web de Human Task Manager	HTMDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/human-task/types/7.0
Servicio Web de devolución de llamada de Human Task Manager	HTMJAXWSCallbackService.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/callback-services/7.0/Binding
Interfaz de servicio web de devolución de llamada de Human Task Manager	HTMJAXWSCallbackInterface.wsdl	http://www.ibm.com/xmlns/prod/websphere/human-task/callback-services/7.0
Tipos de datos comunes de Business Process Choreographer	BPCDataTypes.xsd	http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/7.0

API de servicios Web de Business Process Choreographer: Estándares

Utilice los enlaces siguientes para buscar información suplementaria pertinente acerca de los estándares que se aplican a las aplicaciones Web. La información reside en sitios de Internet que no son de IBM, cuyos patrocinadores controlan la precisión técnica de la información.

Estos enlaces se proporcionan por comodidad. Normalmente, la información no es específica de IBM WebSphere Process Server, pero es útil para conocer los servicios Web en general.

- Java API for XML-based Web Services (JAX-WS 2.0) (JSR-224; Java Community Process)
- Java Architecture for XML Binding (JAXB) 2.0 (JSR-222; Java Community Process)
- Web Services Description Language (WSDL) 1.1 (W3C)
- XML Schema Part 0: Primer Second Edition (W3C)
- XML Schema Part 1: Structures Second Edition (W3C)
- XML Schema Part 2: Datatypes Second Edition (W3C)
- Simple Object Access Protocol (SOAP) 1.1 (W3C)
- Web Services Policy Framework (WS-Policy) 1.5 (W3C)
- WS-Security 1.1 (OASIS)
- WS-Security UserName Token Profile 1.1 (OASIS)
- WS-AtomicTransaction 1.2 (OASIS)
- WS-Interoperability Basic Profile 1.1 (WS-Interoperability Organization)

Publicación y exportación de artefactos del entorno de servidor para aplicaciones cliente de servicios Web

Para poder desarrollar aplicaciones cliente para acceder a las API de servicios Web de Business Process Choreographer, debe publicar y exportar varios artefactos del entorno de servidor WebSphere.

Acerca de esta tarea

Los artefactos que se han de exportar son:

- Los archivos WSDL (Web Service Definition Language) que describen el punto final de servicio Web, los tipos de puerto y las operaciones que forman la API de servicios Web de Business Process Choreographer (siempre necesaria para la generación de proxy de servicio Web).
- Los archivos XSD (XML Schema Definition) que contienen definiciones de tipos de datos a los que hacen referencia los servicios de los archivos WSDL de Business Process Choreographer (siempre es necesario para la generación de proxy de servicio Web).
- Los archivos WSDL y XSD propios que describen interfaces y tipos de datos para los procesos empresariales o tareas de usuario que se ejecutan en el servidor WebSphere. Estos archivos adicionales sólo son necesarios si la aplicación cliente necesita interactuar directamente con los procesos empresariales o las tareas de usuario mediante las API de servicios Web. No son necesarios si la aplicación cliente sólo va a invocar operaciones que pueden ser llevadas a cabo por Business Process Choreographer sin interacción directa con las instancias de proceso o tarea, por ejemplo la emisión de consultas.

- Archivos Web Service Policy (WS-Policy) que describen la calidad de los atributos de servicio para la API de servicios Web. Éstos se puede exportar a fin de servir de base para crear políticas de servicio Web del lado del cliente.

  WS-Security
  El mensaje de solicitud debe contener una señal de nombre de usuario o una señal LPTA.

  WS-Transaction
  El mensaje de solicitud puede contener un contexto WS-AtomicTransaction. Si este contexto está presente, la solicitud se procesa en el ámbito de transacción del emisor de la llamada.

Después de publicar estos artefactos, necesita copiarlos en el entorno de programación de cliente, donde se utilizan para generar un proxy de servicio Web y clases de ayuda.

Publicación de archivos WSDL de Business Process Choreographer

Los archivos WSDL (Web Service Definition Language) contienen una descripción detallada de todas las operaciones disponibles con las API de servicios Web. Se dispone de varios archivos WSDL para las API de servicio Web de Business Flow Manager y Human Task Manager. Estos archivos se utilizan para generar un proxy de servicio Web para la aplicación.

Antes de empezar

Antes de publicar los archivos WSDL, asegúrese de especificar la dirección de punto final de los servicios Web correcta. Se trata del URL que la aplicación cliente utiliza para acceder a las API de servicios Web.

Acerca de esta tarea

Debe publicar estos archivos WSDL y cualquier archivo XSD al que se haga referencia en los archivos WSDL. A continuación, puede copiarlos del entorno de WebSphere en el entorno de desarrollo, donde se utilizan para generar un proxy de servicio Web y clases de ayuda. Sólo necesita publicar los archivos WSDL de Business Process Choreographer una vez.

Publicación del archivo WSDL de proceso empresarial para aplicaciones de servicio Web:

Utilice la consola administrativa para publicar el archivo WSDL

Procedimiento

1. Inicie la sesión en la consola administrativa con el ID de usuario con derechos de administrador.
2. Pulse **Aplicaciones → Módulos SCA**.

   Nota: También puede pulsar **Aplicaciones → Tipos de aplicación → WebSphere Enterprise Applications** para visualizar una lista de todas las aplicaciones empresariales disponibles.
3. Seleccione la aplicación **BPEContainer** de la lista de módulos SCA o aplicaciones.
4. Seleccione **Publicar archivos WSDL** de la lista de **Propiedades adicionales**
5. Pulse el archivo .zip en la lista.

6. En la ventana Descarga de archivos que se muestra, pulse **Guardar**.
7. Vaya a la carpeta local y pulse **Guardar**.

Resultados

El archivo .zip exportado se denomina `BPEContainer_nombreNodo_nombreServidor_WSDLFiles.zip`. El archivo .zip contiene un archivo WSDL que describe los servicios Web y los archivos XSD a los que hace referencia el archivo WSDL.

Nota: El archivo .zip exportado contiene artefactos WSDL y XSD del servicio Web de JAX-WS presentada en la Versión 7 y el servicio Web de JAX-RPC utilizado en la Versión 6. Al generar el proxy de servicio Web utilizando la herramienta wsimport, se seleccionan los artefactos de servicio Web de JAX-WS y se ignoran los artefactos de JAX-RPC.

Publicación del archivo WSDL de tarea de usuario para aplicaciones de servicios Web:

Utilice la consola administrativa para publicar el archivo WSDL

Procedimiento

1. Inicie la sesión en la consola administrativa con el ID de usuario con derechos de administrador.
2. Pulse **Aplicaciones → Módulos SCA**.

   Nota: También puede pulsar **Aplicaciones → Tipos de aplicación → WebSphere Enterprise Applications** para visualizar una lista de todas las aplicaciones empresariales disponibles.
3. Seleccione la aplicación **TaskContainer** de la lista de módulos SCA o aplicaciones.
4. Seleccione **Publicar archivos WSDL** de la lista de **Propiedades adicionales**
5. Pulse el archivo .zip en la lista.
6. En la ventana Descarga de archivos que se muestra, pulse **Guardar**.
7. Vaya a la carpeta local y pulse **Guardar**.

Resultados

El archivo .zip exportado se denomina `TaskContainer_nombreNodo_nombreServidor_WSDLFiles.zip`. El archivo .zip contiene un archivo WSDL que describe los servicios Web y los archivos XSD a los que hace referencia el archivo WSDL.

Nota: El archivo .zip exportado contiene artefactos WSDL y XSD del servicio Web de JAX-WS presentado en la Versión 7 y del servicio Web de JAX-RPC utilizado en la Versión 6. Al generar el proxy de servicio Web utilizando la herramienta wsimport, se seleccionan los artefactos de servicio Web de JAX-WS y se ignoran los artefactos de JAX-RPC.

Exportación de archivos WSDL y XSD para aplicaciones de servicios Web de tareas de usuario y procesos empresariales

Los procesos empresariales y las tareas de usuario tienen interfaces bien definidas que les permiten acceder externamente como servicios Web. Es necesario exportar

las definiciones de interfaz WSDL y las definiciones de tipo de datos de esquema XML al entorno de programación de cliente.

Acerca de esta tarea

Este procedimiento se debe repetir para cada proceso empresarial o tarea de usuario con los que la aplicación cliente necesite interactuar.

Por ejemplo, para crear e iniciar una tarea de usuario, se deben transferir los siguientes elementos de información a la interfaz de tareas:

- El nombre de la plantilla de la tarea
- El espacio de nombres de la plantilla de tarea.
- Un mensaje de entrada, que contiene los datos de empresa con formato.
- Una envoltura de respuesta para devolver el mensaje de respuesta.
- Un mensaje de error para devolver los errores y las excepciones.

Estos elementos se encapsulan dentro de un solo objeto de empresa. Todas las operaciones de la interfaz de servicio web se modelan como una operación de envoltorio documento/literal. Los parámetros de entrada y salida de estas operaciones se encapsulan en documentos de envoltorio. Otros objetos de empresa definen los formatos de respuesta y error correspondientes.

Para crear e iniciar el proceso empresarial o la tarea de usuario mediante un servicio Web, estos objetos de envoltorio deben estar disponibles para la aplicación de cliente en el extremo del cliente.

Esto se logra exportando los objetos de empresa del entorno de WebSphere como archivos WSDL (Web Service Definition Language) y XSD (XML Schema Definition) e importando las definiciones de tipo de datos al entorno de programación de cliente.

Procedimiento

1. Inicie el espacio de trabajo de WebSphere Integration Developer si aún no está en ejecución.
2. Seleccione el módulo de biblioteca que contiene los objetos de empresa que se van a exportar. Un módulo de biblioteca es un archivo comprimido que contiene los objetos de empresa necesarios.
3. Exporte el módulo de biblioteca.
4. Copie los archivos exportados al entorno de desarrollo de aplicaciones cliente.

Ejemplo

Supongamos que un proceso empresarial expone la siguiente operación de servicio Web:

```
<wsdl:operation name="updateCustomer">
    <wsdl:input message="tns:updateCustomerRequestMsg"
                name="updateCustomerRequest"/>
    <wsdl:output message="tns:updateCustomerResponseMsg"
                 name="updateCustomerResponse"/>
    <wsdl:fault message="tns:updateCustomerFaultMsg"
                name="updateCustomerFault"/>
  </wsdl:operation>
```

con los mensajes WSDL definidos como:

```
<wsdl:message name="updateCustomerRequestMsg">
    <wsdl:part element="types:updateCustomer"
               name="updateCustomerParameters"/>
  </wsdl:message>
  <wsdl:message name="updateCustomerResponseMsg">
    <wsdl:part element="types:updateCustomerResponse"
               name="updateCustomerResult"/>
  </wsdl:message>
  <wsdl:message name="updateCustomerFaultMsg">
    <wsdl:part element="types:updateCustomerFault"
               name="updateCustomerFault"/>
  </wsdl:message>
```

Los elementos *concretos* definidos por el cliente `types:updateCustomer`, `types:updateCustomerResponse` y `types:updateCustomerFault` se deben pasar a las API de servicios Web y recibir de las mismas utilizando los parámetros `UserData` en todas las operaciones *genéricas* (call, sendMessage, etc.) realizadas por la aplicación cliente.

Los elementos definidos por el cliente se crean, serializan y deserializan en el lado de la aplicación cliente utilizando clases generadas con los archivo XSD exportados. La generación de estas clases forma parte de la generación de proxy de servicio Web donde se incluyen los archivos WSDL y XSD exportados.

Las operaciones genéricas de la interfaz de servicio web propagan el elemento derivador del documento en la operación implementada por el proceso empresarial o tarea humana. Para la operación de ejemplo del ejemplo anterior, un mensaje SOAP de servicio web podría tener un aspecto parecido al siguiente:

```
<soapenv:Envelope xmlns:soapenv="..." ...>
  <soapenv:Header>
    ...
  </soapenv:Header>
  <soapenv:Body>
    <bfm:sendMessage
         xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/7.0">
      <processTemplateName>customerProcessTemplate</processTemplateName>
      <portType xmlns:cns="http://example.com/customerProcess">cns:customerProcessPortType</portType>
      <operation>updateCustomer</operation>
      <input>
        <cns:updateCustomer xmlns:cns="http://example.com/customerProcess">
          <street>1600 Pennsylvania Avenue Northwest</street>
          <city>Washington, DC 20006</city>
        </cns:updateCustomer>
      </input>
    </bfm:sendMessage>
  </soapenv:Body>
</soapenv:Envelope>
```

Desarrollo de aplicaciones cliente en el entorno de servicios Web Java

Puede utilizar cualquier entorno de desarrollo basado en Java compatible con los servicios Web Java para desarrollar aplicaciones de cliente para las API de servicios Web de Business Process Choreographer.

Generación de un proxy de servicio Web (servicios Web Java)

Las aplicaciones cliente de servicios Web Java utilizan un *proxy de servicio Web* para interactuar con las API de servicios Web de Business Process Choreographer.

Acerca de esta tarea

Un proxy de servicio Web para servicios Web Java contiene varias clases JavaBeans a las que la aplicación cliente llama para realizar solicitudes de servicio Web. El proxy de servicio Web maneja el ensamblaje de los parámetros de servicio en mensajes SOAP, envía mensajes SOAP al servicio Web a través de HTTP, recibe respuesta del servicio Web y pasa los datos devueltos a la aplicación cliente.

Por consiguiente, un proxy de servicio Web permite básicamente que una aplicación cliente llame a un servicio Web como si fuera una función local.

Nota: Sólo necesita generar un proxy de servicio Web una vez. Todas las aplicaciones cliente que acceden a la misma API de servicios Web pueden utilizar entonces el mismo proxy de servicio Web.

En el entorno de servicio de servicios Web de IBM, puede generar un proxy de servicio Web de una de las siguientes maneras.

- Utilice los entornos de desarrollo integrados de Rational Application Developer o WebSphere Integration Developer.
- Utilice la herramienta de línea de mandatos wsimport.

Otros entornos de desarrollo de servicios Web Java suelen incluir la herramienta wsimport o los recursos de generación de aplicación cliente de propietario.

Utilización de Rational Application Developer para generar un proxy de servicio Web para una aplicación de servicios Web:

Puede utilizar el entorno de desarrollo integrado de Rational Application Developer para generar un proxy de servicio Web para la aplicación cliente de servicios Web. La secuencia siguiente de pasos se aplica a Rational Application Developer Versión 7.5.3.

Antes de empezar

Antes de generar un proxy de servicio Web, tiene que haber exportado anteriormente los archivos WSDL y XSD que describen las interfaces de servicios Web de proceso empresarial o tarea de usuario del entorno WebSphere y haberlos copiado en el entorno de programación de cliente.

Procedimiento

1. Añada el archivo WSDL adecuado al proyecto:
   - Para procesos empresariales:
     a. Descomprima el archivo exportado `BPEContainer_nombre-nodo_nombre-servidor_WSDLFiles.zip` en un directorio temporal. No cambie el contenido de este directorio y tenga en cuenta que sólo se utilizan los siguientes archivos WSDL y XSD para la generación de proxy de servicio Web para las interacciones con procesos empresariales:
        – BFMJAXWSService.wsdl
        – BFMJAXWSInterface.wsdl
        – BFMJAXWSCallbackService.wsdl
        – BFMJAXWSCallbackInterface.wsdl
        – BFMDataTypes.xsd
        – BPCDataTypes.xsd

- wsa.xsd

b. Importe el subdirectorio `META-INF` del directorio descomprimido `BPEContainer_`*`nombre-nodo_nombre-servidor`*`.ear/bfmjaxws.jar`.

- Para tareas de usuario:

a. Descomprima el archivo exportado `TaskContainer_nombre-nodo_nombre-servidor_WSDLFiles.zip` en un directorio temporal. No cambie el contenido de este directorio y tenga en cuenta que sólo se utilizan los siguientes archivos WSDL y XSD para la generación de proxy de servicio Web para las interacciones con tareas de usuario:

- HTMJAXWSService.wsdl
- HTMJAXWSInterface.wsdl
- HTMJAXWSCallbackService.wsdl
- HTMJAXWSCallbackInterface.wsdl
- HTMDataTypes.xsd
- BPCDataTypes.xsd
- wsa.xsd

b. Importe el subdirectorio `META-INF` del directorio descomprimido `TaskContainer_`*`nombre-nodo_nombre-servidor`*`.ear/htmjaxws.jar`.

Se creará una nueva estructura de directorio y subdirectorio `wsdl` en el proyecto.

2. Seleccione el archivo `BFMJAXWSService.wsdl` ubicado en el directorio `wsdl` recién creado.
3. Pulse el botón derecho del ratón y seleccione **Servicios Web → Generar cliente**.

   Antes de continuar con los pasos restantes, asegúrese de que se ha iniciado el servidor.
4. En la ventana Servicios Web, pulse **Siguiente** para aceptar todos los valores por omisión.
5. En la ventana Configuración de cliente de servicio Web de JAX-WS de Servicio Web, cambie a 2.0 la versión del código JAX-WS que se debe generar y pulse **Finalizar** para aceptar todos los demás valores por omisión
6. Vuelva a realizar los pasos 2 a 5 de este procedimiento con `HTMJAXWSService.wsdl` y grabe encima de todos los archivos si se solicita que lo haga.

Resultados

Se generará y se añadirá al proyecto un proxy de servicio Web, compuesto por varias clases de proxy, ubicador y JAXB.

Utilización de la herramienta de línea de mandatos wsimport para generar un proxy de servicio Web para una aplicación de servicios Web:

Puede utilizar la herramienta de línea de mandatos wsimport para generar un proxy de servicio Web para una aplicación de servicios Web.

Antes de empezar

Antes de generar un proxy de servicio Web, debe haber exportado previamente los archivos WSDL que describen las API de servicios Web de procesos empresariales o tareas de usuario del entorno de WebSphere y haberlos copiado en el entorno de programación de cliente.

Procedimiento

1. Genere un proxy de servicio Web para la API de servicios Web de Business Process Choreographer:

   Nota: Para obtener una descripción detallada de la herramienta de línea de mandatos wsimport para aplicaciones JAX-WS, consulte la documentación de la herramienta de línea de mandatos wsimport de WebSphere Application Server.

   ```
   wsimport.bat BFMJAXWSService.wsdl myService1.wsdl myService2.wsdl
    –d proxy-bfm
    -wsdllocation <ubicación_bfm>

   wsimport.bat HTMJAXWSService.wsdl myService1.wsdl myService2.wsdl
    –d proxy-htm
    -wsdllocation <ubicación_htm>
   ```

   En este ejemplo, `myService1.wsdl` y `myService2.wsdl` contienen definiciones de interfaz de procesos empresariales personalizados y/o tareas de usuario. Además, *<ubicación_bfm>* y *<ubicación_htm>* se pueden obtener del elemento *<puerto>* de WSDL en `BFMJAXWSService.wsdl` y `HTMJAXWSService.wsdl`, respectivamente.

   Puede fusionar ambos proxies en un directorio común (por ejemplo proxy-bpc) y grabar encima de los archivos existentes si se le solicita que lo haga.
2. Incluya los archivos de clase generados en el proyecto.

Tareas relacionadas

"Creación de una aplicación cliente para procesos empresariales y tareas de usuario (servicios Web Java)"
Una aplicación cliente envía solicitudes a las API de servicios Web de Business Process Choreographer y recibe respuestas de las mismas. Mediante el uso de un proxy de servicio Web para gestionar las comunicaciones y las clases de ayuda para formatear tipos de datos complejos, una aplicación cliente puede invocar los métodos de servicio Web como si fueran funciones locales.

Creación de una aplicación cliente para procesos empresariales y tareas de usuario (servicios Web Java)

Una aplicación cliente envía solicitudes a las API de servicios Web de Business Process Choreographer y recibe respuestas de las mismas. Mediante el uso de un proxy de servicio Web para gestionar las comunicaciones y las clases de ayuda para formatear tipos de datos complejos, una aplicación cliente puede invocar los métodos de servicio Web como si fueran funciones locales.

Antes de empezar

Antes de empezar a crear una aplicación cliente, genere el proxy de servicio Web.

Acerca de esta tarea

Puede desarrollar aplicaciones cliente utilizando cualquier herramienta de desarrollo compatible con los servicios Web, por ejemplo IBM Rational Application Developer. Puede generar cualquier tipo de aplicación de servicios Web para llamar a las API de servicios Web.

Procedimiento

1. Cree un nuevo proyecto de aplicación cliente.
2. Genere el proxy de servicio Web.
3. Codifique la aplicación cliente.

4. Genere el proyecto.
5. Ejecute la aplicación cliente.

Ejemplo

El ejemplo siguiente muestra cómo se utiliza la API de servicio Web de Business Flow Manager.

```
try {
   // crear proxy bfm
   BFMJAXWSPortType bfm = new BFMJAXWSService().getBFMJAXWSPort();

   // llamar a getProcessTemplate
   ProcessTemplateType ptt =
     bfm.getProcessTemplate("MY_PROCESS_TEMPLATE_NAME");

   // manejar valor de retorno
   System.out.println("Process template '" + ptt.getName() +
       "' found, details following:");
   System.out.println("Execution mode: " +
       ptt.getExecutionMode());
   System.out.println("Schema version: " +
       ptt.getSchemaVersion());
 } catch (Exception e) {
   if ( e instanceof ProcessFaultMsg )
   {
     ProcessFaultMsg pfm = (ProcessFaultMsg) e;
     List<FaultStackType> list =
         ( pfm.getFaultInfo() ).getFaultStack();
     FaultStackType fault = list.get( 0 );
     System.out.println( "ProcessFaultMessage: " +
         fault.getMessage() );
   }
   else    {
     e.printStackTrace( System.out );
   }
 }
```

Tareas relacionadas

"Utilización de la herramienta de línea de mandatos wsimport para generar un proxy de servicio Web para una aplicación de servicios Web" en la página 464
Puede utilizar la herramienta de línea de mandatos wsimport para generar un proxy de servicio Web para una aplicación de servicios Web.

"Generación de un proxy de servicio Web (servicios Web Java)" en la página 462
Las aplicaciones cliente de servicios Web Java utilizan un *proxy de servicio Web* para interactuar con las API de servicios Web de Business Process Choreographer.

Adición de seguridad

El servicio web de Business Process Choreographer necesita que configure la aplicación cliente para un mecanismo de autenticación.

Acerca de esta tarea

Por omisión, Business Process Choreographer soporta los siguientes mecanismos de autenticación:

Señal de nombre de usuario
Un consumidor de servicio web proporciona una señal de nombre de usuario como medio de identificar el solicitante por el "nombre de usuario" y, opcionalmente, utilizando una contraseña para autenticar esa identidad en el proveedor de servicios web.

Señal de seguridad binaria – Señal LTPA (Lightweight Third-Party Authentication)
Un consumidor de servicio web proporciona una señal LTPA como medio de autenticación del solicitante en el proveedor de servicios web

Puede sustituir la política de seguridad de servicio web de Business Process Choreographer por un mecanismo de autenticación alternativo. Sin embargo, dado que no es posible invocar operaciones de servicio web de Business Process Choreographer como usuario no autenticado, se necesita siempre un mecanismo de autenticación.

Adición de soporte de transacción

Se pueden configurar aplicaciones cliente de servicio Web para permitir que el proceso de solicitudes del lado del servidor participe en la transacción del cliente, pasando un contexto de aplicación de cliente como parte de la solicitud de servicio. Este soporte de transacciones atómicas se define en la especificación WS-AT (Web Services-Atomic Transaction).

Acerca de esta tarea

Business Process Choreographer ejecuta cada solicitud de operación de servicio Web como una transacción global independiente. Las aplicaciones de cliente se pueden configurar para utilizar el soporte de transacciones de uno de los modos siguientes:

- Propague el contexto de transacción del cliente. El proceso de solicitudes del lado de servidor se realiza en el contexto de transacción de aplicación cliente y, por consiguiente, se confirma (o retrotrae) junto con la transacción del cliente. Y, a la inversa, si el servidor encuentra un problema mientras se ejecuta la operación de servicio Web y solicita una retrotracción, la transacción de la aplicación cliente también se retrotrae.
- No utilice el soporte de transacciones. Business Process Choreographer crea una nueva transacción global en la que se debe ejecutar la solicitud, pero el proceso de solicitud del lado del servidor no se realiza con el contexto de transacción de aplicación cliente

La política de servicio Web conectada al servicio Web de Business Process Choreographer permite que cada mensaje de solicitud pueda contener un contexto de transacción de WS-AT como se describe más arriba. Si ha elegido invocar las operaciones de servicio Web sin pasar un contexto de transacción de cliente, no es peligroso ignorar la política de transacción del lado de proveedor y configurar el cliente de servicio Web sin una política de transacción.

Desarrollo de aplicaciones cliente con la API JMS de Business Process Choreographer

Puede desarrollar aplicaciones cliente que acceden a las aplicaciones de proceso empresarialde manera asíncrona mediante la API de JMS (Java Messaging Service).

Acerca de esta tarea

Las aplicaciones de cliente JMS intercambian mensajes de petición y respuesta con la API de JMS. Para crear un mensaje de petición, la aplicación de cliente cumplimenta un cuerpo de mensaje TextMessage JMS con un elemento XML que representa la envoltura document/literal de la operación correspondiente.

Requisitos para los procesos de empresa

Los procesos de empresa desarrollados con WebSphere Integration Developer para que se ejecuten en Business Process Choreographer deben ajustarse a normas específicas para que se pueda acceder a los mismos desde la API de JMS.

Los requisitos son:

1. Las interfaces de los procesos de empresa se deben definir con el estilo "documento/literal con envoltura" definido en la API Java para la especificación RPC basada en XML, JAX-RPC 1.1. Este es el estilo por omisión para todos los procesos de empresa y tareas de usuario desarrollados con WebSphere Integration Developer.
2. Los mensajes de error que exponen los procesos de empresa y las tareas de usuario para las operaciones de servicios Web deben constar de una parte de mensaje WSDL individual definida con el elemento de esquema XML. Por ejemplo:

   ```
   <wsdl:part name="myFault" element="myNamespace:myFaultElement"/>
   ```

Información relacionada

Página de descargas de la API Java para RPC basado en XML, JAX-RPC

¿Qué estilo de WSDL debe utilizar?

Autorización para representaciones JMS

Para autorizar la utilización de la interfaz JMS, debe habilitar los valores de seguridad en WebSphere Application Server.

Cuando el contenedor de proceso empresarial está instalado, el rol **JMSAPIUser** debe correlacionarse con un ID de usuario. Este ID de usuario se utiliza para emitir todas las peticiones de la API de JMS. Por ejemplo, si **JMSAPIUser** se correlaciona con "Usuario A", todas las peticiones de la API de JMS aparecen en el motor de procesos para que se origine desde "Usuario A".

Se deben asignar las autorizaciones siguientes al rol **JMSAPIUser**:

Solicitud	Autorización necesaria
forceTerminate	Administrador de procesos
sendEvent	Propietario de actividad potencial o administrador de procesos

Nota: Para el resto de peticiones, no se necesitan autorizaciones especiales.

Se otorga una autorización especial a una persona con el rol de administrador de procesos empresariales. Un administrador de procesos empresariales es un rol especial; es diferente del administrador de procesos o de una instancia de proceso. Un administrador de procesos empresariales tiene todos los privilegios.

No puede suprimir el ID de usuario del iniciador del proceso desde el registro de usuarios mientras existe la instancia de proceso. Si lo hace, la navegación de este proceso no podrá continuar. Recibirá la excepción siguiente en el archivo de anotaciones cronológicas del sistema:

```
El ID no es exclusivo para:
<ID de usuario>
```

Acceso a la interfaz JMS

Para enviar y recibir mensajes mediante la interfaz JMS, una aplicación debe crear en primer lugar una conexión con BPC.nombre_célula.Bus, crear una sesión y, a continuación, generar productores y consumidores de mensajes.

Acerca de esta tarea

El servidor de procesos acepta mensajes JMS (Java Message Service) que siguen el paradigma de punto a punto. Una aplicación que envía o recibe mensajes JMS debe realizar las acciones siguientes.

En el ejemplo siguiente se supone que el cliente JMS se ejecuta en un entorno gestionado (EJB, cliente de aplicaciones o contenedor cliente Web).

Procedimiento

1. Cree una conexión con BPC.*nombre_célula*.Bus. No existe un fábrica de conexiones preconfigurada para las peticiones de una aplicación cliente: una aplicación cliente puede utilizar ReplyConnectionFactory de la API JMS o crear su propia fábrica de conexiones, en cuyo caso utiliza la búsqueda JNDI (Java Naming and Directory Interface) para recuperar la fábrica de conexiones. El nombre de búsqueda JNDI debe ser el mismo que el especificado al configurar la cola de solicitudes externas de Business Process Choreographer. En el ejemplo siguiente se supone que la aplicación cliente crea su propia fábrica de conexiones denominada "jms/clientCF".

   ```
   // Obtener el contexto inicial por omisión de JNDI.
   Context initialContext = new InitialContext();

   // Buscar la fábrica de conexiones.
   // Crear una fábrica de conexiones que se conecte al bus BPC.
   // Darle un nombre, por ejemplo, "jms/clientCF".
   // Configurar también un alias de autenticación adecuado.
   ConnectionFactory connectionFactory =
               (ConnectionFactory)initialcontext.lookup("jms/clientCF");

   // Crear la conexión.
   Connection connection = connectionFactory.createConnection();
   ```

2. Cree una sesión para que se puedan crear productores y consumidores de mensajes.

   ```
   // Crear una sesión de transacción mediante el acuse de recibo automático.
   Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
   ```

3. Crear un productor de mensajes para enviar mensajes. El nombre de búsqueda JNDI debe ser el mismo que el que se especificado al configurar la cola de solicitudes externas de Business Process Choreographer.

   ```
   // Buscar el destino de la cola de entrada de Business Process Choreographer
   // a la que se van a enviar mensajes.
   Queue sendQueue = (Queue) initialcontext.lookup("jms/BFMJMSAPIQueue");

   // Crear un productor de mensajes.
   MessageProducer producer = session.createProducer(sendQueue);
   ```

4. Cree un consumidor de mensajes para recibir respuestas. El nombre de búsqueda JNDI del destino de respuesta puede especificar un destino definido por el usuario, pero también puede especificar el destino de respuesta predeterminado (definido por Business Process Choreographer)`jms/BFMJMSReplyQueue`. En ambos casos, el destino de respuesta se debe basar en `BPC.<nombre_célula>.Bus`.

   ```
   // Buscar el destino de la cola de respuesta.
   Queue  replyQueue = (Queue) initialcontext.lookup("jms/BFMJMSReplyQueue");

   // Crear un consumidor de mensajes.
   MessageConsumer consumer = session.createConsumer(replyQueue);
   ```

5. Envíe un mensaje.

```
// Iniciar la conexión.
connection.start();

// Crear un mensaje (consulte las descripciones de tareas para ver ejemplos)
// y enviarlo.
// Est método se define en otra parte ...
String payload = createXMLDocumentForRequest();
TextMessage requestMessage = session.createTextMessage(payload);

// Establecer cabecera JMS obligatoria.
// targetFunctionName es el nombre de operación de la API JMS
// (por ejemplo, getProcessTemplate, sendMessage)
requestMessage.setStringProperty("TargetFunctionName", targetFunctionName);

// Establecer la cola de respuesta; esto es obligatorio si replyQueue
// no es la cola predeterminada (como lo es en este ejemplo).
requestMessage.setJMSReplyTo(replyQueue);

// Enviar el mensaje.
producer.send(requestMessage);

// Obtener el ID de mensaje.
String jmsMessageID = requestMessage.getJMSMessageID();

session.commit();
```

6. Reciba la respuesta.

```
// Recibir el mensaje de respuesta y analizar la respuesta.
TextMessage  replyMessage = (TextMessage) consumer.receive();

// Obtener la carga útil.
String payload = replyMessage.getText();

session.commit();
```

7. Cierre la conexión y libere los recursos.

```
 // Limpieza final; liberar los recursos.
session.close();
connection.close();
```

Nota: No es necesario cerrar la conexión después de cada transacción. Una vez que se ha iniciado una conexión, se puede intercambiar cualquier número de mensajes de solicitud y respuesta antes de que se cierre la conexión. En el ejemplo se muestra un caso sencillo con una sola llamada dentro de un solo método empresarial.

Estructura de un mensaje JMS de Business Process Choreographer

La cabecera y cuerpo de los mensajes JMS deben tener una estructura predefinida.

Un mensaje JMS (Java Message Service) consiste en:

- Una cabecera de mensaje para la identificación de mensajes y la información de direccionamiento.
- El cuerpo (carga útil) del mensaje que almacena el contenido.

Business Process Choreographer sólo da soporte a formatos de mensajes de texto.

Cabecera del mensaje

JMS permite que los clientes accedan a un número de campos de cabecera del mensaje.

Un cliente JMS de Business Process Choreographer puede establecer los siguientes campos de cabecera:

JMSReplyTo
El destino al que se envía una respuesta a la petición. Si este campo no se especifica en el mensaje de petición, la respuesta se envía al destino de respuesta predeterminado de la interfaz de exportación (una exportación es una representación de interfaz de cliente de un componente de proceso empresarial). Este destino se puede obtener mediante la utilización de `initialContext.lookup("jms/BFMJMSReplyQueue");`

TargetFunctionName
El nombre de la operación WSDL, por ejemplo, "queryProcessTemplates". Este campo siempre debe establecerse. Tenga en cuenta que TargetFunctionName especifica la operación de la interfaz de mensajes JMS genérica que se describe aquí. Esta operación no debe confundirse con otras proporcionadas por procesos o tareas determinadas que se pueden invocar indirectamente, por ejemplo, mediante operaciones call o sendMessage.

Un cliente de Business Process Choreographer también puede acceder a los siguientes campos de cabecera:

JMSMessageID
Identifica un mensaje de manera exclusiva. El proveedor JMS lo establece al enviar el mensaje. Si el cliente establece el JMSMessageID antes de enviar el mensaje, el proveedor JMS lo sobrescribe. Si se necesita el ID del mensaje para finalidades de autenticación, el cliente puede recuperar el JMSMessageID antes de enviar el mensaje.

JMSCorrelationID
Enlaza mensajes. No establezca este campo. Un mensaje de respuesta de Business Process Choreographer contiene el JMSMessageID del mensaje de respuesta.

Cada mensaje de respuesta contiene los siguientes campos de cabecera JMS:

- **IsBusinessException**

  "False" para mensajes de salida WSDL o "true" para mensajes de error WSDL.

No se devuelven excepciones ServiceRuntimeExceptions para aplicaciones de cliente asíncronas. Cuando se produce una excepción grave durante el proceso de un mensaje de petición JMS, tiene como resultado un error en tiempo de ejecución que provoca la retrotracción de la transacción que procesa este mensaje de petición. A continuación, el mensaje de petición JMS se vuelve a entregar. Si la anomalía se produce al principio, durante el proceso del mensaje como parte de la exportación de SCA (por ejemplo, al deserializar el mensaje), se efectúan reintentos hasta el número máximo de entregas con error especificado por el destino de recepción de la exportación de SCA. Después de alcanzarse el número máximo de entregas con error, el mensaje de petición se añade al destino de excepción del sistema del bus de Business Process Choreographer. No obstante, si el error se produce durante el proceso real de la petición efectuada por el componente SCA de Business Flow Manager, la infraestructura de gestión de sucesos anómalos de WebSphere Process Server gestionará el mensaje de petición con error, es decir, puede terminar en la base de datos de gestión de sucesos con error si los reintentos no resuelven la situación de excepción.

Cuerpo del mensaje

Las operaciones expuestas por procesos empresariales o tareas humanas deben cumplir con el estilo de envoltorio document/literal. El cuerpo de mensaje JMS es una serie que contiene un documento XML que representa el elemento de envoltorio document/literal de la operación. Las operaciones genéricas de la interfaz de mensajes JMS propagan el elemento de envoltorio de documento hacia y desde la operación implementada por el proceso empresarial o la tarea humana.

El ejemplo siguiente muestra un cuerpo de mensaje de solicitud válido simple:

```
<bfm:queryProcessTemplates
     xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0">
  <whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</bfm:queryProcessTemplates>
```

El ejemplo siguiente muestra un cuerpo de mensaje de solicitud válido más complejo. La aplicación cliente tiene una operación de API sendMessage para enviar un mensaje a un proceso específico. El mensaje de entrada del proceso es uno de los parámetros de la API; este mensaje es el mensaje de entrada de una operación empresarial expuesta por un proceso de cliente. El proceso contiene una actividad recibir que consume el mensaje.

El elemento `bfm:sendMessage` es el elemento de envoltorio del documento de la aplicación de la API de JMS. Incluye el elemento `cns:updateCustomer`, que es el elemento de envoltorio del documento para la operación implementada por el proceso. Este proceso tiene, por ejemplo, una actividad `bpel:receive` que hace referencia al tipo de puerto WSDL `cns:customerProcessPortType` y a la operación WSDL `updateCustomer`.

```
<bfm:sendMessage
     xmlns:bfm="http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0">
  <processTemplateName>customerProcessTemplate</processTemplateName>
  <portType xmlns:cns="http://example.com/customerProcess">cns:customerProcessPortType</portType>
  <operation>updateCustomer</operation>
  <cns:updateCustomer xmlns:cns="http://example.com/customerProcess">
    <street>1600 Pennsylvania Avenue Northwest</street>
    <city>Washington, DC 20006</city>
  </cns:updateCustomer>
</bfm:sendMessage>
```

Tareas relacionadas

"Comprobación del mensaje de respuesta para excepciones empresarial" en la página 473
Las aplicaciones de cliente JMS tienen que comprobar si existen excepciones empresariales en la cabecera de mensaje de todos los mensajes de respuesta.

Copia de artefactos para aplicaciones de cliente JMS

Se pueden copiar varios artefactos desde el entorno WebSphere Process Server para ayudar a crear las aplicaciones de cliente JMS.

Acerca de esta tarea

Estos artefactos sólo son obligatorios si utiliza BOXMLSerializer para crear el cuerpo del mensaje JMS. Para la API de JMS, estos artefactos son:

- `BFMIF.wsdl`
- `BFMIF.xsd`
- `BPCGen.xsd`
- `wsa.xsd`

Debe publicar y exportar estos archivos del entorno de WebSphere Process Server al entorno de desarrollo.

Publicación del archivo WSDL de proceso empresarial para aplicaciones JMS

Utilice la consola administrativa para publicar el archivo WSDL

Procedimiento

1. Inicie la sesión en la consola administrativa con el ID de usuario con derechos de administrador.
2. Pulse **Aplicaciones → Módulos SCA**.

   Nota: También puede pulsar **Aplicaciones → Tipos de aplicación → WebSphere Enterprise Applications** para visualizar una lista de todas las aplicaciones empresariales disponibles.
3. Seleccione la aplicación **BPEContainer** de la lista de módulos SCA o aplicaciones.
4. Seleccione **Publicar archivos WSDL** de la lista de **Propiedades adicionales**
5. Pulse el archivo .zip en la lista.
6. En la ventana Descarga de archivos que se muestra, pulse **Guardar**.
7. Vaya a la carpeta local y pulse **Guardar**.

Resultados

El archivo .zip exportado se denomina `BPEContainer_WSDLFiles.zip`. El archivo .zip contiene un archivo WSDL y los archivos XSD a los que el archivo WSDL hace referencia.

Comprobación del mensaje de respuesta para excepciones empresarial

Las aplicaciones de cliente JMS tienen que comprobar si existen excepciones empresariales en la cabecera de mensaje de todos los mensajes de respuesta.

Acerca de esta tarea

Una aplicación de cliente JMS tiene que comprobar, en primer lugar, la propiedad **IsBusinessException** en la cabecera del mensaje de respuesta.

Por ejemplo:

Ejemplo

```
// recibir mensaje de respuesta
Message receivedMessage = ((JmsProxy) getToBeInvokedUponObject().receiveMessage();
String strResponse = ((TextMessage) receivedMessage).getText();

if (receivedMessage.getStringProperty("IsBusinessException") {
  // strResponse es un error de empresa
  // las api pueden terminar con processFaultMsg
  // la api de llamada también con businessFaultMsg
 }
else {
  // strResponse es el mensaje de salida
}
```

Conceptos relacionados

"Estructura de un mensaje JMS de Business Process Choreographer" en la página 470
La cabecera y cuerpo de los mensajes JMS deben tener una estructura predefinida.

Ejemplo: ejecución de un proceso de larga duración con la API JMS de Business Process Choreographer

Este ejemplo muestra cómo crear una aplicación cliente genérica que utiliza la API JMS para trabajar con procesos de larga duración.

Procedimiento

1. Configure el entorno JMS, tal como se describe en "Acceso a la interfaz JMS" en la página 468.
2. Obtenga una lista de definiciones de procesos instalados.
   - Envíe `queryProcessTemplates`.
   - Esto devuelve una lista de objetos ProcessTemplate.
3. Obtenga una lista de actividades de inicio (de recepción o selección con `createInstance="yes"`).
   - Envíe `getStartActivities`.
   - Esto devuelve una lista de objetos InboundOperationTemplate.
4. Cree un mensaje de entrada. Esto es específico de entorno y puede que necesite utilizar artefactos predesplegados específicos de proceso.
5. Cree una instancia de proceso.
   - Emita un `sendMessage`.

   Con la API JMS, también puede usar la operación `call` para interactuar con operaciones de petición-respuesta de larga duración proporcionadas por un proceso de empresa. Esta operación devuelve el resultado o error de la operación en el destino de respuesta especificado, incluso después de un largo período de tiempo. Por tanto, si utiliza la operación `call`, no es necesario utilizar las operaciones `query` y `getOutputMessage` para obtener el mensaje de salida o error del proceso.
6. Opcional: Obtenga mensajes de salida desde instancias de proceso repitiendo los pasos siguientes:
   a. Emita `query` para obtener el estado de finalizado de la instancia de proceso.
   b. Emita `getOutputMessage` .
7. Opcional: Trabaje con operaciones adicionales expuestas por el proceso:
   a. Emita `getWaitingActivities` o `getActiveEventHandlers` para obtener una lista de objetos InboundOperationTemplate.
   b. Cree mensajes de entrada.
   c. Envíe mensajes con `sendMessage`.
8. Opcional: Obtenga y establezca propiedades personalizadas definidas en el proceso o actividades contenidas con `getCustomProperties` y `setCustomProperties`.
9. Deje de trabajar con una instancia de proceso:
   a. Envíe `delete` y `terminate` para dejar de trabajar con el proceso de larga duración.

Desarrollo de aplicaciones Web para procesos empresariales y tareas de usuario, utilizando componentes JSF

Business Process Choreographer proporciona varios componentes JSF (JavaServer Faces). Puede ampliar e integrar estos componentes para añadir funcionalidad de procesos empresariales y de tareas de usuario a las aplicaciones Web.

Acerca de esta tarea

Puede utilizar WebSphere Integration Developer para construir la aplicación Web. Para aquellas aplicaciones que incluyan tareas de usuario, puede generar un cliente personalizado JSF. Para obtener más información sobre cómo generar un cliente personalizado JSF, vaya al Centro de información de WebSphere Integration Developer.

También puede desarrollar el cliente Web utilizando los componentes JSF proporcionados por Business Process Choreographer.

Procedimiento

1. Cree un proyecto dinámico y cambie las propiedades de las Características de proyecto Web para incluir los componentes base de JSF.

   Para obtener más información sobre cómo crear un proyecto Web, vaya al Centro de información de WebSphere Integration Developer.
2. Añada los archivos JAR (Java Archive) de Business Process Choreographer Explorer de prerrequisito.

   Añada los archivos siguientes al directorio `WEB-INF/lib` del proyecto:

   - bpcclientcore.jar
   - bfmclientmodel.jar
   - htmclientmodel.jar
   - bpcjsfcomponents.jar

   En WebSphere Process Server, estos archivos se encuentran en el directorio siguiente:

   - Windows En las plataformas Windows: *raíz_instalación*\ProcessChoreographer\client
   - Linux UNIX En las plataformas Linux® y UNIX®: *raíz_instalación*/ProcessChoreographer/client
3. Añada las referencias de EJB que necesita al descriptor de despliegue de aplicaciones Web, el archivo `web.xml`.

```
<ejb-ref id="EjbRef_1">
  <ejb-ref-name>ejb/BusinessProcessHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.ibm.bpe.api.BusinessFlowManagerHome</home>
  <remote>com.ibm.bpe.api.BusinessFlowManager</remote>
</ejb-ref>
<ejb-ref id="EjbRef_2">
  <ejb-ref-name>ejb/HumanTaskManagerEJB</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>com.ibm.task.api.HumanTaskManagerHome</home>
  <remote>com.ibm.task.api.HumanTaskManager</remote>
</ejb-ref>
<ejb-local-ref id="EjbLocalRef_1">
  <ejb-ref-name>ejb/LocalBusinessProcessHome</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>com.ibm.bpe.api.LocalBusinessFlowManagerHome</local-home>
```

```
  <local>com.ibm.bpe.api.LocalBusinessFlowManager</local>
</ejb-local-ref>
<ejb-local-ref id="EjbLocalRef_2">
  <ejb-ref-name>ejb/LocalHumanTaskManagerEJB</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <local-home>com.ibm.task.api.LocalHumanTaskManagerHome</local-home>
  <local>com.ibm.task.api.LocalHumanTaskManager</local>
</ejb-local-ref>
```

4. Añada los componentes JSF de Business Process Choreographer Explorer a la aplicación JSF.
   a. Añada a los archivos JSP (JavaServer Pages) las referencias de bibliotecas de códigos que necesita para las aplicaciones. Normalmente, necesita las bibliotecas de códigos JSF y HTML y la biblioteca de códigos necesaria para los componentes JSF.
      - `<%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>`
      - `<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>`
      - `<%@ taglib uri="http://com.ibm.bpe.jsf/taglib" prefix="bpe" %>`
   b. Añada un código `<f:view>` al cuerpo de la página JSP y un código `<h:form>` al código `<f:view>`.
   c. Añada los componentes de JSF a los archivos JSP.

      En función de la aplicación, añada a los archivos JSP los componentes List, Details, CommandBar o Message. Puede añadir varias instancias de cada componente.
   d. Configure los beans gestionados en el archivo de configuración de JSF.

      Por omisión, el archivo de configuración es el archivo `faces-config.xml`. Este archivo está en el directorio `WEB-INF` de la aplicación Web.

      En función del componente que añada al archivo JSP, también tiene que añadir las referencias a la consulta y otros objetos de reiniciador al archivo de configuración de JSF. Para garantizar un manejo de errores correcto, también ha de definir un bean de error y un destino de navegación para la página de errores del archivo de configuración JSF. Asegúrese de que utiliza `BPCError` para el nombre del bean de error y `error` para el nombre del destino de navegación de la página de error.

```
<faces-config>
...
<managed-bean>
  <managed-bean-name>BPCError</managed-bean-name>
  <managed-bean-class>com.ibm.bpc.clientcore.util.ErrorBeanImpl
  </managed-bean-class>
  <managed-bean-scope>session</managed-bean-scope>
</managed-bean>

...
<navigation-rule>
...
<navigation-case>
<description>
La página de error general.
</description>
<from-outcome>error</from-outcome>
<to-view-id>/Error.jsp</to-view-id>
</navigation-case>
...
</navigation-rule>
</faces-config>
```

      En situaciones de error que desencadenan la página de errores, se establece la excepción en el bean de error.

e. Implemente el código personalizado que tiene que dar soporte a los componentes JSF.

5. Despliegue la aplicación.

   Si va a desplegar la aplicación en un entorno de despliegue de red, cambie los nombres JNDI (Java Naming and Directory Interface) por valores donde se puedan encontrar las API de Business Flow Manager y de Human Task Manager en la célula.

   - Si los contenedores de procesos empresariales están configurados en otro servidor de la misma célula gestionada, los nombres tienen esta estructura:

     célula/nodos/*nombre_nodo*/servidores/*nombre_servidor*
     /com/ibm/bpe/api/BusinessManagerHome
     célula/nodos/*nombre_nodo*/servidores/*nombre_servidor*
     /com/ibm/task/api/HumanTaskManagerHome

   - Si los contenedores de procesos empresariales están configurados en otro servidor de la misma célula, los nombres tienen esta estructura:

     célula/clústeres/*nombre_clúster*/com/ibm/bpe/api/BusinessFlowManagerHome
     célula/clústeres/*nombre_clúster*/com/ibm/task/api/HumanTaskManagerHome

   Correlacione las referencias de EJB a nombres JNDI o añada manualmente las referencias al archivo ibm-web-bnd.xmi.

   En la tabla siguiente se listan los enlaces de referencia y sus correlaciones por omisión.

Tabla 70. Correlación de los enlaces de referencia con nombres JNDI

Enlace de referencia	Nombre JNDI	Comentarios
ejb/BusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	Bean de sesión remota
ejb/LocalBusinessProcessHome	com/ibm/bpe/api/BusinessFlowManagerHome	Bean de sesión local
ejb/HumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	Bean de sesión remota
ejb/LocalHumanTaskManagerEJB	com/ibm/task/api/HumanTaskManagerHome	Bean de sesión local

Resultados

La aplicación Web desplegada contiene la funcionalidad proporcionada por los componentes de Business Process Choreographer Explorer.

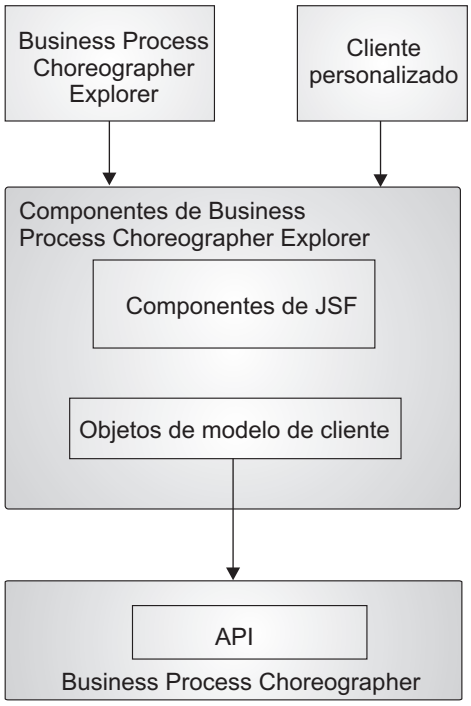
Qué hacer a continuación

Si utiliza JSP personalizados para los mensajes de procesos y de tareas, debe correlacionar los módulos Web que se utilizan para desplegar los JSP con los mismos servidores con los que está correlacionado el cliente JSF personalizado.

Componentes de Business Process Choreographer Explorer

Los componentes de Business Process Choreographer Explorer son un conjunto de elementos configurables y reutilizables que están basados en la tecnología JSF (JavaServer Faces). Puede incorporar estos elementos en aplicaciones Web. Las aplicaciones Web pueden acceder a las aplicaciones instaladas de procesos de empresa y tareas de usuario.

Los componentes constan de un conjunto de componentes JSF y un conjunto de objetos de modelo de cliente. La relación de los componentes con Business Process Choreographer, Business Process Choreographer Explorer y otros clientes personalizados se muestran en la figura siguiente.



Componentes de JSF

Los componentes de Business Process Choreographer Explorer incluyen los siguientes componentes de JSF. Debe incorporar estos componentes JSF en los archivos JSP (JavaServer Pages) cuando cree aplicaciones Web para trabajar con procesos de empresa y tareas de usuario.

- Componente List

  El componente List muestra una lista de objetos de aplicación en una tabla, por ejemplo, tareas, actividades, instancias de proceso, plantillas de proceso, elementos de trabajo o escaladas. Este componente tiene un manejador de lista asociado.
- Componente Details

  El componente Details muestra las propiedades de las tareas, elementos de trabajo, actividades, instancias de proceso y plantillas de proceso. Este componente tiene un manejador de detalles asociado.
- Componente CommandBar

  El componente CommandBar muestra una barra con botones. Estos botones representan mandatos que operan en la vista de detalles del objeto o en los objetos seleccionados en una lista. Un manejador de listas o un manejador de detalles proporciona estos objetos.
- Componente Message

  El componente Message muestra un mensaje que puede contener un SDO (Service Data Object) o un tipo simple.

Objetos de modelo de cliente

Los objetos de modelo de cliente se utilizan con los componentes JSF. Los objetos implementan algunas de las interfaces de la API de Business Process Choreographer subyacente y acomodan el objeto original. Los objetos de modelo

de cliente proporcionan soporte de idioma nacional para las etiquetas y convertidores para algunas propiedades.

Manejo de errores en componentes JSF

Los componentes JSF (JavaServer Faces) utilizan un bean gestionado definido previamente, BPCError, para el manejo de errores. En situaciones de error que desencadenan la página de errores, se establece la excepción en el bean de error.

Este bean implementa la interfaz com.ibm.bpc.clientcore.util.ErrorBean. La página de errores se muestra en estas situaciones:

- Si se produce un error durante la ejecución de una consulta que se define para un manejador de listas y el método execute de un mandato genera el error como un error de ClientException
- Si el método execute de un mandato genera un error de ClientException y este error no es un error de ErrorsInCommandException ni tampoco implementa la interfaz CommandBarMessage
- Si se muestra una mensaje de error en el componente y sigue el hiperenlace del mensaje

Está disponible una implementación por omisión de la interfaz com.ibm.bpc.clientcore.util.ErrorBeanImpl.

La interfaz se define como se detalla a continuación:

```
public interface ErrorBean {

    public void setException(Exception ex);

    /*
     * Esta llamada al método setter permite que se pase un entorno local
     * y la excepción. Esto permite que los métodos
     * getExceptionMessage devuelvan series adaptadas
     *
     */
    public void setException(Exception ex, Locale locale);

    public Exception getException();
    public String getStack();
    public String getNestedExceptionMessage();
    public String getNestedExceptionStack();
    public String getRootExceptionMessage();
    public String getRootExceptionStack();

    /*
     * Este método devuelve el mensaje de excepción
     * concatenado recursivamente con los mensajes de todas
     * las excepciones anidadas.
     */
    public String getAllExceptionMessages();

    /*
     * Este método devuelve la pila de excepciones
     * concatenada recursivamente con las pilas de todas
     * las excepciones anidadas.
     */
    public String getAllExceptionStacks();
}
```

Conceptos relacionados

"Manejo de errores del componente List" en la página 485
Cuando se utiliza el componente List para visualizar listas en la aplicación JSF, puede aprovechar las funciones de manejo de errores proporcionadas por la clase com.ibm.bpe.jsf.handler.BPCListHandler.

Convertidores y etiquetas por omisión para objetos de modelo de cliente

Los objetos de modelo de cliente implementan las interfaces correspondientes de la API de Business Process Choreographer.

Los componentes List y Details funcionan en cualquier bean. Puede visualizar todas las propiedades de un bean. Sin embargo, si desea establecer los convertidores y las etiquetas que se utilizan para las propiedades de un bean, debe utilizar el código `column` para el componente List, o bien el código `property` para el componente Details. En lugar de establecer los convertidores y las etiquetas, puede definir convertidores y etiquetas por omisión para las propiedades definiendo los siguientes métodos estáticos. Puede definir los siguientes métodos estáticos:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
       getConverter(String property);
```

En la tabla siguiente se muestran los objetos de modelo de cliente que implementan las clases de API de Business Flow Manager y del Gestor de tareas de usuario correspondientes y proporcionan etiquetas y convertidores por omisión para sus propiedades. Esta acomodación de las interfaces proporciona etiquetas sensibles al entorno local y convertidores de un conjunto de propiedades. La tabla siguiente muestra la correlación de las interfaces de Business Process Choreographer con los objetos de modelo de cliente correspondientes.

Tabla 71. Cómo las interfaces de Business Process Choreographer se correlacionan con objetos de modelo de cliente

Interfaz de Business Process Choreographer	Clase de objeto de modelo de cliente
com.ibm.bpe.api.ActivityInstanceData	com.ibm.bpe.clientmodel.bean.ActivityInstanceBean
com.ibm.bpe.api.ActivityServiceTemplateData	com.ibm.bpe.clientmodel.bean.ActivityServiceTemplateBean
com.ibm.bpe.api.ProcessInstanceData	com.ibm.bpe.clientmodel.bean.ProcessInstanceBean
com.ibm.bpe.api.ProcessTemplateData	com.ibm.bpe.clientmodel.bean.ProcessTemplateBean
com.ibm.task.api.Escalation	com.ibm.task.clientmodel.bean.EscalationBean
com.ibm.task.api.Task	com.ibm.task.clientmodel.bean.TaskInstanceBean
com.ibm.task.api.TaskTemplate	com.ibm.task.clientmodel.bean.TaskTemplateBean

Adición del componente List a una aplicación JSF

Utilice el componente List de Business Process Choreographer Explorer para visualizar una lista de objetos de modelo de cliente, por ejemplo, instancias de proceso empresarial o instancias de tarea.

Procedimiento

1. Añada el componente List al archivo JSP (JavaServer Pages).

Añada el código bpe:list al código h:form. El código bpe:list debe incluir un atributo model. Añada los códigos bpe:column al código bpe:list para añadir las propiedades de los objetos que han de aparecer en cada una de las filas de la lista.

El siguiente ejemplo muestra cómo añadir un componente List para visualizar instancias de tarea.

```
<h:form>

   <bpe:list model="#{TaskPool}">
      <bpe:column name="name" action="taskInstanceDetails" />
      <bpe:column name="state" />
      <bpe:column name="kind" />
      <bpe:column name="owner" />
      <bpe:column name="originator" />
   </bpe:list>

</h:form>
```

El atributo model hace referencia a un bean gestionado, TaskPool. El bean gestionado proporciona la lista de objetos Java que la lista reitera y luego visualiza en filas individuales.

2. Configure el bean gestionado al que se hace referencia en el código bpe:list.

   Para el componente List, este bean gestionado debe ser una instancia de la clase com.ibm.bpe.jsf.handler.BPCListHandler.

   El siguiente ejemplo muestra cómo añadir el bean gestionado TaskPool al archivo de configuración.

```
<managed-bean>
<managed-bean-name>TaskPool</managed-bean-name>
<managed-bean-class>com.ibm.bpe.jsf.handler.BPCListHandler</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
   <managed-property>
      <property-name>query</property-name>
      <value>#{TaskPoolQuery}</value>
   </managed-property>
   <managed-property>
      <property-name>type</property-name>
      <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
   </managed-property>
</managed-bean>

<managed-bean>
<managed-bean-name>TaskPoolQuery</managed-bean-name>
<managed-bean-class>sample.TaskPoolQuery</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
   <managed-property>
     <property-name>type</property-name>
     <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
   </managed-property>
</managed-bean>

<managed-bean>
<managed-bean-name>htmConnection</managed-bean-name>
<managed-bean-class>com.ibm.task.clientmodel.HTMConnection</managed-bean-class>
<managed-bean-scope>application</managed-bean-scope>
   <managed-property>
      <property-name>jndiName</property-name>
      <value>java:comp/env/ejb/LocalHumanTaskManagerEJB</value>
   </managed-property>
</managed-bean>
```

En el ejemplo se muestra que TaskPool tiene dos propiedades configurables: query y type. El valor de la propiedad query hace referencia a otro bean gestionado, TaskPoolQuery. El valor de la propiedad type especifica la clase de

bean, cuyas propiedades se muestran en las columnas de la lista visualizada. La instancia de consulta asociada también puede tener un tipo de propiedad. Si se especifica un tipo de propiedad, debe ser el mismo que el tipo especificado para el manejador de listas.

Puede añadir cualquier tipo de lógica de consulta a la aplicación JSF siempre que el resultado de la consulta pueda representarse como una lista de beans de tipo fuerte. Por ejemplo, TaskPoolQuery se implementa mediante una lista de objetos com.ibm.task.clientmodel.bean.TaskInstanceBean.

3. Añada el código personalizado del bean gestionado al que hace referencia el manejador de listas.

   El siguiente ejemplo muestra cómo añadir código personalizado al bean gestionado TaskPool.

```
public class TaskPoolQuery implements Query {

  public List execute throws ClientException {

    // Examinar el archivo faces-config para un bean gestionado "htmConnection".
    //
    FacesContext ctx = FacesContext.getCurrentInstance();
    Application  app = ctx.getApplication();
    ValueBinding htmVb = app.createValueBinding("#{htmConnection}");
    htmConnection = (HTMConnection) htmVb.getValue(ctx);
    HumanTaskManagerService taskService =
         htmConnection.getHumanTaskManagerService();

    // A continuación, llamar al método de consulta actual en el servicio de
    // Human Task Manager.
    //
    // Añada las columnas de base de datos de todas las propiedades que desee mostrar
    // en la lista de la sentencia seleccionada
    //
    QueryResultSet queryResult = taskService.query(
     "DISTINCT TASK.TKIID, TASK.NAME, TASK.KIND, TASK.STATE, TASK.TYPE,"
      + "TASK.STARTER, TASK.OWNER, TASK.STARTED, TASK.ACTIVATED, TASK.DUE,
         TASK.EXPIRES, TASK.PRIORITY",
      "TASK.KIND IN(101,102,105) AND TASK.STATE IN(2)
        AND WORK_ITEM.REASON IN (1)",
      (String)null,
      (Integer)null,
      (TimeZone)null);
    List applicationObjects = transformToTaskList ( queryResult );
    return applicationObjects ;
  }


  private List transformToTaskList(QueryResultSet result) {

ArrayList array = null;
int entries = result.size();
array = new ArrayList( entries );

// Transforma cada fila de QueryResultSet en beans de instancia de tarea.
 for (int i = 0; i < entries; i++) {
   result.next();
   array.add( new TaskInstanceBean( result, connection ));
 }
 return array ;
  }
}
```

   El bean TaskPoolQuery consulta las propiedades de los objetos Java. Este bean debe implementar la interfaz com.ibm.bpc.clientcore.Query. Cuando el manejador de listas renueva su contenido, llama al método execute de la

consulta. La llamada devuelve una lista de objetos Java. El método getType debe devolver el nombre de clase de los objetos Java devueltos.

Resultados

La aplicación JSF contiene ahora una página JavaServer que muestra las propiedades de la lista de objetos solicitada, por ejemplo, el estado, clase, propietario y originador de las instancias de tarea que están disponibles para el usuario.

Cómo se procesan las listas

Todas las instancias del componente List tienen asociada una instancia de la clase com.ibm.bpe.jsf.handler.BPCListHandler.

Este manejador de listas realiza un seguimiento de los elementos seleccionados y proporciona una mecanismo de notificación para asociar las entradas de lista a las páginas de detalles para los distintos tipos de elementos. El manejador de la lista se enlaza con el componente List mediante el atributo **model** del código `bpe:list`.

El mecanismo de notificación del manejador de listas se implementa utilizando la interfaz com.ibm.bpe.jsf.handler.ItemListener. Puede registrar las implementaciones de esta interfaz en el archivo de configuración de la aplicación JSF (JavaServer Faces).

Se desencadena la notificación cuando se pulsa un enlace de la lista. Los enlaces se representan para todas las columnas para los que se ha establecido el atributo **action**. El valor del atributo **action** es un destino de navegación JSF o un método de acción JSF que devuelve un destino de navegación JSF.

La clase BPCListHandler también proporciona un método refreshList. Puede utilizar este método en enlaces de método JSF para implementar un control de interfaz de usuario para ejecutar de nuevo la consulta.

Implementaciones de consulta

Puede utilizar el manejador de listas para mostrar todos los tipos de objetos y sus propiedades. El contenido de la lista que se muestra depende de la lista de objetos que la implementación de la interfaz com.ibm.bpc.clientcore.Query que está configurada para el manejador de listas devuelve. Puede establecer la consulta mediante programación utilizando el método setQuery de la clase BPCListHandler o puede configurarla en los archivos de configuración JSF de la aplicación.

Puede ejecutar las consultas sólo con las API de Business Process Choreographer, pero también con cualquier otra fuente de información que sea accesible desde la aplicación, por ejemplo, un sistema de gestión de contenido o una base de datos. El único requisito es que el resultado de la consulta se devuelva como una `java.util.List` de objetos del método execute.

El tipo de los objetos devueltos debe garantizar que están disponibles los métodos getter adecuados para todas las propiedades que se muestran en las columnas de la lista para la que se define la consulta. Para asegurarse de que el tipo del objeto que se devuelve se ajusta a las definiciones de lista, puede establecer el valor de la propiedad type en la instancia de BPCListHandler que se define en el archivo de configuración de Faces con el nombre de clase plenamente cualificado de los objetos devueltos. Puede devolver este nombre en la llamada a getType de la

implementación de consulta. Durante la ejecución, el manejador de listas comprueba que los tipos de objetos son conforme a las definiciones.

Para correlacionar los mensajes de error con entradas específicas de una lista, los objetos devueltos por la consulta deben implementar un método con la signatura `public Object getID()`.

Convertidores y etiquetas por omisión

Los elementos devueltos por una consulta deben ser beans y sus clases deben coincidir con la clase especificada como el tipo en la definición de la clase BPCListHandler o de la interfaz com.ibm.bpc.clientcore.Query. Además, el componente List comprueba si la clase del elemento o una superclase implementa los métodos siguientes:

```
static public String getLabel(String property,Locale locale);
static public com.ibm.bpc.clientcore.converter.SimpleConverter
       getConverter(String property);
```

Si estos métodos se definen para los beans, el componente List utiliza la etiqueta como etiqueta predeterminada para la lista y SimpleConverter como convertidor predeterminado para la propiedad. Puede sobrescribir estos valores con los atributos **label** y **converterID** del código `bpe:list`. Si desea obtener más información, consulte el Javadoc para la interfaz SimpleConverter y la clase ColumnTag.

Información de huso horario específica del usuario

Los componentes JSF (JavaServer Faces) proporcionan un programa de utilidad para gestionar información de huso horario específica del usuario en el componente List.

La clase BPCListHandler utiliza la interfaz com.ibm.bpc.clientcore.util.User para obtener información sobre el huso horario y el entorno local de cada usuario. El componente List espera que la implementación de la interfaz se configure con **user** como el nombre de bean gestionado en el archivo de configuración JSF (JavaServer Faces). Si faltara esta entrada en el archivo de configuración, se devolvería el huso horario en el que se ejecuta WebSphere Process Server.

La interfaz com.ibm.bpc.clientcore.util.User se define de la siguiente manera:

```
public interface User {

    /**
     * El entorno local utilizado por el cliente del usuario.
     * @return El entorno local.
     */
    public Locale getLocale();
   /**
    * El huso horario utilizado por el cliente del usuario.
    * @return El huso horario.
    */
    public TimeZone getTimeZone();

   /**
    * El nombre del usuario.
    * @return el nombre del usuario.
    */
    public String getName();
}
```

Manejo de errores del componente List

Cuando se utiliza el componente List para visualizar listas en la aplicación JSF, puede aprovechar las funciones de manejo de errores proporcionadas por la clase com.ibm.bpe.jsf.handler.BPCListHandler.

Errores que se producen cuando se ejecutan consultas o se ejecutan mandatos

Si se produce un error durante la ejecución de una consulta, la clase BPCListHandler distingue entre errores que se han producido por derechos de acceso insuficientes y otras excepciones. Para obtener errores por derechos de acceso insuficientes, el parámetro **rootCause** de la excepción ClientException que el método execute de la consulta genera debe ser una excepción com.ibm.bpe.api.EngineNotAuthorizedException o com.ibm.task.api.NotAuthorizedException. El componente List muestra el mensaje de error en lugar del resultado de la consulta.

Si el error no se ha producido por derechos de acceso insuficientes, la clase BPCListHandler pasa el objeto de excepción a la implementación de la interfaz com.ibm.bpc.clientcore.util.ErrorBean que se define mediante la clave BPCError en el archivo de configuración de la aplicación JSF. Cuando se establece la excepción, se llama al destino de navegación de errores.

Errores que se producen al trabajar con elementos que se muestran en la lista

La clase BPCListHandler implementa la interfaz com.ibm.bpe.jsf.handler.ErrorHandler. Puede proporcionar información sobre estos errores con el parámetro map de tipo java.util.Map del método setErrors. Esta correlación contiene identificadores como claves y las excepciones como valores. Los identificadores deben ser los valores devueltos por el método getID del objeto que ha producido el error. Si se establece la correlación y cualquiera de los ID coincide con cualquiera de los elementos mostrados en la lista, el manejador de listas añade automáticamente una columna que contiene el mensaje de error a la lista.

Para impedir mensajes de error obsoletos en la lista, restablezca la correlación de errores. En las siguientes situaciones, se restablece automáticamente la correlación:

- Se llama a la clase BPCListHandler del método refreshList.
- Se establece una nueva consulta en la clase BPCListHandler.
- Se utiliza el componente CommandBar para desencadenar acciones sobre los elementos de la lista. El componente CommandBar utiliza este mecanismo como uno de los métodos de manejo de errores.

Conceptos relacionados

"Manejo de errores en componentes JSF" en la página 479
Los componentes JSF (JavaServer Faces) utilizan un bean gestionado definido previamente, BPCError, para el manejo de errores. En situaciones de error que desencadenan la página de errores, se establece la excepción en el bean de error.

Componente List: definiciones de código

El componente List de Business Process Choreographer Explorer muestra una lista de objetos en una tabla, por ejemplo, tareas, actividades, instancias de proceso, plantillas de proceso, elementos de trabajo y escaladas.

El componente List consta de los códigos de componente JSF: `bpe:list` y `bpe:column`. El código `bpe:column` es un subelemento del código `bpe:list`.

Clase de componente

com.ibm.bpe.jsf.component.ListComponent

Sintaxis de ejemplo

```
<bpe:list model="#{ProcessTemplateList}">
          rows="20"
          styleClass="list"
          headerStyleClass="listHeader"
          rowClasses="normal">

    <bpe:column name="name" action="processTemplateDetails"/>
    <bpe:column name="validFromTime"/>
    <bpe:column name="executionMode" label="Execution mode"/>
    <bpe:column name="state" converterID="my.state.converter"/>
    <bpe:column name="autoDelete"/>
    <bpe:column name="description"/>

</bpe:list>
```

Atributos de código

El cuerpo del código `bpe:list` sólo puede contener códigos `bpe:column`. Cuando se representa la tabla, el componente List itera por la lista de objetos de aplicación y representa todas las columnas para cada objeto.

Tabla 72. Atributos de bpe:list

Atributo	Necesario	Descripción
buttonStyleClass	no	Clase de estilo de hoja de estilos en cascada (CSS) para representar los botones del área de pie de página.
cellStyleClass	no	Clase de estilo CSS para representar celdas de tabla individuales.
checkbox	no	Determina si se representa el recuadro de selección para seleccionar varios elementos. El atributo tiene el valor `true` o `false`. Si el valor se establece en `true`, se representa la columna del recuadro de selección.
headerStyleClass	no	Clase de estilo CSS para representar la cabecera de tabla.
model	sí	Enlace de valor para un bean gestionado de la clase com.ibm.bpe.jsf.handler.BPCListHandler.
rows	no	Número de filas que se muestran en una página. Si el número de elementos sobrepasa el número de filas, se muestran botones de paginación al final de la tabla. Las expresiones de valores no están soportadas para este atributo.
rowClasses	no	Clase de estilo CSS para representar las filas de la tabla.

Tabla 72. Atributos de bpe:list (continuación)

Atributo	Necesario	Descripción
selectAll	no	Si este atributo se establece en `true`, todos los elementos de la lista se seleccionan por omisión.
styleClass	no	Clase de estilo CSS para representar la tabla general que contiene títulos, filas y botones de paginación.

Tabla 73. Atributos de bpe:column

Atributo	Necesario	Descripción
action	no	Si se especifica el atributo, se representa un enlace en la columna. Un método de acción JavaServer Faces o el destino de navegación Faces se desencadena cuando se pulsa este enlace. Un método de acción JavaServer Faces tiene la signatura siguiente: String method().
converterID	no	El ID de convertidor Faces que se utiliza para convertir el valor de propiedad. Si no se establece este atributo, se utiliza cualquier ID de convertidor Faces que se proporcione por el modelo para esta propiedad.
label	no	Un literal o expresión de enlace de valor que se utiliza como etiqueta para la cabecera de la columna o la celda de la fila de cabecera de la tabla. Si no se establece este atributo, se utiliza cualquier etiqueta que se proporcione por el modelo para esta propiedad.
name	sí	Nombre de la propiedad que se visualiza en esta columna.

Adición del componente Details a una aplicación JSF

Utilice el componente Details de Business Process Choreographer Explorer para visualizar las propiedades de tareas, elementos de trabajo, actividades, instancias de proceso y plantillas de proceso.

Procedimiento

1. Añada el componente Details al archivo JSP (JavaServer Pages).

   Añada el código `bpe:details` al código `<h:form>`. El código `bpe:details` debe contener un atributo **model**. Puede añadir propiedades al componente Details con el código `bpe:property`.

   En el ejemplo siguiente se muestra cómo añadir un componente Details para visualizar algunas de las propiedades de una instancia de tarea.

   ```
   <h:form>

      <bpe:details model="#{TaskInstanceDetails}">
         <bpe:property name="displayName" />
         <bpe:property name="owner" />
         <bpe:property name="kind" />
         <bpe:property name="state" />
   ```

```
        <bpe:property name="escalated" />
        <bpe:property name="suspended" />
        <bpe:property name="originator" />
        <bpe:property name="activationTime" />
        <bpe:property name="expirationTime" />
      </bpe:details>

</h:form>
```

El atributo **model** hace referencia a un bean gestionado, TaskInstanceDetails. El bean proporciona las propiedades del objeto Java.

2. Configure el bean gestionado al que se hace referencia en el código `bpe:details`.

   Para el componente Details, este bean gestionado debe ser una instancia de la clase com.ibm.bpe.jsf.handler.BPCDetailsHandler. Esta clase de manejador reinicia un objeto Java y expone sus propiedades públicas al componente de detalles.

   En el ejemplo siguiente se muestra cómo añadir el bean gestionado TaskInstanceDetails al archivo de configuración.

```
<managed-bean>
   <managed-bean-name>TaskInstanceDetails</managed-bean-name>
   <managed-bean-class>com.ibm.bpe.jsf.handler.BPCDetailsHandler</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>
   <managed-property>
      <property-name>type</property-name>
      <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
   </managed-property>
</managed-bean>
```

   En el ejemplo se muestra que el bean TaskInstanceDetails tiene una propiedad `type` configurable. El valor de la propiedad type especifica la clase de bean (com.ibm.task.clientmodel.bean.TaskInstanceBean), cuyas propiedades se muestran en las filas de los detalles visualizados. La clase de bean puede ser cualquier clase de JavaBeans. Si el bean proporciona por omisión etiquetas de propiedades y convertidor, el convertidor y la etiqueta se utilizan para la representación, de igual manera que para el componente List.

Resultados

La aplicación JSF contiene ahora una página JavaServer que muestra los detalles del objeto especificado, por ejemplo, los detalles de una instancia de tarea.

Componente Details: definiciones de código

El componente Details de Business Process Choreographer Explorer muestra las propiedades de tareas, elementos de trabajo, actividades, instancias de proceso y plantillas de proceso.

El componente Details consta de los códigos de componente JSF: `bpe:details` y `bpe:property`. El código `bpe:property` es un subelemento del código `bpe:details`.

Clase de componente

com.ibm.bpe.jsf.component.DetailsComponent

Sintaxis de ejemplo

```
<bpe:details model="#{MyActivityDetails}">
    <bpe:property name="name"/>
    <bpe:property name="owner"/>
    <bpe:property name="activated"/>
</bpe:details>
```

```
<bpe:details model="#{MyActivityDetails}" style="style" styleClass="cssStyle">
             style="style"
             styleClass="cssStyle"
</bpe:details>
```

Atributos de código

Utilice códigos `bpe:property` para especificar tanto el subconjunto de atributos que se muestran como el orden en que se muestran estos atributos. Si el código de detalles no contiene códigos de atributo, representa todos los atributos disponibles del objeto modelo.

Tabla 74. Atributos de bpe:details

Atributo	Necesario	Descripción
columnClasses	no	Lista de clases de estilo de la hoja de estilo en cascada (CSS), separada por comas, para representar columnas.
id	no	ID de JavaServer Faces del componente.
model	sí	Enlace de valor para un bean gestionado de la clase com.ibm.bpe.jsf.handler.BPCDetailsHandler.
rowClasses	no	Lista de clases de estilo CSS, separados por comas, para representar filas.
styleClass	no	Clase CSS que se utiliza para representar el elemento HTML.

Tabla 75. Atributos de bpe:property

Atributo	Necesario	Descripción
converterID	no	ID utilizado para registrar el convertidor en el archivo de configuración de JSF (JavaServer Faces).
label	no	Etiqueta de la propiedad. Si no se establece este atributo, la clase de modelo de cliente proporciona una etiqueta por omisión.
name	sí	Nombre de la propiedad que va a visualizarse. Este nombre debe corresponder a una propiedad con nombre tal como se define en la clase de modelo de cliente correspondiente.

Adición del componente CommandBar a una aplicación JSF

Utilice el componente CommandBar de Business Process Choreographer Explorer para visualizar una barra con botones. Estos botones representan mandatos que operan en la vista de detalles de un objeto o los objetos seleccionados en una lista.

Acerca de esta tarea

Cuando el usuario pulsa un botón en la interfaz de usuario, se ejecuta el mandato correspondiente en los objetos seleccionados. Puede añadir y ampliar el componente CommandBar en la aplicación JavaServer Faces (JSF).

Procedimiento

1. Añada el componente CommandBar al archivo JSP (JavaServer Pages).

Añada el código bpe:commandbar al código <h:form>. El código bpe:commandbar debe contener un atributo model.

El ejemplo siguiente muestra cómo añadir un componente CommandBar que proporciona mandatos refresh y claim para una lista de instancias de tareas.

```
<h:form>

   <bpe:commandbar model="#{TaskInstanceList}">
      <bpe:command commandID="Refresh" >
                   action="#{TaskInstanceList.refreshList}"
                   label="Refresh"/>

      <bpe:command commandID="MyClaimCommand" >
                   label="Claim" >
                   commandClass="<customcode>"/>
   </bpe:commandbar>

</h:form>
```

El atributo **model** hace referencia a un bean gestionado. Este bean debe implementar la interfaz ItemProvider y proporcionar los objetos Java seleccionados. El componente CommandBar suele utilizarse con el componente List o el componente Details en el mismo archivo JSP. Generalmente, el modelo especificado en el código es el mismo que el especificado en el componente List o en el componente Details de la misma página. Así pues, para un componente List, por ejemplo, el mandato actúa sobre los elementos seleccionados de la lista.

En este ejemplo, el atributo **model** hace referencia al bean gestionado TaskInstanceList. Este bean proporciona los objetos seleccionados en la lista de instancias de tareas. El bean debe implementar la interfaz ItemProvider. Las clases BPCListHandler y BPCDetailsHandler implementan esta interfaz.

2. Opcional: Configure el bean gestionado al que se hace referencia en el código bpe:commandbar.

   Si el atributo **model** de CommandBar hace referencia a un bean gestionado que ya está configurado, por ejemplo, para un manejador de lista o de detalles, no se necesita ninguna configuración adicional. Si no utiliza la clase BPCListHandler ni la clase BPCDetailsHandler para el modelo, debe hacer referencia a otro objeto que tenga una clase que implemente la interfaz ItemProvider.

3. Añada el código que implementa los mandatos personalizados en la aplicación JSF.

   El siguiente fragmento de código muestra cómo escribir una clase de mandato que implemente la interfaz de mandatos. Se hace referencia a esta clase de mandato (MyClaimCommand) mediante el código bpe:command en el archivo JSP.

```
public class MyClaimCommand implements Command {

  public String execute(List selectedObjects) throws ClientException {
     if( selectedObjects != null && selectedObjects.size() > 0 ) {
        try {
         // Determinar HumanTaskManagerService de un bean HTMConnection.
         // Configurar el bean en faces-config.xml para facilitar el acceso
         // en la aplicación JSF.
         FacesContext ctx = FacesContext.getCurrentInstance();
         ValueBinding vb =
           ctx.getApplication().createValueBinding("{htmConnection}");
         HTMConnection htmConnection = (HTMConnection) htmVB.getValue(ctx);
         HumanTaskManagerService htm =
            htmConnection.getHumanTaskManagerService();
```

```
        Iterator iter = selectedObjects.iterator() ;
        while( iter.hasNext() ) {
          try {
               TaskInstanceBean task = (TaskInstanceBean) iter.next() ;
               TKIID tiid = task.getID() ;

               htm.claim( tiid ) ;
               task.setState( new Integer(TaskInstanceBean.STATE_CLAIMED ) ) ;

          }
          catch( Exception e ) {
            ;   // Error al iterar o reclamar una instancia de tarea.
                 // Omitir para una mejor comprensión del ejemplo.
          }
        }
      }
      catch( Exception e ) {
        ;  // Error de configuración o comunicación.
          // Ignorar para una mejor comprensión del ejemplo
      }
    }
    return null;
  }

 // Default implementations
 public boolean isMultiSelectEnabled() { return false; }
 public boolean[] isApplicable(List itemsOnList) {return null; }
 public void setContext(Object targetModel) {; // Not used here }
}
```

El mandato se procesa de la manera siguiente:

a. Se invoca un mandato cuando un usuario pulsa el botón correspondiente de la barra de mandatos. El componente CommandBar recupera los elementos seleccionados del proveedor de elementos especificado en el atributo **model** y pasa la lista de objetos seleccionados al método execute de la instancia de CommandBar.
b. Opcional: El atributo **commandClass** hace referencia a una implementación de mandatos personalizada que implementa la interfaz de mandatos. Esto significa que el mandato debe implementar el método public String execute(List selectedObjects) throws ClientException. El mandato devuelve un resultado que se utiliza para determinar la siguiente norma de navegación para la aplicación JSF.
c. Opcional: Después de completar el mandato, el componente CommandBar evalúa el atributo **action**. El atributo **action** puede ser una serie estática o un enlace de método a un método de acción JSF con la signatura `public String Method()`. Utilice el atributo **action** para alterar temporalmente el resultado de una clase de mandato o para especificar explícitamente un resultado para las normas de navegación. No se procesará el atributo **action** si el mandato genera una excepción distinta de ErrorsInCommandException.
d. Si el atributo **commandClass** no tiene especificada una clase de mandatos, se llama inmediatamente a la acción. Por ejemplo, para el mandato de renovación, se llama a la expresión de valor JSF `#{TaskInstanceList.refreshList}` en lugar de llamar a un mandato.

Resultados

La aplicación JSF contiene ahora una página JavaServer que implementa una barra de mandatos personalizada.

Proceso de los mandatos

Utilice el componente CommandBar para añadir los botones de acción a la aplicación. El componente crea los botones para las acciones de la interfaz de usuario y gestiona los sucesos que se crean al pulsar un botón.

Estos botones desencadenan funciones que actúan sobre los objetos que la interfaz com.ibm.bpe.jsf.handler.ItemProvider devuelve, como la clase BPCListHandler o la clase BPCDetailsHandler. El componente CommandBar utiliza el proveedor de elementos definido mediante el valor del atributo **model** del código `bpe:commandbar`.

Cuando se pulsa un botón de la sección de barra de mandatos de la interfaz de usuario de la aplicación, el componente CommandBar gestiona el suceso asociado del siguiente modo:

1. El componente CommandBar identifica la implementación de la interfaz com.ibm.bpc.clientcore.Command que se especifica para el botón que ha generado el suceso.
2. Si el modelo asociado al componente CommandBar implementa la interfaz com.ibm.bpe.jsf.handler.ErrorHandler, se invoca el método clearErrorMap para eliminar los mensajes de error de sucesos anteriores.
3. Se llama al método getSelectedItems de la interfaz ItemProvider. La lista de elementos que se devuelve se pasa al método execute del mandato y se invoca el mandato.
4. El componente CommandBar determina el destino de navegación JSF (JavaServer Faces). Si no se especifica un atributo **action** en el código `bpe:commandbar`, el valor de retorno del método execute especifica el destino de navegación. Si el atributo **action** se establece en un enlace de método JSF, la serie que devuelve el método se interpreta como el destino de navegación. El atributo **action** también puede especificar un destino de navegación explícito.

Componente CommandBar: definiciones de código

El componente CommandBar de Business Process Choreographer Explorer muestra una barra con botones. Estos botones operan en el objeto en una vista de detalles o en los objetos seleccionados en una lista.

El componente CommandBar consta de los códigos de componente JSF: `bpe:commandbar` y `bpe:command`. El código `bpe:command` es un subelemento del código `bpe:commandbar`.

Clase de componente

com.ibm.bpe.jsf.component.CommandBarComponent

Sintaxis de ejemplo

```
<bpe:commandbar model="#{TaskInstanceList}">

    <bpe:command
       commandID="Work on"
       label="Work on..."
       commandClass="com.ibm.bpc.explorer.command.WorkOnTaskCommand"
       context="#{TaskInstanceDetailsBean}"/>

    <bpe:command
       commandID="Cancel"
       label="Cancel"
```

```
        commandClass="com.ibm.task.clientmodel.command.CancelClaimTaskCommand"
        context="#{TaskInstanceList}"/>

</bpe:commandbar>
```

Atributos de código

Tabla 76. Atributos de bpe:commandbar

Atributo	Necesario	Descripción
buttonStyleClass	no	Clase de estilo de hoja de estilos en cascada (CSS) que se utiliza para representar los botones de la barra de mandatos.
id	no	ID de JavaServer Faces del componente.
model	sí	Una expresión de enlace de valor a un bean gestionado que implementa la interfaz ItemProvider. Este bean gestionado suele ser la clase com.ibm.bpe.jsf.handler.BPCListHandler o com.ibm.bpe.jsf.handler.BPCDetailsHandler que utiliza el componente List o Details del mismo archivo JSP (JavaServer Pages) que el componente CommandBar.
styleClass	no	Clase de estilo CSS que se utiliza para representar la barra de mandatos.

Tabla 77. Atributos de bpe:command

Atributo	Necesario	Descripción
action	no	Un método de acción JavaServer Faces o el destino de navegación Faces que el botón del mandato va a desencadenar. El destino de navegación que devuelve la acción sobrescribe todas las otras normas de navegación. Se llama a la acción si no se genera una excepción o si el mandato genera una excepción ErrorsInCommandException.
commandClass	no	El nombre de la clase de mandato. El componente CommandBar crea una instancia de la clase y se ejecuta si se selecciona el botón de mandato.
commandID	sí	ID del mandato.
context	no	Un objeto que proporciona contexto para mandatos especificados mediante el atributo **commandClass**. El objeto de contexto se recupera cuando se accede a la barra de mandatos por primera vez.
immediate	no	Especifica cuando se desencadena el mandato. Si el valor de este atributo es `true`, el mandato se desencadena antes de procesar la entrada de la página. El valor predeterminado es `false`.
label	sí	Etiqueta del botón que se representa en la barra de mandatos.

Tabla 77. Atributos de bpe:command (continuación)

Atributo	Necesario	Descripción
rendered	no	Determina si un botón se ha representado. El valor del atributo puede ser un valor booleano o una expresión de valor.
styleClass	no	Clase de estilo CSS que se utiliza para representar el botón. Este estilo altera temporalmente el estilo de botón definido para la barra de mandatos.

Adición del componente Message a una aplicación JSF

Utilice el componente Message de Business Process Choreographer Explorer para representar objetos de datos y tipos primitivos en aplicaciones JSF (JavaServer Faces).

Acerca de esta tarea

Si el tipo de mensaje es un tipo primitivo, se representan una etiqueta y un campo de entrada. Si el tipo de mensaje es un objeto de datos, el componente atraviesa el objeto y representa los elementos en el objeto.

Procedimiento

1. Añada el componente Message al archivo JSP (JavaServer Pages).

   Añada el código `bpe:form` al código `<h:form>`. El código `bpe:form` debe incluir un atributo model.

   En el ejemplo siguiente se muestra cómo añadir un componente Message.

   ```
   <h:form>

      <h:outputText value="Input Message" />
      <bpe:form model="#{MyHandler.inputMessage}" readOnly="true" />

      <h:outputText value="Output Message" />
      <bpe:form model="#{MyHandler.outputMessage}" />

   </h:form>
   ```

   El atributo **model** del componente Message hace referencia a un objeto com.ibm.bpc.clientcore.MessageWrapper. Este objeto de envoltorio envuelve un objeto SDO (Service Data Object) o un tipo primitivo Java, por ejemplo, `int` o `boolean`. En el ejemplo, el mensaje lo suministra una propiedad del bean gestionado MyHandler.

2. Configure el bean gestionado al que se hace referencia en el código `bpe:form`.

   El siguiente ejemplo muestra cómo añadir el bean gestionado MyHandler al archivo de configuración.

   ```
   <managed-bean>
   <managed-bean-name>MyHandler</managed-bean-name>
   <managed-bean-class>com.ibm.bpe.sample.jsf.MyHandler</managed-bean-class>
   <managed-bean-scope>session</managed-bean-scope>

      <managed-property>
         <property-name>type</property-name>
         <value>com.ibm.task.clientmodel.bean.TaskInstanceBean</value>
      </managed-property>

   </managed-bean>
   ```

3. Añada el código personalizado a la aplicación JSF.

El siguiente ejemplo muestra cómo implementar mensajes de entrada y salida.

```
public class MyHandler implements ItemListener {

  private TaskInstanceBean taskBean;
  private MessageWrapper inputMessage, outputMessage

  /* Método de receptor, por ejemplo, cuando se ha seleccionado una instancia de
   * tarea en un manejador de listas.
   * Asegúrese de que el manejador se registre en faces-config.xml o
   * manualmente.
   */
  public void itemChanged(Object item) {
    if( item instanceof TaskInstanceBean ) {
        taskBean = (TaskInstanceBean) item ;
    }
  }

  /* Obtener el envoltorio de mensajes de entrada
   */
  public MessageWrapper getInputMessage() {
    try{
        inputMessage = taskBean.getInputMessageWrapper() ;
    }
    catch( Exception e ) {
        ;       //...pasar por alto errores para simplicidad
    }
    return inputMessage;
  }

  /* Obtener el envoltorio de mensajes de salida
   */
  public MessageWrapper getOutputMessage() {
    // Recuperar el mensaje del bean. Si no hay ningún mensaje, cree uno
    // si la tarea ha sido reclamada por el usuario. Asegúrese de que sólo
    // los propietarios potenciales o los propietarios pueden manejar el
    // mensaje de salida.
    try{
        outputMessage = taskBean.getOutputMessageWrapper();
        if( outputMessage == null
         && taskBean.getState() == TaskInstanceBean.STATE_CLAIMED ) {
            HumanTaskManagerService htm = getHumanTaskManagerService();
            outputMessage = new MessageWrapperImpl();
            outputMessage.setMessage(
               htm.createOutputMessage( taskBean.getID() ).getObject()
            );
         }
    }
    catch( Exception e ) {
        ;       //...pasar por alto errores para simplicidad
    }
    return outputMessage
  }
}
```

El bean gestionado MyHandler implementa la interfaz com.ibm.jsf.handler.ItemListener de manera que pueda registrarse como un receptor de elementos de manejadores de lista. Cuando el usuario pulsa un elemento de la lista, se envía una notificación al bean MyHandler sobre el elemento seleccionado en su método itemChanged(Object item). El manejador comprueba el tipo de elemento y, a continuación, almacena una referencia al objeto TaskInstanceBean asociado. Para utilizar esta interfaz, añada una entrada en la lista itemListener del manejador de lista adecuado en el archivo `faces-config.xml`.

El bean MyHandler proporciona los métodos getInputMessage y getOutputMessage. Ambos métodos devuelven un objeto MessageWrapper. Los métodos delegan las llamadas al bean de instancia de tarea al que se hace referencia. Si el bean de instancia de tarea devuelve un valor nulo, por ejemplo, porque no se haya establecido un mensaje, el manejador crea y almacena un mensaje nuevo y vacío. El componente Message muestra los mensajes proporcionados por el bean MyHandler.

Resultados

La aplicación JSF contiene ahora una página JavaServer que puede representar objetos de datos y tipos primitivos.

Componente Message: definiciones de código

El componente Message de Business Process Choreographer Explorer representa objetos `commonj.sdo.DataObject` y tipos primitivos como, por ejemplo, enteros y series, de una aplicación JSF (JavaServer Faces).

El componente Message consta del código de componente JSF: `bpe:form`.

Clase de componente

com.ibm.bpe.jsf.component.MessageComponent

Sintaxis de ejemplo

```
<bpe:form model="#{TaskInstanceDetailsBean.inputMessageWrapper}"
          simplification="true" readOnly="true"
          styleClass4table="messageData"
          styleClass4output="messageDataOutput">
</bpe:form>
```

Atributos de código

Tabla 78. Atributos de bpe:form

Atributo	Necesario	Descripción
id	no	ID de JavaServer Faces del componente.
model	sí	Expresión de enlace de valor que hace referencia a un objeto `commonj.sdo.DataObject` o un objeto `com.ibm.bpc.clientcore.MessageWrapper`.
readOnly	no	Si este atributo se establece en `true`, se representa un formato de sólo lectura. Por omisión, este atributo se establece en `false`.
simplification	no	Si se establece este atributo en `true`, se muestran las propiedades que contienen tipos simples y tienen una cardinalidad de cero o uno. Por omisión, este atributo se establece en `true`.
style4validinput	no	Estilo de hoja de estilos en cascada (CSS) para la entrada de representación que es válida.
style4invalidinput	no	Estilo de CSS para representar la entrada que no es válida.

Tabla 78. Atributos de bpe:form (continuación)

Atributo	Necesario	Descripción
styleClass4invalidInput	no	Nombre de clase de estilo CSS para representar la entrada que no es válida.
styleClass4output	no	Nombre de clase de estilo CSS para representar los elementos de salida.
styleClass4table	no	Nombre de clase del estilo de tabla CSS para representar las tablas representadas por el componente de mensajes.
styleClass4validInput	no	Nombre de clase de estilo CSS para representar la entrada que es válida.

Desarrollo de páginas JSP para mensajes de tareas y de procesos

La interfaz de Business Process Choreographer Explorer proporciona formularios de entrada y salida por omisión para visualizar y entrar datos de empresa. Puede utilizar páginas JSP para proporcionar formularios de entrada y salida personalizados.

Acerca de esta tarea

Para incluir páginas JSP (JavaServer Pages) definidos por el usuario en el cliente Web, debe especificarlos cuando modele una tarea de usuario en WebSphere Integration Developer. Por ejemplo, puede proporcionar páginas JSP para una tarea específica y sus mensajes de entrada y salida, y para un rol de usuario específico o para todos los roles de usuario. Durante la ejecución, las páginas JSP definidos por el usuario se incluyen en la interfaz de usuario para visualizar datos de salida y recopilar datos de entrada.

Los formularios personalizados no son páginas Web autocontenidas: son fragmentos de código HTML que Business Process Choreographer Explorer incorpora en un formulario HTML, por ejemplo, fragmentos para todas las etiquetas y campos de entrada de un mensaje.

Cuando se pulsa un botón en la página que contiene los formularios personalizados, la entrada se envía y se valida en Business Process Choreographer Explorer. La validación se basa en el tipo de las propiedades proporcionadas y el entorno local que se utiliza en el navegador. Si no puede validarse la entrada, se mostrará la misma página de nuevo y se proporcionará información sobre los errores de validación en el atributo de petición messageValidationErrors. La información se proporciona como una correlación de la expresión de vía de acceso XML (XPath) de las propiedades que no son válidas con las excepciones de validación que se han producido.

Para añadir formularios personalizados a Business Process Choreographer Explorer, complete los siguientes pasos utilizando WebSphere Integration Developer.

Procedimiento

1. Cree los formularios personalizados.

   Las páginas JSP definidos por el usuario para los formularios de entrada y salida utilizados en la interfaz Web necesitan acceder a los datos de mensaje.

Utilice fragmentos de Java en una JSP o el lenguaje de ejecución JSP para acceder a los datos de mensaje. Los datos de los formularios están disponibles a través del contexto de petición.

2. Asigne las páginas JSP a una tarea.

   Abra la tarea de usuario en el editor de tareas de usuario. En los valores de cliente, especifique la ubicación de las páginas JSP definidas por el usuario y el rol al que se aplica el formulario personalizado, por ejemplo, el de administrador. Los valores de cliente de Business Process Choreographer Explorer se almacenan en la plantilla de tareas. Durante la ejecución, estos valores se recuperan con la plantilla de tarea.

3. Empaquete las páginas JSP definidas por el usuario en un archivador Web (archivo WAR).

   Puede incluir el archivo WAR en el archivador de empresa con el módulo que contiene las tareas o desplegar el archivo WAR por separado. Si las JSP se despliegan independientemente, las JSP deben estar disponibles en el servidor donde se despliega Business Process Choreographer Explorer o el cliente personalizado.

   Si va a utilizar JSP personalizadas para los mensajes de proceso y de tarea, debe correlacionar los módulos Web utilizados para desplegar las JSP en los mismos servidores con los que esté correlacionado el cliente JSF personalizado.

Resultados

Los formularios personalizados se representan en Business Process Choreographer durante la ejecución.

Fragmentos JSP definidos por el usuario

Los fragmentos JSP (JavaServer Pages) definidos por el usuario se han incorporado en un código de formulario HTML. Durante la ejecución, Business Process Choreographer Explorer incluye estos fragmentos en la página representada.

El fragmento JSP definido por el usuario para el mensaje de entrada se incorpora antes del fragmento JSP para el mensaje de salida.

```
<html....>
   ...
   <form...>
     JSP de entrada (visualiza el mensaje de entrada de tarea)

     JSP de salida (visualiza el mensaje de salida de tarea)

   </form>
   ...
</html>
```

Dado que los fragmentos JSP definidos por el usuario se incorporan en un código de formulario HTML, puede añadir elementos de entrada. El nombre del elemento de entrada debe coincidir con la expresión XPath (XML Path Language) del elemento de datos. Es importante utilizar como prefijo el nombre del elemento de entrada con el valor de prefijo que se proporciona:

```
<input id="address"
       type="text"
       name="${prefix}/selectPromotionalGiftResponse/address"
       value="${messageMap['/selectPromotionalGiftResponse/address"]}
       size="60"
       align="left" />
```

El valor de prefijo se proporciona como atributo de petición. El atributo asegura que el nombre de entrada será exclusivo en el formulario que lo incluye. El prefijo lo genera Business Process Choreographer Explorer y no debe modificarse:

```
String prefix = (String)request.getAttribute("prefix");
```

Sólo se establece el elemento de prefijo si el mensaje puede editarse en el contexto dado. Los datos de salida pueden visualizarse de distintas maneras, en función del estado de la tarea de usuario. Por ejemplo, si la tarea está en estado de reclamado, los datos de salida pueden modificarse. Sin embargo, si la tarea está en estado de finalizado, los datos sólo pueden visualizarse. En el fragmento JSP, puede probar si el elemento de prefijo existe y presentar el mensaje de acuerdo a ello. La siguiente sentencia JSTL muestra cómo puede probar si se ha establecido el elemento de prefijo.

```
...
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
< c:choose>
  <c:when test="${not empty prefix}">
    <!--Modalidad de lectura/grabación-->>
  </c:when>
  <c:otherwise>
    <!--Modalidad de sólo lectura-->>
  </c:otherwise>
</c:choose>
```

Creación de los plug-in para personalizar las funciones de las tareas de usuario

Business Process Choreographer proporciona una infraestructura de gestión de sucesos para los sucesos que se producen durante el proceso de las tareas de usuario. También se proporcionan puntos de plug-in para que pueda adaptar las funciones a sus necesidades. Puede utilizar las interfaces del proveedor de servicios (SPI) para crear los plug-ins personalizados para manejar sucesos y procesar posteriormente los resultados de las consultas de personal.

Acerca de esta tarea

Puede crear plug-in para sucesos de API de tareas de usuario y sucesos de notificación de escalada. También puede crear un plug-in que procesa los resultados devueltos de una resolución de personas. Por ejemplo, en periodos de hora punta quizá desee añadir usuarios a la lista de resultados para ayudar a equilibrar la carga de trabajo.

Antes de poder utilizar los plug-ins, debe instalarlos y registrarlos. Puede registrar el plug-in para procesar posteriormente los resultados de consultas de personal con la aplicación TaskContainer. El plug-in estará disponible para todas las tareas.

Creación de manejadores de sucesos de API para Business Process Choreographer

Se produce un suceso de la API cuando un método de la API manipula una tarea de usuario. Utilice la interfaz SPI (Service Provider Interface) del plug-in de manejador de sucesos de API para gestionar los sucesos de tarea enviados por la API o los sucesos internos que tienen sucesos de API equivalentes.

Acerca de esta tarea

Complete los pasos siguientes para crear un manejador de sucesos de API.

Procedimiento

1. Escriba una clase que implemente la interfaz APIEventHandlerPlugin5 o amplíe la clase de implementación de APIEventHandler. Esta clase puede invocar los métodos de otras clases.
   - Si utiliza la interfaz APIEventHandlerPlugin5, debe implementar todos los métodos de la interfaz APIEventHandlerPlugin5 y la interfaz APIEventHandlerPlugin.
   - Si amplía la clase de implementación APIEventHandler, sobrescriba los métodos que necesite.

   Esta clase se ejecuta en el contexto de una aplicación Java Platform, Enterprise Edition (Java EE) Enterprise. Asegúrese de que esta clase y sus clases helper siguen la especificación EJB.

   Nota: Si desea llamar a la interfaz HumanTaskManagerService desde esta clase, no llame a un método que actualiza la tarea que ha producido el suceso. Esta acción podría generar datos de tarea incoherentes en la base de datos.
2. Ensamble la clase de plug-in y sus clases de ayuda en un archivo JAR.

   Puede hacer que el archivo JAR esté disponible de una de las formas siguientes:
   - Como un archivo JAR de programa de utilidad en el archivo EAR de la aplicación.
   - Como una biblioteca compartida que se instala con el archivo EAR de la aplicación.
   - Como una biblioteca compartida que se instala con la aplicación TaskContainer. En este caso, el plug-in está disponible para todas las tareas.
3. Cree un archivo de configuración de proveedor de servicio para el plug-in del directorio `META-INF/services/` del archivo JAR.

   El archivo de configuración proporciona el mecanismo para identificar y cargar el plug-in. Este archivo se ajusta a la especificación de interfaz de proveedor de servicios Java EE.

   a. Cree un archivo con el nombre `com.ibm.task.spi.`*`nombre_plug-in`*`APIEventHandlerPlugin`, donde *nombre_plug-in* es el nombre del plug-in.

      Por ejemplo, si el plug-in se denomina `Customer` e implementa la interfaz com.ibm.task.spi.APIEventHandlerPlugin5, el nombre del archivo de configuración es `com.ibm.task.spi.`**`Customer`**`APIEventHandlerPlugin`.

   b. En la primera línea del archivo que ni es una línea de comentarios (una línea que empieza con un símbolo de almohadilla (#)) ni una línea en blanco, especifique el nombre completo de la clase del plug-in que ha creado en el paso 1.

      Por ejemplo, si la clase de plug-in se denomina `MyAPIEventHandler` y está en el paquete `com.customer.plugins`, entonces la primera línea del archivo de configuración debe contener la entrada siguiente: `com.customer.plugins.MyAPIEventHandler`.

Resultados

Tiene un archivo JAR instalable que contiene un plug-in que maneja sucesos de la API y un archivo de configuración de proveedor de servicio que se puede utilizar para cargar el plug-in.

Notas: Sólo tiene una propiedad eventHandlerName disponible para registrar los manejadores de sucesos de la API y los manejadores de sucesos de notificación. Si desea utilizar un manejador de sucesos de la API y un manejador de sucesos de notificación, las implementaciones del plug-in deben tener el mismo nombre, por ejemplo `Customer`, que el nombre del manejador de sucesos de la implementación de SPI.

Puede implementar ambos plug-ins utilizando una sola clase o dos clases diferentes. En ambos casos, debe crear dos archivos en el directorio `META-INF/services/` del archivo JAR, por ejemplo, com.ibm.task.spi.CustomerNotificationEventHandlerPlugin y com.ibm.task.spi.CustomerAPIEventHandlerPlugin.

Empaquete la implementación de plug-in y las clases helper en un solo archivo JAR.

Para que un cambio sea efectivo en una implementación, sustituya el archivo JAR en la biblioteca compartida, vuelva a desplegar el archivo EAR asociado y reinicie el servidor.

Qué hacer a continuación

Ahora tiene que instalar y registrar el plug-in de modo que esté disponible para el contenedor de tareas de usuario durante la ejecución. Puede registrar los manejadores de sucesos de la API con una instancia de tarea, una plantilla de tarea o un componente de aplicación.

Manejadores de sucesos de API

Los sucesos de la API se producen cuando se modifica una tarea de usuario o cuando cambia su estado. Para gestionar estos sucesos de API, se invoca directamente el manejador de sucesos antes de modificarse la tarea (método anterior al suceso) y justo antes de que la llamada a la API devuelva el control al sistema (método posterior al suceso).

Si el método de pre-suceso genera un excepción ApplicationVetoException, no se realiza la acción de la API, se devuelve la excepción al proceso que invoca la API y se retrotrae la transacción asociada al suceso. Si el método de pre-suceso lo desencadena un suceso interno y se genera un excepción ApplicationVetoException, no se ejecuta el suceso interno como, por ejemplo, una reclamación automática, pero no se devuelve una excepción a la aplicación cliente. En este caso, se graba un mensaje informativo al archivo SystemOut.log. Si el método de la API genera una excepción durante el proceso, se captura la excepción y se pasa al método de post-suceso. Se vuelve a pasar la excepción al proceso que efectúa la llamada después de que se devuelve el método de post-suceso.

Se aplican estas reglas a los métodos anteriores al suceso:

- Los métodos anteriores al suceso reciben los parámetros del método de API o suceso interno asociado.
- Los métodos de pre-suceso pueden generar una excepción ApplicationVetoException para impedir que continúe el proceso.

Se aplican estas reglas a los métodos posteriores al suceso:

- Los métodos de post-suceso reciben los parámetros que se han proporcionado a la llamada a la API y el valor de retorno. Si la implementación del método de la API genera una excepción, el método de post-suceso también recibe la excepción.
- Los métodos de post-suceso no pueden modificar valores de retorno.
- Los métodos de post-suceso no pueden generar excepciones; las excepciones de tiempo de ejecución se registran cronológicamente e impiden que continúe el proceso.

Para implementar manejadores de sucesos de API, puede implementar la interfaz APIEventHandlerPlugin3, que amplía la interfaz APIEventHandlerPlugin, o ampliar la clase de implementación de la SPI com.ibm.task.spi.APIEventHandler por omisión. Si el manejador de sucesos se hereda de la clase de implementación por omisión, siempre implementa la versión más reciente de SPI. Si realiza la actualización a una versión más reciente de Business Process Choreographer, serán necesarios pocas modificaciones si desea explotar nuevos métodos SPI.

Si dispone de un manejador de sucesos de notificación y de un manejador de sucesos de API, estos dos manejadores deben tener el mismo nombre porque sólo puede registrar un nombre de manejador de sucesos.

Creación de manejadores de sucesos de notificación para Business Process Choreographer

Se producen los sucesos de notificación cuando se escalan las tareas de usuario. Business Process Choreographer proporciona funciones para manejar la escalada como, por ejemplo, la creación de elementos de trabajo de la escalada o el envío de correos electrónicos. Puede crear manejadores de sucesos de notificación para personalizar el modo en que se maneja la escalada.

Acerca de esta tarea

Para implementar manejadores de sucesos de notificación, puede implementar la interfaz NotificationEventHandlerPlugin, o puede ampliar la clase de implementación de SPI (Service Provider Interface) com.ibm.task.spi.NotificationEventHandler por omisión.

Complete los pasos siguientes para crear un manejador de sucesos de notificación.

Procedimiento

1. Grabe una clase que implementa la interfaz NotificationEventHandlerPlugin o amplía la clase de implementación NotificationEventHandler. Esta clase puede invocar los métodos de otras clases.

   Si utiliza la interfaz NotificationEventHandlerPlugin, debe implementar todos los métodos de interfaz. Si amplía la clase de implementación de SPI, sobrescriba los métodos necesarios.

   Esta clase se ejecuta en el contexto de una aplicación Java Platform, Enterprise Edition (Java EE) Enterprise. Asegúrese de que esta clase y sus clases helper siguen la especificación EJB.

   El plug-in se invoca con la autorización del rol EscalationUser. Este rol se define cuando se configura el contenedor de tareas de usuario.

Nota: Si desea llamar a la interfaz HumanTaskManagerService desde esta clase, no llame a un método que actualiza la tarea que ha producido el suceso. Esta acción podría generar datos de tarea incoherentes en la base de datos.

2. Ensamble la clase de plug-in y sus clases de ayuda en un archivo JAR.

   Puede hacer que el archivo JAR esté disponible de una de las formas siguientes:

   - Como un archivo JAR de programa de utilidad en el archivo EAR de la aplicación.
   - Como una biblioteca compartida que se instala con el archivo EAR de la aplicación.
   - Como una biblioteca compartida que se instala con la aplicación TaskContainer. En este caso, el plug-in está disponible para todas las tareas.

3. Ensamble la clase de plug-in y sus clases de ayuda en un archivo JAR.

   Si varias aplicaciones Java EE utilizan las clases de ayuda, puede empaquetar estas clases en un archivo JAR independiente que se registra como una biblioteca compartida.

4. Cree un archivo de configuración de proveedor de servicio para el plug-in del directorio `META-INF/services/` del archivo JAR.

   El archivo de configuración proporciona el mecanismo para identificar y cargar el plug-in. Este archivo se ajusta a la especificación de interfaz de proveedor de servicios Java EE.

   a. Cree un archivo con el nombre `com.ibm.task.spi.`*`nombre_plug-in`*`NotificationEventHandlerPlugin`, donde *nombre_plug-in* es el nombre del plug-in.

      Por ejemplo, si el plug-in se denomina `HelpDeskRequest` (nombre del manejador de sucesos) e implementa la interfaz com.ibm.task.spi.NotificationEventHandlerPlugin, el nombre del archivo de configuración será `com.ibm.task.spi.`**`HelpDeskRequest`**`NotificationEventHandlerPlugin`.

   b. En la primera línea del archivo que ni es una línea de comentarios (una línea que empieza con un símbolo de almohadilla (#)) ni una línea en blanco, especifique el nombre completo de la clase del plug-in que ha creado en el paso 1.

      Por ejemplo, si la clase de plug-in se denomina `MyEventHandler` y se encuentra en el paquete `com.customer.plugins`, entonces la primera línea del archivo de configuración debe contener la siguiente entrada: `com.customer.plugins.MyEventHandler`.

Resultados

Tiene un archivo JAR instalable que contiene un plug-in que maneja sucesos de notificación y un archivo de configuración de proveedor de servicio que se puede utilizar para cargar el plug-in. Puede registrar los manejadores de sucesos de la API con una instancia de tarea, una plantilla de tarea o un componente de aplicación.

Notas: Sólo tiene una propiedad eventHandlerName disponible para registrar los manejadores de sucesos de la API y los manejadores de sucesos de notificación. Si desea utilizar un manejador de sucesos de la API y un manejador de sucesos de notificación, las implementaciones del plug-in deben tener el mismo nombre, por ejemplo `Customer`, que el nombre del manejador de sucesos de la implementación de SPI.

Puede implementar ambos plug-ins utilizando una sola clase o dos clases diferentes. En ambos casos, debe crear dos archivos en el directorio `META-INF/services/` del archivo JAR, por ejemplo, com.ibm.task.spi.CustomerNotificationEventHandlerPlugin y com.ibm.task.spi.CustomerAPIEventHandlerPlugin.

Empaquete la implementación de plug-in y las clases helper en un solo archivo JAR.

Para que un cambio sea efectivo en una implementación, sustituya el archivo JAR en la biblioteca compartida, vuelva a desplegar el archivo EAR asociado y reinicie el servidor.

Qué hacer a continuación

Ahora tiene que instalar y registrar el plug-in de modo que esté disponible para el contenedor de tareas de usuario durante la ejecución. Puede registrar los manejadores de sucesos de la notification con una instancia de tarea, una plantilla de tarea o un componente de aplicación.

Instalación de plug-ins de manejador de sucesos de API y manejador de sucesos de notificación para tareas de usuario

Para utilizar plug-ins de manejador de sucesos de API y de manejador de sucesos de notificación, debe instalar el plug-in para que el contenedor de tareas de usuario pueda acceder a él.

Acerca de esta tarea

La manera en que se instala el plug-in depende de que el plug-in lo deban utilizar una sola aplicación Java Platform, Enterprise Edition (Java EE) o varias aplicaciones.

Complete uno de los pasos siguientes para instalar un plug-in.

Procedimiento

- Instale un plug-in para que lo utilice una aplicación Java EE individual.

  Añada el archivo JAR del plug-in al archivo EAR de la aplicación. En el editor de descriptores de despliegue de WebSphere Integration Developer, instale el archivo JAR para el plug-in como un archivo JAR de programa de utilidad de proyecto para la aplicación Java EE del módulo EJB (Enterprise JavaBeans) principal.
- Instale un plug-in para que lo utilicen varias aplicaciones Java EE.

  Coloque el archivo JAR en una biblioteca compartida de WebSphere Application Server y asocie la biblioteca a las aplicaciones que tienen que acceder al plug-in. Para que el archivo JAR esté disponible en un entorno de despliegue en red, distribuya manualmente el archivo JAR en cada nodo que aloje un miembro de servidor o de clúster en cualquiera de las aplicaciones desplegadas. Puede utilizar el ámbito del destino de despliegue, que es el servidor o el clúster en el que se despliegan las aplicaciones, o el ámbito de célula. Tenga en cuenta que es en ese momento cuando las clases de plug-in son visibles a lo largo del ámbito de despliegue seleccionado.

Qué hacer a continuación

Ahora puede registrar el plug-in.

Registro de los plug-ins del manejador de sucesos de API y del manejador de sucesos de notificación con plantillas de tarea, modelos de tarea y tareas

Puede registrar plug-ins para los manejadores de sucesos de API y manejadores de sucesos de notificación con tareas, plantillas de tarea y modelos de tarea en momentos diferentes: cuando cree una tarea ad-hoc, actualice una tarea existente, cree un modelo de tarea ad-hoc o defina una plantilla de tarea.

Acerca de esta tarea

Puede registrar los plug-ins para los manejadores de sucesos de API y los manejadores de sucesos de notificación con tareas en los niveles siguientes:

Plantilla de tarea
Todas las tareas creadas utilizando la plantilla utilizan los mismos manejadores

Modelo de tarea ad-hoc
Las tareas creadas utilizando el modelo utilizan los mismos manejadores

Tarea ad-hoc
La tarea creada utiliza los manejadores especificados

Tarea existente
La tarea utiliza los manejadores especificados

Puede registrar un plug-in de uno de los modos siguientes.

Procedimiento

- Para las plantillas de tarea que tienen un modelo en WebSphere Integration Developer, especifique el plug-in en el modelo de tarea.
- Para las tareas ad-hoc o los modelos de tarea ad-hoc, especifique el plug-in al crear la tarea o el modelo de tarea.

  Utilice el método setEventHandlerName de la clase TTask para registrar el nombre del manejador de sucesos.
- Cambie el manejador de sucesos para una instancia de tarea en el tiempo de ejecución.

  Utilice el método update(Task task) para utilizar un manejador de tareas diferente para una instancia de tarea durante la ejecución. El usuario que realice la llamada debe tener autorización de administrador para actualizar esta propiedad.

Utilización de un plug-in para el proceso posterior de los resultados de consultas de personas

La resolución de personas en Business Process Choreographer devuelve una lista de los usuarios asignados a un rol específico, por ejemplo, los propietarios potenciales de una tarea. Puede crear un plug-in que cambie los resultados de las consultas de personas devueltas por la resolución de personas. Por ejemplo, para mejorar el equilibrio de carga de trabajo, podría eliminar los usuarios del resultado de la consulta que ya tienen una alta carga de trabajo.

Acerca de esta tarea

Para modificar los resultados devueltos por la asignación de personas y la sustitución de personas, debe escribir una clase que implemente la interfaz de plug-in, ensamblar un archivo JAR para el plug-in y, a continuación, instalarlo y activarlo.

Complete los pasos siguientes para crear un plug-in para el proceso posterior de los resultados de consulta de personas.

Procedimiento

1. Implemente el plug-in del proceso posterior del resultado de la consulta de personas Escriba una clase que implemente la interfaz StaffQueryResultPostProcessorPlugin o la interfaz StaffQueryResultPostProcessorPlugin2.
2. Cree un archivo JAR instalable.
   a. Ensamble la clase de plug-in y sus clases de ayuda en un archivo JAR.
   b. Cree un archivo de configuración de proveedor de servicio para el plug-in del directorio `META-INF/services/` del archivo JAR. El archivo de configuración proporciona el mecanismo para identificar y cargar el plug-in. Este archivo debe ser compatible con la especificación de la interfaz del proveedor de servicios Java EE.
      1) En un editor de texto, cree un archivo de configuración del proveedor de servicios con el nombre `com.ibm.task.spi.`*`nombre_plug-in`*`StaffQueryResultPostProcessorPlugin`, donde *nombre_plug-in* es el nombre del plug-in. El nombre del archivo de configuración no depende del nombre de la interfaz que ha implementado. Por ejemplo, si el plug-in se llama `MyHandler` e implementa la interfaz com.ibm.task.spi.StaffQueryResultPostProcessorPlugin2, el nombre del archivo de configuración es `com.ibm.task.spi.`**`MyHandler`**`StaffQueryResultPostProcessorPlugin`.
      2) En la primera línea del archivo que no sea una línea de comentario (una línea que empieza con un signo de número (#)) ni una línea en blanco, especifique el nombre completo de la clase del plug-in que ha creado en el paso 1. Por ejemplo, si la clase de plug-in se llama `StaffPostProcessor` y está en el paquete `com.customer.plugins`, la primera línea del archivo de configuración debe contener la siguiente entrada: `com.customer.plugins.StaffPostProcessor`.

   Tiene un archivo JAR instalable que contiene un plug-in que procese posteriormente los resultados de la consulta de personas y un archivo de configuración de proveedor de servicios que se puede utilizar para cargar el plug-in.
3. Instale el archivo JAR en una biblioteca compartida en el servidor de aplicaciones y asócielo a la aplicación Human Task Manager.
   a. Defina una biblioteca compartida de WebSphere Application Server para el plug-in en el ámbito del servidor o clúster en el que está configurado Business Process Choreographer.
   b. Asocie la biblioteca compartida con la aplicación `TaskContainer`.
   c. Deje el archivo JAR de plug-in disponible para cada instalación de WebSphere Process Server afectada que contenga un servidor o un miembro de clúster.
4. Configure el Human Task Manager para utilizar el plug-in.

a. En la consola administrativa, vaya a la página Propiedades personalizadas de Human Task Manager.

   Pulse o **Servidores** → **Clústeres** → **Clústeres de WebSphere Application Server** → *nombre_clúster* o **Servidores** → **Tipos de servidor** → **WebSphere Application Servers** → *nombre_servidor*, en la pestaña **Configuración**, en la sección **Business Integration**, expanda **Business Process Choreographer**, y pulse **Human Task Manager**. En **Propiedades adicionales**, seleccione **Propiedades personalizadas**.

b. Añada una propiedad personalizada con el nombre **Staff.PostProcessorPlugin**, y un valor del nombre que le ha dado al plug-in, por ejemplo, `MyHandler`.

Ahora el plug-in está disponible para el proceso posterior de los resultados de consultas de personal.

5. Reinicie el servidor para activar el plug-in. El plug-in de proceso posterior se invoca después de que se hayan ejecutado tanto la asignación de personas como la sustitución de personas.

   Nota: Si modifica el plug-in, debe sustituir el archivo JAR en la biblioteca compartida y reiniciar el servidor.

Parte 2. Despliegue de aplicaciones

Capítulo 5. Visión general de la preparación e instalación de módulos

La instalación de módulos (también conocida como despliegue) activa los módulos en un entorno de prueba o de producción. Esta visión general describe brevemente los entornos de prueba y de producción y algunos de los pasos implicados en la instalación de módulos.

Nota: El proceso para instalar aplicaciones en un entorno de producción es similar al proceso descrito en el apartado "Desarrollo y despliegue de aplicaciones" del centro de información de WebSphere Application Server Network Deployment. Si no está familiarizado con estos temas, revíselos antes.

Antes de instalar un módulo en un entorno de producción, verifique siempre los cambios en un servidor de prueba. Para instalar módulos en un entorno de prueba, utilice WebSphere Integration Developer (vea el centro de información de WebSphere Integration Developer. Para instalar módulos en un entorno de producción, utilice WebSphere Process Server.

Este tema describe los conceptos y tareas necesarios para preparar e instalar módulos en un entorno de producción. Otros temas describen los archivos que alojan los objetos que el módulo utiliza y le ayudan a mover el módulo desde el entorno de prueba al entorno de producción. Es importante entender estos archivos y lo que contienen para que pueda estar seguro de que ha instalado correctamente los módulos.

Visión general de bibliotecas y archivos JAR

A menudo, los módulos utilizan artefactos que se encuentran en bibliotecas, que son proyectos especiales en WebSphere Integration Developer utilizados para almacenar recursos compartidos. Durante el despliegue, las bibliotecas de WebSphere Integration Developer se transforman en archivos JAR de programa de utilidad y se empaquetan en las aplicaciones que se van a ejecutar.

Mientras se desarrolla un módulo, es posible identificar determinados recursos o componentes que podrían ser utilizados por otros módulos. Estos artefactos se pueden compartir utilizando una biblioteca.

¿Qué es una biblioteca?

Una biblioteca es un proyecto especial en WebSphere Integration Developer que se utiliza para el desarrollo, la gestión de versiones y la organización de recursos compartidos como, por ejemplo, aquellos recursos que normalmente están compartidos entre módulos. Sólo se puede crear y almacenar en una biblioteca un subconjunto de tipos de artefacto, incluidos:

- Descriptores de interfaces o servicios Web (archivos con la extensión .wsdl)
- Definiciones de esquema XML de objetos de negocio (archivos con la extensión .xsd)
- Correlaciones de objetos de negocio (archivos con la extensión .map)
- Definiciones de relación y de rol (archivos con la extensión .rel y .rol)

Durante el despliegue, estas bibliotecas de WebSphere Integration Developer se transforman en archivos JAR de programa de utilidad en las aplicaciones que se van a ejecutar.

Cuando un módulo necesita un artefacto, el servidor localiza el artefacto de la classpath EAR y carga el artefacto, si todavía no se ha cargado, en la memoria. La Figura 78 muestra cómo una aplicación contiene componentes y bibliotecas relacionadas.

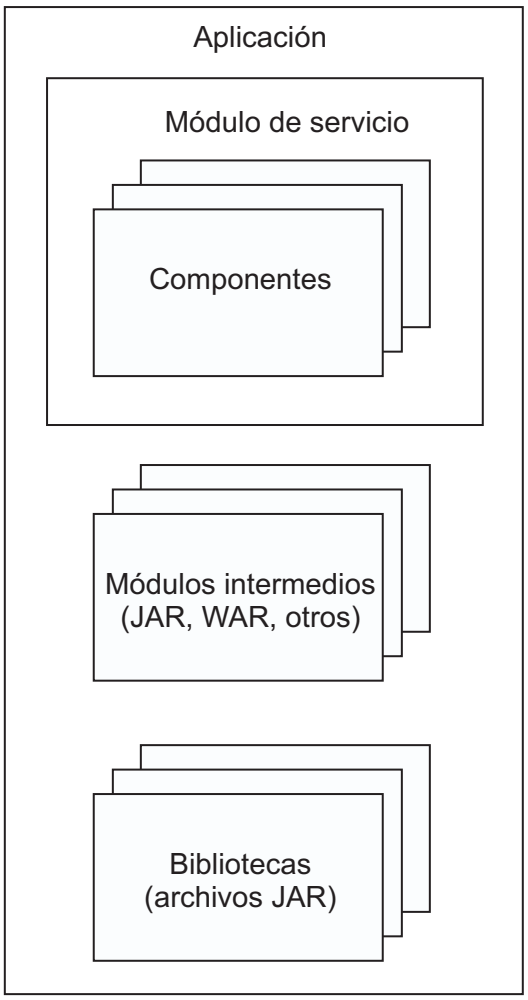


Figura 78. Relaciones entre módulo, componente y biblioteca

¿Qué son los archivos JAR, RAR y WAR?

Hay varios archivos que pueden contener componentes de un módulo. Estos archivos están completamente descritos en la especificación Java Platform Enterprise Edition. Pueden encontrarse detalles sobre los archivos JAR en la >especificación JAR.

En WebSphere Process Server, un archivo JAR también contiene una aplicación, que es la versión ensamblada del módulo con todas las referencias e interfaces de soporte a otros componentes utilizados por el módulo. Para instalar por completo la aplicación, necesita este archivo JAR, cualquier otro JAR dependiente, archivos WAR (archivadores de servicios web) y RAR (archivadores de recursos), bibliotecas intermedias, archivos JAR EJB (Enterprise Java Beans - EJB) y cualquier otro

archivo. Cree un archivo EAR instalable utilizando el mandato serviceDeploy.

Convenios de denominación para módulos intermedios.

En la biblioteca, hay requisitos para los nombres de los módulos intermedios. Estos nombres son exclusivos para un módulo específico. Indique cualquier otro módulo necesario para desplegar la aplicación, de manera que no se produzcan conflictos con los nombres de los módulos intermedios. Para un módulo denominado *myService*, los nombres de los módulos intermedios son:
- *myService*App
- *myService*Web

Nota: Los módulos intermedios *myService*EJB y *myService*EJBClient ya no son creado por serviceDeploy. Sin embargo, estos nombres de archivo no se deben utilizar, porque podrían ser suprimidos por el mandato serviceDeploy.

Consideraciones al utilizar bibliotecas

El uso de bibliotecas proporciona coherencia de objetos de negocio y coherencia de proceso entre módulos, porque cada módulo llamante tiene su propia copia de un componente específico. Para evitar incoherencias y anomalías, es importante asegurarse de que los cambios en componentes y objetos de negocio utilizados por módulos llamantes se coordinen con todos los módulos llamantes. Actualice los módulos llamantes mediante las acciones siguientes:

1. Copiar el módulo y la copia más reciente de las bibliotecas en el servidor de producción
2. Reconstruir el archivo EAR instalable mediante el mandato serviceDeploy
3. Detener la aplicación en ejecución que contiene el módulo llamante y volver a instalarlo
4. Reiniciar la aplicación que contiene el módulo llamante

Visión general del archivo EAR

Un archivo EAR es un elemento crítico en el despliegue de una aplicación de servicio en un servidor de producción.

Un archivo EAR (de archivo de empresa) es un archivo comprimido que contiene las bibliotecas, beans de empresa y archivos JAR que la aplicación requiere para el despliegue.

Se crea un archivo JAR al exportar los módulos de aplicación de WebSphere Integration Developer. Utilice este archivo JAR y otras bibliotecas u objetos de artefactos como entrada en el proceso de instalación. El mandato serviceDeploy crea un archivo EAR de los archivos de entrada que contiene las descripciones de componente y el código Java que conforma la aplicación.

Preparación para desplegar en un servidor

Después de desarrollar y probar un módulo, debe exportar el módulo desde un sistema de prueba y traerlo a un entorno de producción para su despliegue. Para instalar una aplicación, también debe conocer las vías de acceso necesarias al exportar el módulo y las bibliotecas que el módulo necesite.

Antes de empezar

Antes de iniciar esta tarea, debe haber desarrollado y probado los módulos en un servidor de prueba y haber resuelto los problemas y cuestiones de rendimiento.

Importante: Para que no se sustituya una aplicación o un módulo que ya se esté ejecutando en un entorno de despliegue, asegúrese de que el nombre del módulo o la aplicación sea distinto de los ya instalados.

Acerca de esta tarea

Esta tarea verifica que todas las piezas necesarias de una aplicación estén disponibles y empaquetadas en los archivos correctos para llevarlas al servidor de producción.

Nota: También puede exportar un archivo EAR (Enterprise ARchive) desde WebSphere Integration Developer e instalarlo directamente en WebSphere Process Server.

Importante: Si los servicios de un componente utilizan una base de datos, instale la aplicación en un servidor conectado directamente a la base de datos.

Procedimiento

1. Localice la carpeta que contiene los componentes correspondientes al módulo que va a desplegar.

   La carpeta de componentes debe denominarse *nombre-módulo* con un archivo denominado *módulo*`.module`, el módulo base.
2. Verifique que todos los componentes contenidos en el módulo están en subcarpetas de componente debajo de la carpeta de módulo.

   Para su facilidad de uso, indique la subcarpeta similar a *módulo*`/`*componente*.
3. Verifique que todos los archivos que abarcan cada componente están contenidos en la subcarpeta de componentes correspondiente y tiene un nombre similar a *nombre-archivo-componente*`.component`.

   Los archivos de componente contienen las definiciones para cada componente individual en el módulo.
4. Verifique que todos los componentes y artefactos están en las subcarpetas del componente que los necesita.

   En este paso se asegura que las referencias a artefactos que un componente necesita estén disponibles. Los nombres de componentes no deben estar en conflicto con los nombres que el mandato serviceDeploy utiliza para los módulos intermedios. Consulte el apartado Convenios de denominación para módulos intermedios
5. Verifique que un archivo de referencias, *módulo*`.references`, existe en la carpeta de módulo del paso 1.

   El archivo de referencias define las referencias y las interfaces en el módulo.
6. Verifique que un archivo de cables, *módulo*`.wires`, existe en la carpeta de componentes.

   El archivo de cables completa las conexiones entre las referencias y las interfaces en el módulo.
7. Verifique que un archivo de manifiesto, *módulo*`.manifest`, existe en la carpeta de componentes.

El manifiesto lista el módulo y todos los componentes que abarcan el módulo. También contiene una sentencia class path de manera que el mandato serviceDeploy pueda localizar los demás módulos que el módulo necesite.

8. Cree un archivo comprimido o JAR del módulo como entrada al mandato serviceDeploy que utilizará para preparar el módulo para la instalación en el servidor de producción.

Ejemplo de estructura de carpetas para el módulo MyValue antes del despliegue

El ejemplo siguiente ilustra la estructura de directorios del módulo MyValueModule, que se compone de los componentes MyValue, CustomerInfo y StockQuote.

```
MyValueModule
   MyValueModule.manifest
   MyValueModule.references
   MyValueModule.wiring
   MyValueClient.jsp
process/myvalue
   MyValue.component
   MyValue.java
   MyValueImpl.java
service/customerinfo
   CustomerInfo.component
   CustomerInfo.java
   Customer.java
   CustomerInfoImpl.java
service/stockquote
   StockQuote.component
   StockQuote.java
   StockQuoteAsynch.java
   StockQuoteCallback.java
   StockQuoteImpl.java
```

Qué hacer a continuación

Instale el módulo en los sistemas de producción como se describe en el apartado Instalación de un módulo en un servidor de producción.

Consideraciones sobre la instalación de aplicaciones de servicio en clústeres

La instalación de una aplicación de servicio en un clúster pone requisitos adicionales en el usuario. Es importante que tenga en cuenta estas consideraciones al instalar aplicaciones de servicio en un clúster.

Los clústeres pueden proporcionar muchas ventajas al entorno de proceso, ya que proporcionan economías de escala que le ayudarán a equilibrar la carga de trabajo de peticiones entre los servidores y proporcionan un nivel de disponibilidad a los clientes de las aplicaciones. Tenga en cuenta lo siguiente antes de instalar una aplicación que contenga servicios en un clúster:

- ¿Los usuarios de la aplicación necesitarán la potencia de proceso y la disponibilidad que proporciona la agrupación en clúster?

  Si la respuesta es afirmativa, la solución correcta es la agrupación en clúster. La agrupación en clúster aumentará la disponibilidad y la capacidad de las aplicaciones.
- ¿Está el clúster preparado correctamente para las aplicaciones de servicio?

Debe configurar el clúster correctamente antes de instalar e iniciar la primera aplicación que contiene un servicio. Una anomalía al configurar correctamente el clúster impide que las aplicaciones procesen peticiones de forma correcta.

- ¿Tiene el clúster una copia de seguridad?

  También debe instalar la aplicación en el clúster de copia de seguridad.

Capítulo 6. Instalación de aplicaciones de procesos de empresa y tareas de usuario

Puede distribuir módulos SCA (Service Component Architecture) que contengan procesos empresarialesy/o tareas de usuario para destinos de despliegue. Un destino de despliegue puede ser un servidor o un clúster.

Antes de empezar

Verifique que Business Flow Manager y Human Task Manager estén instalados y configurados para cada servidor de aplicaciones o clúster donde desee instalar la aplicación.

Acerca de esta tarea

Puede instalar aplicaciones de procesos empresarialesy tareas desde la consola administrativa, desde la línea de mandatos o ejecutando un script administrativo.

Resultados

Después de que se instala una aplicación de proceso empresarialo de tarea de usuario, todas las plantillas de proceso empresarialy de tarea de usuario se colocan en el estado de inicio. Puede crear instancias de proceso e instancias de tareas a partir de estas plantillas.

Qué hacer a continuación

Antes de que pueda crear instancias de proceso o de tarea, debe iniciar la aplicación.

Cómo las aplicaciones de procesos de empresa y de tareas de usuario se instalan en un entorno de Network Deployment

Cuando las plantillas de proceso o las plantillas de tarea de usuario se instalan en un entorno de Network Deployment, la instalación de la aplicación efectúa automáticamente las acciones siguientes.

La aplicación se instala por etapas. Cada etapa debe completarse satisfactoriamente para que la etapa siguiente pueda empezar.

1. La instalación de la aplicación se inicia en el gestor de despliegue.

   Durante esta etapa, las plantillas de proceso de empresa o las plantillas de tarea de usuario se configuran en el repositorio de configuración de WebSphere. La aplicación también se valida. Si se producen errores, estos aparecen en el archivo System.out, en el archivo System.err o como entradas FFDC en el gestor de despliegue.
2. La instalación de la aplicación continúa en el agente de nodo.

   Durante esta etapa, la instalación de la aplicación se desencadena en una instancia del servidor de aplicaciones. Esta instancia del servidor de aplicaciones es el destino de despliegue (o forma parte del mismo). Si el destino de despliegue es un clúster con diversos miembros, la instancia de servidor se elige arbitrariamente entre los miembros de este clúster. Si se

producen errores durante esta etapa, estos aparecen en el archivo SystemOut.log, en el archivo SystemErr.log o como entradas FFDC en el agente de nodo.

3. La aplicación se ejecuta en la instancia de servidor.

   Durante esta etapa, las plantillas de proceso o las plantillas de usuario se despliegan en la base de datos de Business Process Choreographer en el destino de despliegue. Si se producen errores, estos aparecen en el archivo System.out, en el archivo SystemErr.log o como entradas FFDC en esta instancia de servidor.

Despliegue de los procesos empresariales y las tareas de usuario

Utilice WebSphere Integration Developer o serviceDeploy para empaquetar componentes de proceso o componentes de tarea en un archivo de aplicación empresarial (EAR). Cada versión nueva de un modelo que se vaya a desplegar debe estar empaquetada en una nueva aplicación de empresa.

Cuando instale una aplicación de empresa que contenga procesos empresariales o tareas de usuario, estos se almacenan como plantillas de proceso o plantillas de tarea de usuario, según corresponda, en la base de datos de Business Process Choreographer. Por omisión, las plantillas recién instaladas están en el estado de iniciado. Sin embargo, la aplicación de empresa recién instalada está en el estado de detenido. Todas las aplicaciones de empresa se pueden iniciar y detener individualmente.

Puede desplegar muchas versiones diferentes de una plantilla de proceso o de una plantilla de tarea, cada una en una aplicación de empresa diferente. Las versiones se diferencian por sus fechas iniciales válidas. Cuando se instala una nueva aplicación de empresa, la versión de la plantilla que se instala viene determinada de la manera siguiente:

- Si el nombre de la plantilla y el espacio de nombres de destino todavía no existen, se instala una plantilla nueva.
- Si el nombre de la plantilla y el espacio de nombres de destino son los mismos que los de una plantilla existente, pero la fecha de válido-desde es diferente, se instala una versión nueva de una plantilla existente.

Nota: El nombre de la plantilla se deriva del nombre del componente y no del proceso empresarial ni de la tarea de usuario.

Si no se especifica una fecha de válido-desde, la fecha se determinará de la manera siguiente:

- Si utiliza WebSphere Integration Developer, la fecha de válido-desde es la fecha en la se modeló la tarea de usuario o el proceso empresarial.
- Si utiliza el despliegue de servicios, la fecha de válido-desde es la fecha en la que se ejecutó el mandato serviceDeploy. Solo las tareas de colaboración obtienen la fecha en la que se instaló la aplicación como fecha de válido-desde.

Instalación interactiva de aplicaciones de procesos de empresa y tareas de usuario

Puede instalar interactivamente un aplicación durante la ejecución mediante la herramienta wsadmin y el script installInteractive. Puede utilizar este script para cambiar los valores que no se pueden modificar si utiliza la consola administrativa para instalar la aplicación.

Acerca de esta tarea

Efectúe los pasos siguientes para instalar las aplicaciones de proceso de empresa de forma interactiva.

Procedimiento

1. Inicie la herramienta wsadmin.

   En el directorio *raíz_perfil*/bin, escriba wsadmin.
2. Instale la aplicación.

   En el indicador de línea de mandatos wsadmin, escriba el mandato siguiente:

   ```
   $AdminApp installInteractive application.ear
   ```

   donde *application.ear* es el nombre cualificado del archivo EAR (Enterprise Archive) que contiene la aplicación de proceso. A través de una serie de tareas verá indicadores donde podrá cambiar los valores de la aplicación.
3. Guarde los cambios de configuración.

   En el indicador de línea de mandatos wsadmin, escriba el mandato siguiente:

   ```
   $AdminConfig save
   ```

   Debe guardar los cambios para transferir las actualizaciones al depósito de configuración maestro. Si un proceso de script finaliza y no ha guardado los cambios, se descartan los cambios.

Configuración de los orígenes de datos y de las referencias del conjunto de las aplicaciones de procesos

Es posible que tenga configurar las aplicaciones de proceso que ejecutan sentencias SQL para la infraestructura de base de datos específica. Estas sentencias SQL pueden proceder de las actividades del servicio de información o pueden ser sentencias que se ejecutan durante la instalación del proceso o el arranque de la instancia.

Acerca de esta tarea

Cuando instala la aplicación, puede especificar los siguientes tipos de orígenes de datos:

- Orígenes de datos que ejecutan sentencias SQL durante la instalación del proceso
- Orígenes de datos que ejecutan sentencias SQL durante el arranque de una instancia de proceso
- Orígenes de datos que ejecutan actividades de snippets SQL

El origen de datos necesario para ejecutar una actividad de snippet SQL se define en una variable BPEL de tipo tDataSource. Los nombres de esquema y tabla de base de datos son necesarios para cualquier actividad de snippet SQL definida en las variables BPEL de tipo tSetReference. Puede configurar los valores iniciales de estas dos variables.

Puede utilizar la herramienta wsadmin para especificar los orígenes de datos.

Procedimiento

1. Instale la aplicación de proceso interactivamente utilizando la herramienta wsadmin.
2. Recorra las tareas hasta que encuentre las tareas para actualizar los orígenes de datos y las referencias del conjunto.

Configure estos valores para su entorno. El ejemplo siguiente muestra los valores que puede modificar para cada una de estas tareas.

3. Guarde los cambios.

Ejemplo: Actualización de los orígenes de datos y de las referencias del conjunto, mediante la herramienta wsadmin

En la tarea **Actualización de orígenes de datos**, puede cambiar los valores de los orígenes de datos para los valores de variables iniciales y las sentencias que se utilizan durante la instancia del proceso o durante el inicio del proceso. En la tarea **Actualizando referencias del conjunto**, puede configurar los valores relacionados con el esquema de base de datos y los nombres de las tablas.

```
Tarea[24]: Actualización de orígenes de datos

//Cambiar los valores de los orígenes de datos para los valores de variables
//iniciales durante el inicio del proceso

Nombre de proceso:  Test
// Nombre de la plantilla de proceso
Inicio del proceso o tiempo de instalación:  Inicio del proceso
// Indica si se evalúa el valor especificado
//durante el inicio del proceso o la instalación del proceso
Sentencia o variable:  Variable
// Indica que se ha de modificar una variable de origen de datos
Nombre de origen de datos:  MyDataSource
// Nombre de la variable
Nombre JNDI:[jdbc/sample]:jdbc/newName
// Establece el nombre JNDI en jdbc/newName
Tarea[25]: Actualizando referencias del conjunto

// Cambio de los valores de referencia del conjunto que se utilizan como valores
// iniciales para variables BPEL

Nombre de proceso:  Test
// Nombre de la plantilla de proceso
Variable:  SetRef
// El nombre de la variable BPEL
Nombre JNDI:[jdbc/sample]:jdbc/newName
// Establece el nombre JNDI del origen de datos de la referencia del
// conjunto en jdbc/newName
Nombre de esquema: [IISAMPLE]
// El nombre del esquema de base de datos
Prefijo de esquema:  []:
// El prefijo del nombre del esquema.
// Este valor sólo se aplica si se genera el nombre de esquema.
Nombre de tabla: [SETREFTAB]: NEWTABLE
// Establece el nombre de la tabla de base de datos en NEWTABLE
Prefijo de tabla:  []:
// El prefijo del nombre de tabla.
// Este valor sólo se aplica si se genera el nombre de prefijo.
```

Desinstalación de aplicaciones de procesos empresariales y tareas de usuario utilizando la consola administrativa.

Puede utilizar la consola administrativa para desinstalar aplicaciones que contienen procesos empresariales o tareas de usuario.

Antes de empezar

Para desinstalar una aplicación que contenga procesos empresariales o tareas de usuario, deben aplicarse las condiciones siguientes:

- Si la aplicación está instalada en un servidor autónomo, el servidor debe estar en ejecución y debe tener acceso a la base de datos Business Process Choreographer.
- Si la aplicación está instalada en un clúster, el gestor de despliegue y al menos un miembro de clúster deben estar en ejecución. El miembro de clúster debe tener acceso a la base de datos Business Process Choreographer.
- Si la aplicación está instalada en un servidor gestionado, el gestor de despliegue y el servidor gestionado deben estar en ejecución. El servidor debe tener acceso a la base de datos Business Process Choreographer.
- No existen instancias de plantillas de procesos empresariales o de tareas de usuarios en ningún estado o tiene un servidor autónomo que se ejecuta en modalidad de desarrollo.
- Si una instancia de proceso se ha migrado a una versión más nueva del proceso pero está esperando una invocación de servicio para responder, la aplicación que contiene la versión anterior no se puede desinstalar hasta que se reciba la respuesta. En todos los demás casos, las instancias que se han migrado se consideran instancias de la nueva versión y la aplicación que contiene la versión más antigua del proceso se puede desinstalar.

Acerca de esta tarea

Para desinstalar una aplicación empresarial que contiene procesos empresariales o tareas de usuario, realice las acciones siguientes:

Procedimiento

1. En la consola administrativa, pulse **Aplicaciones** → **Tipos de aplicación** → **WebSphere Enterprise Applications**.
2. Seleccione la aplicación que desea desinstalar y pulse **Detener**.

   Este paso produce un error si todavía existe alguna instancia de proceso o de tarea en la aplicación. Puede utilizar Business Process Choreographer Explorer para suprimir las instancias o la opción **-force** del script administrativo bpcTamplates.jacl para detener o suprimir estas instancias antes de que se desinstale la aplicación.
3. Seleccione la aplicación que desea desinstalar y pulse **Desinstalar**.
4. Pulse **Guardar** para guardar los cambios.

Resultados

Se desinstalará la aplicación.

Tareas relacionadas

"Desinstalación de aplicaciones de procesos empresariales y tareas de usuario utilizando el mandato administrativo."
La utilización del script bpcTemplates.jacl proporciona una alternativa a la consola administrativa para la desinstalación de aplicaciones que contienen procesos empresariales o tareas de usuario.

Desinstalación de aplicaciones de procesos empresariales y tareas de usuario utilizando el mandato administrativo.

La utilización del script bpcTemplates.jacl proporciona una alternativa a la consola administrativa para la desinstalación de aplicaciones que contienen procesos empresariales o tareas de usuario.

Antes de empezar

Para desinstalar una aplicación que contenga procesos empresariales o tareas de usuario, deben aplicarse las condiciones siguientes:

- Si la aplicación está instalada en un servidor autónomo, el servidor debe estar en ejecución y debe tener acceso a la base de datos Business Process Choreographer.
- Si la aplicación está instalada en un clúster, el gestor de despliegue y al menos un miembro de clúster deben estar en ejecución. El miembro de clúster debe tener acceso a la base de datos Business Process Choreographer.
- Si la aplicación está instalada en un servidor gestionado, el gestor de despliegue y el servidor gestionado deben estar en ejecución. El servidor debe tener acceso a la base de datos Business Process Choreographer.
- Asegúrese de que el proceso servidor con el que se conecta el cliente administrativo esté en ejecución. Para asegurarse de que el cliente administrativo conecte automáticamente con el proceso servidor, no utilice la opción `-conntype NONE` de opción de mandato.
- Si la seguridad administrativa de WebSphere está habilitada y el ID de usuario no tiene autorización de operador o administrador, incluya las opciones de wsadmin `-user` y `-password` para especificar un ID de usuario que tenga autorización de operador o administrador. La opción `-uninstall` necesita autorización de operador y la opción `-force` necesita autorización de administrador.
- Una o varias de las situaciones siguientes son verdaderas:
  - No hay instancias de plantillas de proceso de empresa o de tarea de usuario presentes en ningún estado.
  - Tiene la intención de utilizar la opción **-force**.
  - Tiene un servidor autónomo que se ejecuta en modalidad de desarrollo.
- Si una instancia de proceso se ha migrado a una versión más nueva del proceso pero está esperando una invocación de servicio para responder, la aplicación que contiene la versión anterior no se puede desinstalar hasta que se reciba la respuesta. En todos los demás casos, las instancias que se han migrado se consideran instancias de la nueva versión y la aplicación que contiene la versión más antigua del proceso se puede desinstalar.

Acerca de esta tarea

En los pasos siguientes se describe cómo utilizar el script bpcTemplates.jacl para desinstalar aplicaciones que contienen plantillas de procesos empresariales o plantillas de tareas de usuario.

Procedimiento

1. Si todavía hay instancias de proceso o de tarea asociadas con las plantillas de la aplicación que desea desinstalar, realice una de las acciones siguientes, o ambas:
   - Utilice Business Process Choreographer Explorer para suprimir las instancias.
   - En los casos en los que está seguro de que ningún otro proceso empresarial depende de las plantillas de proceso que están definidas en la aplicación que desea desinstalar, puede utilizar la opción **-force**.

     PRECAUCIÓN:
     Si utiliza el script con esta opción, se suprimen las instancias que están asociadas con las plantillas y todos los datos que están asociados con cualquier instancia en ejecución, se detienen las plantillas y se desinstala la aplicación en un solo paso. Utilice esta opción con extrema precaución.

2. Vaya al subdirectorio de Business Process Choreographer donde se encuentran los scripts administrativos. Entre el mandato siguiente:

   ```
   cd raíz_instalación/ProcessChoreographer/admin
   ```

   Linux UNIX En las plataformas Linux y UNIX, entre el mandato siguiente:

   ```
   cd raíz_instalación/ProcessChoreographer/admin
   ```

   En la plataforma i5/OS, entre el mandato siguiente:

   ```
   cd raíz_instalación/ProcessChoreographer/admin
   ```

   Windows En las plataformas Windows, entre el siguiente mandato:

   ```
   cd raíz_instalación\ProcessChoreographer\admin
   ```

3. Detenga las plantillas y desinstale la aplicación correspondiente.

   Windows En las plataformas Windows, entre:

   ```
   raíz_instalación\bin\wsadmin -f bpcTemplates.jacl
                                -uninstall nombre_aplicación
                               [-force]
   ```

   Linux UNIX En las plataformas Linux y UNIX, entre:

   ```
   raíz_instalación/bin/wsadmin -f bpcTemplates.jacl
                                -uninstall nombre_aplicación
                               [-force]
   ```

   Donde:

   -uninstall *nombre_aplicación*
   Especifica el nombre de la aplicación que se debe desinstalar.

   -force
   Esta opción hace que las instancias en ejecución se detengan y se supriman antes de que se desinstale la aplicación. Utilice esta opción con precaución porque también suprime todos los datos asociados con las instancias en ejecución.

Resultados

Se desinstalará la aplicación.

Tareas relacionadas

"Desinstalación de aplicaciones de procesos empresariales y tareas de usuario utilizando la consola administrativa." en la página 520
Puede utilizar la consola administrativa para desinstalar aplicaciones que contienen procesos empresariales o tareas de usuario.

Capítulo 7. Adaptadores y su instalación

Los adaptadores permiten que la aplicación se comunique con otros componentes del sistema de información de empresa.

El proceso que se utiliza para instalar adaptadores está descrito en Configuración y uso de adaptadores en el centro de información de WebSphere Integration Developer.

Capítulo 8. Resolución de problemas de un despliegue anómalo

Este tema describe los pasos que deben realizarse para determinar la causa de un problema al desplegar una aplicación. También presenta algunas soluciones posibles.

Antes de empezar

Este tema da por supuestas las afirmaciones siguientes:
- El usuario tiene una comprensión básica de cómo depurar un módulo.
- El registro cronológico y el rastreo es activo mientras se despliegue el módulo.

Acerca de esta tarea

La tarea de resolución de problemas de un despliegue se inicia después de recibir la notificación de un error. Hay varios síntomas de un despliegue anómalo que tiene que inspeccionarse antes de emprender una acción.

Procedimiento

1. Determine si la instalación de aplicación ha sido anómala.

   Examine si hay mensajes que especifiquen la causa de la anomalía en el archivo `SystemOut.log`. Algunos de los motivos de que es posible que no se instale una aplicación son los siguientes:
   - Intenta instalar una aplicación en varios servidores en la misma célula de Network Deployment.
   - Una aplicación tiene el mismo nombre que un módulo existente en la célula de Network Deployment en la que se instala la aplicación.
   - Intenta desplegar módulos Java EE en un archivo EAR en servidores de destino diferentes.

   Importante: Si se ha producido un error en la instalación y la aplicación contiene servicios, debe eliminar los destinos de SIBus o las especificaciones de activación JCA creados antes de la anomalía antes de intentar volver a instalar la aplicación. El modo más sencillo de eliminar estos artefactos es pulsar **Guardar > Descartar todo** después de la anomalía. Si guarda los cambios sin querer, debe eliminar manualmente los destinos de SIBus y las especificaciones de activación JCA (consulte Supresión de los destinos de SIBus y Supresión de las especificaciones de activación JCA de la sección Administración).
2. Si la aplicación se ha instalado correctamente, examínela para determinar si se ha iniciado de manera satisfactoria.

   Si la aplicación no se ha iniciado satisfactoriamente, la anomalía se ha produjo cuando el servidor intentó iniciar los recursos para la aplicación.

   a. Examine si hay mensajes que le orienten sobre cómo continuar en el archivo `SystemOut.log`.

   b. Determine si los recursos que necesita la aplicación están disponibles y/o se han iniciado satisfactoriamente.

      Los recursos no iniciados impiden que se ejecute una aplicación. Esta acción protege la información perdida. Los motivos de que no se inicie un recurso son, entre otros:

- Los enlaces se especifican incorrectamente
- Los recursos no se configuran correctamente
- Los recursos no se incluyen en el archivo RAR (de archivo de recursos)
- Los recursos Web no incluidos en el archivo WAR (de archivo de servicios)

c. Determine si faltan componentes.

El motivo de que falte un componente es un archivo EAR (de archivo de empresa) construido incorrectamente. Asegúrese de que todos los componentes que necesita el módulo se encuentren en las carpetas correctas del sistema de prueba en el que ha compilado el archivo JAR (Java Archive). "Preparación para desplegar en un servidor" contiene información adicional.

3. Examine la aplicación para ver si hay información que fluya a través de ella.

Incluso una aplicación en ejecución puede dejar de procesar información. Las razones de ello son similares a las mencionadas en el paso 2b en la página 527.

a. Determine si la aplicación utiliza servicios incluidos en otra aplicación. Asegúrese de que la otra aplicación esté instalada y se haya iniciado satisfactoriamente.

b. Determine si los enlaces de importación y exportación de los dispositivos contenidos en otras aplicaciones que la aplicación con anomalía utiliza están configurados correctamente. Utilice la consola administrativa para examinar y corregir los enlaces.

4. Corrija el problema y reinicie la aplicación.

Supresión de las especificaciones de activación JCA

El sistema construye especificaciones de aplicación JCA al instalar una aplicación que contenga servicios. Hay ocasiones en que debe suprimir estas especificaciones antes de volver a instalar la aplicación.

Antes de empezar

Si suprime la especificación a causa de una instalación de una aplicación con anomalías, asegúrese de que el módulo del nombre JNDI (Java Naming and Directory Interface) coincida con el nombre del módulo que no se ha podido instalar. La segunda parte del nombre JNDI es el nombre del módulo que ha implementado el destino. Por ejemplo, en `sca/SimpleBOCrsmA/ActivationSpec`, **SimpleBOCrsmA** es el nombre de módulo.

Rol de seguridad necesario para esta tarea: Cuando está habilitada la seguridad y la autorización según el rol, debe haber iniciado la sesión como administrador o configurador para realizar esta tarea.

Acerca de esta tarea

Suprima especificaciones de activación de JCA cuando guarde de forma inadvertida una configuración después de instalar una aplicación que contenga servicios y no requiera las especificaciones.

Procedimiento

1. Localice la especificación de la activación que desea suprimir.

Las especificaciones están contenidas en el panel del adaptador de recursos. Para ir a este panel, pulse **Recursos > Adaptadores de recursos**.

a. Localice el **Adaptador de recursos SPI de componente de mensajería de plataforma.**

   Para localizar este adaptador, debe estar en el ámbito **nodo** de un servidor autónomo o en el ámbito **servidor** de un entorno de despliegue.

2. Visualice las especificaciones de activación de JCA asociadas con el Adaptador de recursos SPI de componente de mensajería de plataforma.

   Pulse en el nombre de adaptador de recursos y el panel siguiente muestra las especificaciones asociadas.

3. Suprima todas las especificaciones con un **Nombre JNDI** que coincidan con el nombre de módulo que va a suprimir.
   a. Pulse el recuadro de selección situado junto a las especificaciones adecuadas.
   b. Pulse **Suprimir**.

Resultados

El sistema elimina las especificaciones seleccionadas de la pantalla.

Qué hacer a continuación

Guarde los cambios.

Supresión de los destinos de SIBus

Los destinos de bus de servicio de integración (SIBus) se utilizan para mantener mensajes que procesan los módulos SCA. Si ocurre un problema, es posible que deba eliminar los destinos de bus para resolver el problema.

Antes de empezar

Si suprime el destino a causa de la instalación de una aplicación con anomalías, asegúrese de que el módulo del nombre de destino coincida con el nombre del módulo que no se ha podido instalar. La segunda parte del destino es el nombre del módulo que ha implementado el destino. Por ejemplo, en `sca/SimpleBOCrsmA/component/test/sca/cros/simple/cust/Customer`, **SimpleBOCrsmA** es el nombre de módulo.

Rol de seguridad necesario para esta tarea: Cuando está habilitada la seguridad y la autorización según el rol, debe haber iniciado la sesión como administrador o configurador para realizar esta tarea.

Acerca de esta tarea

Suprima los destinos de SIBus cuando guarde inintencionadamente una configuración después de instalar una aplicación que contenga servicios o ya no necesite los destinos.

Nota: Esta tarea suprime el destino sólo del bus del sistema SCA. También debe eliminar las entradas del bus de aplicación antes de volver a instalar una aplicación que contenga servicios (consulte Supresión de las especificaciones de activación de JCA en la sección Administración de este centro de información).

Procedimiento

1. Inicie una sesión en la consola administrativa.
2. Visualice los destinos en el bus del sistema SCA.
   a. En el panel de navegación, pulse **Integración de servicios → buses**
   b. En el panel de contenido, pulse **SCA.SYSTEM.*nombre_celda*.Bus**
   c. En Recursos de destino, pulse **Destinos**
3. Seleccione el recuadro de selección situado junto a cada destino con un nombre de módulo que coincida con el módulo que está eliminando.
4. Pulse **Suprimir**.

Resultados

El panel sólo muestra los destinos restantes.

Qué hacer a continuación

Suprima las especificaciones de activación JCA relacionadas con el módulo que ha creado estos destinos.