WebSphere® Process Server for Multiplatforms

Versión 6.2.0

Desarrollo y despliegue de módulos

WebSphere® Process Server for Multiplatforms

Versión 6.2.0

Desarrollo y despliegue de módulos

Nota

Antes de utilizar esta información, asegúrese de leer la información general de la sección Avisos al final de este documento.

12 de diciembre de 2008

Esta edición se aplica a la versión 6, release 2, 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.

Cuando se envía información a IBM, se otorga a IBM un derecho no exclusivo de utilizar o distribuir la información del modo que estime apropiado sin incurrir por ello en ninguna obligación con el remitente.

Manuales en PDF y Centro de información

Estos manuales en PDF se proporcionan a efectos prácticos para su impresión o su lectura cuando esté fuera de línea. Para ver la información más reciente, vea el Centro de información en línea.

En conjunto, los manuales en PDF contienen el mismo contenido que el Centro de información.

La documentación en PDF está disponible en un plazo de un trimestre después de un release importante del Centro de información como, por ejemplo, las versiones 6.0 ó 6.1.

La documentación en PDF se actualiza con menos frecuencia que el Centro de información, pero con más frecuencia que los Redbooks. En general, los manuales en PDF se actualizan cuando se han acumulado cambios suficientes para el manual.

Los enlaces externos del manual en PDF se dirigen al Centro de información en la Web. Los enlaces a destinos externos del manual en PDF están marcados con iconos que indican si el destino es un manual en PDF o una página Web.

Tabla 1. Iconos que preceden a enlaces externos a este manual

Icono	Descripción

	Un enlace a una página Web, incluido un enlace a una página del Centro de información.
	
	Los enlaces al Centro de información van a través de un servicio de direccionamiento indirecto, de modo que siguen funcionando aunque el tema de destino se haya desplazado a otra ubicación.
	
	Si desea encontrar una página enlazada en un Centro de información local, puede buscar el título del enlace. Alternativamente, puede buscar el ID del tema. Si los resultados de la búsqueda abarcan diferentes temas para diferentes variantes de producto, puede utilizar los controles de resultado de búsqueda **Agrupar por** para identificar el tema que desea ver. Por ejemplo:
	1. Copie el URL de enlace; por ejemplo, haga clic con el botón derecho en el enlace y seleccione Copiar ubicación del enlace. Por ejemplo: `http://www14.software.ibm.com/webapp/wsbroker/redirect?version=wbpm620 &product=wesb-dist&topic=tins_apply_service`
	2. Copie el ID de tema tras `&topic=`. Por ejemplo: `tins_apply_service`
	3. En el campo de búsqueda del Centro de información local, pegue el ID del tema. Si la característica de documentación está instalada de forma local, el resultado de búsqueda incluirá el tema. Por ejemplo:
	`1 resultado(s) encontrado para`
	
	`Agrupar por: Ninguno | Plataforma | Versión | Producto`
	`Mostrar resumen`
	
	`Instalación de fixpacks y paquetes de renovación con el instalador de actualizaciones`
	
	4. Haga clic en el enlace del resultado de la búsqueda para mostrar el tema.

Tabla 1. Iconos que preceden a enlaces externos a este manual (continuación)

Icono	Descripción
	Un enlace a un manual en PDF.

Contenido

Figuras

Tablas

Parte 1. Desarrollo de aplicaciones

Capítulo 1. Desarrollo de soluciones de integración empresarial

En esta sección se describen los principios básicos del modelo de programación de integración empresarial. Presenta el estándar SCA (Service Component Architecture) y describe los patrones relacionados con la integración empresarial.

La integración empresarial es la disciplina que permite a las empresas identificar, consolidar y optimizar los procesos empresariales. El objetivo es mejorar la productividad y maximizar la eficacia organizativa. El interés en la integración empresarial se ha intensificado a medida que las empresas se fusionan y consolidan, y a medida que acumulan un patrimonio de activos de información de procedencias muy variadas. Dichos activos, a menudo, no disponen de coherencia ni coordinación, hecho que genera "islas de información".

La integración empresarial está estrechamente vinculada a Business Process Management (BPM) y a Service-Oriented Architecture (SOA). Dependiendo de la naturaleza de la empresa, y del alcance de las necesidades de la integración, la integración empresarial plantea unos requisitos diferentes para los departamentos de 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 de integración empresarial:

- **Integración de aplicaciones**: es un requisito común. La complejidad de los proyectos de integración de aplicaciones varía desde los casos sencillos, en los que necesita asegurarse de que un pequeño número de aplicaciones pueda compartir información, a situaciones más complejas, en las que las transacciones y los intercambios de datos deben reflejarse simultáneamente en varias aplicaciones "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. Por conectividad se entiende tanto el flujo de información que existe entre organizaciones o empresas, como la posibilidad de acceder a los servicios de 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 diferentes protocolos y mecanismos para poder acceder a los servicios de TI que se pueden haber desarrollado utilizando tecnologías muy variadas.

- 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, la integración empresarial engloba muchos de los temas y elementos que también son comunes a SOA. La visión que IBM tiene de la integración empresarial se basa en muchos de los mismos conceptos fundacionales que se pueden encontrar en SOA. Una de las consecuencias inmediatas de esta visión es que las soluciones de integración empresarial pueden requerir varios productos para poder llevar a cabo 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 que IBM tiene de la integración empresarial, debe permitir a las empresas definir, crear, fusionar, consolidar y modernizar los procesos empresariales utilizando aplicaciones que se ejecuten en una infraestructura de TI SOA. El trabajo de la integración empresarial se basa, realmente, en los roles. A gran nivel, implica modelar, desarrollar, dirigir, gestionar y supervisar las aplicaciones de proceso empresarial. Con la ayuda de las herramientas y los procedimientos adecuados, le permite automatizar los procesos empresariales en los que intervienen personas y sistemas heterogéneos, tanto dentro como fuera de la empresa. Uno de los aspectos clave de la integración empresarial es la capacidad de optimizar las operaciones empresariales de forma que sean los suficientemente eficientes, escalables, fiables y flexibles para poder manejar los cambios que se efectúen.

La integración empresarial 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 la integración empresarial, se dará cuenta de que tendrá que 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 de herencia.

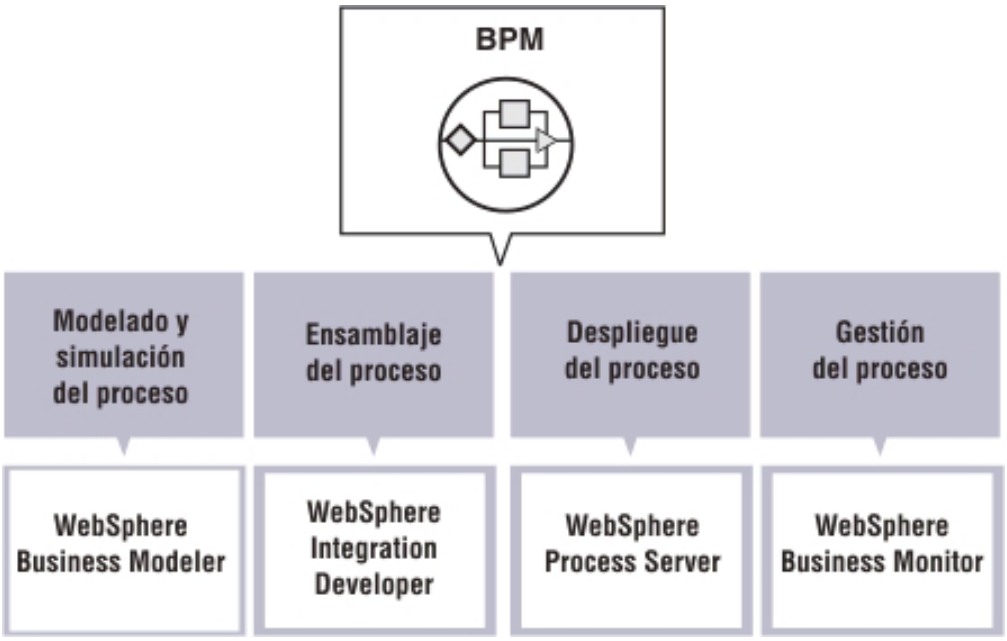


Figura 1. Las herramientas de IBM distribuyen el ciclo de vida de BPM completo, permitiéndole modelar, ensamblar, desplegar y gestionar sus procesos.

La integración empresarial 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. La integración empresarial engloba muchos de los servicios y elementos de la arquitectura de referencia SOA.

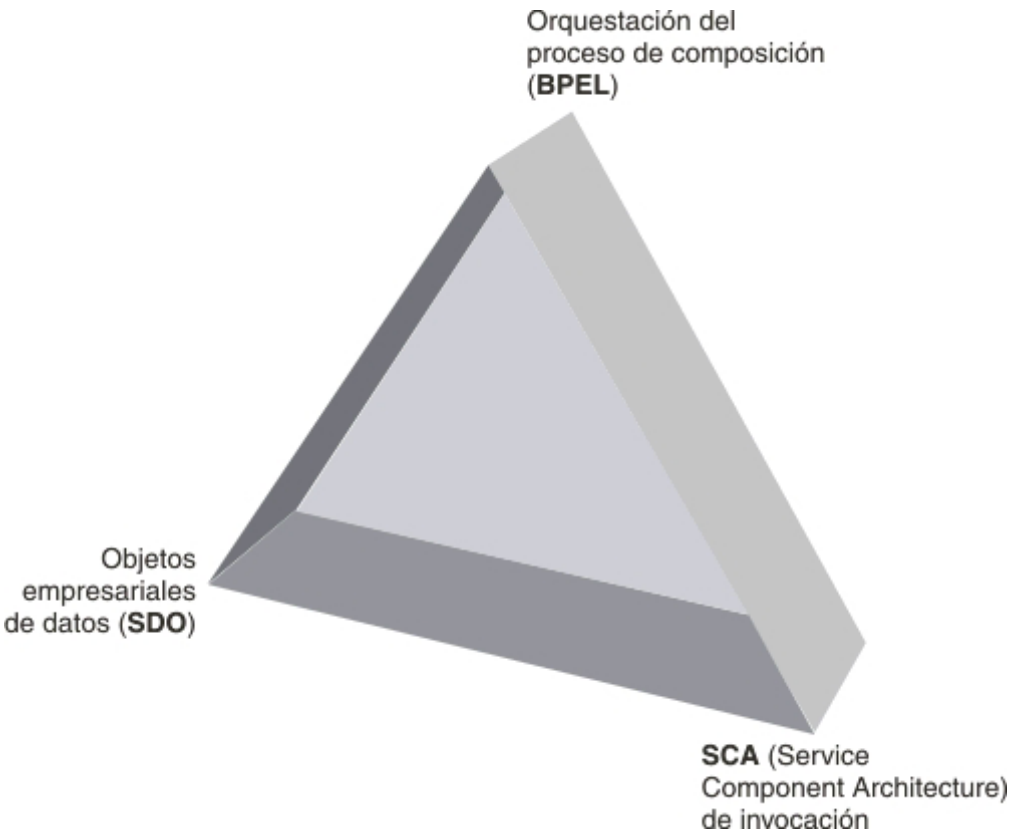
Para obtener más detalles sobre estos conceptos, junto con ejemplos de programación, consulte:

- *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.

Modelo de programación de integración empresarial

La integración empresarial no es una tarea fácil. Existen tantas tecnologías y tantos modos de representar o interactuar con los datos, que lograr la integración no es una tarea fácil. Si tiene en cuenta los tres aspectos de un modelo de programación, que son los datos, la invocación y la composición, y aplica alguno de los nuevos paradigmas de un enfoque basado en servicios, el nuevo modelo de programación para SOA empieza a tomar importancia.

En primer lugar, vemos que los datos se representan, principalmente, mediante XML (Extensible Markup Language), y que se programan con SDO (Service Data Objects) o a través de recursos XML como, por ejemplo, XPath o XSLT (Extensible Stylesheet Language Transformation). En segundo lugar, la invocación de servicios se correlaciona con SCA (Service Component Architecture). Para terminar, la composición se incorpora en la orquestación de procesos mediante BPEL (Business Process Execution Language). En la figura se muestran los tres aspectos de este nuevo modelo de programación.



Service Component Architecture

Además de proporcionar una sintaxis y un mecanismo coherentes para la invocación de servicios, SCA es la infraestructura de invocación que proporciona a los desarrolladores un modo de encapsular las implementaciones de servicios en componentes reutilizables. Permite a los desarrolladores definir interfaces, implementaciones y referencias de un modo independiente de la tecnología, que le ofrece la oportunidad de enlazar los elementos a cualquier tecnología que elija. SCA separa la lógica empresarial de la infraestructura, de forma que los programadores de aplicaciones pueden centrarse en cómo resolver los problemas empresariales.

Arquitectura y patrones de la integración empresarial

Un típico proyecto de integración empresarial implica coordinar varios activos de TI diferentes que, potencialmente, se ejecutan en plataformas distintas, 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 resultan aplicables a la integración empresarial, entre los que se incluyen los patrones para direccionamiento de petición y respuesta, patrones de canales (como, 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 integración de procesos empresariales que más prevalecen son los siguientes:

- **Intermediario de integración**: en este escenario, la solución de integración empresarial actúa como un intermediario ubicado entre varias aplicaciones “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. Posteriormente, el flujo desencadena los demás pasos, creando elementos de trabajo para la persona responsable de efectuar las acciones pertinentes o bien llamando a los servicios adecuados.

En ambos escenarios, la solución de integración debe hacer lo siguiente:

1. Trabajar con fuentes de información dispares y formatos de datos distintos, y poder convertir la información entre formatos diferentes.

2. Poder invocar varios servicios que, potencialmente, pueden utilizar mecanismos de invocación y protocolos distintos.

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 los 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 desplegador de soluciones debe concentrarse en la calidad del servicio (QoS).

Un modelo de objeto empresarial 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. Más concretamente, componentes distintos pueden utilizar técnicas diferentes para representar los elementos empresariales como, por ejemplo, los datos de un pedido, la 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. Le permite definir interfaces, implementaciones y referencias de un modo independiente de la tecnología, que le 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.

La infraestructura del objeto empresarial

La industria de software informático ha desarrollado varias infraestructuras y modelos de programación que permiten a los desarrolladores encapsular la información del OE (objeto empresarial). Por lo general, una infraestructura de OE debe proporcionar independencia de la base de datos, correlacionar de forma transparente los objetos empresariales personalizados a las tablas de base de datos o a las estructuras de datos de los sistemas de información de empresa, y enlazar

objetos empresariales a las interfaces de usuario. Últimamente, los esquemas ML son, quizás, la manera más popular y más aceptada de representar la estructura de un objeto empresarial.

Desde la perspectiva de las herramientas, WebSphere Integration Developer proporciona a los desarrolladores un modelo de OE común para representar distintos tipos de entidades de dominios diferentes. A la hora de desarrollar, WebSphere Integration Developer representa los objetos empresariales como esquemas XML. En el tiempo de ejecución, no obstante, esos mismos objetos empresariales se representan en la memoria mediante un instancia Java de un SDO. SDO es una especificación estándar que IBM y BEA Systems han acordado y desarrollado conjuntamente. IBM ha ampliado la especificación SDO al incluir algunos servicios adicionales que facilitan la manipulación de los datos dentro de los objetos empresariales.

Antes de adentrarnos en la infraestructura del OE, echemos un vistazo a los tipos de datos básicos que se manipulan:

- **Datos de instancia**: son los datos y las estructuras de datos reales, a partir de objetos básicos sencillos con propiedades escalares hasta jerarquías de objetos complejos de gran tamaño. Esto también incluye las definiciones de datos como, por ejemplo, una descripción de los tipos de atributo básicos, información de tipos complejos, cardinalidad y los valores por omisión.
- **Metadatos de instancia**: son datos específicos de la instancia. La información incremental se añade a la base de datos como, por ejemplo, el rastreo de cambios (también conocido como resumen de cambios), la información de contexto asociada con cómo se creó el objeto o los datos, y las cabeceras y los pies del mensaje.
- **Metadatos de tipo**: normalmente es información específica de aplicación como, por ejemplo, correlaciones a nivel de atributo a columnas de datos de información del sistema de información de empresa (EIS) (por ejemplo, correlación de un nombre de campo de OE a un nombre de columna de tabla SAP).
- **Servicios**: son, básicamente, servicios de ayuda que obtienen datos, establecen datos, obtienen el resumen de cambios o proporcionan acceso al tipo de definición de datos.

En la tabla se muestra cómo se implementan los tipos de datos básicos en la plataforma WebSphere.

Tabla 2. Abstracciones de datos y las correspondientes implementaciones

Abstracción de datos	**Implementación**
Datos de instancia	Objeto empresarial (SDO)
Metadatos de instancia	Gráfico de empresa
Metadatos de tipo	Metadatos de empresa, metadatos de tipo de objeto empresarial
Servicios	Servicios de objeto empresarial

Trabajo con la infraestructura del objeto empresarial de IBM

Tal como ya se ha mencionado, la infraestructura del OE de WebSphere Process Server es una ampliación del estándar SDO. Por tanto, los objetos empresariales intercambiados entre los componentes de WebSphere Process Server son instancias de la clase `commonj.sdo.DataObject`. No obstante, la infraestructura del OE 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 empresariales, la infraestructura del OE de WebSphere amplía las especificaciones SDO al proporcionar un conjunto de servicios Java. Dichos servicios forman parte del paquete denominado `com.ibm.websphere.bo`:

- **BOFactory**: el servicio clave que proporciona varias maneras de crear instancias de objetos empresariales.
- **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 empresariales (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 empresariales contienen la misma información. Da soporte a la igualdad profunda y superficial.
- **BOType** y **BOTypeMetaData**: estos servicios materializan las instancias de `commonj.sdo`.Type y le permiten 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.

Service Component Architecture

SCA es una abstracción que puede implementar de varias maneras. No obliga a utilizar ninguna tecnología, lenguaje de programación, protocolo de invocación ni mecanismo de transporte concretos. Los componentes SCA se describen mediante SCDL (Service Component Definition Language), que es un lenguaje basado en XML.

Un componente SCA tiene las características siguientes:

- Envuelve un artefacto de implementación, que contiene la lógica que puede ejecutar el componente.
- Expone una o más interfaces.
- Puede exponer una o más referencias a otros componentes. La lógica de la implementación determina si un componente expone una referencia. Si la implementación requiere invocar otros servicios, el componente SCA necesita exponer una referencia.

Esta información se centra en la implementación SCA que ofrece WebSphere Process Server y en la herramienta WebSphere Integration Developer, disponible para crear y combinar componentes SCA. WebSphere Process Server y WebSphere Integration Developer dan soporte a los artefactos de implementación siguientes:

- Objetos Java sencillos
- Procesos empresariales
- Máquinas de estado de empresa
- Tareas de usuario
- Normas empresariales
- Flujo de mediación

SCA separa la lógica empresarial de la infraestructura, de forma que los programadores de aplicaciones pueden centrarse en cómo resolver los problemas empresariales. El producto WebSphere Process Server de IBM se basa en esta misma premisa. La Figura 2 muestra el modelo de la arquitectura de WebSphere Process Server.

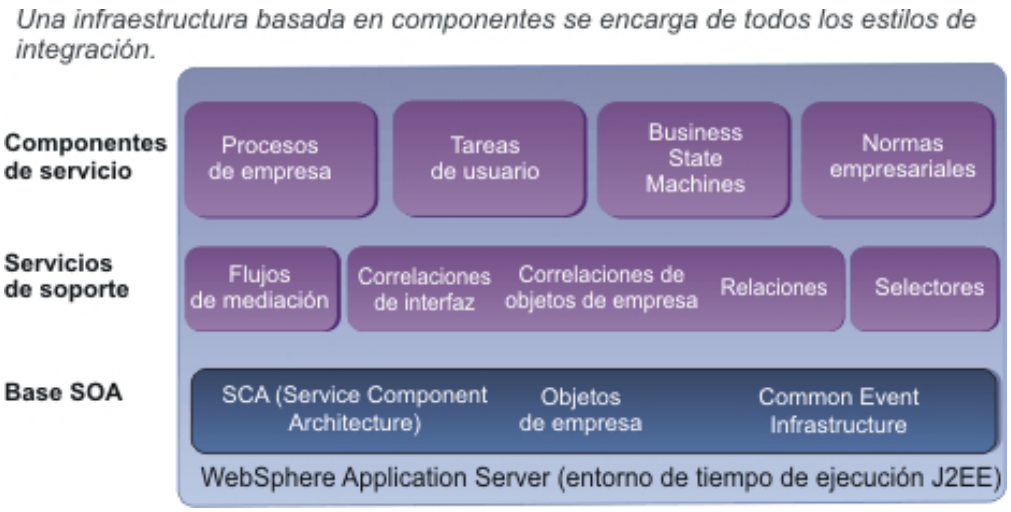


Figura 2. Infraestructura basada en componentes de WebSphere Process Server

En el entorno de WebSphere, la infraestructura SCA se basa en el entorno de ejecución de Java 2 Platform, Enterprise Edition (J2EE) de WebSphere Application Server. La infraestructura global de WebSphere Process Server consta de SOA Core, Servicios de soporte y los Componentes de servicio. Hay disponible, en WebSphere Enterprise Service Bus, la misma infraestructura con un subconjunto de esta posibilidad global, orientada más específicamente a las necesidades de conectividad e integración de aplicaciones de la integración empresarial.

La interfaz de un componente SCA, tal como se ilustra en la Figura 3 en la página 13, se puede representar con uno de los aspectos siguientes:

- Una interfaz Java
- Un tipo de puerto WSDL (en WSDL 2.0, al tipo de puerto se le conoce como interfaz)

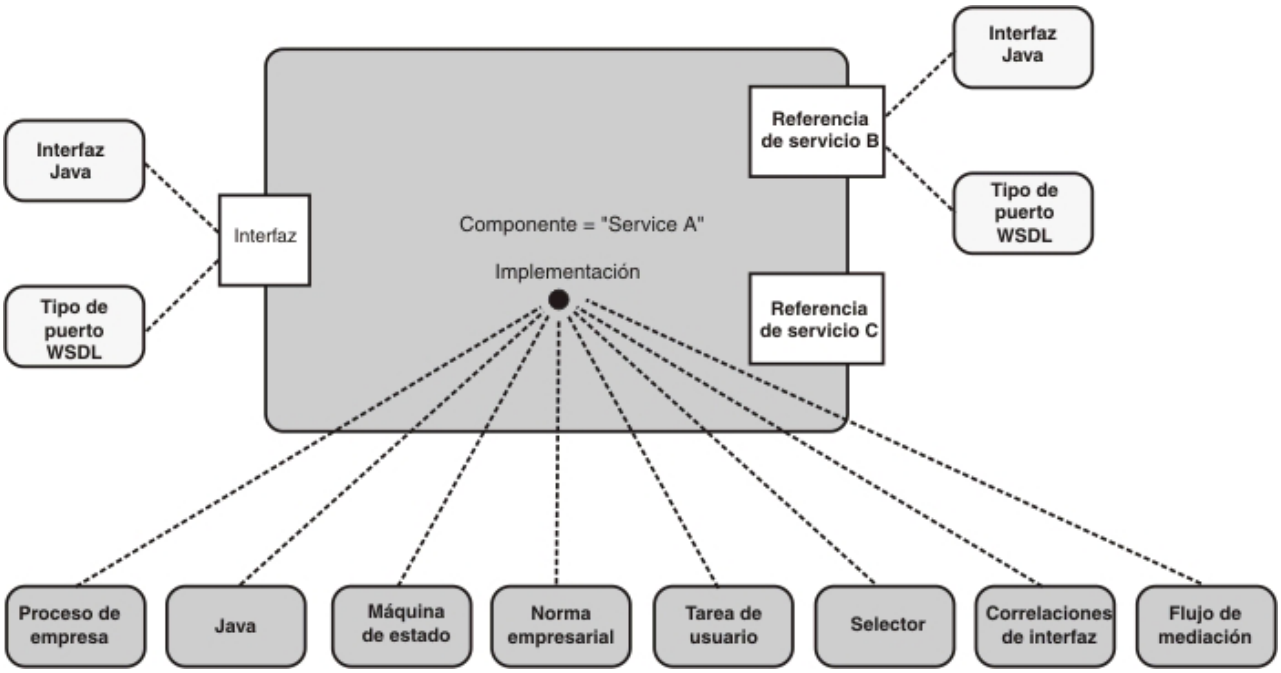


Figura 3. SCA en WebSphere Process Server

Un módulo SCA es un grupo de componentes que se conectan juntos, al enlazar directamente referencias e implementaciones. En WebSphere Integration Developer, cada módulo SCA tiene un diagrama de ensamblaje asociado al mismo, que representa la aplicación empresarial integrada, que consta de los componentes SCA y las conexiones que los conectan. Una de las principales responsabilidades del desarrollador de integración es crear el diagrama de ensamblaje, conectando los componentes que forman la solución. WebSphere Integration Developer proporciona un editor de ensamblaje gráfico, para ayudarle a realizar esta tarea. Al crear el diagrama de ensamblaje, el desarrollador de integración puede proceder de una de dos maneras:

- **De arriba-abajo** define los componentes, sus interfaces y sus interacciones antes de crear la implementación. El desarrollador de integración puede definir la estructura del proceso, identificar los componentes necesarios y sus tipos de implementación y, a continuación, generar un esqueleto de implementación.
- **De abajo arriba** combina los componentes existentes. En este caso, el desarrollador de integración tiene, simplemente, que arrastrar y soltar las implementaciones existentes en el diagrama de ensamblaje.

El enfoque ″de abajo arriba″ se utiliza más a menudo cuando los clientes tienen servicios existentes que desean reutilizar y combinar. Cuando tenga que crear nuevos objetos empresariales desde cero, es posible que adopte el enfoque ″de arriba-abajo″.

El modelo de programación SCA: Conceptos básicos

El modelo de programación SCA se basa en el concepto de *componente* de software. Tal como ya se ha mencionado, un componente es una unidad que implementa cierta lógica y que hace que esté disponible para los demás 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 4 tiene tres componentes, C1, C2 y C3. 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.

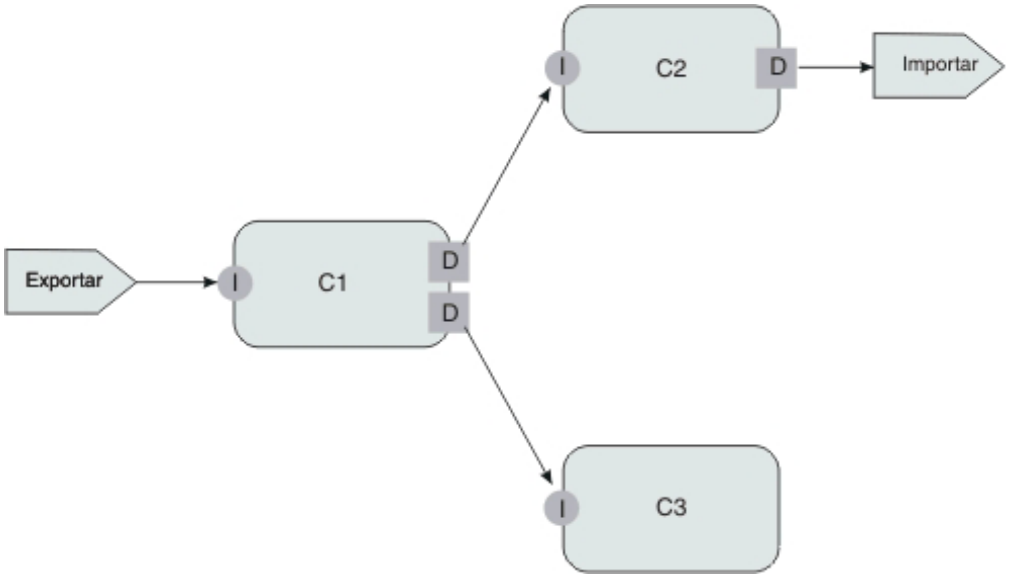


Figura 4. Diagrama de ensamblaje

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. Éste es el mecanismo de invocación clásico.
- **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 parecido al 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.

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 manera similar, si existen aplicaciones externas que deban acceder al componente, que el caso más probable, debe hacer que resulte accesible. Para ello, puede utilizar un componente especial que exponga la lógica al "mundo exterior". Dicho tipo de componente se denomina *exportación*. También pueden invocarse de manera síncrona o asíncrona.

Referencias autónomas

En WebSphere Process Server, un módulo de servicio SCA se empaqueta como un archivo EAR J2EE, que contiene otros submódulos J2EE. Los elementos J2EE como, por ejemplo, un archivo WAR, se pueden empaquetar junto con el módulo SCA. Los artefactos que no son SCA como, por ejemplo, JSP, también se pueden empaquetar junto con un módulo de servicio SCA. Esto les permite invocar los servicios SCA a través del modelo de programación de clientes SCA, mediante un tipo de componente especial 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 componentes de servicio, interfaces e implementaciones y, normalmente, son externos respecto 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). Esto se lleva a cabo en WebSphere Integration Developer. Normalmente, los calificadores de QoS se aplican cuando se está preparado para considerar el despliegue de soluciones. Para obtener más información, consulte Quality of service qualifier reference.

Procesos empresariales

Los procesos empresariales, más específicamente, los procesos empresariales basados en BPEL, forman la piedra angular de los componentes de servicio en SCA.

Independientemente de si se trata de una simple aprobación de pedido o de un complejo proceso de fabricación, las empresas siempre tienen procesos empresariales. Un *proceso empresarial* es un conjunto de actividades, relacionadas con la empresa, que se invocan en una secuencia específica para alcanzar un objetivo de empresa. En el mundo de la integración empresarial, un proceso empresarial se define utilizando algún tipo de lenguaje de marcación.

Dichos procesos empresariales pueden invocar otros servicios de soporte u otros componentes de servicio como, por ejemplo, máquinas de estado de empresa, tareas de usuario, normas empresariales o correlaciones de datos. Y, cuando se despliegan, estos procesos se pueden llevar a cabo rápidamente, o ejecutarse durante un largo período de tiempo. A veces, estos procesos pueden ejecutarse durante años.

Al igual que ocurre con la mayoría de componentes del mundo J2EE, los procesos empresariales se ejecutan dentro de un contenedor. En la plataforma de IBM WebSphere, este contenedor específico se llama Business Process Choreographer. En WebSphere Process Server, Business Process Choreographer es el responsable de ejecutar los procesos empresariales y las tareas de usuario.

Tareas de usuario

Una tarea de usuario es un componente que permite a las personas y servicios interactuar.

Algunas tareas de usuario representan tareas a realizar para personas. Estas tareas puede iniciarlas una persona o un servicio automatizado. Las tareas de usuario se pueden utilizar para implementar actividades de procesos empresarialesque requieren interacciones de usuario como, por ejemplo, las aprobaciones y el manejo de excepciones manual. Otras tareas de usuario se pueden utilizar para invocar un servicio o para coordinar la colaboración entre personas. No obstante, independientemente de cómo se haya iniciado la tarea, un persona de un grupo de personas, a la que se ha asignado la tarea, realiza el trabajo asociado a la tarea.

Se asignan tareas de usuario a las personas ya sea de forma estática o especificando criterios, como un rol o grupo, que se resuelven en tiempo de ejecución utilizando un directorio de personas. De forma alternativa, se utilizan los datos de entrada de una tarea de usuario o los datos de un proceso empresarialpara encontrar las personas adecuadas para trabajar en una tarea.

Creación de aplicaciones de integración empresarial

La expresión *integración empresarial* significa integrar aplicaciones, datos y procesos dentro de una empresa, o entre un conjunto de empresas. El concepto de integración también significa desarrollar procesos, porque hay cierta lógica en la secuencia de las aplicaciones que se ensamblan para integrarlas. WebSphere Integration Developer se utiliza para crear aplicaciones de integración empresarial.

En esta sección se proporciona información sobre el proceso de desarrollo para un módulo de integración empresarial.

El flujo de desarrollo típico para los módulos y los módulos de mediación es el siguiente:

1. Iniciar WebSphere Integration Developer y abrir un espacio de trabajo.
2. Ir a la perspectiva Business Integration para llevar a cabo el desarrollo.
3. Crear una biblioteca donde almacenar los artefactos como, por ejemplo, objetos empresariales e interfaces, que se comparten entre varios módulos.
4. Crear un módulo o módulo de medicación nuevo.
5. Crear los objetos empresariales para que contengan los datos de aplicación como, por ejemplo, datos de clientes o de pedidos.
6. Crear la interfaz y definir las operaciones de interfaz para cada componente. La interfaz determina qué datos se pueden pasar de un componente a otro.
7. Crear e implementar los componentes de servicio.
8. Crear el ensamblaje del módulo añadiendo los componentes de servicio, importaciones y exportaciones al diagrama de ensamblaje. Conectar los componentes entre sí. Enlazar las importaciones y las exportaciones al protocolo.
9. Probar el módulo en el entorno de prueba integrado.
10. Desplegar el módulo en WebSphere Process Server.
11. Comparta el módulo probado con los demás miembros del equipo, colocándolo en un repositorio.

Capítulo 2. 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.

Por qué y cuándo se efectúa 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 en "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. 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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);
```

Visión general del aislamiento de módulos y destinos

Al desarrollar módulos, identificará servicios que varios módulos pueden utilizar. Al reforzar los servicios de este modo se minimiza el ciclo de desarrollo y los costes. Si dispone de un servicio que muchos módulos utilizan, aisle los módulos de invocación del destino, de manera que si se actualiza el destino, el cambio al nuevo servicio sea transparente al módulo llamante. En este tema se contrasta el modelo de invocación sencilla y el modelo de invocación aislada y se proporciona un ejemplo de cómo puede resultar de utilidad el aislamiento. Aunque se describe un ejemplo específico, no es el único modo de aislar módulos de destinos.

Modelo de invocación sencilla

Al desarrollar módulos, puede utilizar servicios que se encuentren en otros módulos. Para hacer esto, puede importar el servicio al módulo y luego invocar ese servicio. El servicio importado se “conecta” con el servicio exportado por el otro módulo, ya sea en WebSphere Integration Developer o enlazando el servicio en la consola administrativa. El modelo de invocación sencilla ilustra este modelo.

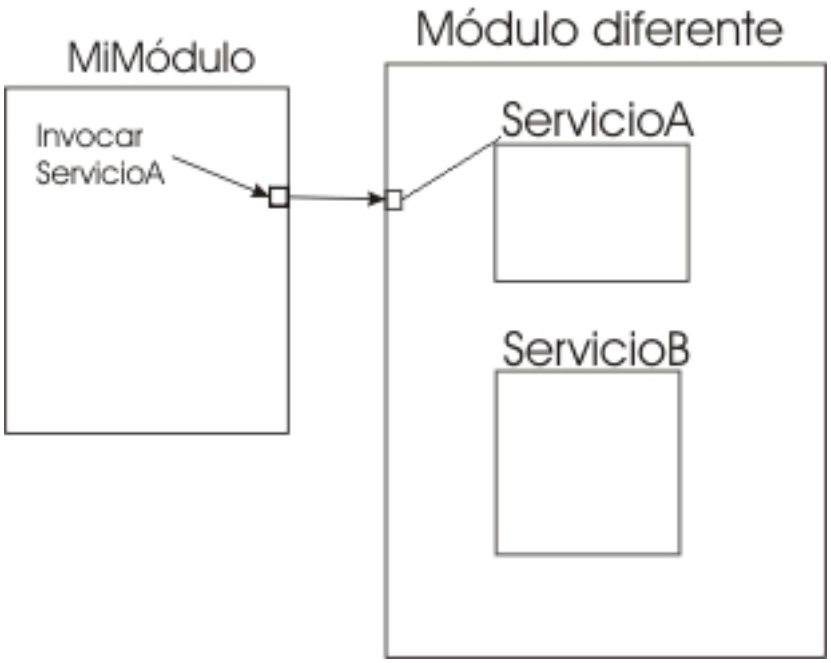


Figura 5. Modelo de invocación sencilla

Modelo de invocación aislada

Para cambiar el destino de una invocación sin detener la invocación de módulos, puede aislar los módulos de invocación del destino de la invocación. Esto permite que los módulos continúen procesando mientras cambia el destino porque no está cambiando el módulo en sí sino el destino en sentido descendente. En el Ejemplo de aislamiento de aplicaciones se muestra cómo el aislamiento permite cambiar el destino sin afectar al estado del módulo de invocación.

Ejemplo de aislamiento de aplicaciones

Utilizando el modelo de invocación sencillo, varios módulos que invocan al mismo servicio se parecerían bastante a varias aplicaciones que invocan a un solo servicio . MODA, MODB y MODC invocan a CalculateFinalCost.

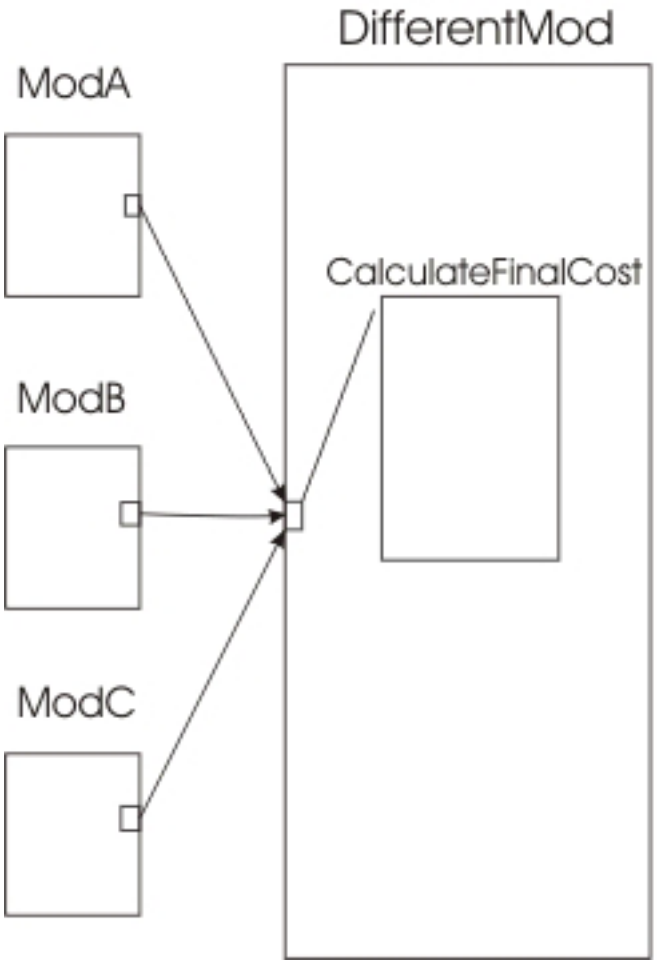


Figura 6. Varias aplicaciones que invocan a un solo servicio

El servicio proporcionado por CalculateFinalCost debe actualizarse de manera que los nuevos costes se reflejen en todos los módulos que utilizan el servicio. El equipo de desarrollo crea y prueba un nuevo servicio UpdatedCalculateFinal para incorporar los cambios. Está preparado para implantar el nuevo servicio en producción. Sin el aislamiento, tendría que actualizar todos los módulos que invocan a CalculateFinalCost para invocar UpdateCalculateFinal. Con el aislamiento, sólo tiene que cambiar el enlace que conecta el módulo de almacenamiento intermedio con el destino.

Nota: Con el cambio del servicio de esta manera se permite continuar para proporcionar el servicio original a otros módulos que puedan necesitarlo.

Utilizando el aislamiento, puede crear un módulo de almacenamiento intermedio entre las aplicaciones y el destino (consulte Modelo de invocación aislada que invoca a UpdateCalculateFinal).

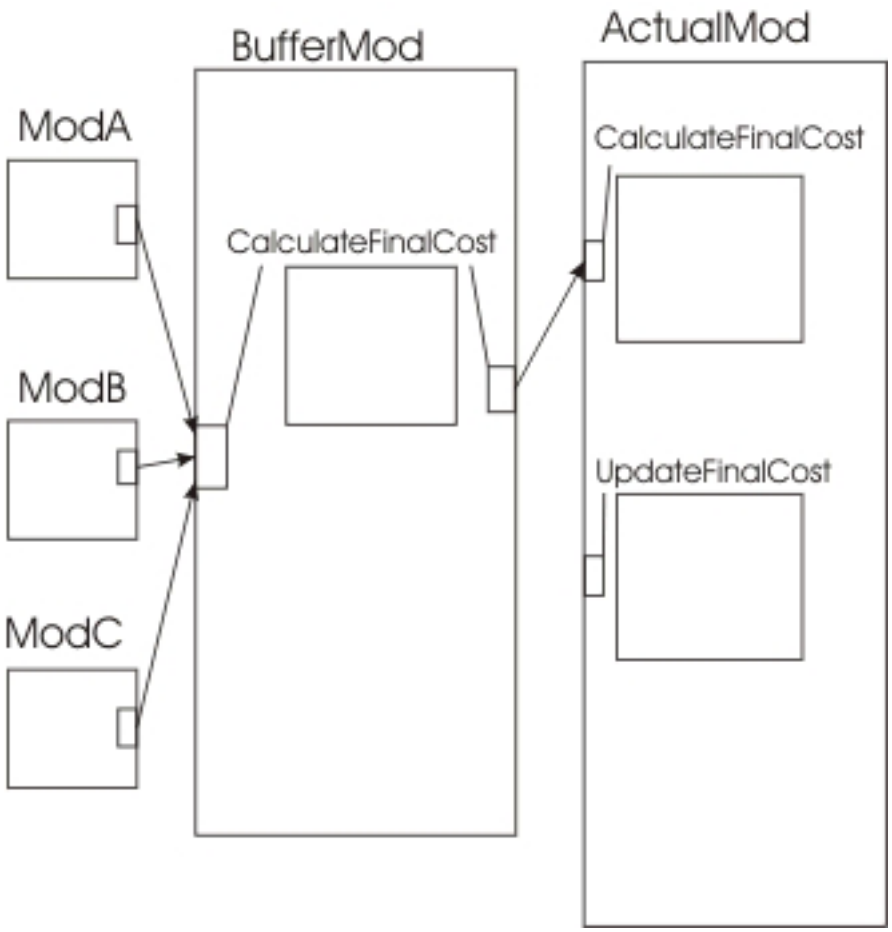


Figura 7. Modelo de invocación aislada que invoca a UpdateCalculateFinal

Con este modelo, los módulos de invocación no cambian, sólo tiene que cambiar el enlace de la importación del módulo de almacenamiento intermedio al destino (consulte Modelo de invocación aislada que invoca a UpdatedCalculateFinal).

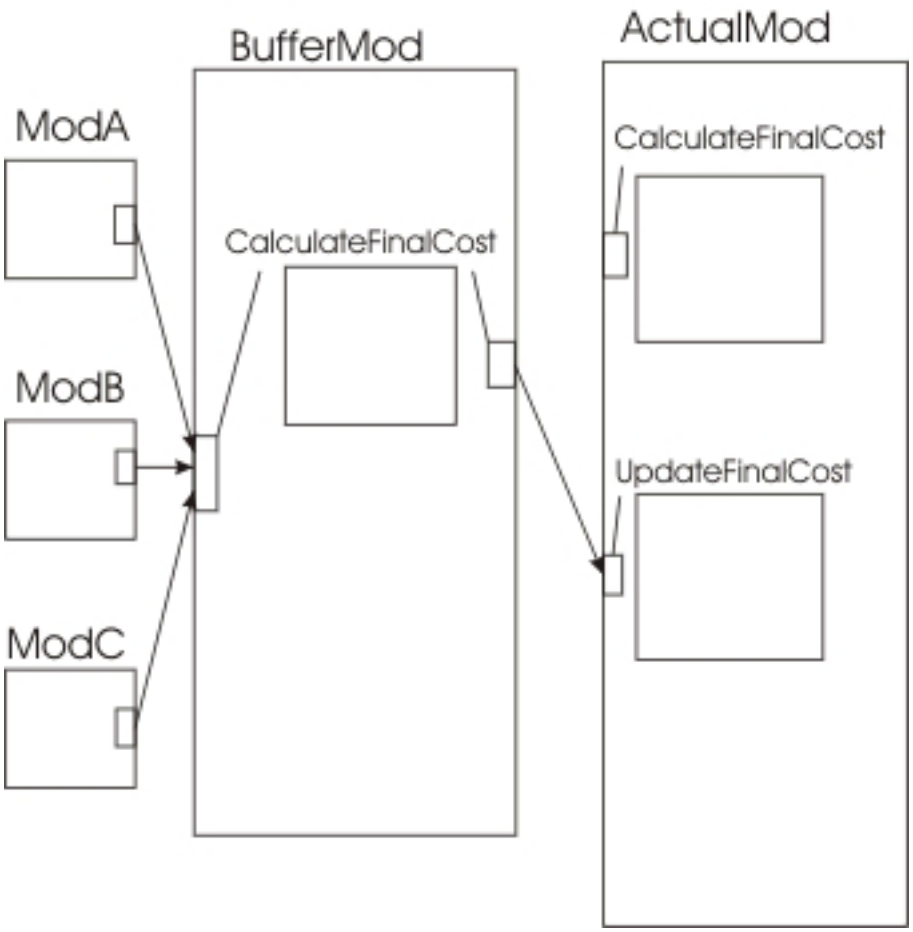


Figura 8. Modelo de invocación aislada que invoca a UpdatedCalculateFinal

Si el módulo de almacenamiento intermedio invoca el destino de forma síncrona, cuando reinicia el módulo de almacenamiento intermedio (ya sea un módulo de mediación o un servicio para un módulo de empresa) los resultados devueltos a la aplicación original proceden del nuevo destino. Si el módulo de almacenamiento intermedio invoca el destino de forma asíncrona, los resultados devueltos a la aplicación original proceden del nuevo destino en la siguiente invocación.

Enlaces HTTP

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

Además, una red de entornos de ejecución SCA se comunica en toda una infraestructura HTTP existente.

El enlace HTTP expone varias características HTTP:

- Los mensajes se presentan a los componentes de mediación de una manera que conserva el formato HTTP y la información de cabecera del mensaje. Esto proporciona una vista más familiar para los programadores de aplicaciones HTTP, usuarios y administradores.

- Una infraestructura de enlace de datos existente se amplía para las convenciones HTTP y proporciona una correlación entre mensajes SCA y cabeceras y cuerpos de mensajes HTTP.
- Se pueden configurar importaciones y exportaciones para admitir una gama de características HTTP comunes.
- Al instalar un módulo SCA que contiene importaciones o exportaciones HTTP, se configura automáticamente el entorno de ejecución de forma adecuada para permitir la conectividad con HTTP.

Puede encontrar instrucciones detalladas sobre la creación de importaciones y exportaciones HTTP en el Centro de información de en **WebSphere Integration Developer** > **Desarrollo de aplicaciones de integración** > **Enlace de datos HTTP**.

Tareas relacionadas


Visualización de enlaces HTTP
Después de desplegar una aplicación, puede desear examinar los enlaces HTTP para asegurarse de que son correctos.


Cambio de enlaces de exportación HTTP
La consola administrativa permite cambiar la configuración de enlaces de exportación HTTP sin cambiar el código fuente original y, a continuación, volver a desplegar la aplicación.


Cambio de enlaces de importación HTTP
La consola administrativa permite cambiar la configuración de enlaces de importación HTTP sin cambiar el código fuente original y, a continuación, volver a desplegar la aplicación.

Capítulo 3. Guías y técnicas de programación

Esta sección incluye guías y ejemplos de programación.

Los siguientes enlaces contienen guías de información para programar varios componentes, aplicaciones y soluciones de integración empresarial.

- Programación de objetos empresariales: mejora del esquema y soporte del esquema del sector
- Guía de programación de la gestión normas empresariales
- Desarrollo de aplicaciones XMS
- Desarrollo de aplicaciones cliente para procesos empresarialesy tareas

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.

Capítulo 4. 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 de la clase 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.

Por qué y cuándo se efectúa 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.

Conceptos relacionados

Capítulo 7, “Normas de tiempo de ejecución utilizadas para la conversión de Java a Service Data Objects”, en la página 41
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.

Referencia relacionada

 Conversión de Java a XML
El sistema genera XML basado en tipos Java mediante la utilización de normas predefinidas.

 Programa de utilidad de línea de mandatos genMapper
Utilice el mandato genMapper para generar un componente que hace de puente de una referencia de SCA (Service Component Architecture) con una interfaz Java.

Capítulo 5. 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.

Por qué y cuándo se efectúa 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.

Conceptos relacionados

Capítulo 7, “Normas de tiempo de ejecución utilizadas para la conversión de Java a Service Data Objects”, en la página 41
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.

Referencia relacionada

 Conversión de Java a XML
El sistema genera XML basado en tipos Java mediante la utilización de normas predefinidas.

 Programa de utilidad de línea de mandatos genMapper
Utilice el mandato genMapper para generar un componente que hace de puente de una referencia de SCA (Service Component Architecture) con una interfaz Java.

Capítulo 6. 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 sobrescribirá el contexto actual. Cuando ejecute una invocación de un componente SCA a otro, una respuesta contendrá un contexto distinto. Un componente de servicio tendrá un contexto de entrada, pero cuando invoque otro servicio, éste último sobrescribirá el 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. Recoge el contexto de la 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.

Ejemplo

Por ejemplo, el contexto que incluye una cabecera de protocolo se propaga a través de la vía de acceso de invocación comenzando por una petición que llega a BPEL procedente de un servicio Web SOAP. BPEL la procesa, y las llamadas fuera de BPEL se efectúan, de forma secuencial, a una salida de enlace de servicio Web y, a continuación, a otra salida de enlace de servicio Web. Una petición procedente del servicio Web SOAP utiliza el servicio de contexto para pasar la cabecera de protocolo. Recoge el servicio de contexto procedente de la petición de entrada y la pasa a la salida de la cabecera de protocolo.

Puede ver un comportamiento parecido con otro componente SCA en lugar del BPEL de este ejemplo.

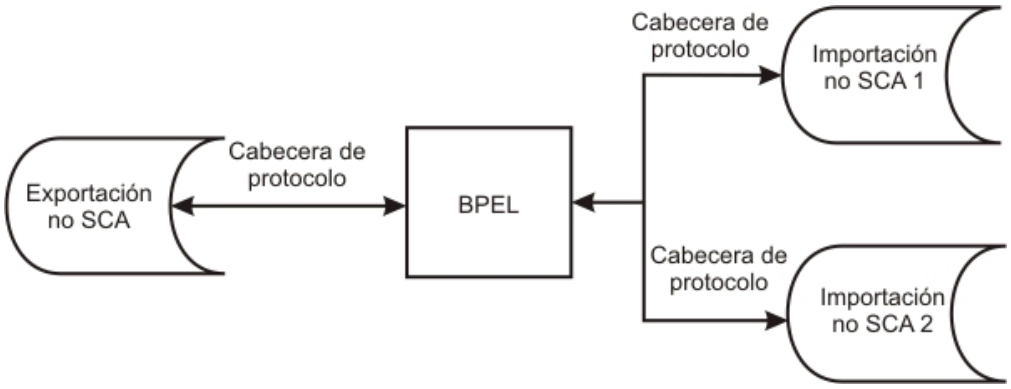


Figura 9. Propagación de contexto que incluye la cabecera de protocolo

He aquí un ejemplo de código.

```
//Importar las clases necesarias;
import com.ibm.bpm.context.ContextService;
import com.ibm.websphere.sca.ServiceManager;
import com.ibm.bpm.context.cobo.ContextObject;
import com.ibm.bpm.context.cobo.ContextObjectFactory;
import com.ibm.bpm.context.cobo.HeaderInfoType;
import com.ibm.bpm.context.cobo.UserDefinedContextType;

//Localizar ContextService;
ContextService contextService =
   (ContextService)ServiceManager.INSTANCE.locateService("com/ibm/bpm/context/ContextService");

// Obtener la información de cabecera
HeaderInfo headerInfo = contextService.getHeaderInfo();
// Obtener el contexto definido de usuario del contexto de ejecución actual
UserDefinedContextType userDefinedContext = contextService.getUserDefinedContext();
if(userDefinedContext == null){ // crear un nuevo contexto si el contexto es nulo
userDefinedContext = ContextObjectFactory.eINSTANCE.createUserDefinedContextType()
}

// Efectuar alguna modificación en la información de cabecera y en userDefinedContext

// Volver a establecer el contexto definido de usuario en el contexto de ejecución actual.
contextService.setUserDefinedContext(userDefinedContext);

// Volver a establecer la información de cabecera en el contexto de ejecución actual.
contextService.setHeaderInfo(headerInfo);
```

Nota: En el componente de flujo de mediación, no deben utilizarse las API `ContxtService`. Utilice el modelo de programación SMO para acceder al contexto.

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.

Capítulo 7. 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 3. Conversión del tipo WSDL a la clase Java

Interfaz	Clases concretas por omisión
Colección	HashSet
Correlación	HashMap
Lista	ArrayList
Conjunto	HashSet

Tareas relacionadas

Capítulo 4, “Alteración temporal de la implementación de la Service Component Architecture generada”, en la página 35
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 de la clase SCA (Service Component Architecture) por la suya propia.

Capítulo 5, “Alteración temporal de una conversión de Service Data Object a Java”, en la página 37
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.

Referencia relacionada

 Conversión de Java a XML
El sistema genera XML basado en tipos Java mediante la utilización de normas predefinidas.

 Programa de utilidad de línea de mandatos genMapper
Utilice el mandato genMapper para generar un componente que hace de puente de una referencia de SCA (Service Component Architecture) con una interfaz Java.

Capítulo 8. Objetos empresariales: mejora del esquema y soporte del esquema del sector

La infraestructura SOA (Service Data Objects) proporciona la base para los datos de objetos empresariales que utiliza WebSphere Process Server

En esta guía se proporciona información acerca de las áreas problemáticas del manejo de las construcciones del esquema para algunas características. Para obtener información acerca de cómo se define el objeto empresarial, las directrices de desarrollo de objetos empresariales y cómo utilizar las API de programación de objetos empresariales, consulte los artículos de la sección ″Información relacionada″ siguientes.

Información relacionada

Web Services Description Language (WSDL) 1.1

Introducción a SDO (Service Data Objects)

Examen de objetos empresariales en WebSphere Process Server

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″ pone todos los hijos de estas estructuras en el mismo nivel. Esto 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 EMF, a todos los nombres de propiedad duplicados que se encuentren 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"));
```

Conceptos relacionados

Capítulo 9, “Matrices en los objetos empresariales”, en la página 65
Puede definir matrices para un elemento en un objeto empresarial de forma que el elemento pueda contener más de una instancia de datos.

Tareas relacionadas

“Utilización de objetos empresariales en grupos de modelos” en la página 71
Utilice los patrones de vía de acceso de grupo de modelos cuando trabaje con objetos empresariales anidados que formen parte de un grupo de modelos.

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.

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/>.

Conceptos relacionados

Capítulo 9, “Matrices en los objetos empresariales”, en la página 65
Puede definir matrices para un elemento en un objeto empresarial de forma que el elemento pueda contener más de una instancia de datos.

¿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 se 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
```

Tareas relacionadas

"Utilización de objetos empresariales en grupos de modelos" en la página 71
Utilice los patrones de vía de acceso de grupo de modelos cuando trabaje con objetos empresariales anidados que formen parte de un grupo de modelos.

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.

Tareas relacionadas

“Utilización de un objeto empresarial anidado definido mediante un comodín” en la página 70
Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

¿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 e 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");  // o 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"
```

Tareas relacionadas

"Utilización de un objeto empresarial anidado definido mediante un comodín" en la página 70
Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

¿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.

Tareas relacionadas

"Utilización de un objeto empresarial anidado definido mediante un comodín" en la página 70
Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

¿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 e isSequenced() = false, debe tener un código anyAttribute, si isOpen() = true e 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
```

Tareas relacionadas

"Utilización de un objeto empresarial anidado definido mediante un comodín" en la página 70

Puede especificar el tipo xsd:any en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

¿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"
```

Tareas relacionadas

"Utilización de un objeto empresarial anidado definido mediante un comodín" en la página 70
Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

¿Cuáles son correlaciones válidas para datos en un anyAttribute?

El código AnyAttribute es parecido al código any y consta de 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

Capítulo 9. 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 rellenar los elementos de datos para la matriz del índice de lista utilizando datos de anchura fija, o delimitados, colocados 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.

Conceptos relacionados

“Utilización del objeto Sequence para establecer el orden de datos” en la página 50
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.

“Soporte de grupo de modelos (todos, opción, secuencia y referencias de grupo)” en la página 46
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.

Capítulo 10. 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>
```

Por qué y cuándo se efectúa esta tarea

Si ha utilizado el método ″de arriba abajo″ tradicional para establecer los datos de objeto empresarial, tendrá que procesar el código siguiente especificando los objetos hijo y nieto antes de establecer 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.

Tareas relacionadas

“Creación de varias instancias de objetos empresariales anidados”
Utilice la función `setWithCreate` para crear varias instancias de objeto empresarial anidado.

“Utilización de un objeto empresarial anidado definido mediante un comodín” en la página 70
Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

“Utilización de objetos empresariales en grupos de modelos” en la página 71
Utilice los patrones de vía de acceso de grupo de modelos cuando trabaje con objetos empresariales anidados que formen parte de un grupo de modelos.

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`.

Por qué y cuándo se efectúa 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 en la matriz, se utilizará si dicho elemento si no es nulo (es decir, contiene datos). 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");
```

Tareas relacionadas

"Instancia individual de un objeto empresarial anidado" en la página 67
Utilice la función `setWithCreate` para crear una instancia individual de objeto empresarial anidado.

"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.

"Utilización de objetos empresariales en grupos de modelos" en la página 71
Utilice los patrones de vía de acceso de grupo de modelos cuando trabaje con objetos empresariales anidados que formen parte de un grupo de modelos.

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.

Por qué y cuándo se efectúa 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.

Conceptos relacionados

"¿Cómo puedo saber si miDataObject tiene una etiqueta anyAttribute?" en la página 62
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.

"Utilización de AnyAttribute para establecer atributos globales para tipos complejos" en la página 62
El código <anyAttribute/> permite establecer cualquier número de atributos globales para un tipo complejo.

"¿Cómo puedo obtener/establecer valores?" en la página 60
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.

"¿Cómo puedo obtener/establecer valores AnyAttribute?" en la página 63
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.

"Utilización de Any para establecer elementos globales para tipos complejos" en la página 59
Puede utilizar el código <any/> para establecer elementos globales en un tipo complejo.

Tareas relacionadas

"Instancia individual de un objeto empresarial anidado" en la página 67
Utilice la función `setWithCreate` para crear una instancia individual de objeto empresarial anidado.

"Creación de varias instancias de objetos empresariales anidados" en la página 68
Utilice la función `setWithCreate` para crear varias instancias de objeto empresarial anidado.

"Utilización de objetos empresariales en grupos de modelos"
Utilice los patrones de vía de acceso de grupo de modelos cuando trabaje con objetos empresariales anidados que formen parte de un grupo de modelos.

Utilización de objetos empresariales en grupos de modelos

Utilice los patrones de vía de acceso de grupo de modelos cuando trabaje con objetos empresariales anidados que formen parte de un grupo de modelos.

Por qué y cuándo se efectúa esta tarea

Los modelos de grupos utilizan el código `xsd:choice`, que puede utilizar para crear objetos empresariales a partir de un objeto empresarial padre. No obstante, EMF (Eclipse Modeling Framework), puede provocar conflictos de nomenclatura que pueden generar una excepción. En el siguiente código de ejemplo se muestra cómo se puede dar esta situación:

```
<?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>
```

Tenga en cuenta que puede haber varias instancia de los elementos denominados ″child1″ y ″child2″.

Debe utilizar los patrones de vía de acceso de SDO (Service Data Object) para los grupos de modelos para resolver estos conflictos.

Resultados

Se obtendrán matrices que utilizan el patrón de vía de acceso SDO para manejar los grupos de modelos, tal como se muestra en el código de ejemplo siguiente:

```
set("child1/grandchild/name", "Bob");

set("child11/grandchild/name", "Joe");
```

Conceptos relacionados

“Soporte de grupo de modelos (todos, opción, secuencia y referencias de grupo)” en la página 46
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.

“¿Cómo puedo trabajar con una matriz de grupo de modelos?” en la página 53
Una matriz de grupo de modelos se crea cuando un grupo de modelos tiene un valor para maxOccurs > 1.

Tareas relacionadas

“Instancia individual de un objeto empresarial anidado” en la página 67
Utilice la función `setWithCreate` para crear una instancia individual de objeto empresarial anidado.

“Creación de varias instancias de objetos empresariales anidados” en la página 68
Utilice la función `setWithCreate` para crear varias instancias de objeto empresarial anidado.

“Utilización de un objeto empresarial anidado definido mediante un comodín” en la página 70
Puede especificar el tipo `xsd:any` en un objeto padre para especificar un objeto hijo, pero sólo si este último ya existe.

Capítulo 11. 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:
```

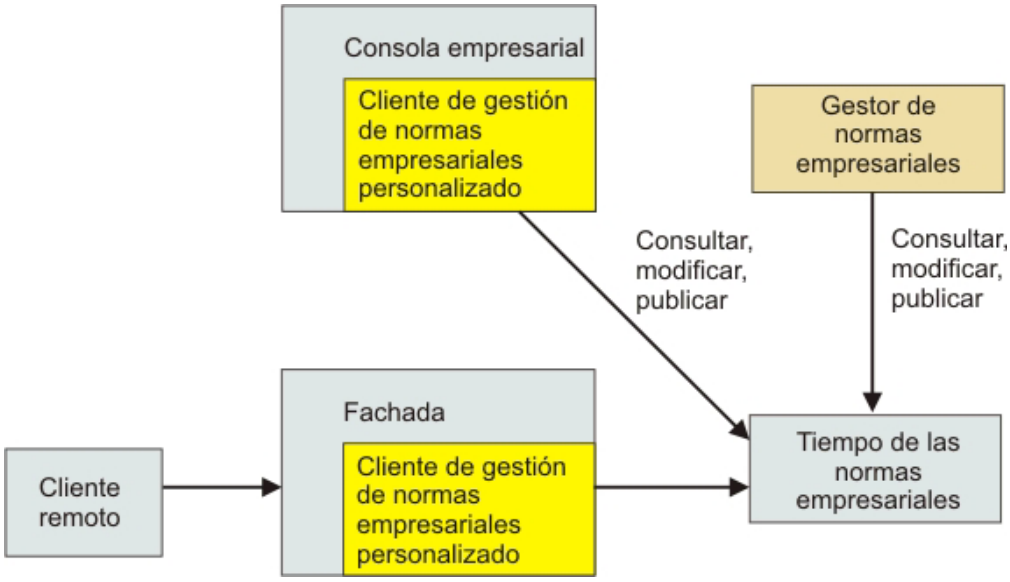
Para obtener más información sobre los servicios de validación, consulte la interfaz `BOInstanceValidator`, en la documentación de API y SPI generada, en la sección Consulta.

Capítulo 12. 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 empresariales 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 empresarial 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.

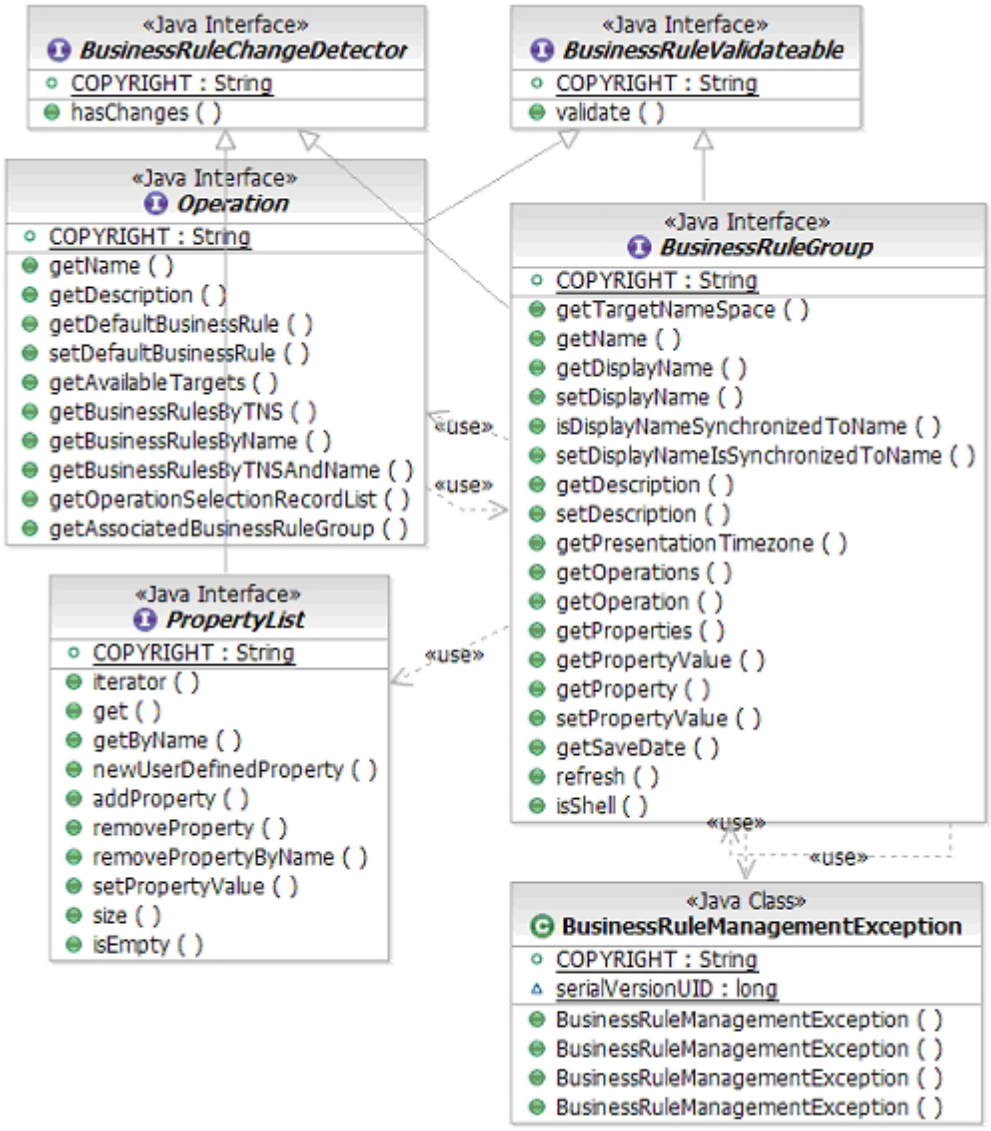


Figura 10. Diagrama de clase de BusinessRuleGroup y clases relacionadas

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.

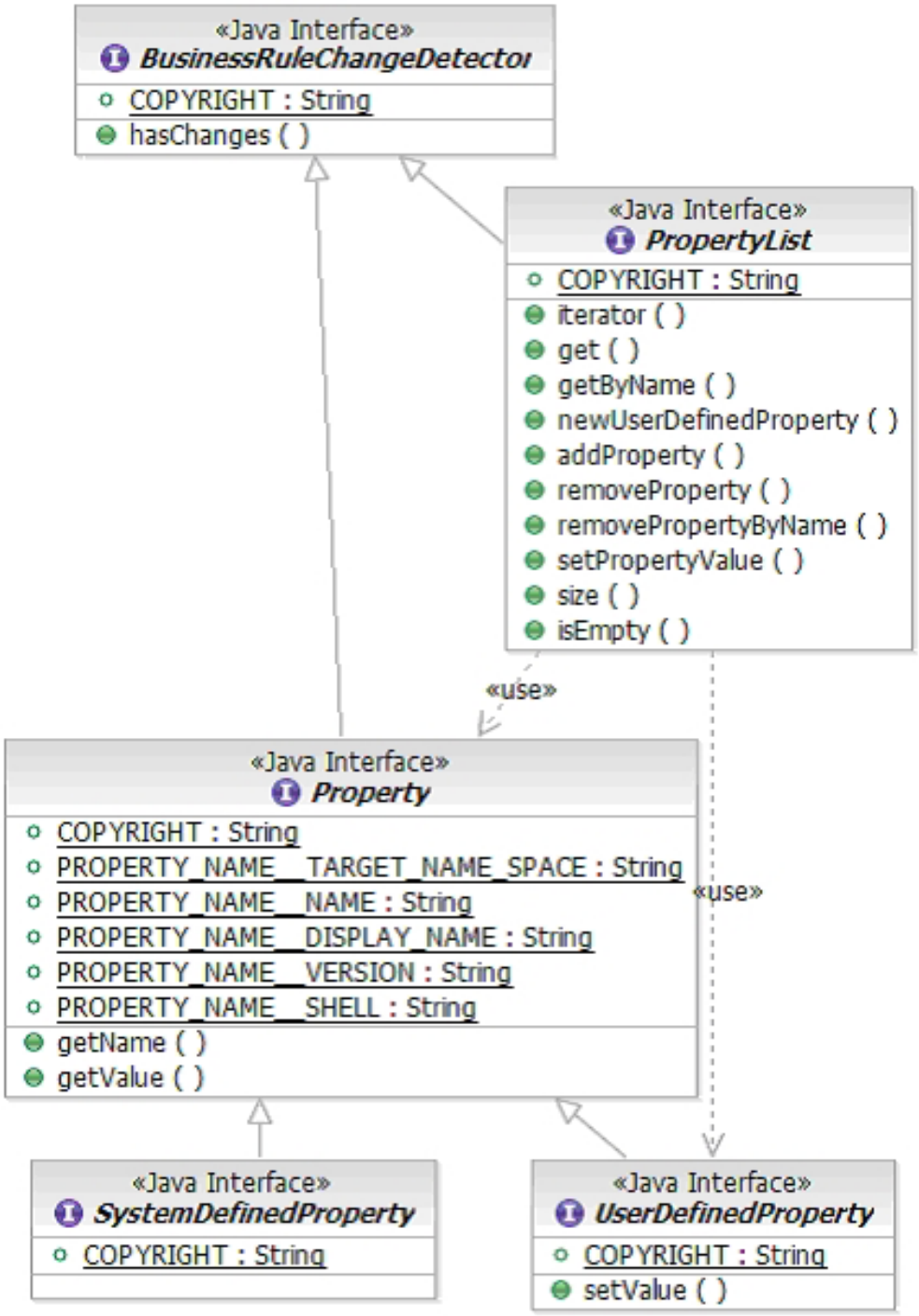


Figura 11. Diagrama de clase de Property y clases relacionadas

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 especificar 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.

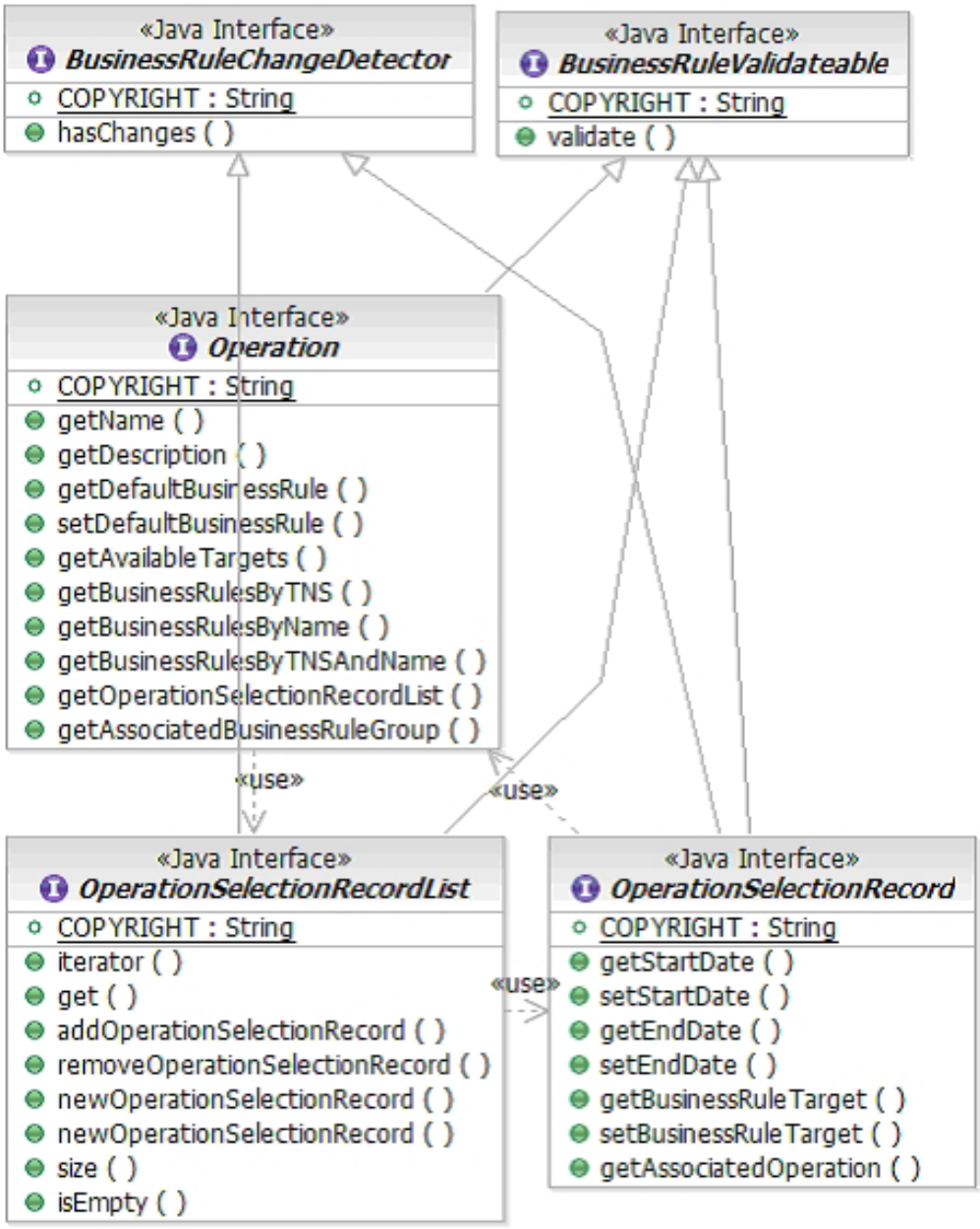


Figura 12. Diagrama de clase para Operation y clases Relacionadas

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

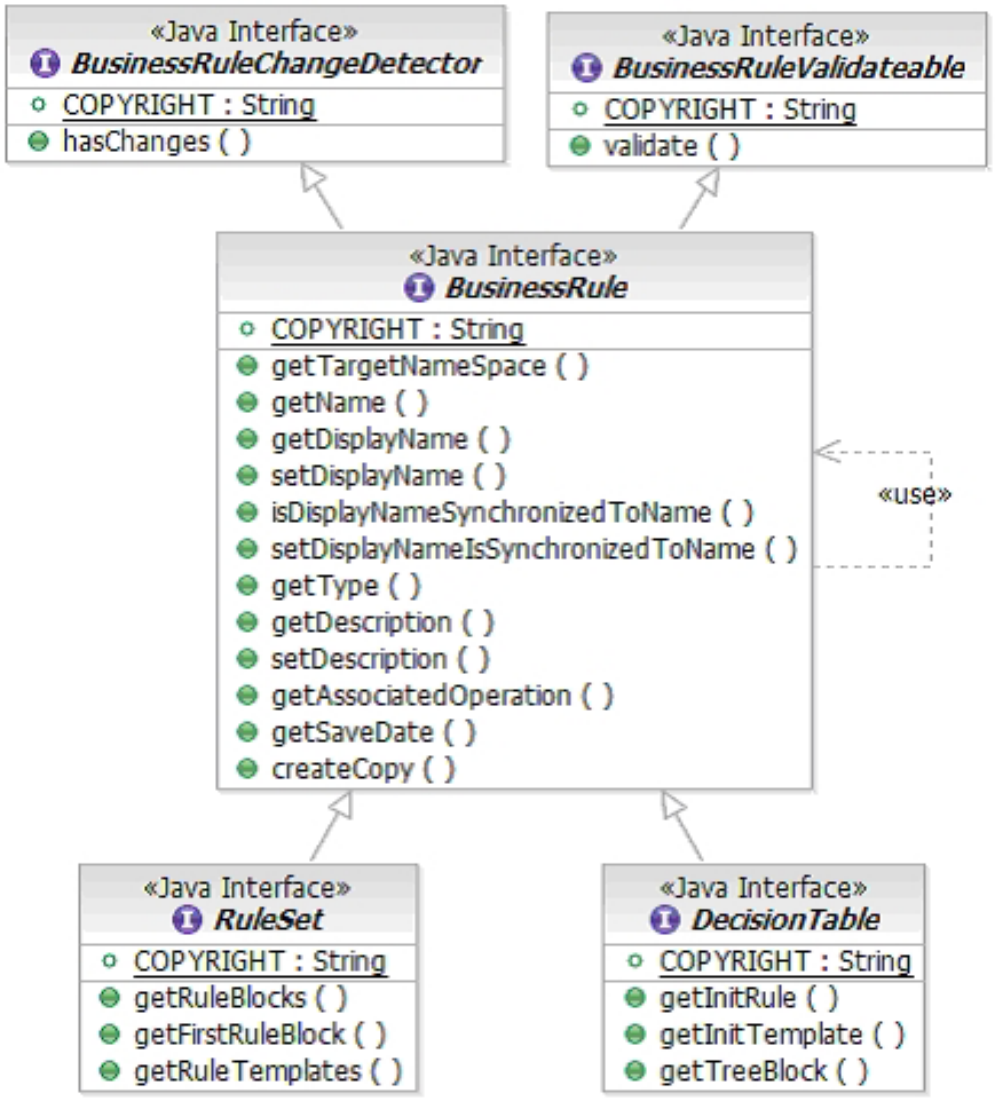


Figura 13. Diagrama de clase de BusinessRule y clases relacionadas

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

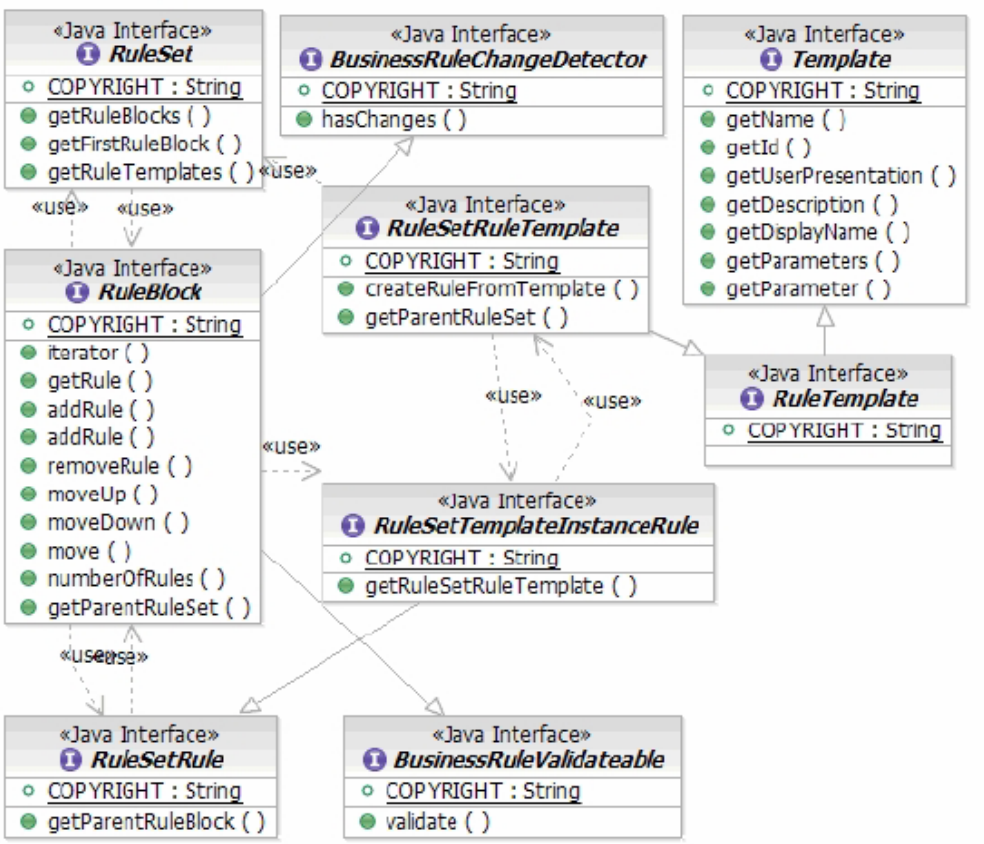


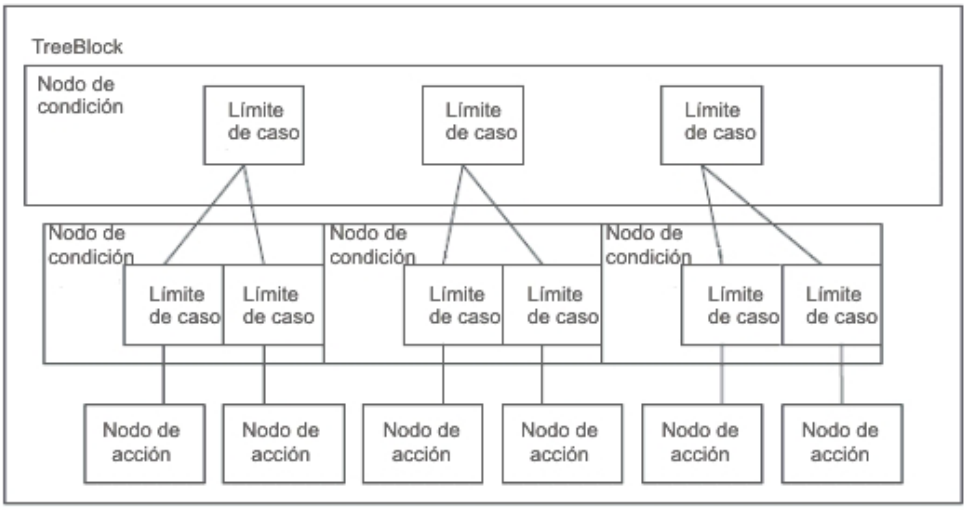
Figura 14. Diagrama de clase de BusinessRule y clases relacionadas

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

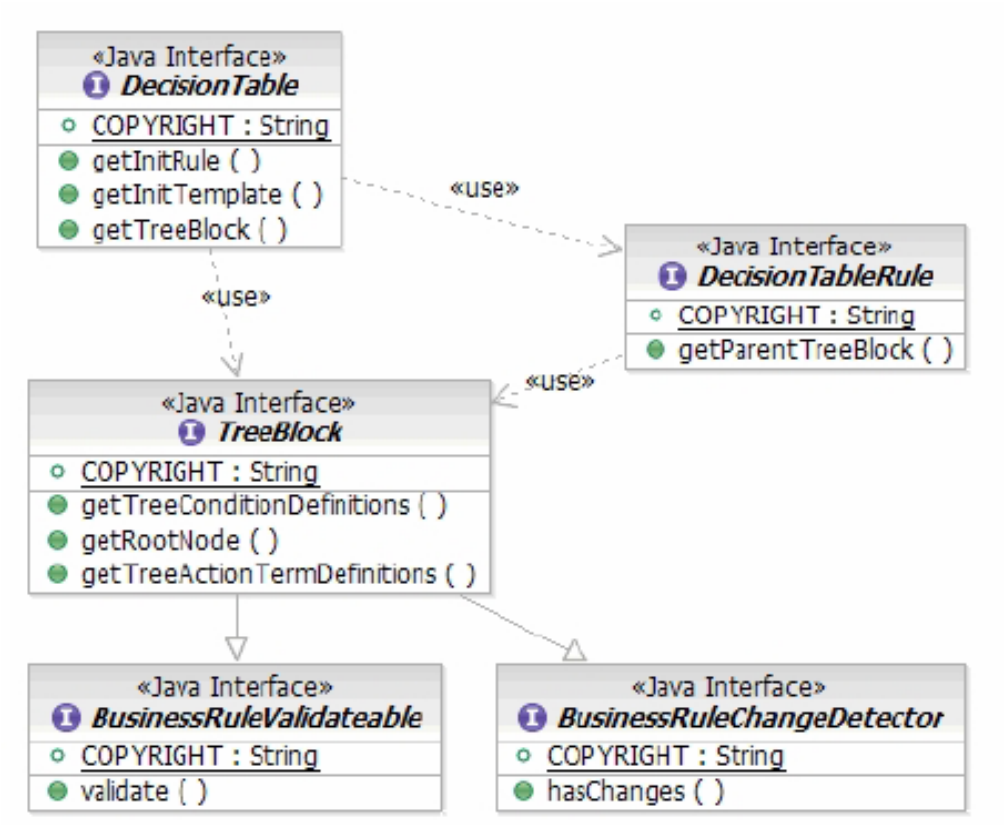
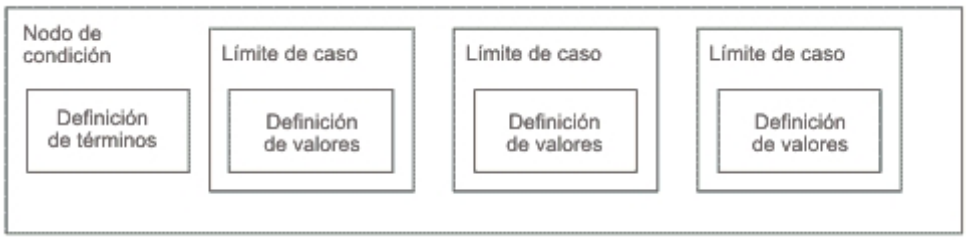


Figura 15. Diagrama de clase de DecisionTable y clases relacionadas

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 de término contiene el operando izquierdo de la expresión de 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 (`TemplateInstanceExpression`) contiene en realidad 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

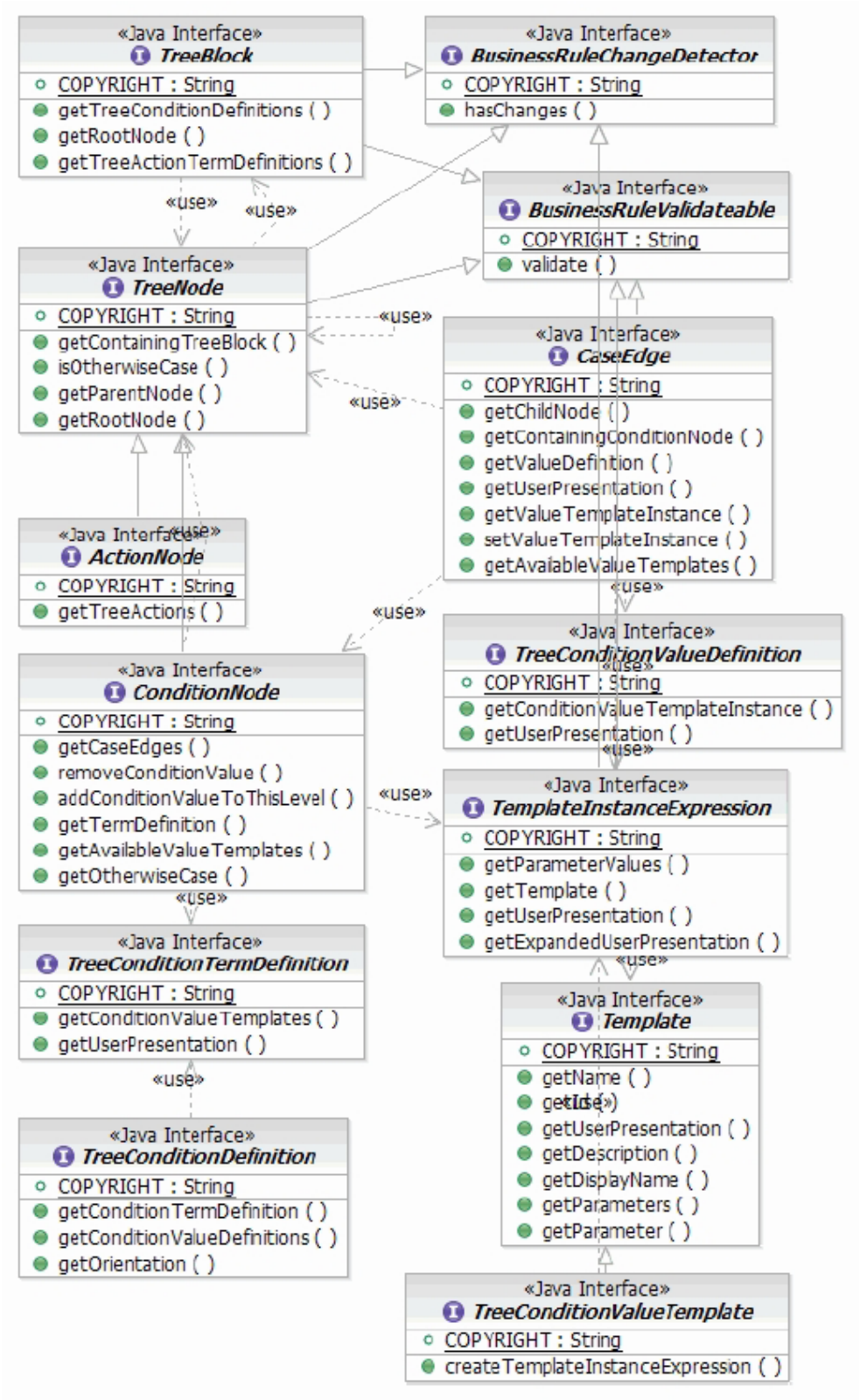


Figura 16. Diagrama de clase para TreeNode y clases relacionadas

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, entonces 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 en realidad 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.

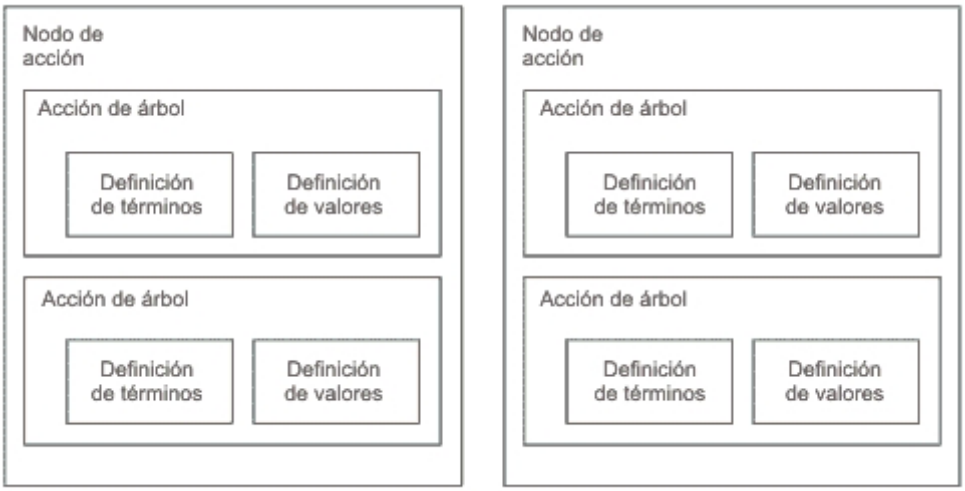
Si una definición de valor no se crea con una plantilla, no se puede cambiar. Para aplicaciones cliente, se puede utilizar la presentación de usuario en la visualización si se especificó en el momento de la creación.

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

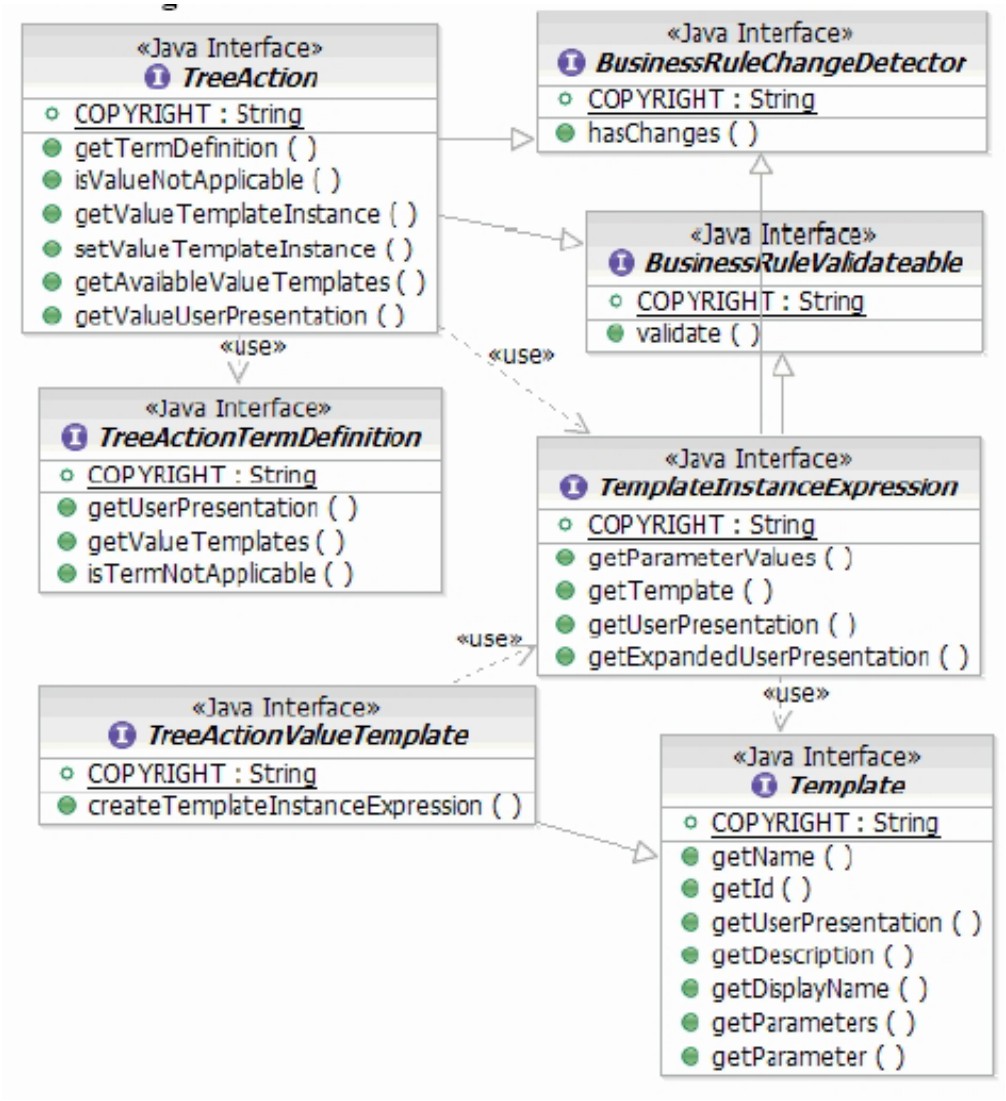


Figura 17. Diagrama de clase de TreeAction y clases relacionadas

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

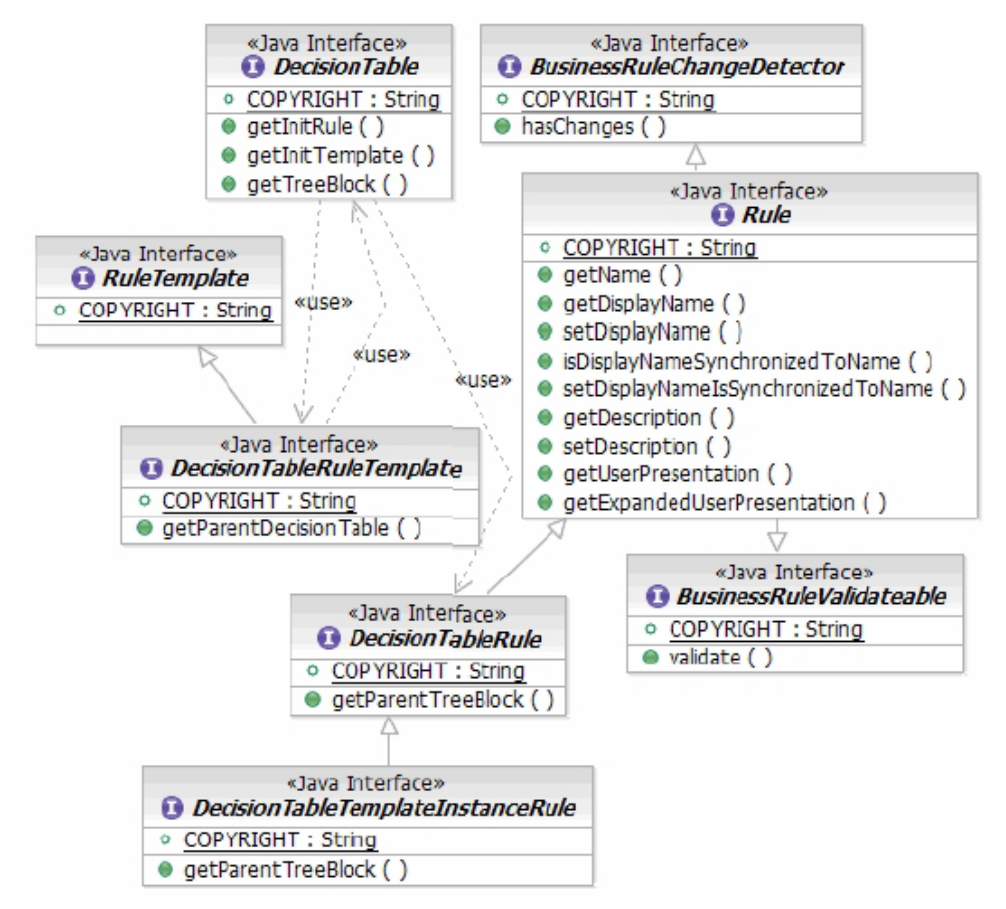


Figura 18. Diagrama de clase para DecisionTableRule y clases relacionadas

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

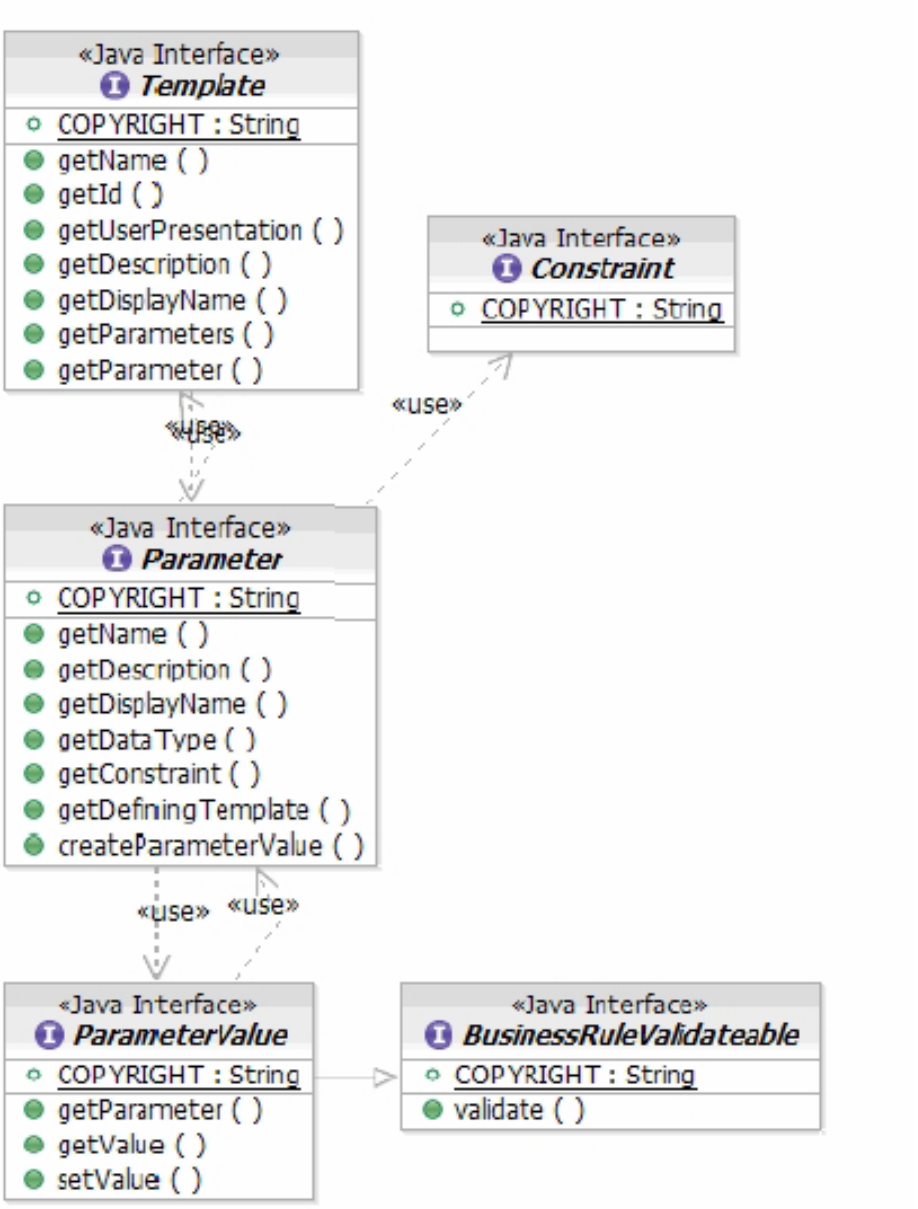


Figura 19. Diagrama de clase para Template, Parameter y clases relacionadas

Validación

Muchos de los objetos principales tienen un método de validación que permite comprobar la corrección y completitud 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

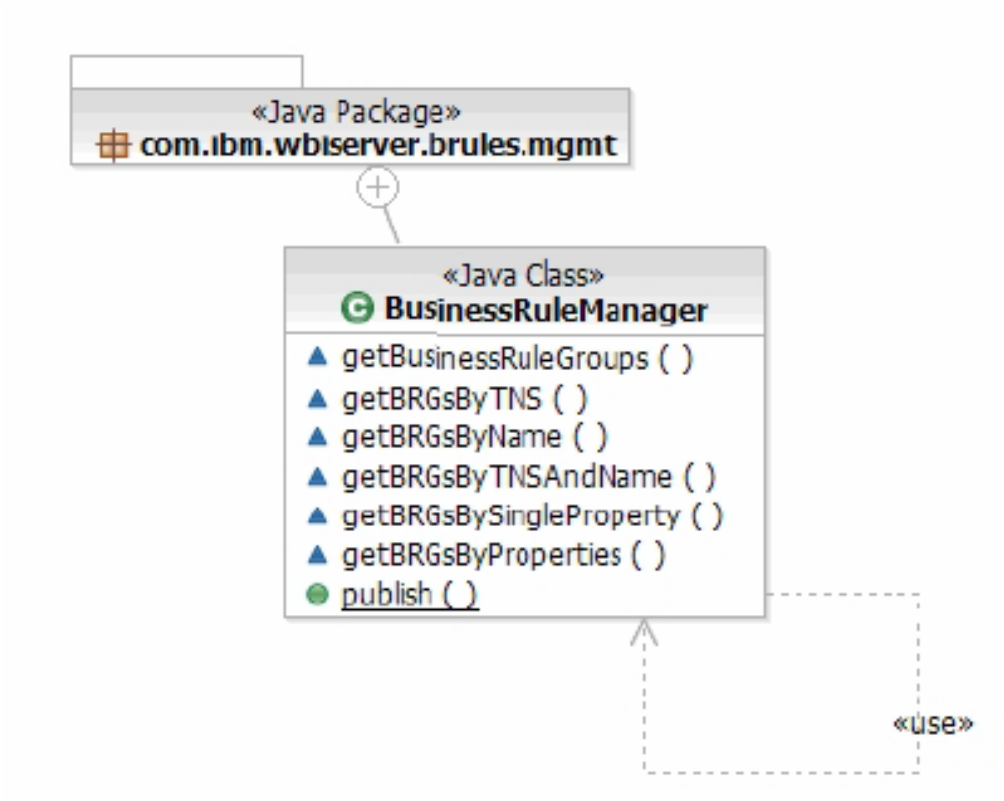


Figura 20. Diagrama de clase para BusinessRuleManager y el paquete

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, los lados derecho e izquierdo del operador deben especificarse con 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 usar en el lado izquierdo (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.

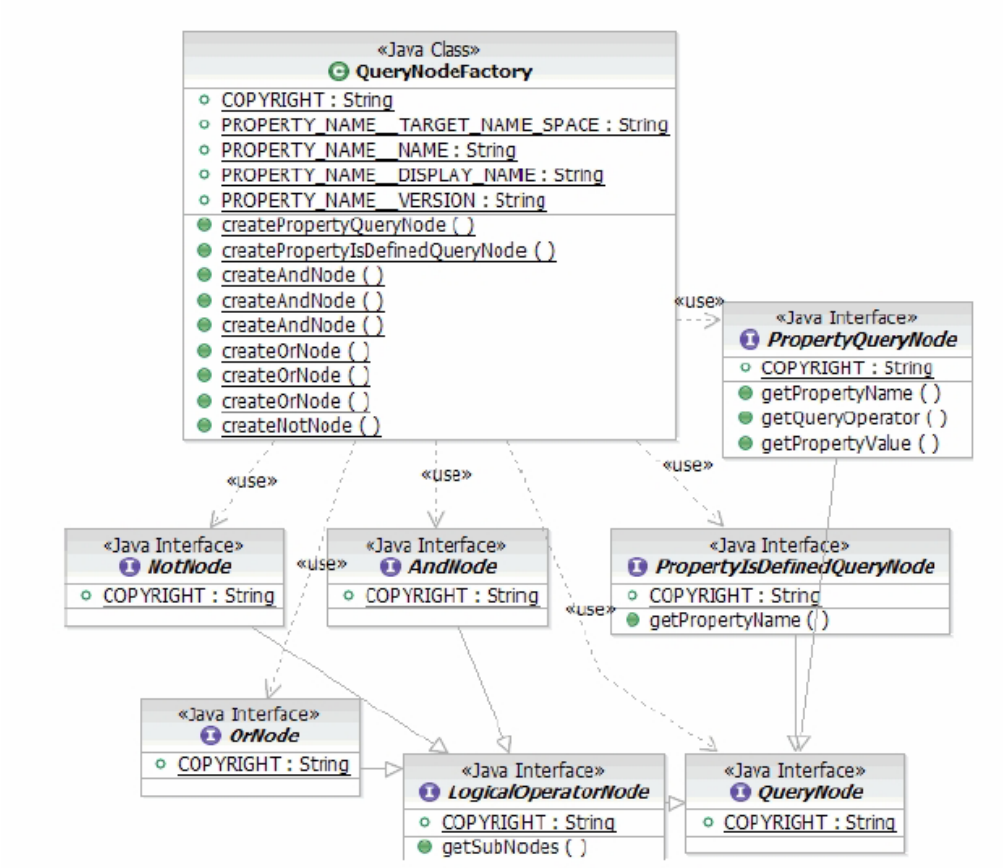


Figura 21. Diagrama de clase para QueryNodeFactory y clases relacionadas

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 los ha cambiado 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.

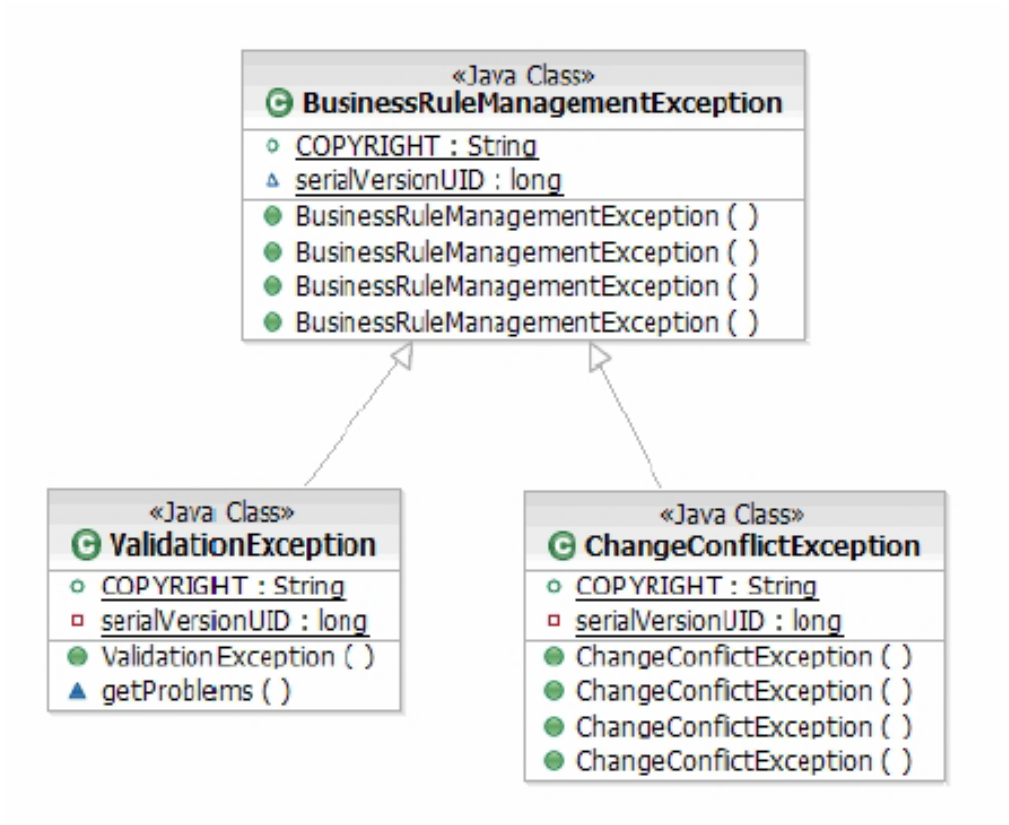


Figura 22. Diagrama de clase para BusinessRuleManagementException y clases relacionadas

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 4. 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.

Tabla 4. Problemas de grupos de normas empresariales (continuación)

Excepción	Descripción
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, deben comprobarse las fechas inicial o final de los demás registros de selección de operación para garantizar que existe 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 5. 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 de ″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 deben comprobarse antes de establecerlos en la plantilla. Puede producirse en plantillas RuleTemplate, TreeActionValueTemplate o TreeConditionValueTemplate.

Tabla 5. Problemas de conjuntos de normas y tablas de decisiones (continuación)

Excepción	Descripción
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, debe especificarse una serie que se pueda convertir al tipo del parámetro (boolean, byte, short, int, long, float y double).
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.

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_principal>:<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 a lado izquierdo, un término a lado derecho y una condición. Cada término a lado izquierdo y derecho pueden formar 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");
```

El lado izquierdo de la condición se combina con el lado derecho 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.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 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.printlnBold("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 publicar

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.println1nBold("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.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);
    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 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
// 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);
```

Validar la operación para garantizar que no existe 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 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.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. Las fechas
inicial y final también se imprimen además del nombre del artefacto de norma para la hora
planificada.
*/
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 en 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;
          }
         }
      }
      // Usar 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;
     }
    }

    // Usar 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.println1nBold("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 destino 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
// 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();
 // Obtener la expresión de condición definida con una 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. Debido a que cada acción es
// 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 actualizadas, se
      // puede publicar el grupo de normas empresariales.
      // Usar 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.println1nBold("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))
     {
      // Expresión encontrada con plantilla coincidente
      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, en realidad se 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.println1nBold("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;
```

Al utilizar nombres de plantilla exclusivos en cada nivel de nodo de condición de la tabla de decisiones, puede garantizar más fácilmente que el valor de condición se está añadiendo en el 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
 // 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. Mover arriba
 // 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 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();
  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 en 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 destino 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);
  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");

       // Establecer un valor incorrecto para este parámetro

       // Esto causará un error de validación
       parameter.setValue("$3500");
       out.println("Establecido valor de parámetro incorrecto");
       break;
     }
   }
 }
 // A este código nunca debería llegarse debido al error
```

```
    // introducido
    // anteriormente

    // Con el valor de condición y las acciones actualizadas, se
    // 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
  // llevar a cabo la acción apropiada (informar del error, corregir
  // el error, etc.)
  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 destino 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);
   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 instancia de plantilla
   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
```

```
// es recomendable comprobar si hay problemas y a continuación
// llevar a cabo la acción correcta 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("Valor incorrecto introducido para un parámetro");
   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
// 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.
```

Anexo

El anexo contiene clases adicionales que se han utilizado en los ejemplos para simplificar las operaciones comunes y ejemplos adicionales de creación de consultas complejas para buscar grupos de normas empresariales utilizando comodines.

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("Nombre
        del parámetro: " +
        parameter.getName());
        out.println("Valor del
        parámetro: "
        +
        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("Nombre
del parámetro: " +
parameter.getName());
out.println("Valor del
parámetro: "
  +
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>
  ");
  // Imprimir ConditionNode
  Otherwise
  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("Nombre
        del parámetro: " +
        parameter.getName());
        out.println("Valor
        del parámetro: "
        +
        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.println1nBold("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 expandida
  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("Nombre
           del parámetro: " +
            parameter.getName());
            out.println("Valor del
            parámetro: "
              +
            templateInstance.getPar
            ameterValue(
            parameter.getName()).ge
            tValue());
            }
          }
        }
        out.println("");
        return out.toString();
      }
}
```

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
```

Capítulo 13. Desarrollo de aplicaciones cliente para procesos empresarialesy tareas

Puede utilizar una herramienta de creación de modelos para generar y desplegar procesos empresarialesy 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.

Por qué y cuándo se efectúa 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 empresarialesy 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 empresarialesy 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, la tabla siguiente compara 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 de empresa, 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 de empresa, 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 de empresa. Utilice esta interfaz para crear clientes de mensajería para un conjunto conocido de procesos.	Esta interfaz está disponible tanto para los procesos de empresa, 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 empresariales. Si la aplicación cliente de EJB se ejecuta en la misma célula que WebSphere Process Server al que se conecta, los esquemas necesarios para los objetos empresariales 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 también se puede utilizar entre células, si la aplicación cliente se ejecuta en una instalación de servidor WebSphere Process Server completa. Sin embargo, RAL no se puede utilizar en una configuración de varias células donde la aplicación cliente se ejecuta en una instalación de cliente de 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 Java 2, Enterprise Edition (J2EE)	Seguridad de servicios Web.	La seguridad Java 2, Enterprise Edition (J2EE) para la instalación de WebSphere Process Server. También puede proteger las colas donde la aplicación cliente de JMS coloca 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.

Tareas relacionadas

"Desarrollo de aplicaciones de cliente EJB para los procesos empresariales y tareas de usuario" en la página 218
Las API de EJB proporcionan un conjunto de métodos genéricos para desarrollar aplicaciones cliente EJB para trabajar con los procesos empresarialesy tareas de usuario instalados en WebSphere Process Server.

"Desarrollo de aplicaciones cliente de la API de servicio Web" en la página 278
Puede desarrollar aplicaciones cliente que acceden a las aplicaciones de proceso empresarialy de tareas de usuario mediante las API de servicios Web.

"Desarrollo de aplicaciones cliente con la API JMS de Business Process Choreographer" en la página 306
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 empresarialesy datos de tarea

Los procesos empresarialesde 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 plantillas de las plantillas de procesos empresarialesy de tareas de usuario utilizando una interfaz de consulta.

Las interfaces de consulta EJB siguientes están disponibles con Business Process Choreographer:

	Descripción
API de consulta EJB de Business Process Choreographer	Proporciona acceso a datos de instancia y a datos de plantillas. Se puede acceder a todos los datos relacionados con procesos y tareas del sistema a través de esta interfaz. Entre los métodos relacionados en Business Flow Manager y/o Human Task Manager se incluyen los siguientes: • query • queryAll • queryProcessTemplates • queryTaskTemplates
API de tabla de consulta EJB de Business Process Choreographer	Proporciona acceso a datos de instancia y a datos de plantillas. Esta interfaz se utiliza para examinar tablas de consulta de Business Process Choreographer, que están especializadas en la consulta de procesos y listas de tareas. Los métodos relacionados de Business Flow Manager son los siguientes: • queryEntities • queryEntityCount • queryRows • queryRowCount

Una o varias de las interfaces que se muestran anteriormente puede ser la apropiada dependiendo de los clientes que accedan a datos relacionados con procesos o tareas. También hay API de servicios REST y Web disponibles en Business Process Choreographer para consultar datos lista de tareas y procesos. Sin embargo, por cuestiones de rendimiento, para consultar listas de procesos y tareas de grandes volúmenes utilice la API de tabla de consulta EJB de Business Process Choreographer.

Tablas de consulta en Business Process Choreographer

Una tabla de consulta es una definición abstracta de la información a la que se hace referencia como listas de tareas y listas de instancias de procesos, que se presenta a los usuarios que trabajan con tareas o procesos de empresa. Las tablas de consulta se pueden personalizar; por ejemplo, opciones de configuración pueden determinar que una tabla de consulta sólo contenga aquellas tareas o instancias de proceso cuyas que sean relevantes en una situación determinada. Allí donde el rendimiento sea importante, como al consultar listas de procesos y tareas de grandes volúmenes, utilice tablas de consulta.

Las tablas de consulta mejoran las vistas de base de datos predefinidas y las interfaces de consulta existentes de Business Process Choreographer en términos de funcionalidad y rendimiento.

- Las tablas de consulta se han optimizado para ejecutar consultas de lista de procesos y tareas, utilizando patrones de acceso para optimizados a efectos de rendimiento.
- Las tablas de consulta son una definición abstracta del contenido que aparece dentro de una lista de tareas o procesos. Una vez definidas, las tablas de consulta simplifican y consolidan el acceso a la información necesaria.
- Las tablas de consulta permiten una configuración de alta precisión de las opciones de autorización y filtro.
- La definición del contenido de las tablas de consulta resulta natural respecto a las listas de tareas y procesos. Por ejemplo elija, en primer lugar, la entidad en la que desee trabajar como, por ejemplo, tareas, procesos o escaladas. A continuación, elija la información adicional necesaria para visualizar la entidad como, por ejemplo, descripciones de tareas o propiedades de consultas.

Existen tres tipos de tablas de consulta: tablas de consulta predefinidas, tablas de consulta suplementarias y tablas de consulta compuestas. Todos estos tipos de tablas se consultan mediante la API de tablas de consulta. Las tablas de consulta compuestas y suplementarias se desarrollan mediante la herramienta Query Table Builder.

Tareas relacionadas

Administración de tablas de consulta
Utilice el script manageQueryTable.py para administrar tablas de consulta en Business Process Choreographer, desarrolladas mediante Query Table Builder.

Despliegue de tablas de consulta
Utilice el script manageQueryTable.py para desplegar tablas de consulta suplementarias y compuestas en Business Process Choreographer. Las tablas de consulta se despliegan en un servidor autónomo que está en funcionamiento o en un clúster que tenga al menos un miembro en funcionamiento. La anulación del despliegue de tablas de consulta suplementarias y compuestas también se realiza en servidores en funcionamiento. En las tablas de consulta suplementarias, deben crearse, si no existen ya, los objetos de base de datos físicos relacionados, normalmente una vista o tabla de base de datos, antes de poder usar la tabla de consulta.

Referencia relacionada

Vistas de base de datos para Business Process Choreographer
Esta información de referencia describe las columnas en las vistas de base de datos predefinidas.

Tablas de consulta predefinidas

Las tablas de consulta predefinidas de Business Process Choreographer son la representación de las vistas de bases de datos predefinidas de Business Process Choreographer como, por ejemplo, TASK o PROCESS_INSTANCE. Proporcionan una vista simple de los datos de la base de datos de Business Process Choreographer. No obstante, las tablas de consulta predefinidas son distintas de las vistas de base de datos predefinidas en términos de autorización, funcionalidad y rendimiento.

Cuando utilice las tablas de consulta, podrá tener una configuración de alta precisión de las opciones de autorización y filtro.

Las tablas de consulta predefinidas utilizan los mismos datos físicos subyacentes y, por tanto, tienen la misma estructura que las vistas de base de datos predefinidas. No obstante, las tablas de consulta predefinidas amplían la funcionalidad y el rendimiento de las vistas de base de datos predefinidas, ya que se han optimizado para ejecutar consultas de lista de procesos y tareas.

Las tablas de consulta predefinidas se pueden consultar directamente utilizando la API de tabla de consulta. No obstante, se recomienda utilizar las tablas de consulta para desarrollar una tabla de consulta compuesta que contenga la información que se deba recuperar posteriormente cuando se ejecute la consulta.

Puede someter parámetros cuando utilice la API de tabla de consulta para ejecutar una consulta en las tablas de consulta. Las tablas de consulta predefinidas no dan soporte a los parámetros.

Las tablas de consulta predefinidas siguientes están disponibles para consultas directas o como tablas de consulta primarias o conectadas de una tabla de consulta compuesta.

Estas tablas de consulta predefinidas contienen datos de instancia y las pueden consultar todos los usuarios autenticados:

- ACTIVITY

- ACTIVITY_ATTRIBUTE
- ACTIVITY_SERVICE
- APPLICATION_COMP
- ESCALATION
- ESCALATION_CPROP
- ESCALATION_DESC
- PROCESS_ATTRIBUTE
- PROCESS_INSTANCE
- QUERY_PROPERTY
- TASK
- TASK_CPROP
- TASK_DESC

La autorización se habilita para todos los elementos de trabajo, es decir, los tipos Todos, Individual, Grupo y Heredado. En las tablas de consulta predefinidas que tienen datos de instancia, a menos que se especifique de otro modo, la API toma como valor por omisión los elementos de trabajo Todos, Individual y Grupo.

Estas tablas de consulta predefinidas contienen datos de plantillas y las pueden consultar los usuarios administrativos que utilicen la API de tabla de consulta:

- ESC_TEMPL
- ESC_TEMPL_CPROP
- ESC_TEMPL_DESC
- PROCESS_TEMPLATE
- TASK_TEMPL
- TASK_TEMPL_CPROP
- TASK_TEMPL_DESC

Las tablas de consulta predefinidas que tienen datos de plantillas no aplican la autorización con elementos de trabajo; sólo pueden consultarlas aquellos administradores que utilicen el objeto AdminAuthorizationOptions.

Conceptos relacionados

"Tablas de consulta suplementarias"
En general, las tablas de consulta suplementarias de Business Process Choreographer exponen a la API de tabla de consulta datos procedentes de tablas de bases de datos externas o de vistas o de objetos de bases de datos. Con las tablas de consulta suplementarias, estos datos externos se pueden unir con información de instancias de procesos empresarialeso información sobre tareas de usuario. Las tablas de consulta suplementarias se pueden consultar directamente utilizando la API de tabla de consulta.

"Tablas de consulta compuestas" en la página 194
Las tablas de consulta compuestas de Business Process Choreographer están formadas por tablas de consulta predefinidas y tablas de consulta suplementarias. Combinan datos de tablas o vistas existentes. Normalmente, una tabla de consulta compuesta se utiliza para recuperar la información que se muestra en una lista de instancias de procesos o en una lista de tareas, como el caso de My To Dos.

"Desarrollo de tablas de consulta" en la página 197
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante la herramienta 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.

"Visión general de la API de tabla de consulta" en la página 200
Puede utilizar consultas basadas en entidad y basadas en fila, en la API de tabla de consulta, para ejecutar consultas contra una tabla de consulta en Business Process Choreographer.

Referencia relacionada

Vistas de base de datos para Business Process Choreographer
Esta información de referencia describe las columnas en las vistas de base de datos predefinidas.

Tablas de consulta suplementarias

En general, las tablas de consulta suplementarias de Business Process Choreographer exponen a la API de tabla de consulta datos procedentes de tablas de bases de datos externas o de vistas o de objetos de bases de datos. Con las tablas de consulta suplementarias, estos datos externos se pueden unir con información de instancias de procesos empresarialeso información sobre tareas de usuario. Las tablas de consulta suplementarias se pueden consultar directamente utilizando la API de tabla de consulta.

Las tablas de consulta suplementarias describen un objeto de la base de datos que contiene datos que son adicionales respecto a los datos que mantiene Business Process Choreographer. Normalmente, se trata de una vista de base de datos o una tabla de base de datos. La tabla de consulta suplementaria describe las columnas del objeto de base de datos relacionado. El nombre de una tabla de consulta suplementaria debe constar de un prefijo y un nombre, por ejemplo, COMPANY.EXT_DATA.

Nota: En el contexto de las tablas de consulta y la API de tabla de consulta, normalmente se hace referencia a las columnas como atributos. Puesto que el contenido de las tablas de consulta se almacena en las bases de datos, también se puede utilizar el término 'columna'.

Puede someter parámetros cuando utilice la API de tabla de consulta para ejecutar una consulta en las tablas de consulta. Las tablas de consulta suplementarias no dan soporte a los parámetros.

No se da soporte a la autorización con elementos de trabajo para las tablas de consulta suplementarias. Todos los usuarios autenticados pueden acceder al contenido de las tablas de consulta suplementarias.

Conceptos relacionados

"Tablas de consulta predefinidas" en la página 191
Las tablas de consulta predefinidas de Business Process Choreographer son la representación de las vistas de bases de datos predefinidas de Business Process Choreographer como, por ejemplo, TASK o PROCESS_INSTANCE. Proporcionan una vista simple de los datos de la base de datos de Business Process Choreographer. No obstante, las tablas de consulta predefinidas son distintas de las vistas de base de datos predefinidas en términos de autorización, funcionalidad y rendimiento.

"Tablas de consulta compuestas"
Las tablas de consulta compuestas de Business Process Choreographer están formadas por tablas de consulta predefinidas y tablas de consulta suplementarias. Combinan datos de tablas o vistas existentes. Normalmente, una tabla de consulta compuesta se utiliza para recuperar la información que se muestra en una lista de instancias de procesos o en una lista de tareas, como el caso de My To Dos.

"Desarrollo de tablas de consulta" en la página 197
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante la herramienta 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.

"Visión general de la API de tabla de consulta" en la página 200
Puede utilizar consultas basadas en entidad y basadas en fila, en la API de tabla de consulta, para ejecutar consultas contra una tabla de consulta en Business Process Choreographer.

Tablas de consulta compuestas

Las tablas de consulta compuestas de Business Process Choreographer están formadas por tablas de consulta predefinidas y tablas de consulta suplementarias. Combinan datos de tablas o vistas existentes. Normalmente, una tabla de consulta compuesta se utiliza para recuperar la información que se muestra en una lista de instancias de procesos o en una lista de tareas, como el caso de My To Dos.

Las tablas de consulta compuestas son una combinación de información que está disponible en tablas de consulta predefinidas y suplementarias. Hay varias opciones de configuración disponibles, la mayoría de las cuales influyen en los tiempos de respuesta de las consultas, para especificar la información que está disponible en una tabla de consulta compuesta.

Estructura

Las tablas de consulta compuesta constan de una tabla de consulta primaria y cero o más tablas de consulta conectadas. El nombre de una tabla de consulta compuesta debe constar de un prefijo y un nombre, por ejemplo, COMPANY.TODO_TASK_LIST.

- La tabla de consulta primaria constituye la información principal que aparece en una tabla de consulta compuesta.

Si se requiere autorización, los objetos de la tabla de consulta primaria se comprueban respecto a los elementos de trabajo disponibles, y se tienen en cuenta las opciones de autorización. Por ejemplo, la consulta se define para devolver sólo tareas de usuario de las cuales el usuario sea un propietario potencial.

Además, cada objeto de la tabla de consulta compuesta se puede identificar exclusivamente por la clave primaria de la tabla de consulta primaria. Por ejemplo, para TASK, sería el ID de tarea TKIID. Sólo las tablas de consulta predefinidas se pueden elegir como tablas de consulta primarias. Normalmente, la tabla de consulta primaria es la tabla de consulta predefinida TASK o la tabla de consulta predefinida PROCESS_INSTANCE.

- En una tabla de consulta compuesta se pueden definir cero o más tablas de consulta conectadas.

  Cada objeto que aparezca en una tabla de consulta compuesta como resultado del filtro y las opciones de autorización, así como también el filtro primario, se puede completar con información adicional que aparezca en las tablas de consulta conectadas. Por ejemplo, se pueden añadir las descripciones de tareas de un entorno local específico a una tabla de consulta compuesta con la tabla de consulta primaria TASK.

  Se debe mantener una relación de uno con uno o uno con cero, si fuera necesario utilizando criterios de selección, entre la tabla de consulta primaria y sus tablas de consulta conectadas. Las tablas de consulta conectadas puede ser tablas de consulta predefinidas y tablas de consulta suplementarias, que ya se hayan desplegado en el sistema.

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 elijan.

- 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.

Implementación

Las tablas de consulta compuestas no tienen una representación física en la base de datos. Las tablas de consulta compuestas se realizan con SQL, que se optimiza para las consultas de lista de tareas y procesos.

Autorización

Las tablas de consulta compuestas pueden configurarse para que requieran o no autorización. Si se requiere autorización, los objetos de la tabla de consulta primaria se comprueban respecto a la tabla de consulta WORK_ITEM, mediante una unión SQL, para un elemento de trabajo relacionado. Éste es el valor por omisión si la tabla de consulta primaria contiene datos de instancia como, por ejemplo, la tabla de consulta TASK o PROCESS_INSTANCE.

Si se requiere autorización, hay disponibles las opciones de autorización siguientes en la definición de una tabla de consulta compuesta:

- **elemento de trabajo Todos**: si se especifica, los objetos que tengan un elemento de trabajo Todos relacionado aparecen en la tabla de consulta compuesta.
- **elemento de trabajo Individual**: si se especifica, los objetos que tengan un elemento de trabajo Individual relacionado aparecen en la tabla de consulta compuesta.
- **elemento de trabajo Grupo**: si se especifica, los objetos que tengan un elemento de trabajo Grupo relacionado aparecen en la tabla de consulta compuesta.
- **elemento de trabajo Heredado**: si se especifica, los objetos que tengan una instancia de proceso como padre como, por ejemplo, una tarea de usuario participante, con un elemento de trabajo Todos, Individual o Grupo relacionado como configurado, aparecen en la tabla de consulta compuesta. Normalmente, los elementos de trabajo de tipo Heredado resultan útiles solo a los administradores.

Parámetros

Los parámetros se pueden utilizar en los filtros y los criterios de selección de las tablas de consulta para que las partes de los filtros definidos y los criterios de selección sean dinámicas.

Filtros

Los filtros se utilizan para restringir el contenido de una tabla de consulta:

- **filtro primario**: este filtro se define en la tabla de consulta primaria. Restringe el contenido de una tabla de consulta compuesta, utilizando las condiciones de las columnas que se hayan definido en la tabla de consulta primaria.
- **filtro de autorización**: restringe el contenido de una tabla de consulta compuesta, utilizando las columnas que se hayan definido en la tabla de consulta predefinida WORK_ITEM, que se usa para realizar la autorización. La creación de elementos de trabajo se define utilizando verbos de personal en los procesos y las tareas de usuario de Business Process Choreographer.

Nota: En el contexto de las tablas de consulta y la API de tabla de consulta, normalmente se hace referencia a las columnas como atributos. Puesto que el contenido de las tablas de consulta se almacena en las bases de datos, también se utiliza el término 'columna'.

Criterios de selección

Se debe mantener una relación de uno con uno o uno con cero, entre la tabla de consulta primaria y los objetos de una tabla de consulta conectada. Esto se lleva a cabo mediante un criterio de selección en las tablas de consulta conectadas. Por ejemplo, si TASK es la tabla de consulta primaria y TASK_DESC es la tabla de consulta conectada, normalmente el criterio de selección selecciona un entorno local específico para la descripción que se añade a la tarea de usuario en la tabla de consulta compuesta. Un ejemplo de esto es `LOCALE='en_ES'`.

Conceptos relacionados

"Tablas de consulta predefinidas" en la página 191
Las tablas de consulta predefinidas de Business Process Choreographer son la representación de las vistas de bases de datos predefinidas de Business Process Choreographer como, por ejemplo, TASK o PROCESS_INSTANCE. Proporcionan una vista simple de los datos de la base de datos de Business Process Choreographer. No obstante, las tablas de consulta predefinidas son distintas de las vistas de base de datos predefinidas en términos de autorización, funcionalidad y rendimiento.

"Tablas de consulta suplementarias" en la página 193
En general, las tablas de consulta suplementarias de Business Process Choreographer exponen a la API de tabla de consulta datos procedentes de tablas de bases de datos externas o de vistas o de objetos de bases de datos. Con las tablas de consulta suplementarias, estos datos externos se pueden unir con información de instancias de procesos empresarialeso información sobre tareas de usuario. Las tablas de consulta suplementarias se pueden consultar directamente utilizando la API de tabla de consulta.

"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 la herramienta 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.

"Visión general de la API de tabla de consulta" en la página 200
Puede utilizar consultas basadas en entidad y basadas en fila, en la API de tabla de consulta, para ejecutar consultas contra una tabla de consulta en Business Process Choreographer.

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 la herramienta 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.

La herramienta Query Table Builder está disponible como un plugin de Eclipse y se pueden 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.

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 se proporcionan para consultar una tabla de consulta compuesta, que se presupone que se ha desplegado en el sistema. En el desarrollo

de aplicaciones, se deben 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
// Bussines Flow Manager; tenga en cuenta que el inicio de EJB Business
// Flow Manager se debe almacenar en la memoria caché por cuestiones de
// rendimiento; además, se supone que existe una referencia EJB
// a EJB de Business Flow Manager local
Context ctx = new InitialContext();
LocalBusinessFlowManagerHome home =
 (LocalBusinessFlowManagerHome)
 ctx.lookup("java:comp/env/ejb/BFM");

// crear el módulo de programa del lado del cliente de Business Flow Manager
LocalBusinessFlowManager bfm = home.create();

// ************************************************
// ****************** ejemplo 1 *******************
// ************************************************

// ejecutar una consulta contra la tabla de consulta predefinida
// TASK; esto está relacionado con una lista de tareas de tipo Tareas pendientes 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

"Tablas de consulta predefinidas" en la página 191
Las tablas de consulta predefinidas de Business Process Choreographer son la representación de las vistas de bases de datos predefinidas de Business Process Choreographer como, por ejemplo, TASK o PROCESS_INSTANCE. Proporcionan una vista simple de los datos de la base de datos de Business Process Choreographer. No obstante, las tablas de consulta predefinidas son distintas de las vistas de base de datos predefinidas en términos de autorización, funcionalidad y rendimiento.

"Tablas de consulta suplementarias" en la página 193
En general, las tablas de consulta suplementarias de Business Process Choreographer exponen a la API de tabla de consulta datos procedentes de tablas de bases de datos externas o de vistas o de objetos de bases de datos. Con las tablas de consulta suplementarias, estos datos externos se pueden unir con información de instancias de procesos empresarialeso información sobre tareas de usuario. Las tablas de consulta suplementarias se pueden consultar directamente utilizando la API de tabla de consulta.

"Tablas de consulta compuestas" en la página 194
Las tablas de consulta compuestas de Business Process Choreographer están formadas por tablas de consulta predefinidas y tablas de consulta suplementarias. Combinan datos de tablas o vistas existentes. Normalmente, una tabla de consulta compuesta se utiliza para recuperar la información que se muestra en una lista de instancias de procesos o en una lista de tareas, como el caso de My To Dos.

"Visión general de la API de tabla de consulta"
Puede utilizar consultas basadas en entidad y basadas en fila, en la API de tabla de consulta, para ejecutar consultas contra una tabla de consulta en Business Process Choreographer.

Tareas relacionadas

Administración de tablas de consulta
Utilice el script manageQueryTable.py para administrar tablas de consulta en Business Process Choreographer, desarrolladas mediante Query Table Builder.

Despliegue de tablas de consulta
Utilice el script manageQueryTable.py para desplegar tablas de consulta suplementarias y compuestas en Business Process Choreographer. Las tablas de consulta se despliegan en un servidor autónomo que está en funcionamiento o en un clúster que tenga al menos un miembro en funcionamiento. La anulación del despliegue de tablas de consulta suplementarias y compuestas también se realiza en servidores en funcionamiento. En las tablas de consulta suplementarias, deben crearse, si no existen ya, los objetos de base de datos físicos relacionados, normalmente una vista o tabla de base de datos, antes de poder usar la tabla de consulta.

Visión general de la API de tabla de consulta

Puede utilizar consultas basadas en entidad y basadas en fila, en la API de tabla de consulta, para ejecutar consultas contra una tabla de consulta en Business Process Choreographer.

Una consulta se ejecuta solamente en una tabla de consulta específica. La relación entre varias tablas de consulta, es decir, en términos de la API de consulta estándar, vistas de base de datos, se define con tablas de consulta compuestas.

En la API de tabla de consulta, hay disponibles dos conceptos distintos para ejecutar consultas contra una tabla de consulta en Business Process Choreographer:

- **Consultas basadas en entidad**: las consultas de entidad, que utilizan los métodos queryEntities y queryEntityCount, presuponen que una tabla de consulta contiene entidades identificables de forma exclusiva, tal como se define en la tabla de consulta primaria. Estas entidades se identifican mediante la clave primaria de la tabla de consulta primaria.
- **Consultas basadas en fila**: las consultas de fila, que utilizan los métodos queryRows y queryRowCount, devuelven un conjunto de resultados como JDBC. La misma entidad, por ejemplo, una tarea de usuario que se identifique por su ID de tarea como, por ejemplo, TKIID, puede aparecer varias veces en el conjunto de resultados.

Los conjuntos de resultados de entidades, que devuelve el método queryEntities, resaltan la semántica de la tabla de consulta primaria en una tabla de consulta compuesta. Las tablas de consulta compuesta constan de una tabla de consulta primaria y cero o más tablas de consulta conectadas. 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.

Cada entidad es exclusiva dentro de una tabla de consulta y, por tanto, cada entidad es exclusiva dentro de su conjunto de resultados de la entidad. La exclusividad de una entidad se mantiene mediante la clave primaria de los datos subyacentes. Por ejemplo, para la entidad TASK, sería el ID de tarea TKIID.

Los conjuntos de resultados de fila, que devuelve el método queryRows, constan de las filas que devuelve la consulta JDBC que se ejecuta contra las vistas y tablas de base datos subyacentes. Se puede comparar con QueryResultSet, que devuelve la API de consulta estándar. En general, el número de filas es mayor que el número de entidades que contiene una tabla de consulta. Un conjunto de resultados de fila puede contener entradas duplicadas para una tarea específica, por ejemplo, si se selecciona WI.REASON en la consulta.

Visión general de la API de tabla de consulta

Cada método de API de tabla de consulta tiene los parámetros siguientes:

- **String queryTableName**: el nombre de la tabla de consulta que se 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 valor *prefijo.nombre* .
- **FilterOptions filterOptions**: las opciones que restringen el conjunto de resultados y mediante el cual puede especificar los criterios de clasificación.
- **AuthorizationOptions authOptions**: las opciones que especifican los elementos de trabajo que deben tenerse en cuenta; las consultas en nombre de otro usuario; las consultas administrativas que pueden ejecutarse utilizando AdminAuthorizationOptions.
- **List parameters**: las tablas de consulta compuestas pueden definirse con parámetros en los filtros y los criterios de selección; los valores para dichos parámetros se especifican con este argumento.

FilterOptions

- **distinct**: este valor es efectivo sólo cuando se ejecuta una consulta basada en fila. Si se establece en true, de devuelven filas distintas.

- **Locale**: el entorno local se puede utilizar como un parámetro del sistema en un filtro o criterio de selección, por ejemplo, `'LOCALE=$LOCALE'`. Si no se establece, se utiliza el entorno local del servidor.
- **TimeZone**: se utiliza para convertir fechas como, por ejemplo, CREATED en la tabla de consulta predefinida TASK. Si no se especifica, se utiliza el huso horario del servidor.
- **threshold**: limita el número de filas o entidades que se devuelven. El valor de umbral puede no ser preciso para las consultas basadas en entidad.
- **skip count**: especifica el número de filas o entidades que deben saltarse en el conjunto de resultados.
- **selected attributes**: es una lista de atributos, separada por comas, que especifica los atributos que debe recuperar la consulta. Si se requiere autorización como, por ejemplo, en las tablas de consulta predefinidas que contienen datos de instancia, se puede recuperar la información de elemento de trabajo, que se indica con el prefijo 'WI.', por ejemplo, WI.REASON, además de los atributos definidos. Si no se especifica ninguno de los atributos seleccionados, se devuelven todos los atributos definidos en la tabla de consulta, y no se devuelve ninguna información de elemento de trabajo.

  Nota: En el contexto de las tablas de consulta y la API de tabla de consulta, normalmente se hace referencia a las columnas como atributos. Puesto que el contenido de las tablas de consulta se almacena en las bases de datos, también se puede utilizar el término 'columna'.
- **query condition**: lleva a cabo filtrado adicional en el conjunto de resultados. Se puede hacer referencia a los atributos definidos en la tabla de consulta, si se ha establecido que se requiere autorización. También se puede hacer referencia a las columnas definidas en la tabla de consulta WORK_ITEM utilizando el prefijo 'WI.', por ejemplo, WI.REASON=REASON_POTENTIAL_OWNER.
- **sort attributes**: es una lista de atributos, separada por comas, que define el criterio de clasificación, por ejemplo, CREATED DESC.

AuthorizationOptions

- **everybody**: si se establece en true, que es el valor por omisión, se tienen en cuenta los elementos de trabajo de tipo Todos (que es el verbo de personal Todos) si se han habilitado en la tabla de consulta.
- **individual**: si se establece en true, que es el valor por omisión, se tienen en cuenta los elementos de trabajo de tipo Individual, por ejemplo, el verbo de personal "Usuarios", si se han habilitado en la tabla de consulta.
- **groups**: si se establece en true, que es el valor por omisión, se tienen en cuenta los elementos de trabajo de tipo Grupo, por ejemplo, el verbo de personal "Grupo", si se han habilitado en la tabla de consulta y en el Contenedor de tareas de usuario.
- **inherited**: si se establece en true, se tienen en cuenta los elementos de trabajo de tipo Heredado, por ejemplo, el administrador de una instancia de proceso puede ver, en ese caso, las instancias de tarea de usuario participante creadas para dicha instancia de proceso, si ésta ejecuta una consulta contra dicha tabla de consulta.

En lugar de un objeto AuthorizationOptions, se puede pasar un objeto AdminAuthorizationOptions a la API de tabla de consulta, que sólo está disponible si el llamante tiene el rol de J2EE BPESystemAdministrator. La clase AdminAuthorizationOptions se deriva de la clase AuthorizationOptions. Hay disponibles las opciones siguientes:

- **onBehalfUser**: si se establece en null, que es el valor por omisión, y la consulta se ejecuta contra una tabla de consulta que requiera autorización: la consulta se ejecuta sin limitar los resultados que utilicen autorización, que se basa en elementos de trabajo, para este usuario específico. Es decir, la consulta devuelve todos los objetos que contiene la tabla de consulta.
- **onBehalfUser**: si se establece en null, que es el valor por omisión, y la consulta se ejecuta contra una tabla de consulta que no requiera autorización: cualquier usuario autenticado ve todo el contenido de la tabla de consulta. El conjunto de resultados de la consulta es la misma independientemente de si se utiliza o no AuthorizationOptions o AdminAuthorizationOptions.
- **onBehalfUser**: si se establece en un nombre de usuario concreto: la consulta se ejecuta en nombre del usuario especificado.
- **onBehalfUser**: si se utiliza para una tabla de consulta predefinida y se consultan los datos de plantilla, onBehalfUser debe establecerse en null.

Parámetros

Las tablas de consulta compuestas pueden definirse con los parámetros que aparecen en filtros o en criterios de selección. Todo parámetro que sea necesario para ejecutar la consulta debe pasarse como parámetro, de la clase com.ibm.bpe.Parameter, a la API de tabla de consulta que aparece en una lista, java.util.List.

Lenguaje de condición de tabla de consulta (QTCL)

El Lenguaje de condición de tabla de consulta (QTCL) se utiliza para especificar filtros y criterios de selección. Utilice este lenguaje definido claramente para especificar condiciones basadas en los atributos de las tablas de consulta. En esta sección se describen las especificaciones del QTCL para la API de tabla de consulta. Para obtener una especificación completa, 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.

Nota: En el contexto de las tablas de consulta y la API de tabla de consulta, normalmente se hace referencia a las columnas como atributos. No obstante, puesto que el contenido de las tablas de consulta se almacena en las bases de datos, también se puede utilizar el término 'columna'.

- Una subexpresión de una expresión QTCL consta de un operando de lado izquierdo, una operación y un operando de lado derecho, o una lista de operandos. Además, están disponibles los operadores unarios como, por ejemplo, IS NULL.
- El operando de lado izquierdo es un nombre de atributo de una tabla de consulta.
- El operando de lado derecho es una constante que se define en el atributo del operando de lado izquierdo, o un literal.
- Una expresión QTCL se ejecuta en un ámbito dado que determina los atributos que son válidos en el lado izquierdo de una expresión. Las condiciones de consulta, o filtros de consulta, se ejecutan en el ámbito de la tabla de consulta en la que se ejecuta la consulta. Los atributos válidos en el lado izquierdo de la expresión son los atributos de la tabla de consulta. Si se requiere autorización en la tabla de consulta, también son válidos los atributos de la tabla de consulta WORK_ITEM, que aparecen con el prefijo 'WI.'.

- Los operadores válidos son: `<`, `>`, `<>`, `<=`, `>=`, `=`, `IN`, `NOT IN`, `IS NULL`, `IS NOT NULL`, `LIKE`, `IS NOT LIKE`.
- Las subexpresiones se conectan mediante `AND` y `OR` con su semántica, bien conocida. Los paréntesis se utilizan para agrupar las subexpresiones. El ejemplo siguiente se ha diseñado 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)'`.

Detalles de resultados de consulta para consultas de recuentos

Los métodos de API de tabla de consulta queryEntityCount y queryRowCount devuelven un valor entero simple. La implementación se ha optimizado para obtener un buen rendimiento al recuperar el número de objetos cualificados.

Detalles de resultados de consulta para EntityResultSet, que devuelve el método queryEntities:

El método queryEntities devuelve los siguientes detalles de resultados de consulta para EntityResultSet.

- El nombre de la tabla de consulta se puede recuperar en EntityResultSet. Es el nombre de la tabla de consulta en la que se ejecuta la consulta.
- Se puede recuperar el nombre del tipo de entidad. Es el nombre de la tabla de consulta primaria, si la consulta se ejecuta en una tabla de consulta compuesta. En caso contrario, es el nombre de la tabla de consulta en la que se ejecuta la consulta.
- Se puede recuperar un objeto EntityInfo. El objeto EntityInfo describe los detalles de las entidades que contiene EntityResultSet. Éstos son los atributos, sus tipos relacionados, así como también el tipo de entidad.
- Una lista de entidades en el orden especificado, si así se define en FilterOptions.
- Se recupera el número de entidades de EntityResultSet mediante el método size() en la lista de entidades.

Detalles de resultados de consulta para RowResultSet, que devuelve el método queryRows

El método queryRows devuelve los siguientes detalles de resultados de consulta para RowResultSet.

- El nombre de la tabla de consulta se puede recuperar en RowResultSet. Es el nombre de la tabla de consulta en la que se ejecuta la consulta.
- Se puede recuperar el nombre de la tabla de consulta primaria. Es el nombre de la tabla de consulta primaria, si la consulta se ejecuta en una tabla de consulta compuesta, en caso contrario, es el nombre de la tabla de consulta en la que se ejecuta la consulta.
- Se puede recuperar una lista de atributos y sus tipos relacionados.
- Se puede navegar por RowResultSet con los métodos next(), previous(), first() y last(), en el orden especificado, si así se ha definido en FilterOptions.
- Se puede recuperar el tamaño de RowResultSet.

Conceptos relacionados

"Tablas de consulta predefinidas" en la página 191
Las tablas de consulta predefinidas de Business Process Choreographer son la representación de las vistas de bases de datos predefinidas de Business Process Choreographer como, por ejemplo, TASK o PROCESS_INSTANCE. Proporcionan una vista simple de los datos de la base de datos de Business Process Choreographer. No obstante, las tablas de consulta predefinidas son distintas de las vistas de base de datos predefinidas en términos de autorización, funcionalidad y rendimiento.

"Tablas de consulta suplementarias" en la página 193
En general, las tablas de consulta suplementarias de Business Process Choreographer exponen a la API de tabla de consulta datos procedentes de tablas de bases de datos externas o de vistas o de objetos de bases de datos. Con las tablas de consulta suplementarias, estos datos externos se pueden unir con información de instancias de procesos empresarialeso información sobre tareas de usuario. Las tablas de consulta suplementarias se pueden consultar directamente utilizando la API de tabla de consulta.

"Tablas de consulta compuestas" en la página 194
Las tablas de consulta compuestas de Business Process Choreographer están formadas por tablas de consulta predefinidas y tablas de consulta suplementarias. Combinan datos de tablas o vistas existentes. Normalmente, una tabla de consulta compuesta se utiliza para recuperar la información que se muestra en una lista de instancias de procesos o en una lista de tareas, como el caso de My To Dos.

"Desarrollo de tablas de consulta" en la página 197
Las tablas de consulta suplementarias y compuestas de Business Process Choreographer se crean durante el desarrollo de una aplicación, mediante la herramienta 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.

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 empresarialesy 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 los usuarios que tienen uno de estos roles de J2EE: BPESystemAdministrator, TaskSystemAdministrator, BPESystemMonitor o TaskSystemMonitor pueden llamar al método queryAll. 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:

- De que alguien con uno de los roles de J2EE haya invocado la consulta.
- 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 consutla cuando defina la consulta almacenada. El criterio se aplica dinámicamente cuando se ejcuta 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 parecida 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 6.

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. Resulta aconsejable especificar las columnas que están en la cláusula select.

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 consula en los escenarios de producción pueden contener miles o incluso millones de elementos, se recomienda

especificar siempre un umbral. 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 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.

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.

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 227
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).

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 llamante 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
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.

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 empresarialesy tareas de usuario instalados en WebSphere Process Server.

Por qué y cuándo se efectúa 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 empresarial

- 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

"Comparación de las interfaces de programación para interactuar con procesos empresarialesy tareas de usuario" en la página 187
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 empresarialesy tareas de usuario. Cada una de estas interfaces tiene características diferentes.

Referencia relacionada

Vistas de base de datos para Business Process Choreographer
Esta información de referencia describe las columnas en las vistas de base de datos predefinidas.

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 empresarialesy aplicaciones de tareas acceden al enterprise bean de sesión adecuado mediante la interfaz inicial del bean.

Por qué y cuándo se efectúa esta tarea

La interfaz BusinessFlowManagerService proporciona los métodos para las aplicaciones de proceso empresarialy 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 empresarialeso tareas de usuario accede a la interfaz remota del bean de sesión a través de la interfaz inicial remota del bean.

Por qué y cuándo se efectúa 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 de cliente J2EE (Java 2 Platform, Enterprise Edition)
   - 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. Empaquete los módulos de programa generados con la aplicación.
   a. Para las aplicaciones de proceso, empaquete el archivo *<raíz_instalación>*/ProcessChoreographer/client/bpe137650.jar con el archivo EAR (Enterprise Archive) de la aplicación.
   b. Para las aplicaciones de tarea, empaquete el archivo *<raíz_instalación>*/ProcessChoreographer/client/task137650.jar con el archivo EAR de la aplicación.
   c. Establezca el parámetro **Classpath** del archivo manifest del módulo de la aplicación para que incluya el archivo JAR.

El módulo de aplicación puede ser una aplicación J2EE, una aplicación Web o una aplicación EJB.

3. Decida cómo va a proporcionar definiciones de objetos empresariales.

   Para trabajar con objetos empresariales en una aplicación de cliente remoto, debe tener acceso a los esquemas correspondientes para los objetos empresariales (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 de J2EE, 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 gestionado de J2EE, empaquete los archivos con el archivo EAR de la aplicación cliente, o bien saque partido del cargador de artefactos remotos.
     a. Utilice los métodos createMessage y ClientObjectWrapper.getObject de la API de EJB de Business Process Choreographer para cargar las definiciones de objetos empresariales remotos 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 empresarial como parte de un objeto empresarial 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 el valor para una propiedad del objeto empresarial que se escribe utilizando el esquema XML any o anyType, utilice los servicios del 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 del 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 empresarial.

        ```
        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();
           }
        ```

4. 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.

5. 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 del principal del proceso que efectúa la llamada, la lista de miembros del grupo e indica si dicho proceso tiene uno de los roles J2EE 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`.

6. 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 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 empresarialeso tareas de usuario accede a la interfaz local del bean de sesión a través de la interfaz inicial local del bean.

Por qué y cuándo se efectúa 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 de cliente J2EE (Java 2 Platform, Enterprise Edition)
   - 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 del principal del proceso que efectúa la llamada, la lista de miembros del grupo e indica si dicho proceso tiene uno de los roles J2EE 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);
```

Consulta sobre los objetos de procesos empresarialesy relativos a tareas

Las aplicaciones cliente trabajan con objetos de procesos empresarialesy relacionados con tareas. Puede consultar objetos de proceso de emprea y relativos a tareas en la base de datos para recuperar propiedades específicas de estos objetos.

Por qué y cuándo se efectúa esta tarea

Durante la configuración de Business Process Choreographer, una base de datos relacional se asocia con el contenedor de procesos empresarialesy con el contenedor de tareas. La base de datos almacena todos los datos de plantillas (modelo) e instancias (tiempo de ejecución) para gestionar procesos empresarialesy tareas. Utilice una sintaxis análoga a SQL para consultar estos datos.

Puede realizar una consulta específica para recuperar una propiedad concreta de un objeto. También puede guardar consultas que utilice a menudo e incluir estas consultas almacenadas en la aplicación.

Referencia relacionada

Vistas de base de datos para Business Process Choreographer
Esta información de referencia describe las columnas en las vistas de base de datos predefinidas.

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.

Por qué y cuándo se efectúa 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.

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).

Por qué y cuándo se efectúa 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" en la página 210
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.

Gestión de consultas almacenadas públicas:

Las consultas almacenadas públicas las crea el administrador del sistema. Estas consultas están disponibles para todos los usuarios.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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 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.

Por qué y cuándo se efectúa esta tarea

Un proceso empresarialpuede ser un microflujo o un proceso de larga ejecución:

- Los microflujos son procesos empresarialesde 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 llamante pueda realizar todas las acciones de un proceso. El llamante 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 llamante		
	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

Roles necesarios para acciones en actividades de procesos de empresa

Acceder a la interfaz BusinessFlowManager no garantiza que el llamante pueda realizar todas las acciones de una actividad. El llamante 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 llamante				
	Lector	Editor	Propietario potencial	Propietario	Administrador
cancelClaim				x	x

Acción	Rol principal del llamante				
	Lector	**Editor**	**Propietario potencial**	**Propietario**	**Administrador**
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
transferWorkItem				x Sólo para propietarios potenciales o administra-dores	x

Gestión del ciclo de vida de un proceso empresarial

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.

Por qué y cuándo se efectúa 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 empresarialdepende 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.EXCECUTION_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.EXCECUTION_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 de empresa:

Puede suspender instancias de proceso de nivel superior de larga ejecución mientras se están ejecutando y reanudarlas de nuevo para completarlas.

Antes de empezar

El llamante debe ser un administrador de la instancia de proceso o un administrador de procesos de empresa. Para suspender una instancia de proceso, debe estar en el estado de ejecución o anómalo.

Por qué y cuándo se efectúa 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.

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. 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. También

se suspenden las tareas en línea asociadas a esta instancia de proceso, pero no se suspenden las tareas autónomas asociadas a esta instancia de proceso.

En este estado, las actividades que se inician pueden finalizase pero no se activan actividades nuevas, por ejemplo, se puede completar una actividad de tareas de usuario en estado reclamado.

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 de empresa:

Puede reiniciar una instancia de proceso que esté en estado finalizado, terminado, anómalo o compensado.

Antes de empezar

El llamante debe ser un administrador de la instancia de proceso o un administrador de procesos de empresa.

Por qué y cuándo se efectúa 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.

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, es necesario que un usuario que tenga autorización de administrador de procesos, termine una instancia de proceso de nivel superior de la que se sabe que está en estado irrecuperable. Dado que una instancia de proceso termina inmediatamente, sin esperar a subprocesos o actividades pendientes, debe terminar una instancia de proceso sólo 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.

Por qué y cuándo se efectúa 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 empresarialesse 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.

Por qué y cuándo se efectúa 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.

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. La API completeAndClaimSuccessor admite el proceso de este tipo de flujo de trabajo.

Por qué y cuándo se efectúa esta tarea

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. También se conoce este tipo de flujo de trabajo como *flujo de página* porque se asocian las definiciones de interfaz de usuario a las actividades para controlar el flujo de los diálogos de la interfaz de usuario.

La API completeAndClaimSuccessor completa una 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. Reclame la primera actividad de la secuencia de actividades.

```
//
//Consultar la lista de actividades que el usuario conectado puede reclamar
//
QueryResultSet result =
     process.query("ACTIVITY.AIID",
                   "PROCESS_INSTANCE.NAME = 'CustomerOrder' AND
                    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);
...
//
//Reclamar la primera actividad
//
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.

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 273
Algunos flujos de trabajo sólo los realiza un usuario, por ejemplo pedir libros de una librería en línea. En este ejemplo se muestra cómo implementar la secuencia de acciones para pedir el libro como una serie de actividades de tareas de usuario (tareas a realizar). 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.

Por qué y cuándo se efectúa 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 empresarial 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

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.

Por qué y cuándo se efectúa 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.

Interfaz BusinessFlowManagerService

La interfaz BusinessFlowManagerService expone funciones de proceso empresarialque 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 empresarialse 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 de empresa. Se puede crear la instancia e iniciarse emitiendo las peticiones correspondiente, por ejemplo, sendMessage(). La ejecución de la instancia de proceso la dirige automáticamente el servidor.

Tabla 7. 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 8. Los métodos API 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.
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 9. Métodos API para controlar el ciclo de vida de las instancias de proceso

Método	**Descripción**
suspender	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.

Tabla 9. Métodos API para controlar el ciclo de vida de las instancias de proceso (continuación)

Método	Descripción
query	Recupera las propiedades de la base de datos que cumplen los criterios de búsqueda.

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 10. 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.
query	Recupera las propiedades de la base de datos que cumplen 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 11. 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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 llamante.

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 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.

Por qué y cuándo se efectúa 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* en la API) e iniciar la instancia de tarea.

Procedimiento

1. Opcional: Liste las plantillas de tarea para encontrar el nombre de la tarea de colaboración 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_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 de 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 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.

Por qué y cuándo se efectúa 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.

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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

- 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.

Por qué y cuándo se efectúa esta tarea

Tal vez desee hacerlo, por ejemplo, cuando la definición de tarea no esté disponible cuando se despliegue la aplicación, las tareas que formen parte de un flujo de trabajo no se conozcan todavía, o se necesite una tarea para cubrir colaboraciones ad-hoc entre un grupo de personas.

Puede modelar ad-hoc tareas a realizar o de colaboración mediante la creación de instancias de la clase com.ibm.task.api.TaskModel y su utilización para crear una

plantilla de tarea reutilizable, o bien para 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 del Gestor de tareas de usuario 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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 empresarialy utiliza una interfaz de la aplicación de proceso.

Por qué y cuándo se efectúa 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 12. Métodos API para plantillas de tareas.

Método	Descripción
getTaskTemplate	Recupera 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.

Tabla 12. Métodos API para plantillas de tareas. (continuación)

Método	Descripción
createAndStartTask	Crea e inicia una instancia de tarea a partir de la plantilla de tarea especificada.
createTask	Crea una instancia de tarea a partir de la plantilla de 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 13. 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.
suspender	Suspende la tarea a realizar o de colaboración.
resume	Reanuda la tarea a realizar o de colaboración.
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.
cancelClaim	Libera una instancia de tarea reclamada de manera que pueda trabajar con ella otro propietario poetencial.
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 14. Métodos API para trabajar con escaladas

Método	Descripción
getEscalation	Recupera la instancia de escalada especificada.

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 15. 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.
getCustomPropertyNames	Recupera los nombres de las propiedades personalizadas de la instancia de tarea.
setCustomProperty	Almacena valores específicos personalizados para la instancia de tarea especificada.

Acciones permitidas para tareas:

Las acciones que pueden realizarse en una tarea dependen de si se trata de una tarea a realizar, una tarea de colaboración, una tarea de invocación o una tarea administrativa.

No puede utilizar todas las acciones proporcionadas por la interfaz HumanTaskManager para todas las clases de tareas. La tabla siguiente muestra las acciones que pueden llevarse a cabo en cada clase de tarea.

	Clase de tarea			
Acción	Tarea a realizar	Tarea de colaboración	Tarea de invocación	Tarea administrativa
callTask			X	
cancelClaim	X	X[1]		
claim	X	X[1]		
complete	X	X[1]		X
completeWithFollowOnTask[4]	X	X[1]		
completeWithFollowOnTask[5]		X[3]	X[3]	
createFaultMessage	X	X	X	X

Acción	Clase de tarea			
	Tarea a realizar	**Tarea de colaboración**	**Tarea de invocación**	**Tarea administrativa**
createInputMessage	X	X	X	X
createOutputMessage	X	X	X	X
createWorkItem	X	X[1]	X	X
delete	X[1]	X[1]	X	X[1]
deleteWorkItem	X	X[1]	X	X
getCustomProperty	X	X[1]	X	X
getDocumentation	X	X[1]	X	X
getFaultNames	X	X[1]		
getFaultMessage	X	X[1]	X	
getInputMessage	X	X[1]	X	
getOutputMessage	X	X[1]	X	
getUsersInRole	X	X[1]	X	X
getTask	X	X[1]	X	X
getUISettings	X	X[1]	X	X
resume	X	X[1]		
setCustomProperty	X	X[1]	X	X
setFaultMessage	X	X[1]		
setOutputMessage	X	X[1]		
startTask	X[1]	X[1]	X	X
startTaskAsSubtask[6]	X	X[1]		
startTaskAsSubtask[7]		X[3]	X[3]	
suspend	X	X[1]		
suspendWithCancelClaim	X	X[1]		
terminate	X[1]	X[1]	X[1]	
transferWorkItem	X	X[1]	X	X
update	X	X[1]	X	X

Notas:

1. Para tareas autónomas, tareas ad-hoc y plantillas de tareas únicamente.
2. Para tareas autónomas, tareas en línea de procesos empresarialesy tareas de tiempo de ejecución únicamente
3. Sólo para tareas autónomas y tareas ad-hoc
4. Los tipos de tareas que pueden tener tareas de continuación
5. Los tipos de tareas que se pueden utilizar como tareas de continuación
6. Los tipos de tareas que pueden tener subtareas
7. Los tipos de tareas que se pueden utilizar como subtareas

Desarrollo de aplicaciones para procesos empresarialesy tareas de usuario

Las personas implicadas en la mayoría de los escenarios de procesos de empresa. Por ejemplo, un proceso empresarialrequiere 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.

Por qué y cuándo se efectúa esta tarea

Para implicar personas en escenarios de procesos de empresa, puede incluir los siguientes tipos de tareas en el proceso de empresa:

- 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 participativa* 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 empresarialrepresentan las tareas a realizar que la gente lleva a cabo en el escenario de proceso de empresa. Puede utilizar la API de Business Flow Manager y la API del Gestor de tareas de usuario para realizar estos escenarios.

- El proceso empresariales 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. En otras palabras, 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 de proceso o actividades que pueden iniciarse

Un proceso empresarial puede iniciarse invocando los métodos call, initiate, o sendMessage de la API de Business Flow Manager. Si el proceso solo tiene una

actividad de inicio, puede utilizar la signatura de método que necesita un nombre de plantilla de proceso como parámetro. Si el proceso tiene más de una actividad de inicio, debe identificarla explícitamente.

Por qué y cuándo se efectúa esta tarea

Cuando se modela un proceso de empresa, el modelador puede decidir que solo un subconjunto de usuarios pueda crear una instancia de proceso a partir de la plantilla de proceso. Esto se lleva a cabo asociando una tarea de invocación en línea a una actividad de inicio del proceso y especificando restricciones de autorización en esa tarea. Sólo las personas que sean administradores o iniciadores potenciales de la tarea pueden crear una instancia de la tarea y, por consiguiente, una instancia de la plantilla de proceso.

Si una tarea de invocación en línea no está asociada con la actividad de inicio o si no se especifican las restricciones de autorización para la tarea, todos podrán crear una instancia de proceso mediante 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 con 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 de iniciado.

   Consejo: El método queryProcessTemplates sólo excluye las plantillas de proceso que forman parte de aplicaciones que todavía no se han iniciado. Por tanto, si utiliza este método sin filtrar los resultados, el método devuelve todas las versiones de las plantillas de proceso independientemente del estado en que se encuentren.

   ```
   // indicación de fecha y hora actuales en formato UTC, convertido
   // en 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 lista de actividades de inicio para las que el usuario está autorizado.

   La lista de plantillas de proceso contiene las que tienen una sola actividad de inicio. Estas actividades no están protegidas o el usuario que ha iniciado la sesión puede iniciarlas. También es posible reunir las plantillas de proceso que se puedan iniciar por parte de al menos una de las actividades de inicio.

   Consejo: Un administrador de proceso también puede iniciar una instancia de proceso. Para obtener una lista completa de plantillas, también debe leer la

plantilla de tarea de administración que esté asociada con la plantilla de proceso y 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 una plantilla de tarea de invocación no existe, la plantilla de proceso no está protegida por esta actividad de inicio.

      En este caso, todas las personas 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 del Gestor de tareas de usuario para comprobar la autorización para el usuario que ha iniciado la sesión.

      En el ejemplo, el usuario que ha iniciado la sesión es Torres. El usuario que ha iniciado la sesión debe ser un iniciador potencial de la tarea de invocación o un administrador.

```
if ( tktid != null )
    {
      isAuthorized =
        task.isUserInRole
            (tkid, "Torres", WorkItem.REASON_POTENTIAL_STARTER) ||
        task.isUserInRole(tktid, "Torres", WorkItem.REASON_ADMINISTRATOR);

      if ( isAuthorized )
      {
        authorizedActivityServiceTemplates.add(activity);
      }
    }
```

      Si el usuario tiene el rol especificado o si los criterios de asignación de personas para el rol no se han especificado, el método isUserInRole devuelve el valor `true`.

5. Compruebe si el proceso puede iniciarse utilizando solo el nombre de plantilla de proceso.

```
if ( isAuthorized && startActivities.length == 1 )
    {
      authorizedProcessTemplates.add(template);
    }
```

6. Termine los bucles.

```
  } // fin del bucle para cada plantilla de servicio de actividad
}   // fin del bucle para cada plantilla 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. En este ejemplo se muestra cómo implementar la secuencia de acciones para pedir el libro como una serie de actividades de tareas de usuario (tareas a realizar). Se utiliza Business Flow Manager y las API de Human Task Manager para procesar el flujo de trabajo.

Por qué y cuándo se efectúa esta tarea

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 tarea de usuario 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 241
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. La API completeAndClaimSuccessor admite el proceso de este tipo de flujo de trabajo.

Manejo de excepciones y errores

Es posible que un proceso BPEL encuentre un error en puntos diferentes del proceso.

Por qué y cuándo se efectúa 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

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 llamante

Por qué y cuándo se efectúa 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. El *método recomendado* es capturar las excepciones genéricas que finalizan 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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 la API de servicio Web

Puede desarrollar aplicaciones cliente que acceden a las aplicaciones de proceso empresarialy de tareas de usuario mediante las API de servicios Web.

Por qué y cuándo se efectúa esta tarea

Las aplicaciones cliente se pueden desarrollar en cualquier entorno de cliente de servicio Web, incluidos los servicios Web Java y Microsoft .NET.

Conceptos relacionados

"Comparación de las interfaces de programación para interactuar con procesos empresarialesy tareas de usuario" en la página 187
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 empresarialesy tareas de usuario. Cada una de estas interfaces tiene características diferentes.

Componentes de servicio Web y secuencia de control

Varios componentes en el cliente y en el servidor participan en la secuencia de control que representa una petición y respuesta de un servicio Web.

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 cliente proxy (que también proporciona el usuario pero que se puede generar automáticamente utilizando programas de utilidad en el cliente) incluye la petición de servicio en un sobre de petición SOAP.
   c. La infraestructura de desarrollo en el cliente reenvía la petición a un URL definido como el 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 descodifica la petición.
   b. La petición se maneja directamente mediante el componente Business Flow Manager o Human Task Manager o se dirige al proceso empresarialo 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 cliente proxy extrae los datos de la respuesta SOAP y los transfiere a la aplicación cliente.
   c. La aplicación cliente procesa los datos devueltos como corresponda.

Visión general de las API de servicios Web

Las API de los servicios Web le permiten desarrollar aplicaciones de cliente que utilizan servicios Web para acceder a los procesos empresarialesy a las tareas de usuario en el entorno de Business Process Choreographer.

La API de servicios Web de Business Process Choreographer proporcionan dos interfaces de servicios Web diferentes (tipos de puerto WSDL):

- La API de Business Flow Manager. Permite que las aplicaciones cliente interactúen con los microflujos y procesos de larga ejecución, por ejemplo:
  - Crear plantillas de proceso e instancias de proceso
  - Reclamar procesos existentes

– Consultar procesos por su ID

Consulte “Desarrollo de aplicaciones para procesos de empresa” en la página 230 para obtener una lista completa de las acciones posibles.

- La API de Human Task Manager. Permite a las aplicaciones cliente:
  – Crear e iniciar tareas
  – Reclamar tareas existentes
  – Completar tareas
  – Consultar tareas por su ID
  – Consultar una colección de tareas.

  Consulte “Desarrollo de aplicaciones para tareas de usuario” en la página 251 para obtener una lista completa de las acciones posibles.

Las aplicaciones de cliente pueden utilizar cualquiera o las dos interfaces de servicios Web.

Ejemplo

La siguiente es una posible descripción de una aplicación de cliente que accede a la API de servicios Web de Human Task Manager para procesar una tarea de usuario participante:

1. La aplicación cliente emite una llamada de servicio Web query a WebSphere Process Server que solicita una lista de tareas participantes en las que trabajará un usuario.
2. La lista de tareas de participación se devuelve en un sobre de respuesta SOAP/HTTP.
3. La aplicación cliente que emite una llamada de servicio Web a claim para reclamar una de las tareas de participación.
4. WebSphere Process Server devuelve el mensaje de entrada de 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 para los procesos empresarialesy las tareas de usuario

Los procesos empresarialesy 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:

1. Las interfaces de los procesos empresarialesy las tareas de usuario 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 empresarialesy tareas de usuario desarrollados con WID.
2. Los mensajes de error que exponen los procesos empresarialesy 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?

Desarrollo de aplicaciones cliente

El proceso de desarrollo de aplicaciones cliente consta de varios pasos.

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. Alternativamente, puede copiar los archivos desde el CD del cliente de WebSphere Process Server.
3. En el entorno de desarrollo de aplicaciones cliente seleccionado, genere un *cliente proxy* utilizando los artefactos exportados.
4. Opcional: Generación de las clases *helper*. Las clases helper son necesarias si la aplicación de cliente interactúa directamente con procesos o tareas concretos en el servidor WebSphere. No obstante, no son necesarios si la aplicación cliente sólo va a realizar tareas genéricas como la emisión de consultas.
5. Desarrolle el código de la aplicación cliente.
6. Añada los mecanismos de seguridad necesarios para la aplicación cliente.

Copia de artefactos

Se deben copiar varios artefactos desde el entorno WebSphere para ayudar a crear las aplicaciones de cliente.

Hay dos modos de obtener estos artefactos:

- Publicar y exportarlas desde el entorno de WebSphere Process Server.
- Copiar los archivos desde el CD del cliente de WebSphere Process Server.

Publicación y exportación de artefactos desde el entorno de servidor

Para poder desarrollar aplicaciones de cliente para acceder a las API de servicios Web, debe publicar y exportar varios artefactos desde el entorno de servidor de WebSphere.

Por qué y cuándo se efectúa esta tarea

Los artefactos que se han de exportar son:

- Los archivos WSDL (Web Service Definition Language) que describen los tipos de puerto y las operaciones que componen las API de servicios Web.
- Los archivos XSD (XML Schema Definition) que contienen definiciones de los tipos de datos a los que hacen referencias los servicios y métodos de los archivos WSDL.
- Archivos WSDL y XSD que describen objetos empresariales. Objetos empresariales que describen procesos empresarialesconcretos o tareas de usuario que se ejecutan en el servidor de WebSphere. Estos archivos adicionales sólo son necesarios si la aplicación de cliente tiene que interactuar directamente con procesos empresarialeso tareas de usuario concretas mediante las API de

servicios Web. No son necesarios si la aplicación de cliente sólo va a realizar tareas genéricas como, por ejemplo, emitir consultas.

Una vez publicados estos artefactos, tiene que copiarlos en el entorno de programación de cliente, en el que se utilizan para generar un cliente proxy y clases helper.

Especificación de la dirección de punto final de servicio Web:

La dirección de punto final de servicio Web es el URL que la aplicación cliente debe especificar para acceder a las API de servicios Web. La dirección de punto final se graba al archivo WSDL que exporta para generar un cliente proxy para la aplicación cliente.

Por qué y cuándo se efectúa esta tarea

La dirección de punto final de servicio Web que se va a utilizar depende de la configuración del servidor WebSphere:

- Escenario 1. Un solo servidor WebSphere. La dirección de punto final de WebSphere que se va a especificar es el nombre de sistema principal y el número de puerto del servidor, por ejemplo, **host1:9080**.
- Escenario 2. Un clúster de WebSphere formado por varios servidores. La dirección de punto final de WebSphere que se va a especificar es el nombre de sistema principal y puerto del servidor que aloja las API de servicios Web, por ejemplo, **host2:9081**.
- Escenario 3. Se utiliza un servidor Web de programa frontal. La dirección de punto final de WebSphere que se va a especificar es el nombre de sistema principal y puerto del servidor Web, por ejemplo, **host:80**.

Por omisión, la dirección de punto final de servicio Web adopta la forma de *protocolo*`://`*sistema_ppal*`:`*puerto*`/`*raíz_contexto*`/`*vía_acceso_fija*. Donde:

- *protocolo*. El protocolo de comunicaciones que se va a utilizar entre la aplicación cliente y el servidor WebSphere. El protocolo por omisión es HTTP. Puede optar por utilizar en su lugar un protocolo HTTPS (HTTP en SSL) más seguro. Se recomienda utilizar HTTPS.
- *sistema_ppal*`:`*puerto*. El nombre de sistema principal y número de puerto utilizados para acceder a la máquina que aloja las API de servicios Web. Estos valores varían en función de la configuración del servidor WebSphere, por ejemplo, si la aplicación cliente va a acceder directamente a la aplicación o mediante un programa frontal de servidor Web.
- *raíz_contexto*. Puede seleccionar el valor de raíz de contexto que desee. El valor elegido debe ser, no obstante, único dentro de cada célula de WebSphere. El valor por omisión utiliza un sufijo "sufijo_nodo/cluster" que elimina el riesgo de conflictos de denominación.
- La *vía_acceso_fija* es `/sca/com/ibm/bpe/api/BFMWS` (para la API de Business Flow Manager) o `/sca/com/ibm/task/api/HTMWS` (para la API de Human Task Manager) y no se puede modificar.

La dirección de punto final de servicio Web se especifica inicialmente al configurar el contenedor de procesos empresarialeso el contenedor de tareas de usuario:

Procedimiento

1. Inicie la sesión en la consola administrativa con el ID de usuario con derechos de administrador.

2. Seleccione **Aplicaciones → Módulos SCA**.

   Nota: También puede seleccionar **Aplicaciones → Aplicaciones de empresa** para mostrar una lista de todas las aplicaciones de empresa disponibles.
3. Seleccione **BPEContainer** (para el contenedor de procesos de empresa) o **TaskContainer** (para el contenedor de tareas de usuario) de la lista de módulos SCA o aplicaciones.
4. Seleccione **Proporcionar información de URL de punto final HTTP** de la lista de **Propiedades adicionales**.
5. Seleccione uno de los prefijos por omisión de la lista o introduzca un prefijo personalizado. Utilice un prefijo de la lista de prefijos por omisión si las aplicaciones cliente se van a conectar directamente con el servidor de aplicaciones que aloja la API de servicios Web. De lo contrario, especifique un prefijo personalizado.
6. Pulse **Aplicar** para copiar el prefijo seleccionado al módulo SCA.
7. Pulse **Aceptar**. Se guardará la información de URL en el espacio de trabajo.

Resultados

Puede consultar el valor actual en la consola administrativa (por ejemplo, para el contenedor de procesos de empresa: **Aplicaciones de empresa → BPEContainer → Ver descriptor de despliegue**).

En el archivo WSDL exportado, el atributo `location` del elemento `soap:address` contiene la dirección de punto final de servicios Web especificada. Por ejemplo:

```
<wsdl:service name="BFMWSService">
  <wsdl:port name="BFMWSPort" binding="this:BFMWSBinding">
   <soap:address location=
         "https://myserver:9080/WebServicesAPIs/sca/com/ibm/bpe/api/BFMWS"/>
```

Publicación de archivos WSDL:

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. Primero debe publicar estos archivos WSDL, luego copiarlos del entorno WebSphere al entorno de desarrollo, donde se utilizan para generar clientes proxy.

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.

Por qué y cuándo se efectúa esta tarea

Sólo tiene que publicar una vez los archivos WSDL.

Nota: Si tiene el CD del cliente de WebSphere Process Server, puede copiar directamente los archivos al entorno de programación de cliente desde el mismo.

Publicación del WSDL de proceso empresarial:

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. Seleccione **Aplicaciones** → **Módulos SCA**

   Nota: También puede seleccionar **Aplicaciones** → **Aplicaciones de empresa** para mostrar una lista de todas las aplicaciones de empresa 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 de 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 tiene el nombre BPEContainer_WSDLFiles.zip. El archivo zip contiene un archivo WSDL que describe los servicios Web y cualquier referencia de archivos XSD contenidas en el archivo WSDL.

Publicación del WSDL de tareas de usuario:

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. Seleccione **Aplicaciones** → **Módulos SCA**

   Nota: También puede seleccionar **Aplicaciones** → **Aplicaciones de empresa** para mostrar una lista de todas las aplicaciones de empresa 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 de 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 tiene el nombre TaskContainer_WSDLFiles.zip. El archivo zip contiene un archivo WSDL que describe los servicios Web y cualquier referencia de archivos XSD contenidas en el archivo WSDL.

Exportación de objetos empresariales:

Los procesos empresarialesy las tareas de usuario tienen interfaces bien definidas que les permiten acceder externamente como servicios Web. Si estas interfaces hacen referencia a los objetos empresariales, tendrá que exportar las definiciones de interfaz y los objetos empresariales al entorno de programación cliente.

Por qué y cuándo se efectúa esta tarea

Este procedimiento debe repetirse para cada objeto empresarial con el que la aplicación cliente tenga que interactuar.

En WebSphere Process Server, los objetos empresariales definen el formato de los mensajes de petición, respuesta y error que interactúan con los procesos empresarialeso las tareas de usuario. Estos mensajes también pueden contener definiciones de tipos de datos complejos.

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 empresarial. Todas las operaciones de la interfaz de servicio Web se crean como una operación "document/literal con envoltorio". Los parámetros de entrada y salida de estas operaciones se encapsulan en documentos de envoltorio. Otros objetos empresariales definen los formatos de respuesta y error correspondientes.

Para crear e iniciar el proceso empresarialo 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.

Para conseguir esto, se exportan los objetos empresariales del entorno WebSphere como archivos WSDL (Web Service Definition Language) y XSD (XML Schema Definition), se importan las definiciones de tipo de datos al entorno de programación cliente, luego se convierten a clases de ayuda para que las utilice la aplicación 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 empresariales que se van a exportar. Un módulo de biblioteca es un archivo comprimido que contiene los objetos empresariales necesarios.
3. Exporte el módulo de biblioteca.
4. Copie los archivos exportados al entorno de desarrollo de aplicaciones cliente.

Ejemplo

Supongamos que un proceso empresarialexpone 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` deben pasarse a las API de servicios Web y recibirse utilizando los parámetros <xsd:any> en todas las operaciones *genéricas* (call, sendMessage, etc.) realizadas por la aplicación cliente. Estos elementos definidos por el cliente se crean, serializan y deserializan en la aplicación cliente con las clases de ayuda que se generan utilizando los archivos XSD exportados.

Utilización de archivos en el CD del cliente

Como alternativa a la exportación de artefactos desde el entorno de servidor WebSphere, puede copiar los archivos necesarios para generar una aplicación de cliente del CD del cliente de WebSphere Process Server.

En este caso, debe modificar manualmente la dirección de punto final de servicios Web por omisión de la API de Business Flow Manager o de la API de Human Task Manager.

Si la aplicación de cliente es para acceder a las dos API, debe editar la dirección del punto final por omisión para las dos API.

Copia de los archivos desde el CD de cliente:

Los archivos necesarios para acceder a las API de servicios Web están disponibles en el CD del cliente de WebSphere Process Server.

Procedimiento

1. Acceda al CD del cliente y vaya al directorio ProcessChoreographer\client.
2. Copie los archivos necesarios en el entorno de desarrollo de aplicaciones de cliente.

   Para la API de Business Flow Manager, copie:

   BFMWS.wsdl
   Describe los servicios Web disponibles en la API de servicios Web de Business Flow Manager. Este archivo contiene la dirección de punto final.

   BFMIF.wsdl
   Describe los parámetros y el tipo de datos de cada servicio Web en la API de servicios Web de Business Flow Manager.

   BFMIF.xsd
   Describe los tipos de datos utilizados en la API de servicios Web de Business Flow Manager.

BPCGEN.xsd
Contiene los tipos de datos que son comunes entre las API de servicio Web de Business Flow Manager y Human Task Manager.

Para la API de Human Task Manager, copie:

HTMWS.wsdl
Describe los servicios Web disponibles en la API de servicios Web de Human Task Manager. Este archivo contiene la dirección de punto final.

HTMIF.wsdl
Describe los parámetros y el tipo de datos de cada servicio Web de la API de servicios Web de Human Task Manager.

HTMIF.xsd
Describe los tipos de datos utilizados en la API de servicios Web de Human Task Manager.

BPCGEN.xsd
Contiene los tipos de datos que son comunes entre las API de servicio Web de Business Flow Manager y Human Task Manager.

Nota: El archivo BPCGen.xsd es común a las dos API.

Qué hacer a continuación

Después de copiar los archivos, debe modificar manualmente la dirección de punto final de la API de servicios Web en los archivos BFMWS.wsdl o HTMWS.wsdl por la dirección en la que WebSphere Application Server aloja las API de servicios Web.

Cambio manual de la dirección de punto final de servicio Web:

Si copia los archivos del CD de cliente, debe cambiar la dirección de punto final de servicio Web especificada en los archivos WSDL por la del servidor que aloja las API de servicios Web.

Por qué y cuándo se efectúa esta tarea

Puede utilizar la consola administrativa para establecer la dirección de punto final de servicio Web antes de exportar los archivos WSDL. No obstante, si copia los archivos WSDL del CD de cliente de WebSphere, debe modificar manualmente la dirección de punto final de servicio Web por omisión.

La dirección de punto final de servicio Web que se va a utilizar depende de la configuración del servidor WebSphere:

- Escenario 1. Hay un solo servidor WebSphere. La dirección de punto final de WebSphere que se va a especificar es el nombre de sistema principal y el número de puerto del servidor, por ejemplo, **host1:9080**.
- Escenario 2. Un clúster de WebSphere formado por varios servidores. La dirección de punto final de WebSphere que se va a especificar es el nombre de sistema principal y puerto del servidor que aloja las API de servicios Web, por ejemplo, **host2:9081**.
- Escenario 3. Se utiliza un servidor Web de programa frontal. La dirección de punto final de WebSphere que se va a especificar es el nombre de sistema principal y puerto del servidor Web, por ejemplo, **host:80**.

Modificación del punto final de la API de Business Flow Manager:

Si copia los archivos de la API de Business Flow Manager del CD del cliente de WebSphere Process Server, debe editar manualmente la dirección de punto final por omisión.

Procedimiento

1. Vaya al directorio que contiene los archivos que ha copiado del CD del cliente.
2. Abra el archivo BFMWS.wsdl en un editor de texto o en un editor XML.
3. Localice el elemento `soap:address` (situado junto a la parte inferior del archivo).
4. Modifique el valor del atributo `location` con el URL HTTP del servidor en el que se ejecuta la API de servicio Web. Para hacerlo:
   a. Opcionalmente, sustituya `http` por `https` de modo que utilice el protocolo HTTPS más seguro.
   b. Sustituya *localhost* por el nombre de sistema principal o dirección IP de la dirección de punto final de servidor de la API de servicios Web.
   c. Sustituya *9080* por el número de puerto del servidor de aplicaciones.
   d. Sustituya *BPEContainer_N1_server1* por la raíz de contexto de la aplicación que ejecuta la API de servicios Web. La raíz de contexto por omisión consta de:
      - *BPEContainer*. El nombre de la aplicación.
      - *N1*. El nombre de nodo.
      - *server1*. El nombre de servidor.
   e. No modifique la parte fija del URL (`/sca/com/ibm/bpe/api/BFMWS`) .

   Por ejemplo, si la aplicación se ejecuta en el servidor **s1.n1.ibm.com** y el servidor acepta peticiones SOAP/HTTP en el puerto **9080**, modifique el elemento `soap:address` del modo siguiente:

```
<soap:address location="http://si.n1.ibm.com:9080/
                  BPEContainer_N1_server1/sca/com/ibm/bpe/api/BFMWS"/>
```

Modificación del punto final de la API de Human Task Manager:

Si copia los archivos de la API de Human Task Manager del CD del cliente de WebSphere Process Server, debe editar manualmente la dirección de punto final por omisión.

Procedimiento

1. Vaya al directorio que contiene los archivos que ha copiado del CD del cliente.
2. Abra el archivo HTMWS.wsdl en un editor de texto o en un editor XML.
3. Localice el elemento `soap:address` (situado junto a la parte inferior del archivo).
4. Modifique el valor del atributo `location` por la dirección de punto final correcta. Para hacerlo:
   a. Opcionalmente, sustituya `http` por `https` de modo que utilice el protocolo HTTPS más seguro.
   b. Sustituya *localhost* por el nombre de sistema principal o dirección IP de la dirección de punto final de servidor de la API de servicios Web.
   c. Sustituya *9080* por el número de puerto del servidor de aplicaciones.
   d. Sustituya *HTMContainer_N1_server1* por la raíz de contexto de la aplicación que ejecuta la API de servicios Web. La raíz de contexto por omisión consta de:
      - *HTMContainer*. El nombre de la aplicación.
      - *N1*. El nombre de nodo.

- *server1*. El nombre de servidor.

e. No modifique la parte fija del URL (`/sca/com/ibm/task/api/HTMWS`) .

Por ejemplo, si la aplicación se ejecuta en el servidor **s1.n1.ibm.com** y el servidor acepta peticiones SOAP/HTTPS en el puerto **9081**, modifique el elemento `soap:address` del modo siguiente:

```
<soap:address location="https://si.n1.ibm.com:9081/
                  HTMContainer_N1_server1/sca/com/ibm/task/api/HTMWS"/>
```

Desarrollo de aplicaciones cliente en el entorno de servicios Web Java

Puede utilizar cualquier entorno de desarrollo Java compatible con los servicios Web Java para desarrollar aplicaciones cliente para las API de servicios Web.

Generación de un cliente de proxy (Servicios Web Java)

Las aplicaciones cliente del servicio Web Java utilizan un *cliente proxy* para interactuar con las API de servicios Web.

Por qué y cuándo se efectúa esta tarea

Un cliente proxy para los servicios Web Java contiene un número de clases de beans Java que las aplicaciones de cliente invocan para ejecutar las peticiones de servicios Web. El cliente proxy maneja el ensamblaje de los parámetros de servicio en mensajes SOAP, envía los mensajes SOAP al servicio Web a través de HTTP, recibe las respuestas del servicio Web y pasa los datos devueltos a la aplicación de cliente.

Por lo tanto, básicamente un cliente proxy permite que una aplicación invoque un servicio Web como si fuera una función local.

Nota: Sólo tiene que generar una vez el cliente proxy. Todas las aplicaciones de cliente que acceden a la misma API de servicios Web pueden utilizar entonces el mismo cliente proxy.

En el entorno de servicios IBM Web, hay dos modos de generar un cliente proxy:

- Utilizando entornos de desarrollo integrado de Rational Application Developer o WebSphere Integration Developer.
- Utilización de la herramienta de línea de mandatos WSDL2Java.

Otros entornos de desarrollo de servicios Web Java generalmente incluyen la herramienta WSDL2Java o los recursos de generación de aplicaciones de cliente.

Utilización de Rational Application Developer para generar un cliente proxy:

El entorno de desarrollo integrado de Rational Application Developer le permite generar un cliente proxy para la aplicación de cliente.

Antes de empezar

Antes de generar un cliente proxy, debe haber exportado anteriormente los archivos WSDL que describen las interfaces de servicios Web de procesos empresarialeso de tareas de usuarios desde el entorno de WebSphere (o desde el CD de cliente de WebSphere Process Server) y copiarlos en el entorno de programación del cliente.

Procedimiento

1. Añada el archivo WSDL adecuado al proyecto:
   - Para procesos de empresa:
     a. Descomprima el archivo exportado BPEContainer_*nombre-nodo_nombre-servidor*_WSDLFiles.zip en un directorio temporal.
     b. Importe el subdirectorio META-INF desde el directorio descomprimido BPEContainer_*nombre-nodo_nombre-servidor*.ear/b.jar.
   - Para tareas de usuario:
     a. Descomprima el archivo exportado TaskContainer_*nombre-nodo_nombre-servidor*_WSDLFiles.zip en un directorio temporal.
     b. Importe el subdirectorio META-INF desde el directorio descomprimido TaskContainer_*nombre-nodo_nombre-servidor*.ear/h.jar.

   Se creará una nueva estructura de directorio wsdl y subdirectorio en el proyecto.
2. Modifique las propiedades del asistente de servicio Web:
   a. En Rational Application Developer, seleccione **Preferencias** → **Servicios Web** → **Generación de código** → **Unidad ejecutable IBM WebSphere**
   b. Seleccione la opción **Generar Java a partir de WSDL mediante el estilo sin acomodación**.

   Nota: Si no puede seleccionar la opción **Servicios Web** en el menú **Preferencias**, primero debe habilitar las posibilidades requeridas del modo siguiente: **Ventana** → **Preferencias** → **Entorno de trabajo** → **Posibilidades**. Pulse **Desarrollador de servicios Web** y **Aceptar**. A continuación, vuelva a abrir la ventana Preferencias y cambie la opción **Generación de código**.
3. Seleccione el archivo BFMWS.WSDL o HTMWS.WSDL ubicados en el directorio wsdl de reciente creación.
4. Pulse el botón derecho del ratón y seleccione **Servicios Web** → **Generar cliente**.

   Antes de seguir con los pasos restantes, compruebe que el servidor se haya iniciado.
5. En la ventana Servicios Web, pulse **Siguiente** para aceptar los valores por omisión.
6. En la ventana Selección de servicios Web, pulse **Siguiente** para aceptar los valores por omisión.
7. En la ventana Configuración del entorno de cliente:
   a. Pulse **Editar** y cambie la opción Tiempo de ejecución de servicio Web en IBM WebSphere
   b. Cambie la opción Versión de J2EE en 1.4.
   c. Pulse **Aceptar**.
   d. Pulse **Siguiente**.
8. Este paso sólo es necesario si debe generar un cliente de servicios Web que incluya las API de servicios Web de procesos empresarialesy de tareas de usuario, ya que hay métodos duplicados en ambos archivos WSDL.
   a. En la ventana Proxy de servicio Web, seleccione Definir correlación personalizada para espacio de nombres con paquete y, a continuación, pulse **Aceptar**.
   b. En la ventana de correlación de cliente de servicio Web de espacio de nombres con paquete, añada los siguientes espacios de nombres y paquete:

Para BFMWS.wsdl:

Espacio de nombres	Paquete
http://www.ibm.com/xmlns/prod/websphere/business-process/types/6.0	com.ibm.sca.bpe
http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0	com.ibm.sca.bpe
http://www.ibm.com/xmlns/prod/websphere/business-process/services/6.0/Binding	com.ibm.sca.bpe
http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0	com.ibm.sca.bpe

For HTMWS.wsdl:

Espacio de nombres	Paquete
http://www.ibm.com/xmlns/prod/websphere/human-task/types/6.0	com.ibm.sca.task
http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0	com.ibm.sca.task
http://www.ibm.com/xmlns/prod/websphere/human-task/services/6.0/Binding	com.ibm.sca.task
http://www.ibm.com/xmlns/prod/websphere/bpc-common/types/6.0	com.ibm.sca.task

Si se le solicita que confirme la sobrescritura, pulse **Sí a todo**.

9. Pulse **Finalizar**.

Resultados

Se generará y añadirá un cliente proxy, formado por un número de clases Java, proxy, locator y helper al proyecto. También se actualizará el descriptor de despliegue.

Utilización de WSDL2Java para generar clientes proxy:

WSDL2Java es una herramienta de línea de mandatos que genera clientes proxy. Los clientes proxy facilitan la programación de aplicaciones cliente.

Antes de empezar

Antes de generar un cliente proxy, debe haber exportado anteriormente los archivos WSDL que describen las API de servicios Web de procesos empresarialeso de tareas de usuarios desde el entorno de WebSphere (o el CD del cliente de WebSphere Process Server) y los ha copiado en el entorno de programación.

Por qué y cuándo se efectúa esta tarea

Procedimiento

1. Utilice la herramienta WSDL2Java para generar un cliente proxy: Escriba:
   wsdl2java *opciones vía_acceso_archivo_WSDL*
   Donde:
   - *opciones* incluyen:

-noWrappedOperations (-w)
Inhabilita la detección de operaciones envueltas. Se generan beans Java de mensajes de petición y respuesta.

Nota: Este no es el valor por omisión.

-role (-r)
Especifique el valor **client** para generar los archivos y archivos de enlace del desarrollo en el cliente.

-container (-c)
El contenedor en el cliente que se va a utilizar. Los argumentos válidos incluyen:

client Un contenedor de cliente

ejb Contenedor de EJB (Enterprise JavaBeans).

none Ningún contenedor

web Un contenedor Web

-output (-o)
La carpeta en la que se van a almacenar los archivos generados.

Para obtener una lista completa de parámetros WSDL2Java, utilice el conmutador de línea de mandatos **-help** o la ayuda en línea de la herramienta WSDL2Java de WID/RAD.

- *vía_acceso_archivo_WSDL* es la vía de acceso y nombre de archivo del archivo WSDL que ha exportado desde el entorno de WebSphere o que ha copiado del CD de cliente.

En el ejemplo siguiente se genera un cliente proxy para la API de servicios Web de actividades de tareas de usuario:

```
call wsdl2java.bat -r client -c client -noWrappedOperations
                   -output c:\ws\proxyClient  c:\ws\bin\HTMWS.wsdl
```

2. Incluya los archivos de clase generados en el proyecto.

Creación de clases de ayuda para procesos BPEL (servicios Web Java)

Los objetos empresariales a los que se hace referencia en peticiones de API en concreto (por ejemplo, sendMessage o call) requieren que las aplicaciones cliente utilicen elementos de estilo ″document/literal con envoltorio″. Las aplicaciones cliente requieren las clases de ayuda para que les ayuden a generar los elementos de envoltorio necesarios.

Antes de empezar

Para crear clases helper, debe haber exportado el archivo WSDL de la API de servicios Web del entorno WebSphere Process Server.

Por qué y cuándo se efectúa esta tarea

Las operaciones call() y sendMessage() de las API de servicios Web permiten la interacción con los procesos BPEL en WebSphere Process Server. El mensaje de entrada de la operación call() espera a que se proporcione la envoltura document/literal del mensaje de entrada del proceso.

Hay varias técnicas posibles para generar clases helper para un proceso BPEL o tarea de usuario, incluidos:

1. Utilice el objeto SoapElement.

   En el entorno Rational Application Developer disponible en WebSphere Integration Developer, el motor de servicio Web da soporte a JAX-RPC 1.1. En JAX-RPC 1.1, el objeto SoapElement amplía un elemento DOM (Document Object Model), por lo tanto, se puede utilizar la API DOM para crear, leer, cargar y guardar mensajes SOAP.

   Por ejemplo, presuponga que el archivo WSDL contiene el siguiente mensaje de entrada para un proceso de flujo de trabajo o tarea de usuario:

   ```
   <xsd:element name="operation1">
           <xsd:complexType>
             <xsd:sequence>
               <xsd:element name="input1" nillable="true" type="xsd:string"/>
             </xsd:sequence>
           </xsd:complexType>
         </xsd:element>
   ```

   El archivo WSDL se crea al desarrollar un módulo de proceso o de tarea de usuario.

   Para crear el mensaje SOAP correspondiente en la aplicación de cliente utilizando la API DOM:

   ```
   SOAPFactory soapfactoryinstance = SOAPFactory.newInstance();
   SOAPElement soapmessage = soapfactoryinstance.createElement
                             ("operation1", namespaceprefix, interfaceURI);
   SOAPElement inputelement = soapfactoryinstance.createElement("input1");
   inputelement.addTextNode( message value);
   soapmessage.addChildElement(outputelement);
   ```

   El ejemplo siguiente muestra cómo se crean los parámetros de entrada para la operación sendMessage en la aplicación de cliente:

   ```
   SendMessage inWsend = new SendMessage();
   inWsend.setProcessTemplateName(processtemplatename);
   inWsend.setPortType(porttype);
   inWsend.setOperation(operationname);
   inWsend.set_any(soapmessage);
   ```

2. Utilice la característica de enlace de datos personalizado de WebSphere.

   Esta técnica se describe en los siguientes artículos de developerWorks:

   - Cómo se selecciona una tecnología de correlación personalizada para servicios Web
   - Desarrollo de servicios Web con los SDO de EMF para un esquema XML complejo

 Interoperatividad con patrones y estrategias para los servicios Web basados en documentos

 Soporte de servicios Web para esquema o WSDL que contienen tipos de esquemas XML JAX-RPC 1.0/1.1

Creación de aplicaciones cliente (servicios Web Java)

Las aplicaciones cliente envían las peticiones a las API de servicios Web y reciben las respuestas. Utilizando un cliente proxy para gestionar las comunicaciones y las clases de ayuda para dar formato a los 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 cliente proxy y las clases de ayuda necesarias.

Por qué y cuándo se efectúa esta tarea

Puede desarrollar aplicaciones cliente utilizando una herramienta de desarrollo compatible con los servicios Web, por ejemplo, RAD (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 cliente proxy y añada las clases de ayuda Java al proyecto.
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.

```
// crear el proxy
       BFMIFProxy proxy = new BFMIFProxy();
 // preparar los datos de entrada para la operación
      GetProcessTemplate iW = new GetProcessTemplate();
      iW.setIdentifier(nombre_plantilla_proceso);

 // invocar la operación
      GetProcessTemplateResponse oW = proxy.getProcessTemplate(iW);

 // operación de proceso de la operación
      ProcessTemplateType ptd = oW.getProcessTemplate();
      System.out.println("getName= " + ptd.getName());
      System.out.println("getPtid= " + ptd.getPtid());
```

Adición de seguridad (servicios Web Java)

Debe proteger las comunicaciones de los servicios Web implementando los mecanismos de seguridad en la aplicación cliente.

Por qué y cuándo se efectúa esta tarea

WebSphere Application Server da soporte actualmente a los siguientes mecanismos de seguridad para las API de servicios Web:

- El símbolo de nombre de usuario
- LTPA (Lightweight Third Party Authentication)

Implementación del símbolo de nombre de usuario:

El mecanismo de seguridad del símbolo de nombre de usuario proporciona los credenciales de nombre de usuario y contraseña.

Por qué y cuándo se efectúa esta tarea

Con el mecanismo de seguridad de símbolo de nombre de usuario, puede optar por implementar varios *manejadores de devolución de llamada*. Dependiendo de su elección.

- Se le solicitará que proporcione un nombre de usuario y una contraseña cada vez que ejecute la aplicación de cliente.
- El nombre de usuario y la contraseña se graban en el descriptor de despliegue.

En cualquiera de los casos, el nombre de usuario y la contraseña proporcionado deben coincidir con los de un rol autorizado en el contenedor de procesos empresarialeso contenedor de tareas de usuario correspondiente.

El nombre de usuario y la contraseña quedan encapsulados en el sobre del mensaje de la petición y, por lo tanto, aparecen ″claramente″ en la cabecera del mensaje SOAP. Por lo tanto, se le recomienda encarecidamente que configure la aplicación de cliente para que utilice el protocolo de comunicaciones HTTPS (HTTP en SSL). A continuación, se cifran todas las comunicaciones. Puede seleccionar el protocolo de comunicaciones HTTPS cuando especifique la dirección del URL del punto final de la API de servicio Web.

Para definir un símbolo de nombre de usuario:

Procedimiento

1. Cree un símbolo de seguridad:
   a. Abra el **editor de despliegue** del módulo.
   b. Pulse la pestaña correspondiente a la **extensión WS**.
   c. En las **referencias de servicio**, se pueden listar las siguientes referencias de servicio Web:
      - service/BFMWSService para procesos de empresa
      - service/HTMWSService para tareas de usuario

      Las referencias que se listen depende de si se ha añadido BFMWS.wsdl (para procesos de empresa), HTMWS.wsdl (para tareas de usuario), o ambas al generar el cliente proxy.
   d. Para ambas referencias de servicio:
      1) Seleccione una de las **referencias de servicio**.
      2) Expanda la sección **Configuración del generador de peticiones**.
      3) Expanda la subsección **Símbolo de seguridad**.
      4) Pulse **Añadir**. Se abrirá la ventana Símbolo de seguridad.
      5) En el campo **Nombre**, escriba un nombre para el nuevo símbolo de seguridad: **UserNameTokenBFM** o **UserNameTokenHTM** .
      6) En la lista desplegable **Tipo de símbolo**, seleccione **Username** (el campo **Nombre local** se rellena automáticamente con un valor por omisión).
      7) Deje el campo **URI** en blanco. No es necesario ningún valor de URI para un símbolo de nombre de usuario.
      8) Pulse **Aceptar**.
2. Cree un generador de símbolos:
   a. Abra el **editor de despliegue** del módulo.
   b. Pulse la pestaña correspondiente al **enlace WS**.
   c. En las **referencias de servicio**, se listan las mismas referencias de servicio Web que en el paso anterior:
      - service/BFMWSService para procesos de empresa
      - service/HTMWSService para tareas de usuario
   d. Para ambas referencias de servicio:
      1) Seleccione una de las **referencias de servicio**.
      2) Expanda la sección **Configuración de enlace del generador de peticiones de seguridad**.
      3) Expanda la subsección **Generador de símbolos**.

4) Pulse **Añadir**. Se abrirá la ventana Generador de símbolos.
5) En el campo **Nombre**, escriba un nombre para el nuevo generador de símbolos, como ″UserNameTokenGeneratorBFM″ o ″UserNameTokenGeneratorHTM″.
6) En el campo **Clase de generador de símbolos**, asegúrese de que se seleccione la siguiente clase de generador de símbolos: **com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator**.
7) En la lista desplegable **Símbolo de seguridad**, seleccione el símbolo de seguridad adecuado creado anteriormente.
8) Marque el recuadro de selección **Utilizar tipo de valor**.
9) En el campo **Tipo de valor**, seleccione **Símbolo Username** (el campo **Nombre local** se rellena automáticamente para reflejar lo que haya elegido en **Símbolo Username**.)
10) En el campo **Manejador de devolución de llamada** escriba ″com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler″ (que solicita el nombre de usuario y la contraseña cuando ejecuta la aplicación de cliente) o ″com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler″.
11) Si selecciona **NonPromptCallbackHandler**, debe especificar un nombre de usuario y una contraseña válidas en el campo correspondiente del descriptor de despliegue.
12) Pulse **Aceptar**.

Información relacionada

IBM WebSphere Developer Technical Journal: Web services security with WebSphere Application Server V6

Implementación del mecanismo de seguridad LTPA:

El mecanismo de seguridad LTPA (Lightweight Third Party Authentication) se puede utilizar cuando se ejecuta la aplicación de cliente en un contexto de seguridad establecido anteriormente.

Por qué y cuándo se efectúa esta tarea

El mecanismo de seguridad LTPA sólo está disponible si la aplicación de cliente se ejecuta en un entorno protegido en el que se ha establecido ya un contexto de seguridad. Por ejemplo, si la aplicación de cliente se ejecuta en un contenedor EJB (Enterprise JavaBeans), el cliente EJB debe iniciar la sesión para poder invocar la aplicación de cliente. A continuación, se establece un contexto de seguridad. Si la aplicación de cliente EJB invoca un servicio Web, el manejador de devoluciones de llamada LTPA recupera el símbolo LTPA del contexto de seguridad y lo añade al mensaje de petición SOAP. En el extremo del servidor, el símbolo LTPA lo maneja el mecanismo LTPA.

Para implementar el mecanismo de seguridad LTPA:

Procedimiento

1. En el entorno Rational Application Developer disponible en WebSphere Integration Developer, seleccione **Enlace WS → Configuración de enlace del generador de peticiones de seguridad → Generador de símbolos**.
2. Cree un símbolo de seguridad:
   a. Abra el **editor de despliegue** del módulo.

b. Pulse la pestaña correspondiente a la **extensión WS**.
c. En las **referencias de servicio**, se pueden listar las siguientes **referencias de servicio Web**:
   - service/BFMWSService para procesos de empresa
   - service/HTMWSService para tareas de usuario

   Las referencias que se listen depende de si se ha añadido BFMWS.wsdl (para procesos de empresa), HTMWS.wsdl (para tareas de usuario), o ambas al generar el cliente proxy.
d. Para ambas referencias de servicio:
   1) Seleccione una de las **referencias de servicio**.
   2) Expanda la sección **Configuración del generador de peticiones**.
   3) Expanda la subsección **Símbolo de seguridad**.
   4) Pulse **Añadir**. Se abrirá la ventana Símbolo de seguridad.
   5) En el campo **Nombre**, escriba un nombre para el nuevo símbolo de seguridad: **LTPATokenBFM** o **LTPATokenHTM**.
   6) En la lista desplegable **Tipo de símbolo**, seleccione **LTPAToken** (los campos **URI** y **Nombre local** se rellenan automáticamente con valores por omisión).
   7) Pulse **Aceptar**.

3. Cree un generador de símbolos:
   a. Abra el **editor de despliegue** del módulo.
   b. Pulse la pestaña correspondiente al **enlace WS**.
   c. En las **referencias de servicio**, se listan las mismas referencias de servicio Web que en el paso anterior:
      - service/BFMWSService para procesos de empresa
      - service/HTMWSService para tareas de usuario
   d. Para ambas referencias de servicio:
      1) Seleccione una de las **referencias de servicio**.
      2) Expanda la sección **Configuración de enlace del generador de peticiones de seguridad**.
      3) Expanda la subsección **Generador de símbolos**.
      4) Pulse **Añadir**. Se abrirá la ventana Generador de símbolos.
      5) En el campo **Nombre**, escriba un nombre para el nuevo generador de símbolos, como ″LTPATokenGeneratorBFM″ o ″LTPATokenGeneratorHTM″.
      6) En el campo **Clase de generador de símbolos**, asegúrese de que se seleccione la siguiente clase de generador de símbolos: **com.ibm.wsspi.wssecurity.token.LTPATokenGenerator**.
      7) En la lista desplegable **Símbolo de seguridad**, seleccione el símbolo de seguridad adecuado creado anteriormente.
      8) Marque el recuadro de selección **Utilizar tipo de valor**.
      9) En el campo **Tipo de valor**, seleccione **LTPAToken** (los campos **URI** y **Nombre local** se rellena automáticamente para reflejar lo que haya elegido en **Símbolo LTPA**).
      10) En el campo **Manejador de devolución de llamada**, escriba ″com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler″.
      11) Pulse **Aceptar**.

Resultados

Durante la ejecución, **LTPATokenCallbackHandler** recupera el símbolo LTPA del contexto de seguridad existente y lo añade al mensaje de petición SOAP.

Adición de soporte de transacciones (servicios Web Java)

Las aplicaciones del cliente de servicios Web Java se pueden configurar de modo que el proceso de las peticiones del extremo del servidor participen en la transacción del cliente pasando un contexto de la aplicación de cliente como parte de la petición de servicio. Este soporte de transacciones atómicas se define en la especificación WS-AT (Web Services-Atomic Transaction).

Por qué y cuándo se efectúa esta tarea

WebSphere Application Server ejecuta cada petición de la API de servicios Web como una transacción atómica. Las aplicaciones de cliente se pueden configurar para utilizar el soporte de transacciones de uno de los modos siguientes:

- Participar en la transacción. El proceso de peticiones del extremo del servidor se efectúa dentro del contexto de la transacción de la aplicación de cliente. Luego, si el servidor se encuentra con un problema cuando se está ejecutando la petición de la API de servicios Web y se retrotrae, la petición de la aplicación cliente también se retrotrae.
- No utilice el soporte de transacciones. WebSphere Application Server continúa creando una transacción nueva en la que se ejecuta la petición pero no se efectúa el proceso de peticiones en el extremo del servidor con el contexto de transacción de aplicaciones de cliente.

Desarrollo de aplicaciones cliente en el entorno .NET

Microsoft .NET ofrece un entorno de desarrollo completo en el que conectar aplicaciones mendiante servicios Web.

Generación de un cliente proxy (.NET)

Las aplicaciones cliente .NET utilizan un *cliente proxy* para interactuar con las API de servicio Web. Un cliente proxy protege las aplicaciones cliente de la complejidad del protocolo de mensajería de servicio Web.

Antes de empezar

Para crear un cliente proxy, primero debe exportar varios archivos WSDL del entorno WebSphere y copiarlos al entorno de programación cliente.

Nota: Si tiene el CD del cliente de WebSphere Process Server, puede copiar los archivos desde allí.

Por qué y cuándo se efectúa esta tarea

Un cliente proxy se compone de un conjunto de clases de bean de C#. Cada clase contiene todos los métodos y objetos expuestos por un solo servicio Web. Los métodos de servicio manejan el ensamblado de los parámetros en mensajes SOAP completos, envían los mensajes SOAP al servicio Web en HTTP, reciben las respuestas del servicio Web y gestionan los datos devueltos.

Nota: Sólo tiene que generar una vez el cliente proxy. Todas las aplicaciones cliente que acceden a las API de servicios Web pueden utilizar entonces el mismo cliente proxy.

Procedimiento

1. Utilice el mandato WSDL para generar un cliente proxy: Escriba:

   wsdl *opciones vía_acceso_archivo_WSDL*

   Donde:

   - *opciones* incluyen:

     /language
     Permite especificar el lenguaje utilizado para crear la clase proxy. Toma por omisión C#. También puede especificar **VB** (Visual Basic), **JS** (JScript) o **VJS** (Visual J#) de argumento de lenguaje.

     /output
     El nombre del archivo de salida, con el sufijo adecuado. Por ejemplo, proxy.cs

     /protocol
     El protocolo implementado en la clase proxy. Toma por omisión el valor **SOAP**.

     Si desea una lista completa de los parámetros **WSDL.exe**, utilice el conmutador de la línea de mandatos **/?** o bien consulte la ayuda en línea de la herramienta WSDL en Visual Studio.

   - *vía_acceso_archivo_WSDL* es la vía de acceso y nombre de archivo del archivo WSDL que ha exportado desde el entorno de WebSphere o que ha copiado del CD de cliente.

   En el ejemplo siguiente se genera un cliente proxy para la API de servicios Web Human Task Manager:

   ```
   wsdl /language:cs /output:proxyclient.cs c:\ws\bin\HTMWS.wsdl
   ```

2. Compile el cliente proxy como un archivo DLL (Dynamic Link Library).

Creación de clases de ayuda para procesos BPEL (.NET)

Determinadas operaciones de la API de servicios Web requieren que las aplicaciones cliente utilicen los elementos de envoltorio de estilo "document/literal". Las aplicaciones cliente requieren las clases de ayuda para que les ayuden a generar los elementos de envoltorio necesarios.

Antes de empezar

Para crear clases helper, debe haber exportado el archivo WSDL de la API de servicios Web del entorno WebSphere Process Server.

Por qué y cuándo se efectúa esta tarea

Las operaciones call() y sendMessage() de las API de servicios Web provocan que los procesos BPEL se inicien en WebSphere Process Server. El mensaje de entrada de la operación call() espera a que se proporcione la envoltura document/literal del mensaje de entrada del proceso BPEL. Para generar los beans y las clases necesarios para el proceso BPEL, copie el elemento `<wsdl:types>` en un archivo XSD nuevo, a continuación, utilice la herramienta xsd.exe para generar clases helper.

Procedimiento

1. Si todavía no lo ha hecho, exporte el archivo WSDL de la interfaz de proceso BPEL desde WebSphere Integration Developer.
2. Abra el archivo WSDL en un editor de texto o un editor XML.

3. Copie el contenido de todos los elementos hijo del elemento <wsdl:types> y péguelo a un nuevo archivo XSD esqueleto.
4. Ejecute la herramienta xsd.exe en el archivo XSD:

   call xsd.exe file.xsd /classes /o

   Donde:

   file.xsd
   Archivo de definición de esquema XML que se va a convertir.

   /classes (/c)
   Genere las clases de ayuda que corresponden al contenido del archivo o archivos XSD especificados.

   /output (/o)
   Especifique el directorio de salida de los archivos generados. Si se omite este directorio, el valor por omisión es el directorio actual.

   Por ejemplo:

   call xsd.exe ProcessCustomer.xsd /classes /output:c:\temp
5. Añada el archivo de clase que se genera a la aplicación cliente. Si utiliza Visual Studio, por ejemplo, puede hacer esto utilizando la opción de menú **Proyecto** → **Agregar elemento existente**.

Ejemplo

Si el archivo ProcessCustomer.wsdl contiene lo siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                  xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                  xmlns:wsdl="http://schemas. xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  name="ProcessCustomer"
                  targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
  <wsdl:types>
    <xsd:schema targetNamespace="http://ProcessTypes/bpel/ProcessCustomer"
                xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
                schemaLocation="xsd-includes/http.com.ibm.bpe.unittest.sca.xsd"/>
      <xsd:element name="doit">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="input1" nillable="true" type="bons1:Customer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="doitResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="output1" nillable="true" type="bons1:Customer"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
    <wsdl:message name="doitRequestMsg">
    <wsdl:part element="tns:doit" name="doitParameters"/>
  </wsdl:message>
    <wsdl:message name="doitResponseMsg">
    <wsdl:part element="tns:doitResponse" name="doitResult"/>
  </wsdl:message>
    <wsdl:portType name="ProcessCustomer">
```

```
    <wsdl:operation name="doit">
      <wsdl:input message="tns:doitRequestMsg" name="doitRequest"/>
      <wsdl:output message="tns:doitResponseMsg" name="doitResponse"/>
    </wsdl:operation>
  </wsdl:portType>
</wsdl:definitions>
```

El archivo XSD producido contendrá:

```
<xsd:schema xmlns:bons1="http://com/ibm/bpe/unittest/sca"
                      xmlns:tns="http://ProcessTypes/bpel/ProcessCustomer"
                      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                      targetNamespace="http://ProcessTypes/bpel/ProcessCustomer">
 <xsd:import namespace="http://com/ibm/bpe/unittest/sca"
              schemaLocation="Customer.xsd"/>
 <xsd:element name="doit">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="input1" type="bons1:Customer" nillable="true"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
 <xsd:element name="doitResponse">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element name="output1" type="bons1:Customer" nillable="true"/>
   </xsd:sequence>
  </xsd:complexType>
 </xsd:element>
</xsd:schema>
```

Información relacionada

Documentación de Microsoft para la herramienta de definición de esquemas XML (XSD.EXE)

Creación de aplicaciones cliente (.NET)

Las aplicaciones cliente envían las peticiones a las API de servicios Web y reciben las respuestas. Utilizando un cliente proxy para gestionar las comunicaciones y las clases de ayuda para dar formato a los 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 cliente proxy y las clases de ayuda necesarias.

Por qué y cuándo se efectúa esta tarea

Puede desarrollar las aplicaciones de cliente .NET utilizando cualquier herramienta de desarrollo compatible con .NET, por ejemplo, Visual Studio .NET. Puede crear cualquier tipo de aplicación .NET para llamar a las API de servicio Web genéricas.

Procedimiento

1. Cree un nuevo proyecto de aplicación cliente. Por ejemplo, cree una **aplicación Windows WinFX** en Visual Studio.
2. En las opciones del proyecto, añada una referencia al archivo DLL (Dynamic Link Library) del cliente proxy. Añada todas las clases helper que contienen

definiciones de objetos empresariales al proyecto. En Visual Studio, por ejemplo, puede hacerlo utilizando la opción **Proyecto → Agregar elemento existente**.

3. Cree un objeto cliente proxy. Por ejemplo:

```
HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService service =
    new HTMClient.HTMReference.HumanTaskManagerComponent1Export_HumanTaskManagerHttpService();
```

4. Declare los tipos de datos de objetos empresariales utilizados en los mensajes que se van a enviar al servicio Web o recibir. Por ejemplo:

```
HTMClient.HTMReference.TKIID id = new HTMClient.HTMReference.TKIID();

ClipBG bg = new ClipBG();
Clip clip = new Clip();
```

5. Llame a las funciones de servicio Web específicas y especifique los parámetros necesarios. Por ejemplo, para crear e iniciar una tarea de usuario:

```
HTMClient.HTMReference.createAndStartTask task =
   new HTMClient.HTMReference.createAndStartTask();
HTMClient.HTMReference.StartTask sTask = new HTMClient.HTMReference.StartTask();

sTask.taskName = "SimpleTask";
sTask.taskNamespace = "http://myProcess/com/acme/task";
sTask.inputMessage = bg;
task.inputTask = sTask;

id = service.createAndStartTask(task).outputTask;
```

6. Las tareas y procesos remotos se identifican con los ID permanentes (`id` en el ejemplo del paso anterior). Por ejemplo, para reclamar una tarea de usuario creada previamente:

```
HTMClient.HTMReference.claimTask claim = new HTMClient.HTMReference.claimTask();
claim.inputTask = id;
```

Adición de la seguridad (.NET)

Puede proteger las comunicaciones de los servicios Web integrando mecanismos de seguridad en la aplicación cliente.

Por qué y cuándo se efectúa esta tarea

Estos mecanismos de seguridad pueden incluir el símbolo de userName (nombre de usuario y contraseña) o los símbolos binarios personalizados y los símbolos de seguridad basados en XML.

Procedimiento

1. Baje e instale WSE (Web Services Enhancements) 2.0 SP3 para Microsoft .NET. Este producto está disponible en:

   http://www.microsoft.com/downloads/details.aspx?familyid=1ba1f631-c3e7-420a-bc1e-ef18bab66122&displaylang=en

2. Modifique el código cliente de proxy generado como se detalla a continuación.

   Cambie:

```
public class Export1_MyMicroflowHttpService : System.Web.Services.Protocols.SoapHttpClientProtocol {
```

   Por:

```
public class Export1_MyMicroflowHttpService : Microsoft.Web.Services2.WebServicesClientProtocol {
```

   Nota: Estas modificaciones se pierden si vuelve a generar el cliente proxy ejecutando la herramienta WSDL.exe.

3. Modifique el código de aplicación cliente añadiendo las siguientes líneas al principio del archivo:

```
using System.Web.Services.Protocols;
using Microsoft.Web.Services2;
using Microsoft.Web.Services2.Security.Tokens;
...
```

4. Añada código para implementar el mecanismo de seguridad deseado. Por ejemplo, el código siguiente añade una protección de nombre de usuario y contraseña:

```
string user = "U1";
string pwd = "password";
UsernameToken token =
   new UsernameToken(user, pwd, PasswordOption.SendPlainText);

me._proxy.RequestSoapContext.Security.Tokens.Clear();
me._proxy.RequestSoapContext.Security.Tokens.Add(token);
```

Consulta sobre los objetos de procesos empresarialesy relativos a tareas

Puede utilizar las API de servicios Web para consultar los objetos de procesos empresarialesy relativos a tareas de la base de datos de Business Process Choreographer con el fin de recuperar propiedades específicas de estos objetos.

Por qué y cuándo se efectúa esta tarea

La base de datos de Business Process Choreographer almacena datos de plantilla (modelo) y de instancia (tiempo de ejecución) para gestionar procesos empresarialesy tareas.

Mediante las API de servicios Web, las aplicaciones cliente pueden emitir consultas para recuperar información de la base de datos sobre los procesos y tareas de empresa.

Las aplicaciones cliente pueden emitir una consulta específica para recuperar una propiedad concreta de un objeto. Habitualmente, se pueden guardar las consultas que utiliza. Luego la aplicación cliente puede recuperar y utilizar estas consultas almacenadas.

Consultas sobre los objetos de procesos empresarialesy relativos a tareas utilizando las API de servicios Web

Utilice la interfaz de consulta de las API de servicios Web para obtener información sobre los procesos empresarialesy tareas.

Las aplicaciones de cliente utilizan una sintaxis de tipo SQL para consultar la base de datos.

Ejemplos de servicios Web Java

```
string processTemplateName = "ProcessCustomerLR";
query query1 = new query();
query1.selectClause = "DISTINCT PROCESS_INSTANCE.STARTED, PROCESS_INSTANCE.PIID";
query1.whereClause =
       "PROCESS_INSTANCE.TEMPLATE_NAME = '" + processTemplateName + "'";
query1.orderByClause = "PROCESS_INSTANCE.STARTED";
query1.threshold = null;
query1.timeZone = "UTC"; query1.skipTuples = null;
queryResponse queryResponse1 = proxy.query(query1);
```

La información recuperada de la base de datos se devuelve mediante las API de servicios Web como un *conjunto de resultados de consulta*.

Por ejemplo:

```
QueryResultSetType queryResultSet = queryResponse1.queryResultSet;
if (queryResultSet != null) {
   Console.WriteLine("--> QueryResultSetType");
   Console.WriteLine(" . size= " + queryResultSet.size);
   Console.WriteLine(" . numberColumns= " + queryResultSet.numberColumns);
   string indent = " . ";

   // -- la información de la columna de consulta
   QueryColumnInfoType[] queryColumnInfo = queryResultSet.QueryColumnInfo;
   if (queryColumnInfo.Length > 0) {
      Console.WriteLine();
      Console.WriteLine("= . QueryColumnInfoType size= " + queryColumnInfo.Length);
      Console.Write( " | tableName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].tableName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | columnName ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         Console.Write( " | " + queryColumnInfo[i].columnName.PadLeft(20) );
      }
      Console.WriteLine();
      Console.Write( " | data type ");
      for (int i = 0; i < queryColumnInfo.Length ; i++) {
         QueryColumnInfoTypeType tt = queryColumnInfo[i].type;
         Console.WriteLine( " | " + tt.ToString());
      }
         Console.WriteLine();
   }
   else {
      Console.WriteLine("--> queryColumnInfo= <null>");
   }

   // - los valores del resultado de la consulta
   string[][] result = queryResultSet.result;
   if (result !=null) {
      Console.WriteLine();
      Console.WriteLine("= . result size= " + result.Length);
      for (int i = 0; i &lt; result.Length; i++) {
        Console.Write(indent +i );
        string[] row = result[i];
        for (int j = 0; j &lt; row.Length; j++ ) {
           Console.Write(" | " + row[j]);
        }
        Console.WriteLine();
      }
   }
   else {
      Console.WriteLine("--> result= <null>");
   }
}
else {
   Console.WriteLine("--> QueryResultSetType= <null>");
}
```

La función query devuelve objetos según la autorización del llamante. El conjunto de resultados de consulta sólo contiene las propiedades de aquellos objetos que el llamante está autorizado a ver.

Se proporcionan vistas de bases de datos predefinidas para que se consulten las propiedades de objeto. Para plantillas de proceso, la función de consulta tiene la sintaxis siguiente:

```
ProcessTemplateData[] queryProcessTemplates
                       (java.lang.String whereClause,
                        java.lang.String orderByClause,
                        java.lang.Integer threshold,
                        java.util.TimeZone timezone);
```

Para plantillas de tarea, la función de consulta tiene la sintaxis siguiente:

```
TaskTemplate[] queryTaskTemplates
                (java.lang.String whereClause,
                 java.lang.String orderByClause,
                 java.lang.Integer threshold,
                 java.util.TimeZone timezone);
```

Para los demás objetos de proceso de empresa, la función de consulta tiene la sintaxis siguiente:

```
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);
```

La interfaz de consulta también contiene un método queryAll. Puede utilizar este método para recuperar todos los datos relevantes acerca de un objeto como, por ejemplo, a fines de supervisión. El llamante del método queryAll debe tener uno de los siguientes roles de J2EE (Java 2 Platform, Enterprise Edition): BPESystemAdministrator, BPESystemMonitor, TaskSystemAdministrator o TaskSystemMonitor. No se aplica la comprobación de autorización utilizando el elemento de trabajo correspondiente del objeto.

Ejemplo para .NET

```
ProcessTemplateType[] templates = null;

   try {
        queryProcessTemplates iW = new queryProcessTemplates();
        iW.whereClause = "PROCESS_TEMPLATE.STATE=PROCESS_TEMPLATE.STATE.STATE_STARTED";
        iW.orderByClause = null;
        iW.threshold = null;
        iW.timeZone = null;

        Console.WriteLine("--> queryProcessTemplates ... ");
        Console.WriteLine("--> query: WHERE " + iW.whereClause + " ORDER BY " +
          iW.orderByClause + " THRESHOLD " + iW.threshold + " TIMEZONE" + iW.timeZone);

        templates = proxy.queryProcessTemplates(iW);

        if (templates.Length < 1) {
          Console.WriteLine("--> No templates found :-(");
        }
        else {
          for (int i = 0; i < templates.Length ; i++) {
            Console.Write("--> found template with ptid: " + templates[i].ptid);
            Console.WriteLine(" and name: " + templates[i].name);
           /*  ... other properties of ProcessTemplateType ... */
          }
        }
      }
      catch (Exception e) {
        Console.WriteLine("exception= " + e);
      }
```

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).

Por qué y cuándo se efectúa 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.

Gestión de consultas almacenadas públicas
Las consultas almacenadas públicas las crea el administrador del sistema. Estas consultas están disponibles para todos los usuarios.

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.

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.

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).

Por qué y cuándo se efectúa 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.

Conceptos relacionados

"Comparación de las interfaces de programación para interactuar con procesos empresarialesy tareas de usuario" en la página 187
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 empresarialesy tareas de usuario. Cada una de estas interfaces tiene características diferentes.

Requisitos para los procesos empresariales

Los procesos empresarialesdesarrollados 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 empresarialesse 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 empresarialesy tareas de usuario desarrollados con WebSphere Integration Developer.
2. Los mensajes de error que exponen los procesos empresarialesy 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 empresarialestá 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**:

Petición	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 de empresa. Un administrador de procesos empresarialeses un rol especial; es diferente del administrador de procesos o de una instancia de proceso. Un administrador de procesos empresarialestiene 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 a través de la interfaz JMS, una aplicación primero debe crear una conexión a BPC.cellname.Bus, crear una sesión y, a continuación, generar productores y consumidores de mensajes.

Por qué y cuándo se efectúa esta tarea

El servidor de procesos acepta mensajes JMS (Java Message Service) que sigan el paradigma de punto a punto. Una aplicación que envíe o reciba mensajes JMS debe efectuar las acciones siguientes.

En el ejemplo siguiente se da por supuesto que el cliente JMS se ejecuta en un entorno gestionado (EJB, cliente de aplicaciones o contenedor de cliente Web). Si desea ejecutar el cliente JMS en un entorno J2SE, consulte el documento ″IBM Client for JMS on J2SE with IBM WebSphere Application Server″ en http://www-1.ibm.com/support/docview.wss?uid=swg24012804.

Procedimiento

1. Cree una conexión con BPC.*cellname*.Bus. No existe una fábrica de conexiones preconfigurada para las peticiones de una aplicación de cliente: una aplicación de cliente puede utilizar ReplyConnectionFactory de la API de JMS o crear su propia fábrica de conexiones, en cuyo caso puede utilizar 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 nombre especificado al configurar la cola de peticiones externas de Business Process Choreographer. En el ejemplo siguiente se da por supuesto que la aplicación de cliente crea su propia fábrica de conexiones llamada ″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 puedan crearse los productores y consumidores de mensajen.

   ```
   // Crear una sesión de transacción mediante autoreconocimiento.
   Session session = connection.createSession(true, Session.AUTO_ACKNOWLEDGE);
   ```

3. Cree un productor de mensajes para enviar mensajes. El nombre de búsqueda JNDI debe ser el mismo que el nombre especificado al configurar la cola de peticiones externas de Business Process Choreographer.

   ```
   // Buscar el destino de la cola de entrada de Business Process Choreographer
   // a la que 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 por omisión (definido por Business Process Choreographer) `jms/BFMJMSReplyQueue`. En ambos casos, el destino de respuesta debe encontrarse en `BPC.<cellname>.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.
// Este método está definido en otro lugar...
String payload = createXMLDocumentForRequest();
TextMessage requestMessage = session.createTextMessage(payload);

// Establecer cabecera JMS obligatoria.
// targetFunctionName es el nombre de operación de la API de JMS
// (por ejemplo, getProcessTemplate, sendMessage)
requestMessage.setStringProperty("TargetFunctionName", targetFunctionName);

// Establecer la cola de respuesta; esto es obligatorio si replyQueue
// no es la cola por omisión (como sucede 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 se haya iniciado una conexión, se puede intercambiar cualquier número de mensajes de petición y respuesta antes de cerrar la conexión. En el ejemplo se muestra un caso simple con una sola llamada en un único método de empresa.

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 por omisión de la interfaz de exportación (una exportación es una representación de interfaz de cliente de un componente de proceso de empresa). 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 con error 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

El cuerpo de mensaje JMS es una serie que contiene un documento XML que representa el elemento de envoltorio document/literal de la operación.

Un ejemplo simple de un cuerpo de mensaje de petición válido es:

```
<?xml version="1.0" encoding="UTF-8"?>
<_6:queryProcessTemplates xmlns:_6="http://www.ibm.com/xmlns/prod/
      websphere/business-process/services/6.0">
<whereClause>PROCESS_TEMPLATE.STATE IN (1)</whereClause>
</_6:queryProcessTemplates>
```

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.

Por qué y cuándo se efectúa 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

Puede obtener estos artefactos de las siguientes maneras:

- Publicar y exportar los artefactos desde el entorno de WebSphere Process Server.

  Estos artefactos de cliente se encuentran en el directorio *raíz_instalación*\ ProcessChoreographer\client.
- Copiar los archivos desde directorio *raíz_instalación*\ProcessChoreographer\ client del CD del cliente de WebSphere Process Server.

Resultados

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.

Por qué y cuándo se efectúa 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
}
```

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 307.
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 empresarialesy 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 empresarialesy de tareas de usuario a las aplicaciones Web.

Por qué y cuándo se efectúa 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

   Si va a desplegar la aplicación Web en un servidor remoto, añada también los archivos siguientes. Estos archivos son necesarios para acceder de forma remota a las API de Business Process Choreographer.

   - bpe137650.jar
   - task137650.jar

   En WebSphere Process Server, estos archivos se encuentran en el directorio siguiente:

   - En los sistemas Windows: *install_root*\ProcessChoreographer\client
   - En los sistemas UNIX, Linux e i5/OS: *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. Habitualmente, se necesitan las bibliotecas de códigos JSF y HTML, y la biblioteca de códigos que necesitan los componentes de 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 Network Deployment, 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 empresarialesestán 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 empresarialesestán 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 16. Correlación de los enlaces de referencia a 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

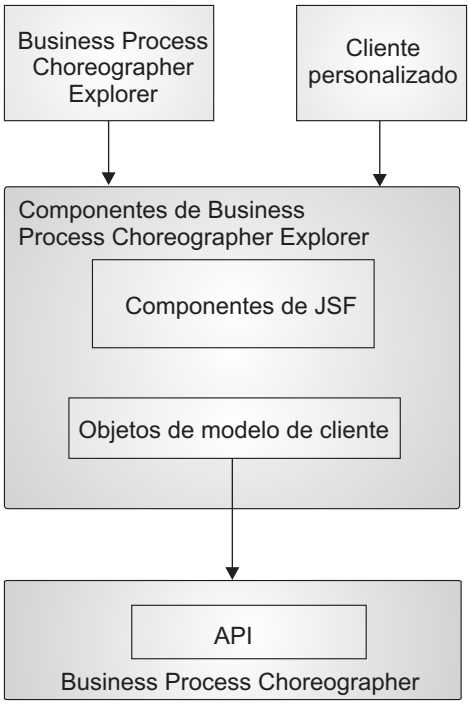
Si utiliza JSP personalizados para los mensajes de procesos y de tareas, debe correlacionar los módulos Web que se utilizan para desplegar los JSP con los mismos servidores con los que está correlacionado el cliente JSF personalizado.

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 empresarialesy 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 empresarialesy 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 un 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();
}
```

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 17. 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 empresarialo 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
    // Gestor de tareas de usuario.
    //
    // 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 por omisión para la lista y SimpleConverter como convertidor por omisión 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.

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 18. 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.
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 19. 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 20. 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	El 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 21. 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.

Por qué y cuándo se efectúa 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;
  }

 // Implementaciones por omisión
 public boolean isMultiSelectEnabled() { return false; }
 public boolean[] isApplicable(List itemsOnList) {return null; }
 public void setContext(Object targetModel) {; // No se utiliza aquí }
}
```

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. 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. 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 22. 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.

Tabla 22. Atributos de bpe:commandbar (continuación)

Atributo	Necesario	Descripción
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 23. 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 por omisión es `false`.
label	sí	Etiqueta del botón que se representa en la barra de mandatos.
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	eClase 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).

Por qué y cuándo se efectúa 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 24. Atributos de bpe:form

Atributo	Necesario	Descripción
id	no	El 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`, las propiedades que contienen tipos simples y cuya cardinalidad es cero o uno no se muestran. 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.
styleClass4invalidInput	no	Nombre de clase de estilo CSS para representar la entrada que no es válida.
styleClass4output	no	Nombre de clase del 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.

Tabla 24. Atributos de bpe:form (continuación)

Atributo	Necesario	Descripción
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.

Por qué y cuándo se efectúa 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.

Por qué y cuándo se efectúa 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

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.

Por qué y cuándo se efectúa esta tarea

Complete los pasos siguientes para crear un manejador de sucesos de API.

Procedimiento

1. Escriba una clase que implementa la interfaz APIEventHandlerPlugin3 o amplíe la clase de implementación APIEventHandler. Esta clase puede invocar los métodos de otras clases.
   - Si utiliza la interfaz APIEventHandlerPlugin3, debe implementar todos estos métodos de la interfaz APIEventHandlerPlugin3 y la interfaz APIEventHandlerPlugin.

- Si amplía la clase de implementación APIEventHandler, sobrescriba los métodos que necesite.

Este clase se ejecuta en el contexto de una aplicación Java 2 Enterprise Edition (J2EE) 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 la interfaz del proveedor de servicio de Java 2.

   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.APIEventHandlerPlugin3, 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 una 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 una 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 pero se ignoran.

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

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.

Por qué y cuándo se efectúa 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 2 Enterprise Edition (J2EE). 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 J2EE utilizan las clases helper, puede empaquetar estas clases en un archivo JAR individual que puede registrar 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 la interfaz del proveedor de servicio de Java 2.

   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 los plug-ins del manejador de sucesos de API y del manejador de sucesos de notificación

Para utilizar los plug-ins del manejador de sucesos de API y del manejador de sucesos de notificación, debe instalar el plug-in para que el contenedor de tareas pueda acceder al mismo.

Por qué y cuándo se efectúa esta tarea

La manera en que puede instalar el plug-in depende de que el plug-in lo vaya a utilizar una sola aplicación J2EE (Java2 Enterprise Edition) o varias.

Complete uno de los pasos siguientes para instalar un plug-in.

- Instale un plug-in para que sólo lo utilice una aplicación J2EE.

  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 del plug-in como un archivo JAR de programa de utilidad de proyecto para la aplicación J2EE del módulo EJB (Enterprise JavaBeans) principal.
- Instale un plug-in para que lo utilicen varias aplicaciones J2EE.

  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 los manejadores de sucesos de notificación con tareas, plantillas de tarea y modelos de tarea en distintos momentos: cuando cree una tarea ad-hoc, actualice una tarea existente, cree un modelo de tarea ad-hoc o defina una plantilla de tarea.

Por qué y cuándo se efectúa 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.

- 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 cuando cree 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.

Creación, instalación y ejecución de los plug-ins para el proceso posterior de los resultados de consultas de personal

La resolución de personal devuelve una lista de los usuarios que se asignan a un rol específico, por ejemplo, el propietario potencial de una tarea. Puede crear un plug-in para cambiar los resultados de las consultas de personas devueltos por la resolución de personas. Por ejemplo, para mejorar el equilibrio de la carga de trabajo, podría disponer de un plug-in que elimina del resultado de consulta los usuarios que ya tienen una alta carga de trabajo.

Por qué y cuándo se efectúa esta tarea

Sólo puede tener un plug-in de proceso posterior; esto significa que el plug-in debe manejar los resultados de consultas de personal de todas las tareas. El plug-in puede añadir o eliminar usuarios o cambiar la información de usuario o grupo. También puede cambiar el tipo de resultado, por ejemplo, de una lista de usuarios por un grupo o por todos.

Dado que se ejecuta el plug-in después de que finaliza la resolución de personas, ya se han aplicado las reglas que tenga para mantener la confidencialidad o la seguridad. El plug-in recibe información sobre los usuarios que se han eliminado durante la resolución de personal (en la clave de correlación HTM_REMOVED_USERS). Debe asegurarse de que el plug-in utiliza esta información de contexto para conservar cualquier confidencialidad o las normas de seguridad que puede tener.

Para implementar el proceso posterior a los resultados de consulta de personas, puede utilizar la interfaz StaffQueryResultPostProcessorPlugin. La interfaz dispone de métodos para modificar los resultados de consulta para tareas, escaladas, plantillas de tarea y componentes de la aplicación.

Complete los pasos siguientes para crear un plug-in para el proceso posterior de los resultados de consulta de personas.

Procedimiento

1. Grabe una clase que implementa la interfaz StaffQueryResultPostProcessorPlugin.

   Este clase se ejecuta en el contexto de una aplicación Java 2 Enterprise Edition (J2EE) Enterprise. Esta clase puede invocar los métodos de otras clases. 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.

   Debe implementar todos los métodos de la interfaz. Estos métodos incluyen información relativa a la asignación de criterios de asignación de personas para la plantilla de tarea, rol de tarea o escalada.

   - La definición de los criterios de asignación de personas se especifica como una entrada en el parámetro **context** de tipo `Map`. Para acceder a esta información, prosiga tal como se indica a continuación:

     ```
     Map pacAsMap = (Map) context.get("HTM_VERB");

     // para recuperar el nombre del PAC
     String pacName = (String) pacAsMap.get("HTM_VERB_NAME");

     // para recuperar los nombres del parámetro PAC
     Set paramNames = pacAsMap.keySet();

     // para recuperar el valor de un parámetro específico
     String paramValue = (String) pacAsMap.get(paramName);
     ```

   - Las variables de sustitución especificadas como valores de parámetro de los criterios de asignación de personas son entradas del parámetro **context** de tipo `Map`. Para acceder a esta información, prosiga tal como se indica a continuación:

     ```
     Object replVarObj = pacAsMap.get(replVarName);
     if (replVarObj instanceof String)
          String replVarValue = (String) replVarObj;
     if (replVarObj instanceof String[])
          String[] replVarValues = (String[]) replVarObj;
     ```

   - El objeto StaffQueryResult que se ha creado al acceder a un directorio de personas durante la resolución de personas, por ejemplo, al acceder al directorio de personas de Virtual Member Manager.

     El objeto StaffQueryResult contiene la información relativa a las entradas de usuario que se hayan recuperado durante la resolución de personas. Para obtener más información, consulte la información de consulta de Javadoc para la interfaz StaffQueryResultPostProcessorPlugin.

   - La lista de usuarios que la resolución de personas haya excluido explícitamente aparece como una entrada del parámetro **context** de tipo `Map`. Para acceder a esta información, prosiga tal como se indica a continuación:

     ```
     String[] removedUserIDs = (String[]) context.get("HTM_REMOVED_USERS");
     ```

   En el ejemplo siguiente se muestra cómo podría cambiar el rol de editor de una tarea denominada SpecialTask.

   ```
   public StaffQueryResult processStaffQueryResult
                          (StaffQueryResult originalStaffQueryResult,
                                            Task task,
                                            int  role,
                                            Map  context)
   {
     StaffQueryResult newStaffQueryResult = originalStaffQueryResult;
     StaffQueryResultFactory staffResultFactory =
                            StaffQueryResultFactory.newInstance();
   ```

```
 if (role == com.ibm.task.api.WorkItem.REASON_EDITOR &&
     task.getName() != null &&
     task.getName().equals("SpecialTask"))
 {
     UserData user = staffResultFactory.newUserData
                          ("SuperEditor",
                            new Locale("en-US"),
                           "SuperEditor@company.com");
     ArrayList userList = new ArrayList();
     userList.add(user);

     newStaffQueryResult = staffResultFactory.newStaffQueryResult(userList);
  }
  return(newStaffQueryResult);
}
```

2. Ensamble la clase de plug-in y sus clases de ayuda en un archivo JAR.

   Puede hacer que el archivo JAR esté disponible como una biblioteca compartida y asociarlo al contenedor de tareas. De esta forma, 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 la interfaz del proveedor de servicio de Java 2.

   a. Cree un archivo con el nombre `com.ibm.task.spi.`*`nombre_plug-in`*`StaffQueryResultPostProcessorPlugin`, donde *nombre_plug-in* es el nombre del plug-in.

      Por ejemplo, si el plug-in se denomina `MyHandler` e implementa la interfaz com.ibm.task.spi.StaffQueryResultPostProcessorPlugin, el nombre del archivo de configuración será `com.ibm.task.spi.`**`MyHandler`**`StaffQueryResultPostProcessorPlugin`.

   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 `StaffPostProcessor` 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.StaffPostProcessor`. Tiene un archivo JAR instalable que contiene un plug-in que realiza un proceso posterior de los resultados de la consulta de personas y un archivo de configuración de proveedor de servicio que se puede utilizar para cargar el plug-in.

4. Instale el plug-in.

   Puede tener sólo un plug-in de proceso posterior para los resultados de consulta de personas. Debe instalar el plug-in como una biblioteca compartida.

   a. Defina una biblioteca compartida de WebSphere Application Server para el plug-in. Defina la biblioteca compartida en el ámbito del servidor o del clúster donde está configurado Business Process Choreographer. A continuación, asocie esta biblioteca compartida con la aplicación TaskContainer. Este paso debe realizarse sólo una vez.

   b. Haga que el archivo JAR del plug-in esté disponible para cada instalación de WebSphere Process Server afectada que aloje un miembro de servidor o de clúster.

5. Registre el plug-in.

a. En la consola administrativa, vaya a la página Propiedades personalizadas de Human Task Manager

   Pulse **Servidores** → **Servidores de aplicaciones** → ***nombre_servidor*** en un entorno autónomo, o **Servidores** → **Clústeres** → ***nombre_clúster*** si Business Process Choreographer está configurado en un clúster. En **Business Integration**, seleccione **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 ha asignado al plug-in, en este ejemplo `MyHandler`.

Ahora el plug-in está disponible para el proceso posterior de los resultados de consultas de personal. Si modifica el archivo JAR, sustituya el archivo en la biblioteca compartida y reinicie el servidor.

6. Ejecute 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. El plug-in se invoca con la información que especifica la interfaz StaffQueryResultPostProcessorPlugin.

Parte 2. Despliegue de aplicaciones

Capítulo 14. 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 de instalación de aplicaciones en un entorno de producción es similar al proceso descrito en "Desarrollo y despliegue de aplicaciones" en el Centro de información de WebSphere Application Server Network Deployment, versión 6. 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

Los módulos suelen utilizar artefactos que están ubicados en bibliotecas. Los artefactos y bibliotecas están contenidos en archivos Java (JAR) que se identifican al desplegar un módulo.

Al desarrollar un módulo, puede que haya identificado ciertos recursos o componentes que diversos elementos del módulo podrían utilizar. Estos recursos o componentes pueden ser objetos que se crearon al desarrollar el módulo u objetos ya existentes que residen en una biblioteca que ya está desplegada en el servidor. Este tema describe las bibliotecas y los archivos que se necesitarán al instalar una aplicación.

¿Qué es una biblioteca?

Una biblioteca contiene objetos y recursos utilizados por varios módulos de WebSphere Integration Developer. Los artefactos pueden estar en archivos JAR, archivos de recursos (RAR) o archivos de servicio Web (WAR). Algunos de estos artefactos son los siguientes:

- Descriptores de interfaces o servicios Web (archivos con la extensión .wsdl)
- Definiciones de esquema XML de objetos empresariales (archivos con la extensión .xsd)
- Correlaciones de objetos empresariales (archivos con la extensión .map)
- Definiciones de relación y de rol (archivos con la extensión .rel y .rol)

Cuando un módulo necesita un artefacto, el servidor localiza el artefacto a partir de la vía de acceso de clase EAR y carga el artefacto en la memoria, si no está cargado ya. A partir de ese momento, cualquier petición del artefacto utilizará esa copia hasta que sea sustituida. La Figura 23 muestra cómo una aplicación contiene componentes y bibliotecas relacionadas.

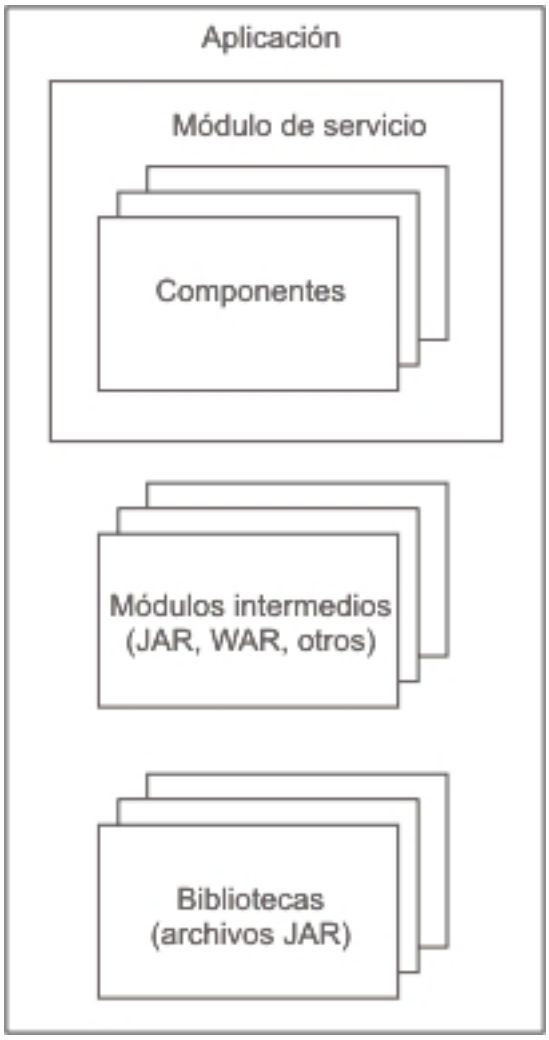


Figura 23. Relación entre módulo, componente y biblioteca

¿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 la aplicación por completo, necesita este archivo JAR, otras bibliotecas como archivos JAR, archivos WAR (archivadores de servicios Web) y RAR (archivadores de recursos), archivos JAR EJB (Enterprise Java Beans) de bibliotecas intermedias o cualquier

otro archivador, y crear un archivo EAR instalable mediante 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*EJB
- *myService*EJBClient
- *myService*Web

Nota: El mandato serviceDeploy sólo crea el módulo intermedio Web *myService* si el servicio incluye un servicio de tipo de puerto WSDL.

Consideraciones al utilizar bibliotecas

El uso de bibliotecas proporciona coherencia de objetos empresariales 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 empresariales 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 (Enterprise Archive) 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 a partir de los archivos de entrada que contienen las descripciones de componente y código Java que abarcan 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.

Por qué y cuándo se efectúa 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 classpath de manera que el mandato serviceDeploy pueda localizar los demás módulos que el módulo necesita.

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 producciónInstalación 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 15. Instalación de aplicaciones de procesos empresarialesy 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.

Por qué y cuándo se efectúa 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 empresarialesy 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 empresarialo 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 empresarialesy las tareas de usuario

Cuando WebSphere Integration Developer o el despliegue de servicio genera el código de despliegue para el proceso o tarea, cada componente de proceso o componente de tarea se correlaciona con un enterprise bean de sesión. Todo el código de despliegue se empaqueta en el archivo de aplicación de empresa (EAR). Además, para cada proceso, una clase Java que representa código Java en este proceso se genera y se incorpora en el archivo EAR durante la instalación de la aplicación de empresa. 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 empresarialeso 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. 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 empresarialni 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 de empresa.
- Si utiliza el despliegue de servicios, la fecha de válido-desde es la fecha en la que se ejecutó el mandato serviceDeploy. Sólo 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 empresarialesy 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.

Por qué y cuándo se efectúa esta tarea

Efectúe los pasos siguientes para instalar las aplicaciones de proceso empresarialde 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.

Por qué y cuándo se efectúa 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 empresarialesy tareas de usuario utilizando la consola administrativa.

Puede utilizar la consola administrativa para desinstalar aplicaciones que contienen procesos empresarialeso tareas de usuario.

Antes de empezar

Para desinstalar una aplicación que contenga procesos empresarialeso tareas de usuario, deben cumplirse los siguiente requisitos previos:

- 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 este servidor deben estar en ejecución. El servidor debe tener acceso a la base de datos Business Process Choreographer.
- No hay presentes instancias de plantillas de proceso empresarialni de tarea de usuario en ningún estado, a menos que utilice la opción **-force**.

Por qué y cuándo se efectúa esta tarea

Para desinstalar una aplicación de empresa que contenga procesos empresarialeso tareas de usuario, siga estos pasos:

Procedimiento

1. Verifique que están en ejecución: la base de datos, al menos un servidor de aplicaciones para cada clúster y el servidor autónomo donde se despliega la aplicación.

   En un entorno de Network Deployment, el servidor de despliegue, todos los servidores de aplicaciones autónomos gestionados y, como mínimo, un servidor de aplicaciones deben estar en ejecución para cada clúster donde esté instalada la aplicación.
2. Verifique que la aplicación no tiene instancias de proceso empresarialo de tarea de usuario.

   Si es necesario, un administrador puede utilizar Business Process Choreographer Explorer para suprimir instancias o tareas de proceso. No es necesario que detenga las plantillas de proceso y tarea porque se detienen automáticamente cuando se desinstala la aplicación.
3. Detenga y desinstale la aplicación:
   a. Pulse **Aplicaciones → Aplicaciones de empresa** en el panel de navegación de la consola administrativa.
   b. Seleccione la aplicación que desee desinstalar y pulse **Detener**.

      Este paso produce un error si todavía existe alguna instancia de proceso o de tarea en la aplicación.
   c. Seleccione de nuevo la aplicación que desea desinstalar y pulse **Desinstalar**.
   d. Pulse **Guardar** para guardar los cambios.

Resultados

Se desinstalará la aplicación.

Desinstalación de aplicaciones de procesos empresarialesy tareas de usuario utilizando mandatos administrativos

Los mandatos administrativos proporcionan una alternativa a la consola administrativa para desinstalar aplicaciones que contienen procesos empresarialeso tareas de usuario.

Antes de empezar

Para desinstalar una aplicación que contenga procesos empresarialeso tareas de usuario, deben cumplirse los siguiente requisitos previos:

- 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 este servidor deben estar en ejecución. El servidor debe tener acceso a la base de datos Business Process Choreographer.
- No hay presentes instancias de plantillas de proceso empresarialni de tarea de usuario en ningún estado, a menos que utilice la opción **-force**.

Además, si está habilitada la seguridad administrativa, verifique que el ID de usuario tiene autorización de administrador u operador. Para utilizar la opción **-force**, necesita tener autorización de administrador.

Asegúrese de que el proceso servidor con el que se conecta el cliente de administración 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.

Por qué y cuándo se efectúa esta tarea

En los pasos siguientes se describe cómo utilizar el script bpcTemplates.jacl para desinstalar aplicaciones que contienen plantillas de procesos empresarialeso plantillas de tareas de usuario.

Antes de desinstalar las aplicaciones, puede suprimir todas las instancias de proceso o de tarea asociadas a las plantillas de las aplicaciones, por ejemplo, utilizando Business Process Choreographer Explorer. También puede utilizar la opción **-force** con el script bpcTemplates.jacl para suprimir todas las instancias que están asociadas a las plantillas, detenerlas plantillas y desinstalarlas en un paso.

PRECAUCIÓN:
Dado que la opción -force suprime todos los datos de la instancia de proceso y la instancia de tareas, debe utilizar esta opción con precaución.

Procedimiento

1. Vaya al directorio de ejemplos de Business Process Choreographer.
   En las plataformas Windows, introduzca:
   `cd raíz_instalación\ProcessChoreographer\admin`
   En las plataformas Linux, UNIX e i5/OS, introduzca:
   `cd raíz_instalación/ProcessChoreographer/admin`

2. Detenga las plantillas y desinstale la aplicación correspondiente.

   En las plataformas Windows, introduzca:

   ```
   raíz_instalación\bin\wsadmin -f bpcTemplates.jacl
                                [-user nombre_usuario]
                                [-password contraseña de usuario]
                                -uninstall nombre_aplicación
                                [-force]
   ```

   En las plataformas Linux, UNIX e i5/OS, introduzca:

   ```
   raíz_instalación/bin/wsadmin -f bpcTemplates.jacl
                                [-user nombre_usuario]
                                [-password contraseña de usuario]
                                -uninstall nombre_aplicación
                                [-force]
   ```

   Donde:

   nombre_usuario
   Si la seguridad administrativa está habilitada, proporcione el ID de usuario para la autenticación.

   contraseña_usuario
   Si está habilitada la seguridad administrativa, proporcione la contraseña de usuario para la autenticación.

   nombre_aplicación
   Proporciona el nombre de la aplicación que se va a desinstalar.

Resultados

Se desinstalará la aplicación.

Capítulo 16. 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 se describe en Configuración y utilización de adaptadores en el Centro de información de WebSphere Integration Developer.

Capítulo 17. 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.

Por qué y cuándo se efectúa 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 J2EE 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 J2C 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 J2C (consulte Supresión de los destinos de SIBus y Supresión de las especificaciones de activación J2C 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 365.

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 J2C

El sistema construye especificaciones de aplicación J2C 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.

Por qué y cuándo se efectúa esta tarea

Suprima especificaciones de activación de J2C 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 J2C 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 SIBus son las conexiones que ponen los servicios a disposición de las aplicaciones. Hay ocasiones en que tendrá que eliminar destinos.

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.

Por qué y cuándo se efectúa esta tarea

Suprima destinos de SIBus cuando guarde una configuración sin darse cuenta después de instalar una aplicación que contenga servicios o ya no necesite los destinos.

Nota: Esta tarea sólo suprime el destino del bus de 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 J2C en la sección Administración de este centro de información).

Procedimiento

1. Inicie la sesión en la consola administrativa.
2. Visualice los destinos en el bus del sistema SCA.

Desplácese hasta el panel pulsando **Integración de servicios > Buses**

3. Seleccione los destinos de bus del sistema SCA.

   En la pantalla, pulse en **SCA.SYSTEM.*nombrecélula*.Bus**, donde *nombrecélula* es el nombre de la célula que contiene el módulo con los destinos que va a suprimir.

4. Suprima los destinos que contienen un nombre de módulo que coincida con el módulo que va a eliminar.

   a. Pulse en el recuadro de selección situado junto a los destinos pertinentes.

   b. Pulse **Suprimir**.

Resultados

El panel sólo muestra los destinos restantes.

Qué hacer a continuación

Suprima las especificaciones de activación J2C relacionadas con el módulo que ha creado estos destinos.

Parte 3. Apéndices

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en Estados Unidos.

Es posible que en otros países IBM no ofrezca los productos, los servicios o las características que se describen en este documento. Consulte al representante de IBM de su localidad para obtener información acerca de los productos y servicios que están actualmente disponibles en su localidad. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar o implicar que sólo se pueda utilizar dicho producto, programa o servicio de IBM. En su lugar, se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM

IBM puede tener patentes o solicitudes de patentes pendientes que cubran el tema principal que se describe en este documento. El suministro de este documento no le otorga ninguna licencia sobre estas patentes. Puede enviar preguntas acerca de licencias por escrito a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
EE.UU.

Para realizar consultas sobre licencias relativas a la información del juego de caracteres de doble byte (DBCS), póngase en contacto con el Departamento de propiedad intelectual de IBM de su país o envíe sus consultas, por escrito, a:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japón

El párrafo siguiente no se aplica al Reino Unido ni a ningún otro país donde tales disposiciones estén en contradicción con la legislación local:INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL" SIN GARANTÍA DE NINGUNA CLASE, EXPRESA O IMPLÍCITA, INCLUYENDO, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO INFRACCIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunas legislaciones no contemplan la exclusión de garantías, explícitas o implícitas en algunas transacciones, por lo que puede haber usuarios a los que no les afecte dicha declaración.

Esta publicación puede contener imprecisiones técnicas o errores tipográficos. La información que ofrece está sometida a modificaciones periódicas, las cuales se van incorporando en ediciones posteriores. IBM puede reservarse el derecho de realizar mejoras y/o cambios en los productos y/o programas descritos en esta publicación en cualquier momento sin previo aviso.

Cualquier referencia en esta información a sitios Web que no son de IBM se proporciona solamente para su comodidad y no equivale de ninguna manera a una aprobación de esos sitios Web. Los materiales de esos sitios Web no forman parte de los materiales de este producto de IBM y la utilización de esos sitios Web se realiza bajo el propio riesgo del usuario.

IBM puede utilizar o distribuir la información que se le proporcione del modo que estime apropiado sin incurrir por ello en ninguna obligación con el remitente.

Los propietarios de licencia de este programa que deseen tener información sobre el mismo con el fin de poder: (i) intercambiar información entre programas creados de forma independiente y otros programas (incluido éste) y (ii) utilizar de forma mutua la información que se ha intercambiado, deberán ponerse en contacto con:

IBM Corporation
1001 Hillsdale Blvd., Suite 400
Foster City, CA 94404
EE.UU.

Esta información puede estar disponible, bajo las condiciones y los términos adecuados, incluyendo en algunos casos, el pago de una cuota.

IBM proporciona el programa bajo licencia descrito en este documento y todo el material con licencia disponible para el mismo bajo los términos del Acuerdo de cliente de IBM, el Acuerdo de licencia de programa internacional de IBM o cualquier acuerdo equivalente entre las dos partes.

Cualquier información de rendimiento contenida aquí fue determinada en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Pueden haberse realizado algunas mediciones en sistemas en nivel de desarrollo y no existen garantías de que estas mediciones sean las mismas en sistemas disponibles para todos los usuarios. Además, algunas mediciones pueden haberse calculado mediante extrapolaciones. Los resultados reales pueden variar. Los usuarios de este documento deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los suministradores de estos productos, sus anuncios publicados u otras fuentes disponibles para el público. IBM no ha probado esos productos y no puede confirmar la precisión del rendimiento, la compatibilidad ni ninguna otra afirmación relacionada con los productos no IBM. Las preguntas acerca de las posibilidades de productos que no son de IBM deben dirigirse a los suministradores de estos productos.

Todas las declaraciones referentes a acciones e intenciones futuras de IBM pueden cambiar o ser retiradas sin previo aviso y solamente representan objetivos.

Esta información contiene ejemplos de datos e informes utilizados en operaciones cotidianas de negocios. Para ilustrarlos de la manera más completa posible, los ejemplos incluyen nombres de personas, compañías, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones utilizadas por una empresa de negocios real es mera coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente, que ilustran cómo se realiza la programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de cualquier modo sin realizar ningún pago a IBM, con el fin de desarrollar, utilizar, comercializar o distribuir programas de aplicación que se ajusten a la interfaz de programación de aplicaciones para la plataforma operativa para la que se han escrito los programas de ejemplo. Estos ejemplos no se han probado a fondo en todas las condiciones. Por consiguiente, IBM no puede garantizar ni implicar la fiabilidad, la capacidad de servicio o el funcionamiento de estos programas.

Todas las copias o fragmentos de estos programas de ejemplo o cualquier trabajo derivado, deben incluir un aviso de copyright como se muestra a continuación: (c) (nombre de la empresa) (año). Partes de este código se derivan de los programas de ejemplo de IBM Corp. (c) Copyright IBM Corp. _entre el año o los años_. Reservados todos los derechos.

Si ve esta información en copia software, es posible que no aparezcan las fotografías y las ilustraciones en color.

Información de interfaz de programación

La información de interfaz de programación, si se proporciona, está pensada para ayudarle a crear software de aplicación utilizando este programa.

Las interfaces de programación de uso general permiten escribir software de aplicación que obtienen los servicios de las herramientas de este programa.

Sin embargo, este información puede contener también información de diagnosis, modificación y ajuste. La información de diagnosis, modificación y ajuste se proporciona para ayudarle a depurar el software de aplicación.

Aviso: no utilice esta información de diagnosis, modificación y ajuste como interfaz de programación porque está sujeta a cambios.

Marcas registradas y marcas de servicio

IBM, el logotipo de IBM e ibm.com son marcas registradas de International Business Machines Corporation en Estados Unidos y/o en otros países. Si estos términos de IBM u otros términos de marca registrada aparecen por primera vez en esta información con un símbolo de marca registrada (^{R} o ^{TM}), significa que son marcas registradas de EE.UU propiedad de IBM en el momento en que se publicó esta información. Dichas marcas registradas también pueden ser marcas registradas o marcas registradas de derecho común en otros países. Se dispone de una lista de marcas registradas de IBM en el apartado"Copyright and trademark information" del sitio Web: www.ibm.com/legal/copytrade.shtml.

Java es una marca registrada de Microsystems, Inc. en Estados Unidos y/o en otros países.

Otros nombres de compañías, productos o servicios pueden ser marcas registradas o de servicio de terceros.

Este producto incluye software desarrollado por Eclipse Project (http://www.eclipse.org).