IBM WebSphere Commerce

Guía del programador

Versión 54

GC10-3831-00

IBM WebSphere Commerce

Guía del programador

Versión 54

GC10-3831-00

Nota:
Antes de utilizar esta información y el producto al que da soporte, lea la información del apartado Avisos.

Primera edición (marzo de 2002), release 2

Esta edición se aplica a los siguientes productos:

- IBM WebSphere Commerce Business Edition para Windows NT y Windows 2000, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Business Edition para AIX, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Business Edition para el software Solaris Operating Environment, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Business Edition para Linux, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Studio, Business Developer Edition para Windows NT y Windows 2000, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Professional Edition para Windows NT y Windows 2000, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Professional Edition para AIX, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Professional Edition para el software Solaris Operating Environment, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Professional Edition para Linux, Versión 5.4 (Programa 5724-A18)
- IBM WebSphere Commerce Studio, Professional Developer Edition para Windows NT y Windows 2000, Versión 5.4 (Programa 5724-A18)

y a todos los releases y modificaciones posteriores de los productos antes citados hasta que se indique lo contrario en nuevas ediciones. Asegúrese de que utiliza la edición correcta para el nivel del producto.

Efectúe el pedido de publicaciones a través del representante de IBM o de la sucursal de IBM que atiende a su localidad. En la dirección que figura a continuación no hay existencias de publicaciones.

IBM agradece sus comentarios. Puede enviar sus comentarios sobre esta publicación mediante uno de los métodos siguientes:

1. Por correo electrónico, a la siguiente dirección:

   `hojacom@vnet.ibm.com`

2. Por correo a la siguiente dirección:

   IBM S.A.
   National Language Solutions Center
   Av. Diagonal 571, Edif. L'illa
   08029 Barcelona
   España

Cuando se envía información a IBM, se otorga a IBM un derecho no exclusivo para utilizar o distribuir la información de la forma que considere apropiada, sin incurrir por ello en ninguna obligación para con el remitente.

Antes de empezar

La publicación *IBM WebSphere Commerce, Guía del programador* proporciona información sobre la arquitectura y el modelo de programación de WebSphere Commerce. En concreto, proporciona detalles sobre los siguientes temas:

- Interacciones entre componentes
- Patrones de diseño
- Modelo de objeto persistente
- Control de acceso
- Manejo de errores y mensajes
- Implementación de mandatos
- Herramientas de desarrollo
- Despliegue de código personalizado

Además, este manual incluye las siguientes guías de aprendizaje:

Creación de nueva lógica de negocio
Esta guía de aprendizaje muestra cómo crear mandatos nuevos, beans de datos y beans enterprise siguiendo el modelo de programación de WebSphere Commerce. También muestra cómo integrar la lógica en una tienda existente y difundir el código a un WebSphere Commerce Server de destino.

Modificación y ampliación de la lógica de negocio existente
Esta guía de aprendizaje consta de dos secciones. La primera muestra cómo añadir nueva lógica a un mandato de controlador existente. La segunda muestra cómo modificar un mandato de tarea existente y un bean de entidad de WebSphere Commerce. También muestra cómo integrar las modificaciones en una tienda existente y desplegar el código a un WebSphere Commerce Server de destino.

Convenios utilizados en esta publicación

En esta publicación se utilizan los siguientes convenios para resaltar el texto:

La **negrita** indica mandatos o controles de la interfaz gráfica de usuario (GUI) tales como nombres de campos, botones o elecciones de menús.

El `monoespaciado` indica ejemplos de texto que deben escribirse exactamente tal como se muestran, así como vías de acceso a directorios.

La *cursiva* se utiliza para enfatizar palabras y para indicar variables que el usuario debe sustituir por sus propios valores.

Este icono indica una sugerencia: información adicional que puede ayudarle a realizar una tarea.

Windows indica información específica de WebSphere Commerce para Windows NT y Windows 2000.

AIX indica información específica de WebSphere Commerce para AIX.

Solaris indica información específica de WebSphere Commerce para el software Solaris™ Operating Environment.

400 indica información específica de WebSphere Commerce para IBM @server iSeries 400 (anteriormente denominado AS/400)

DB2 indica información específica de DB2 Universal Database

Oracle indica información específica de Oracle®.

Professional indica información específica de WebSphere Commerce Professional Edition.

Business indica información específica de WebSphere Commerce Business Edition.

Conocimientos necesarios

Esta publicación va dirigida a los desarrolladores de tienda que necesitan saber cómo personalizar una aplicación de WebSphere Commerce. Los desarrolladores de tienda que llevan a cabo ampliaciones por programa deben tener conocimientos en las siguientes áreas:

- Java
- Arquitectura de componentes Enterprise JavaBeans (EJB)
- Tecnología JavaServer Pages
- HTML
- Tecnología de base de datos
- VisualAge para Java, Enterprise Edition, Versión 3.5

Dónde encontrar más información

Es posible que esta publicación se actualice más adelante. Encontrará las actualizaciones en los siguientes sitios Web de WebSphere Commerce:

Business

`http://www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html`

Professional

`http://www.ibm.com/software/webservers/commerce/wc_pe/lit-tech-general.html`

Las actualizaciones pueden incluir información nueva, guías de aprendizaje adicionales o actualizadas, y código de ejemplo relacionado con esas guías de aprendizaje.

Contenido

Antes de empezar iii
Convenios utilizados en esta publicación. iii
Conocimientos necesarios iv
Dónde encontrar más información iv

Parte 1. Conceptos y arquitectura . . 1

Capítulo 1. Visión general 3
Componentes de software de WebSphere Commerce 3
Arquitectura de la aplicación WebSphere Commerce 4
Arquitectura de ejecución de WebSphere Commerce 6
Motor de servlets 7
Gestor de adaptadores 8
Escuchas de protocolo 8
Adaptadores 8
Controlador Web 10
Mandatos 10
Beans de entidad de WebSphere Commerce. . . 11
Beans de datos 12
Gestor de beans de datos 12
Plantillas JavaServer Pages 12
Archivo de configuración *nombre_instancia*.xml 12
Resumen de una petición 12
Diferencias clave en la personalización respecto a los releases anteriores 14

Parte 2. Modelo de programación 17

Capítulo 2. Patrones de diseño 19
Patrón de diseño de modelo-vista-controlador . . . 19
Patrón de diseño de mandatos 20
Infraestructura de mandatos 21
Fábrica de mandatos 23
Flujo de mandatos 23
Estructura del registro de mandatos 25
Patrón de diseño de visualización 33
Plantillas JSP y beans de datos 33
Tipos de beans de datos 34
Llamada a mandatos de controlador desde una plantilla JSP 37
Recuperación de datos de recopilación diferida 37
Establecimiento de atributos JSP - Visión general . . 38
Valores de propiedades necesarios 40

Capítulo 3. Modelo de objeto persistente 41
Implementación de los beans de entidad de WebSphere Commerce 41
Beans de entidad de WebSphere Commerce - Visión general 41
Descriptores de despliegue para beans enterprise de WebSphere Commerce. 42
Ampliación del modelo de objeto de WebSphere Commerce 44
Ciclos de vida de los objetos. 64
Transacciones 65
Otras consideraciones sobre los beans de entidad 65
Utilización de los beans de entidad 68
Consideraciones sobre la base de datos 69
Consideraciones sobre los nombres de los objetos del esquema de base de datos 69
Consideraciones sobre el tipo de datos de las columnas de la base de datos 71
Diferencias de los tipos de datos entre bases de datos 72

Capítulo 4. Control de acceso 75
Información sobre el control de acceso 75
Visión general de la protección de recursos en WebSphere Application Server 75
Introducción a las políticas de control de acceso de WebSphere Commerce. 77
Tipos de control de acceso 84
Interacciones del control de acceso. 86
Interfaz protegible 88
Interfaz de agrupación. 88
Información adicional sobre el control de acceso 88
Implementación del control de acceso 89
Identificación de recursos protegibles 89
Implementación del control de acceso en beans enterprise 89
Implementación del control de acceso en beans de datos 91
Implementación del control de acceso en mandatos de controlador 92
Implementación de políticas de control de acceso en vistas 94

Capítulo 5. Manejo de errores y mensajes. 95
Manejo de errores de mandatos. 95
Tipos de excepciones 95
Archivos de propiedades de mensajes de error . 96
Flujo del manejo de excepciones 96
Manejo de excepciones en el código personalizado 97
Creación de mensajes 99
Rastreo del flujo de ejecución 102
Manejo de errores de las plantillas JSP 103

Capítulo 6. Implementación de mandatos 105
Nuevos mandatos - Introducción 105
Creación de paquetes de código personalizado . . 107
Contexto de mandatos 108
Nuevos mandatos de controlador. 109
Método isGeneric 109

Método isRetriable 110
Método setRequestProperties 110
Método validateParameters 111
Método getResources 111
Método performExecute 111
Mandatos de controlador de larga ejecución . . 112
Formato de las propiedades de entrada para ver mandatos 112
Reducción de parámetros de entrada a una serie de consulta para HttpRedirectView 113
Manejo de un URL de redirección de longitud limitada 113
Establecimiento de atributos en el objeto HttpServletRequest para HttpForwardView . . 114
Compromisos y restituciones de la base de datos para mandatos de controlador 115
Ejemplo de ámbito de transacción con un mandato de controlador 116
Nuevos mandatos de tarea 117
Personalización de mandatos existentes. 118
Personalización de mandatos de controlador existentes 118
Personalización de los mandatos de tarea existentes. 122
Personalización de los beans de datos 123

Capítulo 7. Acuerdos comerciales y políticas de negocio (Business Edition) 125
Introducción. 125
Mandatos y objetos de política de negocio. . . . 126
Datos del contrato de la tienda de ejemplo ToolTech 127
Datos de ejemplo de la tabla CONTRACT. . . 128
Datos de ejemplo de la tabla TERMCOND . . 128
Datos de ejemplo de la tabla POLICYTC . . . 128
Datos de ejemplo de la tabla POLICY 129
Datos de ejemplo de la tabla TRADEPOSCN 129
Datos de ejemplo de la tabla SHIPMODE . . . 129
Ampliación del modelo de contrato existente. . . 129
Creación de una nueva política de negocio . . . 130
Creación de un nuevo tipo de política de negocio 130
Creación de un nuevo mandato de política de negocio 132
Registro de la nueva política de negocio y el nuevo mandato de política de negocio 134
Cómo relacionar un objeto de términos y condiciones con una nueva política de negocio . . 134
Creación de nuevos términos y condiciones . . 135
Invocación de la nueva política de negocio . . . 146
Creación de un contrato 147
Escenarios de personalización de contrato 147
Escenario de rebaja 147

Parte 3. Entorno de desarrollo. . . 155

Capítulo 8. Herramientas de desarrollo y despliegue. 157
Entorno de desarrollo 157
WebSphere Commerce Studio 157
Características y funciones de VisualAge para Java 158
Depósito de código de WebSphere Commerce . . 158
Despliegue del código 159
Información sobre el código desplegado de EJB 159
Despliegue de mandatos y beans de datos nuevos 160
Despliegue de beans de entidad nuevos . . . 161
Despliegue de las ampliaciones para mandatos y beans de datos existentes. 163
Despliegue de beans de entidad públicos de WebSphere Commerce modificados 163
Despliegue de beans de datos nuevos para utilizarlos en Commerce Studio 165
Despliegue de beans de entidad públicos personalizados para utilizarlos en Commerce Studio. 165
Archivo de anotaciones cronológicas 166
Método de pago de prueba. 166
Utilización de un Payment Manager remoto . . 167

Parte 4. Guías de aprendizaje . . . 169

Capítulo 9. Guía de aprendizaje: Creación de nueva lógica de negocio . 171
Entorno para la guía de aprendizaje. 171
Pasos para el despliegue de código 171
Preparación del proyecto de ejemplo 172
Creación de mandatos 173
Creación de un mandato de controlador . . . 173
Modificación de MyNewControllerCmd . . . 178
Creación de un bean de entidad nuevo. 198
Creación de la nueva tabla de base de datos . . 199
Creación del bean de entidad BonusBean . . . 199
(Opcional) Utilización del depurador en VisualAge para Java. 218
Adición del punto de interrupción al código . . 218
Verificación de los valores de las variables. . . 219
Supresión del punto de interrupción. 220
Integración de MyNewControllerCmd con la tienda de ejemplo en el Entorno de prueba WebSphere. . 220
(Opcional) Despliegue de nueva lógica de negocio en un WebSphere Commerce Server remoto . . . 221
Creación del archivo JAR para la nueva lógica de mandato 221
Creación del archivo JAR para el nuevo grupo EJB. 222
Creación del archivo JAR de implementación para el nuevo bean enterprise 223
Copia de los archivos JSP en el WebSphere Commerce Server de destino 224
Copia de los archivos JAR en el WebSphere Commerce Server de destino 224
Ejecución de la herramienta de despliegue de EJB. 225
Modificación del nivel de aislamiento de transacción para los bean de bonificación . . . 226
Actualización de la base de datos de destino 227

Carga de las políticas de control de acceso para los nuevos recursos 228
Exportación de la aplicación de empresa actual de WebSphere Application Server 230
Exportación de la información de configuración XML para la aplicación de empresa 231
Integración del nuevo grupo EJB en la aplicación de empresa 233
Importación de la nueva aplicación de empresa a WebSphere Application Server 235
Prueba de MyNewControllerCmd 236

Capítulo 10. Modificación y ampliación de la lógica de negocio existente 239
Ampliación de un mandato de controlador existente 239
Creación del paquete nuevo para `OrderProcessCmdBonusImpl` 239
Creación de la clase `OrderProcessCmdBonusImpl` 240
Adición de campos y métodos a `OrderProcessCmdBonusImpl` 240
Modificación del registro de mandatos para utilizar `OrderProcessCmdBonusImpl` 243
Modificación de la plantilla `confirmation.jsp` 244
Comprobación de `OrderProcessCmdBonusImpl` en el Entorno de prueba WebSphere 245
(Opcional) Despliegue de la lógica de negocio personalizada en un WebSphere Commerce Server remoto 245
Modificación de un bean de entidad existente y ampliación de un mandato de tarea existente . . . 249
Adición de un nuevo campo bonusPoint al bean de entidad User 251
Creación e inserción de datos en la tabla BONUS 252
Actualización del esquema y correlación de tablas 254
Generación del código desplegado y el bean de acceso 256
Comprobación de la modificación utilizando el cliente de prueba 257
Creación de la interfaz `GetNewProductContractUnitPriceCmd` 257
Creación de la clase de implementación `GetNewContractUnitPriceCmdImpl` 259
Creación del bean de datos `NewProductDataBean` 262
Adición del nuevo precio después de bonificación a la plantilla de visualización de producto 263
Comprobación de la ampliación del bean enterprise 264
(Opcional) Despliegue de la lógica de negocio personalizada en un WebSphere Commerce Server remoto 265

Parte 5. Apéndices 279

Apéndice A. Inicio y detención del Entorno de prueba WebSphere 281
Inicio y detención del servidor de nombres persistentes 281
Inicio y detención del servidor EJB 281
Inicio y detención del motor de servlets 282

Apéndice B. Información detallada sobre el despliegue 283
Correlación con el sistema de archivos integrado (iSeries) 283
Archivos JAR para mandatos y beans de datos personalizados 283
Creación de archivos JAR para beans de entidad nuevos 284
Creación del archivo JAR de exportación de EJB 1.1 285
Creación del archivo JAR de implementación 285
Creación de archivos JAR para beans de entidad de WebSphere Commerce personalizados 286
Creación del archivo JAR de exportación de EJB 1.1 287
Creación del archivo JAR de cliente 288
Almacenamiento de elementos en el WebSphere Commerce Server de destino 288
Actualización de la base de datos de destino . . . 291
Generación de código desplegado 292
Modificación del nivel de aislamiento de transacción de los beans de entidad 294
Exportación de la aplicación de empresa actual de WebSphere Commerce 295
Exportación de la información de configuración para los beans enterprise 296
Integración de nuevos beans enterprise en una aplicación de empresa 301
Integración de beans enterprise modificados en una aplicación de empresa 304
Detención y supresión de una aplicación de empresa 308
Importación de una aplicación de empresa . . . 308
Inicio de una aplicación de empresa 310

Apéndice C. Consejos para VisualAge para Java 311
Cambio de las propiedades del motor de servlets en el Entorno de prueba WebSphere 311
Resolución de problemas del servidor de nombres persistentes 312
Supresión de archivos JSP compilados 312

Avisos 313
Marcas registradas y marcas de servicio 315

Índice 317

Parte 1. Conceptos y arquitectura

Capítulo 1. Visión general

Componentes de software de WebSphere Commerce

Antes de examinar cómo funciona WebSphere Commerce Server, consulte la ilustración de los componentes de software relacionados con el proceso de personalización. El diagrama siguiente muestra una vista simplificada de estos productos de software:

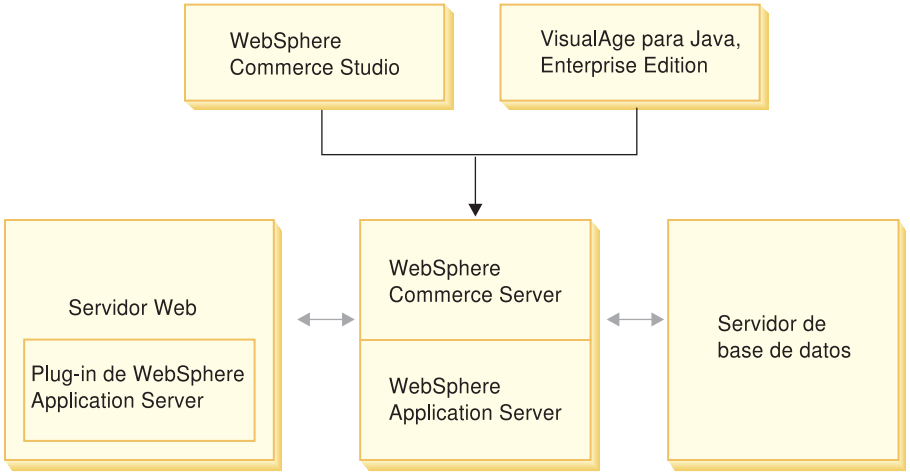

Figura 1.

El servidor Web es el primer punto de contacto de las peticiones HTTP de entrada para la aplicación de comercio electrónico. Para poder intercambiar información de forma eficaz con WebSphere Application Server, utiliza el plug-in de WebSphere Application Server.

WebSphere Commerce Server se ejecuta dentro de WebSphere Application Server, lo que le permite beneficiarse de muchas de las funciones del servidor de aplicaciones. El servidor de base de datos contiene la mayor parte de los datos de la aplicación, incluyendo los datos de productos y compradores. En general, las ampliaciones a la aplicación se llevan a cabo modificando o ampliando el código para WebSphere Commerce Server. Además, es posible que necesite almacenar datos en su base de datos que se encuentran fuera del dominio del esquema de base de datos de WebSphere Commerce.

Los desarrolladores utilizan dos herramientas principales para crear lógica de negocio personalizada: WebSphere Commerce Studio y VisualAge para Java, Enterprise Edition. WebSphere Commerce Studio se utiliza para crear y gestionar elementos de escaparate de la tienda (por ejemplo, plantillas JSP). VisualAge para Java, Enterprise Edition se utiliza para crear nueva lógica de negocio, en Java, que amplía las funciones ya existentes o crea funciones completamente nuevas. Si su aplicación requiere que se amplíe el esquema de la base de datos, los desarrolladores de la base de datos deberán utilizar sus herramientas de desarrollo de base de datos para crear las tablas nuevas.

Arquitectura de la aplicación WebSphere Commerce

Ahora que ha visto cómo se ajustan entre sí los diferentes componentes de software relacionados con la personalización, es importante conocer la arquitectura de la aplicación. Esto le ayudará a comprender qué partes forman parte de la infraestructura y qué partes puede modificar. El diagrama siguiente muestra las diversas capas que componen la arquitectura de la aplicación:

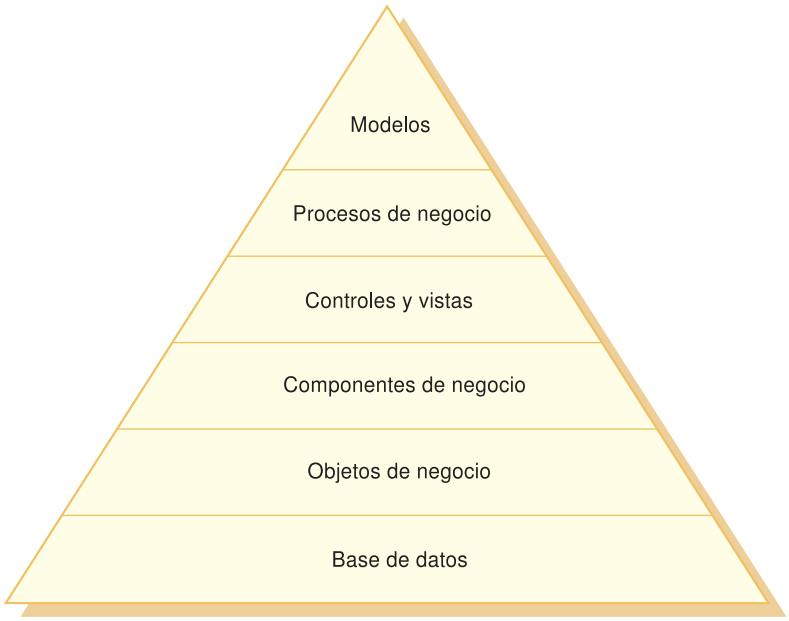

Figura 2.

A continuación se describe cada capa de la arquitectura de la aplicación:

Base de datos

WebSphere Commerce utiliza un esquema de base de datos diseñado específicamente para aplicaciones de comercio electrónico y sus necesidades de datos. A continuación, se muestran ejemplos de tablas de este esquema:

- Usuario
- Pedido
- Producto

Objetos de negocio

Los objetos de negocio representan entidades dentro del dominio de comercio y encapsulan la lógica central de datos necesaria para extraer o interpretar la información contenida en la base de datos. Estas entidades cumplen la especificación EJB (Enterprise JavaBeans).

Estos beans de entidad actúan como una interfaz entre la aplicación de comercio y la base de datos. Además, los beans de entidad son más fáciles de comprender que las complejas relaciones entre las columnas de las tablas de base de datos.

Componentes de negocio

Los componentes de negocio son unidades de la lógica de negocio. Se encargan de ejecutar los procedimientos más básicos de la lógica de negocio. La lógica se implementa utilizando el modelo de mandatos de

controlador y mandatos de tarea de WebSphere Commerce. Un ejemplo de este tipo de componente es el mandato de controlador OrderProcess. Este mandato concreto encapsula toda la lógica de negocio necesaria para procesar un pedido típico. La aplicación de comercio electrónico llama al mandato OrderProcess que, a su vez, llama a varios mandatos de tarea para ejecutar unidades de trabajo individuales. Por ejemplo, los mandatos de tarea individuales se aseguran de que haya suficientes existencias disponibles para satisfacer los requisitos del pedido, procesan el pago, actualizan el estado del pedido y, cuando el proceso se ha completado, reducen las existencias en la cantidad adecuada.

Controles y vistas

Un controlador Web determina la implementación correcta del mandato de controlador y la vista que debe utilizarse. Las implementaciones pueden ser específicas a la tienda.

Las vistas muestran el resultado de los mandatos y las acciones del usuario. Éstas se implementan utilizando plantillas JSP. Los ejemplos de vistas incluyen ProductDisplay (que devuelve una página de producto con la información relativa al producto que el comprador ha seleccionado) y OrderPrepare (que presenta al comprador un formulario para someter la información de pedido correspondiente).

Procesos de negocio

El uso conjunto de componentes de negocio y vistas crean los procesos de flujo de trabajo y de flujo del sitio Web conocidos como procesos de negocio. Los ejemplos de procesos de negocio incluyen:

Registro de usuarios

Este proceso de negocio incluye los componentes de negocio (por ejemplo, el mandato UserRegistrationAdd que crea una entrada de registro para un usuario nuevo) y las vistas relacionadas con todos los pasos implicados en el proceso de registro de usuarios.

Navegación por el catálogo

Este proceso de negocio incluye los componentes de negocio (por ejemplo, los mandatos StoreCatalogDisplay y CategoryDisplay que muestran, respectivamente, los catálogos para una tienda y las categorías dentro del catálogo) y las vistas relacionadas con todos los pasos que afectan al proceso de navegar por un catálogo.

Modelos

El conjunto de capas inferiores del diagrama conforman los modelos de negocio de comercio electrónico. Un ejemplo de un modelo de negocio de e-commerce es el modelo de empresa a cliente que se utiliza en la tienda de ejemplo InFashion. Otro ejemplo es el modelo empresa a empresa que se utiliza en la tienda de ejemplo ToolTech.

Arquitectura de ejecución de WebSphere Commerce

La sección anterior presentaba la arquitectura de la aplicación, que muestra las diversas capas de la aplicación WebSphere Commerce, desde el punto de vista de una aplicación de negocios. Esta sección describe cómo se implementa la arquitectura de ejecución.

Los componentes principales de la arquitectura de ejecución de WebSphere Commerce son:

- Motor de servlets
- Escuchas de protocolo
- Gestor de adaptadores
- Adaptadores
- Controlador Web
- Mandatos
- Beans de entidad
- Beans de datos
- Gestor de beans de datos
- Páginas de visualización
- Archivos XML

Las interacciones entre los componentes de WebSphere Commerce se muestran en el siguiente diagrama. Puede encontrar más detalles sobre cada componente en las secciones posteriores.

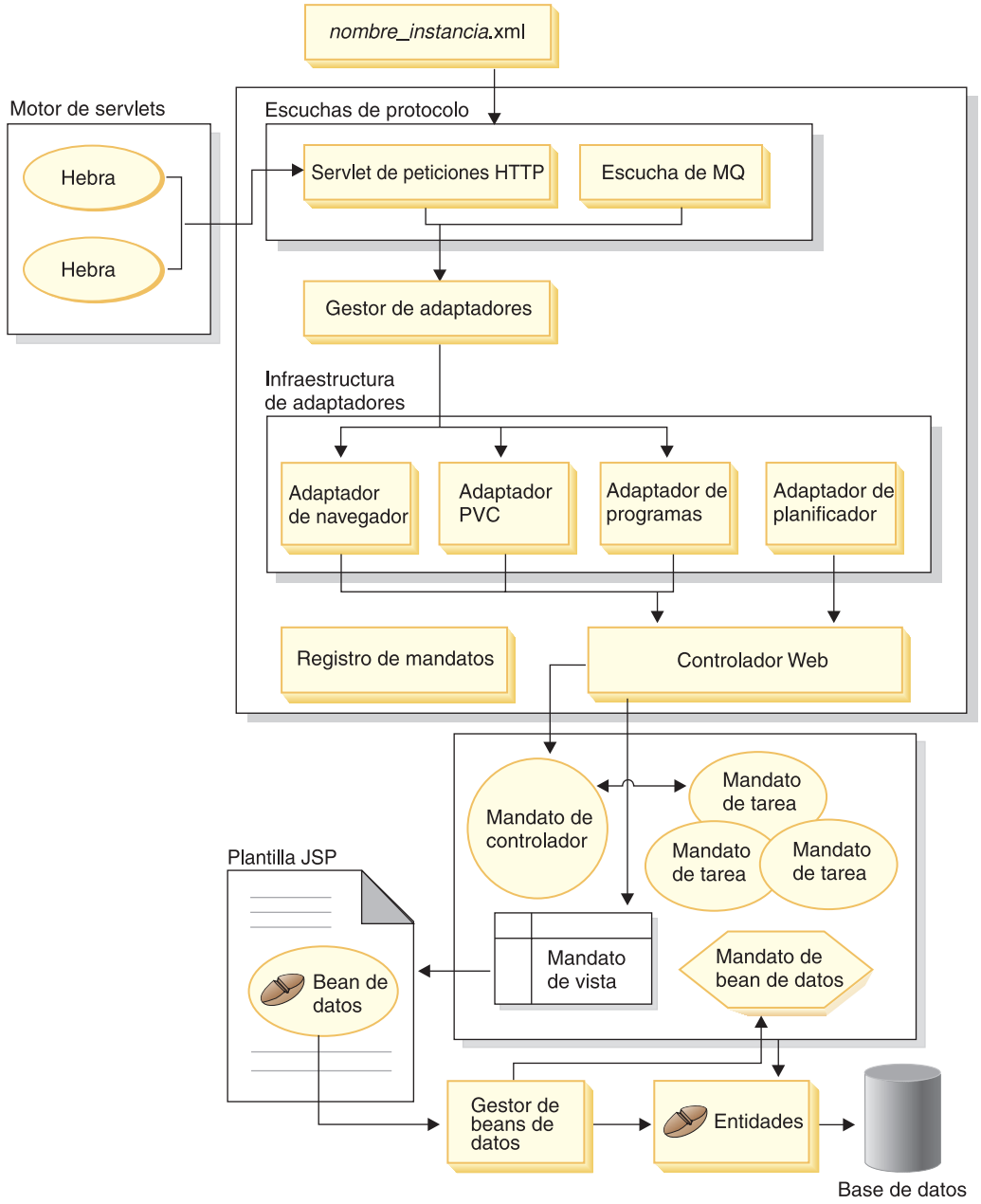

Figura 3.

Motor de servlets

El motor de servlets es la parte del entorno de ejecución de WebSphere Application Server que actúa como asignador de peticiones para las peticiones de URL de entrada. El motor de servlets gestiona una agrupación de hebras para manejar las peticiones. Cada petición de entrada se ejecuta en una hebra separada.

Las versiones anteriores de WebSphere Commerce utilizaban un servidor de aplicaciones C++ que implementaba su propio asignador de tareas para las peticiones de URL mediante la utilización de un conjunto preasignado de procesos del sistema. Este modelo anterior, que utilizaba procesos del sistema, consumía más recursos que el nuevo modelo que utiliza hebras de Java. Con la utilización

del motor de servlets, la nueva arquitectura de ejecución de WebSphere Commerce proporciona una solución de comercio más escalable.

Gestor de adaptadores

El gestor de adaptadores determina qué adaptador es capaz de manejar la petición y, a continuación, envía la petición a ese adaptador.

Escuchas de protocolo

Los mandatos de WebSphere Commerce pueden invocarse desde diversos dispositivos. Ejemplos de dispositivos que pueden invocar mandatos son:

- Navegadores estándar de Internet
- Teléfonos móviles que utilizan navegadores de Internet
- Aplicaciones de empresa a empresa que envían mensajes XML mediante MQSeries
- Sistemas de compra que envían peticiones utilizando XML sobre HTTP
- El planificador de WebSphere Commerce que ejecuta un trabajo en segundo plano

Los dispositivos pueden utilizar distintos protocolos de comunicación. Un escucha de protocolo es un componente de ejecución que recibe peticiones de entrada de los transportes y luego transmite las peticiones a los adaptadores apropiados según el protocolo utilizado. Los escuchas de protocolo son:

- Servlet de peticiones
- Escucha de MQSeries

Cuando el servlet de peticiones recibe una petición de URL del motor de servlets, pasa la petición al gestor de adaptadores. A continuación, el gestor de adaptadores consulta los tipos de adaptador para determinar qué adaptador puede procesar la petición. Una vez se determina el adaptador específico, la petición se pasa al adaptador.

Cuando el servlet de peticiones se inicializa, lee el archivo de configuración *nombre_instancia*.xml. Uno de los bloques de configuración del archivo XML define todos los adaptadores. El método init() del servlet de peticiones inicializa todos los adaptadores definidos.

El escucha de MQSeries recibe mensajes MQSeries basados en XML de los programas remotos y transmite las peticiones al gestor de adaptadores no HTTP.

El planificador de trabajos no requiere un escucha de protocolo.

Adaptadores

Los adaptadores de WebSphere Commerce son componentes específicos de cada dispositivo que realizan funciones de proceso antes de pasar una petición al controlador Web. Ejemplos de las tareas de proceso que realiza un adaptador son:

- Indicar al controlador Web que procese la petición de una manera específica para el tipo de dispositivo. Por ejemplo, un adaptador de dispositivo PVC ("pervasive computing") puede indicar al controlador Web que ignore la comprobación HTTPS en la petición original.
- Transformar el formato de mensaje de la petición de entrada en un conjunto de propiedades que los mandatos de WebSphere Commerce puedan analizar.
- Proporcionar persistencia de sesiones específica para cada dispositivo.

En el siguiente diagrama se muestra la jerarquía de clases de implementación para la infraestructura de adaptadores de WebSphere Commerce.

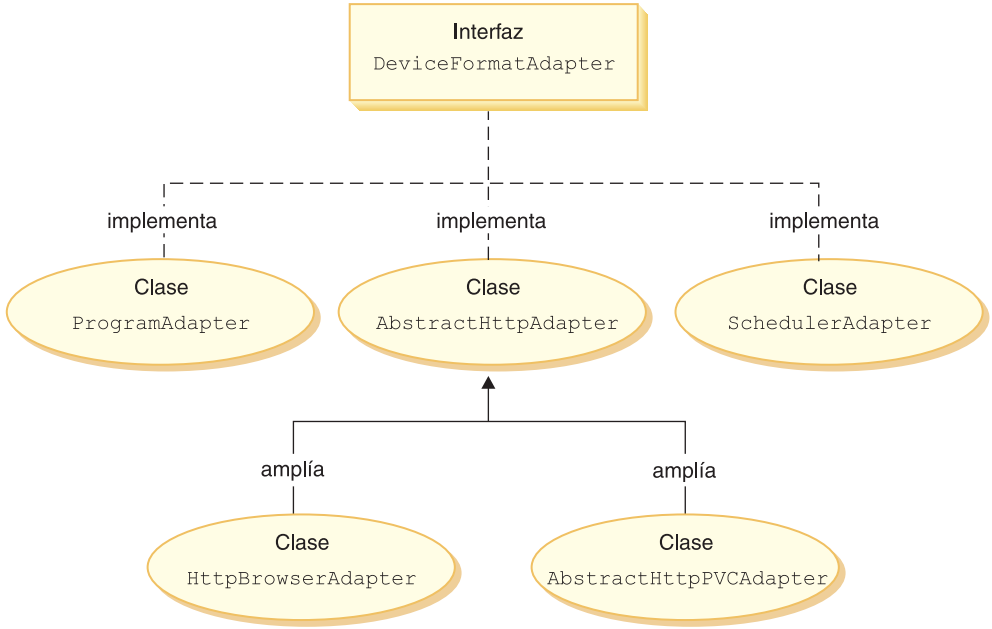

Figura 4.

Tal como se muestra en el diagrama anterior, todos los adaptadores implementan la interfaz DeviceFormatAdapter. A continuación se describen los adaptadores utilizados en el entorno de ejecución de WebSphere Commerce:

Adaptador de programas
El adaptador de programas proporciona soporte para programas remotos que invocan mandatos de WebSphere Commerce. El adaptador de programas recibe peticiones y utiliza un correlacionador de mensajes para convertir la petición en un objeto CommandProperty. Después de la conversión, el adaptador de programas utiliza el objeto CommandProperty y ejecuta la petición.

Adaptador de planificador
El adaptador de planificador proporciona soporte para los mandatos WebSphere Commerce que se ejecutan como trabajos en segundo plano.

Adaptador de navegador HTTP
El adaptador de navegador HTTP proporciona soporte para que las peticiones invoquen los mandatos WebSphere Commerce que se reciben desde los navegadores HTTP.

Adaptador PvC HTTP
Es una clase de adaptador abstracta que puede utilizarse para desarrollar adaptadores de dispositivo PvC específicos. Por ejemplo, si tiene que desarrollar un adaptador para una determinada aplicación de teléfonos móviles, puede hacerlo a partir de este adaptador.

Si es necesario, la infraestructura de adaptadores se puede ampliar de las dos maneras siguientes:

- Crear un adaptador para un dispositivo PvC específico (por ejemplo, crear HttpIModePVCAdapterImpl para dispositivos de modalidad interactiva). Un adaptador de este tipo debe ampliar la clase AbstractHttpAdapterImpl.

- Crear un nuevo adaptador que se conecte con un nuevo escucha de protocolo. Este nuevo adaptador debe implementar la interfaz DeviceFormatAdapter.

Controlador Web

El controlador Web de WebSphere Commerce es un contenedor de aplicaciones que sigue un patrón de diseño similar al de un contenedor EJB. Este contenedor simplifica el rol de los mandatos proporcionando servicios tales como gestión de sesiones (según la persistencia de sesión establecida por el adaptador), control de transacciones, control de acceso y autenticación.

El controlador Web también desempeña un papel al imponer el modelo de programación para la aplicación de comercio. Por ejemplo, el modelo de programación define los tipos de mandatos que debe escribir una aplicación. Cada tipo de mandato tiene una finalidad específica. La lógica de negocio debe implementarse en mandatos de controlador y la lógica de vista en mandatos de vista. El controlador Web espera que el mandato de controlador devuelva un nombre de vista. Si no se devuelve un nombre de vista, se genera una excepción.

Para las peticiones HTTP, el controlador Web lleva a cabo las siguientes tareas:

- Empieza la transacción utilizando la interfaz UserTransaction del paquete javax.transaction.
- Obtiene datos de sesión del adaptador.
- Determina si el usuario debe estar conectado antes de invocar el mandato. Si es necesario, redirige el navegador del usuario a un URL de conexión.
- Comprueba si es necesario HTTPS seguro para el URL. Si es necesario pero la petición actual no utiliza HTTPS, redirige el navegador Web a un URL HTTPS.
- Invoca el mandato de controlador y le pasa los objetos contexto de mandatos y propiedades de entrada.
- Si se produce una excepción de retrotracción de transacción y el mandato de controlador puede reintentarse, reintenta el mandato de controlador.
- Normalmente, un mandato de controlador devuelve un nombre de vista cuando hay un mandato de vista susceptible de ser devuelto al cliente. El controlador Web invoca el mandato de vista para la vista correspondiente. Hay diversas maneras de formar una vista de respuesta. Por ejemplo, redirigir a un URL distinto, reenviar a una plantilla JSP o escribir un documento HTML al objeto de respuesta.
- Guarda los datos de sesión.
- Compromete los datos de sesión.
- Compromete la transacción actual, si ésta es satisfactoria.
- Retrotrae la transacción actual en caso de anomalía (según las circunstancias).

Mandatos

Los mandatos de WebSphere Commerce son beans que contienen la lógica de programación asociada al manejo de una petición determinada.

Existen cuatro tipos principales de mandatos WebSphere Commerce:

Mandato de controlador

Un mandato de controlador encapsula la lógica relacionada con un proceso de negocio determinado. Ejemplos de mandatos de controlador son el mandato *OrderProcessCmd* para el proceso de pedidos y el mandato *UserRegistrationAddCmd* para la creación de nuevos usuarios registrados. En general, un mandato de controlador contiene las sentencias de control (por

ejemplo, if, then, else) e invoca mandatos de tarea para realizar tareas individuales en el proceso de negocio. Al completarse, un mandato de controlador devuelve un nombre de vista. A continuación, el controlador Web determina la clase de implementación adecuada para el mandato de vista y ejecuta el mandato de vista.

Mandato de tarea
Un mandato de tarea implementa una unidad específica de lógica de aplicación. Por lo general, un mandato de controlador y un conjunto de mandatos de tarea implementan, conjuntamente, la lógica de aplicación para una petición de URL. Un mandato de tarea se ejecuta siempre en el mismo contenedor que el mandato de controlador.

Mandato de bean de datos
Una plantilla JSP invoca un mandato de bean de datos cuando se crea una instancia de un bean de datos. La función principal de un mandato de bean de datos es insertar datos en el bean de datos.

Mandato de vista
Un mandato de vista compone una vista como respuesta a una petición del cliente. Existen tres tipos de mandatos de vista:

Mandato redirigir vista
Este mandato de vista envía la vista utilizando un protocolo de redirección como, por ejemplo, la redirección de URL. Un mandato de controlador debe devolver un mandato de vista de este tipo de vista, para que se devuelva la vista utilizando un protocolo de redirección. Cuando se utiliza un protocolo de redirección, éste cambia las pilas de URL en el navegador. Cuando se entra una clave de recarga, se ejecuta el URL redirigido en lugar del URL original.

Mandato dirigir vista
Este mandato de vista envía la vista de respuesta directamente al cliente.

Mandato reenviar vista
Este mandato de vista reenvía la petición de vista a otro componente Web, como por ejemplo una plantilla JSP.

Un mandato de vista puede invocarse de tres maneras distintas:

- Un mandato de controlador especifica el nombre de un mandato de vista cuando la petición se ha completado satisfactoriamente.
- Un cliente solicita directamente una vista.
- Un mandato detecta un error y debe ejecutarse una tarea de error para procesarlo. El mandato genera una excepción con el nombre de un mandato de vista. Cuando la excepción se propaga al controlador Web, éste ejecuta el mandato de vista de error y devuelve la respuesta al cliente.

Beans de entidad de WebSphere Commerce

Los beans de entidad son objetos de comercio transaccionales persistentes proporcionados por WebSphere Commerce. Si está familiarizado con el dominio de comercio, los beans de entidad representan los datos de WebSphere Commerce de forma intuitiva. Es decir, en lugar de tener que comprender todo el esquema de base de datos, puede acceder a los datos desde un bean de entidad que modela con más detalle conceptos y objetos del dominio del comercio. Puede ampliar los

beans de entidad existentes. Además, para satisfacer los requisitos de negocio específicos de su propia aplicación, puede desplegar beans de entidad totalmente nuevos.

Los beans de entidad se implementan según el modelo de componentes Enterprise JavaBeans (EJB).

Para obtener más información sobre beans de entidad, consulte "Implementación de los beans de entidad de WebSphere Commerce" en la página 41.

Beans de datos

Los beans de datos son beans Java utilizados principalmente por los diseñadores de Web. Generalmente proporcionan acceso a una entidad de WebSphere Commerce. Un diseñador de Web puede colocar estos beans en una plantilla JSP, lo cual permite insertar información dinámica en la página en el momento de visualizarla. El diseñador de Web sólo necesita saber qué datos puede proporcionar el bean y qué datos requiere el bean como entrada. Consecuente con la idea de separar lo que se visualiza de la lógica de negocio, no es necesario que el diseñador de Web comprenda el funcionamiento del bean.

Gestor de beans de datos

Cuando un bean de datos de WebSphere Commerce se inserta en una plantilla JSP mediante WebSphere Studio Page Designer, se genera una línea de código que inserta datos en el bean de datos, durante la ejecución, invocando el gestor de beans de datos.

A continuación se muestra un ejemplo de código de Page Designer:

```
com.ibm.commerce.beans.DataBeanManager.activate(bean_datos, request)
```

Plantillas JavaServer Pages

Las plantillas JSP son servlets especializados que se utilizan generalmente a efectos de visualización. Al finalizar una petición de URL, el controlador Web invoca un mandato de vista que invoca una plantilla JSP. Un cliente también puede invocar una plantilla JSP directamente desde el navegador sin un mandato asociado. En este caso, el URL para la plantilla JSP debe incluir el servlet de peticiones en su vía de acceso, para que todos los beans de datos que necesita una plantilla JSP puedan activarse en una sola transacción. El servlet de peticiones puede reenviar una petición de URL a una plantilla JSP y ejecutar la plantilla JSP en una sola transacción.

El gestor de beans de datos rechaza cualquier URL para una plantilla JSP que no incluya el servlet de peticiones en su vía de acceso. Para obtener más información sobre la protección de plantillas JSP y otros recursos, consulte el Capítulo 4, "Control de acceso" en la página 75.

Archivo de configuración *nombre_instancia*.xml

El archivo de configuración *nombre_instancia*.xml establece la información de configuración para la instalación. Dicho archivo se lee cuando se inicializa el servlet de peticiones.

Resumen de una petición

Esta sección proporciona un resumen del flujo interactivo entre componentes al formular una respuesta a una petición.

A continuación del diagrama se muestra una descripción de cada uno de los pasos.

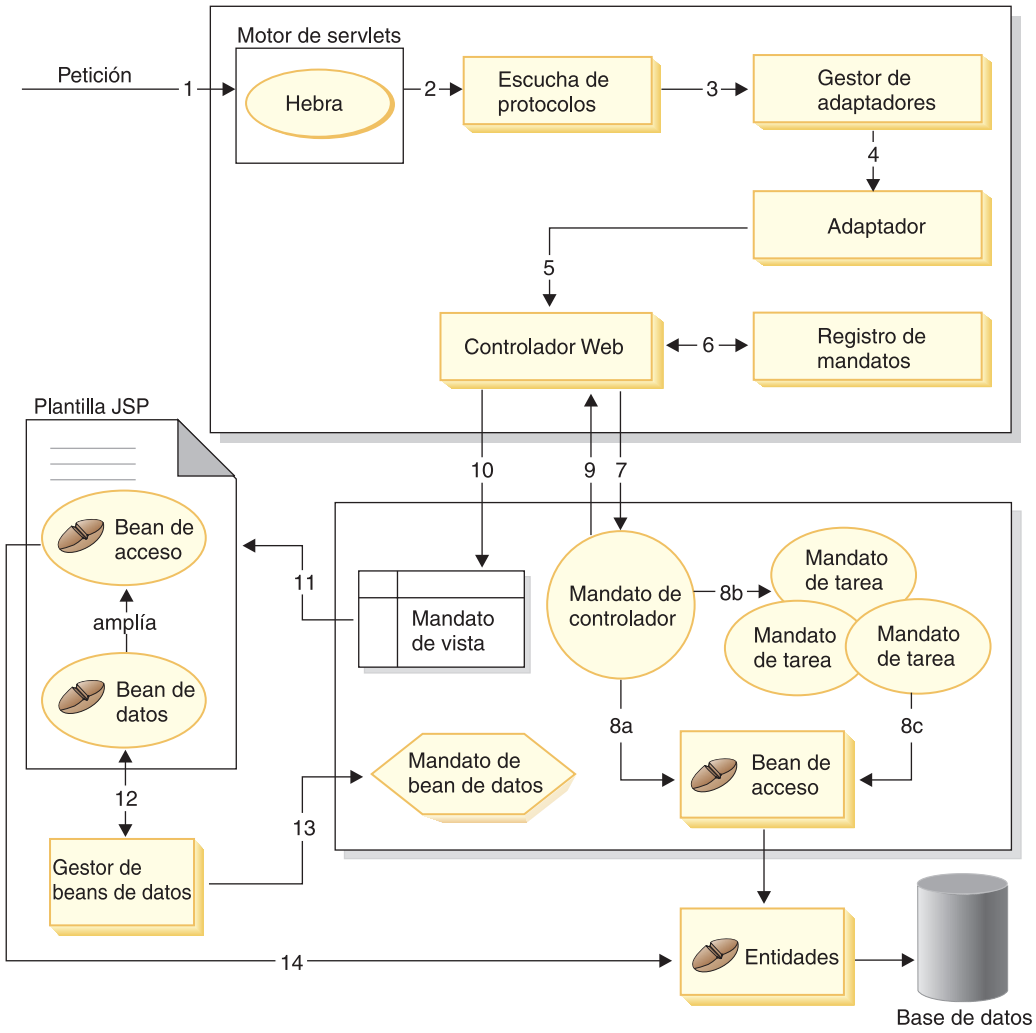

Figura 5.

La siguiente información corresponde al diagrama anterior.

1. El plug-in de WebSphere Application Server dirige la petición al motor de servlets.
2. La petición se ejecuta en su propia hebra. El motor de servlets envía la petición a un escucha de protocolo. El escucha de protocolo puede ser el servlet de peticiones HTTP o el escucha MQ.
3. El escucha de protocolo pasa la petición al gestor de adaptadores.
4. El gestor de adaptadores determina qué adaptador es capaz de manejar la petición y, a continuación, envía la petición al adaptador adecuado. Por ejemplo, si la petición viene de un navegador de Internet, el gestor de adaptadores dirige la petición al adaptador de navegadores HTTP.
5. El adaptador pasa la petición al controlador Web.
6. El controlador Web determina qué mandato invocar consultando el registro de mandatos.
7. Dando por supuesto que la petición necesita utilizar un mandato de controlador, el controlador Web invoca el mandato de controlador adecuado.

8. Una vez que un mandato de controlador empieza su ejecución, puede haber las siguientes vías posibles:
   a. El mandato de controlador puede acceder a la base de datos utilizando un bean de acceso y su bean de entidad correspondiente.
   b. El mandato de controlador puede invocar uno o más mandatos de tarea. A continuación, los mandatos de tarea pueden acceder a la base de datos utilizando beans de acceso y sus beans de entidad correspondientes (vea 8 (c)).
9. Al completarse, el mandato de controlador devuelve un nombre de vista al controlador Web.
10. El controlador Web busca el nombre de vista en la tabla VIEWREG. Invoca la implementación del mandato de vista que está registrada para el tipo de dispositivo del peticionario.
11. El mandato de vista reenvía la petición a una plantilla JSP.
12. En la plantilla JSP, es necesario un bean de datos para recuperar la información dinámica de la base de datos. El gestor de beans de datos activa el bean de datos.
13. El gestor de beans de datos invoca un mandato de bean de datos, si es necesario.
14. El bean de acceso del cual se amplía el bean de datos accede a la base de datos utilizando su bean de entidad correspondiente.

Diferencias clave en la personalización respecto a los releases anteriores

A partir de WebSphere Commerce Suite Versión 5.1, WebSphere Commerce Server se ha escrito de nuevo en su totalidad en Java. Por lo tanto, deberá utilizar Java para personalizar las funciones. Este modelo es muy distinto del utilizado en WebSphere Commerce Suite, Versión 4.1 (y las versiones anteriores de Net.Commerce) en el que se utilizaba C++ y macros Net.Data para la personalización.

En la tabla siguiente se resumen los cambios principales.

	Versión 4.1 (y anteriores) en C++	**Versión 5.1 (y posteriores) en Java**
Modelos de aplicación	Mandatos	Mandatos de controlador
	Tareas	Mandatos de tarea
	Funciones reemplazables	Mandatos de tarea
	Tareas de vista	Mandatos de vista
	Tareas de error	Mandatos de vista
Modelos de visualización	Macros Net.Data	Plantillas JSP, beans de datos, mandatos de bean de datos y mandatos de vista
Modelos de persistencia	Net.Data y ODBC	Beans de entidad
Asignador de URL	Asignador de URL C++ de WebSphere Commerce	Motor de servlets WebSphere
Modelos de tarea	Procesos del sistema	Hebras Java
Adaptador de mandatos	Ninguno	Adaptadores y controlador Web

En esta publicación no se describe el proceso de migración. Para obtener más información sobre la migración, consulte la publicación *WebSphere Commerce, Guía para la migración*.

Parte 2. Modelo de programación

Capítulo 2. Patrones de diseño

Para desarrollar la infraestructura de WebSphere Commerce se utilizan diversos patrones de diseño y mecanismos. WebSphere Commerce proporciona un patrón de diseño de alto nivel que deben satisfacer todas las aplicaciones de WebSphere Commerce. En este capítulo se tratan los siguientes patrones de diseño:

- El patrón de diseño de modelo-vista-controlador
- El patrón de diseño de mandatos
- El patrón de diseño de visualización

Patrón de diseño de modelo-vista-controlador

El patrón de diseño de modelo-vista-controlador (MVC) especifica que una aplicación está formada por un modelo de datos, información de presentación e información de control. El patrón requiere que cada uno de estos elementos esté separado en distintos objetos.

El *modelo* (por ejemplo, la información de datos) contiene únicamente los datos puros de aplicación; no contiene lógica que describe cómo pueden presentarse los datos a un usuario.

La *vista* (por ejemplo, la información de presentación) presenta al usuario los datos del modelo. La vista sabe cómo acceder a los datos del modelo, pero no sabe el significado de estos datos ni lo que el usuario puede hacer para manipularlos.

Por último, el *controlador* (por ejemplo, la información de control) está entre la vista y el modelo. Escucha los sucesos desencadenados por la vista (u otro origen externo) y ejecuta la reacción apropiada a estos sucesos. En la mayoría de los casos, la reacción es llamar a un método del modelo. Puesto que la vista y el modelo están conectados a través de un mecanismo de notificación, el resultado de esta acción se reflejará automáticamente en la vista.

La mayoría de las aplicaciones hoy en día siguen este patrón, muchas con ligeras variaciones. Por ejemplo, algunas aplicaciones combinan la vista y el controlador en una clase porque ya están estrechamente unidos. Todas las variaciones recomiendan enérgicamente la separación de los datos de su presentación. Esto no sólo simplifica la estructura de una aplicación sino que también permite reutilizar el código.

Puesto que hay muchas publicaciones que describen el patrón, así como numerosos ejemplos, este documento no describe el patrón con mucho detalle.

En el siguiente diagrama se muestra cómo se aplica a WebSphere Commerce el patrón de diseño MVC.

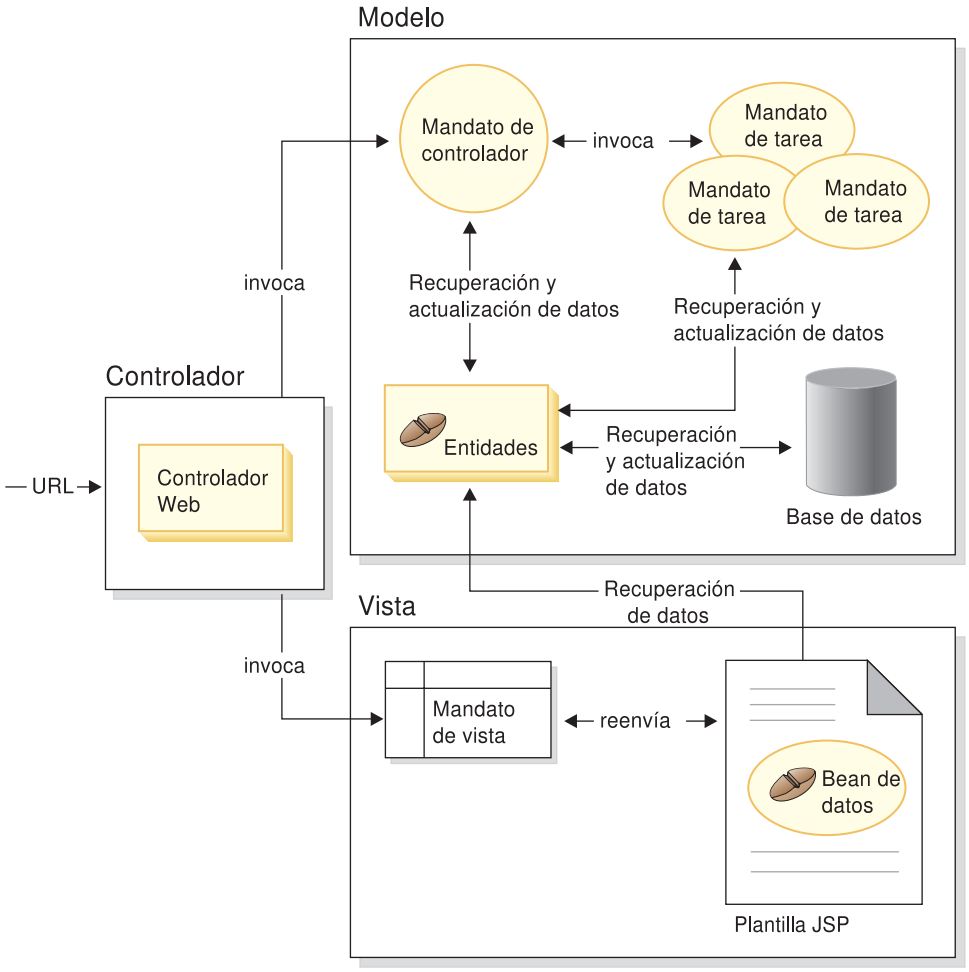

Figura 6.

Patrón de diseño de mandatos

WebSphere Commerce Server acepta solicitudes de aplicaciones de cliente reducido basadas en navegador, así como de otras aplicaciones remotas. Por ejemplo, una solicitud puede venir de un sistema de compra remota o de otro servidor de comercio.

Todas las solicitudes, en sus diversos formatos, se convierten a un formato común mediante los adaptadores que forman la estructura de adaptadores. Una vez las solicitudes están en este formato común, los mandatos de WebSphere Commerce pueden entenderlos.

Los mandatos son beans que ejecutan lógica de negocio. Representan la lógica de procedimiento en forma de lógica de proceso de alto nivel o de distintas tareas de lógica de negocio. Un mandato basado en el proceso actúa como un controlador que abarca varias entidades y otros mandatos, mientras que un mandato de tarea lleva a cabo una tarea específica y sólo puede acceder a un único objeto.

Infraestructura de mandatos

Los beans de mandatos siguen un patrón de diseño específico. Cada mandato incluye una clase de interfaz (por ejemplo, `CategoryDisplayCmd`) y una clase de implementación (por ejemplo, `CategoryDisplayCmdImpl`). Desde la perspectiva del emisor de la llamada, la lógica de invocación implica configurar propiedades de entrada, invocar un método `execute()` y recuperar propiedades de salida.

Desde la perspectiva del implementador de mandatos, los mandatos siguen la infraestructura de mandatos de WebSphere, que implementa el patrón de diseño de mandatos estándar permitiendo un nivel de direccionamiento indirecto entre el que emite la llamada y la implementación. Los mecanismos clave habilitados en este nivel de direccionamiento indirecto son:

1. La capacidad de invocar un gestor de políticas de control de acceso que determine si al usuario se le permite invocar el mandato.
2. La capacidad de ejecutar una implementación de mandato distinta para diferentes tiendas, basándose en el identificador de la tienda.
3. La capacidad de ejecutar una implementación de vista diferente basada en el tipo de dispositivo del peticionario.

En el siguiente diagrama se muestra una visión general conceptual de las interfaces correspondientes a los cuatro tipos de mandato principales:

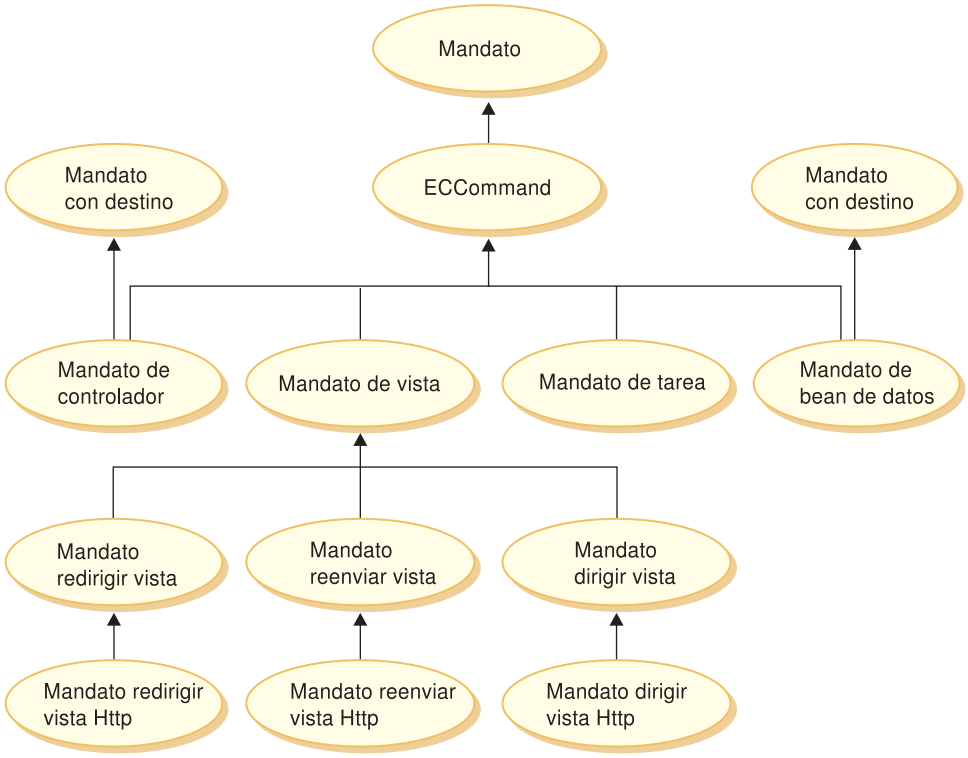

Figura 7.

Mandato de controlador

Un mandato de controlador encapsula la lógica relacionada con un proceso de negocio determinado. Ejemplos de mandatos de controlador son el mandato *OrderProcessCmd* para el proceso de pedidos y el mandato *UserRegistrationAddCmd* para la creación de nuevos usuarios registrados. En

general, un mandato de controlador contiene las sentencias de control (por ejemplo, if, then, else) e invoca mandatos de tarea para realizar tareas individuales en el proceso de negocio. Al completarse, un mandato de controlador devuelve un nombre de vista. A continuación, el controlador Web determina la clase de implementación adecuada para la tarea de vista y ejecuta la tarea de vista.

Cuando un mandato de controlador es un mandato con destino, sólo se da soporte al destino local.

Mandato de tarea

Un mandato de tarea implementa una unidad específica de lógica de aplicación. Por lo general, un mandato de controlador y un conjunto de mandatos de tarea implementan, conjuntamente, la lógica de aplicación para una petición de URL. Un mandato de tarea se ejecuta siempre en el mismo contenedor que el mandato de controlador.

Mandato de bean de datos

Una página JSP invoca un mandato de bean de datos cuando se crea una instancia de un bean de datos. La función principal de un mandato de bean de datos es insertar datos en los campos del bean de datos.

Cuando un mandato de bean de datos es un mandato con destino, sólo se da soporte al destino local.

Mandato de vista

Un mandato de vista compone una vista como respuesta a una petición del cliente. Un mandato de vista puede invocarse de tres maneras distintas:

- Un mandato de controlador especifica el nombre de un mandato de vista cuando la petición se completa satisfactoriamente.
- Un cliente puede solicitar directamente una vista.
- Un mandato de controlador o de tarea detecta un error y decide que debe ejecutarse una tarea de error para procesarlo y genera una excepción con el nombre de un mandato de vista. Cuando la excepción se propaga al controlador Web, éste ejecuta el mandato de vista y devuelve la respuesta al cliente.

Existen tres tipos de mandatos de vista:

Mandato redirigir vista

Este mandato de vista envía la vista utilizando un protocolo de redirección como, por ejemplo, la redirección de URL. Un mandato de controlador debe devolver un mandato de vista de este tipo de vista, cuando se solicite un protocolo de redirección. Cuando se utiliza un protocolo de redirección, éste cambia las pilas de URL en el navegador. Cuando se entra una clave de recarga, se ejecuta el URL redirigido en lugar del URL original.

Mandato dirigir vista

Este mandato de vista envía la vista de respuesta directamente al cliente.

Mandato reenviar vista

Este mandato de vista reenvía la petición de vista a otro componente Web, como por ejemplo una plantilla JSP.

Fábrica de mandatos

Para poder crear nuevos objetos de mandato, el llamador del mandato utiliza la *fábrica de mandatos*. La fábrica de mandatos es un bean que se utiliza para crear instancias de mandatos. Se basa en el patrón de diseño de fábrica, que traspasa la creación de la instancia de un objeto, de la clase que invoca a la clase de fábrica que sabe de qué clase de implementación ha de crearse la instancia.

La fábrica proporciona una manera inteligente de crear instancias de nuevos objetos. En este caso, la fábrica de mandatos proporciona una forma de determinar la clase de implementación correcta al crear un nuevo objeto de mandato, según la tienda de que se trate. El nombre de la interfaz de mandatos y el identificador de tienda concreto se pasan al nuevo objeto de mandato cuando se crea la instancia.

Hay dos formas de especificar la clase de implementación de un mandato. Puede especificarse directamente una clase de implementación por omisión en el código para la interfaz de mandatos, mediante la variable `defaultCommandClassName`. Por ejemplo, el código siguiente existe en la interfaz CategoryDisplayCmd:

```
String defaultCommandClassName =
     "com.ibm.commerce.catalog.commands.CategoryDisplayCmdImpl"
```

La segunda manera de especificar la clase de implementación es utilizando el registro de mandatos de WebSphere Commerce. Cuando la clase de implementación varía según la tienda, siempre debe utilizarse el registro de mandatos. En la página 25 encontrará más información sobre el registro de mandatos.

En el caso de que se especifique una clase de implementación por omisión en el código para la interfaz y otra clase de implementación en el registro de mandatos, tiene prioridad el registro de mandatos.

La sintaxis para utilizar la fábrica de mandatos es la siguiente:

```
cmd = CommandFactory.createCommand(nombreInterfaz, commandContext.getStoreId())
```

donde *nombreInterfaz* es el nombre de interfaz para el nuevo bean de mandato y el método getStoreId determina la tienda para la que se va a utilizar el mandato.

Nota: La sintaxis para utilizar la fábrica de mandatos para crear mandatos de política de negocios es distinta de la sección de código anterior. Para obtener más información sobre la utilización de la fábrica de mandatos para crear mandatos de política de negocio, consulte “Invocación de la nueva política de negocio” en la página 146.

Flujo de mandatos

En esta sección se proporciona una visión general del flujo lógico entre los mandatos y la base de datos de WebSphere Commerce. El diagrama y las descripciones siguientes muestran este flujo.

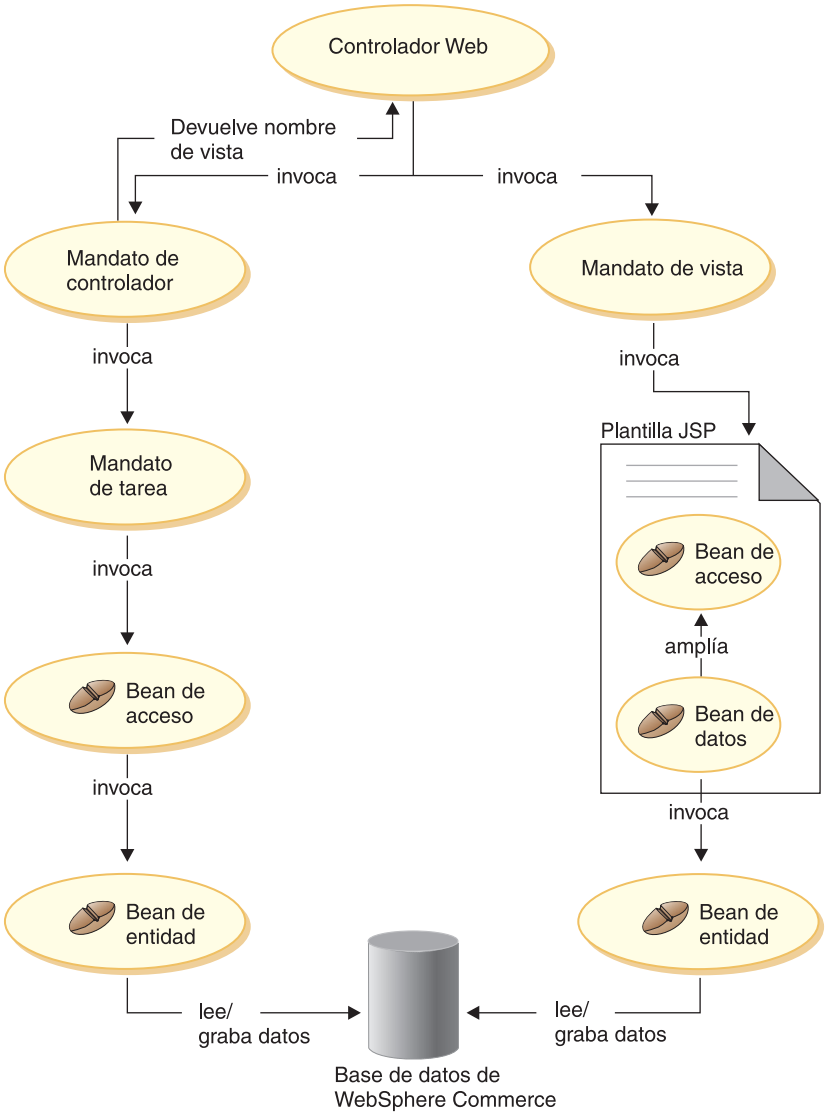

Figura 8.

Cuando el controlador Web recibe una petición, determina si la petición requiere la invocación de un mandato de controlador o un mandato de vista. En cualquiera de los dos casos, el controlador Web también determina la clase de implementación para el mandato y, a continuación, lo invoca.

Primero examine el lado izquierdo del diagrama. Puesto que los mandatos de controlador encapsulan la lógica de un proceso de negocio, suelen invocar mandatos de tarea individuales para ejecutar unidades de trabajo específicas del proceso de negocio. Los beans de acceso se invocan cuando es necesario recuperar o actualizar información de la base de datos. Tanto un mandato de tarea como un mandato de controlador pueden invocar beans de acceso. A continuación, las peticiones fluyen de los beans de acceso a los beans de entidad, que pueden leer y grabar en la base de datos de WebSphere Commerce.

Ahora examine el lado derecho del diagrama. El controlador Web invoca un mandato de vista, ya sea cuando un mandato de controlador ha terminado de procesarse y devuelve el nombre de un mandato de vista a invocar o bien cuando se produce un error y debe mostrarse una vista de error.

Los mandatos de vista normalmente invocan una plantilla JSP para mostrar la respuesta al cliente. En la plantilla JSP, los beans de datos se utilizan para insertar información dinámica en la página. El gestor de beans de datos activa los beans de datos. El bean de datos (que es una ampliación de un bean de acceso) invoca su bean de entidad correspondiente. Cuando se accede a él indirectamente desde una plantilla JSP, un bean de entidad normalmente recupera información de la base de datos (en lugar de grabar información en la misma).

Estructura del registro de mandatos

Los mandatos de tarea y de controlador WebSphere Commerce están registrados en el registro de mandatos. Las tres tablas siguientes componen el registro de mandatos:

- URLREG
- CMDREG
- VIEWREG

Nota: Business Esta sección no se aplica al registro de los mandatos de política de negocio. Para obtener información sobre el registro de nuevos mandatos de política de negocios, consulte "Registro de la nueva política de negocio y el nuevo mandato de política de negocio" en la página 134.

Tabla URLREG

La tabla URLREG correlaciona indicadores universales de recursos (URI) con interfaces de mandatos de controlador. Los URI proporcionan un mecanismo simple y ampliable para la identificación de recursos. Un URI es una serie de caracteres relativamente corta que se utiliza para identificar un recurso abstracto o físico. En WebSphere Commerce, los URI sólo contienen información de mandatos. En el siguiente URL, la sección de URI se muestra en negrita:

```
http://nombresistpral/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
```

Aunque hay una correlación uno a uno entre un URI y un nombre de interfaz, cada tienda puede especificar si se requiere HTTPS o autenticación para el mandato. Para cada petición de URL de entrada, el controlador Web busca el nombre de interfaz para el mandato de controlador y después utiliza dicho nombre para determinar la clase de implementación correcta, tal como está registrada en la tabla CMDREG.

En la siguiente tabla se describe la información incluida en la tabla de base de datos URLREG.

Nombre de columna	Descripción	Comentarios
URL	Nombre de URI	Por ejemplo, `MyNewCommand` o `ProductDisplay`
STOREENT_ID	Identificador de entidad de tienda	Puede tener el valor `0` para utilizar el mandato para todas las tiendas, o un identificador de tienda exclusivo para indicar que el mandato sólo se utiliza para una tienda concreta.
INTERFACENAME	Nombre de interfaz de mandatos de controlador	Por ejemplo, `com.ibm.commerce.catalog.commands.GetProductDisplay.TemplateCmd`

Nombre de columna	Descripción	Comentarios
HTTPS	Es necesario HTTP seguro para esta petición de URL	Utilice 1 cuando sea necesario HTTPS y 0 cuando no lo sea.
DESCRIPTION	Descripción de URI	Por ejemplo, Este mandato se utiliza con fines de prueba.
AUTHENTICATED	Es necesario que el usuario esté conectado para esta petición de URL	Utilice 1 cuando sea necesario autenticación y 0 cuando no lo sea.
INTERNAL	Indica si el mandato es o no interno para WebSphere Commerce	Utilice 1 cuando el mandato sea interno y 0 cuando sea externo.

Cuando el controlador Web recibe una petición de URL, recupera el nombre de interfaz del mandato de controlador solicitado y lo utiliza para buscar el nombre de la clase de implementación en la tabla CMDREG. También determina si es necesario HTTPS para la petición de URL consultando la columna HTTPS de la tabla URLREG.

Sólo es necesario registrar en la tabla URLREG los mandatos que se invocan a través de peticiones de URL. Por lo tanto, sólo deben registrarse los mandatos de controlador, no los mandatos de tarea ni de vista.

La siguiente sentencia SQL crea una entrada para MyNewControllerCommand, que utiliza una tienda concreta (cuyo identificador de tienda es 5):

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCommand',
5,'com.ibm.commerce.commands.MyNewControllerCommand',0,
'Es un mandato de prueba.',null)
```

La sintaxis genérica para la sentencia insert es como se indica a continuación:

```
insert into nombre_tabla (nombre_col1,nombre_col2, ... ,nombre_coln)
values (valor_col1,valor_col2,...,valor_coln)
```

Los valores que sean series deben indicarse entre apóstrofos.

Tabla CMDREG

CMDREG es la tabla de registro de mandatos. Esta tabla proporciona un mecanismo para correlacionar la interfaz de mandatos con su clase de implementación. Múltiples implementaciones de una interfaz permiten la personalización de mandatos para cada tienda.

En la tabla CMDREG sólo se registran los mandatos de controlador y los mandatos de tarea. Los mandatos de vista se registran en la tabla VIEWREG.

A continuación se describe la información incluida en la tabla de base de datos CMDREG.

Nombre de columna	Descripción	Comentarios
STOREENT_ID	Identificador de entidad de tienda	Puede tener el valor 0 para utilizar el mandato para todas las tiendas, o un identificador de tienda exclusivo para indicar que el mandato sólo se utiliza para una tienda concreta.

Nombre de columna	Descripción	Comentarios
INTERFACENAME	Nombre de interfaz de mandatos	Define la interfaz; utilice el mismo nombre que utilizó en la tabla URLREG.
DESCRIPTION	Descripción de este mandato	Por ejemplo, `Este mandato se utiliza con fines de prueba.`
CLASSNAME	Nombre de clase de implementación del mandato	Normalmente el nombre de interfaz con ″Impl″ añadido al final.
PROPERTIES	Las parejas de nombre-valor por omisión establecidas como propiedades de entrada para el mandato.	El formato es el mismo que la serie de consulta de URL. Por ejemplo, ″parm1=val1&parm2=val2″
LASTUPDATE	Última actualización de esta entrada de mandato	
TARGET	Nombre de destino de mandato. Es donde el mandato se ejecuta realmente.	Sólo se da soporte al destino local.

En general, al crear un nuevo mandato de controlador o de tarea, debe crear la correspondiente entrada en la tabla CMDREG. Por ejemplo, la siguiente sentencia SQL crea una entrada para MyNewCommand, que utiliza una tienda concreta (cuyo identificador es 5):

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION, CLASSNAME,
PROPERTIES, LASTUPDATE, TARGET) values
(5,'com.ibm.commerce.catalog.commands.MyNewCommand', 'Es un mandato de
prueba', 'com.ibm.commerce.catalog.commands.MyNewCommandImpl',
'myDefaulParm1=myDefaultVal1', '0000-12-01', 'Local')
```

La sintaxis genérica para la sentencia insert es como se indica a continuación:

```
 insert into nombre_tabla (nombre_col1,nombre_col2, ... ,nombre_coln)
values (valor_col1,valor_col2,...,valor_coln)
```

Los valores que sean series deben indicarse entre apóstrofos.

Si el mandato que está escribiendo siempre utiliza la misma clase de implementación, no es obligatorio que registre el mandato en la tabla CMDREG. En ese caso, puede utilizar el atributo defaultCommandClassName de la interfaz para especificar la clase de implementación. Por ejemplo, en el código de la interfaz, incluiría lo siguiente:

```
String defaultCommandClassName =
     "com.ibm.commerce.command.MyNewCommandImpl"
```

Si especifica la clase de implementación de esta forma, no puede pasar las propiedades por omisión a la clase de implementación y debe utilizarse la misma clase de implementación para todas las tiendas.

Ejemplo de un mandato de controlador registrado

Suponga que su sitio dispone de dos tiendas: TiendaA y TiendaB. Cada tienda tiene requisitos de seguridad diferentes para el mandato de controlador MyUrl, así como implementaciones diferentes del mandato. En esta sección se muestra cómo se utiliza el registro de mandatos para permitir esta personalización.

La siguiente tabla muestra las entradas para TiendaA y TiendaB en la tabla URLREG:

Nombre de columna	Entrada para TiendaA	Entrada para TiendaB
URL	MyUrl	MyUrl
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands.myUrl
HTTPS	1	1
DESCRIPTION	Entrada de ejemplo de la tabla URLREG.	Entrada de ejemplo de la tabla URLREG.
AUTHENTICATED	1	0
INTERNAL	null	null

Nota: Los espacios que se muestran en los valores de INTERFACENAME sólo se indican a efectos de visualización. Cada valor es en realidad una serie continua.

Basándose en las entradas de la tabla URLREG, el controlador Web determina que el nombre de interfaz para el URI MyURL es com.ibm.commerce.mycommands.MyUrl. También determina que la TiendaA requiere que el mandato se ejecute utilizando HTTPS y autenticación, y la TiendaB sólo requiere HTTPS. Los valores de HTTPS y autenticación los utiliza el controlador Web, no la interfaz.

En el siguiente diagrama se muestra este flujo:

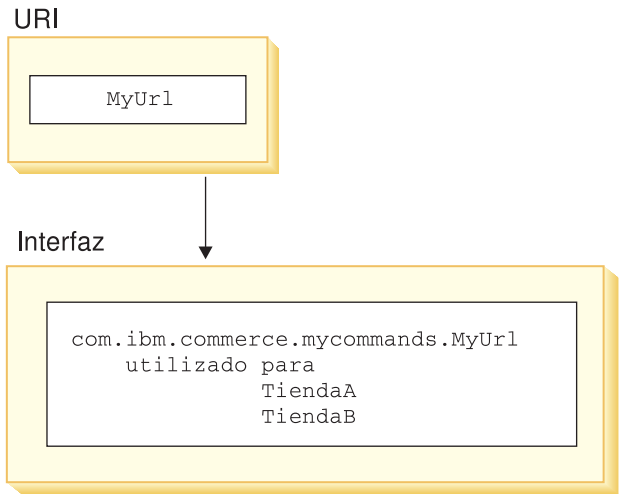

Figura 9.

En la siguiente tabla se muestran las entradas de la tabla CMDREG. Sólo se muestran las columnas necesarias a efectos de este ejemplo:

Nombre de columna	Entrada para TiendaA	Entrada para TiendaB
STOREENT_ID	11	22
INTERFACENAME	com.ibm.commerce. mycommands.myUrl	com.ibm.commerce. mycommands.myUrl

Nombre de columna	Entrada para TiendaA	Entrada para TiendaB
CLASSNAME	com.ibm.commerce. mycommands. myUrlStoreAImpl	com.ibm.commerce. mycommands.myUrlStoreBImpl

Nota: Los espacios que se muestran en los valores de INTERFACENAME y CLASSNAME sólo se indican a efectos de visualización. Cada valor es en realidad una serie continua.

Basándose en las entradas de la tabla CMDREG, el controlador Web determina que para la TiendaA, la clase de implementación para la interfaz com.ibm.commerce.mycommands.MyUrl es com.ibm.commerce.mycommands.MyUrlStoreAImpl. También determina que para la TiendaB, la clase de implementación para la misma interfaz es com.ibm.commerce.mycommands.MyUrlStoreBImpl. En el siguiente diagrama se muestra este flujo:

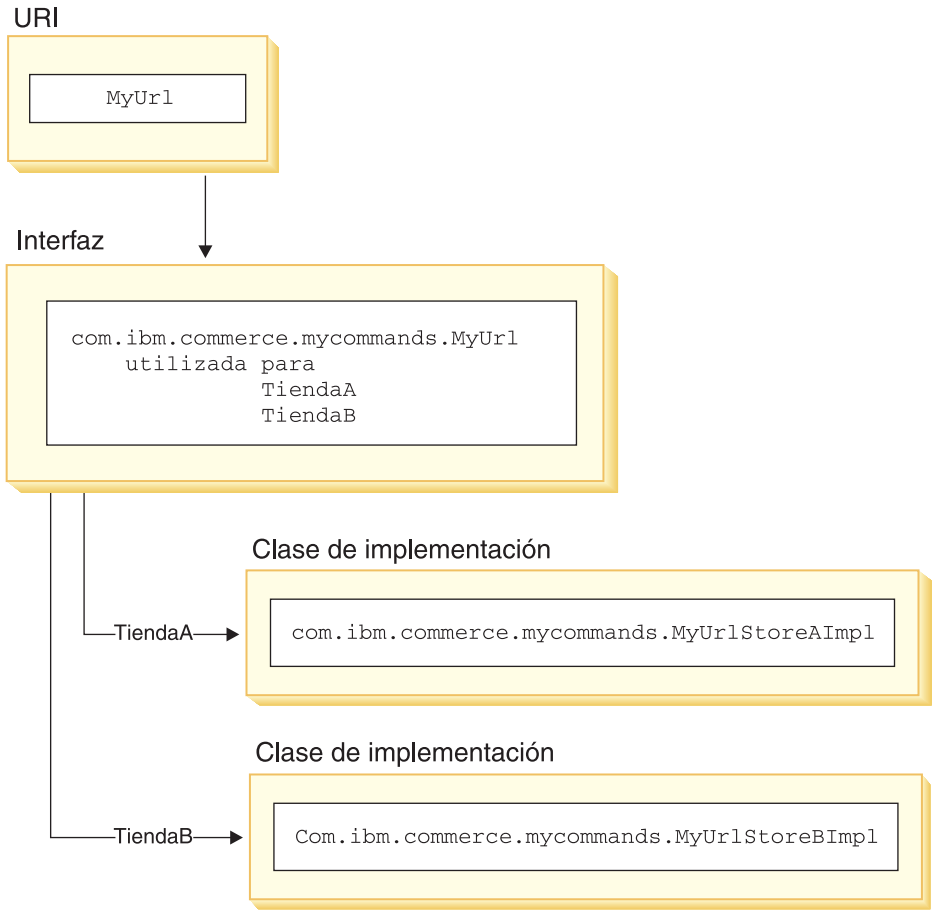

Figura 10.

Tabla VIEWREG

La tabla VIEWREG permite el registro de implementaciones de mandatos de vista específicos de un dispositivo. Utilizando esta tabla, pueden registrarse varias implementaciones de una vista. La infraestructura de mandatos podrá entonces devolver distintas vistas a diferentes clientes.

Cuando el nombre de un mandato de vista se devuelve desde un mandato de controlador o se especifica en una excepción, el controlador Web determina la clase de mandato de vista a partir de la tabla VIEWREG. Pueden correlacionarse varios nombres de mandatos de vista con la misma clase de implementación.

Nombre de columna	Descripción	Comentarios
VIEWNAME	Nombre de vista	Por ejemplo, `AddressForm`
DEVICEFMT_ID	Identificador de tipo de dispositivo	Las opciones disponibles son: • BROWSER (valor por omisión) • I_MODE • Correo electrónico • MQXML • MQNC
STOREENT_ID	Identificador de entidad de tienda	Puede tener el valor `0` para utilizar el mandato para todas las tiendas, o un identificador de tienda exclusivo para indicar que el mandato sólo se utiliza para una tienda concreta.
INTERFACENAME	Nombre de interfaz de mandatos de vista	Las opciones por omisión son ForwardView, DirectView y RedirectView.
CLASSNAME	Nombre de clase de implementación de mandato de vista	Puede utilizar la implementación por omisión.
PROPERTIES	Las parejas de nombre-valor por omisión establecidas como propiedades de entrada para el mandato.	Si siempre se visualiza la misma página, establezca el nombre de archivo JSP en esta propiedad (`docname=`*`nombre_jsp`*`.jsp`). Si se utiliza la misma plantilla JSP para todas las tiendas, establezca el valor `storeDir=no` para impedir que se utilice un directorio de una tienda concreta Si un usuario genérico puede invocar el mandato, establezca el valor `isGeneric=true`.
DESCRIPTION	Descripción de este mandato	
HTTPS	Es necesario HTTP seguro para esta petición de URL	Utilice `1` cuando sea necesario HTTPS y `0` cuando no lo sea.
LASTUPDATE	Última actualización de esta entrada	
INTERNAL	Indica si el mandato es o no interno para WebSphere Commerce	Utilice `1` cuando el mandato sea interno y `0` cuando sea externo.

Al crear un nuevo mandato de vista, puede que sea necesario crear su correspondiente entrada en la tabla VIEWREG. Si se cumple una de las siguientes condiciones, el mandato de vista debe registrarse en la tabla VIEWREG:

- El mandato de vista se ejecuta bajo el control de acceso

- Existen varias implementaciones del mandato de vista
- Hay propiedades establecidas en la columna PROPERTIES

Se puede acceder a los mandatos de vista registrados mediante el registro de mandatos utilizando el nombre de vista, o directamente utilizando el nombre de archivo de visualización real. Sólo se puede acceder a las vistas que no están registradas en la tabla VIEWREG cuando el cliente utiliza el nombre de archivo de visualización real.

Considere el ejemplo de una vista denominada *MyView*, con la entrada VIEWREG de la forma siguiente:

Nombre de columna	**Entrada**
VIEWNAME	MyView
DEVICEFMT_ID	BROWSER
STOREENT_ID	0
INTERFACENAME	com.ibm.commerce.commands.ForwardViewCommand
CLASSNAME	com.ibm.commerce.commands.HTTPForwardViewCommandImp
PROPERTIES	docname=MyView.jsp
DESCRIPTION	Un ejemplo para llamar a una plantilla JSP utilizando el nombre de vista o directamente desde un URL.
HTTPS	0
LASTUPDATE	30-11-2000
INTERNAL	0

Puesto que MyView es una vista registrada, un cliente puede acceder a la vista utilizando el nombre de mandato o sustituyendo el nombre de mandato por el nombre de archivo de visualización real. Si utiliza el nombre de vista, un ejemplo de URL sería:

http://*nombresistpral*.com/webapp/wcs/stores/servlet/MyView

y, si utiliza el nombre de archivo, un ejemplo de URL sería:

http://*nombresistpral*.com/webapp/wcs/stores/servlet/MyView.jsp

Si existe la posibilidad de que el cliente invoque directamente una vista registrada (utilizando el nombre de archivo de visualización), debe registrar el mandato utilizando para la vista el mismo nombre que para el archivo de visualización real, tal como se muestra en este ejemplo (MyView y MyView.jsp).

Una vista que no esté registrada en la tabla sólo puede invocarse utilizando el nombre de archivo de visualización. Por lo tanto, si hay una vista no registrada que utilice el archivo MyUnregisteredView.jsp, el URL para acceder a esta vista será el siguiente:

http://*nombresistpral*.com/webapp/wcs/stores/servlet/MyUnregisteredView.jsp

La siguiente sentencia SQL de ejemplo crea una entrada para *MyNewViewCommand* que utiliza una tienda concreta:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE, INTERNAL) values
('MyNewViewCommand', 'BROWSER', 5,
'com.ibm.commerce.command.ForwardViewCommand',
```

```
'com.ibm.commmerce.command.HttpForwardViewCommandImpl',
'docname=MyNewViewCommand.jsp', 'Un mandato de vista de prueba.', 0,
'0000-12-01', 0)
```

En la siguiente tabla se proporciona otra tabla VIEWREG de ejemplo con información clave:

COMMAND - NAME	DEVICE - FMT_ID	INTERFACE - NAME	CLASSNAME	PROPERTIES
ProductDisplayView	BROWSER	Mandato reenviar vista	HttpForwardView CommandImpl	
InterestItemAddView	BROWSER	Mandato redirigir vista	HttpRedirectView CommandImpl	docname =item.jsp
InterestItemDelete View	BROWSER	Mandato redirigir vista	HttpRedirectView CommandImpl	docname =item.jsp
GenericApplication Error	BROWSER	Mandato redirigir vista	HttpRedirectView CommandImpl	docname =usererr.jsp
GenericSystemError	BROWSER	Mandato redirigir vista	HttpRedirectView CommandImpl	docname =syserr.jsp
Logon	BROWSER	Mandato reenviar vista	HttpForwardView CommandImpl	docname= logon.jsp & storeDir=no

Nota: Los espacios que se muestran en los valores de COMMANDNAME, INTERFACENAME, CLASSNAME y PROPERTIES sólo se indican a efectos de visualización. Cada valor es en realidad una serie continua. Los guiones que aparecen en los nombres de columna también son a efectos de visualización.

La tabla anterior ilustra los siguientes escenarios:

- Un mandato de controlador (en este caso ProductDisplay) devuelve el nombre de vista *ProductDisplayView* al controlador Web. El controlador Web determina el nombre de interfaz de mandatos de vista y el nombre de clase mediante el nombre de mandato de vista ProductDisplayView y su identificador de dispositivo. Un mandato de vista puede tener distintas clases de implementación para diferentes tiendas e identificadores de dispositivo. Sin embargo, el nombre de interfaz debe permanecer igual, puesto que define el tipo de mandato de vista.
- Los mandatos InterestItemAdd e InterestItemDelete devuelven los nombres de vista *InterestItemAddView* e *InterestItemDeleteView*, respectivamente, al controlador Web. Ambos mandatos requieren redirigir vistas, por lo tanto, el nombre de interfaz de mandato de vista para ambas vistas es RedirectViewCommand. Existe una plantilla JSP común registrada para ambas vistas. El controlador Web recopila las propiedades (docname=item.jsp) y las pasa al mandato de vista (HttpRedirectViewCommandImpl).
- Si un mandato de controlador o de tarea genera una excepción ECApplication para un parámetro de usuario erróneo, puede suceder lo siguiente:
  - Si hay una vista especificada en el mandato de controlador a la que debe llamarse en el caso de que se produzca una excepción de la aplicación, la entrada para dicha vista se recupera de la tabla VIEWREG y se procesa según proceda.

- Si no se especifica ninguna vista, se llama al mandato GenericApplicationError y se visualiza la plantilla JSP registrada en la base de datos. Si se utilizara la tabla anterior como ejemplo, se visualizaría la plantilla `usererr.jsp`.

- Si un mandato de controlador o de tarea genera una excepción ECSystem para una excepción del sistema, puede suceder lo siguiente:
  - Si hay una vista especificada en el mandato de controlador a la que debe llamarse en el caso de que se produzca una excepción del sistema, la entrada para dicha vista se recupera de la tabla VIEWREG y se procesa según proceda.
  - Si no se especifica ninguna vista, se llama al mandato GenericSystemError y se visualiza la plantilla JSP registrada en la base de datos. Si se utilizara la tabla anterior como ejemplo, se visualizaría la plantilla `syserr.jsp`.
- Los clientes del navegador pueden invocar la página de conexión entrando en el URL de conexión. Puesto que la propiedad storeDir tiene el valor "no", la información de la tienda concreta no se incluye en la vía de acceso de la plantilla JSP. Por ello, se visualiza la misma página de conexión para los clientes en todas las tiendas.

Patrón de diseño de visualización

Las páginas de visualización devuelven una respuesta a un cliente. Normalmente, las páginas de visualización se implementan como plantillas JSP (el método recomendado), sin embargo, pueden escribirse directamente como servlets.

Para dar soporte a varios tipos de dispositivos, un acceso de URL a un mandato de vista debe utilizar el nombre de vista, no el nombre del archivo JSP real.

La idea que hay detrás de este nivel de direccionamiento indirecto es que la plantilla JSP representa una vista. La posibilidad de seleccionar la vista apropiada (por ejemplo, basándose en el entorno nacional, el tipo de dispositivo u otros datos del contexto de la petición) es muy conveniente, especialmente porque una única petición tiene a menudo varias vistas posibles. Suponga el ejemplo de dos compradores que solicitan la página de presentación de una tienda; un comprador utiliza un navegador Web normal y el otro un teléfono móvil. Evidentemente no debería mostrarse la misma página de presentación a cada uno de los compradores. El controlador Web será el responsable de aceptar la petición y, basándose en la información de la infraestructura del registro de mandatos, determinar la vista que va a mostrarse a cada comprador.

Plantillas JSP y beans de datos

Un bean de datos es un bean Java que se utiliza en una plantilla JSP para proporcionar contenido dinámico. Un bean de datos generalmente proporciona una representación sencilla de un bean de entidad de WebSphere Commerce. El bean de datos encapsula las propiedades que se pueden recuperar o establecer en el bean de entidad. Como tal, el bean de datos simplifica la tarea de incorporar datos dinámicos en las plantillas JSP.

Un bean de datos tiene una clase *BeanInfo* que define las propiedades que pueden utilizarse en la página de visualización. La clase BeanInfo también permite utilizar beans de datos en sitios multiculturales proporcionando nombres de propiedades en todos los idiomas soportados en WebSphere Commerce.

Un bean de datos se activa mediante la siguiente llamada:

```
com.ibm.commerce.beans.DataBeanManager.activate(DataBean, HTTPServletRequest)
```

WebSphere Studio Page Designer genera automáticamente esta línea de código cuando un bean de datos de WebSphere Commerce se inserta en una plantilla JSP.

Al crear plantillas JSP, los desarrolladores de tienda deben tener en cuenta las propiedades de la tienda y las cuestiones sobre la habilitación del soporte multicultural. Para obtener más información sobre la habilitación del soporte multicultural consulte la ayuda en línea de WebSphere Commerce.

Consideraciones sobre seguridad de las plantillas JSP y los beans de datos

Una práctica de codificación específica para el uso de las plantillas JSP y los beans de datos minimiza la posibilidad de que usuarios mal intencionados puedan acceder a su base de datos sin autorización. Las partes de inserción, selección, actualización y supresión de las sentencias SQL deben crearse en el momento del desarrollo. Utilice la inserción de parámetros para reunir la información de entrada de ejecución.

A continuación se muestra un ejemplo de utilización de inserción de parámetro para reunir información de entrada de ejecución:

```
select * from Order where owner =?
```

Por el contrario, debe evitar el uso de series de entrada para componer una sentencia SQL. A continuación se muestra un ejemplo de utilización de serie de entrada:

```
select * from Order where owner = "serie_entrada"
```

Tipos de beans de datos

Un bean de datos es un bean Java que se utiliza principalmente para proporcionar datos dinámicos en plantillas JSP. Hay dos tipos de beans de datos: beans de datos inteligentes y beans de datos de mandato.

Un bean de datos inteligente utiliza un método de *recopilación diferida* para recuperar sus propios datos. Este tipo de bean de datos puede proporcionar un mejor rendimiento en situaciones en las que no son necesarios todos los datos del bean de acceso, puesto que sólo recupera los datos necesarios. Los beans de datos inteligentes que necesitan acceder a la base de datos deben ampliarse del bean de acceso para el bean de entidad correspondiente e implementar la interfaz com.ibm.commerce.SmartDataBean. Por ejemplo, el bean de datos ProductData amplía el bean de acceso ProductAccessBean, que corresponde al bean de entidad Product.

Algunos beans de datos inteligentes no necesitan acceder a la base de datos. Por ejemplo, el bean de datos inteligente PropertyResource recupera datos de un paquete de recursos, en vez de recuperarlos de la base de datos. Cuando no es necesario acceder a la base de datos, el bean de datos inteligente debe ampliar la clase SmartDataBeanImpl.

Un bean de datos de mandato necesita utilizar un mandato para recuperar sus datos y es un bean de datos menos potente. El mandato recupera a la vez todos los atributos del bean de datos, independientemente de si la plantilla JSP los necesita o no. Por ello, para las plantillas JSP que sólo utilizan una selección de atributos del bean de datos, un bean de datos de mandato puede ser muy costoso en términos

de rendimiento. Para las plantillas JSP que requieren la mayoría de los atributos, o todos, el bean de datos de mandato puede ser muy conveniente.

Los beans de datos de mandato también pueden ampliarse de sus beans de acceso correspondientes e implementar la interfaz com.ibm.commerce.CommandDataBean.

Interfaces de bean de datos

Los beans de datos implementan una o todas las interfaces Java siguientes:

- com.ibm.commerce.SmartDataBean.
- com.ibm.commerce.CommandDataBean
- com.ibm.commerce.InputDataBean (opcional)

Cada interfaz Java describe la fuente de los datos que se insertan en un bean de datos. Si implementa varias interfaces, el bean de datos puede acceder a datos de distintas fuentes. A continuación se proporciona más información sobre cada una de las interfaces.

Interfaz SmartDataBean: Un bean de datos que implementa la interfaz SmartDataBean puede recuperar sus propios datos sin tener asociado ningún mandato de bean de datos. Un bean de datos inteligente normalmente se amplía a partir del bean de acceso de un bean de entidad correspondiente. Al activar un bean de datos inteligente, el gestor de beans de datos invoca el método de inserción de datos del bean de datos. Si utiliza el método de inserción de datos, el bean de datos puede recuperar todos los atributos a excepción de los atributos de los objetos asociados. Por ejemplo, si el bean de datos se amplía de una clase de bean de acceso de un bean de entidad, el bean de datos invoca el método refreshCopyHelper. Todos los atributos del bean de entidad correspondiente se insertan automáticamente en el bean de datos inteligente. Sin embargo, si el bean de entidad tiene objetos asociados, los atributos de dichos objetos no se recuperan. Las principales ventajas de utilizar beans de datos inteligentes son las siguientes:

- La implementación es sencilla y no es necesario escribir un mandato de bean de datos.
- Al añadir nuevos campos al bean de entidad, no es necesario realizar cambios en el bean de datos. Después de modificar el bean de entidad, se debe volver a generar el bean de acceso (con las herramientas de VisualAge para Java). Tan pronto se haya vuelto a generar el bean de acceso, el bean de datos inteligente podrá disponer automáticamente de todos los nuevos atributos.
- Los beans de entidad a menudo contienen atributos que representan objetos asociados. Por motivos de rendimiento, el bean de datos inteligente no recupera automáticamente estos atributos. En lugar de eso, es preferible retrasar la recuperación de dichos atributos hasta que sean necesarios, tal como se muestra en el siguiente diagrama:

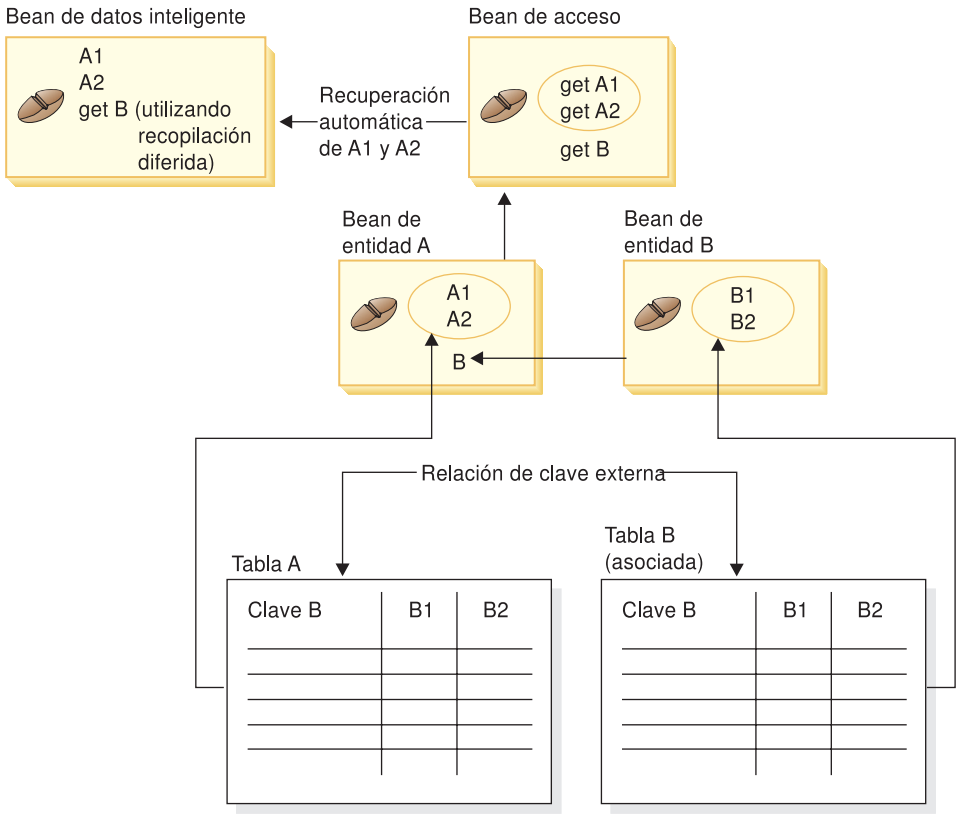

Figura 11.

Para obtener más información sobre cómo implementar una recuperación de recopilación diferida, consulte “Recuperación de datos de recopilación diferida” en la página 37.

Interfaz CommandDataBean: Un bean de datos que implementa la interfaz CommandDataBean recupera los datos de un mandato de bean de datos. Un bean de datos de este tipo es un objeto poco potente; depende de que un mandato de bean de datos inserte sus datos. El bean de datos debe implementar el método getCommandInterfaceName() (tal como lo define la interfaz com.ibm.commerce.CommandDataBean), que devuelve el nombre de interfaz del mandato de bean de datos.

Interfaz InputDataBean: Un bean de datos que implementa la interfaz InputDataBean recupera datos de los parámetros de URL o de los atributos establecidos por el mandato de vista.

Los atributos definidos en esta interfaz pueden utilizarse como campos de clave primaria para obtener datos adicionales. Al invocar una plantilla JSP, el código de servlet JSP generado inserta datos en todos los atributos que coinciden con los parámetros de URL y, a continuación, activa el bean de datos pasándolo al gestor de beans de datos. Después, el gestor de beans de datos invoca el método setRequestProperties() del bean de datos (tal como define la interfaz com.ibm.commerce.InputDataBean) para pasar todos los atributos establecidos por el mandato de vista. Debe tenerse en cuenta que WebSphere Studio genera el siguiente código para cada bean de datos que se inserta en las páginas utilizando Page Designer:

```
com.ibm.commerce.beans.DataBeanManager.activate(DataBean, HTTPServletRequest);
```

Clase BeanInfo

Un bean de datos no está completo sin una clase BeanInfo que implemente la interfaz java.lang.Object.BeanInfo. La clase BeanInfo se utiliza para proporcionar información explícita sobre los métodos y propiedades del bean de datos. Puede utilizarse para esconder del diseñador Web los métodos de ejecución públicos de la clase de implementación del bean de datos o para establecer la serie de visualización adecuada para cada uno de los atributos del bean de datos.

Para obtener más información sobre cómo implementar una clase BeanInfo, consulte la especificación JavaBeans de Sun Microsystems.

Activación de los beans de datos

Los beans de datos pueden activarse utilizando los métodos activate o silentActivate que están en la clase com.ibm.commerce.beans.DataBeanManager. El método activate es un método de activación completo en el que el suceso de activación es satisfactorio sólo si están disponibles todos los atributos. Incluso en el caso de que sólo falte un atributo, se generará una excepción para todo el proceso de activación.

El método silentActivate no genera excepciones cuando no están disponibles atributos individuales.

Llamada a mandatos de controlador desde una plantilla JSP

Aunque efectuar llamadas a mandatos de controlador desde una plantilla JSP no es muy coherente con separar la lógica de la visualización, pueden producirse situaciones en las que sea necesario hacerlo. En ese caso, puede utilizar `ControllerCommandInvokerDataBean`.

Utilizando este bean de datos, puede especificar el nombre de interfaz del mandato a llamar, o puede establecer directamente el nombre de ese mandato. También puede establecer las propiedades de petición para el mandato.

Cuando el gestor de bean de datos activa el bean de datos, se ejecuta el mandato de controlador y las propiedades de respuesta se ponen a disposición de la plantilla JSP.

Una vez ejecutado el mandato de controlador, puede ejecutar la vista.

Recuperación de datos de recopilación diferida

Cuando se activa un bean de datos, éste puede rellenarse con datos mediante un mandato de bean de datos o mediante el método populate() del bean de datos. Los atributos que se recuperan proceden del bean de entidad correspondiente al bean de datos. Un bean de entidad también puede tener objetos asociados, que a su vez tienen varios atributos.

Si, durante la activación, se recuperan automáticamente los atributos de todos los objetos asociados, puede que haya un problema de rendimiento. El rendimiento puede disminuir a medida que aumente el número de objetos asociados.

Suponga un bean de datos de producto que contenga un gran número de productos de venta cruzada, venta ascendente o accesorios (objetos asociados). Es posible insertar datos en todos los objetos asociados tan pronto se active el bean de datos de producto. Sin embargo, si se insertan datos de esta forma será necesario realizar múltiples consultas a la base de datos. Si la página no necesita todos los atributos, puede que las múltiples consultas a la base de datos sean inútiles.

En general, una página no necesita todos los atributos y, por lo tanto, el mejor patrón de diseño es realizar una recopilación diferida tal como se muestra a continuación:

```
getCrossSellProducts () {
   if (crossSellDataBeans == null)
     crossSellDataBeans= getCrossSellDataBeans();
   return crossSellDataBean;
    }
```

Establecimiento de atributos JSP - Visión general

El modelo de programa de WebSphere Commerce fomenta el uso del patrón de diseño MVC. Como tal, la presentación del resultado de una petición de URL se separa de los mandatos de tarea y de controlador. Estos mandatos son independientes del dispositivo. Implementan la lógica de negocio y generan datos que se han de devolver al cliente, sin tener información sobre el cliente. Por el contrario, un mandato de vista es específico del dispositivo.

Aunque los mandatos de controlador y de tarea no creen directamente la vista, sí pasan información a la vista. Es importante comprender cómo se pasa la información a la vista. El siguiente diagrama muestra cómo se pasan las propiedades entre el controlador Web, el registro de mandatos, el mandato de controlador y el mandato de vista.

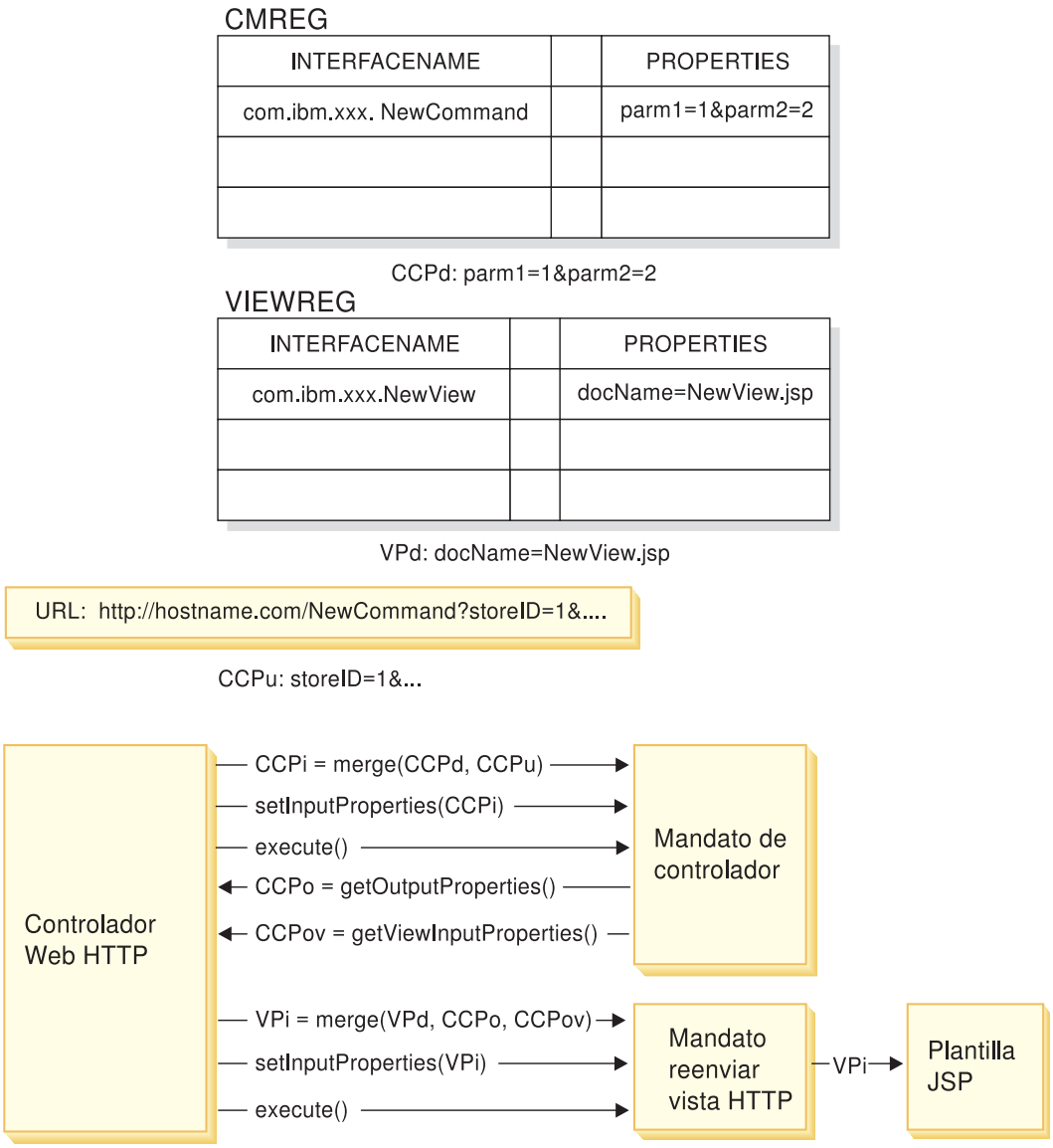

Figura 12.

En el diagrama anterior se muestran las siguientes interacciones:

- El controlador Web fusiona las propiedades de entrada de los parámetros de URL (CCPu) y la entrada de la tabla CMDREG para el mandato de controlador (CCPd). Esto crea CCPi.
- El controlador Web pasa las propiedades fusionadas (CCPi) al mandato de controlador y ejecuta el mandato de controlador.
- El mandato de controlador establece las propiedades de salida, como CCPo. Estas son las propiedades de salida generadas por el propio mandato. Una de las propiedades de salida, viewCommandName, se establece en el nombre de mandato de vista deseado. El controlador Web recupera estas propiedades utilizando el método get.
- El mandato de controlador establece otro conjunto de propiedades de salida, como CCPov. Por omisión, se establecen en el valor de las propiedades de entrada fusionadas originales (CCPi). Si se desea, es posible personalizar estas propiedades. Por ejemplo, es posible que no sea necesario pasar todos los parámetros de entrada al mandato de vista.

- El controlador Web fusiona los tres conjuntos de propiedades, CCPo, CCPov y VPd (las propiedades que están registradas en la tabla VIEWREG) en las propiedades de entrada para el mandato de vista (VPi).
- El controlador Web establece las propiedades fusionadas, VPi, y ejecuta el mandato de vista.
- El mandato de vista establece los atributos para la plantilla JSP tomándolos de las propiedades de entrada.

Al escribir nuevos mandatos, no tiene que llevar a cabo de forma explícita la fusión de las propiedades. Las clases de mandatos abstractas incluyen un método mergeProperties. Para obtener más información sobre este método, consulte la sección "Referencias" de la ayuda en línea de WebSphere Commerce.

Valores de propiedades necesarios

Un mandato de controlador debe establecer las siguientes propiedades para cada tipo de mandato de vista. Si el mandato no establece las propiedades, éstas deben definirse en la tabla VIEWREG.

- Si utiliza el mandato ForwardView, establezca `docname = nombre_archivo_vista`, donde *nombre_archivo_vista* es el nombre de la plantilla de visualización. Por ejemplo, `docname = productDisplay.jsp`.
- Si utiliza el mandato DirectView, efectúe una de las siguientes acciones:
  - Establezca el valor `textDocument = xxx`, donde *xxx* es un objeto java.io.InputStream que contiene el documento en formato de texto.
  - Establezca el valor `rawDocument = yyy`, donde *yyy* es un objeto java.io.InputStream que contiene el documento en formato binario.

  Si utiliza el mandato DirectView, es opcional establecer `contentType = ttt`, donde *ttt* es el tipo de contenido del documento.
- Si utiliza el mandato RedirectView, establezca `url = uuu`, donde *uuu* es el URL de redirección.

Capítulo 3. Modelo de objeto persistente

WebSphere Commerce se ocupa de una gran cantidad de datos persistentes. Hay más de 520 tablas definidas en el esquema de base de datos actual. Incluso con este amplio esquema, es posible que sea necesario ampliar o personalizar el esquema de base de datos para satisfacer las necesidades de cada negocio.

WebSphere Commerce utiliza beans de entidad que se basan en la arquitectura de componentes Enterprise JavaBeans (EJB) como capa de objetos persistente. Estos beans de entidad representan datos de WebSphere Commerce de una forma que modela conceptos y objetos del dominio de comercio. Esta capa de persistencia proporciona una infraestructura que puede ampliarse.

VisualAge para Java, Enterprise Edition proporciona unas sofisticadas herramientas EJB y un entorno de prueba de unidades que da soporte al desarrollo de esta infraestructura.

Implementación de los beans de entidad de WebSphere Commerce

Beans de entidad de WebSphere Commerce - Visión general

Como se ha indicado anteriormente, la capa de persistencia incluida en la arquitectura de WebSphere Commerce se implementa según la arquitectura de componentes EJB. La arquitectura EJB define dos tipos de beans enterprise: beans de entidad y beans de sesión. Los beans de entidad se subdividen en beans CMP (″container-managed persistence″ o persistencia gestionada por contenedor) y beans BMP (″bean-managed persistence″ o persistencia gestionada por bean).

La mayoría de los beans de entidad de WebSphere Commerce son beans de entidad CMP. Se utiliza un pequeño número de beans de sesión sin estado para manejar operaciones que utilizan muchos recursos de la base de datos como, por ejemplo, efectuar una suma de todas las filas de una determinada columna. Una de las ventajas de utilizar beans de entidad CMP es que los desarrolladores pueden emplear las herramientas EJB que se suministran en VisualAge para Java, Enterprise Edition. Estas herramientas permiten a los desarrolladores definir objetos Java y sus correlaciones con tablas de base de datos. La herramientas generan automáticamente los métodos de persistencia necesarios para los beans de entidad. Los métodos de persistencia son objetos Java que hacen que los campos Java sean persistentes en la base de datos y rellenan los campos Java con datos de la base de datos.

VisualAge para Java proporciona dos ampliaciones a las especificaciones EJB actuales: la asociación y la herencia EJB. La herencia EJB permite que un bean enterprise herede las propiedades, los métodos y los atributos del descriptor de control a nivel del método de otro bean enterprise que resida en el mismo grupo. Una asociación es una relación que existe entre dos beans de entidad CMP.

Algunos de los beans de entidad de WebSphere Commerce aprovechan la característica de herencia EJB. Los beans de entidad de WebSphere Commerce no utilizan la característica de asociaciones que proporciona VisualAge para Java. Al crear sus propios beans de entidad, se recomienda no utilizar la característica de asociación de VisualAge para Java. Esta recomendación tiene por finalidad minimizar la complejidad del modelo de objeto. En vez de utilizar la característica

de asociación que proporciona VisualAge para Java, puede establecerse una relación de objetos entre beans enterprise añadiendo métodos get explícitos en los beans enterprise.

WebSphere Commerce proporciona dos conjuntos de beans enterprise: privados y públicos. Los beans enterprise privados los utilizan las herramientas y el entorno de ejecución de WebSphere Commerce. *No debe* utilizar ni modificar estos beans.

Por otro lado, los beans enterprise públicos los emplean las aplicaciones de comercio y pueden utilizarse y ampliarse. Estos beans enterprise públicos se organizan en los siguientes grupos EJB:

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon
- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder
- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

Algunos de los grupos EJB arriba indicados contienen beans de sesión. Para simplificar la migración en el futuro, no debe modificar una clase de bean de sesión. Si es necesario, puede crear un nuevo bean de sesión en un nuevo grupo EJB. Para obtener más información sobre la creación de beans de sesión nuevos, consulte “Creación de nuevos beans de sesión” en la página 63.

Descriptores de despliegue para beans enterprise de WebSphere Commerce

Un descriptor de despliegue es una clase especial que está serializada y que contiene los valores de ejecución para un bean enterprise. Los descriptores de despliegue para los beans enterprise de WebSphere Commerce se establecen de una determinada forma y no deben modificarse.

Al crear beans enterprise nuevos (o beans de sesión o de entidad), establezca los descriptores de despliegue en el entorno de desarrollo EJB (de VisualAge para Java) pulsando el botón derecho del ratón y seleccionando **Propiedades**. Los descriptores de despliegue para los beans enterprise nuevos deben seguir los mismos convenios que aquellos para los beans enterprise de WebSphere Commerce. En particular, asegúrese de establecer los atributos tal como se muestra

a continuación:

Atributo	Valor
Atributo de transacción	TX_REQUIRED
Nivel de aislamiento	TRANSACTION_READ_COMMITTED
Modalidad ejecutar como	SYSTEM_IDENTITY o CLIENT_IDENTITY
Reentrante	Compruebe que no esté seleccionado.

Un bean enterprise a menudo contiene métodos que sólo leen información de la base de datos, pero nunca realizan actualizaciones de la base de datos. Estos métodos se conocen como métodos de *sólo lectura*. Todos los métodos de sólo lectura deben indicarse explícitamente como tales (pulse con el botón derecho del ratón en el método y seleccione **Atributos de método EJB > Método de sólo lectura**). Si los métodos de sólo lectura no están indicados de esta manera, el contenedor EJB intenta inútilmente actualizar la base de datos al final de una transacción y produce un error de retrotracción de transacción en la transacción de sólo lectura. Esto ocasiona problemas de rendimiento.

Niveles de aislamiento

El nivel de aislamiento de transacciones que se utiliza para los beans enterprise de WebSphere Commerce dentro de VisualAge para Java es TRANSACTION_READ_COMMITTED. Observe que existe una diferencia entre la implementación de este nivel de aislamiento para el controlador JDBC para DB2 y para el controlador JDBC para Oracle. Por ello, el nivel de aislamiento de transacciones que se utiliza al desplegar en el entorno de WebSphere Application Server varía en función de la base de datos que se utilice.

El nivel de aislamiento utilizado para las bases de datos DB2 cuando funcionan fuera del Entorno de prueba WebSphere es TRANSACTION_REPEATABLE_READ. El nivel de aislamiento utilizado para las bases de datos Oracle cuando funcionan fuera del Entorno de prueba WebSphere es TRANSACTION_READ_COMMITTED.

Si está desplegando en una base de datos DB2, no es necesario cambiar manualmente el nivel de aislamiento de transacciones, se cambia durante el paso del despliegue en el que se emite el mandato modifyIsolationLevel.

La tabla siguiente muestra la relación entre los niveles de aislamiento de transacciones DB2 y Oracle y el nivel de aislamiento de transacciones JDBC correspondiente.

JDBC	DB2	Oracle
Read Uncommitted	Uncommitted Read	Read Uncommitted
Read Committed	Cursor Stability	(No aplicable)
Repeatable Read	Read Stability	Read Committed
Serializable	Repeatable Read	Serializable
Ninguno	(No aplicable)	(No aplicable)

Para el nivel de aislamiento de transacciones Cursor Stability en DB2, sólo se bloquearán de forma exclusiva las filas que se hayan actualizado durante la transacción dada. Si no se ha actualizado ninguna columna de una fila dada, aunque forme parte del conjunto de resultados devueltos de una sentencia SQL, el bloqueo de esa fila específica se liberará cuando el cursor se desplace a otra fila. En

algunas situaciones, como cuando se actualiza el inventario, este comportamiento puede no ser el deseado. Por tanto, al cambiar el nivel de aislamiento de transacciones a Read Stability en DB2, con una ligera diferencia en simultaneidad, puede mejorarse mucho la integridad de datos.

En el caso de Oracle, en el que sólo están disponibles los niveles de aislamiento de transacciones Read Committed o el equivalente JDBC Repeatable Read, la implementación se efectúa de manera que se obtenga el comportamiento más parecido al nivel de aislamiento de transacciones Repeatable Read en DB2.

Ampliación del modelo de objeto de WebSphere Commerce

El modelo de objeto de WebSphere Commerce puede ampliarse de las siguientes maneras:

- Ampliar los beans enterprise públicos de WebSphere Commerce
- Escribir un bean de entidad nuevo
- Escribir un bean de sesión sin estado nuevo

En las siguientes secciones se muestran los detalles sobre cómo llevar a cabo estas ampliaciones.

Metodologías para la ampliación del modelo de objeto

Los requisitos de la aplicación pueden llevarle a ampliar el modelo de objeto de WebSphere Commerce existente. Un ejemplo de tales requisitos es la adición de atributos adicionales a la aplicación. Esto puede llevarse a cabo de una de las siguientes maneras:

Sin modificar un bean de entidad público existente de WebSphere Commerce
Cree una nueva tabla de base de datos y, a continuación, cree un nuevo bean de entidad para dicha tabla. Añada campos y métodos al bean de entidad para manipular el nuevo atributo según sea necesario. Genere código desplegado y un bean de acceso para el nuevo bean de entidad. Cuando la aplicación requiere el nuevo atributo, crea una instancia de un objeto bean de acceso y utiliza los métodos de dicho bean para recuperar, establecer o manipular el atributo.

Modificando un bean de entidad público existente de WebSphere Commerce
Cree una nueva tabla de base de datos y luego cree una unión de tabla entre la nueva tabla y la tabla existente que corresponde al bean enterprise existente que va a modificar. Cree nuevos campos en el bean de entidad público existente de WebSphere Commerce y correlacione los campos con sus columnas correspondientes en la nueva tabla, utilizando una correlación de tabla secundaria. Añada todos los métodos necesarios. Vuelva a generar el código desplegado y el bean de acceso para el bean de entidad existente. Los nuevos atributos estarán disponibles cuando la aplicación cree una instancia del objeto bean de acceso.

Existen diferencias entre estos dos métodos. En general, las diferencias están relacionadas con el rendimiento y el trabajo necesario para llevar a cabo el mantenimiento del código.

Ejemplo de ampliación: Supongamos un ejemplo en el que su aplicación precise la captura del tipo de vivienda que tiene un cliente. Puede crear una tabla denominada USERRES que contenga el ID y el tipo de residencia de los clientes, donde el tipo de residencia (resType) puede ser una casa de propiedad, un condominio o un piso. Este tipo de información son datos estadísticos y, como tales, están relacionados con la tabla USERDEMO existente de Commerce Suite. Al examinar el depósito de código de WebSphere Commerce, descubre que el grupo

EJB WCSUser contiene un bean enterprise ″Demographics″. Este bean tiene los métodos get y set para obtener y establecer la información de datos estadísticos almacenada en la tabla USERDEMO.

Para llevar a cabo la personalización, tiene dos opciones. Puede crear un nuevo bean de entidad que interactúe con la tabla USERRES, o bien puede añadir un nuevo campo (además de los métodos get y set apropiados) al bean Demographics.

Si utiliza el primer método (crear código totalmente nuevo), tendrá que crear un nuevo bean de entidad Userres y correlacionar sus campos con las columnas de la tabla USERRES. Cuando la aplicación necesite el tipo de residencia del cliente, deberá crear una instancia del objeto bean de acceso Userres y recuperar los datos. Si la aplicación requiere otra información de datos estadísticos al mismo tiempo, también deberá crear una instancia de un objeto bean de acceso Demographics y recuperar cualquier otro atributo necesario. Todas las partes de la lógica de la aplicación que intenten recuperar un conjunto completo de datos estadísticos para un cliente deben modificarse para crear una instancia del nuevo bean de acceso así como del original. El diagrama siguiente muestra este método de ampliar el modelo de objeto:

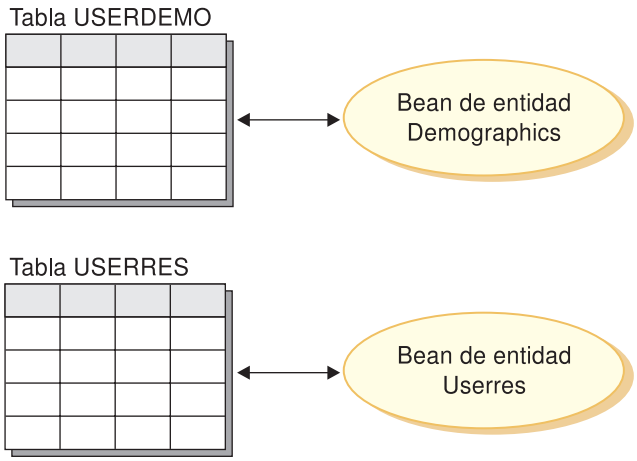

Figura 13.

Desde la perspectiva de una plantilla de visualización, un bean de datos debe poder acceder al nuevo atributo, de manera que la información esté disponible para las plantillas JSP. Para poder presentar una vista unificada al desarrollador Web que crea las plantillas JSP, debería crear un nuevo bean de datos que amplíe el bean de acceso para el bean de entidad original existente. El bean de datos también debería utilizar la delegación para llenar con datos los atributos del nuevo bean de acceso. En el diagrama siguiente se muestra este ejemplo de implementación de bean de datos:

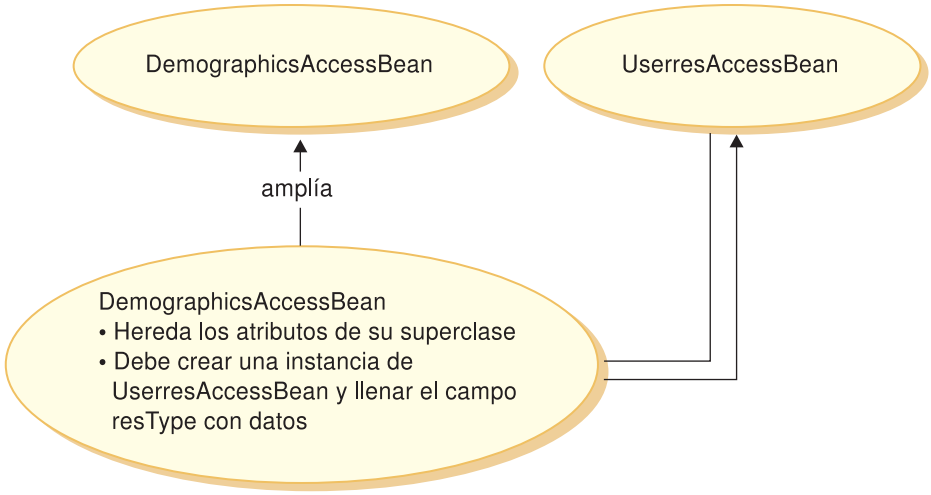

Figura 14.

Si utiliza el segundo método (modificar código existente), tendrá que añadir un nuevo campo al bean de entidad Demographics y crear una correlación de tabla secundaria entre el nuevo campo y la columna adecuada de la tabla USERRES. Cuando la aplicación necesite el tipo de residencia del cliente, creará una instancia del objeto bean de acceso Demographics y recuperará el tipo de residencia. Si la aplicación requiere cualquier otra información de datos estadísticos sobre el cliente, está disponible en la misma llamada al bean. El diagrama siguiente muestra este método para la modificación del bean enterprise:

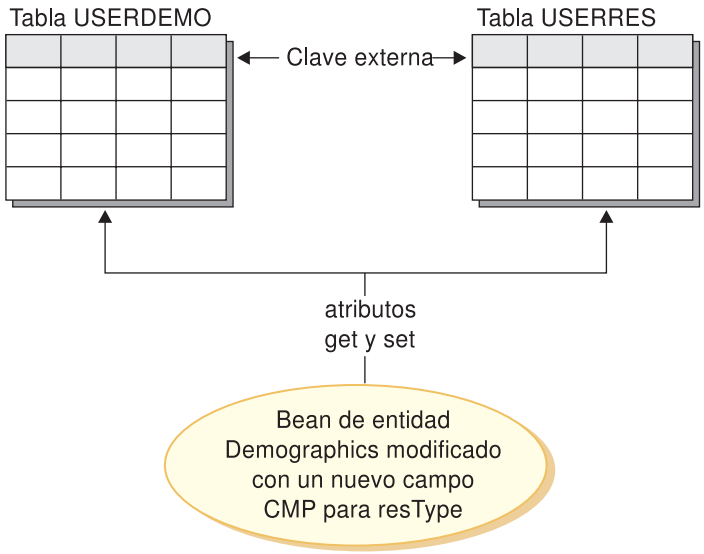

Figura 15.

Desde la perspectiva de una plantilla de visualización, el nuevo atributo (resType) está disponible automáticamente en el bean de datos, en cuanto se vuelva a generar el bean de acceso DemographicsAccessBean.

Tenga en cuenta que cuando expanda el modelo de objeto, *no* debe añadir columnas nuevas a las tablas de base de datos existentes de WebSphere Commerce. Debe crear una tabla nueva para el nuevo atributo. Si intenta añadir columnas

nuevas a tablas existentes, el nuevo atributo se perderá cuando haga una migración a futuros releases de WebSphere Commerce.

Implicaciones en el rendimiento y el mantenimiento del código: El segundo método presenta un mejor nivel de rendimiento durante la ejecución. Esto se debe al hecho de que para obtener o establecer el nuevo atributo sólo se necesita crear la instancia de un único bean de entidad y se utiliza una sola operación de extracción para recuperar todos los atributos necesarios.

Debido al hecho de que el segundo método modifica el código existente de WebSphere Commerce, surgirá un problema de migración cuando haya disponible un nuevo depósito de código de WebSphere Commerce. Debe fusionar su código personalizado con el nuevo código, pero al importar el nuevo depósito de código, la información de correlación entre los campos que ha añadido al bean enterprise y la nueva tabla no se conserva. Como resultado, al hacer una migración a un nuevo release del depósito de código de WebSphere Commerce, deben llevarse a cabo los pasos siguientes:

1. Cree una versión de su código EJB personalizado.
2. Importe la nueva versión del código de WebSphere Commerce.
3. Utilizando las herramientas de VisualAge para Java, compare la versión personalizada del código con el nuevo release del código de WebSphere Commerce. Fusione su código personalizado en su área de trabajo.
4. Vuelva a correlacionar manualmente cualquier atributo que haya añadido a los beans enterprise públicos de WebSphere Commerce con las columnas adecuadas de la base de datos.
5. Vuelva a generar código desplegado y beans de acceso para los beans enterprise que ha modificado en el paso 4.

Para poder simplificar esta migración, es importante documentar de forma completa las ampliaciones del modelo de objeto durante el proceso de desarrollo.

Cuando realice muchas ampliaciones al modelo de objeto tal vez le interese utilizar una combinación de los dos métodos. Puede utilizar el primer método para las áreas del sistema que sean menos susceptibles de sufrir un descenso del rendimiento y utilizar el segundo método cuando el rendimiento sea una cuestión a tener en cuenta. De este modo, puede minimizar el trabajo necesario para una futura migración, sin dejar de mantener buenos niveles de rendimiento del sistema.

Uso recomendado de los beans de sesión

Una de las características más importantes de WebSphere Commerce surgen de su capacidad para aprovechar los beans de entidad CMP (persistencia gestionada por contenedor). Los beans de entidad CMP son componentes Java distribuidos, persistentes y transaccionales en el lado del servidor que se pueden generar con las herramientas proporcionadas con VisualAge para Java. En muchos casos, los beans de entidad CMP son una opción excelente para la persistencia de objetos y se pueden hacer trabajar de forma tan eficaz o más que las otras opciones de correlación objeto a relacional. Por estas razones, WebSphere Commerce ha implementado objetos centrales de comercio utilizando beans de entidad CMP.

Sin embargo, hay situaciones en las que se recomienda utilizar una ayuda JDBC de bean de sesión. Estas situaciones incluyen lo siguiente:

- Un caso en el que una consulta devuelve un conjunto de resultados muy extenso. Este caso se denomina *conjunto de resultados extenso*.
- Un caso en el que una consulta recupera datos de varias tablas. Este caso se denomina *entidad agregada*.

- Un caso en el que una sentencia SQL efectúa una operación intensiva de base de datos. Este caso se denomina *SQL arbitrario*.

En las secciones siguientes se proporcionan más detalles.

Tenga en cuenta que si el bean de sesión se utiliza como wrapper JDBC para recuperar información de la base de datos, será más difícil implementar control de acceso a nivel de recurso. Cuando se utiliza un bean de sesión de esta forma, el desarrollador del bean de sesión debe añadir la cláusula "`where`" en la sentencia "`select`" para impedir que usuarios no autorizados puedan acceder a los recursos.

Conjunto de resultados extenso: Hay casos en los que una consulta devuelve un gran número de resultados y los datos recuperados son para lectura o para que se visualicen. En este caso, es mejor utilizar un bean de sesión sin estado y, dentro de este bean, crear un método de búsqueda que efectúe las mismas funciones que el método de búsqueda en un bean de entidad. Es decir, el método de búsqueda en el bean de sesión sin estado debe hacer lo siguiente:

- Ejecutar una sentencia select de SQL
- Para cada fila que se busca, crear una instancia de un bean de acceso
- Para cada columna recuperada, establezca los atributos correspondientes en el bean de acceso

Cuando se devuelve el bean de acceso, el mandato no sabe si el bean de acceso lo ha devuelto un método de búsqueda en un bean de sesión o un método de búsqueda en un bean de entidad. Como resultado, al utilizar un método de búsqueda en un bean de sesión no se efectúa ningún cambio en el modelo de programación. Sólo el mandato que llama sabe si está invocando un método de búsqueda en un bean de sesión o en un bean de entidad. Es transparente a todas las demás partes del modelo de programación.

Entidad agregada: En este caso, una vista combina partes de varios objetos y una sola página de visualización se rellena con información proveniente de varias tablas de base de datos. Por ejemplo, tomemos el concepto de "Mi cuenta". Este puede constar de información de la tabla de información de cliente (por ejemplo, el nombre de cliente, la edad y el ID de cliente) e información de una tabla de dirección (por ejemplo, una dirección que consta de una calle y una ciudad).

Es posible construir una sentencia SQL sencilla para recuperar toda la información de las distintas tablas ejecutando una función join de SQL. A esto se le puede denominar "recopilación detallada". A continuación se muestra un ejemplo de una sentencia select de SQL para el ejemplo de "Mi cuenta", donde la tabla CUSTOMER es T1 y la tabla ADDRESS es T2:

```
select T1.NAME, T1.AGE, T2.STREET, T2.CITY
  from CUSTOMER T1, ADDRESS T2
  where (T1.ID=? and T1.ID=T2.ID)
```

Las herramientas de EJB en VisualAge para Java no dan soporte a esta noción de recopilación detallada. En su lugar, se efectúa una recopilación diferida que tiene como resultado un select de SQL para cada objeto asociado. Este no es el método preferido para recuperar este tipo de información.

Para efectuar una recopilación detallada se recomienda que utilice un bean de sesión. En ese bean de sesión, debe crear un método de búsqueda para recuperar la información requerida. El método de búsqueda debe hacer lo siguiente:

- Ejecutar una sentencia select de SQL para la recopilación detallada

- Crear una instancia de un bean de acceso para cada fila en la tabla principal así como para cada objeto asociado.
- Para cada columna y objeto asociado en que se busca, establezca el atributo correspondiente en el bean de acceso.

Tenga en cuenta que un bean de acceso no almacena en antememoria un método get que produce una excepción. En ese caso, debe crear una clase de wrapper simple para el bean de acceso utilizando el siguiente esquema:

```
public class CustomerAccessBeanCopy extends CustomerAccessBean {
   private AddressAccessBean address=null;

   /* El siguiente método modifica el método getAddress en
      CustomerAccessBean.
   */
   public AddressAccessBean getAddress() {
      if (address == null)
         address = super.getAddress();
      return address;
      }

    /* El siguiente método establece la dirección a copiar. */

    public void _setAddress(AddressAccessBean aBean) {
       address = aBean;
      }
}
```

Siguiendo con el ejemplo de CUSTOMER y ADDRESS, el método de búsqueda del bean de sesión creará una instancia de CustomerAccessBean para cada fila de la tabla CUSTOMER y un AddressAccessBean para cada fila correspondiente de la tabla ADDRESS. A continuación, para cada columna de la tabla ADDRESS, establece los atributos en AddressAccessBean (calle y ciudad). Para cada columna de la tabla ADDRESS, establece los atributos en CustomerAccessBean (nombre, edad y dirección). Esto se muestra en el siguiente diagrama:

Figura 16.

SQL arbitrario: En este caso, hay un conjunto de sentencias SQL arbitrarias que efectúan operaciones intensivas de base de datos. Por ejemplo, la operación de sumar todas las filas de una tabla se consideraría una operación intensiva de base de datos. Es posible que no todas las filas seleccionadas correspondan a un bean de entidad en el modelo persistente.

Un ejemplo que podría dar como resultado la creación de una sentencia SQL arbitraria es cuando un cliente intenta navegar a través de un conjunto grande de datos. Por ejemplo, cuando el cliente quiere examinar todos los tornillos en una ferretería en línea o todos los vestidos en una tienda de ropa en línea. Esto crea un conjunto de resultados muy grande, pero aparte de este conjunto de resultados, es muy probable que sólo se necesiten unos cuantos campos de cada fila. Es decir, al cliente inicialmente quizá sólo se le presente un resumen mostrando el nombre del artículo, el gráfico y el precio.

En este caso, cree un método de ayuda de bean de sesión. Este método de ayuda de bean de sesión efectúa una operación de lectura o de grabación. Al efectuar una operación de lectura, devuelve un objeto de valor de sólo lectura que se utiliza para visualización.

Con el modelado adecuado de datos, el número de casos de sentencias SQL arbitrarias normalmente puede minimizarse.

Ampliación de beans de entidad públicos

Esta sección describe el patrón de diseño de los beans de entidad públicos de WebSphere Commerce. Este patrón de diseño le permite realizar ampliaciones tales como añadir nuevos campos persistentes, nuevos métodos de negocio o nuevos métodos de buscador.

El siguiente diagrama muestra las clases de implementación del bean de entidad de Catálogo.

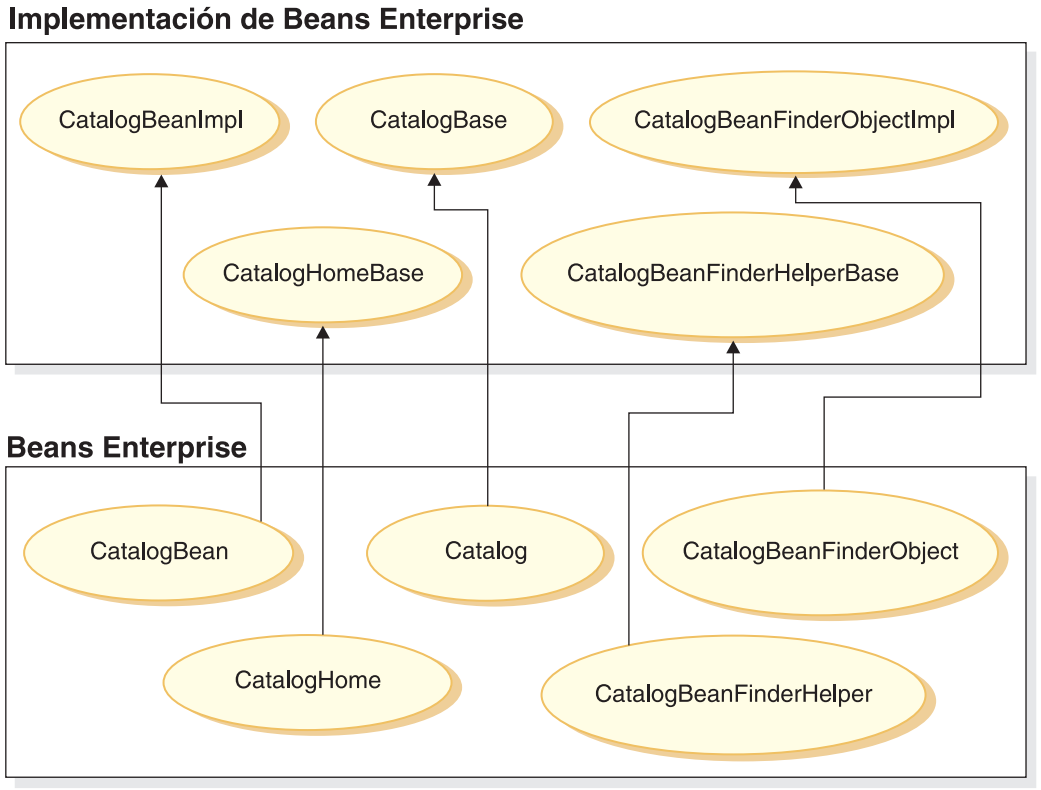

Figura 17.

El diagrama anterior también se aplica a otros beans de entidad porque su estructura es similar y porque siguen el mismo convenio de denominación. Para aplicar el diagrama a otro bean de entidad, sustituya el nombre del bean de entidad por "Catalog". Por ejemplo, la clase InterestItemBean amplía la clase InterestItemBeanImpl y la interfaz InterestItem amplía la interfaz InterestItemBase.

El diagrama muestra que la clase de implementación o interfaz para los beans enterprise públicos se ha separado en dos partes, mediante la característica de herencia de Java. La superclase o interfaz contiene el código de implementación de WebSphere Commerce. Todas estas superclases e interfaces se definen en paquetes y proyectos separados de las interfaces y las clases hijo.

Con la excepción de las clases *nombre_bean*BeanFinderHelperBase y *nombre_bean*BeanFinderObjectImpl, depósito de códigos de WebSphere Commerce contiene código binario para todas las superclases e interfaces. Se incluye el código fuente para las clases *nombre_bean*BeanFinderHelperBase y para *nombre_bean*BeanFinderObjectImpl, por lo tanto, podrá ver cómo se ha definido la sentencia SQL para cada buscador. No modifique estas clases.

Para analizar el código fuente para CatalogBeanFinderHelperBase y CatalogBeanFinderObjectImpl, abra el paquete com.ibm.commerce.catalog.objsrc. Otros paquetes utilizan un convenio de nombres similar.

Las modificaciones pueden realizarse en las interfaces y clases hijo.

Si añade nuevos métodos de buscador a los beans enterprise públicos, debe seguir un convenio de denominación específico para los métodos. Denomine los nuevos métodos findX*descripción_a* donde *descripción_a* es una descripción de su elección. Algunos ejemplos de nombres son findXByOwnerId y findXByOrderStatus. Al utilizar este convenio de denominación se evita el riesgo de colisión de nombres (nombres duplicados) con los métodos de buscador de WebSphere Commerce.

Una forma de modificar un bean de entidad público de WebSphere Commerce es añadir campos adicionales. En este caso, después de añadir los nuevos campos, debe examinar cada uno de los métodos Finder en el bean. Si la parte de la cláusula where de los métodos Finder contienen algunos alias de base de datos (por ejemplo, T1. o T2.), deben eliminarse estos alias.

Creación de un bean enterprise CMP nuevo

Cuando tiene un atributo nuevo que ha de añadir al modelo de objeto de WebSphere Commerce, puede crear una tabla de base de datos nueva con una columna para el atributo necesario. A continuación, también debe incluir este atributo en un bean enterprise, de modo que los mandatos de WebSphere Commerce puedan acceder a la información.

Una modo de integrar el atributo nuevo en el modelo de objeto de WebSphere Commerce es crear un bean enterprise CMP nuevo. En este bean, deberá crear un campo que se corresponda con el atributo de la tabla de base de datos nueva.

Para crear un bean enterprise CMP nuevo, deberá realizar los siguientes pasos en VisualAge para Java:

1. Asegúrese de que el propietario del área de trabajo esté establecido en WCS Developer.
2. Cree un nuevo grupo EJB para el bean.
3. Cree el nuevo bean enterprise CMP, utilizando la herramienta SmartGuide Crear bean enterprise.
4. Para cada columna de la tabla de base de datos correspondiente, añada un nuevo campo CMP al bean.
5. Si fuera necesario, cree una pareja de métodos get y set para cada uno de los campos CMP creados.
6. Si fuera necesario, defina los campos FinderHelper en la interfaz FinderHelper y añada el nuevo método FinderHelper.
7. Cree un nuevo método ejbCreate, si fuera necesario, y promocione el método ejbCreate a la interfaz inicial del bean enterprise. Este paso es necesario si el nuevo bean enterprise debe crear nuevas entradas en su tabla de base de datos correspondiente.

8. Correlacione los campos del bean enterprise con las columnas de la tabla de base de datos.
9. Genere el bean de acceso y el código de despliegue para el bean enterprise.

Para obtener más detalles sobre cada uno de estos pasos, consulte los apartados siguientes.

Nota: Si debe protegerse el nuevo bean enterprise mediante el sistema de control de acceso de WebSphere Commerce, consulte el Capítulo 4, "Control de acceso" en la página 75 para obtener más detalles. El control de acceso puede añadirse después de crear el bean.

Suponga que tiene una tabla nueva llamada USERRES que especifica determinada información sobre el tipo de hogar de un usuario. Esta tabla contiene tres columnas: una columna USERID, una columna HOME que especifica el tipo de hogar y una columna ROOMS que especifica el número de habitaciones del hogar.

Creación de un grupo EJB nuevo: Al crear beans de entidad nuevos, debe crearlos en un grupo EJB distinto de los grupos EJB de WebSphere Commerce. Cuando trabaja con beans enterprise en VisualAge para Java, debe conmutar a la pestaña EJB del área de trabajo. A continuación, en el menú **EJB**, puede seleccionar **Añadir > Grupo EJB** para iniciar el SmartGuide Añadir grupo EJB.

Hay dos elementos importantes que debe especificar cuando crea el grupo EJB: el proyecto en el que está almacenado el código EJB y el nombre del grupo EJB.

El proyecto EJB puede verse desde la pestaña Proyectos del área de trabajo. Cuando crea un grupo EJB nuevo, debe especificar un proyecto diferente de los proyecto de WebSphere Commerce. Por ejemplo, puede seleccionar que el grupo EJB utilice el proyecto MiEJBPersonalizado. No es necesario que este proyecto exista antes de crear el grupo, ya que VisualAge para Java puede crearlo automáticamente. Este proyecto solamente debe utilizarse para el código EJB; no debe utilizar ningún mandato o código de bean de datos. Esta separación de tipos de código es necesaria debido al despliegue. Si separa su propio código personalizado del código de WebSphere Commerce, se reducirá al mínimo el impacto de la migración en futuros releases.

Tanto con el nombre del proyecto como con el nombre del grupo EJB, asegúrese de que sigue los convenios de denominación correctos para su aplicación.

Creación del bean enterprise CMP nuevo: Para crear el nuevo bean enterprise CMP, puede utilizar la herramienta SmartGuide Crear bean enterprise. Para iniciar la herramienta, pulse con el botón derecho del ratón en el grupo EJB al que desee añadir el bean nuevo y seleccione **Añadir > Bean enterprise**.

Seleccione la creación del nuevo bean enterprise y especifique un nombre para el bean. El convenio de nombres de WebSphere Commerce para los beans enterprise es que para el nombre del bean se utilice el mismo nombre que el de la tabla a la que corresponde el bean. Por ejemplo, si la tabla de base de datos nueva se llama USERRES, el bean enterprise deberá llamarse UserRes.

El campo Proyecto se rellenará automáticamente con el nombre del proyecto. Deberá especificar un nombre de paquete en el proyecto en el que debe almacenarse el código. Un ejemplo del código que se almacenará en el paquete es el código para el bean de acceso que crea para el bean enterprise. Una vez más, al

darle nombre al paquete, asegúrese de que sigue los convenios de nombres adecuados para su aplicación. Por ejemplo, un nombre de paquete puede ser `com.mycompany.mycustombeans`.

En la clase de bean, VisualAge para Java crea el campo privado denominado `EntityContext`. WebSphere Commerce proporciona su propio campo de contexto de entidad en ECEntityBean y su nuevo bean de entidad debe usar este campo, en lugar del campo generado en su propia clase. Por ello, debe eliminar el `EntityContext` generado de su nuevo bean de entidad.

El nuevo bean enterprise debe contener un campo `serialVersionUID`. Si utiliza el SmartGuide para crear su nuevo bean, VisualAge para Java genera este campo automáticamente. Si no utiliza la herramienta para crear su bean, debe añadir este campo.

En el campo de superclase, debe especificar la clase `com.ibm.commerce.base.objects.ECEntityBean`. El siguiente ejemplo de código muestra las funciones proporcionadas por la superclase:

```
public class myEJB extends com.ibm.commerce.base.objects.ECEntityBean {
   public void ejbLoad()throws java.rmi.RemoteException {
      super.ejbLoad();--el supermétodo añadirá el rastreo EJB
       --su lógica --
   }
   public void ejbStore()throws java.rmi.RemoteException {
      super.ejbStore();--el supermétodo añadirá el rastreo EJB
      --su lógica --
   }
}
```

También debe eliminar cualquier método ejbCreate() y ejbPostCreate() que no tome argumentos. Si deja estos métodos en el bean de entidad, pueden producirse errores de ejecución.

Para cada columna de la tabla de base de datos, debe crear un campo CMP nuevo en el bean (debe pulsar Siguiente en SmartGuide para ver la opción Añadir campos CMP al bean). Para cada campo, especifique un nombre de campo y el tipo de datos del campo. Para cualquier columna que sea la clave primaria o que forme parte de la clave primaria, habilite el recuadro de selección Campo de clave. Para todas las demás columnas, habilite el recuadro de selección Promocionar métodos getter y setter en la interfaz remota.

Una vez rellenados todos los campos, pulse Finalizar y VisualAge para Java creará el nuevo bean enterprise CMP.

Creación de nuevos campos FinderHelper en la interfaz *Nombre_Bean*`FinderHelper`: La interfaz *Nombre_Bean*`FinderHelper` contiene cláusulas de búsqueda SQL que corresponden a todos los métodos FinderHelper a excepción del método findByPrimaryKey.

Un nuevo bean enterprise utiliza métodos `findXBy`*NombreArg* (donde *NombreArg* es el nombre de un argumento) definidos en la interfaz FinderHelper del bean junto con los métodos FinderHelper para componer consultas SQL. Utilice el convenio de nombres “findXBy” para el nombre de su campo y así se asegurará de que sus nombres de campos siempre serán exclusivos en relación con los nombres de campo de WebSphere Commerce.

Para crear los nuevos campos FinderHelper en su bean, deberá seleccionar la interfaz *Nombre_Bean*_FinderHelper y modificar el código fuente para establecer cómo se han de componer las sentencias de selección. A continuación, se muestra un ejemplo:

```
public  interface UserResFinderHelper {
 public static final String findXByHomeAndRoomsWhereClause = "(T1.HOME = ?
      and T1.ROOMS = ?)";
}
```

La interfaz local necesitará a continuación un método llamado findXByHomeAndRooms que tome los parámetros de entrada para cada HOME y ROOMS que rellenen los valores representados mediante los caracteres ?. Este tipo de construcción de consulta se denomina *inserción de parámetros*.

Si los parámetros de entrada fuera ″pareada″ y ″3″, la sentencia SQL generada sería

```
select * from USERRES where HOME=pareada y ROOMS=3
```

Por razones de seguridad, al crear métodos FinderHelper para un nuevo bean de entidad, debe utilizar las inserciones de parámetros. El motivo de esta recomendación es que, de esta manera, los usuarios no pueden modificar la consulta. Otro enfoque alternativo sería utilizar una construcción parecida a la siguiente:

```
public static final String
     findXByOwnerIdWhereClause = " (T1.OWNERID = serie_entrada) ";
```

donde *serie_entrada* es el valor de una serie que se ha pasado desde un URL. Esto no es conveniente, ya que un usuario mal intencionado podría entrar un valor, como por ejemplo, “‘123’ OR 1=1” que modificara la sentencia SQL. Si un usuario puede cambiar la sentencia SQL, quizá pueda acceder a los datos, sin tener autorización. Por tanto, es muy recomendable utilizar las inserciones de parámetros.

Si no puede utilizar inserciones de parámetros y, por tanto, tiene que utilizar una serie de entrada para componer la sentencia SQL, debe forzar la comprobación de parámetros en la serie de entrada, para asegurarse de que no es un intento malintencionado de acceso a los datos.

Creación de nuevos métodos FinderHelper en la interfaz *Nombre_Bean*Home: Para cada campo FinderHelper que ha especificado en la interfaz *Nombre_Bean*FinderHelper, debe crear un método FinderHelper en la interfaz local del bean. El nombre de los métodos FinderHelper debe coincidir exactamente con el nombre del campo FinderHelper, con la excepción de que ″WhereClause″ se inhabilita. Es decir, en el nombre del campo de ejemplo de findXByHomeAndRoomsWhereClause, el nombre de método correspondiente es findXByHomeAndRooms.

Para crear los nuevos métodos de FinderHelper, efectúe lo siguiente:

1. Pulse el botón derecho en la interfaz *Nombre_Bean***Home** y seleccione **Añadir > Método**.
   Se abre el SmartGuide Crear método.
2. Seleccione **Crear un nuevo método** y pulse **Siguiente**.
3. En el campo **Nombre del método**, especifique el nombre del método FinderHelper. Este nombre de los métodos FinderHelper debe coincidir exactamente con el nombre del campo FinderHelper, con la excepción de que ″WhereClause″ se inhabilita. Por ejemplo, entre findXByHomeAndRooms.

4. En el campo **Tipo de retorno**, especifique uno de los siguientes:
   - Si el método FinderHelper utiliza la clave primaria para consultar la base de datos y el método debe devolver un registro exclusivo, especifique el objeto EJB como el tipo de retorno. Por ejemplo, entre `UserRes`.
   - Si el método FinderHelper devuelve un conjunto de resultados en lugar de un registro exclusivo, especifique el tipo de retorno como `java.util.Enumeration`.
5. Pulse el botón **Añadir** junto a **¿Qué parámetros debe tener este método?** para añadir los parámetros adecuados. Por ejemplo, puede añadir `argHome` de tipo `Serie` para que contenga el tipo de hogar y `argRooms` de tipo `byte` para que contenga el número de habitaciones.
6. Cuando haya terminado de añadir todos los parámetros, pulse **Siguiente**.
7. Pulse el botón **Añadir** junto a **¿Qué excepciones puede generar este método?** y añada las excepciones siguientes:
   - `java.rmi.RemoteException`
   - `javax.ejb.FinderException`

   Nota: La lista anterior de excepciones muestra el conjunto mínimo de excepciones que debe emitir el método. Dependiendo de su propio código, quizá tenga que especificar otras excepciones.
8. Pulse **Terminar**.

Creación de un método ejbCreate nuevo: Cuando se crea el bean enterprise, se genera automáticamente el método ejbCreate. Este método se promociona a la interfaz remota, de modo que esté disponible en el bean de acceso. El método ejbCreate por omisión, solamente contiene parámetros que son la clave primaria o que forman parte de la clave primaria. Esto significa que al crear una instancia solamente se obtiene una instancia de estos valores.

Si el bean enterprise contiene campos que no forman parte de la clave primaria y que son campos que no pueden contener nulos, debe crear un método ejbCreate nuevo en el que generar una instancia específica de estos campos. De este modo, cada vez que se cree un registro nuevo, todos los campos que no pueden contener nulos se rellenarán con los datos correctos.

Para crear un método ejbCreate nuevo, efectúe lo siguiente:

1. En el panel Tipos, expanda la clase ***Nombre_Bean*Bean**. Por ejemplo, seleccione **UserResBean**.
2. Pulse el método ejbCreate existente para ver el código fuente. (Tenga en cuenta que puede ser ejbCreate(String), ejbCreate(String, int) o que puede aceptar otros parámetros de entrada, dependiendo de la clave primaria de su bean enterprise.)
3. Debe modificar el código fuente de modo que cada campo CMP se incluya como un parámetro de entrada en el método y se genere una instancia de cada campo CMP con el valor correcto. En el ejemplo UserRes, en el que UserId es la clave primaria, el código fuente aparece inicialmente como:

```
public void ejbCreate(int argUserId)
   throws javax.ejb.CreateException, java.rmi.RemoteExeption {
   _initLinks();
   userId = argUserId;
}
```

Pero es posible que desee asegurarse de que se inicialicen tanto el número de habitaciones como el tipo de hogar. En este caso, deberá cambiar el código del modo siguiente:

```
public void ejbCreate(int argUserId, String argHome, byte Rooms)
   throws javax.ejb.CreateException, java.rmi.RemoteExeption {
   _initLinks();
   // Todos los campos CMP deben inicializarse aquí
   userId = argUserId;
   home = argHome;
   rooms = argRooms;
}
```

Nota: Si desea utilizar una clave primaria generada por el sistema, consulte "Claves primarias" en la página 67 para obtener detalles.

4. Guarde el método modificado. Cuando guarda el código, VisualAge para Java crea un método ejbCreate nuevo que toma los parámetros nuevos. El método ejbCreate original permanece.
5. Suprima el método ejbCreate original (el que no tiene argumentos).
6. Pulse con el botón derecho del ratón en el método ejbCreate nuevo y seleccione **Añadir > Interfaz local EJB**.
7. Cree un método ejbPostCreate correspondiente. (No es necesario añadir este método a la interfaz local.)

Correlación de la tabla de base de datos con el bean enterprise nuevo: Cuando ha creado el nuevo bean enterprise, debe crear una correlación entre los campos CMP del bean y las columnas de la tabla de base de datos. Esta correlación se denomina un esquema. VisualAge para Java proporciona herramientas que simplifican esta tarea.

Para crear el esquema, efectúe lo siguiente:

1. En el menú **EJB**, seleccione **Abrir en > Esquemas de base de datos**. Se abrirá el Examinador de esquemas.
2. En el menú **Esquemas**, seleccione **Importar/Exportar esquema > Importar esquema de base de datos**.
3. Entre un nombre para el esquema nuevo y pulse **Aceptar**.
4. En la ventana Conexión de base de datos, entre la información siguiente:

Atributo	Valor DB2	Valor Oracle
Tipo de conexión	`COM.ibm.db2.jdbc.app.DB2Driver`	`Oracle.jdbc.driver.OracleDriver`
Origen de datos	`jdbc:db2:nombre_basedatos_wc`	`jdbc:oracle:thin@`*`nombsistpral`*`:`*`puerto`*`:`*`SID_Oracle`*
Nombre de usuario	*`nombre_usuario_bd_wc`*	*`nombre_usuario_bd_wc`*
Contraseña	*`contraseña_bd_wc`*	*`contraseña_bd_wc`*

sustituyendo los valores de la forma siguiente:

- DB2 *`nombre_base_datos_wc`* es el nombre de la base de datos de WebSphere Commerce
- Oracle *`nombsistpral`* es el nombre de sistema principal del servidor Oracle.
- Oracle *`puerto`* es el número de puerto de la base de datos Oracle.

- Oracle *SID_Oracle* es el ID de instancia Oracle.
- *nombre_usuario_bd_wc* es el nombre de usuario para la base de datos.
- *contraseña_bd_wc* es la contraseña de la base de datos.

5. En la lista **Calificadores**, seleccione el calificador de su base de datos (puede ser su nombre de usuario de base de datos).
6. Pulse **Crear lista de tablas**.
7. Seleccione *su_tabla_nueva* (de la lista generada) y pulse Aceptar para generar el esquema.
8. Una vez generado el esquema, pulse *su_tabla_nueva* en el panel Esquema.
9. Resalte y luego pulse dos veces *su_tabla_nueva* en el panel Tabla. Aparecerá la ventana Editor de tablas.
10. Suprima la información que haya en el campo **Calificador**. Esto hará que el bean enterprise se pueda portar a otras máquinas.
11. En el menú **Esquemas**, seleccione **Guardar esquema**. Entre los nombres de proyecto, paquete y clase correctos.

A continuación, deberá crear la correlación de esquemas. La correlación de esquemas relaciona las columnas de base de datos y los campos en el bean enterprise.

Para crear la correlación de esquemas, efectúe lo siguiente:

1. En el menú **EJB**, seleccione **Abrir en > Correlaciones de esquemas**. Se abrirá la ventana Correlación de almacén de datos.
2. Entre el nombre de la correlación. Consulte “Consideraciones sobre los nombres de los objetos del esquema de base de datos” en la página 69 para obtener recomendaciones sobre la denominación de la nueva correlación.
3. Seleccione el esquema y el grupo EJB. Consulte “Consideraciones sobre los nombres de los objetos del esquema de base de datos” en la página 69 para obtener recomendaciones sobre la denominación del nuevo esquema.
4. En el panel Correlaciones de almacén de datos, seleccione su correlación.
5. En el panel Clases permanentes, seleccione su clase.
6. En el menú **Correlaciones de tablas**, selecione **Nueva correlación de tablas > Añadir correlación de tabla sin Herencia**.
7. En la lista desplegable **Tabla**, seleccione su tabla y pulse **Aceptar**.
8. En el panel Correlaciones de tablas, resalte y, a continuación, pulse con el botón derecho la correlación de tablas y seleccione **Editar correlaciones de propiedades**. Se abrirá el Editor de correlaciones de propiedades.
9. Para cada uno de los atributos de clase (los campos CMP del bean) debe especificar el tipo de correlación y la columna de base de datos con la que debe correlacionarse. Por ejemplo, debe correlacionar el atributo de clase UserId utilizando una correlación de tipo ″Simple″ con la columna USERID de la tabla de base de datos.
10. Una vez correlacionados todos los campos con su columna de base de datos correspondiente deberá guardar la correlación de esquemas. En el menú **Correlación de base de datos**, seleccione **Guardar correlación de almacenes de datos**. Entre los nombres de proyecto, paquete y clase correctos para guardar la correlación. Pulse **Terminar**.

Creación del bean de acceso y generación del código desplegado: Un bean de acceso actúa como una envoltura del bean enterprise que simplifica el modo en que otros componentes interactúan con el bean enterprise. Deberá crear un bean de acceso para su bean enterprise.

Al generar código desplegado, las herramientas de VisualAge para Java analizan el bean para asegurar que se satisfacen las normas de las especificaciones Sun Microsystems EJB, así como las normas específicas del servidor EJB.

Para crear un bean de acceso para su nuevo bean enterprise, efectúe lo siguiente:

1. En el panel Beans enterprise, pulse el botón derecho en su bean enterprise nuevo y seleccione **Añadir > Bean de acceso**. (Es posible que haya ampliado en primer lugar el grupo EJB que contiene su nuevo bean, para poder ver el bean.) Se abre el SmartGuide Crear bean de acceso.
2. En los campos **Grupo EJB**, **Bean enterprise** y **Nombre de bean de acceso**, especifique el grupo EJB, el nombre de bean y el nombre de bean de acceso adecuados.
3. Seleccione **CopyHelper para bean de entidad como Tipo de bean de acceso** y pulse **Siguiente**.
4. En la lista desplegable **Seleccionar método local para constructor sin argumentos**, seleccione **findByPrimaryKey**.
5. En la lista desplegable **Convertidor**, seleccione **WCSStringConverter** para las propiedades iniciales y pulse **Siguiente**.
6. En la ventana Seleccionar y personalizar propiedades del bean para CopyHelper, seleccione **WCSStringConverter** para cada campo.
7. Pulse **Terminar**.

Para generar el código desplegado, efectúe lo siguiente:

1. En el panel Beans enterprise, pulse con el botón derecho el bean enterprise nuevo y seleccione **Generar código desplegado**.

Observe que el código desplegado que se genera utilizando esta herramienta satisface la especificación EJB 1.0 y sólo se utiliza al ejecutar el bean enterprise dentro de VisualAge para Java. En una fase posterior, cuando despliegue el bean enterprise en una aplicación de WebSphere Commerce ejecutándose dentro de WebSphere Application Server V4.0, se le solicitará que genere un archivo JAR que contenga código desplegado que satisfaga las especificaciones EJB 1.1. Para obtener más información sobre la creación de este archivo JAR Export de EJB 1.1, consulte “Información sobre el código desplegado de EJB” en la página 159 y “Generación de código desplegado” en la página 292.

Utilización del cliente de prueba para probar el bean enterprise: VisualAge para Java proporciona un cliente de prueba que se puede utilizar para probar beans enterprise. Para utilizar el cliente de prueba y comprobar el nuevo bean, efectúe lo siguiente:

1. Inicie el servidor EJB que contiene el bean enterprise que va a probar.
2. Pulse con el botón derecho del ratón en el bean enterprise y seleccione **Ejecutar cliente de prueba**.
   Se abre las ventanas Cliente de prueba EJB y Búsqueda de EJB.
3. En el campo **Nombre JNDI**, entre el nombre JNDI del bean enterprise y pulse **Buscar**.
4. Pulse con el botón derecho del ratón en el método **findByPrimaryKey** que tiene los argumentos llenos y seleccione **Invocar**.

Codificación: Deben tenerse en cuenta las siguientes normas de codificación en los bean enterprise:

- No utilice el tipo de datos BLOB ni CLOB.
- Ningún campo CMP del tipo de datos LONG (también denominado LONG VARCHAR) debe ser el primer o último miembro en la lista de campos CMP para el bean enterprise. Para verificar la lista, compruebe el método `EJSJDBCPersister._hydrate()` y observe si el primer o último elemento de la lista es del tipo LONG VARCHAR. Si el primer campo es de este tipo, haga lo siguiente:
  1. Desactive el primer campo. Denomínelo `campoA`.
  2. Desactive otro campo que *no* sea del tipo LONG VARCHAR. Denomínelo `campoB`.
  3. Restablezca el `campoA`.
  4. Restablezca el `campoB`.
  5. Abra el examinador de correlaciones de esquemas y edite la correlación de tablas para que reflejen estos cambios. Guarde la correlación.
  6. Vuelva a generar el código desplegado.
- El código de beans enterprise no debe hacer referencia a objetos externos de los paquetes de beans enterprise. Por ejemplo, no debe hacer referencia a mandatos ni beans de datos
- Para habilitar el control de acceso para su bean enterprise, añada la interfaz com.ibm.commerce.security.Protectable y/o la interfaz com.ibm.commerce.security.Groupable a la interfaz remota del bean enterprise. Después de añadir estas interfaces, vuelva a generar el código desplegado del bean y el bean de acceso. También debe crear un objeto de clase de ayuda de acceso en el paquete objsrc.

Creación de un bean de datos sencillo

Un bean de datos es un bean que se utiliza en las plantillas JSP para recuperar información del bean enterprise. Un bean de datos sencillo amplía su bean de acceso correspondiente e implementa la interfaz SmartDataBean. La mayor parte del código del bean de datos lo genera automáticamente VisualAge para Java.

Para crear un bean de datos sencillo, deberá realizar los pasos siguientes:

1. Cree un proyecto y un paquete para almacenar el código del bean de datos.
2. Cree un bean de datos que amplíe el bean de acceso correspondiente y que implemente la interfaz de bean de datos adecuada.
3. Cree los métodos set para el bean de datos.
4. Cree los métodos get para el bean de datos.

Creación del proyecto y del paquete para el código del bean de datos: Al crear un proyecto y un paquete se crea un espacio en el que se puede almacenar su código de bean de datos.

Para crear un nuevo proyecto, haga lo siguiente:

1. Seleccione la pestaña **Proyectos**.
2. En el menú **Seleccionado**, seleccione **Añadir > Proyecto**.
   Aparece el SmartGuide Añadir proyecto.
3. Compruebe que **Crear un nuevo proyecto denominado** esté seleccionado y entre el nombre de su nuevo proyecto. Por ejemplo, entre `Mis beans de datos`.
4. Pulse **Terminar**.

Para crear un nuevo paquete, haga lo siguiente:

1. Pulse con el botón derecho del ratón en el proyecto que ha creado para el código del bean de datos y seleccione **Añadir > Paquete**. Por ejemplo, pulse con el botón derecho del ratón en **Mis beans de datos** y seleccione **Añadir > Paquete**.
   Se abre el SmartGuide Añadir paquete.
2. Compruebe que **Crear un nuevo paquete denominado** esté seleccionado y entre un nombre apropiado para su paquete de bean de datos. Por ejemplo, entre `com.mycompany.mydatabeans`.
3. Pulse **Terminar**.

Creación de un bean de datos: Un bean de datos es un bean Java que se utiliza en una plantilla JSP para proporcionar contenido dinámico a la página. Normalmente proporciona una representación sencilla (indirectamente) de un bean de entidad ampliando un bean de acceso. El bean de datos encapsula las propiedades que se pueden recuperar o establecer en el bean de entidad.

Para crear un bean de datos, efectúe lo siguiente:

1. Pulse con el botón derecho del ratón en el paquete en el que desea almacenar el bean de datos y seleccione **Añadir > Clase**.
   Se abre el SmartGuide Crear clase.
2. Los campos de nombre de proyecto y paquete ya están rellenados.
3. Asegúrese de que **Crear una nueva clase** esté seleccionado y pulse **Siguiente**.
4. En el campo **Nombre de clase**, entre un nombre para el bean de datos nuevo. Por ejemplo, para crear un bean de datos que amplíe UserResAccessBean, entre `UserResDataBean`.
5. Para especificar la superclase, pulse **Examinar**, luego en el campo de patrón, entre el nombre del bean de acceso correspondiente. Por ejemplo, entre `UserResAccessBean` y pulse **Aceptar**.
6. Pulse **Siguiente**.
7. Para especificar las interfaces que debe implementar el bean de datos, pulse **Añadir**. En la ventana Interfaz, haga lo siguiente:
   a. En el campo **Patrón**, entre `com.ibm.commerce.beans.SmartDataBean` y, a continuación, pulse **Añadir**.
   b. En el campo **Patrón**, entre `com.ibm.commerce.beans.InputDataBean` y, a continuación, pulse **Añadir**.
   c. Pulse **Cerrar**.
8. Pulse **Terminar**.

Adición de campos obligatorios al bean de datos: Esta sección describe cómo añadir campos obligatorios a su nuevo bean de datos.

Para añadir el campo iCommandContext, haga lo siguiente:

1. Pulse con el botón derecho del ratón en el nuevo bean de datos (por ejemplo, UserResDataBean) y seleccione Añadir > Campo.
   Se abre el SmartGuide Crear campo.
2. En el campo **Nombre de campo**, entre `iCommandContext`.
3. Pulse **Examinar** para añadir el tipo de campo y entre `com.ibm.commerce.command.CommandContext`. Pulse **Aceptar**.
4. Para los modificadores de acceso, seleccione **Protegido**.
5. Pulse **Terminar**.

Para añadir el campo iRequestProperties, haga lo siguiente:

1. Pulse con el botón derecho del ratón en el nuevo bean de datos (por ejemplo, UserResDataBean) y seleccione Añadir > Campo.
   Se abre el SmartGuide Crear campo.
2. En el campo **Nombre de campo**, entre `iRequestProperties`.
3. Pulse **Examinar** para añadir el tipo de campo y entre `com.ibm.commerce.datatype.TypedProperty`. Pulse **Aceptar**.
4. Para los modificadores de acceso, seleccione **Protegido**.
5. Pulse **Terminar**.

Modificación de los métodos set del bean de datos: Después de crear su bean de datos, debe modificar el código en algunos de los métodos set generados.

Para actualizar los métodos set, haga lo siguiente:

1. Expanda su nuevo bean de datos para ver sus campos y métodos.
2. Seleccione el método **`setCommandContext(CommandContext)`** para ver el código fuente.
   El panel Fuente muestra el código fuente de la forma siguiente:

```
public void setCommandContext(com.ibm.commerce.comand.CommandContext arg1)
 {
 }
```

3. Modifique el código de forma que el método quede así:

```
public void setCommandContext(com.ibm.commerce.comand.CommandContext arg1)
{
   iCommandContext = arg1;
}
```

   Guarde el trabajo (Control+S).
4. Seleccione el método **setRequestProperties(TypedProperty)** para ver el código fuente.
   El panel Fuente muestra el código fuente de la forma siguiente:

```
public void setRequestProperties(
 com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception
   {
   }
```

5. Quizá desee modificar el código fuente para rellenar la clave primaria del bean de acceso correspondiente. La forma recomendada para hacerlo es utilizar el gestor de bean de datos para establecer indirectamente este valor. Este método indirecto está pensado para asegurar que un valor de clave primaria tomado de las propiedades del URL no modificará la clave primaria, si ésta se ha establecido anteriormente. Para que su método setRequestProperties siga este modelo, codifíquelo de forma parecida a la siguiente sección de código. Observe que en el ejemplo siguiente, la clave primaria es el id de usuario. Esto puede variar, según la situación (así, el siguiente código quizá no se compile inmediatamente en su aplicación).

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception
   {
      iRequestProperties = arg1;
      try {
          if (// comprobar si hay nulos
              getDataBeanKeyUserId() == null)
             {
                super.setInitKey_UserId(aUserId);
```

```
            }
        } catch (com.ibm.commerce.exception.ParameterNotFoundException e)
             {
             }
    }
```

Hay dos maneras adicionales de establecer la clave primaria para el bean de acceso. Se puede hacer externamente desde el bean de datos, por ejemplo en la plantilla JSP. En ese caso, antes de activar el bean de datos en la plantilla JSP, llame de forma explícita al método set del bean de datos para la clave primaria. Por ejemplo, JSP podría incluir código parecido al siguiente (donde db es el objeto de bean de datos):

```
db.setInitKey_UserId(/*parámetro de entrada*/)
db.activate();
```

De forma alternativa, la clave primaria puede establecerse de forma directa. Es decir, la plantilla JSP sólo contiene el método db.activate y, a continuación, el gestor de beans de datos establece de forma explícita la clave primaria en el bean de acceso. Por ejemplo, el código para el método setRequestProperties de los beans de datos se parecería a éste:

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty arg1)
   throws Exception
      {
         iRequestProperties = arg1;
         try
             {
                super.setInitKey_UserId(aUserId);
             }
       } catch (com.ibm.commerce.exception.ParameterNotFoundException e)
           {
           }
      }
```

Observe que el procedimiento recomendado para establecer la clave primaria es el método indirecto, que se muestra en el paso 5.

Modificación de los métodos get del bean de datos: Después de crear su bean de datos, debe modificar el código en algunos de los métodos get generados.

Para actualizar los métodos get, haga lo siguiente:

1. Expanda su nuevo bean de datos para ver sus campos y métodos.
2. Seleccione el método **getCommandContext()** para ver el código fuente. El panel Fuente muestra el código fuente de la forma siguiente:

   ```
   public com.ibm.commerce.comand.CommandContext getCommandContext ()
   {
      return null;
   }
   ```

3. Modifique el código de forma que el método quede así:

   ```
   public com.ibm.commerce.comand.CommandContext getCommandContext ()
   {
      return iCommandContext;
   }
   ```

   Guarde el trabajo (Control+S).
4. Seleccione el método **getRequestProperties()** para ver el código fuente. El panel Fuente muestra el código fuente de la forma siguiente:

```
public com.ibm.commerce.datatype.TypedProperty setRequestProperties()
{
   return null;
}
```

5. Modifique el código fuente para que quede así:

```
public com.ibm.commerce.datatype.TypedProperty setRequestProperties()
{
   return iRequestProperties;
}
```

Guarde el trabajo (Control+S).

Modificación del método populate(): Debe modificar el método populate, haciendo lo siguiente:

1. Expanda su nuevo bean de datos para ver sus campos y métodos.
2. Seleccione el método **`populate()`** para ver el código fuente.
   El panel Fuente muestra el código fuente de la forma siguiente:

```
public void populate () throws Exception {}
```

3. Modifique el código de forma que el método quede así:

```
public void populate() throws Exception {
   super.refreshCopyHelper();
}
```

Guarde el trabajo (Control+S).

Creación de nuevos beans de sesión

Al crear beans de sesión nuevos, debe crearlos en un grupo EJB distinto de los grupos EJB de WebSphere Commerce. Guarde este nuevo grupo EJB en un nuevo proyecto separado de los proyectos de WebSphere Commerce. Por ejemplo, puede crear el grupo EJB `MyCustomBeans` dentro del paquete `com.mycompany.mycustomcode`. Si separa su propio código personalizado del código de WebSphere Commerce, se reducirá al mínimo el impacto de la migración en futuros releases.

El nuevo bean de sesión debe ampliar la clase com.ibm.commerce.base.helpers.BaseJDBCHelper. La superclase proporciona métodos que permiten obtener un objeto de conexión JDBC del objeto origen de datos utilizado por el servidor de comercio, de forma que el bean de sesión tome parte en la misma transacción que los demás beans de entidad. A continuación se muestra un ejemplo de código que muestra las funciones proporcionadas por la superclase:

```
public class mySessionBean extends com.ibm.commerce.base.helpers.BaseJDBCHelper
   implements SessionBean {

   public Object myMethod () throws javax.naming.NamingException,
      java.rmi.RemoteException, SQLException {

       ////////////////////////////////////////////////
      // su código, como por ejemplo, inicialización //
      ////////////////////////////////////////////////

         try {
             // obtener una conexión del origen de datos de WebSphere Commerce
             makeConnection();
             PreparedStatement stmt = getPreparedStatement(
                "su serie sql");
             ////////////////////////////////////////////////////////////
             // su código, como por ejemplo, parámetro set en sentencia//
             // preparada --                                           //
             ////////////////////////////////////////////////////////////
```

```
            ResultSet rs = executeQuery(stmt, false);

            ////////////////////////////////////////////////
            // -- su código para procesar ResultSet    -- //
            ////////////////////////////////////////////////

                }
            finally {
               // devolver la conexión a la fuente de datos WebSphere Commerce
              closeConnection();
            }

         ////////////////////////////////////////////////////
        // -- su código para devolver el resultado ---      //
        ////////////////////////////////////////////////////

    }

}
```

En el ejemplo de código anterior, el método `executeQuery` tiene dos parámetros de entrada. El primero es una sentencia preparada y el segundo es un distintivo booleano relacionado con una operación de borrado de antememoria. Establezca este distintivo en `true` si necesita que el contenedor vacíe todos los objetos de entidad para la transacción actual de la antememoria, antes de ejecutar la consulta. Es necesaria esta operación si ha efectuado actualizaciones en algunos objetos de entidad y necesita que la consulta busque en estos objetos actualizados. Si el indicador se establece en `false`, estas actualizaciones de los objetos de entidad no se grabarían en la base de datos hasta el final de la transacción.

Debe limitar el uso de este distintivo y establecerlo, normalmente, en `false`, excepto en los casos en que sea realmente necesario. La operación de vaciado es una operación de uso intensivo de recursos.

Ciclos de vida de los objetos

Los beans enterprise del modelo de objeto incluyen tanto objetos *independientes* como objetos *dependientes*. Un objeto independiente tiene su propio ciclo de vida, controlado directamente por las peticiones de creación o eliminación de la lógica de negocio que invoca al objeto. Un objeto dependiente tiene un ciclo de vida vinculado a otro objeto, denominado *objeto propietario* (que, a su vez, puede ser también un objeto dependiente, aunque si se sigue la jerarquía de asociaciones se encontrará un objeto independiente). Cuando se suprime el objeto propietario, todos los objetos dependientes se suprimen también. Las supresiones reales se controlan mediante especificaciones de supresión en cascada en la base de datos.

Por ejemplo, supongamos que un objeto usuario devuelve un objeto listín de direcciones y una lista de objetos pedido; si se suprime el objeto usuario, se suprimirá también su objeto listín (puesto que es propiedad del usuario), así como todos los objetos dirección del listín (puesto que son propiedad del listín). Sin embargo, los objetos pedido no se suprimirán ya que el propietario de los pedidos es el objeto tienda, no el objeto usuario.

Para la creación de objetos dependientes se utiliza un patrón de diseño específico. El método de creación de un objeto dependiente debe suministrar una referencia a su objeto propietario; por lo tanto, el objeto propietario ya debe existir para poder crear el objeto dependiente.

Transacciones

La arquitectura de Enterprise JavaBeans V1.0 especifica tres opciones alternativas en el momento del compromiso con respecto al estado de la instancia. Se describen como opciones A, B y C en el documento de especificaciones. Para obtener más información sobre estas opciones, consulte el documento de especificaciones de Enterprise JavaBeans V1.0 de Sun Microsystems.

Aunque WebSphere Application Server implementa las opciones A y C, la opción A presupone que la base de datos no es compartida.

En la opción C, el contenedor de beans enterprise no almacena en la antememoria una instancia "preparada" entre transacciones. Tan pronto una transacción finaliza, la instancia se devuelve a la agrupación de instancias disponibles. WebSphere Commerce utiliza la opción C porque la base de datos se comparte a través de varias aplicaciones de WebSphere Commerce. En esta implementación, el contenedor carga datos persistentes para beans de entidad al principio de cada transacción y los beans de entidad sólo se almacenan en antememoria durante el tiempo que dura la transacción. El contenedor activa varias instancias de un bean de entidad, una para cada transacción en la se accede a la entidad. La sincronización de las transacciones la lleva a cabo la base de datos.

El atributo de transacción de cada bean enterprise se establece en `TX_REQUIRED`. Puesto que el controlador Web inicia una transacción antes de ejecutar un mandato que accede a un bean enterprise (mediante su correspondiente bean de acceso), los métodos de negocio del bean enterprise se invocan dentro del contexto de esta transacción.

Otras consideraciones sobre los beans de entidad

Find for Update

Cuando múltiples aplicaciones pueden acceder a la misma fila de una base de datos para actualizar esa fila, esta situación se denomina *actualización simultánea*. Hay casos en los que pueden permitirse las actualizaciones simultáneas y otros en los que no son deseables en absoluto.

Si la actualización de la base de datos se graba encima, y el nuevo valor no tiene relación con el valor anterior, pueden permitirse las actualizaciones simultáneas. Si se permite la actualización simultánea y varias aplicaciones intentan actualizar la misma fila en una base de datos, el último intento es el que se actualiza en la base de datos.

Si la actualización de la base de datos depende del valor actual, no es deseable la actualización simultánea. Por ejemplo, si una aplicación está actualizando un inventario de productos, debe permitirse que sólo una aplicación pueda actualizar el inventario a la vez.

Los factores que influyen sobre si se permite o no la actualización simultánea incluyen los bloqueos de base de datos y los niveles de aislamiento de bean enterprise.

Para evitar que una segunda aplicación actualice una fila simultáneamente, la primera aplicación que accede a la fila debe buscar la fila utilizando la opción "find for update". Cuando se utiliza la opción "for update", se aplica un *bloqueo de*

grabación (también denominado bloqueo exclusivo) a la fila. Con este bloqueo de grabación aplicado a la fila, se bloquea cualquier aplicación que intente acceder a la fila utilizando “find for update”.

Si la aplicación permite actualizaciones simultáneas, puede ir a buscar los datos, sin bloquear la fila.

Considere el escenario OrderProcess, en el que UpdateInventory necesita encontrar todos los productos incluidos en un pedido y actualizar el inventario según corresponda. Puesto que los mismos productos pueden estar incluidos en muchos otros pedidos, debe utilizarse *find for update* y debe hacerse lo antes posible dentro del ámbito de la transacción para disminuir la posibilidad de que se produzcan puntos muertos. Por lo tanto, el algoritmo UpdateInventory puede representarse mediante el siguiente pseudocódigo:

```
UpdateInventory
 buscar todos los artículos del pedido
 para cada artículo de pedido
  buscar su inventario utilizando "find for update"
  ...
```

En un escenario de empresa a empresa de larga ejecución en el que un pedido puede tener muchos artículos, debe utilizarse ″find for update″ tan pronto como sea posible. La lógica puede ser la siguiente:

```
find for update el inventario de todos los productos de un pedido
para cada producto
 if (cantidad total pedida de ese producto < inventario)
        restar cantidad del inventario
 else
        error
```

Método de vaciado remoto

Puesto que WebSphere Application Server no graba en la base de datos los cambios efectuados en los beans de entidad hasta el momento en que se comprometen las transacciones, es posible que la base de datos esté temporalmente sin sincronizar con los datos almacenados en antememoria del contenedor del bean de entidad.

Se proporciona un método de vaciado remoto (en la clase `com.ibm.commerce.base.helpers.BaseJDBCHelper`) que graba todos los cambios comprometidos realizados en todas las transacciones (es decir, obtiene información de la antememoria de beans enterprise) y actualiza la base de datos. Este método remoto puede ser llamado por un mandato. Utilice este método, sólo cuando sea totalmente necesario, ya que utiliza muchos recursos y tiene un impacto negativo en el rendimiento.

Suponga el caso de un mandato de conexión que tiene la siguiente sección de código:

```
UserAccessBean uab = ...;
 uab.setRegisteredTimestamp(currentTimestamp);
 uab.commitCopyHelper();
```

Antes de que la transacción se haya comprometido, la columna REGISTEREDSTAMP de la tabla USER no estará actualizada con la indicación de la hora actual. La actualización sólo se produce cuando se compromete la transacción. El método de vaciado tiene que utilizarse de modo que todas las consultas JDBC directas (en la misma transacción), por ejemplo, *select from user where registeredstamp ...* devuelvan el usuario con la indicación de la hora de registro especificada.

Proteger los beans enterprise

Si utiliza WebSphere Application Server para proteger los beans enterprise, debe asignar los métodos de los beans enterprise nuevos al grupo de métodos de seguridad `WCSMethodGroup`, utilizando la Consola de administración de WebSphere Application Server. Efectúe este paso al desplegar los beans enterprise nuevos. Además, si modifica beans de entidad de WebSphere Commerce existentes, debe asignar los métodos de todos los beans de entidad en el grupo EJB afectado al grupo de métodos de seguridad `WCSMethodGroup`. Para obtener una descripción del proceso de despliegue del código personalizado, consulte "Despliegue del código" en la página 159.

Claves primarias

Una clave primaria es una clave exclusiva que forma parte de la definición de una tabla. Se puede utilizar para diferenciar un registro de otros. Todos los registros deben tener una clave primaria. Cuando se crea un registro nuevo en una tabla, es posible que necesite generar una clave primaria exclusiva para el registro.

En el modelo de programación WebSphere Commerce, la capa de persistencia incluye los beans de entidad que interactúan con la base de datos. De este modo, es posible crear registros de base de datos cuando se crea una instancia de un bean de entidad. Por lo tanto, es posible que el método ejbCreate para crear una instancia de un bean de entidad deba incluir la lógica necesaria para generar una clave primaria para los registros nuevos.

Cuando una aplicación requiere información de la base de datos, indirectamente utiliza los beans de entidad creando una instancia del bean de acceso correspondiente al bean y ejecutando get o set en los diferentes campos. Se crea una instancia de un bean de acceso para un registro determinado de una base de datos (por ejemplo, para un perfil de usuario determinado) y se utiliza la clave primaria para seleccionar la información correcta de la base de datos.

Las secciones siguientes describen cómo crear una clave primaria exclusiva y cómo realizar la selección por clave primaria.

Creación de claves primarias: El método `ejbCreate` se utiliza para crear una instancia de las nuevas instancias de un bean de entidad. Este método se genera automáticamente pero el método generado no incluye la lógica necesaria para inicializar las claves primarias en un valor estático.

Es posible que necesite asegurarse de que el valor de la clave primaria sea un valor nuevo y exclusivo. En este caso, es posible que tenga un método ejbCreate similar al siguiente extracto de código:

```
Public void ejbCreate(int argMyOtherValue)
   throws javax.ejb.CreateException,
   java.rmi.RemoteException {
      //Inicializar campos CMP
      MyKeyValue = com.ibm.commerce.key.ECKeyManager.
         singleton().getNextKey("nombre_tabla");
      MyOtherValue = argMyOtherValue;
}
```

En el extracto de código anterior, el método getNextKey genera un entero exclusivo para la clave primaria. El parámetro de entrada *nombre_tabla* del método debe coincidir exactamente con el valor de TABLENAME definido en la tabla KEYS. Asegúrese de que los caracteres y las mayúsculas/minúsculas coincidan con exactitud.

Además de incluir el código anterior en el método ejbCreate, también debe crear una entrada en la tabla KEYS. A continuación se muestra una sentencia SQL de ejemplo para efectuar la entrada en la tabla KEYS:

```
insert into KEYS (TABLENAME, COUNTER, KEYS_ID)
   values ("nombre_tabla", 0, 1)
```

Tenga en cuenta que con la sentencia SQL anterior se aceptan valores por omisión para las otras columnas de la tabla KEYS. El valor para COUNTER indica el valor en el que se debe iniciar la cuenta. El valor para KEYS_ID debe ser cualquier valor positivo.

Si su clave primaria se ha definido como un tipo de datos long (BIGINT para DB2 o NUMBER para Oracle), utilice el método `getNextKeyAsLong`.

Selección por clave primaria: En un bean de acceso, debe seleccionar el registro de base de datos adecuado utilizando la clave primaria. La sección de código siguiente muestra cómo efectuar esta selección. También incluye código adicional, que se explica posteriormente.

```
UserProfileAccessBean abUserProfile = new UserProfileAccessBean();
abUserProfile.setInitKey_UserId(getUserId().toString());
abUserProfile.refreshCopyHelper();
```

La primera línea del extracto de código anterior crea una instancia de un UserProfileAccessBean nuevo que se llama ″abUserProfile″. La segunda línea establece la clave primaria en el bean de acceso. VisualAge para Java utiliza el convenio de nombres `setInitKey_`*xxx* (donde *xxx* es el nombre del campo de clave primaria) para los nombres de los métodos set de las claves primarias. Cuando se crea una instancia de un bean de acceso, es necesario asegurarse de que todos los campos establecidos por un método setInitKey_xxx se hayan inicializado antes de utilizar el método refreshCopyHelper. El orden en que se llama a los métodos setInitKey_xxx no es importante.

Una vez se ha llamado a todos los métodos setInitKey_xxx, es necesario inicializar todos los campos necesarios y se puede utilizar el método refreshCopyHelper para recuperar la información de la base de datos.

Si se actualizan los valores de la antememoria local del bean de acceso, deberá incluir también una llamada commitCopyHelper para actualizar la base de datos con la información actualizada. Por ejemplo, si después de recuperar datos utilizando el método refreshCopyHelper actualiza el nombre de un cliente (estableciendo el valor de nombre), debe llamar a `abUserProfile.commitCopyHelper()` para actualizar la base de datos con la nueva información.

Utilización de los beans de entidad

Un programa que utilice beans enterprise debe tratar con la interfaz Java Naming and Directory Interface (JNDI) así como con las interfaces locales y remotas de los beans enterprise. Para simplificar el modelo de programación, se genera un bean de acceso para cada bean enterprise. Al crear sus propios beans enterprise, utilice las herramientas de VisualAge para Java para generar este bean de acceso.

Los mandatos de WebSphere Commerce interactúan con beans de acceso en vez de hacerlo directamente con los beans de entidad. Tal como muestra el diagrama, si se utiliza el bean de acceso se obtendrán las siguientes ventajas:

- Una interfaz de programación más sencilla. El bean de acceso tiene un comportamiento parecido a un bean Java y oculta a los clientes todas las interfaces de programación específicas de los beans enterprise como, por ejemplo, JNDI y las interfaces local y remota
- Durante la ejecución, el bean de acceso almacena en antememoria el objeto local de bean enterprise porque las búsquedas en el objeto local son muy costosas en términos de tiempo y utilización de recursos.
- El bean de acceso implementa un objeto copyHelper que reduce el número de llamadas al bean enterprise cuando los mandatos obtienen y establecen atributos de bean enterprise. Por lo tanto, al leer o grabar varios atributos de bean enterprise, sólo es necesaria una única llamada al bean enterprise.

En el siguiente diagrama se muestra la interacción entre mandatos, beans de acceso, beans de entidad y la base de datos.

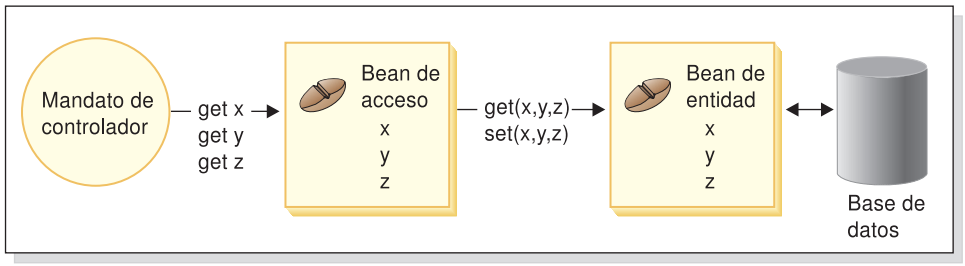

Figura 18.

Consideraciones sobre la base de datos

Al personalizar su aplicación de comercio electrónico, puede crear nuevas tablas de bases de datos. Al crear estas tablas, se recomienda seguir un conjunto de convenios, de forma que las tablas se creen de forma coherente con las tablas de WebSphere Commerce.

Consideraciones sobre los nombres de los objetos del esquema de base de datos

Las secciones siguientes proporcionan una guía para la denominación de los objetos del esquema de base de datos.

Convenios de denominación para tablas y vistas

La lista siguiente proporciona una guía para la denominación de nuevas tablas y vistas:

- Para evitar conflictos con los nombres (nombres duplicados) con las tablas y vistas de WebSphere Commerce en releases futuros, el primer carácter del nombre de la tabla o la vista debe ser una `X`. Por ejemplo, `XMYTABLE`.
- El nombre de la tabla o la vista no debe tener más de 10 caracteres de longitud. Si el nombre que desea sobrepasa este límite, hágalo más corto eliminando las vocales a partir del final del nombre, hasta que sólo queden 10 caracteres.
- El nombre de la tabla o la vista no debe contener ningún carácter especial, como por ejemplo, “_”, “+”, “$”, “%”, ni espacios en blanco.
- No utilice palabras reservadas de base de datos como nombres de tabla o de vista.
- Los nombres de las vistas deben finalizar con `VW`.
- Los nombres de tablas y vistas deben ser sustantivos en singular.

Convenios de denominación para columnas

La lista siguiente proporciona una guía para la denominación de columnas en tablas nuevas:

- Para evitar conflictos con los nombres (nombres duplicados) de las columnas de las tablas de WebSphere Commerce en releases futuros, el primer carácter del nombre de la columna debe ser una `X`. Por ejemplo, `XMYCOLUMN`.
- El nombre de la columna no debe tener más de 18 caracteres de longitud. Si el nombre que desea sobrepasa este límite, hágalo más corto eliminando las vocales a partir del final del nombre, hasta que sólo queden 18 caracteres.
- Los nombres de las columnas no deben contener ningún carácter especial, como por ejemplo, "_", "+", "$", "%", ni espacios en blanco.
- No utilice palabras reservadas de base de datos como nombre de columna.
- Las palabras combinadas pueden utilizarse como nombres de columna utilizando la combinación activa de voz. Por ejemplo, `COMBINERESULT`.
- Las columnas de clave primaria generadas deben denominarse *`tabla`*`_id`. Por ejemplo, la clave primaria para la tabla USERS es `USERS_ID`.
- No deben cambiarse los nombres de columna de clave externa generados.
- Si reserva columnas para una personalización futura, deben denominarse `campo`*`x`* donde *`x`* es un dígito numérico que empieza por el 1.

Convenios de denominación para índices

La lista siguiente proporciona una guía para la denominación de índices en tablas nuevas:

- El nombre del índice no debe tener más de 18 caracteres de longitud.
- El nombre del índice no debe contener espacios en blanco.
- El nombre del índice no debe contener palabras reservadas de base de datos.
- Un índice no exclusivo debe denominarse `I_`*`tablax`* donde *`tabla`* es el nombre de la tabla y *`x`* es un número, empezando por 1. Por ejemplo, un índice no exclusivo para la tabla USERS es `I_USERS1`.
- Un índice exclusivo debe denominarse `UI_`*`tablax`* donde *`tabla`* es el nombre de la tabla y *`x`* es un número, empezando por 1. Por ejemplo, un índice exclusivo para la tabla USERS es `UI_USERS1`.
- El tamaño total del índice no debe ser superior a 254 bytes.
- El nombre del índice debe ser exclusivo en todo el esquema de base de datos.

Convenios de denominación para claves primarias

La lista siguiente proporciona una guía para la denominación de claves primarias en tablas nuevas:

- El nombre de la clave primaria no debe tener más de 18 caracteres de longitud.
- El nombre de la clave primaria no debe contener espacios en blanco.
- El nombre de la clave primaria no debe contener palabras reservadas de base de datos.
- La clave primaria debe denominarse `P_`*`tabla`* donde *`tabla`* es el nombre de la tabla. Por ejemplo, la clave primaria para la tabla USERS es `P_USERS`.
- El nombre de la clave primaria debe ser exclusivo en todo el esquema de base de datos.

Convenios de denominación para claves externas

La lista siguiente proporciona una guía para la denominación de claves externas en tablas nuevas:

- El nombre de la clave externa no debe tener más de 18 caracteres de longitud.

- El nombre de la clave externa no debe contener espacios en blanco.
- El nombre de la clave externa no debe contener palabras reservadas de base de datos.
- La clave externa debe denominarse F_*tabla* donde *tabla* es el nombre de la tabla. Por ejemplo, la clave externa para la tabla USERS es F_USERS1.
- El nombre de la clave externa debe ser exclusivo en todo el esquema de base de datos.

Convenios de denominación para desencadenantes de base de datos

La lista siguiente proporciona una guía para la denominación de desencadenantes de base de datos:

- El nombre del desencadenante de base de datos no debe tener más de 18 caracteres de longitud.
- El nombre del desencadenante de base de datos no debe contener espacios en blanco.
- El nombre del desencadenante de base de datos no debe contener palabras reservadas de base de datos.
- El desencadenante de base de datos debe denominarse T_*tabla* donde *tabla* es el nombre de la tabla. Por ejemplo, el nombre de desencadenante de base de datos para la tabla USERS es T_USERS1.
- El nombre del desencadenante de base de datos debe ser exclusivo en todo el esquema de base de datos.

Consideraciones sobre el tipo de datos de las columnas de la base de datos

Esta sección presenta los tipos de datos de las columnas que se pueden utilizar al crear tablas nuevas. Las descripciones de los distintos tipos de datos utilizan la terminología de DB2. El apartado "Diferencias de los tipos de datos entre bases de datos" en la página 72 describe las diferencias si utiliza una base de datos distinta.

BIGINT

Es un entero con signo de 64 bits que tiene un rango de -9223372036854775807 a 9223372036854775807. Compárelo con INTEGER, cuyo tamaño es la mitad del de BIGINT.

INTEGER

Es un entero con signo de 32 bits que tiene un rango de -2147483647 a 2147483647. En general, INTEGER debe ser el tipo de datos numérico finito por omisión, en lugar de BIGINT. A menos que haya un buen motivo para utilizar BIGINT, a efectos de rendimiento es mejor utilizar INTEGER como tipo de datos numérico. Un usuario común del tipo de datos BIGINT es una clave generada por el sistema.

No es aconsejable el uso de los tipos de datos SMALLINT o SHORT porque están correlacionados con un tipo de datos Java no objeto y estos tipos de datos no objeto producirán problemas en algunas instancias de objetos bean enterprise.

TIMESTAMP

Es un valor que consta de siete partes (año, mes, día, hora, minuto, segundo y microsegundo) y que designa una fecha y una hora; incluye una especificación fraccional de microsegundos. La representación interna de la indicación de la hora es una serie de 10 bytes, cada uno de los cuales

consta de 2 dígitos decimales empaquetados. Los 4 primeros bytes representan la fecha, los 3 bytes siguientes la hora y los 3 últimos los microsegundos.

CHAR
Es una serie de caracteres con una longitud fija igual a INTEGER, que puede constar de 1 a 254 caracteres. Si se omite la especificación de la longitud, se toma la longitud de un carácter. Puesto que CHAR es una columna de base de datos de longitud fija, los espacios de caracteres de cola no utilizados se cambian por espacios en blanco. A menos que se utilice por motivos de rendimiento, no es recomendable utilizar el tipo de datos CHAR porque CHAR no es flexible y la longitud no se puede cambiar posteriormente. Como norma básica, si su columna de serie tiene menos de 64 caracteres de longitud y se actualiza o recupera regularmente, utilice CHAR en su lugar para obtener un mejor rendimiento.

VARCHAR
Es una serie de caracteres de longitud variable cuyo valor máximo es igual a integer, que puede constar de 1 a 32672 caracteres. Sin embargo, a diferencia de CHAR, donde los datos de la columna se almacenan junto con la tabla, VARCHAR se representa internamente como un puntero de referencia dentro de una página de base de datos. Por tanto, la longitud de la columna VARCHAR puede cambiarse en cualquier momento después de su creación.

LONG VARCHAR
Es la serie de caracteres de longitud variable que se puede utilizar si no se puede crear VARCHAR dentro de la misma página de base de datos. LONG VARCHAR es muy similar a VARCHAR a excepción de que puede abarcar varias páginas de base de datos. Restrinja el uso del tipo de datos LONG VARCHAR sólo a los casos en que sea absolutamente necesario, porque los objetos LONG VARCHAR normalmente son muy costosos en términos de rendimiento.

CLOB Es otra serie de caracteres de longitud variable que se puede utilizar si la longitud de la columna debe sobrepasar el límite de 32 KB de LONG VARCHAR. La longitud de un objeto CLOB puede alcanzar 1 GB sin modificar la configuración de la base de datos. Los datos de texto almacenados como CLOB se convierten adecuadamente cuando se desplazan entre diferentes sistemas.

BLOB Es una serie de caracteres binarios de longitud variable que almacena datos no estructurados en la base de datos. Los objetos BLOB pueden almacenar hasta 4 GB de datos binarios. En general, debe evitar la utilización de BLOB como tipo de datos de columna, a menos que sea absolutamente necesario. En términos de rendimiento, un objeto BLOB se considera uno de los objetos de base de datos más costosos.

DECIMAL(20,5)
Este tipo de datos está especialmente definido para que se utilice con la mayoría de números con coma decimal fija, como es el caso de las monedas. Para los números decimales de coma flotante, puede utilizarse FLOAT.

Diferencias de los tipos de datos entre bases de datos

La tabla siguiente lista los tipos de datos utilizados en el esquema de base de datos de WebSphere Commerce y muestra los tipos de datos correspondientes para las distintas implementaciones de base de datos.

Objeto JDBC	Windows AIX Solaris **DB2**	Windows AIX Solaris **Oracle®**	400 **DB2**
Hashtable	BLOB()	BLOB	BLOB()
Timestamp	TIMESTAMP	DATE	TIMESTAMP
Integer	INTEGER	INTEGER	INTEGER
BigDecimal	DECIMAL(,)	DECIMAL(,)	DECIMAL(,)
Long	BIGINT	NUMBER	BIGINT
Double	FLOAT	NUMBER	FLOAT
String	CHAR()	VARCHAR2()	GRAPHIC() CCSID 13488
byte[]	CHAR() para datos de bit	RAW()	CHAR() para datos de bit
String	VARCHAR()	VARCHAR2()	VARGRAPHIC() CCSID 13488
String	LONG VARCHAR	VARCHAR2() (Para más detalles, consulte la nota que hay al final de la tabla.)	VARGRAPHIC(4000) ALLOCATE() CCSID 13488
byte[]	LONG VARCHAR para datos de bit	LONG RAW	VARCHAR(8000) ALLOCATE() para datos de bit
String	CLOB()	CLOB()	DBCLOB() CCSID 13488

Nota:

Debido a que se obtiene una proporción incoherente de resultados satisfactorios cuando el controlador JDBC de Oracle maneja información cuyo tipo de datos es LONG, se recomienda evitar la utilización del tipo de datos LONG siempre que sea posible. El error que se produce con más frecuencia en esta situación es “Stream has already been closed” (La corriente ya se ha cerrado).

Si tiene que utilizar este tipo de datos, sólo puede tener una columna por tabla de base de datos que utilice el tipo LONG. Además, al crear una sentencia select, no ponga la columna LONG como primer ni como último elemento en select. Otra solución en las operaciones con una gran carga es evitar la correlación de esta columna particular con un campo CMP en un bean de entidad. En lugar de ello, utilice un bean de sesión para llevar a cabo las recuperaciones y actualizaciones en esta columna.

Capítulo 4. Control de acceso

Información sobre el control de acceso

El modelo de control de acceso de una aplicación de WebSphere Commerce tiene tres conceptos principales: usuarios, acciones y recursos. Los usuarios son las personas que utilizan el sistema. Los recursos son las entidades que se encuentran en la aplicación o son gestionadas por ella. Por ejemplo, los recursos pueden ser productos, documentos o pedidos. Los perfiles de usuario que representan a las personas también son recursos. Las acciones son las actividades que los usuarios pueden efectuar en los recursos. El control de acceso es el componente de la aplicación de comercio electrónico que determina si un usuario específico puede efectuar una acción concreta en un recurso dado.

En una aplicación WebSphere Commerce, hay dos niveles principales de control de acceso. El primer nivel lo gestiona WebSphere Application Server. En este caso, WebSphere Commerce utiliza WebSphere Application Server para proteger los beans enterprise y los servlets. El segundo nivel de control de acceso es el sistema de control de acceso refinado de WebSphere Commerce.

La estructura de control de acceso de WebSphere Commerce utiliza políticas de control de acceso para determinar si a un usuario específico se le permite efectuar una acción concreta en un recurso dado. Esta estructura de control de acceso proporciona un control de acceso muy selectivo. Funciona conjuntamente con el sistema de control de acceso de WebSphere Application Server pero no le sustituye.

Visión general de la protección de recursos en WebSphere Application Server

Los siguientes recursos de WebSphere Commerce se protegen mediante el control de acceso de WebSphere Application Server:

- Beans de entidad
  Estos beans modelan objetos de una aplicación de comercio electrónico. Son objetos distribuidos a los que pueden acceder clientes remotos.
- Plantillas JSP
  WebSphere Commerce utiliza plantillas JSP para páginas de visualización. Cada plantilla JSP puede contener uno o más beans de datos que recuperan datos de beans de entidad. Los clientes pueden solicitar páginas JSP creando una petición de URL.
- Mandatos de controlador y de tarea
  Los clientes pueden solicitar mandatos de controlador y de tarea creando peticiones de URL. Además, una página de visualización puede contener un enlace a otra utilizando el nombre de archivo JSP o el nombre de vista, tal como está registrado en la tabla VIEWREG.

WebSphere Commerce Server normalmente se configura de modo que utilice las siguientes vías de acceso Web:

- `/webapp/wcs/stores/servlet/*`
  Se utiliza para peticiones al servlet de peticiones.
- `/webapp/wcs/stores/*.jsp`
  Se utiliza para peticiones al servlet JSP.

En el siguiente diagrama se muestra la ruta que pueden seguir dichas peticiones para acceder a los recursos de WebSphere Commerce, para la configuración de vía de acceso a la Web indicada anteriormente.

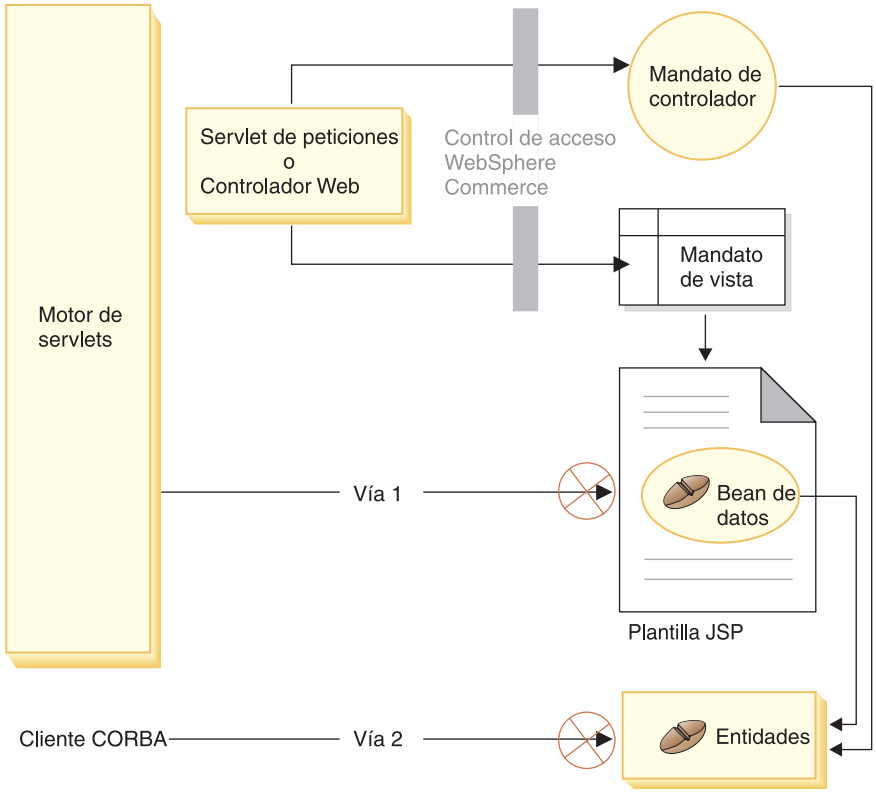

Figura 19.

Todas las peticiones legítimas deben dirigirse al servlet de peticiones, que después las dirigirá al controlador Web. El controlador Web implementa el control de acceso para mandatos de controlador y vistas. Sin embargo, las vías de acceso Web mostradas anteriormente permiten que los usuarios malintencionados accedan directamente a las plantillas JSP (vía 1) y los beans de entidad (vía 2). Para impedir que estos ataques obtengan un resultado satisfactorio, deben rechazarse durante la ejecución.

El acceso directo a las plantillas JSP y beans de entidad puede impedirse utilizando uno de los siguientes enfoques:

Seguridad de WebSphere Application Server
: WebSphere Application Server proporciona una característica de seguridad. Si se utiliza este enfoque, todos los enfoques de beans enterprise y plantillas JSP se configuran de modo que sólo los invoque la identidad del sistema. Para acceder a estos recursos de WebSphere Commerce, una petición de URL debe direccionarse al servlet de peticiones que establece la identidad del sistema en la hebra actual, antes de pasarla al controlador Web. El controlador Web se asegura entonces de que el llamante tenga la autorización necesaria antes de pasar la petición al correspondiente mandato de controlador o vista. El componente de seguridad de WebSphere Application Server rechazará todos los intentos de acceder directamente a plantillas JSP y beans de entidad (es decir, sin utilizar el controlador Web).

Para obtener más información sobre la configuración de WebSphere Application Server para proteger los recursos de WebSphere Commerce, consulte la publicación *WebSphere Commerce, Guía de instalación*. Para obtener información sobre seguridad dentro de WebSphere Application Server, consulte el tema sobre Administración del sistema en la documentación de WebSphere Application Server.

Para obtener información sobre la configuración de la seguridad de WebSphere Application Server para los métodos de los beans enterprise personalizados, consulte "Integración de nuevos beans enterprise en una aplicación de empresa" en la página 301 y "Integración de beans enterprise modificados en una aplicación de empresa" en la página 304.

Protección mediante cortafuegos

Cuando WebSphere Commerce Server se ejecuta detrás del cortafuegos, los clientes de Internet no pueden acceder directamente a los beans de entidad. Si se utiliza este método, la protección de las plantillas JSP la proporciona el bean de datos que se incluye en la página. Al bean de datos lo activa el gestor de beans de datos. El gestor de beans de datos detecta si la plantilla JSP ha sido reenviada por un mandato de vista. Si no ha sido reenviada por un mandato de vista, se genera una excepción y la petición para la plantilla JSP se rechaza.

Introducción a las políticas de control de acceso de WebSphere Commerce

El modelo de control de acceso de WebSphere Commerce se basa en la implantación de políticas de control de acceso. Estas políticas permiten externalizar las normas de control de acceso del código de la lógica de negocio, por lo se erradica la necesidad de codificarlas en el código. Por ejemplo, no es necesario incluir código como este:

```
if (user.isAdministrator())
   then {}
```

Las políticas de control de acceso las implementa el gestor de políticas de control de acceso. En general, cuando un usuario intenta acceder a un recurso protegido, el gestor de políticas de control de acceso determina las políticas de control de acceso que son aplicables para ese recurso protegido y, a continuación, basándose en esas políticas, determina si el usuario puede acceder a los recursos solicitados.

Una política de control de acceso es una política que consta de 4 registros y que se almacena en la tabla ACPOLICY. Las políticas de control de acceso toman la siguiente forma:

```
AccessControlPolicy [UserGroup, ActionGroup, ResourceGroup, Relationship]
```

Los elementos en la política de control de acceso de 4 registros especifican que un usuario que pertenezca a un grupo concreto de usuarios puede llevar a cabo las acciones del grupo de acciones indicadas en los recursos que pertenezcan al grupo de recursos especificado, siempre que el usuario satisfaga las condiciones indicadas en la relación o grupo de relaciones, con respecto al recurso en cuestión. Por ejemplo, `[AllUsers, UpdateDoc, doc, creator]` especifica que todos los usuarios pueden actualizar un documento, si son los creadores del mismo.

El grupo de usuarios es un tipo específico de grupo de miembros que está definido en la tabla de base de datos MBRGRP. Un grupo de usuarios debe asociarse con el tipo de grupo de miembros de `-2`. El valor de `-2` representa un grupo de acceso y

se define en la tabla MBRGRPTYPE. La asociación entre el grupo de usuarios y el tipo de grupo de miembros se almacena en la tabla MBRGRPUSG.

La pertenencia de un usuario a un grupo de usuarios particular puede especificarse de forma explícita o implícita. La especificación explícita se produce si la tabla MBRGRPMBR indica que el usuario pertenece a un grupo de miembros concreto. La especificación implícita se produce si el usuario satisface una condición (por ejemplo, todos los usuarios que tienen el rol de Jefe de producto) indicada en la tabla MBRGRPCOND. También puede haber condiciones combinadas (por ejemplo, todos los usuarios que son Jefe de producto y que han tenido este rol durante los 6 últimos meses como mínimo) o exclusiones explícitas.

La mayor parte de las condiciones para incluir a un usuario en un grupo de usuarios se basan en el rol que tiene. Por ejemplo, podría haber una política de control de acceso que permitiera a los usuarios que tienen el rol de Jefe de producto, efectuar operaciones de gestión de catálogo. En ese caso, cualquier usuario al que se le haya asignado el rol de Jefe de producto en la tabla MBRROLE está implícitamente incluido en el grupo de usuarios.

Para obtener más detalles sobre el subsistema de grupo de miembros, consulte la ayuda en línea de WebSphere Commerce.

El elemento ActionGroup viene de la tabla ACACTGRP. Un grupo de acciones hace referencia a un grupo de acciones especificado explícitamente. La lista de acciones se almacena en la tabla ACACTION y la relación de cada acción con su grupo (o grupos) de acciones se almacena en la tabla ACACTACTGP. Un ejemplo de grupo de acciones es ″OrderWriteCommands″. Este grupo de acciones incluye las siguientes acciones que se utilizan para actualizar pedidos:

- com.ibm.commerce.order.commands.OrderDeleteCmd
- com.ibm.commerce.order.commands.OrderCancelCmd
- com.ibm.commerce.order.commands.OrderProfileUpateCmd
- com.ibm.commerce.order.commands.OrderUnlockCmd
- com.ibm.commerce.order.commands.OrderScheduleCmd
- com.ibm.commerce.order.commands.ScheduledOrderCancelCmd
- com.ibm.commerce.order.commands.ScheduledOrderProcessCmd
- com.ibm.commerce.order.commands.OrderItemAddCmd
- com.ibm.commerce.order.commands.OrderItemDeleteCmd
- com.ibm.commerce.order.commands.OrderItemUpdateCmd
- com.ibm.commerce.order.commands.PayResetPMCmd

Un grupo de recursos es un mecanismo para agrupar tipos de recursos específicos. La pertenencia de un recurso a un grupo de recursos puede especificarse de una de estas dos maneras:

- Utilizando la columna de condiciones en la tabla ACRESGRP
- Utilizando la tabla ACRESGPRES

En la mayoría de casos, es suficiente utilizar la tabla ACRESGPRES para asociar recursos a los grupos de recursos. Utilizando este método, los recursos se definen en la tabla ACRESCGRY utilizando su nombre de clase Java. A continuación, estos recursos se asocian a los grupos de recursos adecuados (tabla ACRESGRP) utilizando la tabla de asociaciones ACRESGPRES. En los casos en que el nombre de clases de Java no sea suficiente, por sí solo, para definir los miembros de un grupo de recursos (por ejemplo, si necesita restringir más los objetos de esta clase

basándose en un atributo del recurso), el grupo de recursos puede definirse enteramente utilizando la columna de condiciones de la tabla ACRESGRP. Tenga en cuenta que para efectuar esta agrupación de recursos basándose en un atributo, el recurso debe también implementar la interfaz Groupable.

El diagrama siguiente muestra una especificación de ejemplo de agrupación de recursos. En este ejemplo, el grupo de recursos `10023` incluye todos los recursos que están asociados a él en la tabla ACRESGPRES. El grupo de recursos `10070` está definido utilizando la columna del campo de condiciones en la tabla ACRESGRP. Este grupo de recursos incluye instancias de la interfaz remota Order, que también tiene el estado = `"Z"` (especificando una lista de solicitudes compartida).

Nota: Encontrará detalles sobre la información de XML para la columna de condiciones de la tabla ACRESGRP en la publicación *WebSphere Commerce, Guía de control de acceso*.

ACRESGRP

AcResGrp_Id	GrpName	Conditions
10023	AccountRepresentatives CmdResourceGroup	null
10070	SharedRequisitionList ResourceGroup	<profile> <andListCondition> <simpleCondition> <variable name="Status"/> <operator name="="/> <value data="Z"/> </simpleCondition> <simpleCondition> <variable name="classname"/> <operator name="="/> <value data="com.ibm.commerce.order.objects.Order"/> </simpleCondition> </andListCondition> </profile>

ACRESGRPES

AcResGrp_Id	AcResCgry_Id
10023	10246
10023	10247
10023	10248
10023	10249
10023	10250

ACRESCGRY

AcResCgry_Id	ResClassname
10246	com.ibm.commerce.contract.commands.ContractCreateCmd
10247	com.ibm.commerce.contract.commands.ContractCreateCmd
10248	com.ibm.commerce.contract.commands.ContractCreateCmd
10249	com.ibm.commerce.contract.commands.ContractCreateCmd
10250	com.ibm.commerce.contract.commands.ContractCreateCmd

Figura 20.

La columna MEMBER_ID de las tablas ACACTGRP, ACRESGRP y ACRELGRP deben tener el valor `-2001` (organización raíz).

La política de control de acceso puede incluir, opcionalmente, un elemento Relationship o RelationshipGroup como cuarto elemento.

Si su política de control de acceso utiliza un elemento Relationship, éste proviene de la tabla ACRELATION. Por otro lado, si incluye un elemento RelationshipGroup, éste proviene de la tabla ACRELGRP. Tenga en cuenta que no es necesario incluir ninguno y tampoco se pueden incluir los dos. Una especificación RelationshipGroup de la tabla ACRELGRP tiene prioridad sobre la información de Relationship de la tabla ACRELATION.

La tabla ACRELATION especifica los tipos de relación existentes entre los usuarios y los recursos. Algunos ejemplos de relación son, creador, sometedor y propietario. Un ejemplo del uso del elemento de relación sería utilizarlo para asegurar que el creador de un pedido pueda siempre actualizarlo.

La tabla ACRELGRP especifica los tipos de grupos de relaciones que se pueden asociar a recursos específicos. Un grupo de relaciones consiste en una agrupación de una o más cadenas de relaciones. Una cadena de relaciones es una serie de una o más relaciones. Un ejemplo de un grupo de relaciones es especificar que un usuario debe ser el creador del recurso y, además, pertenecer a la entidad de la organización compradora a la que se hace referencia en el recurso.

La especificación del grupo de relaciones (o de la relación) es una parte opcional de la política de control de acceso. Normalmente, se utiliza si ha creado sus propios mandatos y esos mandatos no están restringidos a ciertos roles. En esos casos, quizá desee imponer una relación entre el usuario y el recurso. Normalmente, para restringir los mandatos a ciertos roles se utiliza el elemento UserGroup de la política de control de acceso, en lugar de utilizar el elemento Relationship.

Otro importante concepto relacionado con las políticas de control de acceso es el concepto de *propietario* de la política de control de acceso. El propietario es la entidad de organización que posee la política de control de acceso. Es importante saber quien es el propietario de la política de control de acceso porque dicha política sólo se puede aplicar a los recursos que posee ese mismo propietario.

Para cada recurso en cuestión, el gestor de políticas de control de acceso aplica las políticas de control de acceso que posee la entidad de organización propietaria o sus entidades de organización padre en la jerarquía de miembros, hasta que se encuentra una política que otorgue permiso o hasta que se hayan comprobado todas las políticas y ninguna otorgue permiso.

Observe el diagrama siguiente que muestra una jerarquía de miembros.

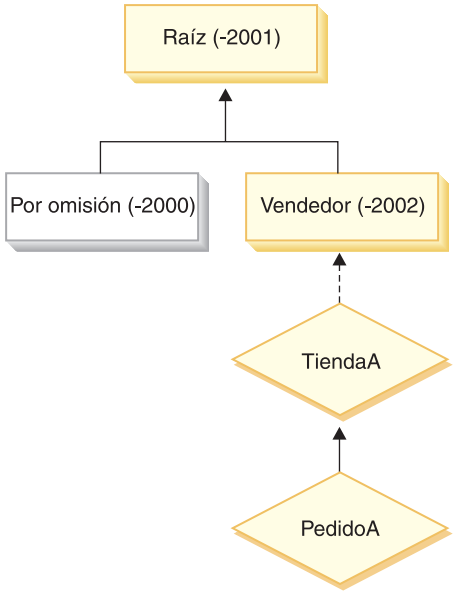

Figura 21.

Para el recurso "PedidoA", puede aplicarse cualquier política de control de acceso que sea propiedad de Vendedor o de la organización Raíz. Si el gestor de políticas de control de acceso encuentra una política que sea propiedad de estas organizaciones y que otorgue permiso al usuario (basándose en los cuatro elementos de la política de control de acceso), cesa inmediatamente de buscar a través de las políticas. Sin embargo, si no encuentra ninguna política de control de acceso cuyo propietario sea una de las organizaciones que otorgan permiso al usuario para efectuar la acción en los recursos protegidos, el acceso se deniega.

Grupos de relaciones

Un grupo de relaciones le permite especificar varias relaciones. Una relación puede vincular directamente un usuario y el recurso en cuestión, o puede consistir en una cadena de relaciones que, de forma indirecta, relacione el usuario con el recurso.

Nota: Para las siguientes secciones sobre los grupos de relaciones, es importante tener en cuenta que las únicas organizaciones disponibles en WebSphere Commerce Professional Edition son la organización raíz, la organización por omisión y la organización vendedora. Los ejemplos que hacen referencia a otras organizaciones sólo se aplican a WebSphere Commerce Business Edition.

Diferencias entre las relaciones y los grupos de relaciones: Las políticas de control de acceso pueden especificar que un usuario debe tener una relación específica con respecto al recurso al que se accede, o pueden especificar que un usuario debe satisfacer las condiciones detalladas en un grupo de relaciones.

En la mayoría de casos, la especificación de una relación debería satisfacer los requisitos de control de acceso de su aplicación. Sin embargo, si la política es tal que debe especificar una relación que no se establece directamente entre el usuario y el recurso, sino que se trata de una serie de relaciones entre el usuario y el recurso, debe utilizar un grupo de relaciones.

Por ejemplo, si debe especificar una asociación entre un usuario y una organización compradora en la que la relación requiere que el usuario tenga un rol

concreto en esa organización o que el usuario sea miembro de la organización compradora, debe utilizar un grupo de relaciones y una cadena de relaciones.

Si sólo necesita imponer una asociación que relaciona directamente al usuario y al recurso en cuestión, puede utilizar una relación simple. Por ejemplo, éste sería el caso si necesita forzar que el usuario sea el creador del recurso.

Si combina varias relaciones simples, por ejemplo, el usuario debe ser el creador *o* el sometedor, entonces se trata de una cadena de relaciones y debe utilizar un grupo de relaciones. Esta combinación de relaciones simples puede tener lugar tanto al utilizar WebSphere Commerce Professional Edition como WebSphere Commerce Business Edition.

Información general sobre grupos de relaciones: Una cadena de relaciones es una serie de una o más relaciones. La longitud de una cadena de relaciones la establece el número de relaciones que contiene. Este número puede determinarse examinando el número de elementos `<parameter name="Nombre" value="Valor" />` en la representación XML de la cadena de relaciones.

El método fulfills() del recurso sólo debe manejar el último elemento `<parameter name="Relación" value="Valor" />` . El gestor de políticas de control de acceso maneja internamente los demás atributos.

Cuando una cadena de relaciones tiene una longitud igual a 2, el primer elemento `<parameter name="Nombre" value="Valor" />` es entre un usuario y una entidad de organización. El último elemento `<parameter name="Nombre" value="Valor" />` es entre una entidad de organización y el recurso.

Si necesita definir grupos de relaciones, debe hacerlo definiendo la información de grupo de relaciones en un archivo XML. Puede modificar el archivo `defaultAccessControlPolicies.xml` o crear su propio archivo XML. Para obtener más información sobre cómo crear esta información basada en XML, consulte la publicación *WebSphere Commerce, Guía de control de acceso*.

Las secciones siguientes muestran ejemplos de distintos tipos de grupos de relaciones.

Grupos de relaciones compuestos de una sola cadena de relaciones: Business Como parte de una política de control de acceso, quizá tenga que imponer que un usuario deba pertenecer a la entidad de organización compradora del recurso. Esto requiere la creación de un grupo de relaciones que esté compuesto por una cadena de relaciones de longitud dos. Se dice que la cadena de relaciones es de longitud ″dos″ porque consta de dos relaciones independientes. La primera relación es entre el usuario y su entidad de organización padre. El usuario es el ″hijo″ en esa relación. Para la segunda relación, el gestor de políticas de control de acceso comprueba si la entidad de organización padre satisface la relación de entidad de organización compradora del recurso. En otras palabras, devuelve ″true″ si es la entidad de organización compradora del recurso.

La siguiente sección de código XML está extraída del archivo `defaultAccessControlPolicies.xml` y muestra cómo definir este tipo de grupo de relaciones:

```
<RelationGroup Name="MemberOf->BuyerOrganizationalEntity"
      OwnerID="RootOrganization">
   <RelationCondition><![CDATA[
      <profile>
```

```
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="HIERARCHY" value="child"/>
            <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
         </openCondition>
      </profile>
   ]]></RelationCondition>
</RelationGroup>
```

Business Otro ejemplo sería forzar que el usuario tenga el rol de Representante de la cuenta para la entidad de organización compradora del recurso en cuestión. De nuevo, se utilizaría un grupo de relaciones compuesto por una cadena de relaciones de longitud dos. La primera parte de la cadena encontraría todas las entidades de organización para las cuales el usuario tiene el rol de Representante de la cuenta. Luego, para este conjunto de entidades de organización, el gestor de políticas de control de acceso comprueba si algunas de ellas satisfacen la relación de entidad de organización compradora con respecto al recurso. En otras palabras, devuelve true si una de ellas es la entidad de organización compradora del recurso.

La siguiente sección de código XML está extraída del archivo `defaultAccessControlPolicies.xml` y muestra cómo definir este tipo de grupo de relaciones:

```
<RelationGroup Name="AccountRep->BuyerOrganizationalEntity"
      OwnerID="RootOrganization">
   <RelationCondition><![CDATA[
      <profile>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="ROLE" value="Account Representative"/>
            <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
         </openCondition>
      </profile>
   ]]></RelationCondition>
</RelationGroup>
```

Grupos de relaciones compuestos de varias cadenas de relaciones: Es posible crear un grupo de relaciones que contenga varias cadenas de relaciones. Al crearlo, debe especificar si el usuario debe satisfacer todas las cadenas de relaciones, indicando que es un entorno *AND* o si el usuario debe satisfacer una de las cadenas de relaciones como mínimo, indicando que es un entorno *OR*.

Business Para demostrar este tipo de relación, la siguiente sección de código XML se utiliza para forzar que el usuario sea el creador del recurso y, además, pertenezca a la entidad de organización compradora especificada en el recurso. La primera cadena, que especifica que el usuario debe ser el creador del recurso, es de longitud uno. La segunda cadena, que especifica que el usuario debe pertenecer a la entidad de organización compradora especificada en el recurso, es de longitud dos.

```
<RelationGroup Name="Creator_And_MemberOf->BuyerOrganizationalEntity"
      OwnerID="RootOrganization">
 <RelationCondition><![CDATA[
  <profile>
   <andListCondition>
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="RELATIONSHIP" value="creator" />
    </openCondition>
    <openCondition name="RELATIONSHIP_CHAIN">
     <parameter name="HIERARCHY" value="child"/>
     <parameter name="RELATIONSHIP" value="BuyingOrganizationalEntity"/>
    </openCondition>
```

```
   </andListCondition>
  </profile>
 ]]></RelationCondition>
</RelationGroup>
```

Si, en lugar del entorno *AND*, es necesario que el usuario satisfaga una de las dos cadenas de relaciones, debe cambiarse el identificador `<andListCondition>` por `<orListCondition>`.

Professional Business Para mostrar un grupo de relaciones que se puede utilizar en WebSphere Commerce Professional Edition (así como en WebSphere Commerce Business Edition), supongamos un grupo de relaciones que se utilice para forzar que el usuario sea el creador o el sometedor del recurso. Esto se muestra en la sección de código XML siguiente.

```
<RelationGroup Name="Creator_Or_Submitter"
     OwnerID="RootOrganization">
   <RelationCondition><![CDATA [
   <profile>
      <orListCondition>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="RELATIONSHIP"value="creator"/>
         </openCondition>
         <openCondition name="RELATIONSHIP_CHAIN">
            <parameter name="RELATIONSHIP"value="submitter"/>
         </openCondition>
      </orListCondition>
   </profile>
   ]]></RelationCondition>
</RelationGroup>
```

Tipos de control de acceso

Hay dos tipos de control de acceso y ambos se basan en la política: control de acceso a nivel de mandato y control de acceso a nivel de recurso.

El control de acceso a nivel de mandato (también denominado “a nivel de rol”) utiliza un tipo de política general. Puede especificar que todos los usuarios de un rol particular puedan ejecutar ciertos tipos de mandatos. Por ejemplo, puede especificar que los usuarios con rol de Representante de cuentas puedan ejecutar cualquier mandato del grupo de recursos AccountRepresentativesCmdResourceGroup. O, como indica el siguiente diagrama, otra política de ejemplo es especificar que todos los administradores de tienda puedan efectuar las acciones especificadas en el Grupo ExecuteCommandAction en cualquier recurso especificado por StoreAdminCmdResourceGrp.

Nota: La información de XML para la columna de condiciones de la tabla MBRGRPCOND se genera cuando utiliza la Consola de administración para definir los grupos de acceso. Para obtener información sobre la utilización de la Consola de administración para definir los grupos de acceso, consulte la ayuda en línea de WebSphere Commerce.

ACPOLICY

PolicyName	Member_Id	MbrGrp_Id	AcActGrp_id	AcResGrp_Id	AcRelGrp_Id
StoreAdministrators ExecuteStoreAdmin CmdResourceGroup	-2001	-8	10052	10018	null

MBRGRP

MbrGrp_Id	MbrGrpName
-8	StoreAdministrators

MBRGRPCOND

MbrGrp_Id	Conditions
-8	<profile> <simpleCondition> <variable name="role"/> <operator name="="/> <value data="Store Administrator"/> </simpleCondition> </profile>

ACACTGRP

AcActGrp_Id	GroupName
10052	ExecuteCommandActionGroup

ACRESGRP

AcResGrp_Id	GrpName
10018	StoreAdminCmdResourceGroup

Figura 22.

Una política de control de acceso a nivel de mandato siempre tiene a `ExecuteCommandActionGroup` como grupo de acciones para mandatos de controlador. Para las vistas, el grupo de recursos siempre es `ViewCommandResourceGroup`.

Todos los mandatos de controlador deben protegerse mediante control de acceso a nivel de mandato. Además, cualquier vista que se pueda llamar directamente o que pueda iniciarse mediante un redireccionamiento desde otro mandato (en oposición a iniciarse reenviando a la vista) debe protegerse mediante control de acceso a nivel de mandato.

El control de acceso a nivel de mandato no tiene en cuenta el recurso sobre el que actúa el mandato. Simplemente determina si al usuario se le permite ejecutar ese mandato específico. En caso afirmativo, se podría aplicar una política de control de acceso a nivel de recurso para determinar si el usuario puede acceder el recurso en cuestión.

Tomemos el ejemplo de un administrador de tienda que intenta efectuar una tarea administrativa. El primer nivel de control de acceso sería determinar si a este usuario se le permite ejecutar ese mandato específico de administración de tienda. Una vez se haya determinado que el usuario puede hacerlo (porque a los

administradores de tienda se les permite ejecutar los mandatos del grupo storeAdminCmds), puede llamarse a una política de control de acceso a nivel de recurso. Esta política puede indicar que a los administradores de tienda sólo se les permite realizar tareas administrativas para las tiendas cuyo propietario sea la organización de la cual el usuario es administrador de tienda.

En resumen, en el control de acceso a nivel de mandato, el "recurso" es el mandato mismo y la "acción" es simplemente ejecutar el mandato (en otras palabras, crear una instancia del objeto de mandatos). La comprobación del control de acceso determina si al usuario se le permite ejecutar el mandato. Por el contrario, en el control de acceso a nivel de recursos, el "recurso" es cualquier recurso protegible al que accede el mandato o el bean y la "acción" es el mandato mismo.

Interacciones del control de acceso

Esta sección presenta el diagrama de interacciones que muestra el funcionamiento del control de acceso en la estructura de la política de control de acceso de WebSphere Commerce.

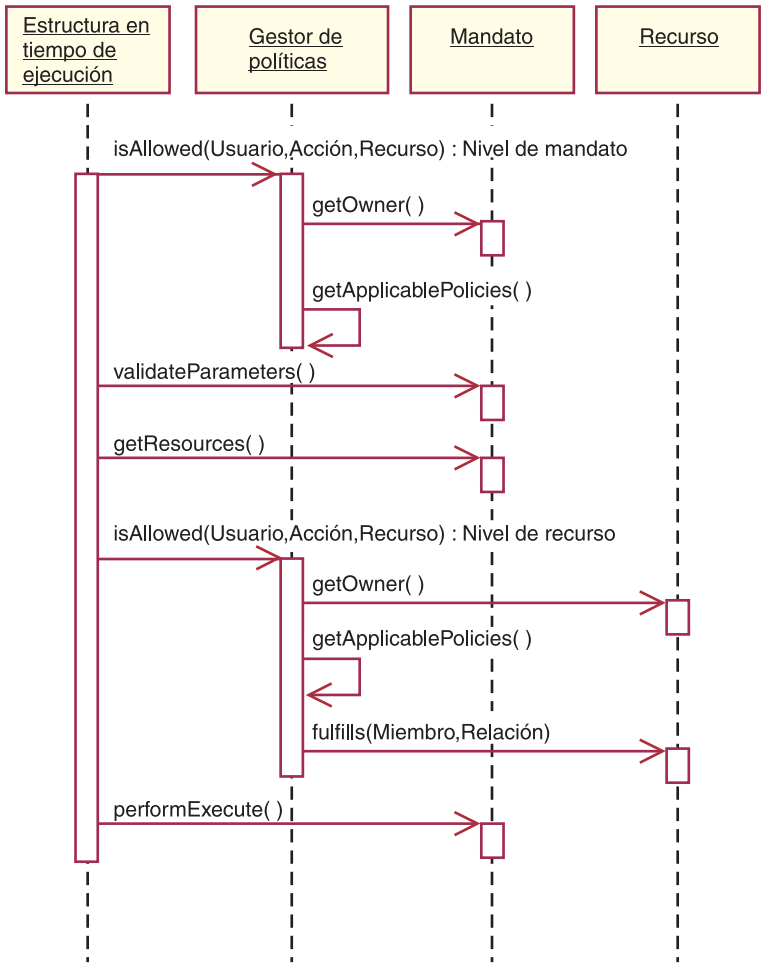

Figura 23.

El diagrama anterior muestra las acciones que realiza el *gestor de políticas*. El gestor de políticas de control de acceso es el componente de control de acceso que determina si al usuario actual se le permite ejecutar la acción especificada en el

recurso indicado. Esto se determina buscando en las políticas que posee el propietario de recursos y sus organizaciones padre. Si al menos una política otorga acceso, el permiso se otorga.

La lista siguiente describe las acciones del diagrama de interacciones anterior. Están ordenadas de arriba a abajo.

1. `isAllowed()`
   Los componentes de ejecución determinan si el usuario tiene acceso a nivel de mandato para la vista o el mandato de controlador.
2. `getOwner()`
   El gestor de políticas de control de acceso determina el propietario del recurso a nivel de mandato. La implementación por omisión devuelve el identificador de miembros (`memberId`) del propietario de la tienda (`storeId`) que está en el contexto del mandato. Si no hay ningún identificador de tienda en el contexto del mandato, se devuelve la organización raíz (`-2001`).
3. `getApplicablePolicies()`
   El gestor de políticas de control de acceso busca y procesa las políticas aplicables, basadas en el usuario, la acción y el recurso especificados.
4. `validateParameters()`
   Efectúa la comprobación y resolución inicial de parámetros.
5. `getResources()`
   Devuelve un vector de acceso que es un vector de parejas recurso-acción.

   Si no se devuelve nada, la comprobación de control de acceso a nivel de recurso no se efectúa. Si hay recursos que deban protegerse, debe devolverse un vector de acceso (consistente en parejas recurso-acción).

   Cada *recurso* es una instancia de un objeto protegible (un objeto que implementa la interfaz `com.ibm.commerce.security.Protectable`). En muchos casos, el recurso es un bean de acceso.

   Un bean de acceso quizá no implemente la interfaz `com.ibm.commerce.security.Protectable`, sin embargo, la comprobación de control de acceso todavía puede tener lugar siempre y cuando el bean enterprise correspondiente esté protegido, según la información que se incluye en “Implementación del control de acceso en beans enterprise” en la página 89.

   La *acción* es una serie que representa la operación a efectuar en el recurso. En la mayoría de casos, la acción es el nombre de interfaz del mandato.
6. `isAllowed()`
   Los componentes de ejecución determinan si el usuario tiene acceso a nivel de recurso a todas las parejas recurso-acción especificadas por `getResources()`.
7. `getOwner()`
   El recurso devuelve el `memberId` de su propietario. Esto determina las políticas que se aplican. Sólo se aplican las políticas que posee el propietario del recurso y sus organizaciones padre.
8. `getApplicablePolicies()`
   El gestor de políticas de control de acceso busca las políticas aplicables y, a continuación, las aplica. Si, como mínimo, se encuentra una política por pareja recurso-acción que otorgue permiso al usuario para acceder al recurso, se otorga el acceso; de lo contrario, el acceso se deniega.
9. `fulfills()`
   Si una política aplicable tiene especificada un grupo de relaciones, se efectúa una comprobación en el recurso para ver si el miembro satisface la relación o relaciones especificadas con respecto al recurso.

10. `performExecute()`
    La lógica de negocio del mandato.

Interfaz protegible

Un factor clave para tener un recurso protegido por las políticas de control de acceso de WebSphere Commerce, es que el recurso debe implementar la interfaz `com.ibm.commerce.security.Protectable`. Esta interfaz se utiliza normalmente con beans enterprise y beans de datos, pero sólo los beans específicos que requieren protección deben implementar la interfaz.

Con la interfaz protegible, un recurso debe proporcionar dos métodos de clave: `getOwner()` y `fulfills(Long member, String relationship)`.

Los propietarios de las políticas de control de acceso son organizaciones o entidades de organización. El método `getOwner` devuelve el memberId del propietario del recurso protegible. Después de que el gestor de políticas de control de acceso determine el propietario del recurso, también obtiene el memberId de cada uno de los padres del propietario en la jerarquía de miembros. A continuación se aplican todas las políticas de control de acceso que pertenecen al propietario de la petición getOwner original, así como todas las políticas de control de acceso que pertenecen a uno de los padres del propietario.

Se aplican las políticas de control de acceso que se aplican al propietario especificado, así como las políticas de control de acceso que se aplican a cualquier padre del propietario de nivel superior en la jerarquía de miembros.

El método `fulfills` sólo devuelve true si el miembro dado satisface la relación requerida con respecto al recurso. Normalmente, el miembro es un solo usuario, pero también puede ser una organización. Sería una organización si utiliza un grupo de relaciones en la política de control de acceso.

Interfaz de agrupación

La aplicación de una política de control de acceso es específica a un grupo de recursos. Pueden agruparse recursos basándose en atributos como por ejemplo, el nombre de la clase, el estado de un pedido o el valor de storeId.

Si un recurso va a agruparse por un atributo que no sea el nombre de su clase, para aplicar políticas de control de acceso, el recurso debe implementar la interfaz `com.ibm.commerce.grouping.Groupable`.

La sección siguiente de código representa la Interfaz de agrupación:

```
Groupable interface {
 Object getGroupingAttributeValue (String attributeName, GroupContext context)
}
```

Por ejemplo, para implementar una política que sólo se aplica a pedidos que se encuentran en estado pendiente (`status = P (pendiente)`), la interfaz remota del bean de entidad de pedido implementa la interfaz de agrupación y el valor de attributeName se establece en `"status"`.

La utilización de la interfaz de agrupación es poco frecuente.

Información adicional sobre el control de acceso

Para obtener más información sobre el modelo de control de acceso de WebSphere Commerce, consulte la publicación *WebSphere Commerce, Guía de control de acceso*.

Esta guía proporciona una visión general detallada sobre el control de acceso y describe cómo utilizar la Consola de administración para crear o modificar políticas, grupos de acciones y grupos de recursos.

Implementación del control de acceso

Esta sección describe cómo implementar el control de acceso en código personalizado.

Identificación de recursos protegibles

En general, los beans enterprise y los beans de datos son recursos que quizá desee proteger. Sin embargo, no todos los beans enterprise ni beans de datos deben protegerse. Dentro de la aplicación WebSphere Commerce existente, los recursos que requieren protección ya implementan la interfaz protegible. La cuestión de qué proteger surge cuando crea nuevos beans enterprise y beans de datos. Los recursos que se han de proteger dependen de su aplicación.

Si un mandato devuelve un bean enterprise en el método `getResources`, el bean enterprise debe protegerse porque el gestor de políticas de control de acceso llamará al método getOwner en el bean enterprise. También se llamará al método `fulfills` si se especifica una relación en la política de control de acceso a nivel de recurso.

Si tuviera que implementar la interfaz protegible (y por tanto, poner el recurso bajo protección) en todos sus propios beans enterprise y beans de datos, su aplicación podría requerir muchas políticas. Al aumentar el número de políticas, el rendimiento puede disminuir y la gestión de políticas se vuelve más compleja.

Se hace una distinción teórica entre los recursos primarios y los dependientes. Un *recurso primario* puede existir por sí mismo. Un *recurso dependiente* sólo existe mientras exista su recurso primario relacionado. Por ejemplo, en el código de aplicación de WebSphere Commerce, tal como se entrega, el bean de entidad Order es un recurso protegible, pero el bean de entidad OrderItem no lo es. El motivo es que la existencia de un OrderItem depende de un Order -- Order es el recurso primario y OrderItem es un recurso dependiente. Si un usuario debe tener acceso a un pedido (Order), también debe tener acceso a los artículos del pedido.

De igual forma, el bean de entidad User (Usuario) es un recurso protegible, pero el bean de entidad Address (Dirección) no lo es. En ese caso, la existencia de la dirección depende del usuario, de manera que todos los que tengan acceso al usuario también deben tener acceso a la dirección.

Deben protegerse los recursos primarios, pero los recursos dependientes a menudo no requieren protección. Si a un usuario se le permite el acceso a un recurso primario, parece lógico que, por omisión, también se le permita acceder a sus recursos dependientes.

Implementación del control de acceso en beans enterprise

Si crea nuevos beans enterprise que requieren protección mediante políticas de control de acceso, debe hacer lo siguiente:

1. Crear un nuevo bean enterprise, asegurando que se amplía desde `com.ibm.commerce.base.objects.ECEntityBean`.
2. Asegurarse de que la interfaz remota del bean amplía la interfaz `com.ibm.commerce.security.Protectable`.

3. Si los recursos con los que interactúa el bean están agrupados mediante un atributo distinto del nombre de clase Java del recurso, la interfaz remota del bean también debe ampliar la interfaz `com.ibm.commerce.grouping.Groupable`.
4. La clase de bean enterprise contiene implementaciones por omisión para los métodos siguientes:
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

   Modifique los métodos que necesite. Como mínimo, debe modificar el método `getOwner`.
   Las implementaciones por omisión de estos métodos se muestran en las siguientes secciones de código.

```
*********************************************************************
public Long getOwner() throws Exception, java.rmi.RemoteException
{
   return null;
}
*********************************************************************
*********************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException
{
      return false;
}
*********************************************************************
*********************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception, java.rmi.RemoteException
{
      return null;
}
*********************************************************************
```

   A continuación se muestran implementaciones de ejemplo de estos métodos basados en el bean OrderBean:

```
*********************************************************************
public Long getOwner() throws Exception, java.rmi.RemoteException
{
   com.ibm.commerce.common.objects.StoreEntityAccessBean storeEntAB = new
   com.ibm.commerce.common.objects.StoreEntityAccessBean();
   storeEntAB.setInitKey_storeEntityId(getStoreEntityId().toString());
   return storeEntAB.getMemberIdInEJBType();
}
*********************************************************************
*********************************************************************
public boolean fulfills(Long member, String relationship)
   throws Exception, java.rmi.RemoteException
   {
      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      else if (relationship.equalsIgnoreCase (
         com.ibm.commerce.base.helpers.EJBConstants.
         SAME_ORGANIZATIONAL_ENTITY_AS_CREATOR_RELATION)) {
            com.ibm.commerce.user.objects.UserAccessBean creator = new
               com.ibm.commerce.user.objects.UserAccessBean();
            creator.setInitKey_MemberId(getMemberId().toString());
            com.ibm.commerce.user.objects.UserAccessBean ab = new
               com.ibm.commerce.user.objects.UserAccessBean();
            ab.setInitKey_MemberId(member.toString());
```

```
            if (ab.getParentMemberId().equals(creator.getParentMemberId()))
               return true;
         }
      return false;
}
*****************************************************************************
*****************************************************************************
public Object getGroupingAttributeValue(String attributeName,
   GroupingContext context) throws Exception
   {
      if (attributeName.equalsIgnoreCase("Status"))
         return getStatus();
   return null;
   }
*****************************************************************************
```

5. Crear (o volver a crear) el código generado y el bean de acceso del bean enterprise.

Implementación del control de acceso en beans de datos

Si ha de protegerse un bean de datos, puede protegerse directa o indirectamente mediante las políticas de control de acceso. Si un bean de datos se protege directamente, significa que hay una política de control de acceso que se aplica a ese bean de datos. Si un bean de datos se protege indirectamente, delega la protección en otro bean, para el que ya existe una política de control de acceso.

Si crea un nuevo bean de datos que debe protegerse directamente con una política de control de acceso, el bean de datos debe:

1. Implementar la interfaz `com.ibm.commerce.security.Protectable`. Para ello, el bean debe proporcionar una implementación de los métodos `getOwner()` y `fulfills(Long member, String relationship)`. Estos métodos deben implementarse en la interfaz remota del bean.

   Cuando un bean de datos implementa la interfaz Protectable, el gestor de beans de datos llama al método `isAllowed` para determinar si el usuario tiene los privilegios de control de acceso adecuados, según la política de control de acceso actual. El método `isAllowed` se describe con la siguiente sección de código:

   ```
   IsAllowed(Context, "Display", protectable_databean);
   ```

2. Si los recursos con los que interactúa el bean están agrupados mediante un atributo distinto del nombre de clase Java del recurso, el bean debe implementar la interfaz `com.ibm.commerce.grouping.Groupable`.
3. Implementar la interfaz `com.ibm.commerce.security.Delegator`. Esta interfaz se describe con la siguiente sección de código:

   ```
   Interface Delegator {
    Protectable getDelegate();
   }
   ```

   Nota: Para obtener una protección directa, el método `getDelegate` debe devolver el bean de datos (es decir, el bean de datos se delega a sí mismo para el control de acceso).

La distinción entre los beans de datos que deben protegerse directamente y los que deben protegerse indirectamente es parecida a la distinción entre recursos primarios y dependientes. Si el objeto de bean de datos puede existir por sí mismo, debe protegerse directamente. Si su existencia depende de la existencia de otro bean de datos, debe delegar la protección al otro bean de datos.

Un ejemplo de un bean de datos que debe protegerse directamente es el bean de datos Order (Pedido). Un ejemplo de un bean de datos que debe protegerse indirectamente es el bean de datos OrderItem.

Si crea un nuevo bean de datos que debe protegerse indirectamente con una política de control de acceso, el bean de datos debe:

1. Implementar la interfaz `com.ibm.commerce.security.Delegator`. Esta interfaz se describe con la siguiente sección de código:

```
Interface Delegator {
 Protectable getDelegate();
}
```

   Nota: El bean de datos que devuelve `getDelegate` debe implementar la interfaz Protectable.

Si un bean de datos no implementa la interfaz Delegator, se rellena sin la protección de las políticas de control de acceso.

Implementación del control de acceso en mandatos de controlador

Al crear un nuevo mandato de controlador, la clase de la implementación para el nuevo mandato debe ampliar la clase `com.ibm.commerce.commands.ControllerCommandImpl` y su interfaz debe ampliar la interfaz `com.ibm.commerce.command.ControllerCommand`.

Para las políticas a nivel de mandato para los mandatos de controlador, el nombre de interfaz del mandato se especifica como un recurso. Para que un recurso esté protegido, debe implementar la interfaz protegible. Según el modelo de programación de WebSphere Commerce, esto se consigue ampliando la interfaz del mandato a partir de la interfaz `com.ibm.commerce.command.ControllerCommand` y la implementación del mandato se amplía a partir de `com.ibm.commerce.commands.ControllerCommandImpl`. La interfaz `ControllerCommand` se amplía a partir de la interfaz `com.ibm.commerce.command.AccCommand` que, a su vez, se amplía a partir de la interfaz `Protegible`. La interfaz `AccCommand` es la interfaz mínima que debe implementar un mandato para estar protegido mediante el control de acceso a nivel de mandato.

Si el mandato accede a recursos que deben protegerse, cree una variable de instancia privada del tipo `AccessVector` para que contenga los recursos. A continuación, modifique el método `getResources` puesto que la implementación por omisión de este método es devolver un valor nulo y, por tanto, no se produce ninguna comprobación de recursos.

En el nuevo método `getResources`, debe devolver un conjunto de recursos o de parejas recurso-acción sobre los que el mandato puede actuar. Cuando una acción no se especifica de forma explícita, la acción toma el valor por omisión del nombre de interfaz del mandato que se ejecuta.

Adicionalmente, se recomienda que el método determine si debe crear una instancia del recurso o si puede utilizar la variable de instancia existente que contiene la referencia al recurso. La comprobación de si el objeto de recurso ya existe puede ayudar a mejorar el rendimiento del sistema. A continuación, puede utilizar el mismo método `getResources`, si fuera necesario, en el método `performExecute` del nuevo mandato de controlador.

A continuación se muestra un ejemplo del método getResources:

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {

   if (resources == null) {
      OrderAccessBean orderAB = new OrderAccessBean();
      orderAB.setInitKey_orderId(getOrderId().toString());
      resouces = new AccessVector(orderAB);
      }
   return resources;
}
```

Tomemos, como ejemplo, el mandato OrderItemUpdate. El método getResources de este mandato devuelve los objetos protegibles Order y User. Puesto que no se especifica la acción, toma el valor por omisión de la interfaz para el mandato OrderItemUpdate.

El método getResources puede devolver diversos recursos. Cuando esto sucede, para poder llevar a cabo la acción debe encontrarse una política que ofrezca acceso al usuario a todos los recursos especificados. Si un usuario tenía acceso a dos de tres recursos, la acción no podrá continuar (son necesarios tres de tres).

Si tiene que efectuar comprobación adicional de parámetros o resolución de parámetros en el mandato de controlador, puede utilizar el método validateParameters(). Esto es opcional.

Comprobación adicional a nivel de recurso

No siempre es posible determinar todos los recursos que necesitan protegerse, en el momento en que se llama al método getResources del mandato de controlador.

Si fuera necesario, un mandato de tarea puede también implementar un método getResources para devolver una lista de recursos, en los que puede actuar el mandato.

Otra manera de invocar la comprobación a nivel de recurso es realizar llamadas directas al gestor de políticas de control de acceso, utilizando el método checkIsAllowed(Object resource, String action). Este método está disponible para cualquier clase que se amplíe a partir de la clase com.ibm.commerce.command.AbstractECTargetableCommand. Por ejemplo, las clases siguientes se amplían de la clase AbstractECTargetableCommand:

- com.ibm.commerce.command.ControllerCommandImpl
- com.ibm.commerce.command.DataBeanCommandImpl

El método checkIsAllowed también está disponible para las clases que amplían la clase com.ibm.commerce.command.AbstractECCommand. Por ejemplo, la siguiente clase se amplía de la clase AbstractECCommand:

- com.ibm.commerce.command.TaskCommandImpl

A continuación se muestra la firma del método checkIsAllowed:

```
void checkIsAllowed(Object resource, String action)
   throws ECException
```

Este método emite una excepción ECApplicationException si el usuario actual no tiene permiso para efectuar la acción especificada en el recurso especificado. Si se otorga el acceso, el método simplemente regresa.

Control de acceso para los mandatos “create”

Puesto que se llama al método `getResources` antes que al método `performExecute` en un mandato, debe tomarse otro enfoque para el control de acceso de recursos que todavía no se han creado. Por ejemplo, si tiene un `WidgetAddCmd`, el método `getResources` no puede devolver el recurso que se va a crear. En ese caso, el método `getResources` debe devolver el creador de los recursos. Por ejemplo, un mandato lo crea una fábrica de mandatos, un pedido se crea dentro de una tienda y un usuario se crea dentro de una organización.

Implementaciones por omisión para control de acceso a nivel de mandato

Para el control de acceso a nivel de mandato, la implementación por omisión del método getOwner() devuelve el memberId del propietario de la tienda, si se especifica el storeId. Si no se especifica el storeId, se devuelve el memberId de la organización raíz (`memberId = -2001`).

La implementación por omisión del método `getResources()` devuelve `null`.

La implementación por omisión de `validateParameters()` no hace nada.

Implementación de políticas de control de acceso en vistas

El control de acceso a nivel de recurso para vistas lo efectúa el gestor de beans de datos. Se llama al gestor de beans de datos en los siguientes casos:

1. Cuando la plantilla JSP incluye el código `<useBean>` y el bean de datos no está en la lista de atributos.
2. Cuando la plantilla JSP incluye el método de activación siguiente:

   ```
   DataBeanManager.activate(xyzDatabean, request);
   ```

Nota: Los bean de datos que se vayan a proteger (tanto directa como indirectamente) deben implementar la interfaz Delegator. Los bean de datos que se vayan a proteger de forma directa, delegarán a sí mismos y, por tanto, deben implementar la interfaz protegible. Los beans de datos que se protejan indirectamente deben delegar a un bean de datos que implemente la interfaz protegible.

Aunque no se recomienda, se evitan las comprobaciones de control de acceso en los casos siguientes:

1. Si la plantilla JSP efectúa llamadas directas a los beans de acceso, en lugar de utilizar los beans de datos.
2. Si la plantilla JSP llama directamente al método populate() de bean de datos.

Si el resultado de un mandato de controlador debe redirigirse a una vista (utilizando ForwardViewCommand), no se lleva a cabo el control de acceso a nivel de mandato en las vistas. Además, si el mandato de controlador coloca los beans de datos con datos (que se utilizan en la vista) en la lista de atributos de la propiedad de respuesta y, a continuación, lo envía a una vista, la plantilla JSP puede acceder a los datos sin pasar por el gestor de beans de datos. Esto requiere la utilización de códigos `<useBean>` en la plantilla JSP. Esto puede ser una manera de hacer una plantilla JSP más eficaz, ya que puede saltarse las comprobaciones de control de acceso a nivel de recurso, de los recursos (beans de datos) a los que ya se ha dado acceso al usuario mediante el mandato de controlador.

Capítulo 5. Manejo de errores y mensajes

Manejo de errores de mandatos

WebSphere Commerce utiliza una infraestructura de manejo de errores de mandatos bien definida que es fácil de utilizar en el código personalizado. Por diseño, la infraestructura maneja los errores de manera que se da soporte a tiendas multiculturales. En las siguientes secciones se describen los tipos de excepciones que puede generar un mandato, cómo se manejan las excepciones, cómo se guarda y utiliza el texto de los mensajes, cómo se anotan las excepciones y cómo utilizar la infraestructura suministrada en sus propios mandatos.

Tipos de excepciones

Un mandato puede generar una de las siguientes excepciones:

ECApplicationException
Esta excepción se genera si el error está relacionado con el usuario. Por ejemplo, cuando un usuario entra un parámetro que no es válido, se genera una excepción ECApplicationException. Cuando se genera esta excepción, el controlador Web no vuelve a intentar ejecutar el mandato aunque se haya especificado como mandato que puede reintentarse.

ECSystemException
Esta excepción se genera si se detecta una excepción de ejecución o un error de configuración de WebSphere Commerce. Ejemplos de este tipo son las excepciones de puntero nulo y las excepciones de retrotracción de transacciones. Cuando se genera este tipo de excepción, el controlador Web vuelve a intentar ejecutar el mandato, si el mandato se puede reintentar y la excepción se generó debido a un punto muerto en la base de datos o una retrotracción en la base de datos.

Las dos excepciones anteriores son clases que provienen de la clase ECException, que está en el paquete com.ibm.commerce.exception.

Para generar una de estas excepciones, debe especificarse la siguiente información:

- Nombre de vista de error
  El controlador Web busca este nombre en la tabla VIEWREG.
- Objeto ECMessage
  Este valor corresponde al texto del mensaje incluido en un archivo de propiedades (.properties).
- Parámetros de error
  Estas parejas nombre-valor se utilizan para sustituir información en el mensaje de error. Por ejemplo, un mensaje puede incluir un parámetro que contenga el nombre del método que generó la excepción. Este parámetro se establece cuando se genera la excepción, y cuando se anota el mensaje de error, el archivo de anotaciones cronológicas contiene el nombre del método real.
- Datos de error
  Son atributos opcionales que puede utilizar la plantilla JSP a través del bean de datos de error.

El manejo de excepciones está integrado en el sistema de anotación cronológica. Cuando se genera una excepción, se anota automáticamente en el registro.

Archivos de propiedades de mensajes de error

Para simplificar el mantenimiento de los mensajes de error y para dar soporte a tiendas multilingües, el texto de los mensajes de error se guarda en archivos de propiedades. El texto de los mensajes de WebSphere Commerce se guarda en el archivo `ecServerMessages_XX_XX.properties`, donde `_XX_XX` es el indicador del entorno nacional (por ejemplo, `_es_ES`).

El contexto del mandato devuelve un identificador para indicar el idioma que utiliza el cliente. Cuando se necesita un mensaje, el controlador Web determina qué archivo de propiedades se debe utilizar según el identificador del idioma.

Hay dos tipos de mensajes definidos en el archivo `ecServerMessagesXX_XX.properties`: mensajes de usuario y mensajes del sistema. Los mensajes de usuario se muestran a los clientes en sus navegadores. Ambos tipos de mensajes se capturan automáticamente en el registro de mensajes.

Cuando se genera un error, uno de los parámetros necesarios es un objeto mensaje. Para ECSystemExceptions, el objeto mensaje debe contener dos claves, una para el mensaje del sistema y otra para el mensaje de usuario. Para ECApplicationExceptions, el objeto mensaje contiene la clave para el mensaje de usuario (los mensajes del sistema no se utilizan).

Todos los mensajes del sistema son predefinidos. No puede crear sus propios mensajes del sistema. Por lo tanto, cuando el código personalizado genera una excepción ECSystemException, debe especificar una clave de mensaje para uno de los mensajes predefinidos. Se pueden crear mensajes de usuario personalizados. Los nuevos mensajes de usuario pueden guardarse en un archivo de propiedades distinto.

Flujo del manejo de excepciones

El siguiente diagrama muestra el flujo de información cuando se detecta una excepción. A continuación se muestra una descripción de cada paso.

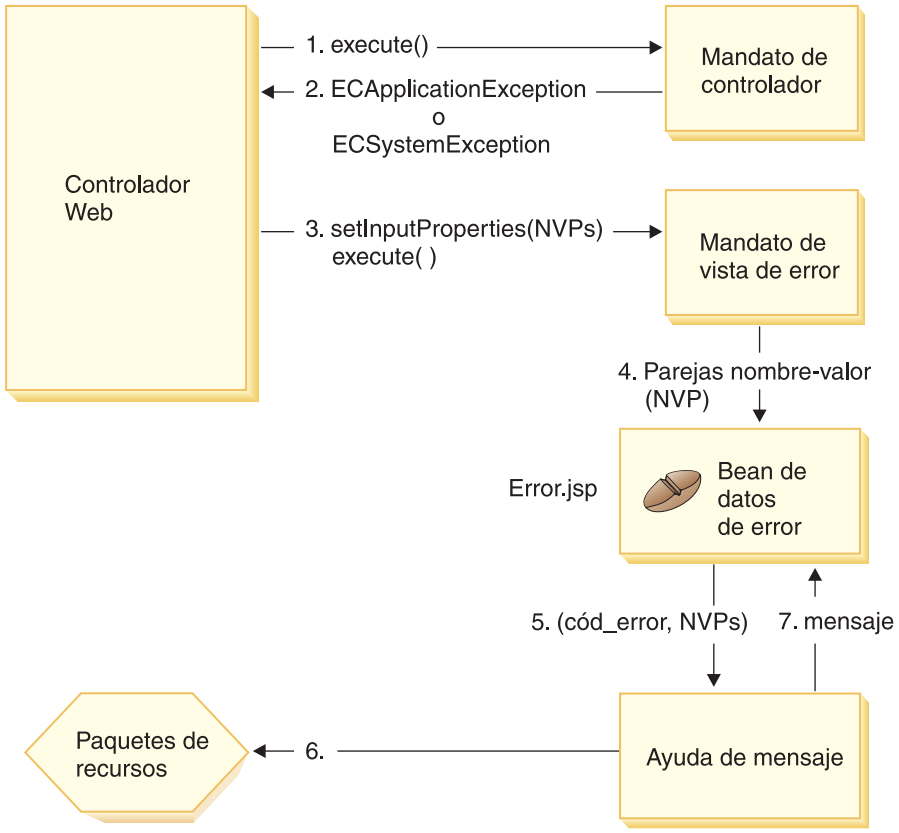

Figura 24.

1. El controlador Web invoca un mandato de controlador.
2. El mandato genera una excepción que detecta el controlador Web. Dicha excepción puede ser ECApplicationException o ECSystemException. El objeto excepción contiene la siguiente información:
   - Nombre de vista de error
   - Objeto ECMessage
   - Parámetros de error
   - Datos de error (opcionales)
3. El controlador Web determina el nombre de la vista de error de la tabla VIEWREG e invoca el mandato de vista de error especificado. Al invocar el mandato, el controlador Web crea un conjunto de propiedades a partir de un objeto ECException y lo establece en el mandato de vista mediante el método setInputProperties del mandato de vista.
4. El mandato de vista invoca una plantilla JSP de error (Error.jsp en este caso) y las parejas de nombre-valor se pasan a la plantilla JSP.
5. El bean de datos de error pasa los parámetros de error al objeto de ayuda de mensaje.
6. El objeto de ayuda de mensaje obtiene el mensaje necesario (utilizando el objeto mensaje y los parámetros de error) del archivo de propiedades correspondiente.
7. El bean de datos de error devuelve el mensaje a la plantilla JSP.

Manejo de excepciones en el código personalizado

Al crear nuevos mandatos es importante incluir el manejo de excepciones adecuado. Puede beneficiarse de la infraestructura de manejo de errores y de

mensajería que se proporciona en WebSphere Commerce, especificando la información necesaria cuando se detecta una excepción.

Al crear su propia lógica de manejo de excepciones debe tener en cuenta lo siguiente:

1. Deben detectarse las excepciones del mandato que necesitan un proceso especial.
2. Debe crearse una excepción ECApplicationException o una ECSystemException, según el tipo de excepción detectada.
3. Si la excepción ECApplicationException utiliza un nuevo mensaje, debe definirse el mensaje en el nuevo archivo de propiedades.

Detección y creación de excepciones

Para ilustrar los dos primeros pasos, se muestra la siguiente sección de código que es un ejemplo de cómo se detecta una excepción del sistema en un mandato:

```
try {
// su lógica de negocio
}
catch(FinderException e) {
     throw new ECSystemException (ECMessage._ERR_FINDER_EXCEPTION,
          className, methodName, new Object [] {e.toString()}, e);
}
```

El objeto ECMessage _ERR_FINDER_EXCEPTION se define de la forma siguiente:

```
public static final ECMessage _ERR_FINDER_EXCEPTION =
new ECMessage (ECMessageSeverity.ERROR, ECMessageType.SYSTEM,
     ECMessageKey._ERR_FINDER_EXCEPTION);
```

El texto del mensaje _ERR_FINDER_EXCEPTION se define dentro del archivo `ecServerMessages_xx_XX.properties` (donde *_xx_XX* es un indicador del entorno nacional como, por ejemplo, *_es_ES*) de la forma indicada a continuación:

```
_ERR_FINDER_EXCEPTION  =
     Se ha producido la siguiente excepción de búsqueda al procesar: "{0}".
```

Al detectarse una excepción del sistema, puede utilizarse un conjunto predefinido de mensajes. Dichos mensajes se describen en la siguiente tabla:

Objeto de mensaje	Descripción
_ERR_FINDER_EXCEPTION	Se genera cuando se devuelve un error desde una llamada de método de buscador EJB.
_ERR_REMOTE_EXCEPTION	Se genera cuando se devuelve un error desde una llamada de método remoto EJB.
_ERR_CREATE_EXCEPTION	Se genera cuando se produce un error al crear una instancia EJB.
_ERR_NAMING_EXCEPTION	Se genera cuando se devuelve un error desde el servidor de nombres.
_ERR_GENERIC	Se genera cuando se produce un error inesperado del sistema. Por ejemplo, una excepción de puntero nulo.

Al detectar una excepción de aplicación, puede utilizar un mensaje existente que esté especificado en el correspondiente archivo `ecServerMessages_xx_xx.properties` o crear un nuevo mensaje que se guarda en un nuevo archivo de propiedades. Tal como se ha especificado anteriormente, *no debe* modificar ninguno de los archivos `ecServerMessages_xx_XX.properties`.

En el siguiente extracto de código se muestra un ejemplo de cómo se detecta una excepción de aplicación en un mandato:

```
try {
// su lógica de negocio

}
// detectar algún nuevo tipo de excepción de aplicación
catch(//la nueva excepción)
 {
     throw new ECApplicationException (MyMessages._ERR_CUSTOMER_INVALID,
          className, methodName, errorTaskName, someNVPs);
}
```

El objeto ECMessage _ERR_CUSTOMER_INVALID anterior se define de la forma siguiente:

```
public static final ECMessage _ERR_CUSTOMER_INVALID =
     new ECMessage (ECMessageSeverity.ERROR, ECMessageType.USER,
     MyMessagesKey._ERR_CUSTOMER_INVALID, "ecCustomerMessages");
```

Al crear nuevos mensajes de usuario, debe asignarles el tipo USER de la forma indicada a continuación:

```
ECMessageType.USER
```

El texto del mensaje _ERR_CUSTOMER_INVALID está incluido en el archivo `ecCustomerMessages.properties`. Este archivo debe estar en un directorio que esté indicado en la variable de entorno CLASSPATH. El texto se define de la forma indicada a continuación:

```
_ERR_CUSTOMER_INVALID = Invalid ID "{0}"
```

Creación de mensajes

Si el mandato genera una excepción ECApplicationException que utiliza un nuevo mensaje, debe crear dicho mensaje. Para crear un nuevo mensaje debe efectuar los siguientes pasos:

1. Crear una nueva clase que contenga las claves de mensaje.
2. Crear una nueva clase que contenga los objetos ECMessage.
3. Crear el paquete de recursos.
4. Comprobar el mensaje.

En las secciones siguientes se proporcionan más detalles sobre cada paso.

Creación de una clase para claves de mensaje

El primer paso necesario para crear nuevos mensajes de usuario es crear una clase que contenga las nuevas claves de mensaje. Una clave de mensaje es un indicador exclusivo que el servicio de anotación cronológica utiliza para localizar el texto de mensaje correspondiente en un paquete de recursos. Esta nueva clase debe crearse dentro de su propio paquete y guardarse en un proyecto separado de los proyectos de WebSphere Commerce.

Suponga un ejemplo, denominado `MyNewMessages`, en el que crea una nueva clase, denominada `MyMessageKeys` que contiene las claves de mensaje `_ERR_CUSTOMER` y `_ERR_CUSTOMER_INVALID_ID` y pone esta clase en el paquete `com.mycompany.messages`. En este caso, la definición de la clase aparece de la forma indicada a continuación:

```
public class MyMessageKeys
{
    public static String _ERR_CUSTOMER="_ERR_CUSTOMER";
    public static String _ERR_CUSTOMER_INVALID_ID="_ERR_CUSTOMER_INVALID_ID";
}
```

Si se proporcionan wrappers de tipo String para las claves de mensaje, el compilador podrá comprobar su validez.

Creación de una clase para objetos ECMessage

En el mismo paquete en el que ha creado la clase para las claves de mensaje, cree otra clase que contenga los objetos ECMessage. La clase ECMessage define la estructura de un objeto mensaje. Se utiliza para recuperar mensajes de texto sensibles al entorno nacional y hacer que sean persistentes.

El objeto mensaje tiene los siguientes atributos: gravedad, tipo, clave, paquete de recursos y paquete de recursos asociado. Hay varios métodos de constructor para esta clase. Para obtener más información, consulte la sección "Referencias" de la ayuda en línea de WebSphere Commerce.

Siguiendo con el ejemplo de MyNewMessages, cree una nueva clase denominada `MyMessages` en el paquete `com.mycompany.messages`, de la forma indicada a continuación:

```
import com.ibm.commerce.ras.*;
public class MyMessages
    {
        static String myResourceBundle = "ecCustomerMessages";

        public static ECMessage _ERR_CUSTOMER = new ECMessage
             (ECMessageSeverity.ERROR,ECMessageType.USER,
                   MyMessageKeys._ERR_CUSTOMER, myResourceBundle);
        public static ECMessage _ERR_CUSTOMER_INVALID_ID = new ECMessage
             (ECMessageSeverity.ERROR, ECMessageType.USER,
                  MyMessageKeys._ERR_CUSTOMER_INVALID_ID,
                  myResourceBundle);

    }
```

En la sección de código anterior, la sentencia import es necesaria para crear el objeto ECMessage. El objeto MyMessage._ERR_CUSTOMER es un mensaje de usuario de gravedad `ERROR`. El servicio de anotación cronológica de WebSphere Commerce utiliza MyMessageKeys._ERR_CUSTOMER para buscar el texto de mensaje incluido en el archivo de propiedades *ecCustomerMessages*.

Creación del paquete de recursos de mensajes de usuario

Debe crear un nuevo paquete de recursos, en el que se guarden las claves de mensaje con su texto de mensaje correspondiente. Este paquete de recursos puede implementarse como un objeto Java o como un archivo de propiedades (.properties). Se recomienda utilizar archivos de propiedades ya que son más fáciles de traducir y mantener. Los archivos de propiedades se utilizan para los mensajes de WebSphere Commerce.

Para continuar con el ejemplo MyNewMessages, debe crear un archivo de texto con el nombre `ecCustomerMessages.properties`. Si los mensajes los va a utilizar un único servlet de tienda, coloque este archivo en el siguiente directorio:

unidad:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*nombre_instancia*.ear\
wcstores.war\WEB-INF\classes

Si los mensajes los va a utilizar un único servlet de herramientas, coloque este archivo en el siguiente directorio:

unidad:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*nombre_instancia*.ear\
wctools.war\WEB-INF\classes

Si los mensajes los utiliza globalmente algún servlet de la aplicación de empresa, coloque el archivo en el siguiente directorio:

unidad:\WebSphere\AppServer\installedApps\WC_EnterpriseApp_*nombre_instancia*.ear\
properties

Puesto que el archivo de propiedades contiene parejas de claves de mensaje y el texto de mensaje correspondiente, el archivo ecCustomerMessages.properties contiene las siguientes líneas:

```
_ERR_CUSTOMER_MESSAGE = El mensaje de cliente "{0}".
_ERR_CUSTOMER_INVALID_ID = ID no válido "{0}".
```

Comprobación de un mensaje

Después de crear las clases que contienen las claves de mensaje, las clases que contienen los objetos ECMessage, y el paquete de recursos, debe comprobar cada uno de los nuevos mensajes.

Para comprobar los nuevos mensajes descritos en las secciones anteriores, haga lo siguiente:

1. Cree una nueva clase con fines de comprobación. En este caso, cree MyTestingClass.
2. Añada la siguiente sentencia import a la clase

   ```
   import com.ibm.commerce.ras.*;
   ```
3. Añada un método main() a la clase.
4. Examine la siguiente sección de código. Modifíquelo según sus propios requisitos (por ejemplo, asegúrese de que la vía de acceso del directorio especifique un archivo de configuración válido en el sistema) e insértelo en el método main()

   ```
   // la variable String fileName debe especificar
   // un archivo de configuración válido de WebSphere Commerce:
     String fileName = "E:\\WebSphere\\CommerceServer\\instances
         \\demo\\xml\\demo.xml";

      LogConfiguration config = LogConfiguration.getUniqueInstance ();
      config.initialize (fileName, "testClone");

      ECMessageLog.out (MyMessage._ERR_CUSTOMER,
           "MyTestingClass", "main", "Hola");
   ```

   Las tres primeras líneas de código inicializan el servicio de anotación cronológica de WebSphere Commerce. La última línea de código indica a ECMessageLog que imprima el mensaje MyMessage._ERR_CUSTOMER. MyTestingClass y main forman parte del formato de rastreo del registro. La serie Hola se coloca en el espacio reservado {0} del mensaje definido en el archivo ecCustomerMessages.properties.
5. Ejecute la clase. En el archivo de anotaciones, el rastreo de mensajes es parecido al que se indica a continuación:

   ```
   ==============
    TimeStamp:      29/11/2000 16:41:42.5
    Thread ID:     <main>
    Class:          MyTestingClass
    Method:         main
    Severity:       1
    Message Text:   El mensaje de cliente "Hola".
   ```

Puede ejecutar una prueba similar para ver el segundo mensaje.

Rastreo del flujo de ejecución

WebSphere Commerce incluye la clase ECTrace que se utiliza para rastrear el flujo de ejecución de componentes que se ejecutan en el WebSphere Commerce Server. La clase ECTrace forma parte del paquete com.ibm.commerce.ras.

Al crear nueva lógica de negocio, se puede insertar un rastreo dentro del código para rastrear un método con fines de depuración. La información del rastreo se captura en el registro de anotaciones de rastreo. Puede especificar un punto de entrada y de salida para el rastreo. Además, puede especificar el rastreo de datos concretos entre estos dos puntos.

Para utilizar el rastreo, éste debe estar habilitado para el componente para el que desee ejecutar el rastreo. Para habilitar el rastreo para un componente concreto, puede utilizar la Consola de administración o el Gestor de configuración.

Al efectuar el rastreo de código personalizado, *debe* utilizar el componente EXTERN. En el Gestor de configuración se denomina *External*.

Para establecer en el código el punto de entrada para un rastreo, utilice la sintaxis siguiente:

```
ECTrace.entry (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName, myMethodName);
```

donde *myClassName* es la representación de serie de caracteres de la clase que contiene el método del que se ha efectuado el rastreo. Puesto que esta serie de caracteres puede utilizarse para rastrear el análisis de archivos, debe incluir el nombre de clase totalmente calificado. Si el método del que se está efectuando el rastreo es estático, una declaración de ejemplo de myClassName es

```
String myClassName = "com.mycompany.agrouping.MyTracedClass";
```

Si el método del que se está efectuando el rastreo no es estático, una declaración de ejemplo de myClassName es

```
String myClassName = this.getClass().getName();
```

Para establecer el punto de rastreo para rastrear los datos de un método, utilice la sintaxis siguiente:

```
ECTrace.trace (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName, myText);
```

donde *myText* es el texto que debe aparecer en el registro de anotaciones de rastreo.

Para establecer en el código el punto de salida para un rastreo, utilice la sintaxis siguiente:

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName);
```

Si necesita efectuar un rastreo del objeto devuelto por el método del que se está efectuando el rastreo, establezca el punto de salida de la forma indicada a continuación:

```
ECTrace.exit (ECTraceIdentifiers.COMPONENT_EXTERN, myClassName,
     myMethodName, returnedObject);
```

donde `returnedObject` representa el objeto Java devuelto por el método.

Consideremos un ejemplo en el que sea necesario efectuar un rastreo del método performExecute de un nuevo mandato de controlador llamado

MyNewControllerCmd. La siguiente sección de código muestra cómo utilizar los métodos ECTrace dentro del método performExecute.

```
public void performExecute() throws ECException {
     ECTrace.entry(ECTraceIdentifiers.COMPONENT_EXTERN,
          this.getClass().getName(), "performExecute");

     super.performExecute();


//////////////////////////////////////////
//    Parte de su lógica de negocio       //
//////////////////////////////////////////


     ECTrace.trace(ECTraceIdentifiers.COMPONENT_EXTERN,
          this.getClass().getName(), "performExecute",
          "Mi código es bueno.");

//////////////////////////////////////////
//    Más lógica de negocio               //
//////////////////////////////////////////


     ECTrace.exit(ECTraceIdentifiers.COMPONENT_EXTERN,
          this.getClass().getName(), "performExecute");

}
```

Cuando se llame al método performExecute anterior, el archivo de anotaciones cronológicas de rastreo capturará la siguiente información:

```
==============
TimeStamp:     2000-12-05 17:32:00.257
Thread ID:    <P=502832:O=0:CT>
Component:     EXTERN
Class:         com.mycompany.agrouping.MyNewControllerCmd
Method:        performExecute
Trace:         ENTRY POINT
==============
TimeStamp:     2000-12-05 17:32:00.257
Thread ID:    <P=502832:O=0:CT>
Component:     EXTERN
Class:         com.mycompany.agrouping.MyNewControllerCmd
Method:        performExecute
Trace:         Mi código es bueno.
==============
TimeStamp:     2000-12-05 17:32:00.258
Thread ID:    <P=502832:O=0:CT>
Component:     EXTERN
Class:         com.mycompany.agrouping.MyNewControllerCmd
Method:        performExecute
Trace:         EXIT POINT
```

Se recomienda utilizar el rastreo sólo en las principales funciones. El rastreo no está habilitado para varios idiomas puesto que está pensado para que lo utilicen los desarrolladores de tienda, en contraposición a los mensajes, que están habilitados para varios idiomas porque los mensajes del sistema se utilizan con fines de administración y los mensajes de usuario se muestran a los clientes.

Manejo de errores de las plantillas JSP

El manejo de errores para las plantillas JSP puede realizarse de varias maneras:

- Manejo de errores desde dentro de la página
  Para los archivos JSP que requieren un manejo y una recuperación de errores más complejo, puede escribir el archivo para que maneje directamente los errores del bean de datos. El archivo JSP puede detectar las excepciones generadas por el bean de datos o puede buscar códigos de error establecidos en cada bean de datos, dependiendo de cómo se haya activado el bean de datos. A continuación, el archivo JSP puede llevar a cabo la acción de recuperación adecuada según el código de error recibido. Tenga en cuenta que un archivo JSP puede utilizar cualquier combinación de los siguientes ámbitos de manejo de errores.
- JSP de error a nivel de página
  Un archivo JSP también puede especificar su propia plantilla JSP de error por omisión procedente de una excepción que se produce dentro de sí misma a través del identificador de error de JSP. Esto permite a un programa JSP especificar su propio manejo de un error. Un archivo JSP que no especifique un identificador de error JSP sufrirá un error que se transmitirá a la plantilla de error JSP a nivel de aplicación. En la JSP de error a nivel de página, se debe llamar a la clase de ayuda JSP (com.ibm.server.JSPHelper) para retrotraer la transacción actual.
- JSP de error a nivel de aplicación
  Una aplicación que se ejecute bajo WebSphere puede especificar una plantilla JSP de error por omisión cuando se produce una excepción desde cualquiera de sus servlets o archivos JSP. La plantilla JSP de error a nivel de aplicación puede utilizarse como un manejador de errores a nivel de centro comercial o a nivel de tienda (para un modelo de tienda única). En la plantilla JSP de error a nivel de aplicación, se debe llamar a la clase de ayuda de servlet para retrotraer la transacción actual. Esto se debe a que el controlador Web no estará en la vía de ejecución para retrotraer la transacción. Siempre que sea posible debe utilizar los dos primeros tipos de manejo de errores JSP. Utilice la estrategia de manejo de errores a nivel de aplicación sólo cuando sea necesario.

Capítulo 6. Implementación de mandatos

Este capítulo proporciona información sobre cómo escribir nuevos mandatos de controlador, tareas y bean de datos. También describe cómo ampliar los mandatos de controlador, tareas y bean de datos ya existentes.

Nota: Business Este capítulo no describe los mandatos de política de negocios. Para obtener información sobre estos mandatos, consulte el Capítulo 7, "Acuerdos comerciales y políticas de negocio (Business Edition)" en la página 125.

Nuevos mandatos - Introducción

El modelo de programación de WebSphere Commerce define cuatro tipos de mandatos: mandatos de controlador, de tarea, de vista y de bean de datos. Cuando crea nueva lógica de negocio para la aplicación de comercio electrónico, está previsto que sea necesario crear nuevos mandatos de controlador, de tarea y de bean de datos. No debería ser necesario crear nuevos mandatos de vista. Más adelante en esta sección encontrará más información sobre los mandatos de vista.

Los nuevos mandatos deben implementar su interfaz correspondiente (que a su vez deben ampliarse de una interfaz existente). Para simplificar la escritura de mandatos, WebSphere Commerce incluye una clase de implementación abstracta para cada tipo de mandato. Los nuevos mandatos deben derivarse de estas clases.

La siguiente tabla proporciona una visión general sobre la clase de implementación de la que debe derivarse un nuevo mandato y qué interfaz debe implementar:

Tipo de mandato	Nombre de mandato de ejemplo	Se obtiene de	Implementa interfaz de ejemplo
Mandato de controlador	MyControllerCmdImpl	com.ibm.commerce.command.ControllerCommandImpl	MyControllerCmd
Mandato de tarea	MyTaskCmdImpl	com.ibm.commerce.command.TaskCommandImpl	MyTaskCmd
Mandato de bean de datos	MyDataBeanCmdImpl	com.ibm.commerce.command.DataBeanCommandImpl	MyDataBean

Nota: Todos los espacios que aparecen en las clases de implementación sólo se muestran a efectos de presentación.

En el siguiente diagrama se muestra la relación entre la interfaz y la clase de implementación de un nuevo mandato de controlador y la interfaz y clase de implementación abstracta existente. La clase abstracta y la interfaz están en el paquete com.ibm.commerce.command.

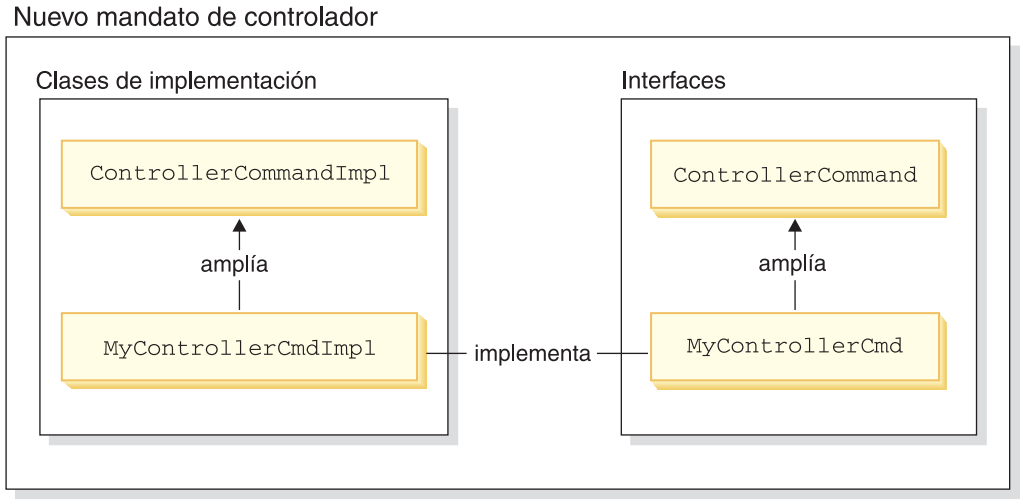

Figura 25.

En el siguiente diagrama se muestra la relación entre la interfaz y la clase de implementación de un nuevo mandato de tarea y la interfaz y clase de implementación abstracta existente. La clase abstracta y la interfaz están en el paquete com.ibm.commerce.command.

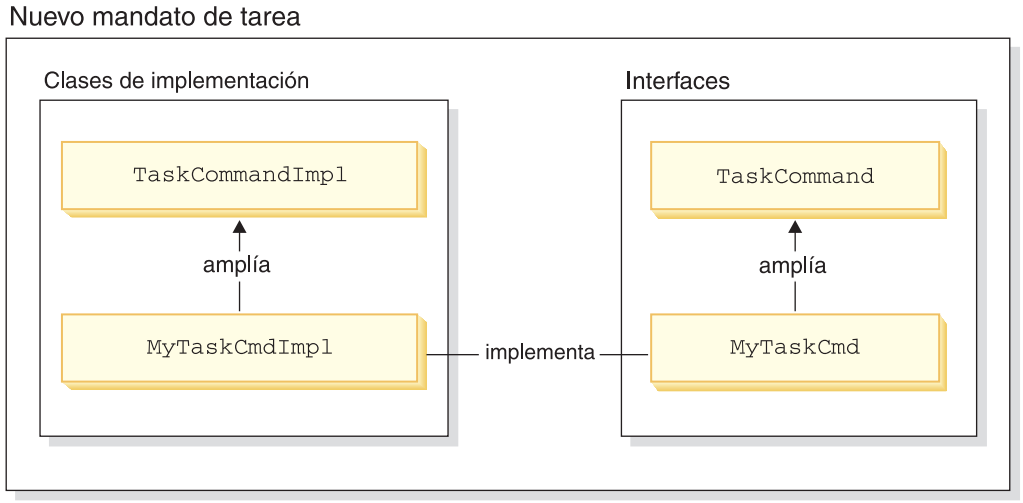

Figura 26.

En el siguiente diagrama se muestra la relación entre la interfaz y la clase de implementación de un nuevo mandato de bean de datos y la interfaz y clase de implementación abstracta existente. La clase abstracta y la interfaz están en el paquete com.ibm.commerce.command.

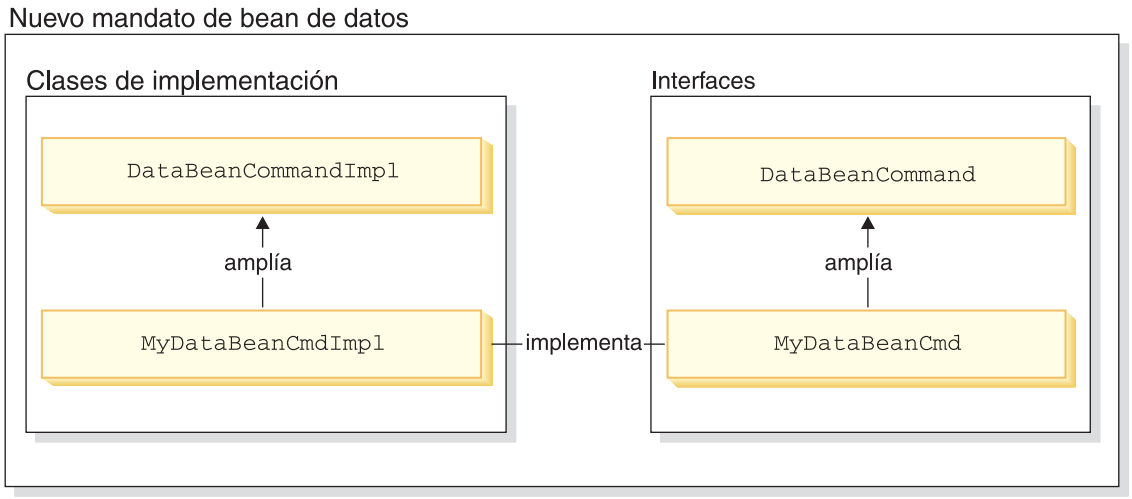

Figura 27.

Un mandato de vista tiene dos funciones principales: dar formato a una respuesta y enviar la respuesta al cliente. Se proporcionan varios mandatos de vista genéricos que devuelven la respuesta a los clientes utilizando distintos protocolos. La función de formato la gestiona normalmente el mandato de vista al invocar una plantilla JSP. Por ejemplo, el mandato de vista RedirectViewCommand dirige el cliente a un URL para que obtenga la respuesta (la respuesta la formatea una plantilla JSP especificada). El mandato de vista ForwardViewCommand reenvía la petición a la plantilla JSP para su formato y la página se muestra al cliente.

Con este modelo de mandato de vista puede crear nuevas *vistas* (la respuesta al cliente) creando nuevas plantillas JSP. No obstante, la plantilla JSP debe ser invocada por uno de los mandatos de vista existentes.

Creación de paquetes de código personalizado

Al crear código personalizado, se debe seguir una estructura de organización de código determinada. En general, el código personalizado se mantiene en paquetes y proyectos separados de los incluidos con WebSphere Commerce.

Al crear nuevos mandatos, debe ponerlos en un paquete cuyo nombre sea adecuado para los requisitos de su negocio. Es decir, si los mandatos se refieren a una tienda concreta, póngalos en un paquete que sea exclusivo para la tienda. Si se aplican a más de una tienda, cree el paquete según proceda. Por ejemplo, podría tener los siguientes paquetes:

- `com.bigbusiness.storeA.commands`
- `com.bigbusiness.storeB.commands`
- `com.bigbusiness.commands`

La estructura de paquetes anterior permite la distinción entre lógica de negocio a nivel de tienda. Además, estos paquetes deben guardarse en un proyecto separado de los proyectos de WebSphere Commerce. Por ejemplo, los paquetes anteriores pueden ponerse en un proyecto llamado `BigBusinessCustomCode`.

Al crear beans de datos nuevos, éstos deben guardarse en un paquete independiente de la lógica de mandatos, aunque este paquete puede guardarse

dentro del proyecto que almacena los paquetes de mandatos. Siguiendo con el ejemplo anterior, podría poner el paquete `com.bigbusiness.databeans` dentro del proyecto `BigBusinessCustomCode`.

Al crear beans de entidad nuevos, éstos deben almacenarse en un proyecto exclusivo. Por lo tanto, podría tener el proyecto `BigBusinessCustomEntityBeans` que incluyera el paquete `com.bigbusiness.objects`.

Esta estrategia de creación de paquetes es necesaria a efectos de despliegue del código.

Contexto de mandatos

El contexto de mandatos es un descriptor de contexto (handle) para el controlador Web. Los mandatos pueden obtener información del controlador Web utilizando el contexto de mandatos. Ejemplos de información disponible son el ID de usuario, el objeto usuario, el ID de idioma y el ID de tienda.

Cuando se escribe un mandato, se tiene acceso al contexto de mandatos llamando al método getCommandContext() de la superclase del mandato. El contexto de mandatos se establece en el valor del mandato de controlador cuando el controlador Web invoca el mandato. Un mandato de controlador debe propagar el contexto de mandatos a todos los mandatos de controlador o de tarea que se invocan durante el proceso. Un mandato puede obtener la siguiente información clave del contexto de mandatos:

getUserId() y getUser()
Obtiene el ID de usuario o el objeto usuario actual. El ID de usuario para la sesión actual se guarda en un contexto de sesiones. El contexto de sesiones puede persistir de dos maneras: utilizando el cookie de WebSphere Commerce o un objeto de sesión persistente de WebSphere Application Server. El contexto de mandatos oculta la complejidad de la gestión de sesiones de un mandato.

getStoreId(), getStore() y getStore(storeId)
Obtiene la tienda asociada a la petición actual. El controlador Web devuelve el ID de tienda del URL. Si el ID de tienda no está especificado en el URL, puede recuperarse del objeto sesión guardado en la petición anterior. El entorno de ejecución de WebSphere Commerce mantiene un conjunto de objetos a los que se accede con frecuencia. Por ejemplo, mantiene el conjunto de objetos tienda. Un mandato siempre debe obtener el objeto tienda del contexto de mandatos para aprovechar al máximo la antememoria de objetos del controlador Web. Puede obtener la tienda actual llamando al método getStore() u obtener un objeto tienda concreto llamando al método getStore(storeId) desde el contexto de mandatos.

getLanguageId()
Devuelve el ID de idioma que se debe utilizar para la solicitud actual. El controlador Web implementa una estructura de globalización. El concepto detrás de esta estructura es determinar un idioma que sea el preferido del usuario y esté soportado por la tienda. Si el URL contiene un ID de idioma, el controlador Web determina si este idioma está soportado por la tienda y, en caso afirmativo, el método getLanguageId() devuelve este ID de idioma. Si no se incluye ningún ID de idioma en el URL, el controlador Web utiliza un árbol de decisiones para determinar si hay un ID de idioma

(que esté soportado por la tienda) en el objeto de sesión actual, o en las preferencias registradas del usuario; de lo contrario, devolverá el ID de idioma por omisión de la tienda.

getCurrency()
Devuelve la moneda a utilizar para la solicitud actual. Puesto que la moneda forma parte de la estructura de globalización, la lógica detrás de este método es parecida a la del método getLanguageId().

getCurrentTradingAgreements() y getTradingAgreement(tradingAgreementId)
Devuelve el conjunto de acuerdos de comercio que se utilizan para la sesión actual. Este conjunto puede incluir todos los acuerdos de comercio a los que tiene derecho el usuario o puede ser un subconjunto definido por el mandato ContractSetInSession. Un mandato siempre debe obtener el objeto acuerdo de comercio del contexto de mandatos para aprovechar al máximo la antememoria de objetos del controlador Web. Puede obtener el acuerdo de comercio actual llamando al método getCurrentTradingAgreements() u obtener un objeto de acuerdo de comercio específico llamando al método getTradingAgreement(tradingAgreementId) desde el contexto de mandatos.

El contexto de mandatos debe utilizarse como un objeto de sólo lectura. No debe llamar a sus métodos set. Los métodos set están reservados para uso del entorno de ejecución de WebSphere Commerce y pueden dejar de utilizarse en futuros releases.

Para obtener más información sobre la API (interfaz de programación de aplicaciones) del contexto de mandatos, consulte la sección "Referencias" de la ayuda en línea de WebSphere Commerce.

Nuevos mandatos de controlador

Tal como se ha indicado anteriormente, un nuevo mandato de controlador debe ampliarse de la clase de mandatos de controlador abstracta (com.ibm.commerce.command.ControllerCommandImpl). Al escribir un nuevo mandato de controlador, debe alterar temporalmente los siguiente métodos de la clase abstracta:

- isGeneric()
- isRetriable()
- setRequestProperties(com.ibm.commerce.datatype.TypedProperty reqParms)
- validateParameters()
- getResources()
- performExecute()

En las siguientes secciones encontrará más información sobre los métodos antes citados.

Método isGeneric

En la implementación estándar de WebSphere Commerce hay varios tipos de usuarios. Estos son: usuarios genéricos, invitados o registrados. Dentro de la agrupación de usuarios registrados hay clientes y administradores.

El usuario genérico tiene una identificación de usuario común que se utiliza en todo el sistema. Esta identificación de usuario da soporte al examen general del sitio, de una forma que se minimiza la utilización de recursos del sistema. Es más

eficaz utilizar este ID de usuario común para examinar el sitio puesto que el controlador Web no necesita recuperar un objeto usuario para mandatos que el usuario genérico puede invocar.

El método isGeneric devuelve un valor booleano que especifica si el usuario genérico puede o no invocar el mandato. El método isGeneric de una superclase del mandato de controlador establece el valor en `false` (lo que significa que la persona que invoca debe ser un usuario registrado o un usuario invitado). Si los usuarios genéricos pueden invocar el nuevo mandato de controlador, altere este método para que devuelva el valor `true`.

Debe alterar este método de modo que devuelva `true` si el nuevo mandato no recopila ni crea recursos asociados con un usuario. Un ejemplo de un mandato que puede invocar un usuario genérico es el mandato ProductDisplay. Permite que cualquier usuario pueda ver los productos. Un ejemplo de un mandato para el que un usuario debe ser un usuario invitado o un usuario registrado (y, por lo tanto, isGeneric devuelve `false`) es el mandato OrderItemAdd.

Cuando isGeneric devuelve el valor `true`, el controlador Web no crea un nuevo objeto usuario para la sesión actual. Como tal, los mandatos que puede invocar el usuario genérico se ejecutan más rápidamente, puesto que el controlador Web no necesita recuperar un objeto usuario.

A continuación se muestra la sintaxis para utilizar este método de modo que permita a los usuarios genéricos invocar un mandato:

```
public boolean isGeneric()
{
     return true;
}
```

Método isRetriable

El método isRetriable devuelve un valor booleano que especifica si el mandato puede o no recuperarse con una excepción de retrotracción de transacción. El método isRetriable de la clase del nuevo mandato de controlador devuelve el valor `false`. Si puede repetirse la ejecución del mandato en una excepción de retrotracción de transacción, debe alterar temporalmente este método y devolver el valor `true`.

Un ejemplo de un mandato que no debe reintentar su ejecución en el caso de una excepción de transacción es el mandato OrderProcess. Este mandato invoca el proceso de autorización de pago de terceros. No puede volver a procesarse puesto que la autorización no puede anularse. Un ejemplo de un mandato que puede volver a intentarse es el mandato ProductDisplay.

A continuación se muestra la sintaxis para habilitar el mandato de modo que pueda volver a procesarse en el caso de una excepción de retrotracción de transacción:

```
public boolean isRetriable()
{
   return true;
}
```

Método setRequestProperties

El método setRequestProperties lo invoca el controlador Web para pasar todas las propiedades de entrada al mandato de controlador. El mandato de controlador debe analizar las propiedades de entrada y establecer de manera explícita cada

propiedad incluida en este método. Esta definición explícita de las propiedades por el mandato de controlador fomenta el concepto de propiedades de tipo seguro.

A continuación se muestra la sintaxis para utilizar este método:

```
public void setRequestProperties(
   com.ibm.commerce.datatype.TypedProperty reqParms)
{

     // analizar propiedades de entrada y definir explícitamente cada parámetro

}
```

Método validateParameters

El método validateParameters se utiliza para efectuar la comprobación inicial de parámetros y las resoluciones de parámetro necesarias. Por ejemplo, podría utilizarse para resolver `orderId=*`. Se llama a este método antes que a los métodos getResources y performExecute. Consulte "Interacciones del control de acceso" en la página 86 para obtener más detalles sobre esta secuencia.

Método getResources

Este método se utiliza para implementar el control de acceso a nivel de recurso. Devuelve un vector de parejas recurso-acción sobre el que intenta actuar el mandato. Si no se devuelve nada, el control de acceso a nivel de recurso no se efectúa. Para obtener más información sobre el control de acceso, consulte el Capítulo 4, "Control de acceso" en la página 75.

Método performExecute

El método performExecute contiene la lógica de negocio para el mandato. Debe invocar el método performExecute de la superclase del mandato antes de ejecutar la nueva lógica de negocio. Al final, debe devolver un nombre de vista.

A continuación se muestra un ejemplo de la sintaxis del método performExecute en un nuevo mandato de controlador. En este caso, la respuesta utiliza un mandato redirigir vista, aunque podría utilizar un mandato reenviar vista o un mandato dirigir vista:

```
public void performExecute() throws ECException
{
     super.performExecute();

//////////////////////////////////////////////
//  su lógica de negocio                    //
//////////////////////////////////////////////

// Crear un nuevo TypedProperty para propiedades de respuesta.
TypedProperty rspProp = new TypedProperty();
// establecer propiedades de respuesta
rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
//////////////////////////////////////////////////////
//  La línea siguientes es opcional. La tabla      //
//  VIEWREG puede especificar el URL redirigido.   //
//////////////////////////////////////////////////////

rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

////////////////////////////////////////////////////////////////////////
//   Si utiliza redirigir vista, puede establecer las propiedades de      //
//   respuesta de la manera siguiente:                                    //
//   TypedProperty rspProp = new TypedProperty();                         //
//   rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");                  //
```

```
//   rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");               //
//                                                                       //
//   De nuevo, es opcional establecer explícitamente el nombre de la     //
//   plantilla JSP. La tabla VIEWREG puede especificar la plantilla JSP. //
/////////////////////////////////////////////////////////////////////////

setResponseProperties(rspProp);
}
```

Si especifica el URL de redirección dentro del método performExecute y hay una entrada en la tabla VIEWREG, el valor especificado en el código tiene prioridad sobre el valor de la tabla VIEWREG. El mismo orden de prioridad se aplica para la especificación de una plantilla JSP incluida en el código.

Mandatos de controlador de larga ejecución

Si un mandato de controlador va a tardar mucho en ejecutarse, puede dividirlo en dos mandatos. El primer mandato, que se ejecuta como resultado de una petición de URL, simplemente añade el segundo mandato al Planificador, para que se ejecute como un trabajo en segundo plano. Esto se muestra en el siguiente diagrama:

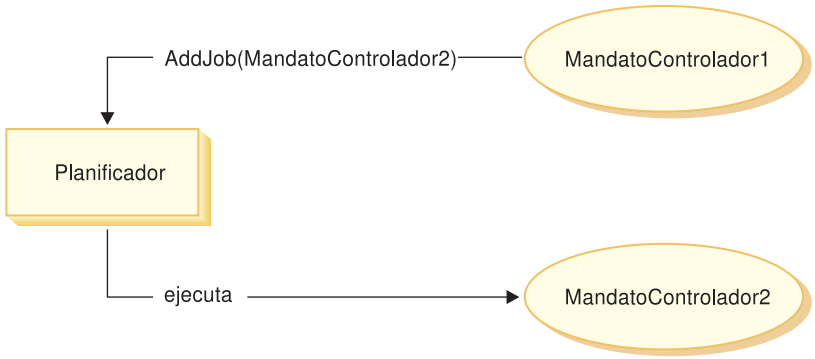

Figura 28.

El flujo que se muestra en el diagrama anterior es el siguiente:

1. El MandatoControlador1 se ejecuta como resultado de una petición de URL.
2. El MandatoControlador1 añade un trabajo al Planificador. El trabajo es MandatoControlador2. El MandatoControlador1 devuelve una vista inmediatamente después de añadir el trabajo al Planificador.
3. El Planificador ejecuta el MandatoControlador2 como un trabajo en segundo plano.

En este escenario, el cliente normalmente sondea el resultado de MandatoControlador2. El MandatoControlador2 debería grabar el estado del trabajo en la base de datos.

Formato de las propiedades de entrada para ver mandatos

Cuando un mandato de controlador finaliza su ejecución, devuelve el nombre de una vista que debe ejecutarse. Esta vista puede requerir que se le pasen diversas propiedades de entrada. Puede haber tres fuentes para estos parámetros, tal como se describe en la lista siguiente:

- Propiedades por omisión almacenadas en la columna PROPERTIES de la tabla CMDREG
- Propiedades por omisión de la columna PROPERTIES de la tabla VIEWREG

- Propiedades de entrada del URL

Para obtener más información sobre cómo se fusionan estas propiedades y se establecen en los atributos para la plantilla JSP, consulte “Establecimiento de atributos JSP - Visión general” en la página 38. Esta sección describe cómo se puede dar formato a las propiedades de entrada de un mandato de vista.

Para los mandatos de redirección de vista, se examinan dos temas:

- Reducción a una serie de consulta para dar soporte al redireccionamiento de URL
- Límites en la longitud del URL de redirección

Para los mandatos reenviar vistas, se examina el tema de enumeración de parámetros de entrada y de establecimiento como atributos en HttpServletRequestObject.

Reducción de parámetros de entrada a una serie de consulta para HttpRedirectView

Todos los parámetros que se pasan a un mandato redirigir vista se reducen a una serie de consulta para redirección de URL. Por ejemplo, suponga que la entrada al mandato redirigir vista contiene las propiedades siguientes:

```
URL = "MyView?p1=v1&p2=v2";
ip1 = "iv1";   // entrada en mandato de controlador original
ip2 = "iv2" ;  // entrada en mandato de controlador original
op1 = "ov1";
op2 = "ov2";
```

Basado en los parámetros de entrada anteriores, el URL final es

```
MyView?p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2
```

Observe que si el mandato va a utilizar SSL, los parámetros se cifran y el URL final aparece como

```
MyView?krypto=valor_cifrado_de"p1=v1&p2=v2&ip1=iv1&ip2=iv2&op1=ov1&op2=ov2"
```

Manejo de un URL de redirección de longitud limitada

Por omisión, todos los parámetros de entrada para el mandato de controlador se propagan al mandato redirigir vista. Si hay un límite en el número de caracteres en el URL de redirección, puede producirse un problema. Un ejemplo de cuándo puede limitarse la longitud es si el cliente utiliza el navegador Internet Explorer. Para este navegador, el URL no puede sobrepasar los 2083 bytes. Si los sobrepasa, se trunca el URL. Por tanto, puede producirse un problema si hay un gran número de parámetros de entrada, o si utiliza el cifrado, porque una serie cifrada suele ser dos o tres veces más larga que sin cifrar.

Hay dos maneras de solucionar el problema de URL de redirección de longitud limitada:

1. Modifique el método getViewInputProperties en el mandato de controlador para que sólo devuelva los conjuntos de parámetros que sea necesario pasar al mandato redirigir vista.
2. Utilice un carácter especial especificado en los parámetros de URL para indicar los parámetros que se pueden eliminar de la serie de parámetros de entrada.

Para demostrar cada una de las acciones anteriores, tomemos, por ejemplo, el siguiente conjunto de parámetros de entrada al mandato de controlador:

```
URL="MyView";
// A continuación se muestran las entradas al mandato de controlador original.
ip1="ipv1";
ip2="ipv2";
ip3="ipv3";
iq1="iqv1";
iq2="iqv2";
ir1="ipr1";
ir2="ipr2";
is="isv";
```

Si modifica el método getViewInputProperties, puede escribir el nuevo método de forma que sólo se pasen los siguientes parámetros al mandato de vista:

```
ir2="ipr2";
is="isv";
```

Utilizando la segunda opción, puede llamarse al mandato vista utilizando parámetros especiales para indicar que deben eliminarse ciertos parámetros de entrada. Por ejemplo, puede obtener el mismo resultado especificando lo siguiente como parámetro de URL:

```
URL="MyView?ip*=&iq*=&ir1="
```

Este parámetro de URL indica lo siguiente a la estructura de ejecución de WebSphere Commerce:

- La especificación `ip*=` indica que deben eliminarse todos los parámetros cuyo nombre empiece por `ip`.
- La especificación `iq*=` indica que deben eliminarse todos los parámetros cuyo nombre empiece por `iq`.
- La especificación `ir1=` indica que debe eliminarse el parámetro `ir1`.

Establecimiento de atributos en el objeto HttpServletRequest para HttpForwardView

El HttpForwardViewCommandImpl por omisión enumera todos los parámetros pasados al mandato y los establece como atributos en el objeto HttpServletRequest.

Por ejemplo, suponga que el objeto requestProperties pasado al mandato reenviar vista contiene los parámetros siguientes:

```
p1="pv1";
p2="pv2";
p3=pv3;   // pv3 es un objeto
```

A continuación, se pasan los atributos siguientes a la plantilla JSP utilizando el método request.setAttribute().

```
request.setAttribute("p1", "pv1");
request.setAttribute("p2", "pv2");
request.setAttribute("p1", pv1);
request.setAttribute("RequestProperties", requestProperties);
request.setAttribute("CommandContext", commandContext);
```

donde `requestProperties` es el objeto TypedProperty que se pasa al mandato, `commandContext` es el objeto de contexto de mandatos que se pasa al mandato y `p1`, `p2` y `p3` son parámetros definidos en el objeto requestProperties.

Compromisos y restituciones de la base de datos para mandatos de controlador

Cuando se ejecuta un mandato de controlador, se suelen actualizar o crear los datos. En muchos casos, es necesario actualizar la base de datos con información nueva al final de la transacción. La transacción la gestiona el controlador Web.

El controlador Web marca el inicio de la transacción antes de llamar al mandato de controlador. Una vez completada la ejecución del mandato de controlador, el mandato de controlador devuelve un nombre de vista al controlador Web. El controlador Web es el responsable de marcar el final de la transacción. El punto real en el que la transacción finaliza (antes o después de invocar la vista) dependerá del tipo de vista utilizado.

Existen tres tipos de mandatos de vista:

- Mandato reenviar vista
- Mandato redirigir vista
- Mandato dirigir vista

El controlador Web determina el mandato de vista que se ha de utilizar para la vista, buscando el nombre de vista en la tabla VIEWREG.

Si la entrada de la tabla VIEWREG especifica que se ha de utilizar ForwardViewCommand, entonces el controlador Web envía los resultados del mandato de controlador a la clase de implementación ForwardViewCommand correspondiente (especificada también en VIEWREG). El mandato de vista se ejecuta en el contexto de la transacción actual. En este caso, el compromiso o la restitución de la base de datos no se lleva a cabo hasta que finaliza el mandato de vista.

Si la entrada de la vista VIEWREG especifica que se ha de utilizar RedirectViewCommand, entonces el controlador Web envía los resultados del mandato de controlador a la clase de implementación RedirectViewCommand correspondiente. El mandato de vista funciona fuera del ámbito de la transacción actual y el compromiso o la restitución de la base de datos se lleva a cabo antes de llamar al mandato de redirección de vista.

Si la entrada de la tabla VIEWREG especifica que se ha de utilizar DirectViewCommand, entonces el controlador Web envía los resultados del mandato de controlador a la clase de implementación DirectViewCommand correspondiente. El mandato de vista se ejecuta en el contexto de la transacción actual. En este caso, el compromiso o la restitución de la base de datos no se lleva a cabo hasta que finaliza el mandato de vista. (Tenga en cuenta que ForwardViewCommand y DirectViewCommand son similares. ForwardViewCommand envía los resultados a una plantilla JSP. Por el contrario, DirectViewCommand recibe los resultados como una corriente de entrada y los pasa como una corriente de salida. Utiliza el método getRawDocument que trata los datos como bytes, o getTextDocument que trata los datos como texto.)

En los casos en los que el mandato de vista se ejecuta bajo el mismo ámbito de transacción que el mandato de controlador, un error en el mandato de vista hace que se produzca una restitución de toda la transacción. Es posible que esta no sea el resultado deseado, dependiendo de la lógica de negocio.

Ejemplo de ámbito de transacción con un mandato de controlador

Para ilustrar las diferencias en el ámbito de transacción de un mandato de controlador, dependiendo del tipo de mandato de vista utilizado, observe los ejemplos siguiente.

Ejemplo 1: Ejecución de la vista en el ámbito de la transacción del mandato de controlador

Suponga que ha creado un nuevo mandato de controlador llamado YourControllerCmdA. El método performExecute deberá incluir entonces lo siguiente:

```
.
.
// Crear un nuevo objeto TypedProperty para la salida.
TypedProperty rspProp = new TypedProperty();

//////////////////////
// Lógica de negocio //
//////////////////////

// Devolver la vista
rspProp.put(ECConstants.EC_VIEWTASKNAME, "YourView");
SetResponseProperties(rspProp);
```

En el extracto de código anterior, el mandato de controlador devuelve “YourView” como la vista. YourView está registrada en la tabla VIEWREG. El siguiente ejemplo muestra una sentencia de inserción para registrar YourView.

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView',  -1, XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView.jsp');
```

donde XX es el identificador de la tienda. Dado que la vista utiliza la clase de implementación com.ibm.commerce.command.HttpForwardViewCommandImpl, el controlador Web utiliza el mandato de vista de envío genérico.

En base al registro del mandato anterior, el controlador Web inicia el archivo YourView.jsp en el ámbito de la transacción del mandato de controlador. Si se produce un error en YourView.jsp, la transacción no se ejecuta correctamente y se produce una restitución de la base de datos. Como resultado, el mandato completo de controlador no se ejecuta correctamente.

Ejemplo 2: Ejecución de la vista fuera del ámbito de transacción del mandato de controlador

Suponga que prefiere tener la información comprometida en la base de datos, incluso llegado el caso en el que se pueda producir un error en la vista. Para que la vista se ejecute fuera del ámbito de la transacción del mandato de controlador, la vista se debe ejecutar como una redirección.

Para ejecutar la vista como una redirección, el método performExecute del mandato de controlador devuelve la vista del modo siguiente:

```
.
.
// Crear un nuevo objeto TypedProperty para la salida.
TypedProperty rspProp = new TypedProperty();

//////////////////////
// Lógica de negocio //
```

```
//////////////////////

// Devolver la vista
rspProp.put(ECConstants.EC_VIEWTASKNAME, EC_GENERIC_REDIRECTVIEW);
rspProp.put(EC_Constants.EC_REDIRECTURL, "YourView2");
```

La sentencia SQL de ejemplo siguiente da soporte a la estrategia de redirección:

```
insert into VIEWREG (ViewName, DeviceFmt_id, storeEnt_id, interfacename,
classname, properties)

values ('YourView2',  -1,  XX,'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl','docname=YourView2.jsp');
```

donde XX es el identificador de la tienda.

Dado que el mandato pasa el valor EC_GENERIC_REDIRECTVIEW como un parámetro de propiedad de respuesta, el controlador Web utiliza el mandato redirigir vista genérico. Este mandato está registrado en la tabla VIEWREG con la siguiente información:

- ViewName = RedirectView
- DeviceFmt_Id = -1
- InterfaceName = com.ibm.commerce.command.RedirectViewCommand
- ClassName = com.ibm.commerce.command.HttpRedirectViewCommandImpl

El controlador Web invoca el mandato de redirección de vista genérico, que toma el URL de redirección como una propiedad de entrada. La respuesta se redirige al URL de redirección. Después de la redirección, se invoca YourView2. Esto se implementa como una vista de envío genérica.

Nuevos mandatos de tarea

Un nuevo mandato de tarea debe ampliarse de la clase abstracta de mandatos de tarea (com.ibm.commerce.command.TaskCommandImpl) e implementar una interfaz que amplíe la interfaz com.ibm.commerce.TaskCommand. Tal como se muestra en el diagrama de la página 106, el nuevo mandato de tarea debe definirse de la siguiente manera:

```
public class MyTaskCmdImpl extends com.ibm.commerce.command.TaskCommandImpl
     implements MyTaskCmd {

}
```

Todas las propiedades de entrada y salida para el mandato de tarea deben definirse en la interfaz de mandatos como, por ejemplo, MyTaskCmd. El emisor de la llamada invoca la interfaz del mandato de tarea en lugar de la clase de implementación de mandato de tarea. Esto le permite tener varias implementaciones del mandato de tarea (una para cada tienda), sin que el emisor de la llamada tenga que preocuparse de a qué clase de implementación debe llamar.

Todos los métodos definidos en la interfaz deben implementarse en la clase de implementación. Puesto que el contexto de mandatos debe establecerlo el emisor de la llamada (un mandato de controlador), el mandato de tarea no necesita establecer el contexto de mandatos. Sin embargo, el mandato de tarea puede obtener información del controlador Web mediante el contexto de mandatos.

Además de implementar los métodos definidos en la interfaz de mandatos de tarea, debe alterar temporalmente el método performExecute de com.ibm.commerce.command.TaskCommandImpl.

El método performExecute contiene la lógica de negocio de la unidad de trabajo concreta que lleva a cabo el mandato de tarea. Debe invocar el método performExecute de la superclase del mandato de tarea antes de ejecutar una lógica de negocio. En la sección de código siguiente se muestra un ejemplo del método performExecute para un mandato de tarea.

```
public void performExecute() throws ECException
{
     super.performExecute();

     // Incluir aquí su lógica de negocio.

     // Establecer propiedades de salida para que el mandato de controlador
     // pueda recuperar el resultado de este mandato de tarea.
}
```

La estructura de ejecución llama al método getResources del mandato de controlador para determinar a qué recursos protegibles accederá el mandato. Puede darse el caso de que un mandato de tarea que se ejecute durante el ámbito de un mandato de controlador intente acceder a recursos que el método getResources del mandato de controlador no haya devuelto. En ese caso, el mandato de tarea mismo puede implementar un método getResources para asegurar que se proporciona control de acceso para recursos protegibles.

Observe que, por omisión, getResources devuelve `null` para un mandato de tarea y no se efectúa la comprobación de control de acceso a nivel de recurso. Por tanto, debe modificar esto si el mandato de tarea accede a recursos protegibles.

Personalización de mandatos existentes

Esta sección describe las distintas formas de personalizar los mandatos de controlador, de tarea y de bean de datos existentes.

Personalización de mandatos de controlador existentes

Un mandato de controlador encapsula la lógica de negocio para un proceso de negocio. La unidades de trabajo individuales del proceso de negocio las pueden realizar los mandatos de tarea. Así pues, hay varias maneras de personalizar un mandato de controlador y algunas de ellas incluyen la personalización de mandatos de tarea.

Cuando se personaliza un mandato de controlador, se puede realizar lo siguiente:

- Añadir proceso y lógica adicionales al mandato de controlador existente. Puede añadirse antes de la lógica de negocio, después de la lógica de negocio y tanto antes como después.
- Sustituir uno o más mandatos de tarea. Esto le permite modificar la forma en que se ejecuta un determinado paso del proceso de negocio.
- Sustituir la vista a la que llama el mandato de controlador.

Las secciones siguientes proporcionan detalles sobre cómo realizar las modificaciones anteriores.

Adición de lógica de negocio nueva al mandato de controlador

Suponga que hay un mandato de controlador WebSphere Commerce, llamado `ExistingControllerCmd`. Según los convenios de nombres de WebSphere

Commerce, este mandato de controlador debe tener una clase de interfaz llamada ExistingControllerCmd y una clase de implementación llamada ExistingControllerCmdImpl. Ahora supongamos que surge un requisito de negocio y que debe añadir nueva lógica de negocio a este mandato ya existente. Una parte de la lógica debe ejecutarse antes de la lógica de mandatos existente y otra parte debe ejecutarse después de la lógica de mandatos existente.

El primer paso para añadir la lógica de negocio nueva es crear una clase de implementación nueva que amplíe la clase de implementación original. En este ejemplo, creará una nueva clase ModifiedControllerCmdImpl que amplía la clase ExistingControllerCmdImpl. La nueva clase de implementación debe implementar la interfaz original (ExistingControllerCmd).

En la nueva clase de implementación debe crear un nuevo método performExecute para modificar el performExecute del mandato existente. En el nuevo método performExecute, hay dos modos de insertar la lógica de negocio nueva: puede incluir directamente el código en el mandato de controlador o puede crear un mandato de tarea nuevo para realizar la lógica de negocio nueva. Si crea un nuevo mandato de tarea, debe crear una instancia del objeto de mandato de tarea nuevo desde el mandato de controlador.

La siguiente sección de código demuestra cómo se añade la nueva lógica al principio y final de un mandato de controlador existente incluyendo la lógica directamente en el mandato de controlador:

```
public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {

            /* Inserte nueva lógica de negocio que debe
               ejecutarse antes del mandato original.
            */

            // Ejecutar la lógica de mandato original.
            super.performExecute();

            /* Inserte nueva lógica de negocio que debe
               ejecutarse después del mandato original.
            */
          }
    }
```

La siguiente sección de código demuestra cómo se añade la nueva lógica al principio de un mandato de controlador existente creando una instancia de un mandato de tarea nuevo desde el mandato de controlador. Además, también se ha de crear la interfaz del mandato de tarea nueva y la clase de implementación y se ha de registrar el mandato de tarea en el registro de mandatos.

```
// Importar el paquete con CommandFactory
import com.ibm.commerce.command.*;

public class ModifiedControllerCmdImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {

      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {
            MyNewTaskCmd cmd = null;
```

```
            cmd = (MyNewTaskCmd) CommandFactory.createCommand(
               "com.mycompany.mycommands.MyNewTaskCommand",
               getStoreId());

            /*
            Establezca los parámetros de entrada del mandato de tarea,
            llame a su método de ejecución y recupere los parámetros de
            salida, según sea necesario.
            */

            super.performExecute();
         }
   }
```

Independientemente de si incluye la nueva lógica de negocio en el mandato de controlador o de si crea un mandato de tarea para ejecutar la lógica, debe también actualizar la tabla CMDREG en el registro de mandatos WebSphere Commerce para asociar la nueva clase de implementación del mandato de controlador con la interfaz del mandato de controlador existente. La siguiente sentencia SQL muestra una actualización de ejemplo:

```
update CMDREG
set CLASSNAME='ModifiedControllerCmdImpl'
where INTERFACENAME='ExistingControllerCmd'
```

Sustitución de los mandatos de tarea llamados por un mandato de controlador

Un mandato de controlador suele llamar a varios mandatos de tarea que realizan tareas individuales. Colectivamente, estas tareas componen el proceso de negocio que representa el mandato de controlador. Es posible que necesite cambiar el modo en que se realiza un paso determinado del proceso, en lugar de añadir lógica de negocio nueva al principio o al final del mandato de controlador. En este caso, deberá sustituir la instancia del mandato de tarea, que desea alterar temporalmente, por la instancia de un mandato de tarea nuevo que ejecute la tarea del modo que desea.

Debido al diseño del modelo de programación de WebSphere Commerce, no es necesario que cree una nueva clase de implementación del mandato de controlador que sustituya al mandato de tarea. El mandato de controlador crea una instancia del mandato de tarea llamando al método `createCommand` de la fábrica de mandatos. La fábrica de mandatos utiliza el nombre de la interfaz del mandato de tarea y luego determina la clase de implementación correcta, basándose en el registro de mandatos. De este modo, para sustituir el mandato de tarea del que se crea una instancia, deberá crear una nueva clase de implementación de mandato de tarea y luego actualizar el registro de mandatos para que el nombre de la interfaz de mandatos de tarea original se asocie a la nueva clase de implementación del mandato de tarea. Consulte “Personalización de los mandatos de tarea existentes” en la página 122 para obtener más información.

Sustitución de la vista llamada mediante un mandato de controlador

Para sustituir la vista que llama mediante un mandato de controlador, puede crear una nueva clase de implementación para el mandato de controlador. Por ejemplo, cree un nuevo `ModifiedControllerCmdImpl` que amplíe `ExistingControllerCmdImpl` e implemente la interfaz `ExistingControllerCmd`.

Dentro de la clase `ModifiedControllerCmdImpl`, modifique el método `performExecute`. En el nuevo método `performExecute`, llame a `super.performExecute` para asegurarse de que tiene lugar todo el proceso de mandatos. Después de ejecutar toda la lógica de mandatos, puede utilizar las

propiedades de respuesta para modificar la vista llamada. La siguiente sección de código muestra cómo modificar la vista mientras se ejecuta como una redirección:

```
// Importar los paquetes que contienen TypedProperty y ECConstants.
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {

            // Ejecutar la lógica de mandato original.
            super.performExecute();

            // Crear un nuevo TypedProperty para propiedades de respuesta.
            TypedProperty rspProp = new TypedProperty();

           // establecer propiedades de respuesta
           rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");
          ////////////////////////////////////////////////////
          //  La línea siguiente es opcional. La tabla      //
          //  VIEWREG puede especificar el URL redirigido.  //
          ////////////////////////////////////////////////////

          rspProp.put(ECConstants.EC_REDIRECTURL, MyURL);

          setResponseProperties(rspProp);

         }
   }
```

La siguiente sección de código muestra cómo modificar la vista cuando ésta se ejecuta como vista de redirección:

```
// Importar los paquetes que contienen TypedProperty y ECConstants.
import com.ibm.commerce.datatype.*;
import com.ibm.commerce.server.*;

public class ModifiedControllerCmdImplImpl extends ExistingControllerCmdImpl
   implements ExistingControllerCmd
   {
      public void performExecute ()
         throws com.ibm.commerce.exception.ECException
         {

            // Ejecutar la lógica de mandato original.
            super.performExecute();

            // Crear un nuevo TypedProperty para propiedades de respuesta.
            TypedProperty rspProp = new TypedProperty();

           // establecer propiedades de respuesta
           rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyView");

          ////////////////////////////////////////////////////
          // Es opcional establecer explícitamente el nombre //
          //  de la plantilla JSP. La tabla VIEWREG puede    //
          //  especificar la plantilla JSP.                  //
          ////////////////////////////////////////////////////

          rspProp.put(ECConstants.EC_DOCPATHNAME, "MyJSP.jsp");
```

```
            setResponseProperties(rspProp);

        }
    }
```

Para determinar la vista que utiliza un mandato de controlador existente, consulte la sección Referencias de la ayuda en línea de WebSphere Commerce.

Personalización de los mandatos de tarea existentes

Hay dos modos estándar de modificar los mandatos de tarea de WebSphere Commerce existentes. Con estos métodos de modificación, puede realizar lo siguiente:

- Añadir proceso y lógica adicionales al mandato de tarea existente. Puede añadirse antes de la lógica de negocio, después de la lógica de negocio y tanto antes como después.
- Sustituir por completo la lógica de negocio existente por su propia lógica de negocio.

Para realizar las modificaciones anteriores, deberá crear una nueva clase de implementación de mandato de tarea. En las secciones siguientes se proporcionan más detalles.

Adición de lógica de negocio nueva a un mandato de tarea

Suponga que hay un mandato de tarea WebSphere Commerce, llamado `ExistingTaskCmd`. Según los convenios de nombres de WebSphere Commerce, este mandato de tarea debe tener una clase de interfaz llamada `ExistingTaskCmd` y una clase de implementación llamada `ExistingTaskCmdImpl`. Ahora supongamos que surge un requisito de negocio y que debe añadir nueva lógica de negocio a este mandato ya existente. Una parte de la lógica debe ejecutarse antes de la lógica de mandatos existente y otra parte debe ejecutarse después de la lógica de mandatos existente.

El primer paso para añadir la lógica de negocio nueva es crear una clase de implementación nueva que amplíe la clase de implementación original. En este ejemplo, creará una nueva clase `ModifiedTaskCmdImpl` que amplía la clase `ExistingTaskCmdImpl`. La nueva clase de implementación debe implementar la interfaz original (`ExistingTaskCmd`).

En el nuevo mandato, debe modificar el método `performExecute` existente e incluir la nueva lógica antes y después de llamar al método `super.performExecute`.

El siguiente pseudo-código muestra cómo se añade la nueva lógica de negocio a un mandato de tarea existente:

```
public class ModifiedTaskCmdImpl extends ExistingTaskCmdImpl
     implements ExistingTaskCmd {


         /* Inserte nueva lógica de negocio que debe
            ejecutarse antes del mandato original.
         */

         // Ejecutar la lógica de mandato original.
         super.performExecute();

         /* Inserte nueva lógica de negocio que debe
            ejecutarse después del mandato original.
         */
}
```

También debe actualizar la tabla CMDREG para asociar la nueva clase de implementación con la interfaz existente. La siguiente sentencia SQL muestra una actualización de ejemplo:

```
update CMDREG
set CLASSNAME='ModifiedTaskCmdImpl'
where INTERFACENAME='ExistingTaskCmd'
```

Sustitución de la lógica de negocio de un mandato de tarea existente

Para sustituir la lógica de negocio de un mandato de tarea existente, debe crear una nueva clase de implementación para el mandato de tarea. Esta nueva clase de implementación debe ampliarse a partir del mandato de tarea existente pero no debe implementar la interfaz existente. Además, en la nueva clase de implementación, no llame al método `performExecute` de la superclase.

La ampliación a partir del mandato exacto que va a sustituir puede parecer poco lógica, pero el motivo de hacerlo así está relacionado con el soporte de versiones futuras de WebSphere Commerce. Este enfoque protege a su código de los posibles cambios que se efectúen en interfaces de mandatos en versiones futuras de WebSphere Commerce.

Por ejemplo, suponga que desea sustituir la lógica de negocio del mandato de tarea `OrderNotifyCmdImpl`. En este caso, deberá crear un nuevo mandato de tarea denominado `CustomizedOrderNotifyCmdImpl`. Este mandato amplía `OrderNotifyCmdImpl`. En el nuevo mandato `CustomizedOrderNotifyCmdImpl`, debe crear la nueva lógica de negocio, pero no llame al método `performExecute` desde la superclase. Si una versión futura de WebSphere Commerce introduce un nuevo método, denominado `newMethod` en la interfaz, la versión correspondiente del mandato `OrderNotifyCmdImpl` incluirá una implementación por omisión del método `newMethod`. Luego, puesto que el nuevo mandato se amplía a partir de `OrderNotifyCmdImpl`, el compilador encontrará la implementación por omisión de este nuevo método en el mandato `OrderNotifyCmdImpl` y su nuevo mandato estará protegido frente a los cambios de interfaz.

Consulte la sección Referencias de la ayuda en línea de WebSphere Commerce para asegurarse de que la nueva clase de implementación proporciona el mismo comportamiento externo que la clase existente.

Personalización de los beans de datos

Un bean de datos generalmente amplía un bean de acceso. El bean de acceso, que puede ser generado por VisualAge para Java, proporciona una forma sencilla de acceder a la información de un bean de entidad. Cuando se realizan modificaciones en un bean de entidad (por ejemplo, cuando se añade un nuevo campo, un nuevo método de negocio o un nuevo buscador), la actualización se refleja en el bean de acceso tan pronto se regenera el bean de acceso. Puesto que el bean de datos amplía el bean de acceso, hereda automáticamente los nuevos atributos. Como resultado de esta relación, no es necesario efectuar ninguna codificación para habilitar el bean de datos de forma que utilice los nuevos atributos del bean de entidad.

Si necesita añadir nuevos atributos a un bean de datos que no se derivan de un bean de entidad, puede ampliar el bean de datos existente utilizando la característica de herencia de Java. Por ejemplo, si desea añadir un nuevo campo a OrderDataBean, defina MyOrderDataBean de la siguiente forma:

```
public class MyOrderDataBean extends OrderDataBean
{
     public String myNewField () {
     //  implemente aquí el nuevo campo
     }

}
```

El nuevo bean de datos también debe tener una clase BeanInfo. A continuación se muestra un ejemplo de la declaración para esta clase:

```
public class MyOrderDataBeanInfo extends java.beans.SimpleBeanInfo
{

}
```

VisualAge para Java proporciona una herramienta que le permite generar esta clase BeanInfo.

Capítulo 7. Acuerdos comerciales y políticas de negocio (Business Edition)

Este capítulo sólo se aplica a WebSphere Commerce Business Edition.

Introducción

Uno de los elementos clave del comercio B2B (de empresa a empresa) es la gestión de relaciones. Un acuerdo comercial se utiliza para gestionar las relaciones comerciales entre el comprador y la organización vendedora. El modelo de acuerdo comercial que utiliza WebSphere Commerce Business Edition soporta varios tipos de acuerdos comerciales, como por ejemplo, el Contrato y la RFQ (Solicitud de presupuesto).

El elemento principal de un acuerdo comercial es un conjunto de términos y condiciones. Cada término y condición define una norma de negocio específica que se utilizará en el comercio. Con WebSphere Commerce Business Edition, se puede negociar un conjunto de términos y condiciones utilizando el proceso en línea de una RFQ, o negociarlo fuera de línea y después capturarlo utilizando las interfaces de gestión de relaciones comerciales en WebSphere Commerce Accelerator.

Hay varias formas de término y condición:

- Un término y condición que selecciona una de las políticas de negocio predefinidas, como por ejemplo, una lista de precios y una política de devoluciones. O puede seleccionar una política de negocio que usted haya creado. Un objeto de término y condición también puede referirse a varios objetos de política de negocio.
- Un término y condición que aplica un ajuste específico a la política de negocio, como por ejemplo, un ajuste a la fijación de precios estándar.
- Un término y condición que define un conjunto de parámetros que controlan un proceso de negocio. Por ejemplo, podría especificar que un contrato concreto tiene que utilizar un centro de despacho de pedidos específico.

Un contrato está formado por un conjunto de términos y condiciones. Esto se muestra en el siguiente diagrama:

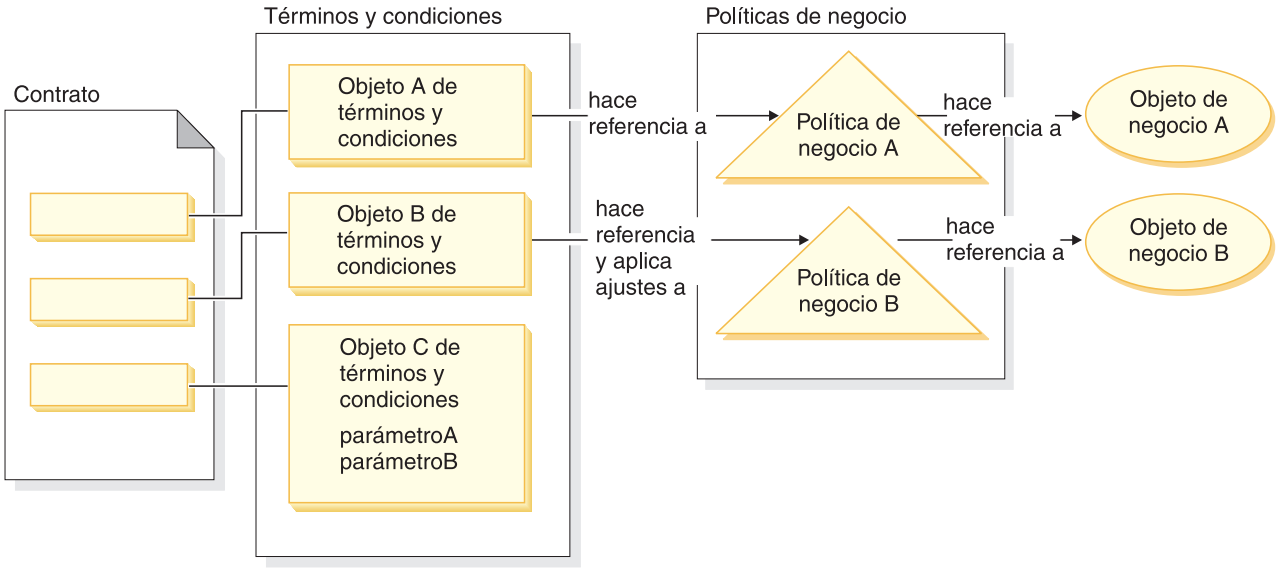

Figura 29.

Observe los siguientes puntos en el diagrama anterior:

- El término "ajuste" hace referencia a una modificación de la política de negocio. Por ejemplo, puede utilizarse para aplicar un descuento al resultado de una política de negocio, de forma que se aplique un 10% de descuento al precio estándar. También puede utilizarse para influir en la política de negocio con un conjunto de parámetros.
- Por ejemplo, el objeto de término y condición A puede representar un objeto de término y condición de envío. En este caso, la política de negocio A puede representar una política de negocio de modalidad de envío y el objeto de negocio A representa la modalidad de envío "A3" de la empresa de transportes XYZ.
- En otro ejemplo, el objeto de término y condición B puede representar un objeto de término y condición de precio que aplica un 50% de descuento en el precio definido por la política de negocio B. En este caso, la política de negocio B es una política de precios y el objeto de negocio B es un contenedor de propuestas de comercio que define la propuesta de comercio para la categoría maestra.

Este capítulo proporciona directrices para los programadores sobre cómo crear nuevas políticas de negocio, y nuevos términos y condiciones.

La tienda de ejemplo ToolTech muestra un objeto de término y condición de envío, y un objeto de término y condición de precio en su flujo de negocio. Para obtener más información sobre los datos de contrato que dan soporte a estos ejemplos, consulte "Datos del contrato de la tienda de ejemplo ToolTech" en la página 127.

Mandatos y objetos de política de negocio

Un objeto de política de negocio contiene la información siguiente:

- ID de la política
  Es la clave primaria del objeto de política de negocio.
- Tipo de política
  Define el tipo de política de negocio. `Price` y `ProductSet` son ejemplos de tipos de política.

- Nombre de la política
  Cada política de negocio debe tener un nombre exclusivo.
- Entidad de tienda
  La tienda o grupo de tiendas en el que se despliega la política de negocio.
- Propiedades
  Un conjunto de propiedades por omisión que se pueden pasar al mandato de política de negocio. Los mandatos asociados al objeto de política de negocio se almacenan en la tabla BusinessPolicyCmd.
- Periodo en vigor
  El periodo en el que el objeto de política de negocio está en vigor.
- Mandato de política de negocio
  Ninguno o algunos mandatos de política de negocio que implementa la política de negocio. Normalmente, un mandato de política de negocio es invocado por un proceso de negocio que puede ser un mandato de tarea o un mandato de controlador. Por ejemplo, el mandato `getContractPrice()` obtiene el término y condición de precios. Este término y condición de precios hace referencia a un mandato de política de precios específico que se utiliza para calcular el precio.

Pueden asociarse varios mandatos de política de negocio a un solo objeto de política de negocio. Cada mandato de política de negocio debe implementar la misma interfaz definida por el objeto tipo de política de negocio. La estructura de un nuevo mandato de política de negocio se muestra en la siguiente ilustración:

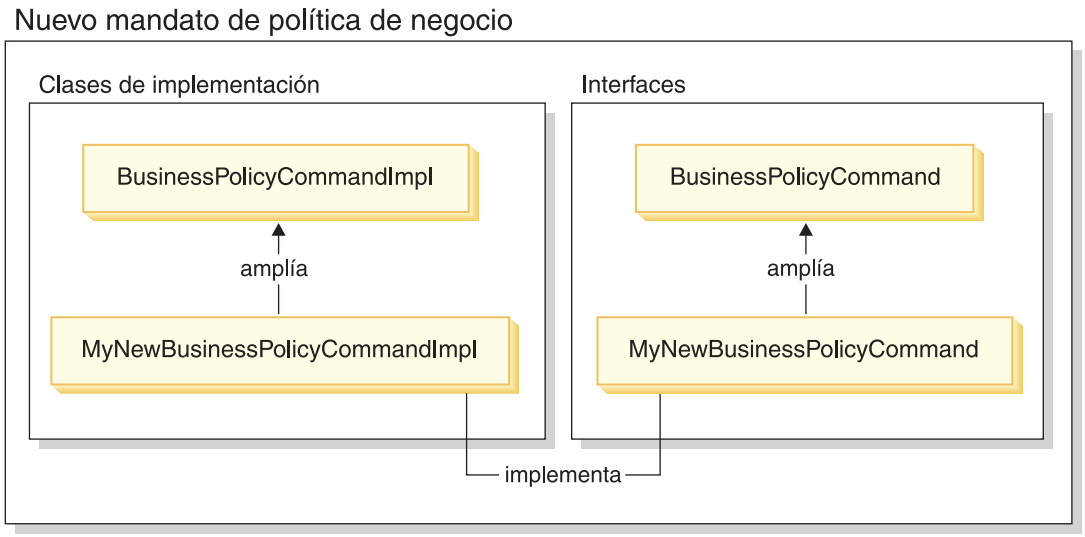

Figura 30.

Tal como se muestra en la ilustración, para crear un nuevo mandato de política de negocio, se crea una nueva clase de implementación que amplíe la clase de implementación BusinessPolicyCmdImpl de WebSphere Commerce. También se crea una nueva interfaz que amplíe la interfaz BusinessPolicyCmd.

Datos del contrato de la tienda de ejemplo ToolTech

Esta sección proporciona una introducción para algunos de los datos de contrato que se utilizan en la tienda de ejemplo ToolTech.

En las secciones siguientes, los datos de ejemplo están organizados por tabla de base de datos. Sólo se muestran las filas y columnas relevantes. Además, tenga en

cuenta que cuando se instale el ejemplo, los identificadores exclusivos (como CONTRACT_ID) pueden tener valores diferentes de los que se muestran aquí.

Datos de ejemplo de la tabla CONTRACT

La tabla siguiente muestra los datos de ejemplo relevantes de la tabla de base de datos CONTRACT de ToolTech. Observe que las cabeceras de las columnas de la base de datos se muestran en la primera columna y la fila de datos de ejemplo de la tabla se muestra en la segunda columna.

Nombre de columna	Datos de ejemplo
CONTRACT_ID	10007
MAJORVERSION	1
MINORVERSION	0
NAME	ToolTechContractNumber 4567
MEMBER_ID	-2001
ORIGIN	0
STATE	3
USAGE	1
MARKFORDELETE	0

Datos de ejemplo de la tabla TERMCOND

La tabla siguiente muestra los datos de ejemplo relevantes de la tabla de base de datos TERMCOND de ToolTech. Observe que las cabeceras de las columnas de la base de datos se muestran en la primera columna y las filas de datos de ejemplo de la tabla se muestran en las columnas segunda y tercera.

Nombre de columna	Fila 1 de datos de ejemplo	Fila 2 de datos de ejemplo
TERMCOND_ID	10025	10030
TCSUBTYPE_ID	PriceTCPriceListWith SelectiveAdjustment	ShippingTCShippingMode
TRADING_ID	10007	10007
STRINGFILED1	ProductSet2	
INTEGERFIELD2	10002	
INTEGERFIELD3	1	
BIGINTFIELD1	10051	
FLOATFIELD1	-50.0	
SEQUENCE	1	6

Datos de ejemplo de la tabla POLICYTC

La tabla siguiente muestra los datos de ejemplo relevantes de la tabla de base de datos POLICYTC de ToolTech. Esta tabla establece la relación entre una política y un objeto de términos y condiciones.

	Nombre de columna	
	POLICY_ID	**TERMCOND_ID**
Fila 1 de datos de ejemplo	10053	10025

	Nombre de columna	
	POLICY_ID	**TERMCOND_ID**
Fila 2 de datos de ejemplo	10056	10030

Datos de ejemplo de la tabla POLICY

La tabla siguiente muestra los datos de ejemplo relevantes de la tabla de base de datos POLICY de ToolTech.

Nombre de columna	Fila 1 de datos de ejemplo	Fila 2 de datos de ejemplo
POLICY_ID	10053	10056
POLICYNAME	MasterCatalogPriceList	A3
POLICYTYPE_ID	Price	ShippingMode
STOREENT_ID	10051	10051
PROPERTIES	name=ToolTech& member_id=-2001	shippingMode=A3
STARTTIME	null	null
ENDTIME	null	null

Datos de ejemplo de la tabla TRADEPOSCN

La tabla siguiente muestra los datos de ejemplo relevantes de la tabla de base de datos TRADEPOSCN de ToolTech.

	Nombre de columna			
	READEPOSCN_ID	**MEMBER_ID**	**NAME**	**TYPE**
Fila de datos de ejemplo	10051	-2001	ToolTech	S

Datos de ejemplo de la tabla SHIPMODE

La tabla siguiente muestra los datos de ejemplo relevantes de la tabla de base de datos SHIPMODE de ToolTech.

	Nombre de columna			
	SHIPMODE_ID	**STOREENTITY_ID**	**CODE**	**CARRIER**
Fila de datos de ejemplo	10053	10051	A3	Transportes XYZ

Ampliación del modelo de contrato existente

Un contrato puede estar formado por uno o más objetos de términos y condiciones, cada uno de los cuales hace referencia a una política. Por ello, las secciones siguientes describen los pasos necesarios para crear una nueva política de negocio e integrarla en el flujo de negocio.

Para tener una visión general del proceso, a continuación se muestran los pasos generales para llevar a cabo esta tarea:

1. Crear una nueva política de negocio.
   Las tareas siguientes están relacionadas con la creación de un nuevo mandato de política de negocio:
   a. Crear un nuevo tipo de política de negocio (si fuera necesario).
      Se proporcionan varios tipos de política de negocio, pero si los tipos estándar no satisfacen sus requisitos, cree un nuevo tipo de política de negocio.
   b. Crear un nuevo mandato de política de negocio.
   c. Registrar la nueva política de negocio y el nuevo mandato de política de negocio.
2. Relacionar un objeto de términos y condiciones con la nueva política de negocio
   Esto puede llevarse a cabo relacionando un objeto de términos y condiciones existente con la nueva política de negocio, o creando un nuevo objeto de términos y condiciones. Si crea un nuevo objeto de términos y condiciones, debe efectuar los pasos siguientes:
   a. Registrar el nuevo término y condición en la base de datos
   b. Registrar el nuevo término y condición en la DTD (definición de tipo de documento) de contratos
   c. Crear un nuevo bean enterprise CMP para el término y condición
   d. Actualizar WebSphere Commerce Accelerator para que refleje el nuevo término y condición
3. Llamar a la nueva política de negocio durante el flujo de negocio.

Creación de una nueva política de negocio

Generalmente, para crear una nueva política de negocio, es necesario registrar una política de negocio exclusiva en la base de datos y crear un nuevo mandato de política de negocio.

Para ello, debe seguir los siguientes pasos generales:

1. Crear un nuevo tipo de política de negocio (si fuera necesario).
2. Escribir el nuevo mandato de política de negocio.
3. Registrar la nueva política de negocio y el nuevo mandato de política de negocio en la base de datos.

Cada uno de los pasos anteriores se describen con más detalle en la secciones siguientes.

Creación de un nuevo tipo de política de negocio

Esta sección describe cómo crear un nuevo tipo de política de negocio. Un tipo de política de negocio indica el dominio de la transacción a la que se aplica la política. Algunos ejemplos de tipos de política de negocio son:

- Price
- ProductSet
- ShippingMode
- ShippingCharge
- Payment
- ReturnCharge
- ReturnApproval
- ReturnPayment

- InvoiceFormat

Si los tipos de política de negocio existentes no satisfacen los requisitos de su negocio, debe crear un nuevo tipo de política de negocio. La creación de un nuevo tipo consiste en definir y registrar el tipo de política de negocio.

Al definir y registrar un nuevo tipo de política, debe actualizar las tablas de base de datos siguientes:

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

La tabla POLICYTYPE especifica el tipo de política de negocio que está creando. Contiene una sola columna, POLICYTYPE_ID, que es la clave primaria. Un valor de ejemplo es `Price`. Si crea un nuevo tipo de política de negocio, asegúrese de especificar un POLICYTYPE_ID exclusivo.

La tabla PLCYTYCMIF es la tabla de especificación de relaciones entre el tipo de política de negocio y la interfaz de mandatos. Es decir, para cada tipo de política de negocio, especifica la interfaz de mandatos Java para el objeto de política de negocio. Aunque puede no haber ningún mandato de política de negocio que implemente una política de negocio, o puede haber algunos, todos los mandatos de política de negocio deben implementar la interfaz especificada aquí.

La tabla PLCYTYPDSC especifica una descripción del tipo de política de negocio. Incluye un identificador de idioma de la descripción y la descripción del tipo de política de negocio.

Para crear un nuevo tipo de política de negocio, cree una entrada en cada una de estas tablas para el nuevo tipo de política de negocio. Las siguientes sentencias SQL proporcionan un ejemplo:

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('MyNewPolicyType');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('MyNewPolicyType',
   'com.mycompany.mybusinesspolicycommands.MyNewPolicy');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('MyNewPolicyType', -5,
   'Mi nuevo tipo de política para el ejemplo.');
```

Como paso final de la creación del nuevo tipo de política de negocio, puede codificar una o varias interfaces nuevas de tipo de política de negocio. Estas interfaces las implementará luego cualquier mandato de política de negocio que esté dentro del dominio de este tipo de política de negocio. Por ejemplo, en la tienda de ejemplo ToolTech, Price se define como un tipo de política de negocio. Por ello, las interfaces `com.ibm.commerce.price.commands.ResolvePriceListsCmd` y `com.ibm.commerce.price.commands.RetrievePricesCmd` las implementan todos los mandatos de política de negocio relacionados con el precio.

Si no va a tener un mandato de política de negocio que realice operaciones en el nuevo tipo de política de negocio, no es necesario que cree una nueva interfaz. Esto es algo inusual, y en la mayoría de los casos al crear un nuevo tipo de política de negocio, debe crear también una nueva interfaz de tipo de política de negocio.

Cuando cree una interfaz de tipo de política de negocio, la nueva interfaz debe ampliar la interfaz `com.ibm.commerce.command.BusinessPolicyCommand`.

Creación de un nuevo mandato de política de negocio

Para crear un nuevo mandato de política de negocio, debe crear un nuevo mandato que implemente la interfaz del tipo de política de negocio con el que está relacionado el mandato. El nuevo mandato también debe ampliar la clase de implementación `com.ibm.commerce.command.BusinessPolicyCommandImpl`. Esto es muy parecido a crear un nuevo mandato de controlador o de tarea.

Hay dos métodos diferentes mediante los que puede pasar propiedades de entrada a un mandato de política de negocio. El primero es tener propiedades de entrada por omisión especificadas en la columna PROPERTIES de la tabla POLICY. Para obtener más información sobre esta tabla, consulte la sección siguiente.

El segundo método es crear un nuevo campo en el mandato para cada una de las propiedades de entrada. Para cada campo, cree una nueva pareja de métodos get y set.

Establecimiento de requestProperties en mandatos de política de negocio

Hay dos maneras de establecer requestProperties en un objeto de mandato de política de negocio. La primera utiliza la columna PROPERTIES de la tabla POLICY para establecer las propiedades por omisión. Esto se logra con el método setRequestProperties. La segunda manera de establecer propiedades es hacer que el mandato (de controlador o de tarea) que llama al mandato de política de negocio establezca explícitamente otras propiedades requeridas.

Al crear un nuevo mandato de política de negocio, debe modificar el método setRequestProperties por omisión para que incluya la lógica para establecer explícitamente cada uno de los parámetros que se incluyen en el objeto requestProperties.

Tomemos como ejemplo un nuevo mandato de política de negocio que tiene el nombre de interfaz MyNewBusinessPolicyCmd y el nombre de clase de implementación MyNewBusinessPolicyCmdImpl.

Supongamos que la entrada en la tabla POLICY para este nuevo mandato de política de negocio incluye los valores siguientes en la columna PROPERTIES:

- defaultProperty1=manzana
- defaultProperty2=naranja
- defaultProperty3=plátano

La interfaz para este nuevo mandato de política de negocio se define de la forma siguiente:

```
public interface MyNewBusinessPolicyCmd extends
   com.ibm.commerce.command.BusinessPolicyCmd {
      java.lang.String defaultCommandClassName =
         'com.mycompany.mycommands.MyNewBusinessPolicyCmdImpl';
      public void setProperty1();
      public void setProperty2();
}
```

La clase de implementación para este nuevo mandato de política de negocio se define de la forma siguiente:

```
public class MyNewBusinessPolicyCmdImpl extends
   com.ibm.commerce.command.BusinessPolicyCmdImpl
   implements com.mycompany.mycommands.MyNewBusinessPolicyCmd {
      // Establecer propiedades por omisión que se almacenan en
```

```
      // la tabla POLICY

      private java.lang.String defaultProperty1;
      private java.lang.String defaultProperty2;
      private java.lang.String defaultProperty3;

      // Empezar a establecer propiedades que debe establecer
     // el mandato que llama.

      //  *** propiedad1   ***
      private java.lang.String property1;
      public java.lang.String getProperty1() {
         return property1;
         }
      public void setProperty1(java.lang.String newProperty1) {
         property1 = newProperty1;
      }

      //  *** propiedad2   ***
      private java.lang.String property2;
      public java.lang.String getProperty2() {
         return property2;
         }
      public void setProperty1(java.lang.String newProperty2) {
         property2 = newProperty2;
         }

      // Fin de establecer propiedades que debe establecer
      // el mandato que llama.

      /* Después de crear una instancia, el mandato de política de
         negocio establece todas las propiedades por omisión de
         la tabla POLICY en el objeto          requestProperties. El mandato
         que llama es          responsable de establecer cualquier otra
         propiedad requerida.
      */

      public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
         requestProperties) {
         // Obtener las propiedades por omisión definidas en la tabla POLICY
         setDefaultProperty1(requestProperties.get("defaultProperty1"));
         setDefaultProperty2(requestProperties.get("defaultProperty2"));
         setDefaultProperty3(requestProperties.get("defaultProperty3"));
         }

}
```

El mandato que llama al nuevo mandato de política de negocio podría definirse de manera similar a la siguiente:

```
public class MyCallerCommandImpl
   extends com.ibm.commerce.command.TaskCommandImpl
   implements com.mycompany.mycommands.MyCallerCommand {

   /* Incluir todos los elementos y procesos requeridos para el
      mandato de tarea.
   */

   // Determinar el ID de política y establecer PolicyId

   // Llamar al mandato de política de negocio.

   cmd = (MyNewBusinessPolicyCmd) CommandFactory
            createPolicyCommand(policyId);

   // Establecer la propiedades requeridas
```

```
    cmd.setProperty1("Ensalada de frutas");
    cmd.setProperty2("Comida preferida");

  cmd.execute();
}
```

Registro de la nueva política de negocio y el nuevo mandato de política de negocio

Después de crear el nuevo mandato de política de negocio, debe registrar la política de negocio y el mandato de política de negocio en la base de datos.

Las políticas de negocio se registran en la tabla POLICY. Esta tabla contiene las siguientes columnas:

- POLICY_ID
  La clave primaria. Es el identificador de la política.
- POLICYNAME
  Un nombre de política exclusivo.
- POLICYTYPE_ID
  El identificador del tipo de política. Es la clave externa de la tabla POLICYTYPE.
- STOREENT_ID
  La tienda o grupo de tiendas a la que se aplica la política.
- PROPERTIES
  Propiedades por omisión que se pueden establecer en el mandato de política de negocio. Se especifican como parejas nombre-valor, por ejemplo, `parm1=val1&parm2=val2`.
- STARTDATE
  La fecha de inicio (especificada como indicación de la fecha) de la política. Si es NULL, la fecha de inicio es inmediata.
- ENDDATE
  La fecha de finalización (especificada como indicación de la fecha) de la política. Si es NULL, no hay fecha de finalización.

Cuando la nueva política ya está registrada en la tabla POLICY, debe registrar una relación entre la política y el mandato de política de negocio que implementa la política de negocio. La tabla POLICYCMD se utiliza para este propósito. Esta tabla contiene las siguientes columnas:

- POLICY_ID
  Referencia de clave externa a la tabla POLICY.
- BUSINESSCMDCLASS
  El mandato de política de negocio que implementa la política.
- PROPERTIES
  Propiedades por omisión que se pueden establecer en el mandato de política de negocio. Se especifican como parejas nombre-valor, por ejemplo, `parm1=val1&parm2=val2`.

Cómo relacionar un objeto de términos y condiciones con una nueva política de negocio

En la estructura de contratos y políticas de WebSphere Commerce, los términos y condiciones (también denominados *términos*) proporcionan una forma de describir un acuerdo entre un comprador y un vendedor. Los términos y condiciones pueden utilizarse en varios tipos de acuerdos comerciales, como por ejemplo, un contrato y una RFQ (solicitud de presupuesto). Los objetos de términos y

condiciones normalmente hacen referencia a políticas de negocio con un ajuste opcional. Por ejemplo, un objeto de términos y condiciones de precio se crea eligiendo uno de los objetos de política de precios. En el término del precio, un gestor de cuentas puede efectuar ajustes al precio estándar de la tienda, como por ejemplo:

- Un porcentaje de descuento sobre la lista de precios estándar
- Un porcentaje de descuento en un conjunto específico de productos

Cada uno de los ajustes se especifica como un término y condición.

Cuando crea una nueva política de negocio, debe haber al menos un objeto de términos y condiciones que haga referencia a esta política de negocio, si esa política se va a utilizar en un contrato. Puede relacionar un objeto de términos y condiciones ya existente con la nueva política de negocio (para ello, se captura la relación entre el objeto de términos y condiciones existente y la nueva política de negocio en el archivo `B2BTrading.dtd`), o puede crear un nuevo objeto de términos y condiciones que esté relacionado con la nueva política de negocio.

Creación de nuevos términos y condiciones

Dentro de la arquitectura de WebSphere Commerce, se crean nuevos objetos de términos y condiciones efectuando estos pasos:

1. Actualizar el esquema de base de datos para que incluya el nuevo término y condición.
2. Actualizar el archivo `B2BTrading.dtd` para que refleje el nuevo término y condición.
3. Crear un nuevo bean enterprise para el término y condición.
4. Actualizar WebSphere Commerce Accelerator para que refleje el nuevo término y condición, o utilizar el mandato cargar contrato para crear un nuevo contrato que utilice el nuevo término y condición.

En las secciones siguientes, el ejemplo de `MyTC` es el nuevo objeto de término y condición.

Registro del nuevo término y condición en la base de datos

Cuando crea un nuevo objeto de término y condición, debe actualizar el esquema de base de datos para que incluya este objeto. Las tablas de base de datos que deben actualizarse son TCTYPE y TCSUBTYPE.

La siguiente sentencia SQL muestra un ejemplo de cómo actualizar el esquema:

```
insert into TCTYPE (TCTYPE_ID) values ('MyTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME,
   DEPLOYCOMMAND)
values ('MySubTC, 'MyTC ',
   'com.ibm.commerce.contract.objects.MySubTCAccessBean',
   'packagename.MySubTCDeployCmd');
```

Registro del nuevo término y condición en la definición de tipo de documento del contrato

El archivo de definición de tipo de documento (DTD) `B2BTrading.dtd` especifica los diversos términos y condiciones que se pueden utilizar dentro de las políticas de negocio. Para que este nuevo término y condición esté disponible en los contratos, debe actualizar este archivo para que incluya el nuevo término y condición.

Cuando haya creado un nuevo término y condición, debe añadirlo a la definición TermCondition y crear un nuevo elemento que describa el término y condición.

Para actualizar el archivo B2BTrading.dtd, haga lo siguiente:

1. Vaya al siguiente directorio:
   - Windows *unidad*:\WebSphere\CommerceServer\xml\trading
   - AIX /usr/WebSphere/CommerceServer/xml/trading
   - Solaris /opt/WebSphere/CommerceServer/xml/trading
   - 400 /QIBM/ProdData/WebCommerce/xml/trading
2. Abra el archivo B2BTrading.dtd.
3. Actualice la definición TermCondition con el nuevo término y condición. Por ejemplo, la actualización se muestra en negrita en la siguiente definición de TermCondition:

   ```
   <!ELEMENT TermCondition (TermConditionDescription?,Participant*,
   CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
   PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
   PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
   OrderTC|MyTC))>
   ```

   Las líneas se muestran partidas sólo a efectos de visualización.
4. Ahora añada el nuevo elemento al archivo B2BTrading.dtd. Por ejemplo, a continuación se muestra la actualización para añadir el elemento MyTC, que hace referencia a una política de negocio y tiene dos atributos necesarios.

   ```
   <!ELEMENT MyTC (MySubTC)>
   <!ELEMENT MySubTC (PolicyReference)>
   <!ATTLIST MySubTC
      attr1 CDATA #REQUIRED
      attr2 CDATA #REQUIRED
   >
   ```
5. Guarde el archivo.

Creación de un nuevo bean enterprise CMP para el término y condición

Debe crear un nuevo bean enterprise CMP para el objeto de término y condición. El bean se crea para el subtipo de término y condición.

Tenga en cuenta que, normalmente, al crear nuevos beans enterprise colocaría los beans en su propio grupo EJB, en lugar de incluirlos en uno de los grupos EJB que contienen beans de entidad de WebSphere Commerce. En este caso, sin embargo, dado que todos los nuevos beans de entidad para términos y condiciones deben heredar del bean TermCondition de WebSphere Commerce, debe colocar sus nuevos beans de término y condición en el grupo EJB WCS Contract.

Para crear el nuevo bean enterprise CMP para el objeto de término y condición, haga lo siguiente en VisualAge para Java:

1. Utilice un asistente para crear el nuevo bean enterprise de la siguiente manera:
   a. En el Entorno de trabajo de VisualAge para Java, seleccione la pestaña EJB.
   b. Resalte y luego pulse con el botón derecho del ratón en el grupo EJB **WCS Contract** y seleccione **Añadir > Bean enterprise con herencia**.
      Se abre el SmartGuide Crear bean enterprise con herencia.
   c. En el SmartGuide, entre la información adecuada para su bean. Por ejemplo, la tabla siguiente muestra valores de ejemplo.

Atributo	Valor
Nombre del bean	MySubTC

Atributo	Valor
Heredar de	`TermCondition`
Paquete	`com.ibm.commerce.contract.objects`
Clase de bean	`MySubTCBean`
Interfaz remota	`MySubTC`
Interfaz inicial	`MySubTCHome`

d. Pulse **Añadir** para añadir los campos CMP al bean y cree nuevos campos para el bean, según sea necesario. Para este ejemplo, se crean dos nuevos campos CMP utilizando la siguiente información:

Atributo	Valor
Nombre de campo	`attr1`
Tipo de campo	`String`
Acceso con métodos get y set	habilitar
Promocionar métodos get y set a interfaz remota	habilitar

Atributo	Valor
Nombre de campo	`attr2`
Tipo de campo	`Integer`
Acceso con métodos get y set	habilitar
Promocionar métodos get y set a interfaz remota	habilitar

e. Pulse **Terminar**.

2. El paso siguiente es correlacionar los campos del nuevo bean con columnas de la tabla TERMCOND. Para crear esta información de correlación, haga lo siguiente:
   a. En el menú **EJB**, seleccione **Abrir en > Correlaciones de esquemas**. Se abre el Examinador de correlaciones.
   b. En el panel Correlaciones de almacenes de datos del examinador de correlaciones, efectúe una doble pulsación en **WCS Contract**.
   c. En el panel Clases persistentes, efectúe una doble pulsación en **TermCondition** y luego seleccione **MySubTC**.
   d. En el menú Correlaciones de tablas, seleccione **Nueva correlación de tabla > Añadir correlación de tabla de herencia única**. Se abre el Editor de correlación de tabla de herencia única.
   e. En el campo **Valor discriminador**, entre el valor de TCSUBTYPE_ID. Por ejemplo, en este caso entre `'MySubTC'` (incluya las comillas) y pulse **Aceptar**.
   f. Asegúrese de que **MySubTC** siga seleccionado en el panel Clases persistentes. En el panel Correlaciones de tablas, resalte y pulse con el botón derecho del ratón en la tabla TERMCOND. Seleccione Editar correlaciones de propiedades. Se abre el Editor de correlaciones de propiedades.
   g. En el Editor de correlaciones de propiedades, establezca los atributos de la siguiente manera:

Atributo de clase	Tipo de correlación	Columna de tabla
attr1	Simple	STRINGFIELD2
attr2	Simple	INTEGERFIELD1

y pulse **Aceptar**.

h. En el menú Correlaciones de almacenes de datos, seleccione Guardar correlación de almacenes de datos. Entre la información siguiente al guardar la correlación:

Atributo	Valor
Proyecto	IBM WCS Enterprise Beans
Paquete	WCSContract EJB Reserved
Nombre de clase	WCSContractMap

Pulse **Terminar** y luego cierre el Examinador de correlaciones.

3. En el nuevo bean enterprise (es decir, en MySubTCBean) cree un nuevo método `ejbCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement)`, de la forma siguiente:

```
public void ejbCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
   java.rmi.RemoteException, javax.naming.NamingException,
   javax.ejb.RemoveException {
    _initLinks();
    super.ejbCreate (argTradingId, argElement);
    this.attr1= null;
    this.attr2= null;
}
```

4. Cree un nuevo método `ejbPostCreate(java.lang.Long argTradingId, org.w3c.dom.Element argElement)` de la forma siguiente:

```
public void ejbPostCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException, javax.ejb.FinderException,
   java.rmi.RemoteException, javax.naming.NamingException,
   javax.ejb.RemoveException
   {
    parseXMLElement(argElement);
   }
```

5. Modifique el método `parseXMLElement(org.w3c.dom.Element argElement)` en MySubTCBean, de la forma siguiente:

```
public void parseXMLElement(org.w3c.dom.Element argElement) throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException,
   javax.ejb.RemoveException
   {
      super.parseXMLElement(argElement);

      if (argElement == null)
         return;

      String nodeName = argElement.getNodeName();
      if (nodeName.equals("TCCopy"))
         return;

      // obtener elemento "MyTC"
      Element eMyTC = ContractUtil.getElementByTag(argElement,"MyTC");
```

```
        // obtener elemento "MySubTC" de elemento "MyTC"
        Element eMySubTC = ContractUtil.getElementByTag(eMyTC ,"MySubTC");
        this.attr1 = eMySubTC .getAttribute("attr1").trim();
        this.attr2 = new Integer (eMySubTC .getAttribute("attr2").trim());

        // obtener elemento "PolicyReference" de "MySubTC"
        Element ePolicyReference = ContractUtil.getElementByTag(eMySubTC,
           "PolicyReference");
        parseElementPolicyReference(ePolicyReference);

    }
```

6. Modifique el método createNewVersion(Long argNewTradingId) en MySubTCBean, de la forma siguiente:

```
public Long createNewVersion(Long argNewTradingId) throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException,
   javax.ejb.RemoveException,
   org.xml.sax.SAXException,
   java.io.IOException

   {

   // Contratar un seqElement ya que tcSequence no puede ser un nulo
   Element seqElement = ContractUtil.getSeqElementFromTCSequence(
      this.tcSequence);
   MySubTCAccessBean newTC = new MySubTCAccessBean(argNewTradingId,
     seqElement);

   Long newTCId = newTC.getReferenceNumberInEJBType();
   newTC.setInitKey_referenceNumber(newTCId.toString());
   newTC.setMandatoryFlag(this.mandatoryFlag);
   newTC.setChangeableFlag(this.changeableFlag);
   // establecer columnas para este TC específico
   newTC.setAttr1(this.attr1);
   newTC.setAttr2(this.attr2);
   newTC.commitCopyHelper();

   return newTCId;
}
```

7. Modifique el método getXMLString() en MySubTCBean, de la forma siguiente:

```
public String getXMLString() throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException
   {
      String xmlTC = "      <MyTC>"  +
                     "%XML_POLICYREFERENCE%" +
                     "         <MySubTC attr1=\"" + this.attr1 +
                     "\"  attr2=\"" + this.attr2.toString() + "\"/>"
                     "         '>" +
                     "      <MYTC></MYTC>" ;
      String xmlPolicy = getXMLStringForElementPolicyReference(
         "ProductSet") ;
      xmlTC = ContractUtil.replace( xmlTC, "%XML_POLICYREFERENCE%",
         xmlPolicy );

      return xmlTC;
}
```

8. Modifique el método markForDelete() en MySubTCBean, de la forma siguiente:

```
public void markForDelete() throws
   javax.ejb.CreateException,
   javax.ejb.FinderException,
   java.rmi.RemoteException,
   javax.naming.NamingException
   {
      // código: elimine entradas de las tablas asociadas que
      // no se pueden suprimir mediante supresión en cascada
}
```

9. Asegúrese de que el método `ejbCreate` se haya añadido a la interfaz inicial y que todos los demás métodos modificados se hayan añadido a la interfaz remota.
10. El paso siguiente es crear un bean de acceso para el bean de entidad MySubTC de la siguiente manera:
    a. Pulse con el botón derecho del ratón en el bean de entidad **MySubTC** y seleccione **Añadir > Bean de acceso**.
    Se abre el SmartGuide Crear bean de acceso.
    b. Asegúrese de entrar la siguiente información:

Tabla 1.

Atributo	Valor
Grupo EJB	`WCSContract`
Bean enterprise	`MySubTC`
Nombre de bean de acceso	`MySubTCAccessBean`
Tipo de bean de acceso	`CopyHelper para Bean de entidad`

    y pulse **Siguiente**.
    c. En la lista desplegable **Seleccionar método local para constructor sin argumentos**, seleccione **findByPrimaryKey(TermConditionKey)**
    d. Para **initKey_referenceNumber** (en la columna Propiedades iniciales), establezca **Conversor** en `com.ibm.commerce.base.objects.WCStringConverter` y pulse **Siguiente**.
    e. Para todos los campos nuevos que se han añadido, asegúrese de que **CopyHelper** esté seleccionado y establezca el valor de conversor para cada uno en `com.ibm.commerce.base.objects.WCStringConverter`. Pulse **Terminar**.
    Una vez completada la generación del código, puede ver el nuevo código haciendo lo siguiente: vaya a la pestaña **Proyectos**, expanda el proyecto **IBM WCS Enterprise Beans** y luego expanda el paquete **com.ibm.commerce.contract.objects**.
11. Vuelva a la pestaña EJB, pulse con el botón derecho del ratón en el bean enterprise **MySubTC** y seleccione **Generar código desplegado**.
12. También debe volver a generar el código desplegado para el bean padre (el bean TermCondition) y todos los beans hermano (todos los demás beans del grupo EJB WCS Contract que contienen "TC" en el nombre). Tenga en cuenta que si ha añadido un nuevo campo o ha modificado la interfaz remota del bean TermCondition existente, tendrá que volver a generar los beans de acceso para el mismo, así como todos sus beans hijo.
    Para volver a generar el código desplegado, haga lo siguiente:
    a. Resalte el bean TermCondition y todos los demás beans que contengan "TC" en el nombre (por ejemplo, DisplayCustomizationTC, FulfillmentTC e InvoiceTC son algunos de los beans hermano).
    b. Con todos estos beans resaltados, pulse el botón derecho del ratón y seleccione **Generar código desplegado**.

13. El paso siguiente es modificar métodos que están en el mandato de tarea ValidateContractCmd. En este mandato, hay tres métodos que quizá desee modificar para dar soporte al nuevo objeto de término y condición. Son:
    - validateTCType()
      Este método comprueba el tipo de término que puede estar en el contrato. Por ejemplo, InvoiceTC pertenece a cuentas y, por tanto, no puede aparecer en un contrato.
    - validateTCOccurrence()
      Este método comprueba la aparición de los términos. Por ejemplo, en la implementación por omisión de este método, un contrato ha de tener un PriceTC como mínimo.
    - otherValidateCheck()
      La implementación por omisión de este método está vacía. Puede añadir cualquier validación adicional que no esté en ninguno de los dos primeros métodos.
14. Si debe desplegarse el término y condición, debe crear un nuevo mandato de despliegue y registrar este mandato en la base de datos. Si es necesario, haga lo siguiente:
    a. En este ejemplo, la nueva interfaz de mandato de despliegue se denomina MySubTCDeployedCmd y la clase de implementación se denomina MySubTCDeployedCmdImpl. Además, el mandato está empaquetado en el paquete packagename. Para registrar este mandato, emita el siguiente mandato de SQL:

       ```
       insert into CMDREG (STOREENT_ID, INTERFACENAME, CLASSNAME, TARGET)
       values (0, 'packagename.MySubTCDeployCmd',
       'packagename.MySubTCDeployCmdImpl', 'Local');
       ```

    b. En el paquete packagename, cree la nueva interfaz MySubTCDeployedCmd. Esta interfaz debe ampliar la interfaz de mandatos com.ibm.commerce.contract.commands.DeployTCCmd. A continuación se describe la nueva interfaz de mandatos:

       ```
       public interface MySubTCDeployCmd extends
          com.ibm.commerce.contract.commands.DeployTCCmd
          {
              // código personalizado
          }
       ```

       Hay un parámetro protegido abTC y un método denominado getTargetStoreId() en DeployTCCmd. El valor de abTC es MySubTCAccessBean y el método getTargetStoreId() devuelve el identificador de la tienda en la que se despliega el contrato.

    c. En el mismo paquete, cree la clase de implementación MySubTCDeployCmdImpl. Esta clase de implementación debe ampliar com.ibm.commerce.contract.commands.DeployTCCmdImpl. A continuación se describe la nueva clase de implementación de mandatos:

       ```
       public class MySubTCDeployCmdImpl
          extends com.ibm.commerce.contract.commands.DeployTCCmdImpl
          implements MySubTCDeployCmd
          {
            // código del cliente
          }
       ```

Actualización de WebSphere Commerce Accelerator para utilizar un nuevo término y condición

Una vez creados los nuevos términos y condiciones, puede actualizar WebSphere Commerce Accelerator para que se pueda utilizar para crear nuevos contratos que

incluyan esos términos y condiciones. La actualización de WebSphere Commerce Accelerator para este propósito incluye los siguientes pasos:

1. Crear un nuevo archivo JavaScript para los nuevos términos y condiciones. Para el propósito del ejemplo en esta sección, a este archivo se le denomina `Extensions.js`.
2. Crear una nueva plantilla JSP que incluya una sección HTML en la que un usuario pueda entrar la información necesaria los nuevos términos y condiciones. Para el propósito del ejemplo en esta sección, a este archivo se le denomina `ContractMyTC.jsp`.
3. Crear un nuevo bean de datos para los nuevos términos y condiciones. Para el propósito del ejemplo en esta sección, a este archivo se le denomina `MyTCDataBean`.
4. Registrar la nueva vista en la tabla VIEWREG.
5. Actualizar el archivo `ContractRB_`*`entorno_nacional`*`.properties` para que incluya los nuevos recursos.
6. Editar el archivo `ContractNotebook.xml` para que incluya la nueva página.

Cada uno de estos pasos se describe con más detalle en las secciones siguientes.

Creación del nuevo archivo JavaScript: El primer paso para actualizar WebSphere Commerce Accelerator para que utilice los nuevos términos y condiciones es crear un nuevo archivo JavaScript para ellos. Como referencia, puede consultar el siguiente archivo de ejemplo:

- Windows `unidad:\WebSphere\CommerceServer\samples\contract\ Extensions.js`
- Windows `unidad:\WebSphere\CommerceServerDev\samples\contract\ Extensions.js`
- AIX `/usr/WebSphere/CommerceServer/samples/contract/Extensions.js`
- Solaris `/opt/WebSphere/CommerceServer/samples/contract/Extensions.js`
- 400 `/QIBM/ProdData/WebCommerce/samples/contract/Extensions.js`

Para utilizar este archivo de ejemplo, cópielo en el directorio siguiente:

- Windows `unidad:\WebSphere\AppServer\installedApps\ WC_Enterprise_App_nombreInstancia.ear\wctools.war\ javascript\tools\contract`
- AIX `/usr/WebSphere/AppServer/installedApps/ WC_Enterprise_App_nombreInstancia.ear/wctools.war/ javascript/tools/contract`
- Solaris `/opt/WebSphere/AppServer/installedApps/ WC_Enterprise_App_nombreInstancia.ear/wctools.war/ javascript/tools/contract`
- 400 `/QIBM/UserData/WebASAdv4/nombreInstanciaAdmin_WAS/ installedApps/WC_Enterprise_App_nombreInstancia.ear/ wctools.war/javascript/tools/contract`

En este nuevo archivo, debe crear un objeto JavaScript para almacenar los datos para el nuevo término y condición. Esto se muestra en la sección de código siguiente:

```
function ContractMyTCModel() {

   this.tcReferenceNumber = "";
```

```
   this.policyReferenceNumber = "";

   this.attr1 = "";
   this.attr2 = "";

   this.policyList = new Array();
   this.selectedPolicyIndex = "0";
}
```

También debe crear un nuevo objeto JavaScript para someter el nuevo término y condición. Esto debe hacerse de forma coherente con las ampliaciones efectuadas en el archivo B2BTrading.dtd. Esto se muestra en la sección de código siguiente:

```
function submitMyTC(termsAndConditions) {

   var tcModel = get("ContractMyTCModel");

   if (tcModel != null) {

      var myTC = new Object();
      myTC.MyTC = new Object();
      myTC.MyTC.MySubTC = new Object();
      myTC.MyTC.MySubTC.attr1 = tcModel.attr1;
      myTC.MyTC.MySubTC.attr2 = tcModel.attr2;

      myTC.MyTC.PolicyReference = new Object();
      myTC.MyTC.PolicyReference.policyName =
         tcModel.policyList[tcModel.selectedPolicyIndex].policyName;
      myTC.MyTC.PolicyReference.policyType = "ProductSet";
      myTC.MyTC.PolicyReference.storeIdentity =
         tcModel.policyList[tcModel.selectedPolicyIndex].storeIdentity;
      myTC.MyTC.PolicyReference.Member =
         tcModel.policyList[tcModel.selectedPolicyIndex].member;

      if (tcModel.tcReferenceNumber != "") {
         // Cambiar el término y condición
         myTC.action = "update";
         myTC.referenceNumber = tcModel.tcReferenceNumber;
      }
      else {
         // Crear un nuevo término y condición
         myTC.action = "new";
      }

      termsAndConditions[termsAndConditions.length] = myTC;
   }

   return true;
}
```

Creación de la nueva plantilla JSP: El paso siguiente es crear una nueva plantilla JSP que incluya una sección HTML en la que un usuario pueda entrar la información necesaria para el nuevo término y condición. Como referencia, puede consultar el siguiente archivo de ejemplo:

- Windows *unidad*:\WebSphere\CommerceServer\samples\contract\ ContractMyTC.jsp
- Windows *unidad*:\WebSphere\CommerceServerDev\samples\contract\ ContractMyTC.jsp
- AIX /usr/WebSphere/CommerceServer/samples/contract/ ContractMyTC.jsp
- Solaris /opt/WebSphere/CommerceServer/samples/contract/ ContractMyTC.jsp
- 400 /QIBM/ProdData/WebCommerce/samples/contract/ ContractMyTC.jsp

Para utilizar este archivo de ejemplo, cópielo en el directorio siguiente:

- Windows *unidad*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*nombreInstancia*.ear\wctools.war\tools\contract
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wctools.war/tools/contract
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wctools.war/tools/contract
- 400 /QIBM/UserData/WebASAdv4/*nombreInstanciaAdmin_WAS*/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wctools.war/tools/contract

La sección de código siguiente muestra un ejemplo de sección HTML de una plantilla JSP que se puede utilizar para MyTC.

```
<!--
////////////////////////////////////
// SECCIÓN HTML
////////////////////////////////////
-->

<BODY onLoad="onLoad()" class="content">

   <H1>
   <%= contractsRB.get("MyTCHeading") %>
   </H1>

<FORM NAME="MyTCForm">

   <%= contractsRB.get("MyTCAttr1Label") %>
   <BR>
   <INPUT type=text name=Attr1 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCAttr2Label") %>
   <BR>
   <INPUT type=text name=Attr2 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCPolicyLabel") %>
   <BR>
   <SELECT NAME="PolicyList" SIZE="1">
   </SELECT>

</FORM>
```

Creación del nuevo bean de datos: En este paso, creará un nuevo bean de datos que carga los datos necesarios del bean de acceso MySubTC. Las secciones relevantes de código se muestran en la siguiente sección de código:

```
public class MyTCDataBean extends MySubTCAccessBean
   implements SmartDataBean, Delegator {
      private java.lang.Long contractId;
      private boolean hasMyTC = false;
      private CommandContext iCommandContext;


      /**
       * Constructor por omisión de MyTCDataBean.
       */
      public MyTCDataBean() {
      }
      /**
```

```
        * Constructor de MyTCDataBean.
        */
       public MyTCDataBean(Long newContractId) {
        contractId = newContractId;
       }


       /*
        * inserta los atributos de TermConditionAccessBean
        */
       public void populate() throws Exception {

             Enumeration myTCEnum = new TermConditionAccessBean().
                      findByTradingAndTCSubType(contractId, "MySubTC");
                   if (myTCEnum != null) {
                         // supongamos un contrato que sólo tiene un MyTC

                         setEJBRef(((TermConditionAccessBean)
                               myTCEnum.nextElement()).getEJBRef());
                         refreshCopyHelper();
                         hasMyTC = true;
                   }

}
```

Registro de la nueva vista en la tabla VIEWREG: Debe registrar la vista que acaba de crear en la tabla VIEWREG. A continuación se muestra una sentencia SQL de ejemplo para registrar la nueva vista.

```
insert into VIEWREG(VIEWNAME,DEVICEFMT_ID,STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, HTTPS, INTERNAL)
values ('ContractMyTCPanelView', -1, 0,
   'com.ibm.commerce.tools.command.ToolsForwardViewCommand',
   'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
   'docname=tools/contract/ContractMyTC.jsp', 1, 1)
```

Actualización del archivo ContractRB_*entorno_nacional*.properties: Debe actualizar el siguiente archivo de propiedades con información específica del nuevo término y condición:

- Windows *unidad*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*nombreInstancia*.ear\properties\com\ibm\commerce\tools\contract\properties\ContractRB_*entorno_nacional*.properties
- AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*entorno_nacional*.properties
- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*entorno_nacional*.properties
- 400 /QIBM/UserData/WebASAdv4/*NombreInstanciaAdmin_WAS*/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*entorno_nacional*.properties

A continuación se muestra un ejemplo de la información que añadiría al archivo.

```
MyTCHeading=Mi TC
attr1Empty=Debe entrar el atributo uno.
attr2Empty=Debe entrar el atributo dos.
```

```
attr1TooLong=El atributo uno es demasiado largo.
attr2TooLong=El atributo dos es demasiado largo.
MyTCAttr1Label=Atributo uno (obligatorio)
MyTCAttr2Label=Atributo dos (obligatorio)
MyTCPolicyLabel=Política
```

Edición del archivo `ContractNotebook.xml`: El último paso para incluir nuevos términos y condiciones en WebSphere Commerce Accelerator es actualizar el siguiente archivo para que incluya la nueva página.

- Windows *unidad*:\WebSphere\CommerceServer\xml\tools\contract\ContractNotebook.xml
- AIX /usr/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- Solaris /opt/WebSphere/CommerceServer/xml/tools/contract/ContractNotebook.xml
- 400 /QIBM/UserData/WebCommerce/instances/*nombreInstancia*/xml/tools/contract/ContractNotebook.xml

A continuación se muestra una sección de código de ejemplo que se utiliza para incluir la nueva página en este ejemplo.

```
<panel name="MyTCHeading"
    url="ContractMyTCPanelView"
    parameters="contractId,accountId"
    helpKey="MC.contract.MyTCPanel.Help" />
```

Importación del nuevo contrato utilizando el nuevo término y condición

Como alternativa a la actualización de las herramientas de WebSphere Commerce para utilizar un nuevo término y condición, puede utilizar el mandato de importación de contrato (consulte la ayuda en línea de WebSphere Commerce para obtener información sobre este mandato) para importar un nuevo contrato que incluya este término y condición. Después de importarlo, la sección relevante del archivo `Contract.xml` tiene este aspecto:

```
<TermCondition>
   <MyTC>
      <MySubTC attr1="adc" attr2="123" />
         <PolicyReference policyName = "Product Set 1"
                          policyType = "ProductSet"
                          storeIdentity = "StoreGroup1" >
            <Member>
               <User distinguishName = "uid=wcsadmin,o=Root Organization"/>
            </Member>
         </PolicyReference>
      </MyTC>
 </TermCondition>
```

Invocación de la nueva política de negocio

Una vez que haya creado una nueva política de negocio y la haya asociado con al menos un objeto de términos y condiciones, debe actualizar la lógica de su aplicación para que invoque los nuevos mandatos de política de negocio.

Los mandatos de política de negocio se invocan desde los mandatos de controlador y de tarea.

La fábrica de mandatos se utiliza para invocar los mandatos de política de negocio. Hay dos métodos `create` que se pueden utilizar para invocar los mandatos de

política de negocio. El primero se utiliza para invocar un mandato de política de negocio cuando sólo hay un mandato de política de negocio asociado a la política de negocio. Esto se muestra en el extracto de código siguiente:

```
CommandFactory createBusinessPolicyCommand(Long policyId);
```

El segundo método se utiliza para invocar un mandato de política de negocio cuando hay más de un mandato asociado a la política de negocio. Esto se muestra en el extracto de código siguiente:

```
CommandFactory createBusinessPolicyCommand(Long policyId, String cmdIfName);
```

En el ejemplo anterior, se utiliza `cmdIfName` para especificar el nombre de la interfaz del mandato de política de negocio a crear.

La fábrica de mandatos busca el objeto de política en la tabla POLICYCMD para determinar el mandato que implementa esta política. También busca cualquier propiedad por omisión en la tabla y las establece como requestProperties en el mandato de política de negocio.

En la siguiente sección de código se muestra un ejemplo de cómo invocar una política de reembolso:

```
RefundPolicyCmd  cmd;

   ////////////////////////////////////////////////////////
   // Obtener el id de política de reembolso del objeto refundTC //
   // y utilizarlo para crear el mandato de política.            //
   ////////////////////////////////////////////////////////
 cmd = (RefundPolicyCmd) CommandFactory
         createPolicyCommand (refundTC.getRefundPolicy);

  cmd.execute()
```

Creación de un contrato

El siguiente paso para integrar completamente la ampliación del modelo de contrato en el proceso de negocio es crear un contrato que incluya los términos y condiciones que hacen referencia a la nueva política de negocio. El contrato se puede crear utilizando WebSphere Commerce Accelerator o uno de los mandatos de URL de contrato (ContractImportApprovedVersion y ContractImportDraftVersion). Para obtener más información sobre la creación de contratos, consulte la ayuda en línea de WebSphere Commerce.

Escenarios de personalización de contrato

Esta sección proporciona una visión general de los pasos relacionados con el siguiente escenario de personalización de contrato:

- Habilitación de una rebaja

Escenario de rebaja

En este escenario de ejemplo se crea una rebaja de importe fijo. Puesto que la tienda de ejemplo ToolTech no incluye un término y condición ni un tipo de política que se corresponda con el escenario de rebaja, estos deben crearse. Además, debe crearse un nueva política de negocio, así como una tabla de base de datos para almacenar los códigos de rebaja.

Para implementar este escenario de rebaja, es necesario realizar los siguientes pasos generales:

1. Creación de la tabla de base de datos XREBATECODE y de un bean de entidad XRebateCodeBean correspondiente que se utiliza para acceder a la información de esta tabla.
2. Creación de una nueva política de negocio 5DollarRebate realizando las subtareas siguientes:
   a. Crear el nuevo tipo de política de negocio correspondiente. Esto define la interfaz (RebatePolicyCmd) que el nuevo mandato de política de negocio implementará.
   b. Crear el nuevo mandato de política de negocio CalculateRebateCmdImpl.
   c. Registrar el nuevo mandato de política de negocio y el nuevo tipo de política de negocio en la base de datos.
3. Creación de un nuevo término y condición (RebateTC) para la rebaja realizando las subtareas siguientes:
   a. Registrar el término y condición RebateTC en la base de datos.
   b. Actualizar el archivo `B2BTrading.dtd` para reflejar el nuevo término y condición, RebateTC.
   c. Crear un nuevo bean enterprise para RebateTC.
   d. Actualizar WebSphere Commerce Accelerator para que refleje el nuevo RebateTC.
4. Creación de un nuevo contrato que utilice el término y condición RebateTC.
5. Integración de la nueva política de negocio en el flujo de compra.

Cada uno de estos pasos se describe con más detalle en las secciones siguientes.

Paso 1: Creación de la nueva tabla y el bean enterprise

Dado que el esquema de base de datos existente no incluye la especificación de un importe y un código de rebaja, debe crearse una nueva tabla. En general, cuando se crea una nueva tabla, también se crea un nuevo bean de entidad que se utiliza para acceder a la información contenida en esa tabla.

A efectos de este ejemplo, supongamos que se crea la tabla de base de datos XREBATECODE siguiente.

Tabla 2. Tabla de base de datos XREBATECODE

	Nombre de columna		
	REBATECODE_ID	AMOUNT	CURRENCY
Datos de ejemplo	201	5	CAD
	202	10	CAD

Además, se crearía un nuevo bean de entidad CMP (XRebateCodeBean). Para obtener más información sobre cómo crear este bean, consulte el apartado “Creación de un bean enterprise CMP nuevo” en la página 51.

Paso 2: Creación de la política de negocio “5DollarRebate”

Para crear esta nueva política de negocio, debe realizar los pasos siguientes:

1. Crear la nueva interfaz de tipo de política de negocio. Esta es la interfaz `RebatePolicyCmd` que el mandato `CalculateRebateCmdImpl` implementará.
2. Crear el nuevo mandato de política de negocio `CalculateRebateCmdImpl`.
3. Registrar la nueva política de negocio y el nuevo mandato de política de negocio en la base de datos.

Creación del tipo de política de negocio "Rebate": Dado que no hay un tipo de política de negocio existente que se corresponda con rebajas, debe crearse uno nuevo. Para crear un nuevo tipo de política de negocio, es necesario definir y registrar un tipo de política en la base de datos. Deben actualizarse las tablas siguientes:

- POLICYTYPE
- PLCYTYCMIF
- PLCYTYPDSC

En este escenario, para crear el nuevo tipo de política REBATE, se utilizarían las siguientes sentencias SQL:

```
insert into POLICYTYPE (POLICYTYPE_ID) values ('Rebate');
insert into PLCYTYCMIF (POLICYTYPE_ID, BUSINESSCMDIF)
   values ('Rebate',
   'com.mycompany.mybusinesspolicycommands.RebatePolicyCmd');
insert into  PLCYTYPDSC (POLICYTYPE_ID, LANGUAGE_ID, DESCRIPTION)
   values ('Rebate', -5,
   'Tipo de política de rebaja.');
```

Como resultado, la tabla siguiente muestra las columnas relevantes de la tabla PLCYTYCMIF, que muestra la relación entre el tipo de política y el mandato de política de negocio con el que está relacionado.

Tabla 3. Actualizaciones realizadas en la tabla PLCYTYCMIF

	Nombre de columna	
	POLICYTYPE_ID	BUSINESSCMDIF
Datos de ejemplo	Rebate	com.mycompany.mybusinesspolicycommands.RebatePolicyCmd

También debe codificar la nueva interfaz `RebatePolicyCmd`. Esta interfaz debe ampliar la interfaz `com.ibm.commerce.command.BusinessPolicyCommand`. Tal como se sugiere en la tabla anterior, empaquete esta interfaz en su propio paquete.

Creación del mandato de política de negocio CalculateRebateCmdImpl: Para crear el nuevo mandato de política de negocio, debe crear un nuevo mandato denominado `CalculateRebateCmdImpl` que amplíe la clase de implementación `com.ibm.commerce.command.BusinessPolicyCommandImpl`. Este mandato debería implementar la interfaz `RebatePolicyCmd` que se ha creado en el paso anterior.

Tenga en cuenta que en este ejemplo, el nombre de la interfaz y el nombre del mandato son diferentes. Estos nombres se eligieron deliberadamente para mostrar que puede haber muchos mandatos de política de negocio que implementen el tipo de política de negocio de rebaja. Cada implementación (es decir, cada mandato de política de negocio) implementaría entonces la rebaja de un modo exclusivo.

La lógica del mandato depende de la implementación particular de cómo el cliente va a adquirir las mercancías. Además, este mandato `CalculateRebateCmdImpl` lo debería invocar otro mandato de controlador o de tarea de su aplicación.

Registro de la nueva política de negocio y el nuevo mandato de política de negocio: La nueva política de negocio debe registrarse en la base de datos. También debe registrar la relación entre la nueva política de negocio y el nuevo mandato de política de negocio.

Para registrar esta información, puede utilizar el mandato com.ibm.commerce.contract.commands.PolicyAddCmd. A continuación se muestra un ejemplo del uso del mandato PolicyAdd para este escenario:

```
http://localhost:8080/webapp/wcs/stores/servlet/PolicyAdd?
   type=Rebate&name=5DollarRebate&plcyStoreId=-1
   &cmd_1=com.mycompany.mybusinesspolicycommands.CalculateRebateCmdImpl
   &startDate=2002-05-08%2000:00:00&endDate=2003-05-09%2000:00:00
   &commonProps=rebatecode_id%3D501&URL=unURLdeRedirección
```

Tenga en cuenta que los caracteres reservados de URL deben sustituirse por sus códigos ASCII para las propiedades de entrada. Por lo tanto, el signo igual (=) típico se sustituye por "%3D", el símbolo & se sustituye por "%26" y el carácter de espacio se sustituye por "%20". El formato de fecha utilizado en el ejemplo anterior es aaaa-mm-dd hh:mm:ss, con código ASCII sustituyendo a los caracteres reservados de URL.

Las tablas siguientes muestran las columnas relevantes de las tablas de base de datos afectadas, después de realizar las actualizaciones.

Tabla 4. Actualizaciones realizadas en la tabla POLICY

	Nombre de columna				
	POLICY _ID	POLICY NAME	POLICYTYPE _ID	STOREENT _ID	PROPERTIES
Datos de ejemplo	301	5DollarRebate	Rebate	-1	rebatecode_id= 201

Tenga en cuenta que también se presupone que los valores de fecha de inicio y fecha de finalización están establecidos en null.

Tabla 5. Actualizaciones realizadas en la tabla POLICYCMD

	Nombre de columna		
	POLICY _ID	BUSINESS CMDCLASS	PROPERTIES
Datos de ejemplo	301	com.mycompany. mybusinesspolicycommands. CalculateRebateCmdImpl	null

Como resultado, ahora tiene una nueva política de negocio denominada "5DollarRebate" que está relacionada con el mandato de política de negocio CalculateRebateCmd.

Paso 3: Creación del término y condición "RebateTC"

Para crear el término y condición "RebateTC", es necesario realizar los pasos siguientes:

1. Registrar el término y condición RebateTC en la base de datos.
2. Actualizar el archivo B2BTrading.dtd para que refleje el nuevo RebateTC.
3. Crear un nuevo bean enterprise para RebateTC.
4. Actualizar WebSphere Commerce Accelerator para que refleje el nuevo RebateTC.

Registro del término y condición "RebateTC" en la base de datos: Cuando crea un nuevo objeto de término y condición, debe actualizar el esquema de base de datos para que incluya este objeto. Las tablas de base de datos que deben actualizarse son TCTYPE y TCSUBTYPE.

La siguiente sentencia SQL muestra un ejemplo de cómo registrar RebateTC en la base de datos:

```
insert into TCTYPE (TCTYPE_ID) values ('RebateTC');
insert into TCSUBTYPE (TCSUBTYPE_ID, TCTYPE_ID, ACCESSBEANNAME,
   DEPLOYCOMMAND)
values ('RebateTC, 'RebateTC ',
   'com.ibm.commerce.contract.objects.RebateTCAccessBean',
   null);
```

Las tablas siguientes muestran un extracto de las columnas relevantes de las tablas TCTYPE y TCSUBTYPE.

Tabla 6. Actualizaciones realizadas en la tabla TCTYPE

	Nombre de columna
	TCTYPE_ID
Datos de ejemplo	RebateTC

Tabla 7. Actualizaciones realizadas en la tabla TCSUBTYPE

	Nombre de columna			
	TCSUBTYPE _ID	TCTYPE _ID	ACCESSBEAN NAME	DEPLOY COMMAND
Datos de ejemplo	RebateTC	RebateTC	com.ibm.commerce. contract.objects. RebateTCAccessBean	null

Actualización del archivo B2BTrading.dtd para que refleje el nuevo RebateTC: Para que el nuevo término y condición pase a estar disponible en los contratos, debe actualizar el archivo `B2BTrading.dtd` para que incluya el nuevo término y condición. Al actualizar este archivo, debe añadir el nuevo término y condición a la definición TermCondition y, a continuación, crear un nuevo elemento que describa el término y condición.

El texto en negrita muestra un ejemplo de cómo añadir el nuevo RebateTC a la definición TermCondition:

```
<!ELEMENT TermCondition (TermConditionDescription?,Participant*,
CreateTime?,UpdateTime?,(PriceTC|ProductSetTC|ShippingTC|FulfillmentTC|
PaymentTC|ReturnTC|InvoiceTC|RightToBuyTC|ObligationToBuyTC|
PurchaseOrderTC|OrderApprovalTC|DisplayCustomizationTC|
OrderTC|RebateTC))>
```

Las líneas se muestran partidas sólo a efectos de visualización.

A continuación debe añadir al archivo una nueva sección que describa este término y condición. El siguiente es un ejemplo para RebateTC:

```
<!ELEMENT RebateTC (PolicyReference?)>
```

Creación de un nuevo bean enterprise para RebateTC: Debe crear un nuevo bean enterprise para el nuevo RebateTC. Este nuevo bean debería heredar del bean TermCondition de WebSphere Commerce.

Normalmente, a un nuevo bean enterprise para un término y condición se le asigna el nombre del subtipo. Tenga en cuenta que en este caso, el subtipo del término y condición es igual que el tipo del término y condición y, por tanto, el nombre del bean es igual que el del tipo del término y condición.

La tabla siguiente muestra información general sobre el nuevo bean que debe crearse. Para obtener más detalles sobre el bean, incluyendo qué métodos se han de modificar, consulte el apartado “Creación de un nuevo bean enterprise CMP para el término y condición” en la página 136.

Tabla 8.

Atributo	Valor
Nombre del bean	`RebateTC`
Inherit from	`TermCondition`
Paquete	`com.ibm.commerce.contract.objects`
Clase de bean	`RebateTCBean`
Interfaz remota	`RebateTC`
Interfaz inicial	`RebateTCHome`

Actualización de WebSphere Commerce Accelerator para que incluya RebateTC: Una vez que haya creado nuevos términos y condiciones, puede actualizar WebSphere Commerce Accelerator para que se pueda utilizar para crear nuevos contratos que incluyan esos nuevos términos y condiciones. Para obtener información sobre cómo actualizar esta herramienta, consulte el apartado “Actualización de WebSphere Commerce Accelerator para utilizar un nuevo término y condición” en la página 141.

Paso 4: Creación de un nuevo contrato

Debe crear un nuevo contrato que incluya el término y condición “RebateTC” y que haga referencia a la política de negocio “5DollarRebate”. Puede utilizar WebSphere Commerce Accelerator o XML para crear un nuevo contrato. Cada uno de estos métodos para crear un nuevo contrato se describe en la ayuda en línea de WebSphere Commerce.

Las tablas siguientes muestran las actualizaciones en la columnas relevantes de las tablas de base de datos TERMCOND y POLICYTC, después de que se haya creado el contrato.

Tabla 9. Actualizaciones realizadas en la tabla TERMCOND

	Nombre de columna		
	TRADING _ID	TERMCOND _ID	TCSUBTYPE _ID
Datos de ejemplo	25	901	RebateTC

Tabla 10. Actualizaciones realizadas en la tabla POLICYTC

	Nombre de columna	
	POLICY _ID	TERMCOND_ID
Datos de ejemplo	301	901

Paso 5: Integración de la nueva política de negocio en el flujo de compra

En este escenario, se presupone que se ha añadido una nueva página a la tienda que permite al cliente conectarse y solicitar su rebaja. Cuando el cliente pulse para solicitar la rebaja, debería invocarse un mandato que llame a la nueva interfaz `RebatePolicyCmd`. Por ejemplo, podría haber un nuevo mandato de controlador

ClaimRebateCmd que llamase a `RebatePolicyCmd`. A continuación, se busca la política de negocio correcta y (en este caso) se aplica la política de negocio "5DollarRebate".

Parte 3. Entorno de desarrollo

Capítulo 8. Herramientas de desarrollo y despliegue

Este capítulo presenta las principales herramientas de desarrollo que se utilizan para personalizar una aplicación de WebSphere Commerce. Describe el proceso para el despliegue de código personalizado de VisualAge para Java en un WebSphere Commerce Server. También describe cómo desplegar código en Commerce Studio, de modo que pueda beneficiarse del uso del código personalizado al desarrollar plantillas JSP.

Entorno de desarrollo

Para crear código personalizado para WebSphere Commerce Business Edition, se recomienda el paquete de desarrollo WebSphere Commerce Studio, Business Developer Edition. Para crear código personalizado para WebSphere Commerce Professional Edition, se recomienda el paquete de desarrollo WebSphere Commerce Studio Professional Developer Edition. Estos dos paquetes incluyen todas las herramientas necesarias para crear código personalizado y llevar a cabo tareas de desarrollo de la Web.

Tanto WebSphere Commerce Studio Business Developer Edition como WebSphere Commerce Studio Professional Developer Edition proporcionan la opción de incluir una tienda de WebSphere Commerce de ejemplo que se utiliza en el componente Entorno de prueba WebSphere de VisualAge para Java. Esto simplifica la configuración del entorno de desarrollo, puesto que no es necesario que un desarrollador utilice las herramientas de WebSphere Commerce para crear una tienda y, a continuación, mover los elementos de la tienda al entorno de desarrollo.

Otro componente clave para el entorno de desarrollo es el depósito de código de WebSphere Commerce. Este depósito debe importarse al área de trabajo de VisualAge para Java, después de completar la instalación del producto. Después de importarlo, el desarrollador debe llevar a cabo algunos pasos de configuración relacionados con el Entorno de prueba WebSphere.

Después de completar todas las tareas de instalación y configuración, el desarrollador tiene una máquina de desarrollo autónoma, donde se puede crear y probar código de WebSphere Commerce. No es necesario que el desarrollador instale WebSphere Commerce en la máquina de desarrollo.

WebSphere Commerce Studio

WebSphere Commerce Studio Business Developer Edition y WebSphere Commerce Studio Professional Developer Edition incluyen VisualAge para Java, Enterprise Edition, Versión 4.0. Esta edición de VisualAge para Java incluye características como, por ejemplo, potentes herramientas para el desarrollo y la depuración de plantillas JSP avanzadas que le ayudan a desarrollar los elementos del escaparate. También incluye integración mejorada con WebSphere Studio, que le permite añadir rápidamente contenido a estas plantillas JSP, aumentando la productividad de los programadores y los desarrolladores Web. Para el desarrollo del código de aplicación interna de la oficina, incluye soporte para las características de tecnología y conectividad de Enterprise JavaBeans como soporte para la integración con otros sistemas, como CICS TS, MQSeries y SAP R/3 entre otros. Además, el Entorno de prueba WebSphere integrado de VisualAge para Java, Enterprise Edition, permite a los desarrolladores ejecutar funciones de WebSphere

Commerce sin salir de VisualAge para Java. Esto significa que la comprobación del código personalizado de WebSphere Commerce puede llevarse a cabo sin desplegar el código en WebSphere Commerce Server.

Nota: El componente Entorno de prueba WebSphere de VisualAge para Java, Enterprise Edition, Versión 4.0 ejecuta aplicaciones dentro de WebSphere Application Server V3.5.4. Tenga en cuenta que tanto WebSphere Commerce Business Edition como WebSphere Commerce Professional Edition utilizan WebSphere Application Server V4.0. Debido a esta diferencia en las versiones, antes de desplegar beans enterprise nuevos o modificados en una instancia de WebSphere Commerce que se ejecute dentro de WebSphere Application Server V4.0, debe generar código desplegado fuera de VisualAge para Java utilizando la herramienta EJBDeploy que se entrega con WebSphere Application Server V4.0. De hecho, este código desplegado debe generarse en una máquina que tenga instalado WebSphere Application Server V4.0. Consulte "Información sobre el código desplegado de EJB" en la página 159 para obtener más detalles.

Para completar las guías de aprendizaje descritas en la Parte 4, "Guías de aprendizaje" en la página 169, debe instalar WebSphere Commerce Studio Business Developer Edition o WebSphere Commerce Studio Professional Developer Edition. Consulte la publicación *WebSphere Commerce Studio Business Developer Edition, Guía de instalación* o *WebSphere Commerce Studio Professional Developer Edition, Guía de instalación* para obtener más información sobre la instalación de Commerce Studio y la configuración de VisualAge para Java.

Características y funciones de VisualAge para Java

Esta *Guía del programador* no está pensada para enseñarle a utilizar VisualAge para Java. Con respecto a este producto, normalmente se muestra cómo efectuar una tarea en VisualAge para Java para poder efectuar una tarea de programación y así crear una aplicación de comercio electrónico para WebSphere Commerce. Por esto, si es un usuario nuevo de VisualAge para Java, aprenderá a realizar algunas tareas en VisualAge para Java en las guías de aprendizaje incluidas en este manual. Sin embargo, debe consultar la documentación, guías de aprendizaje y cursos de VisualAge para Java para llegar a ser un experto en el uso de esta herramienta.

Por ejemplo, el tema (Opcional) Utilización del depurador en VisualAge para Java de la guía de aprendizaje proporciona una introducción rápida a cómo utilizar el depurador para ver el valor de las variables en un mandato de tarea durante la ejecución de código. El componente de depuración de VisualAge para Java es una herramienta muy potente y debería consultar la documentación de VisualAge para Java para obtener más detalles sobre sus características y cómo utilizarlas.

Depósito de código de WebSphere Commerce

Para crear código personalizado para una aplicación de WebSphere Commerce, debe importar un depósito de código de WebSphere Commerce en el área de trabajo de VisualAge para Java. El depósito está disponible en el CD 2 de WebSphere Commerce Business Edition y el CD 2 de WebSphere Commerce Professional Edition. El nombre del depósito actual (en el momento de publicar este documento) es `WC_54.dat`.

Despliegue del código

Al personalizar su aplicación de comercio electrónico, es posible que tenga que efectuar alguna de las siguientes acciones:

- Crear mandatos, beans de datos o beans de entidad nuevos
- Ampliar los beans de entidad de WebSphere Commerce existentes
- Modificar la lógica de los mandatos o los beans de datos existentes

Al desarrollar código dentro de VisualAge para Java, puede probarlo dentro del Entorno de prueba WebSphere. En algún momento, debe desplegar el código en WebSphere Commerce Server fuera del entorno de desarrollo.

En las secciones siguientes, *WebSphere Commerce Server de destino* hace referencia al WebSphere Commerce Server en el que se está desplegando el código personalizado. En algunos escenarios de prueba, puede desplegarse a un WebSphere Commerce Server que está ejecutándose en la misma máquina que VisualAge para Java. En otras situaciones, el WebSphere Commerce Server de destino está en otra máquina, que incluso puede estar ejecutándose en una plataforma distinta.

Las secciones siguientes describen los pasos globales a seguir para desplegar los diversos tipos de código personalizado. Utilícelos para aprender los pasos de los que consta el proceso de despliegue y consulte el Apéndice B, "Información detallada sobre el despliegue" en la página 283 si desea obtener instrucciones paso a paso.

Además de las siguientes secciones que describen el despliegue de código personalizado, si ha creado nuevas políticas de control de acceso en su entorno de desarrollo, deben crearse las mismas políticas de control de acceso en el WebSphere Commerce Server de destino. Consulte "Carga de las políticas de control de acceso para los nuevos recursos" en la página 228 en las Guías de aprendizaje para ver un ejemplo de este procedimiento.

Información sobre el código desplegado de EJB

Es importante tener en cuenta que el componente Entorno de prueba WebSphere de VisualAge para Java V4.0 utiliza WebSphere Application Server V3.5.4. En esta versión de WebSphere Application Server, se da soporte a los beans enterprise al nivel de la especificación Enterprise JavaBeans (EJB) V1.0. Por este motivo, para ejecutar beans enterprise dentro del Entorno de prueba WebSphere, debe utilizar las herramientas de VisualAge para Java para generar código desplegado para los beans enterprise que satisfacen la especificación EJB V1.0.

En cambio, tanto WebSphere Commerce Business Edition como WebSphere Commerce Professional Edition utilizan WebSphere Application Server V4.0. WebSphere Application Server V4.0 soporta la especificación EJB V1.1. Por ello, el código desplegado para beans enterprise que se ejecutan dentro del Entorno de prueba WebSphere es distinto del desplegado para beans enterprise que se ejecutan dentro de WebSphere Application Server V4.0.

Las consecuencias en cuanto a desarrollo y despliegue son las siguientes:

- Para probar los beans enterprise dentro del Entorno de prueba WebSphere, utilice las herramientas de VisualAge para Java para generar el código desplegado que satisface los requisitos del WebSphere Application Server V3.5.4 que se utiliza en el Entorno de prueba WebSphere. Este código se genera

pulsando el botón derecho del ratón en el bean enterprise y seleccionando **Generar código desplegado**. Observe que esto no crea un archivo JAR.

- Para desplegar beans enterprise en un WebSphere Application Server V4.0, debe hacer lo siguiente:
  1. Utilice las herramientas de VisualAge para Java para exportar los beans enterprise en un archivo, que en este documento denominaremos *Archivo JAR de exportación de EJB 1.1*. Este archivo JAR empaqueta el código en un formato que la herramienta EJBDeploy utiliza. Este archivo se crea pulsando el botón derecho del ratón en el grupo EJB que contiene el bean enterprise a desplegar y seleccionando **Exportar > EJB 1.1 JAR**. Tenga presente que, al crear este archivo JAR, debe seleccionar el tipo de base de datos adecuado para la máquina en la que desplegará el código.
  2. Transfiera este archivo JAR de exportación de EJB 1.1 a un servidor de destino que ejecute WebSphere Application Server V4.0.
  3. En el servidor de destino, ejecute la herramienta EJBDeploy con el archivo JAR de exportación de EJB 1.1 como entrada para crear un nuevo archivo JAR de código desplegado que satisfaga la especificación Enterprise JavaBeans V1.1.

Hay otros pasos implicados en el despliegue de los beans enterprise. Para obtener más información, consulte "Despliegue de beans de entidad nuevos" en la página 161 y "Despliegue de beans de entidad públicos de WebSphere Commerce modificados" en la página 163. Para obtener más información sobre la utilización de la herramienta EJBDeploy, consulte "Generación de código desplegado" en la página 292.

Despliegue de mandatos y beans de datos nuevos

Al crear mandatos y beans de datos nuevos, debe ponerlos en un paquete designado de forma adecuada para la aplicación. Por ejemplo, podría crear un nuevo paquete con el nombre `com.mycompany.mycommands` en el que guardar los nuevos mandatos. Este paquete debe guardarse dentro de un proyecto separado de los proyectos de WebSphere Commerce (IBM WC Commerce Server e IBM WC Enterprise Beans). Por ejemplo, podría crear un nuevo proyecto denominado `My Project`. Para asegurarse de que el despliegue sea satisfactorio, es necesario organizar el código meticulosamente.

Con todos los mandatos y beans de datos nuevos guardados en su propio proyecto, el despliegue consta de los siguientes pasos generales:

1. Utilizando la máquina de desarrollo, cree un archivo JAR para el proyecto. En la página Proyecto de VisualAge para Java utilice las herramientas para exportar el proyecto a un archivo JAR. Denomine el archivo JAR de forma adecuada para el código como, por ejemplo, `CustomCommands.jar`. Además, tiene que volver a comprimir el archivo JAR utilizando herramientas externas a VisualAge para Java para asegurarse de que se incluya toda la información necesaria de denominación para el despliegue en WebSphere Application Server. Consulte "Archivos JAR para mandatos y beans de datos personalizados" en la página 283 si desea más información.
2. Copie el archivo JAR, las plantillas JSP y cualquier otro elemento de tienda en el directorio adecuado del WebSphere Commerce Server de destino. Consulte "Almacenamiento de elementos en el WebSphere Commerce Server de destino" en la página 288 para obtener más información.
3. Registre el nuevo mandato en el registro de mandatos en el WebSphere Commerce Server de destino. Consulte "Estructura del registro de mandatos" en la página 25 para obtener más información.

4. Detenga y reinicie la aplicación de empresa de WebSphere Commerce, utilizando la Consola de administración de WebSphere Application Server. Consulte la publicación *WebSphere Commerce, Guía de instalación* correspondiente, para obtener más información sobre cómo iniciar y detener esta aplicación.

Despliegue de beans de entidad nuevos

Al crear beans de entidad nuevos, debe crearlos en un grupo EJB distinto de los grupos EJB que contienen los beans de entidad de WebSphere Commerce. También debe utilizar su propio proyecto y asegurarse de que este proyecto no contenga código de ningún mandato ni bean de datos. Por ejemplo, podría crear un nuevo grupo EJB denominado `MyEntityBeans` y un proyecto con el nombre `MyEntityBeansProject`. Si el proyecto contiene código que no sea el código correspondiente a los beans de entidad nuevos, es posible que el despliegue no sea satisfactorio.

Una vez esté satisfecho con la forma de funcionar del bean de entidad dentro del Entorno de prueba WebSphere, debe desplegarlo. A continuación se proporciona una visión general de los pasos que deben seguirse para el despliegue:

1. Utilizando la máquina de desarrollo, cree un nuevo archivo JAR de exportación de EJB 1.1 para su nuevo grupo EJB. En la página EJB de VisualAge para Java, pulse con el botón derecho del ratón en el nuevo grupo EJB y seleccione exportar el código a un archivo JAR EJB 1.1. Denomine el archivo de forma adecuada para el código como, por ejemplo, `MyEntityBeans_DT.jar`. Consulte “Creación de archivos JAR para beans de entidad nuevos” en la página 284 si desea más información.
2. Cree un nuevo archivo JAR de implementación para el nuevo proyecto EJB. Para crear este archivo JAR, seleccione la página Proyecto, pulse con el botón derecho del ratón en el nuevo proyecto EJB y seleccione exportar el código a un archivo JAR. Denomine el archivo de forma adecuada para el código como, por ejemplo, `MyEntityBeansImpl.jar`. Además, tiene que volver a comprimir el archivo JAR de implementación utilizando herramientas externas a VisualAge para Java para asegurarse de que se incluya toda la información necesaria de denominación para el despliegue en WebSphere Application Server. Consulte “Creación de archivos JAR para beans de entidad nuevos” en la página 284 si desea más información.
3. Copie los archivos JAR en el directorio apropiado del WebSphere Commerce Server de destino. Consulte “Almacenamiento de elementos en el WebSphere Commerce Server de destino” en la página 288 para obtener más información.
4. En el WebSphere Commerce Server de destino, utilice la herramienta de despliegue EJB que se proporciona con WebSphere Application Server para generar el código desplegado para los nuevos beans enterprise. Esta herramienta toma el archivo JAR creado en el paso 1 como entrada y crea el archivo JAR correspondiente que contiene el código desplegado para todos los beans enterprise en su grupo EJB. Consulte “Generación de código desplegado” en la página 292 para obtener más información.
5. En el WebSphere Commerce Server de destino, modifique el nivel de aislamiento de transacción de los beans enterprise que están contenidos en el archivo JAR del código desplegado. Utilice el programa de utilidad de línea de mandatos `modifyIsolationLevel` que se proporciona con WebSphere Commerce para establecer el nivel de aislamiento de transacción en el nivel adecuado para el tipo de base de datos. Consulte “Modificación del nivel de aislamiento de transacción de los beans de entidad” en la página 294 si desea más información.
6. En el WebSphere Commerce Server de destino, cargue las políticas de control de acceso para cualquier nuevo recurso que haya creado. Utilice los mandatos

acpload y acpnlsload de WebSphere Commerce para cargar la información sobre política. En la sección "Carga de las políticas de control de acceso para los nuevos recursos" en la página 228 del Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" encontrará un ejemplo de cómo cargar políticas de control de acceso para nuevos recursos.

7. En el WebSphere Commerce Server de destino, exporte la aplicación de empresa de WebSphere Commerce actual de WebSphere Application Server. Cuando la exportación finaliza, se crea un archivo .ear que contiene toda la aplicación. Para exportar la aplicación actual, abra la Consola de administración de WebSphere Application Server, seleccione la aplicación de empresa de WebSphere Commerce y, a continuación, seleccione la opción de exportación. Al finalizar, se crea el archivo WC_Enterprise_App_*nombreInstancia* .ear. Consulte "Exportación de la aplicación de empresa actual de WebSphere Commerce" en la página 295 si desea más información.
8. En el WebSphere Commerce Server de destino, exporte la información de configuración para los beans enterprise que están contenidos en la aplicación de empresa de WebSphere Commerce actual que se ejecuta dentro de WebSphere Application Server. Esta información se exporta a un archivo XML utilizando la opción -export del programa de utilidad de línea de mandatos XMLConfig que se proporciona con WebSphere Application Server. El archivo XML generado (al que denominamos archivo OutputFile.xml) contiene una sección de información de configuración para cada bean enterprise que contiene la aplicación de empresa. A continuación, en este archivo debe añadir una nueva sección para cada nuevo bean enterprise que esté desplegando. Consulte "Exportación de la información de configuración para los beans enterprise" en la página 296 para obtener más información.
9. Añada el nuevo bean o beans enterprise en la aplicación de empresa utilizando la Herramienta de ensamblaje de aplicaciones que se proporciona con WebSphere Application Server. Con esta herramienta, puede abrir el archivo WC_Enterprise_App_*nombreInstancia*.ear para la aplicación actual y, a continuación, importar los nuevos beans enterprise a la aplicación. También debe establecer la vía de acceso de clases para el nuevo bean o beans enterprise de forma que incluya los archivos JAR dependientes y añadir la implementación del archivo JAR como un archivo en la aplicación. Finalmente, debe utilizar esta herramienta para configurar la seguridad de WebSphere Application Server para métodos que están contenidos en el bean o beans enterprise. Después de completar estos pasos, guarde la aplicación y se creará un nuevo archivo .ear para su aplicación de empresa. Consulte "Integración de nuevos beans enterprise en una aplicación de empresa" en la página 301 para obtener más información.
10. Importe la nueva aplicación de empresa a WebSphere Application Server. Este paso consta de las siguientes tres tareas:
    a. Mediante la Consola de administración de WebSphere Application Server, detenga y elimine la aplicación de empresa de WebSphere Commerce original. Consulte "Detención y supresión de una aplicación de empresa" en la página 308 para obtener más información.
    b. Utilice la opción -import del programa de utilidad de línea de mandatos XMLConfig para importar la nueva aplicación de empresa a WebSphere Application Server. Consulte "Importación de una aplicación de empresa" en la página 308 para obtener más información.
    c. Mediante la Consola de administración de WebSphere Application Server, renueve la vista para que muestre la nueva aplicación de empresa y, a continuación, inicie la nueva aplicación. Consulte "Inicio de una aplicación de empresa" en la página 310 para obtener más información.

Despliegue de las ampliaciones para mandatos y beans de datos existentes

El método para ampliar los mandatos existentes depende del tipo de modificación que se requiera. Estos métodos se describen en la sección "Personalización de mandatos existentes" en la página 118. En general, la modificación de la lógica existente implica crear una nueva clase que hereda las características de la clase que necesita personalización. Altere los métodos de la superclase según sea necesario, para sustituir o modificar la lógica.

Al personalizar beans de datos también se crea una nueva clase que amplía un bean de datos existente. Dentro de la nueva clase, haga las modificaciones necesarias.

Al crear estas nuevas clases, asegúrese de que se almacenan dentro de uno de sus propios paquetes, que a la vez esté almacenado dentro de uno de sus propios proyectos.

Puesto que las ampliaciones se gestionan eficazmente mediante la división en subclases, el despliegue de ampliaciones de mandatos y beans de datos es igual que el despliegue de mandatos y beans de datos nuevos. Para obtener más información, consulte "Despliegue de mandatos y beans de datos nuevos" en la página 160.

Despliegue de beans de entidad públicos de WebSphere Commerce modificados

Al modificar un bean enterprise público de WebSphere Commerce, se modifica el código de WebSphere Commerce. Por lo tanto, la técnica de despliegue para beans de entidad personalizados es ligeramente distinta a la utilizada para los beans de entidad nuevos. A continuación se proporciona una visión general de los pasos de despliegue:

1. Cree un archivo JAR de exportación de EJB 1.1 para el grupo EJB de WebSphere Commerce que contiene el bean de entidad modificado. Con la pestaña EJB del Entorno de trabajo de VisualAge para Java seleccionada, seleccione el grupo EJB adecuado y, a continuación, seleccione exportar este grupo a un archivo JAR de exportación de EJB 1.1. Para poner un nombre a este archivo JAR, puede utilizar el convenio de denominación `Cust_`*`nombreGrupoEJB`*`-ejb_DT.jar` donde *`nombreGrupoEJB`* es el nombre del grupo EJB que se ha modificado y el sufijo _DT se añade al final del nombre del archivo JAR. Por ejemplo, al poner un nombre al archivo JAR para el grupo EJB WCSUser, puede denominar el archivo `Cust_WCSUser-ejb_DT.jar`. Observe que el sufijo _DT se utiliza solamente como recordatorio de que, posteriormente, debe pasar este archivo JAR a la herramienta EJBDeploy para generar el código desplegado para los beans contenidos en el grupo EJB. Consulte "Creación de archivos JAR para beans de entidad de WebSphere Commerce personalizados" en la página 286 para obtener más información.
2. Cree un nuevo archivo JAR de cliente que contenga el código cliente para todos los grupos EJB de WebSphere Commerce. Con todos los grupos EJB de WebSphere Commerce seleccionados (todos los nombres empiezan por `WCS`), seleccione exportar a un archivo JAR de cliente. Consulte "Creación de archivos JAR para beans de entidad de WebSphere Commerce personalizados" en la página 286 para obtener más información.
3. Copie los archivos JAR en el directorio apropiado del WebSphere Commerce Server de destino. Consulte "Almacenamiento de elementos en el WebSphere Commerce Server de destino" en la página 288 para obtener más información.

4. En el WebSphere Commerce Server de destino, utilice la herramienta EJBDeploy que se proporciona con WebSphere Application Server para generar el código desplegado para los beans enterprise contenidos en el grupo EJB que está desplegando. Esta herramienta toma el archivo JAR creado en el paso 1 como entrada y crea el archivo JAR correspondiente que contiene el código desplegado para todos los beans enterprise en su grupo EJB. Consulte "Generación de código desplegado" en la página 292 si desea más información.
5. En el WebSphere Commerce Server de destino, modifique el nivel de aislamiento de transacción de los beans enterprise que están contenidos en el archivo JAR del código desplegado. Utilice el programa de utilidad de línea de mandatos `modifyIsolationLevel` que se proporciona con WebSphere Commerce para establecer el nivel de aislamiento de transacción en el nivel adecuado para el tipo de base de datos. Consulte "Modificación del nivel de aislamiento de transacción de los beans de entidad" en la página 294 si desea más información.
6. En el WebSphere Commerce Server de destino, exporte la aplicación de empresa de WebSphere Commerce actual de WebSphere Application Server. Cuando la exportación finaliza, se crea un archivo .ear que contiene toda la aplicación. Para exportar la aplicación actual, abra la Consola de administración de WebSphere Application Server, seleccione la aplicación de empresa de WebSphere Commerce y, a continuación, seleccione la opción de exportación. Al finalizar, se crea el archivo `WC_Enterprise_App_`*`nombreInstancia`*`.ear`. Consulte "Exportación de la aplicación de empresa actual de WebSphere Commerce" en la página 295 si desea más información.
7. En el WebSphere Commerce Server de destino, exporte la información de configuración para los beans enterprise que están contenidos en la aplicación de empresa de WebSphere Commerce actual que se ejecuta dentro de WebSphere Application Server. Esta información se exporta a un archivo XML utilizando la opción `-export` del programa de utilidad de línea de mandatos `XMLConfig` que se proporciona con WebSphere Application Server. El archivo XML generado (al que denominamos archivo `OutputFile.xml`) contiene una sección de información de configuración para cada bean enterprise que contiene la aplicación de empresa. Consulte "Exportación de la información de configuración para los beans enterprise" en la página 296 para obtener más información. Dentro de este archivo, debe modificar la sección que describe el bean enterprise que ha modificado, para que indique el nuevo archivo `Cust_`*`nombreGrupoEJB`*`-ejb.jar`.
8. En el WebSphere Commerce Server de destino, utilice la Herramienta de ensamblaje de aplicaciones para integrar el grupo EJB modificado en la aplicación de empresa. Con esta herramienta, abra el archivo `.ear` para la aplicación actual. Una vez abierto, efectúe las siguientes tareas:
   a. Tome nota de la información de vía de acceso de clases para la versión original del grupo EJB que ha modificado. Por ejemplo, si ha modificado el bean User en el grupo EJB WCSUser, debe copiar la información de vía de acceso de clases para el grupo WCSUser en un archivo de texto.
   b. Suprima la versión original del grupo EJB que ha modificado (por ejemplo, suprima el grupo WCSUser).
   c. Importe la nueva versión del grupo EJB que ha modificado.
   d. Aplique la información de vía de acceso de clases original al grupo EJB que acaba de importar.
   e. Configure la seguridad de WebSphere Application Server para los métodos contenidos en todos los beans enterprise del grupo EJB modificados.
   f. Guarde la aplicación en un archivo `.ear` nuevo.

Consulte “Integración de beans enterprise modificados en una aplicación de empresa” en la página 304 para obtener más información.

9. Importe la nueva aplicación de empresa a WebSphere Application Server. Este paso consta de las siguientes tres tareas:
   a. Mediante la Consola de administración de WebSphere Application Server, detenga y elimine la aplicación de empresa de WebSphere Commerce original. Consulte “Detención y supresión de una aplicación de empresa” en la página 308 para obtener más información.
   b. Utilice la opción `-import` del programa de utilidad de línea de mandatos `XMLConfig` para importar la nueva aplicación de empresa a WebSphere Application Server. Consulte “Importación de una aplicación de empresa” en la página 308 para obtener más información.
   c. Mediante la Consola de administración de WebSphere Application Server, renueve la vista para que muestre la nueva aplicación de empresa y, a continuación, inicie la nueva aplicación. Consulte “Inicio de una aplicación de empresa” en la página 310 para obtener más información.

Despliegue de beans de datos nuevos para utilizarlos en Commerce Studio

Si utiliza Commerce Studio para crear las plantillas JSP, debe desplegar los beans de datos nuevos en Commerce Studio. En concreto, debe crear un archivo JAR para los beans de datos.

Para crear este archivo JAR, pulse con el botón derecho del ratón en el paquete de beans de datos y seleccione exportarlos a un archivo JAR. En las opciones, seleccione incluir lo siguiente:

- **class**
- **resource**
- **beans**

Además, seleccione **Seleccionar tipos y recursos especificados** para incluir los mandatos y recursos que los beans de datos necesitan.

Después de exportar el archivo JAR, debe actualizar la vía de acceso de clases para Commerce Studio de forma que incluya este archivo JAR.

Cuando sea necesario modificar un bean de datos de WebSphere Commerce existente, debe crear un bean de datos nuevo que amplíe el que requiere la personalización. Por lo tanto, en realidad se está creando un bean de datos nuevo y el despliegue se lleva a cabo siguiendo los datos descritos en esta sección.

Despliegue de beans de entidad públicos personalizados para utilizarlos en Commerce Studio

Si modifica alguno de los beans de entidad públicos y utiliza Commerce Studio para el desarrollo de plantillas JSP, debe crear un nuevo archivo JAR de cliente para todos los beans de entidad públicos y modificar la vía de acceso de clases para Commerce Studio de modo que incluya el nombre del nuevo archivo JAR antes que el nombre del archivo JAR original. Commerce Studio utiliza el archivo JAR de cliente cuando se utilizan beans de datos en la herramienta Page Designer.

Para crear este archivo JAR, utilice las herramientas de VisualAge para Java. En la página EJB de VisualAge para Java, seleccione *todos* los grupos EJB de WebSphere

Commerce (no sólo los que ha modificado) y seleccione exportarlos a un archivo JAR de cliente. Una vez finalizada la exportación, asegúrese de especificar este nuevo archivo JAR al principio de la vía de acceso de clases para Commerce Studio.

Archivo de anotaciones cronológicas

En las distintas fases de instalación del producto, desarrollo del código y despliegue del código, se generan archivos de anotaciones. Esta sección lista algunos de los archivos de anotaciones que puede consultar, en todas estas fases.

Archivos de anotaciones de Commerce Studio
Commerce Studio crea archivos de anotaciones durante el proceso de instalación. Para acceder a estas anotaciones, abra el archivo siguiente:

```
C:\Winnt\WCStudioInstall.log
```

Anotaciones sobre la configuración de Commerce Studio para el Entorno de prueba WebSphere
Se efectúan varios pasos de configuración para el Entorno de prueba WebSphere durante la instalación de WebSphere Commerce Studio, si selecciona que desea realizar tareas de desarrollo de fondo. Por ejemplo, si selecciona incluir una tienda de ejemplo que se ejecute dentro del componente Entorno de prueba WebSphere de VisualAge para Java, se crean archivos de anotaciones relacionados con el proceso de carga masiva y de creación de base de datos. Para acceder a estas anotaciones, vaya al siguiente directorio:

```
unidad:\WebSphere\CommerceServerDev\instances\nombre_instancia\logs
```

Archivos de anotaciones de componentes de WebSphere Commerce en ejecución dentro del Entorno de prueba WebSphere
Al ejecutar una tienda o probar un componente individual dentro del Entorno de prueba WebSphere, pueden generarse un archivo de rastreo y un archivo de mensajes. La ubicación por omisión de estos archivos es el directorio siguiente:

```
unidad:\WebSphere\CommerceServerDev\instances\nombre_instancia\logs
```

Anotaciones al ejecutar el mandato modifyIsolationLevel
Al desplegar beans enterprise, se utiliza el mandato modifyIsolationLevel para modificar el nivel de aislamiento de transacción de cada bean enterprise en el archivo JAR. Los archivos de anotaciones generados se almacenan en el archivo de anotaciones especificado al ejecutar el mandato. Consulte “Modificación del nivel de aislamiento de transacción de los beans de entidad” en la página 294 para obtener más información.

Anotaciones cronológicas en VisualAge para Java
Al ejecutar código dentro del Entorno de prueba WebSphere, la ventana de la Consola se ejecuta como una anotación activa. Además, puede modificar el nivel de rastreo de un servidor EJB para aumentar la cantidad de información que se puede encontrar en la ventana de la Consola. Esta información se especifica como nivel de rastreo, dentro de las propiedades del servidor EJB.

Método de pago de prueba

La tienda de ejemplo InFashion que se ejecuta dentro del Entorno de prueba WebSphere utiliza, por omisión, un método de pago de prueba. Se ha incluido este método de pago de prueba para que pueda completar el flujo de compra dentro del Entorno de prueba WebSphere, sin que sea necesario llamar a un Payment

Manager remoto. Los pedidos sometidos que han utilizado este método de pago tienen el estado 'M' (que indica que el pago se ha iniciado y está esperando su proceso).

Este método de pago de prueba sólo le permite completar una compra, no permite que los pedidos sometidos con este método estén disponibles para procesos adicionales. Por esta razón, el método de pago de prueba sólo debe utilizarse dentro del Entorno de prueba WebSphere.

Las clases de implementación para los mandatos relacionados con el pago son tipos de mandatos de política de negocio. Por eso, la selección de la clase de implementación a utilizar la lleva a cabo una política de negocio. Por omisión, la tienda de ejemplo InFashion que se ejecuta dentro del Entorno de prueba WebSphere incluye una política de negocio (política TestPaymentMethod) que utiliza las siguientes implementaciones de mandatos relacionados con el pago:

- `com.ibm.commerce.payment.commands.DoPayment`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.CheckPaymentAccept`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.DoCancel`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.DoDeposit`**`Test`**`CmdImpl`
- `com.ibm.commerce.payment.commands.DoRefund`**`Test`**`CmdImpl`

Debe subrayarse que estos mandatos se proporcionan solamente para probar el proceso de pasar por caja. Mientras que el conjunto anterior de mandatos se proporciona para completar el proceso, DoDepositTestCmdImpl es un "apéndice" de mandato que emite una excepción si se le llama. No intente utilizar otras funciones de gestión de pedidos con estos pedidos. Si necesita poder probar estas otras funciones, puede configurar su Entorno de prueba WebSphere para utilizar un Payment Manager remoto.

Es muy importante que cuando despliegue el código en el WebSphere Commerce Server de destino, no copie la política de negocio relacionada con el método de pago de prueba, de la máquina de desarrollo a la máquina de destino. Además, no debe registrar los mandatos relacionados con el método de pago de prueba en el WebSphere Commerce Server de destino.

Consulte el capítulo "Configuración de VisualAge para Java" de la publicación *WebSphere Commerce Business Edition, Guía de instalación* o *WebSphere Commerce Professional Edition, Guía de instalación* para obtener más información sobre la inhabilitación del método de pago de prueba.

Utilización de un Payment Manager remoto

Si su trabajo de desarrollo incluye trabajar con pedidos una vez se han formalizado, por ejemplo, modificando un paso del proceso de gestión de pedidos, debe configurar la tienda que se ejecuta en el Entorno de prueba WebSphere para que utilice un Payment Manager remoto. Consulte la publicación *WebSphere Commerce Studio, Business Developer Edition, Guía de instalación* para obtener información detallada sobre los pasos de configuración. Este documento también proporciona información sobre la eliminación de pedidos formalizados con el método de pago de prueba desde su base de datos.

Parte 4. Guías de aprendizaje

Esta sección contiene guías de aprendizaje que le ayudarán a familiarizarse con la personalización de su aplicación de comercio electrónico. Se proporcionan las siguientes guías de aprendizaje:

- Creación de nueva lógica de negocio
  Esta guía de aprendizaje muestra el proceso de desarrollo para crear nueva lógica de negocio. Incluye las siguientes subtareas:
  - Preparar el proyecto de ejemplo
  - Escribir nuevos mandatos
  - Crear un bean de datos nuevo y obtener las propiedades de la petición
  - Utilizar beans de entidad
  - Crear un bean enterprise nuevo
- Actualización de la lógica de negocio existente
  Esta guía de aprendizaje muestra el proceso de desarrollo para actualizar la lógica de negocio de WebSphere Commerce existente. Incluye las siguientes subtareas:
  - Ampliar un mandato de controlador de WebSphere Commerce existente
  - Modificar un bean de entidad público de WebSphere Commerce existente
  - Crear un nuevo mandato de tarea que amplíe un mandato de tarea de WebSphere Commerce existente.

Las guías de aprendizaje contienen secciones de código que deben entrarse en VisualAge para Java. Se le recomienda que obtenga una versión en PDF de este documento, de las siguientes direcciones Web:

Business

```
http://www.ibm.com/software/webservers/commerce/wc_be
   /lit-tech-general.html
```

Professional

```
http://www.ibm.com/software/webservers/commerce/wc_pe
   /lit-tech-general.html
```

Puede cortar y pegar las secciones de código de la versión en PDF de la Guía del programador.

Capítulo 9. Guía de aprendizaje: Creación de nueva lógica de negocio

Entorno para la guía de aprendizaje

Las guías de aprendizaje que se incluyen en este manual requieren que tenga instalado WebSphere Commerce Business Edition V5.4 o WebSphere Commerce Professional Edition V5.4. Además, cuando instale el producto, debe seleccionar la instalación de las opciones siguientes:

- **Desarrollar elementos del escaparate de la tienda utilizando WebSphere Studio**
- **Desarrollar lógica interna de la tienda utilizando VisualAge para Java**
- **Crear base de datos**
- **Incluir tienda de ejemplo**

Consulte la publicación *WebSphere Commerce Studio, Business Developer Edition, Guía de instalación* o *WebSphere Commerce Studio Professional Developer Edition, Guía de instalación* para obtener una completa información sobre la instalación y configuración.

Antes de iniciar las guías de aprendizaje, debe poder ejecutar la tienda de ejemplo dentro del Entorno de prueba WebSphere y completar una compra en la tienda.

Pasos para el despliegue de código

Las guías de aprendizaje que se incluyen en este manual contienen pasos opcionales que describen cómo desplegar el código personalizado en un WebSphere Commerce Server de destino. Se da por supuesto que el WebSphere Commerce Server de destino se ejecuta en Windows NT o en Windows 2000, en una máquina aparte del entorno de desarrollo.Tenga en cuenta que si utiliza WebSphere Commerce Studio Business Developer Edition como entorno de desarrollo, debe desplegar en WebSphere Commerce Business Edition. Si utiliza WebSphere Commerce Studio Professional Developer Edition como entorno de desarrollo, debe desplegar en WebSphere Commerce Professional Edition.

Además, en el WebSphere Commerce Server de destino debe tener publicada una tienda basada en la tienda de ejemplo InFashion. Dentro de esa tienda, debe poder completar una compra. Puede utilizar un Payment Manager local o remoto para el proceso de pagos.

Si está desplegando en un WebSphere Commerce Server de destino que está en la misma máquina que su entorno de desarrollo o que está ejecutando un sistema operativo distinto, consulte el Capítulo 8, “Herramientas de desarrollo y despliegue” en la página 157 y el Apéndice B, “Información detallada sobre el despliegue” en la página 283 para obtener la información de despliegue correspondiente.

Preparación del proyecto de ejemplo

En esta sección, importará el depósito `WC_SAMPLE_54.dat` que contiene el punto de inicio para esta guía de aprendizaje, “Creación de nueva lógica de negocio”.

Para preparar el entorno, haga lo siguiente:

1. Asegúrese de utilizar el depósito `WC_54.dat` de código de WebSphere Commerce. El depósito está disponible en el CD 2 de WebSphere Commerce Business Edition V5.4 o el CD 2 de WebSphere Commerce Professional Edition V5.4.
2. Importe el proyecto de ejemplo a su área de trabajo, realizando las acciones siguientes:
   a. Inserte el siguiente CD en la unidad de CD de la máquina de desarrollo:
      - Business CD 2 de WebSphere Commerce Business Edition, V5.4
      - Professional CD 2 de WebSphere Commerce Professional Edition, V5.4

   b. Abra VisualAge para Java.
   c. En el menú **Archivo**, seleccione **Importar**
   Se abre el SmartGuide Importar.
   d. Seleccione un **Depósito** para importar y pulse **Siguiente**.
   e. En la ventana Importar desde otro depósito, haga lo siguiente:
      1) Seleccione **Depósito local**.
      2) En el campo **Nombre de depósito**, entre *unidad_CD*`:\repository\samples\programguide\WC_SAMPLE_54.dat` donde *unidad_CD* es la unidad de CD.
      3) Seleccione **Proyectos** y pulse **Detalles**. Seleccione el proyecto **_WCSamples**. Seleccione la versión **WC Sample 5.4** y pulse **Aceptar**.
      4) Compruebe que **Añadir la edición más reciente de proyecto al área de trabajo** esté seleccionado.
      5) Pulse **Terminar** para iniciar la importación.
      La importación del proyecto puede tardar varios minutos.
3. Asegúrese de que el propietario del área de trabajo esté establecido en WCS Developer, haciendo lo siguiente:
   a. En el menú **Área de trabajo**, seleccione **Cambiar propietario del área de trabajo**.
   b. Seleccione **WCS Developer** y pulse **Aceptar**.
4. Copie las plantillas JSP para los ejercicios en el directorio adecuado, para que puedan utilizarse en el Entorno de prueba WebSphere. Para copiar estos archivos, haga lo siguiente:
   a. Localice los archivos `Sample.jsp` y `Sample_All.jsp` en el siguiente directorio
   unidad_CD`:\repository\samples\programguide\`

   donde *unidad_CD* es la unidad de CD en el siguiente directorio:
   b. Copie estos dos archivos en el siguiente directorio:
   unidad_vaj`:\VAJava\ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web`

   donde *unidad_vaj* es la unidad en la que ha instalado VisualAge para Java.

5. Copie los archivos de política de control de acceso en el directorio adecuado. Estos archivos se utilizan para cargar nuevas políticas de control de acceso para los nuevos recursos que se crean en las guías de aprendizaje. Para copiar estos archivos, haga lo siguiente:
   a. Vaya al siguiente directorio:
      unidad_CD:\repository\samples\programguide\
   b. Localice los siguientes archivos en ese directorio:
      - SampleCmdACPolicy.xml
        Este archivo XML contiene la política de control de acceso que se utiliza al crear un nuevo mandato de controlador.
      - SampleACPolicy.xml
        Este archivo XML contiene la política de control de acceso que se utiliza al crear un nuevo bean enterprise.
      - SampleACPolicy_*entorno_nacional*.xml
        donde *entorno_nacional* es el identificador de idioma. Este archivo XML contiene la descripción de la política de control de acceso.
   c. Copie los archivos anteriores en el siguiente directorio:
      unidad:\WebSphere\CommerceServerDev\xml\policies\xml
      donde *unidad* es la unidad en la que ha instalado WebSphere Commerce Studio.
6. Pruebe el entorno para asegurarse de que está listo para iniciar las guías de aprendizaje, de la manera siguiente:
   a. Inicie el servidor de nombres persistentes tal como se describe en la página 281.
   b. Inicie el servidor EJB tal como se describe en la página 281.
   c. Inicie el motor de servlets tal como se describe en la página 282
   d. Abra un navegador y entre el siguiente URL:
      ```
      http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
         storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
      ```
      donde *ID_tienda* es el identificador de su tienda de ejemplo (10001 es un valor de ejemplo) e *ID_catálogo* es el identificador del catálogo de su tienda de ejemplo (10001 es un valor de ejemplo).
      Cuando se visualice la página de presentación de la tienda de ejemplo, seleccione un producto y cómprelo. Debe poder alcanzar la página "Confirmación del pedido" en el flujo de compra.

Para comprobar el valor de storeId para su tienda, puede consultar la tabla STOREENT.

   e. Cierre todos los navegadores y detenga el Entorno de prueba WebSphere.

Ahora ya está preparado para empezar las guías de aprendizaje.

Creación de mandatos

Creación de un mandato de controlador

Esta sección muestra cómo escribir un mandato de controlador nuevo. Cuando complete este ejercicio, dispondrá de un nuevo mandato llamado *MyNewControllerCmd*. Este mandato lo utilizan todas las tiendas y cada tienda utiliza la misma implementación del mandato.

Hay tres pasos básicos al crear un nuevo mandato de controlador:

1. Registrar el mandato en la infraestructura del registro de mandatos.
2. Crear la interfaz para el mandato.
3. Crear la clase de implementación para el mandato.

Para obtener más información sobre mandatos, consulte “Patrón de diseño de mandatos” en la página 20.

Antes de iniciar esta guía de aprendizaje

Ha de haber realizado los pasos del apartado “Preparación del proyecto de ejemplo” en la página 172.

Para escribir el nuevo mandato, haga lo siguiente:

1. El primer paso al escribir un mandato de controlador es establecer un nombre para el mandato. En este ejemplo, el mandato se llama *MyNewControllerCmd*.
2. El mandato debe registrarse en la infraestructura del registro de mandatos. El proceso de registro para el nuevo mandato de controlador implica la creación de una entrada en la tabla URLREG.

   DB2 Si utiliza una base de datos DB2, haga lo siguiente para registrar el mandato:

   a. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Centro de mandatos**).
   b. En el menú **Herramientas**, seleccione **Valores de herramientas**.
   c. Seleccione el recuadro **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (**;**)
   d. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla URLREG indicando la siguiente información en la ventana de script:

      ```
      connect to nombre_base_de_datos;
      insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
          DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
          'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
          'Este es un nuevo mandato de controlador con fines de prueba.',
          null)
      ```

      donde *nombre_base_de_datos* es el nombre de su base de datos, y pulse el icono Ejecutar.
      Este mandato lo utilizan todos los comerciantes (indicado por el valor 0 para STOREENT_ID).

   Oracle Si utiliza una base de datos Oracle, haga lo siguiente para registrar el mandato:

   a. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
   b. En el campo **User Name**, entre su nombre de usuario de Oracle.
   c. En el campo **Password**, entre su contraseña de Oracle.
   d. En el campo **Host String**, entre su serie de conexión.
   e. En la ventana de SQL Plus, entre lo siguiente para crear la entrada requerida en la tabla URLREG:

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
    DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
    'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
    'Este es un nuevo mandato de controlador con fines de prueba.',
    null);
```

y pulse Intro para ejecutar la sentencia de SQL.

f. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

y pulse Intro para ejecutar la sentencia de SQL.

Nota: Para simplificar este ejercicio, el nuevo mandato sólo tiene una clase de implementación y, en consecuencia, todas las tiendas utilizan la misma clase de implementación. Esta clase de implementación se especifica directamente en el código correspondiente a la interfaz. Así pues, no es necesario registrar la correlación entre la interfaz y la clase de implementación en la tabla CMDREG. Fuera de un entorno de guía de aprendizaje, debería registrar el mandato de controlador en la tabla CMDREG así como en la tabla URLREG.

3. Los mandatos de controlador deben devolver una vista. El nuevo mandato de controlador que va a crear devolverá la vista SampleViewTask. Debe registrar la vista SampleViewTask en la tabla VIEWREG.

DB2 Si utiliza una base de datos DB2, haga lo siguiente para registrar la vista:

a. Cree una entrada en la tabla VIEWREG, entrando lo siguiente en la ventana de script (observe que primero quizá tenga que borrar la sentencia SQL anterior de la ventana):

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('SampleViewTask',-1, 0,
    'com.ibm.commerce.command.ForwardViewCommand',
    'com.ibm.commerce.command.HttpForwardViewCommandImpl',
    'docname=Sample.jsp','Es una vista de ejemplo para el
    ejercicio de Bonus Point', 0, null)
```

y pulse el icono Ejecutar.

Oracle Si utiliza una base de datos Oracle, haga lo siguiente para registrar su vista en la base de datos:

a. En la ventana de SQL Plus, entre la siguiente sentencia de SQL:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('SampleViewTask',-1, 0,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=Sample.jsp','Es una vista de ejemplo para el
   ejercicio de Bonus Point', 0, null)
```

y pulse Intro para ejecutar la sentencia de SQL.

b. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

y pulse Intro para ejecutar la sentencia de SQL.

4. En la ventana Entorno de trabajo de VisualAge para Java, expanda el proyecto **_WCSamples**.

5. Expanda el paquete **com.ibm.commerce.sample.commands**, pulse con el botón derecho del ratón en la interfaz **MyNewControllerCmd** y seleccione **Añadir > Campo**.
   Se abrirá el SmartGuide Crear campo.
6. Cree un campo que especifique la clase de implementación por omisión para la interfaz, de la siguiente manera:
   a. En el campo **Nombre del campo**, especifique `defaultCommandClassName`.
   b. En la lista desplegable **Tipo de campo**, seleccione `String`.
   c. En el campo Valor inicial, entre `"com.ibm.commerce.sample.commands.MyNewControllerCmdImpl"`. (Asegúrese de incluir los signos de comillas dobles.)
   d. Pulse **Finalizar**.
      Se genera el código para el nuevo campo.

En cualquier momento de esta guía de aprendizaje, cuando modifica un campo o método que ya existe en la superclase, puede visualizarse un aviso indicando que el nuevo campo o método ocultará un campo o método heredado. Si es así, pulse **Sí** para continuar.

7. Expanda la clase **MyNewControllerCmdImpl** y seleccione el método **performExecute** para ver el código fuente.
8. En el código fuente para el método performExecute, descomente la Sección 1 y la Sección 5. La Sección 1 incluye el siguiente código en el método:

```
// Create a new TypedProperties for output.
  TypedProperty rspProp = new TypedProperty();
```

   La Sección 5 incluye el siguiente código en el método:

```
// see how controller command call a JSP

 rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
 setResponseProperties(rspProp);
```

   Guarde el trabajo (**Control + S**). La sección de código anterior establece el nombre de vista que debe devolver el mandato de controlador.
9. Debe especificarse control de acceso a nivel de mandato para este nuevo mandato de controlador. En este caso, la política de control de acceso a nivel de mandato especificará que todos los usuarios pueden ejecutar el mandato. Esta política se especifica en el archivo `SampleCmdACPolicy.xml`. Para cargar la nueva política de control de acceso, haga lo siguiente:
   a. En un indicador de mandatos, vaya al siguiente directorio:
      unidad`:\WebSphere\CommerceServerDev\bin`
   b. Debe emitir el mandato `acpload`, que tiene el siguiente formato:
      `acpload` *`nombre_bd usuario_bd contraseña_bd archXMLentrada`*

      donde
      - *`nombre_bd`* es el nombre de su base de datos
      - *`usuario_bd`* es el nombre de usuario de la base de datos
      - *`contraseña_bd`* es la contraseña de la base de datos
      - *`archXMLentrada`* es el nombre del archivo XML que contiene la política.

      Por ejemplo, puede emitir el mandato siguiente:

```
acpload VAJ_Demo user password SampleCmdACPolicy.xml
```

10. Añada el nuevo proyecto _WCSamples a la vía de acceso de clases para el motor de servlets llevando a cabo lo siguiente:
    a. En el menú **Área de trabajo**, seleccione **Herramientas > Entorno de prueba WebSphere**.
       Se abre el Centro de control del Entorno de prueba WebSphere.
    b. Pulse **Motor de servlets** y, a continuación, **Editar Classpath**.
       En la ventana del Classpath del motor de servlets, pulse **Seleccionar todo** y, a continuación, **Aceptar**.
11. Pruebe el nuevo mandato haciendo lo siguiente:
    a. Inicie el servidor de nombres persistentes tal como se describe en la página 281.
    b. Inicie el servidor EJB tal como se describe en la página 281.
    c. Inicie el motor de servlets tal como se describe en la página 282
    d. Abra un navegador y entre el siguiente URL:
       ```
       http://localhost:8080/webapp/wcs/stores/servlet/
          MyNewControllerCmd
       ```
       Transcurridos unos minutos el navegador visualizará una página que mostrará "Sample JSP", tal como aparece a continuación:

Figura 31.

12. Cree una versión de su código en su estado actual. Esto le permitirá restaurar su código en este estado en cualquier momento. Para crear la versión, haga lo siguiente:
    a. Seleccione los paquetes **com.ibm.commerce.sample.commands** y **com.ibm.commerce.sample.databeans** (mantenga pulsada la tecla Control mientras selecciona los diversos paquetes), pulse con el botón derecho y seleccione **Gestionar > Crear edición abierta**.
    b. Pulse con el botón derecho del ratón en el proyecto **_WCSamples** y seleccione **Gestionar > Crear edición abierta**.
    c. Pulse con el botón derecho del ratón en el proyecto **_WCSamples** de nuevo y seleccione **Gestionar > Versión**.
       Se abre la ventana Creación de versiones de los elementos seleccionados.
    d. Seleccione el botón **Un nombre**, entre mySample 1.2 Completed y pulse **Aceptar**.

Ya ha añadido un nuevo mandato y lo ha probado (brevemente) en el entorno de prueba integrado.

Modificación de MyNewControllerCmd

En la sección anterior creó MyNewControllerCmd. En este ejercicio, examinará el contenido de su nuevo mandato con más detalle para obtener una idea más clara de cómo escribir un mandato de controlador propio personalizado.

Empezaremos por examinar la estructura del código que ha creado. La interfaz MyNewControllerCmd amplía la interfaz ControllerCommand. También define la clase de implementación a utilizar como valor por omisión. Esta clase se utiliza cuando el mandato no está registrado en la tabla CMDREG, o cuando no hay una clase de implementación especificada en dicha tabla.

También ha creado la clase MyNewControllerCmdImpl. La clase de implementación es la clase que contendrá finalmente la lógica de negocio (o que llamará a mandatos de tarea para que realicen tareas de negocio individuales) que desea implementar. Su clase de implementación contiene los métodos siguientes:

Métodos de MyNewControllerCmdImpl	**Descripción**
MyNewControllerCmdImpl()	El método constructor.
validateParameters()	Se utiliza para la validación en el lado del servidor de los parámetros de entrada del mandato.
isGeneric()	Determina si un usuario genérico puede llamar o no al mandato.
isRetriable()	Determina si se reintentará o no el mandato después de retrotraer la base de datos.
performExecute()	Contiene la lógica de negocio para el mandato.

Las secciones siguientes muestran más detalles sobre cómo actualizar su nuevo mandato de controlador.

Pasar variables a la plantilla JSP

En esta sección, modifique MyNewControllerCmd de modo que pase las variables a la plantilla JSP. Para visualizar las variables, debe utilizar un bean de datos nuevo llamado DataBeanSampleBean. Para habilitar la visualización de variables en la plantilla JSP, haga lo siguiente:

1. En la ventana Entorno de trabajo de VisualAge para Java, expanda el proyecto **_WCSamples**.
2. Expanda el paquete **com.ibm.commerce.sample.commands**, después la clase **MyNewControllerCmdImpl** y seleccione su método **performExecute**.
3. En el código fuente para el método performExecute, descomente la Sección 2. Esto inserta el código siguiente en el método:

```
// see how controller command pass in variables to JSP

 // Add additional parameters in controller command
 // to rspProp for response
 //
 rspProp.put("ControllerParm1", "Hello world");
 rspProp.put("ControllerParm2", "Have a nice day!");
```

   La sección de código anterior crea dos nuevos parámetros que se colocan en las propiedades que se pasan a la plantilla JSP. Guarde el trabajo (**Control + S**).

4. El bean de datos DataBeanSampleBean lo utiliza la plantilla JSP para visualizar las variables. El bean se ha creado automáticamente, pero es necesario modificar el código fuente tal como se indica a continuación:
   a. Expanda el paquete **com.ibm.commerce.sample.databeans**.
   b. Expanda la clase **DataBeanSampleBean** y seleccione el método **setRequestProperties** para ver su código fuente.
   c. En el código fuente del método setRequestProperties, descomente la Sección 1. Esto incluye el siguiente código en el método:

```
// copy input TypedProperties to local

 requestProperties = aParam;
```

   Guarde el trabajo. La sección de código anterior copia los valores de aParam (que se define en la superclase) localmente. Se utiliza para obtener las propiedades del objeto petición.
5. Actualice el archivo Sample.jsp de modo que utilice el nuevo bean de datos y visualice las variables:
   a. Utilice un editor de texto para abrir los archivos `Sample.jsp` y `Sample_All.jsp`. Estos archivos se encuentran en el directorio siguiente: *unidad_vaj*`:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host \default_app\web`
   b. Copie la Sección 1 del código de Sample_All.jsp en Sample.jsp entre los marcadores `<!-- SECTION 1 -->` y `<!-- END OF SECTION 1 -->`. Esto inserta el código siguiente en la plantilla JSP:

```
<!-- SECTION 1 -->
<%
DataBeanSampleBean testBean = new DataBeanSampleBean ();
com.ibm.commerce.beans.DataBeanManager.activate (testBean, request);
%>
<!-- END OF SECTION 1 -->
```

   Esta sección de código crea una instancia del bean de datos.
   c. Copie la Sección 2 de Sample_All.jsp en Sample.jsp entre los marcadores `<!-- SECTION 2 -->` y `<!-- END OF SECTION 2 -->`. Esto incluye el siguiente código en la plantilla JSP.

```
<!-- SECTION 2 -->
<%
TypedProperty prop = testBean.getRequestProperties();

 out.print("<B>List of name value pairs in TypedProperties
           object</B><P>");

 // convert from request Properties to query string
 for (Enumeration pns = prop.keys(); pns.hasMoreElements();) {
     String paramName = (String) pns.nextElement();
      // do not add the url parameter to the query string
      Object val = prop.get(paramName,null);
     if (val != null) {
          if (val.getClass().isArray()) {
           // flatten the array
           String[] oarray = (String[]) val;
          int len = java.lang.reflect.Array.getLength(val);
          for (int i = 0; i < len; i++) {
          out.print(paramName + "[" + i + "]=" + oarray[i] + "<br>");
          }
     } else {
     // assume that it is a String
     out.print(paramName + "=" + val.toString() + "<br>");
     }
```

```
    }
  }
%>
<P>
<!-- END OF SECTION 2 -->
```

Guarde este archivo.
Esta sección de código utiliza el método getRequestProperties del objeto de bean de datos testBean. Va pasando por las propiedades y las muestra en el navegador.

6. Pruebe las modificaciones para verificar que las variables se muestran en la plantilla JSP efectuando las siguientes acciones:
   a. Asegúrese de que el Entorno de prueba WebSphere esté en ejecución.
   b. En un navegador, entre el siguiente URL:
      `http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd`

      El archivo JSP de ejemplo se visualiza y muestra las propiedades del mandato de controlador. La salida aparece de la siguiente manera:

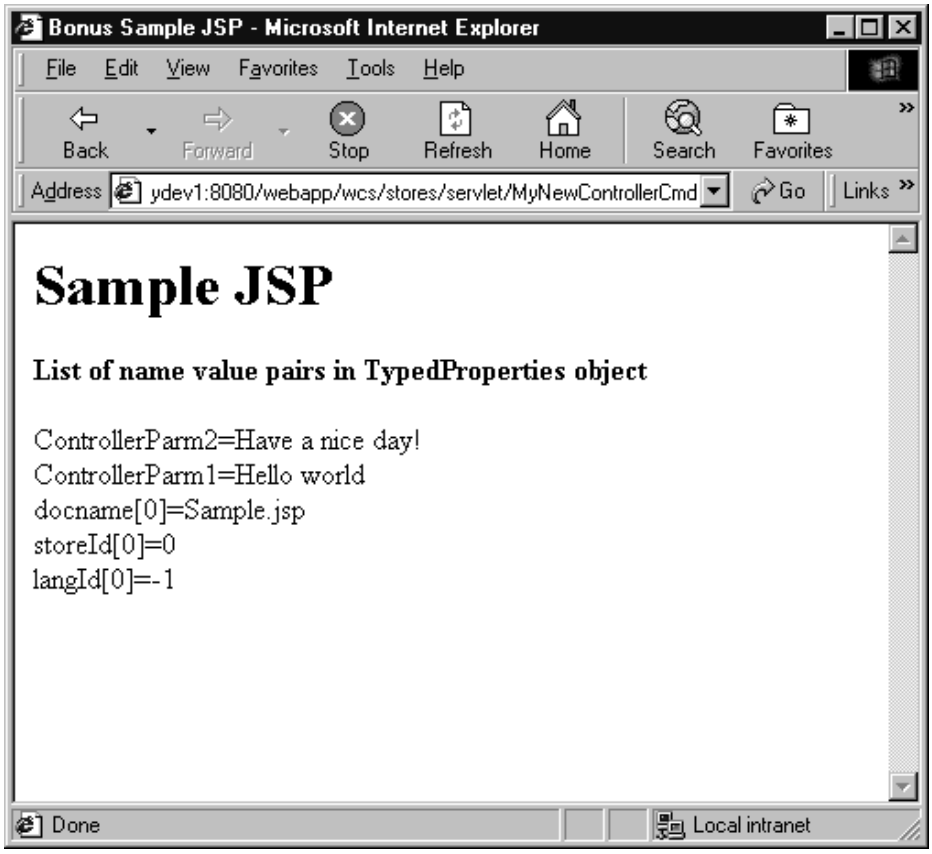

Figura 32.

7. Cree una versión de su código en su estado actual. Denomine esta versión `mySample 1.3 Completed`. Si desea más detalles sobre cómo crear una versión del código, consulte el paso 12 en la página 177.

Modificación del método validateParameters

El método validateParameters que hay actualmente en su nuevo mandato es simplemente un ″apéndice″ del mandato. Consta del siguiente código:

```
public void validateParameters() throws ECException {
     }
```

En esta sección añadirá su propia comprobación de parámetros personalizada al mandato y después pasará los parámetros a la plantilla JSP.

Al modificar el método validateParameters, se añaden nuevos campos a la clase y se utilizan para los parámetros.

Creación de nuevos campos: Cuando se añaden nuevos campos a la interfaz o la clase existentes, puede utilizar el SmartGuide Crear campos en VisualAge para Java. Esta sección describe los pasos genéricos para crear un nuevo campos. Utilice esta información en las secciones siguientes de la guía de aprendizaje, cuando tenga que añadir nuevos campos a clases e interfaces.

Para crear un nuevo campo, haga lo siguiente:

1. Pulse con el botón derecho del ratón en la interfaz o clase a la que desea añadir el nuevo campo; seleccione **Añadir > Campo**.
   Se abre el SmartGuide Crear campo.
2. En el campo **Nombre del campo**, especifique el nombre del nuevo campo.
3. Para especificar el tipo de campo, efectúe una de las siguientes acciones:
   - En la lista desplegable **Tipo de campo**, seleccione el tipo de campo.
   - Si el tipo de campo necesario no está especificado en la lista, pulse **Examinar**. En el campo **Patrón**, entre el nombre (o nombre parcial) del tipo de campo y pulse **Aceptar**.

   Nota: En las guías de aprendizaje, cuando se especifica el tipo de campo `String`, esto proviene del paquete `java.lang`.
4. En el campo **Valor inicial**, entre el valor inicial del campo. Para los valores iniciales que sean series de caracteres, asegúrese de encerrar los valores entre comillas (“ ”).
5. Para el valor **Modificadores de acceso**, seleccione el botón de selección adecuado (public, protected, none o private), si es necesario.
6. Marque el recuadro de selección **Acceso con métodos get y set** si desea generar métodos get y set para el campo. Si lo selecciona, también deberá especificar las propiedades para los métodos get y set, de la siguiente manera:
   - Para el método get, seleccione uno de los botones de selección, public, protected, private o none.
   - Para el método set, seleccione uno de los botones de selección, public, protected, private o none.
7. Pulse **Terminar**.

Para modificar el método validateParameters, haga lo siguiente:

1. Debe crear dos nuevos campos en la clase MyNewControllerCommandImpl. El primero se utiliza para las series de entrada y el segundo para enteros de entrada. Para crear estos campos, haga lo siguiente:
   a. Expanda el paquete **com.ibm.commerce.sample.commands**.
   b. Seleccione la clase **MyNewControllerCmdImpl**.
   c. Añada un campo a la clase, utilizando los valores siguientes. Para obtener pasos detallados sobre cómo crear un nuevo campo, consulte “Creación de nuevos campos”.

Nombre de atributo	Valor
Nombre de campo	`inputString`
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

d. Cree otro campo para los enteros de entrada, utilizando estos valores:

Nombre de atributo	Valor
Nombre de campo	`inputInteger`
Tipo de campo	`Integer` **Nota:** Pulse **Examinar** y entre `Integer`. No seleccione `int`.
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

2. Seleccione el método **performExecute** de la clase MyNewControllerCmdImpl.
3. En el código fuente del método performExecute, descomente la Sección 3 para insertar el siguiente código en el método:

```
// see how controller command pass in input variables to JSP

     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());
```

   Este código pasa variables del mandato de controlador al bean de datos. Guarde el trabajo.
4. Seleccione el método **validateParameters** de la clase MyNewControllerCmdImpl.
5. Descomente la Sección 1 del código fuente validateParameters, para insertar el siguiente código en el método:

```
// uncomment to check parameters

      TypedProperty prop = getRequestProperties();

// retrieve required parameters
 //
 try {
          setInputString(prop.getString("input1"));
 } catch (ParameterNotFoundException e) {
          throw new ECApplicationException(
               ECMessage._ERR_CMD_MISSING_PARAM,
               "MyControllerCmdImpl","validateParameters",
               ECMessageHelper.generateMsgParms(e.getParamName()));
 }
```

```
// retrieve optional Integer
// set input2 = 0 if no input value  //
setInputInteger(prop.getInteger("input2", 0));
```

Guarde su trabajo.
La sección de código anterior comprueba los dos parámetros de entrada. El bloque indicado por try determina si el primer parámetro está ahí y, si no está, se genera una excepción. Puesto que el segundo parámetro es opcional, este código establecerá el parámetro en el valor `0` si dicho parámetro falta o es de un tipo incorrecto.

6. Pruebe el mandato de la siguiente manera:
   a. Asegúrese de que el servidor de nombres persistentes, el servidor EJB y el motor de servlets estén ejecutándose.
   b. En su navegador, entre los siguientes URL:
      - Caso 1: Falta el parámetro. Entre:

        `http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd`

        Puesto que no se pasa ningún parámetro al mandato, se muestra un error de aplicación genérico para indicar que falta un parámetro. El resultado se muestra en la siguiente ilustración:

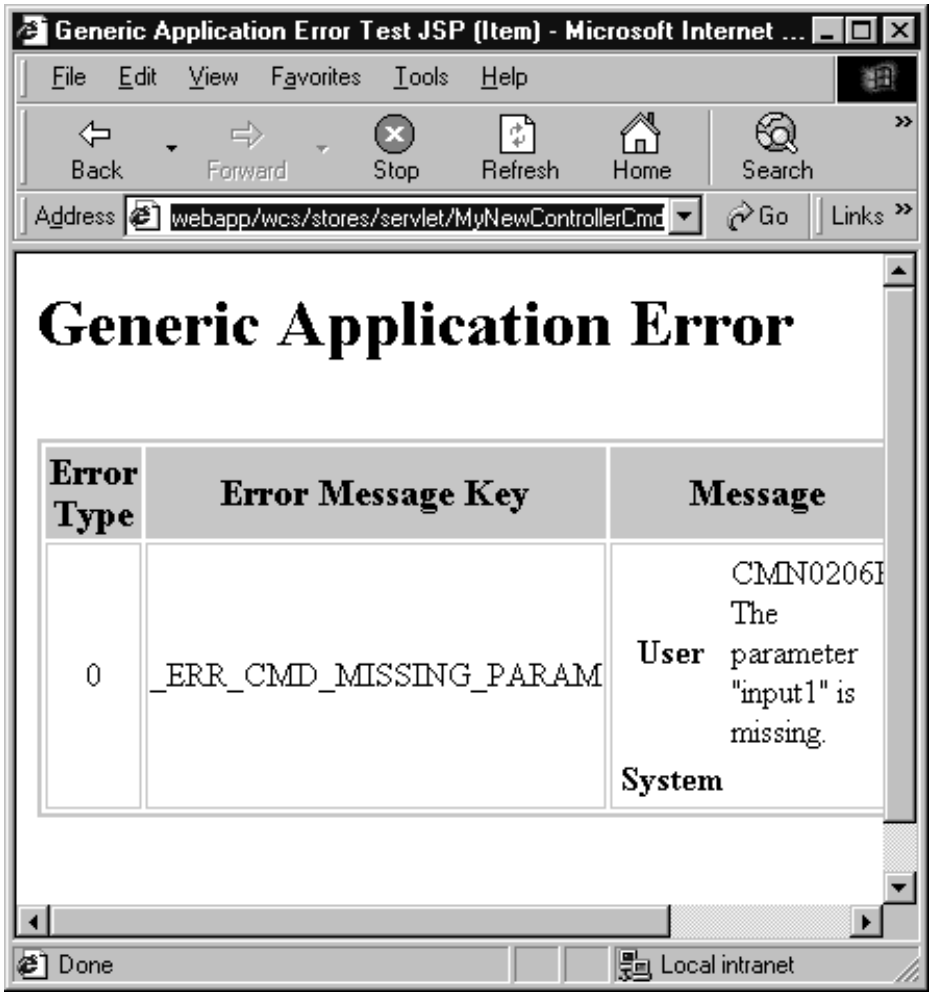

Figura 33.

Nota: Si se muestra el error "Página no encontrada", quizá sea debido a que el motor de servlets se ha detenido. En el Centro de control del Entorno de prueba WebSphere encontrará más información. Si se visualiza la página JSP de ejemplo en vez de la página de error de aplicación genérico, quizá tenga que detener y reiniciar el motor de servlets, o recargar la página en el navegador.

- Caso 2: El primer parámetro es válido, falta el segundo parámetro: Entre

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=abc
```

  El resultado de este mandato es la visualización de la página JSP de ejemplo, a pesar de omitir el segundo parámetro. La siguiente captura de pantalla muestra este resultado. A continuación se muestra una explicación de por qué no se devolvió un error.

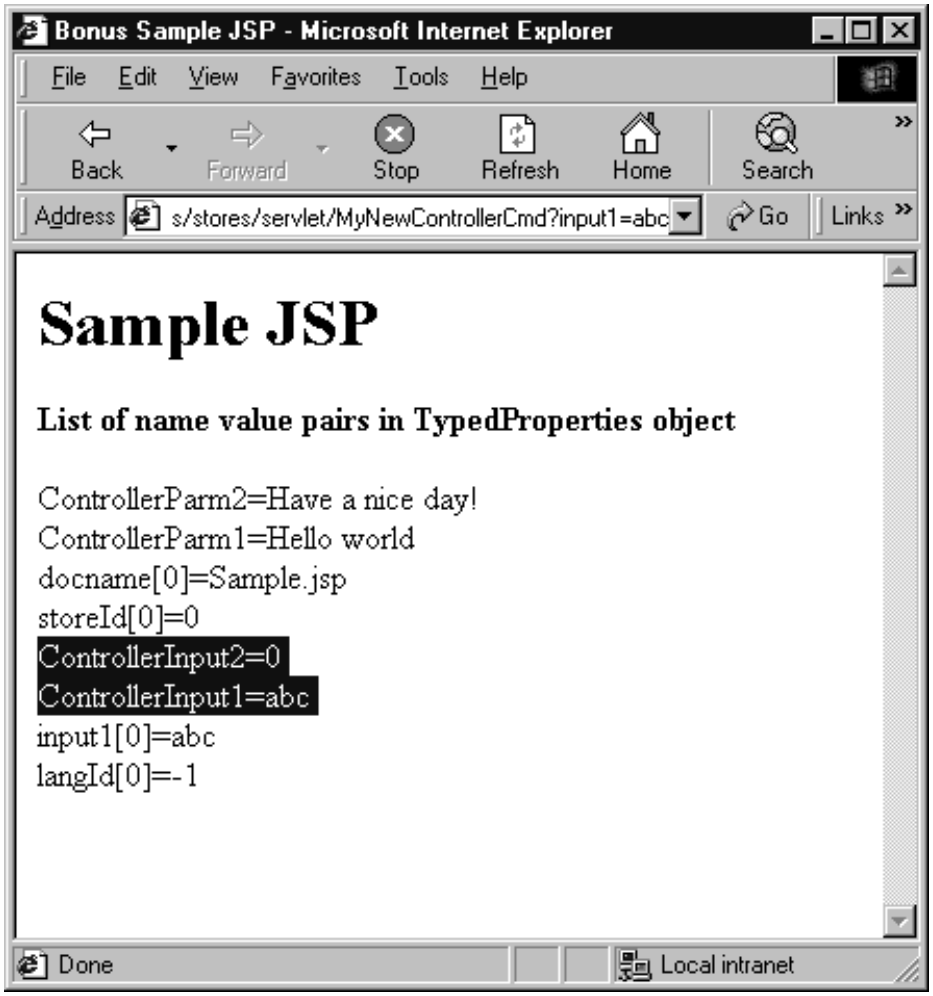

Figura 34.

No se ha devuelto un error debido al modo en que se utilizó el método `getInteger`. Concretamente, la línea de código `setInputInteger(prop.getInteger("input2", 0))` establece un valor por omisión 0 para input2. Este valor por omisión se utiliza cuando falta el parámetro o es del tipo incorrecto. Para forzar la comprobación de tipo en este parámetro, cambie el código a `setInputInteger(prop.getInteger("input2"))` y vuelva a entrar el URL (asegúrese de renovar el navegador). Debería mostrarse una página de error de aplicación genérico.

Nota: Si prueba la modificación `setInputInteger(prop.getInteger("input2"))` en su código, cámbielo por `setInputInteger(prop.getInteger("input2", 0))` antes de continuar con la nueva prueba.

- Caso 3: El primer parámetro es válido, el segundo no es válido:
  Entre

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
    input1=abc&input2=abc
```

  El resultado de este mandato es que se visualiza la página JSP de ejemplo y que el valor del segundo parámetro (un entero) se establece en el valor por omisión `0`. Este es el resultado del método `getInteger` utilizado, tal

como se describe en el caso de prueba anterior. El resultado se muestra en la siguiente ilustración:

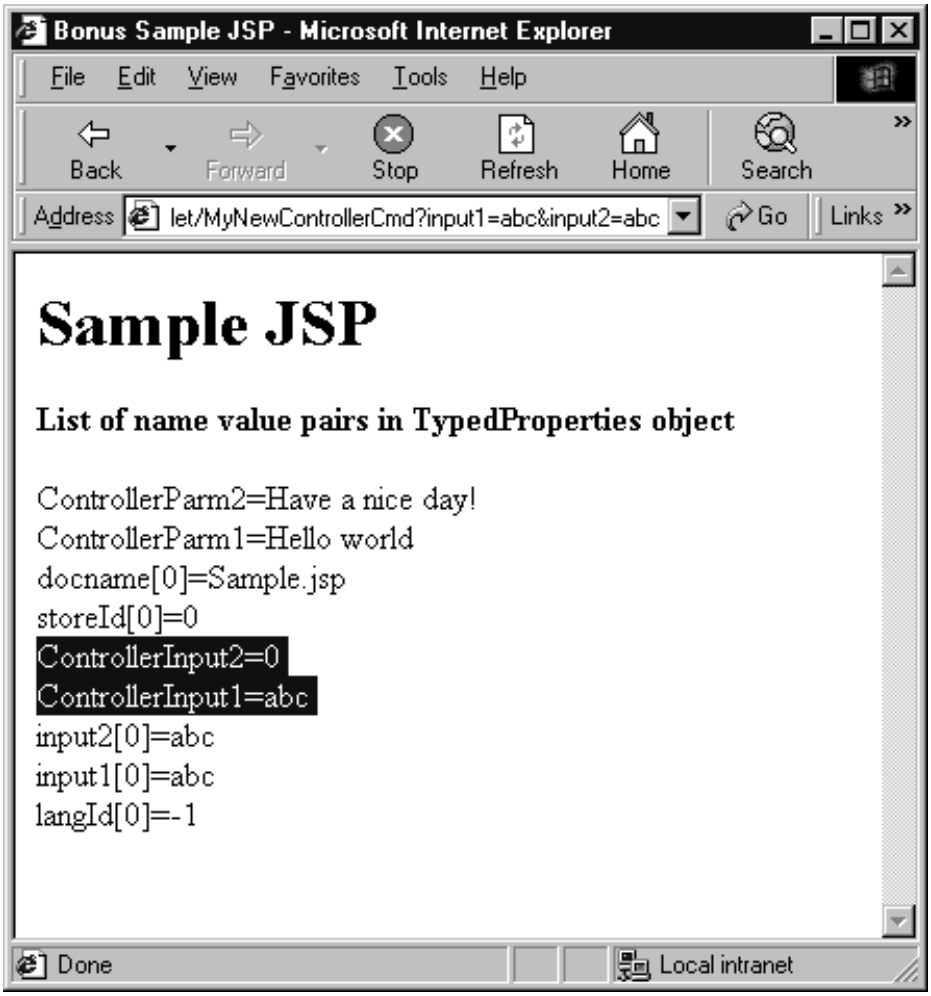

Figura 35.

- Caso 4: Los dos parámetros son válidos:
  Entre

  ```
  http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
       input1=abc&input2=1000
  ```

  El resultado de este mandato es que se visualiza la página JSP de ejemplo y los dos valores de entrada se visualizan tal como se entraron. El resultado se muestra en la siguiente ilustración:

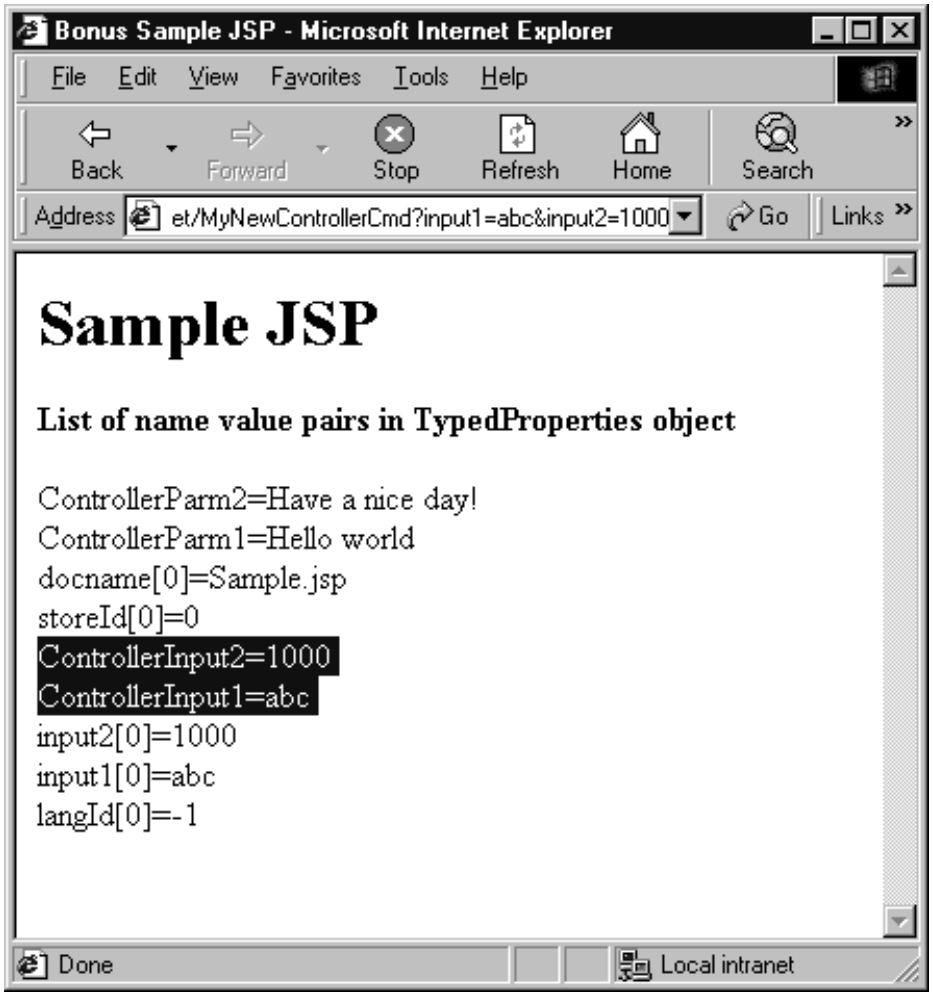

Figura 36.

7. Cree una versión de su código en su estado actual. Denomine esta versión `mySample 1.4 Completed`. Si desea información detallada sobre cómo crear una versión del código, consulte el paso 12 en la página 177.

Creación de un mandato de tarea

Un mandato de controlador representa normalmente un proceso de negocio o una función compleja. Por ejemplo, toda la lógica de negocio relacionada con el proceso de pedidos está encapsulada en el mandato de controlador OrderProcessCmd. A menudo, un proceso de negocio puede dividirse en tareas específicas más pequeñas. Por ejemplo, dentro del mandato de controlador OrderProcessCmd, hay varios mandatos de tarea a los que se llama para realizar unidades de trabajo individuales. Uno de estos mandatos de tarea a los que se llama dentro del mandato de controlador OrderProcessCmd es CalculateOrderTaxTotalCmd.

Actualmente, MyNewControllerCmdImpl no llama a ningún mandato de tarea. Este ejercicio consta de dos secciones. En la primera sección, creará el nuevo mandato de tarea. En la segunda sección, modificará el método performExecute de su mandato de controlador para llamar al mandato de tarea.

La sección de código siguiente muestra el método performExecute actual de la clase MyNewControllerCmdImpl, con todos los comentarios eliminados:

```
public void performExecute() throws ECException {

     super.performExecute();

     TypedProperty rspProp = new TypedProperty();
     rspProp.put("ControllerParm1", "Hello world");
     rspProp.put("ControllerParm2", "Have a nice day!");

     rspProp.put("ControllerInput1", getInputString());
     rspProp.put("ControllerInput2", getInputInteger().toString());

     rspProp.put(ECConstants.EC_VIEWTASKNAME, "SampleViewTask");
     setResponseProperties(rspProp);

}
```

Escribir el código del mandato de tarea: Esta sección muestra cómo escribir un mandato de tarea nuevo. La creación de un mandato de tarea totalmente nuevo implica crear una interfaz y una clase de implementación. Al crear un mandato de tarea, la interfaz debe ampliar `com.ibm.commerce.commands.TaskCommand`. La clase de implementación debe ampliar `com.ibm.commerce.command.TaskCommandImpl`.

Cuando finalice este ejercicio, dispondrá de un nuevo mandato llamado *MyNewTaskCmd*. Este mandato lo utilizan todas las tiendas y cada tienda utiliza la misma implementación del mandato.

En esta parte de la guía de aprendizaje, añadirá campos y métodos a la interfaz del nuevo mandato de tarea. Anteriormente ha creado nuevos campos para la clase MyNewControllerCmdImpl. Intente crear los campos para esta interfaz por si mismo; si necesita más detalles, consulte “Creación de nuevos campos” en la página 181.

Creación de métodos: Esta sección describe los pasos genéricos para añadir métodos a clases e interfaces existentes. Lea estas instrucciones y remítase a las mismas cuando la guía de aprendizaje le indique que debe crear nuevos métodos.

Para crear un nuevo método, haga lo siguiente:

1. Pulse con el botón derecho del ratón en la clase o interfaz a la que desea añadir el método y seleccione **Añadir > Método**.
   Se abre el SmartGuide Crear método.
2. Asegúrese de que **Crear un nuevo método** esté seleccionado y pulse **Siguiente**.
3. En el campo **Nombre del método**, especifique el nombre del nuevo método.
4. Especifique el tipo de retorno del método, haciendo una de las siguientes acciones:
   - En la lista desplegable **Tipo de retorno**, seleccione el tipo de retorno apropiado. Por ejemplo, seleccione `String`.
   - Si el tipo de retorno no está en la lista, pulse **Examinar**. A continuación, en el campo **Patrón**, entre el tipo de retorno y pulse **Aceptar**.

   Nota: En las guías de aprendizaje, cuando se especifica `String` como tipo de retorno, esto proviene del paquete `java.lang`.
5. Si el método lleva parámetros, pulse **Añadir**. En la ventana Parámetros, especifique el nombre del parámetro y demás información necesaria, y pulse **Añadir**. Después de añadir todos los parámetros, pulse **Cerrar**.
6. Pulse **Siguiente**.
   Se abre la ventana Atributos.

7. Si el método genera excepciones, pulse **Añadir** en la ventana Atributos. En el campo **Patrón**, entre el nombre de la excepción y, a continuación, pulse **Añadir**. Después de añadir todas las excepciones, pulse **Cerrar**.
8. Pulse **Finalizar**.
   Se genera el código para el método.

Para crear el mandato MyNewTaskCmd, haga lo siguiente:

1. Expanda el paquete **com.ibm.commerce.sample.commands**.
2. Pulse con el botón derecho del ratón en la interfaz **MyNewTaskCmd** y seleccione **Añadir > Campo**.
3. Utilizando el SmartGuide Crear campo, cree un campo que especifique la clase de implementación por omisión que utilizará la interfaz. Utilice los valores de la siguiente tabla. Si necesita más detalles sobre la creación de un campo, consulte "Creación de nuevos campos" en la página 181.

Nombre de atributo	Valor
Nombre de campo	`defaultCommandClassName`
Tipo de campo	`String`
Valor inicial	`"com.ibm.commerce.sample.commands.MyNewTaskCmdImpl"`

   Cuando se genera el código para el campo, aparece de la manera siguiente:

```
java.lang.String defaultCommandClassName =
     "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";
```

Puesto que se utiliza la misma clase de implementación para todo el sitio y no se pasan propiedades por omisión al mandato, puede especificar esta implementación por omisión directamente en el código. Si dispone de un mandato que tiene varias implementaciones o tiene propiedades por omisión (que se almacenan en la tabla CMDREG), debe registrar el mandato en la tabla CMDREG para crear la correlación entre la interfaz y la clase de implementación.

4. Añada nuevos métodos a la interfaz MyNewTaskCmd, haciendo lo siguiente:
   a. Pulse con el botón derecho del ratón en la interfaz **MyNewTaskCmd** y seleccione **Añadir > Método**. Utilizando el SmartGuide Crear método, cree nuevos métodos utilizando los valores que se especifican en los pasos siguientes. Para obtener información detallada sobre la creación de métodos, consulte "Creación de métodos" en la página 188.
   b. Cree un nuevo método para recuperar el saldo de puntos de bonificación del cliente, utilizando los valores siguientes:

Nombre de atributo	Valor
Nombre de método	`getOldBonusPoint`
Tipo de retorno	`String`
Parámetros	ninguno
Excepciones	ninguno

Nota: Puede aparecer un error indicando que el método abstracto heredado que acaba de crear no está implementado en la clase de implementación MyNewTaskCmdImpl. Esto se corrige en un paso siguiente.

c. Cree un nuevo método que recupere la salida del ID de usuario del mandato de tarea. Al crear este método, utilice los valores siguientes:

Nombre de atributo	Valor
Nombre de método	`getTask_output_userId`
Tipo de retorno	`String`
Parámetros	ninguno
Excepciones	ninguno

d. Cree un nuevo método que recupere un valor de salida del mandato de tarea. Al crear este método, utilice los valores siguientes:

Nombre de atributo	Valor
Nombre de método	`getTask_output1`
Tipo de retorno	`String`
Parámetros	ninguno
Excepciones	ninguno

e. Cree un nuevo método que establezca el primer valor de entrada del mandato de tarea. Al crear este método, utilice los valores siguientes:

Nombre de atributo	Valor
Nombre de método	`setTask_input1`
Tipo de retorno	`void`
Nombre de parámetro	`newTask_input1`
Tipo de referencia	`String`
Excepciones	ninguno

f. Cree un nuevo método que establezca el segundo valor de entrada del mandato de tarea. Al crear este método, utilice los valores siguientes:

Nombre de atributo	Valor
Nombre de método	`setTask_input2`
Tipo de retorno	`void`
Nombre de parámetro	`newTask_input2`
Tipos primitivos	`int`
Excepciones	ninguno

5. Añada nuevos campos a la clase MyNewTaskCmdImpl, haciendo lo siguiente:
   a. Pulse con el botón derecho del ratón en la clase **MyNewTaskCmdImpl** y seleccione **Añadir > Campo**. Utilizando el SmartGuide Crear campo, cree nuevos campos utilizando los valores que se especifican en los pasos siguientes. Para obtener información adicional sobre la creación de nuevos campos, consulte “Creación de nuevos campos” en la página 181.
   b. Cree un nuevo campo en la clase de implementación, utilizando los siguientes valores:

Nombre de atributo	Valor
Nombre de campo	`task_input1`

Nombre de atributo	Valor
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

c. Cree un nuevo campo en la clase de implementación, utilizando los siguientes valores:

Nombre de atributo	Valor
Nombre de campo	`task_input2`
Tipo de campo	`int`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

d. Cree un nuevo campo en la clase de implementación, utilizando los siguientes valores:

Nombre de atributo	Valor
Nombre de campo	`task_output_userId`
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

e. Cree un nuevo campo en la clase de implementación, utilizando los siguientes valores:

Nombre de atributo	Valor
Nombre de campo	`oldBonusPoint`
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`

Nombre de atributo	Valor
Set	`public`

f. Cree un nuevo campo en la clase de implementación, utilizando los siguientes valores:

Nombre de atributo	Valor
Nombre de campo	`task_output1`
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

6. Seleccione el método **performExecute** de la clase **MyNewTaskCmdImpl** para ver su código fuente.
7. En el código fuente, descomente la Sección 1 para insertar el siguiente código en el método:

```
// modify the task_input1 and see it in the NVP list

     setTask_output1( "Hello" + getTask_input1() );
```

Guarde el trabajo.
La sección de código anterior hace que los nuevos atributos estén disponibles como salida del mandato.

Llamar al mandato de tarea: Una vez ha creado el mandato de tarea, tiene que llamarlo desde el mandato de controlador. Los pasos siguientes describen cómo modificar el mandato de controlador de esta forma.

1. En el Entorno de trabajo, seleccione el método **performExecute** de la clase **MyNewControllerCmdImpl**.
2. En el panel Fuente, descomente la Sección 4 para llamar al mandato de tarea. Esto inserta el código siguiente en el método:

```
// see how controller command call a task command

     MyNewTaskCmd cmd = null;
     try {
          cmd = (MyNewTaskCmd) CommandFactory.createCommand(
               "com.ibm.commerce.sample.commands.MyNewTaskCmd",
               getStoreId());
          // Set input parameters to task command
          cmd.setTask_input1(getInputString());
          cmd.setTask_input2(getInputInteger().intValue());
          // This is required for all commands
          cmd.setCommandContext(getCommandContext());
          // Invoke the command's performExecute method
          cmd.execute();
          // retrieve output parameter from task command

          rspProp.put("task_output1", cmd.getTask_output1());

          if (cmd.getTask_output_userId() != null) {
               rspProp.put("task_output_userId",
```

```
                cmd.getTask_output_userId());
            }

            if (cmd.getOldBonusPoint() != null) {
                rspProp.put("task_output_oldBonusPoint",
                cmd.getOldBonusPoint());
            }

        } catch (ECException ex) {
            // throw the exception as is
            throw (ECException) ex;
        }
```

Guarde su trabajo.
La sección de código anterior crea un nuevo mandato de tarea utilizando la fábrica de mandatos. A continuación, establece el contexto de mandatos, invoca el método performExecute del mandato de tarea y recupera los parámetros de salida del mandato de tarea.

3. Pruebe el mandato entrando el URL del mandato de controlador de la siguiente manera:

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
    input1=abc&input2=1000
```

La página JSP de ejemplo muestra la lista de parejas nombre-valor en el objeto petición, incluyendo los valores de salida de la tarea. Debería aparecer de la forma indicada a continuación:

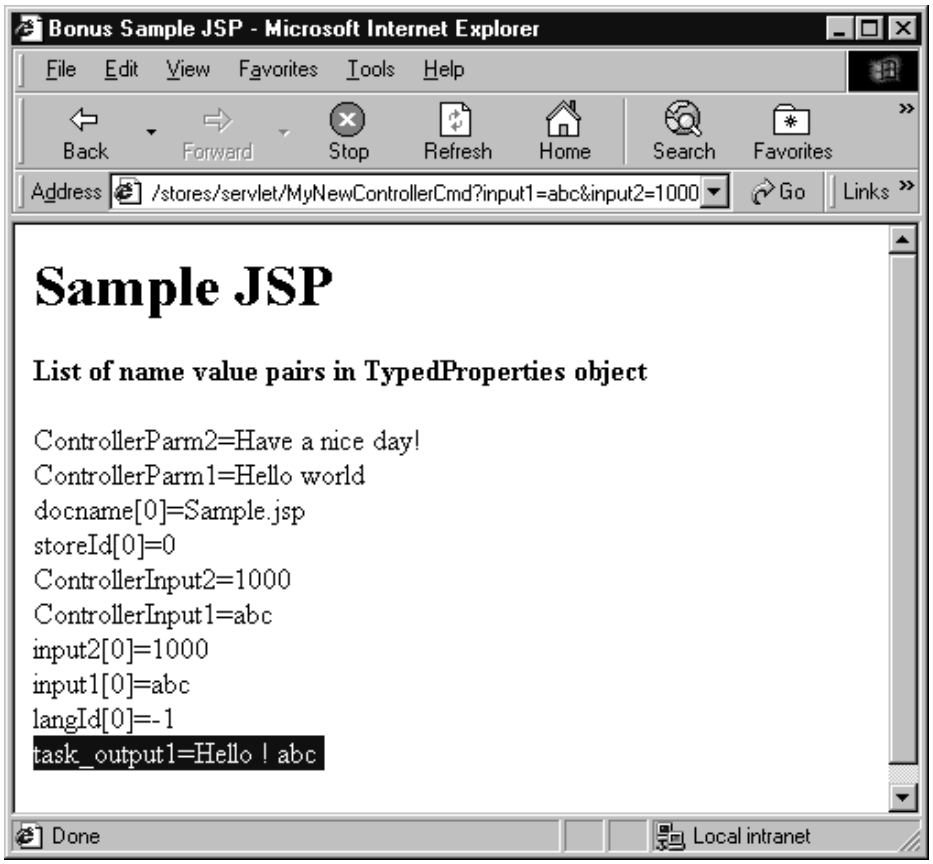

Figura 37.

4. Cree una versión de su código en su estado actual. Denomine esta versión `mySample 1.5 Completed`. Si desea información detallada sobre cómo crear una versión del código, consulte el paso 12 en la página 177.

Validar un ID de usuario

El siguiente paso es modificar el mandato de tarea para que utilice UserRegistryAccessBean con la finalidad de validar si el valor entrado por el usuario es el de un usuario registrado. Además de modificar el mandato de tarea, también debe modificar el DataBeanSampleBean.

Modificar MyNewTaskCmdImpl para la validación del ID de usuario: Debe modificar el método performExecute de la clase MyNewTaskCmdImpl para que valide el ID del usuario, utilizando UserRegistryAccessBean. Para añadir esta nueva función al método performExecute, haga lo siguiente:

1. En el Entorno de trabajo, seleccione el método **performExecute** de la clase **MyNewTaskCmdImpl**.
2. En el panel Fuente, descomente la Sección 2 del método performExecute. Esto inserta el código siguiente en el método:

```
// use UserRegistryAccessBean to check member reference number

     String refNum;

     UserRegistryAccessBean rrb = new UserRegistryAccessBean();

     try {
          rrb = rrb.findByUserLogonId(getTask_input1());
          refNum = rrb.getUserId();

     } catch (javax.ejb.FinderException e) {

     return;

     } catch (javax.naming.NamingException e) {
          throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
          this.getClass().getName(), "performExecute");
     } catch (java.rmi.RemoteException e) {
          throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
          this.getClass().getName(), "performExecute");
     } catch (javax.ejb.CreateException e) {
          throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
          this.getClass().getName(), "performExecute");
     }

      setTask_output_userId(refNum);
```

   Guarde el trabajo.

Modificar DataBeanSampleBean: Debe modificar DataBeanSampleBean para utilizar sus parámetros de entrada predefinidos. Para modificar este bean, haga lo siguiente:

1. En el Entorno de trabajo, expanda el paquete **com.ibm.commerce.sample.databeans**.
2. Añada nuevos campos al bean de datos, haciendo lo siguiente:
   a. Pulse con el botón derecho del ratón en la clase **DataBeanSampleBean** y seleccione **Añadir > Campo**. Utilizando el SmartGuide Crear campo, cree nuevos campos utilizando los valores que se especifican en los pasos siguientes. Para obtener información adicional sobre la creación de campos, consulte “Creación de nuevos campos” en la página 181.

b. Añada un nuevo campo al bean de datos, utilizando los siguientes valores:

Nombre de atributo	**Valor**
Nombre de campo	`input1`
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

c. Añada un nuevo campo al bean de datos, utilizando los siguientes valores:

Nombre de atributo	**Valor**
Nombre de campo	`input2`
Tipo de campo	`int`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

d. Añada un nuevo campo al bean de datos, utilizando los siguientes valores:

Nombre de atributo	**Valor**
Nombre de campo	`task_output_userId`
Tipo de campo	`String`
Valor inicial	Déjelo en blanco.
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

3. Seleccione el método **populate** de la clase DataBeanSampleBean para ver su código fuente.
4. En el código fuente, descomente la Sección 1A y la Sección 1B. Esto inserta el código siguiente en el método:

```
//// Section 1A //////////
// set additional data fields

try {
   setInput1( getRequestProperties().getString("input1"));
   setInput2( getRequestProperties().getIntValue("input2", 0)  );
   setTask_output_userId(getRequestProperties().getString
     ("task_output_userId"));
```

```
//// End of Section 1A /////

   //// Section 2 ////////////
   /*
   // instantiate databean to BonusAccessBean
   setTask_output_oldBonusPoint(getRequestProperties().getString(
      "task_output_oldBonusPoint"));
   */
   //// End of Section 2 ////

//// Section 1A ///////////
// instantiate databean to BonusAccessBean and
// set additional data fields

   }
catch (ParameterNotFoundException e){}

//// End of Section 1B ////
```

Guarde el trabajo.
La sección de código anterior obtiene valores de campos de datos del mandato de controlador, utilizando el método getRequestProperties.

Modificar Sample.jsp para la validación del ID de usuario: Para realizar la validación del ID de usuario debe modificarse la plantilla JSP actual. Para modificar el archivo Sample.jsp, haga lo siguiente:

1. Abra los archivos `Sample.jsp` y `Sample_All.jsp` con un editor de texto.
2. Copie la Sección 3 del código de Sample_All.jsp en Sample.jsp entre los marcadores `<!-- SECTION 3 -->` y `<!-- END OF SECTION 3 -->`. El siguiente código se inserta en la plantilla JSP

```
<!-- SECTION 3 -->

<B>Your first input is &lt; <%=testBean.getInput1()%> &gt;</B>

<%
String userId = testBean.getTask_output_userId();

if (userId == null) {

%>

<UL>
     <LI> This is not a registered user id.
</UL>

<%

   } else {

%>

<B>
<UL>
     <LI> 'lt; <%=testBean.getInput1()%> &gt; is a registed user id.
     <LI> The member reference number of this user is <%=userId%>.
</UL>
</B>

<%

}

%>
```

```
<B>Your second input is < <%=testBean.getInput2()%> ></B> <P>

<!-- END OF SECTION 3 -->
```

Guarde el archivo Sample.jsp.

3. Abra un navegador y entre el siguiente URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

El navegador debe mostrar el archivo JSP de ejemplo indicando que el valor para input1 no es un ID de usuario válido, tal como se muestra en la siguiente captura de pantalla:

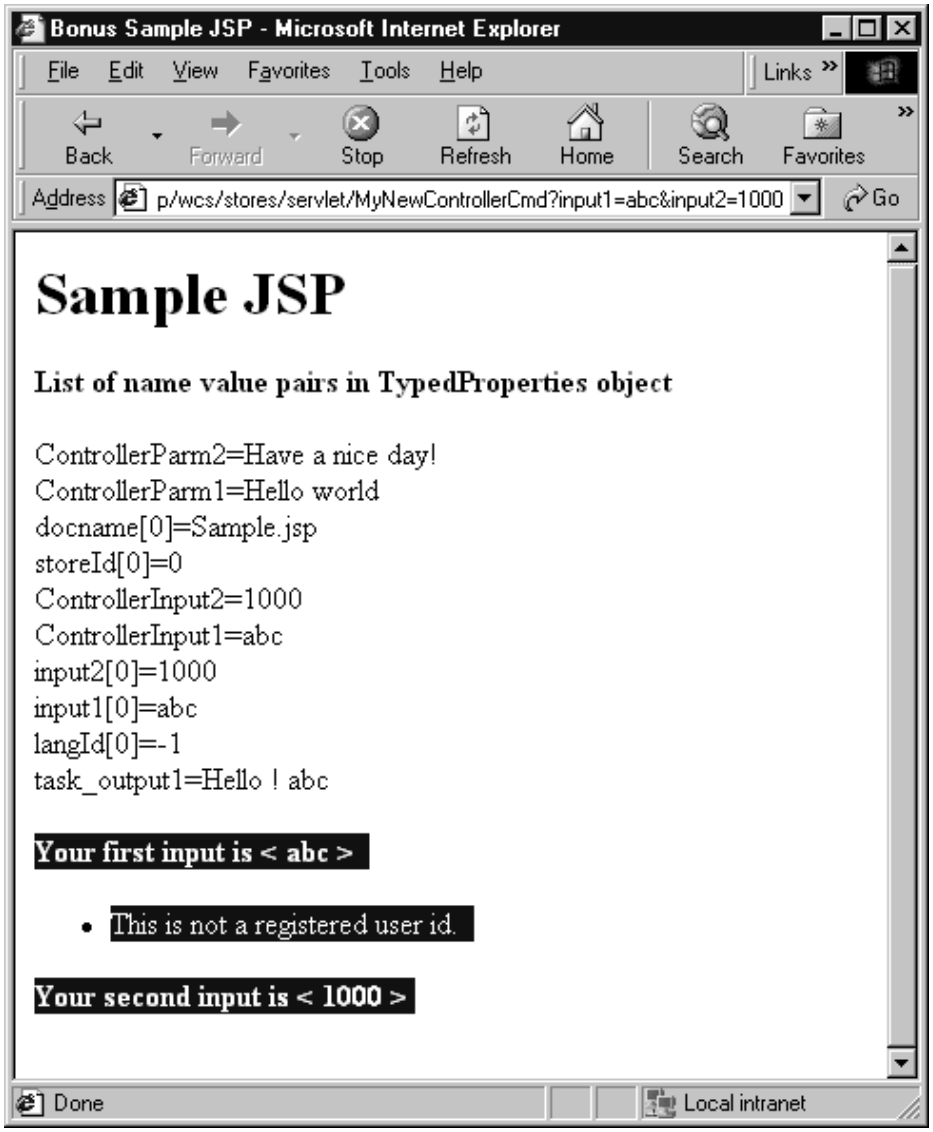

Figura 38.

4. Para ver el resultado cuando el valor para input1 es un ID de usuario válido, entre el siguiente URL:

```
http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=wcsadmin&input2=1000
```

Esto se muestra en la siguiente ilustración:

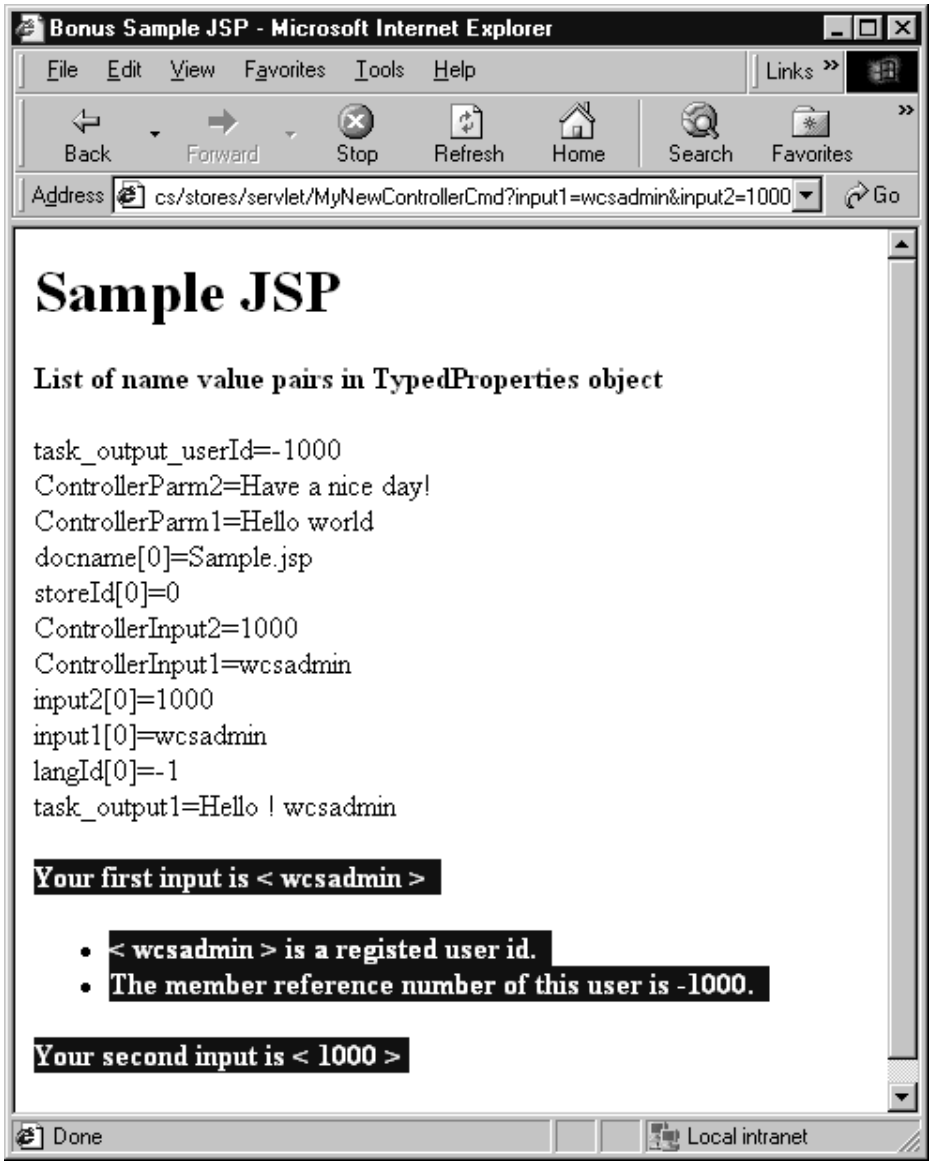

Figura 39.

5. Cree una versión de su código en su estado actual. Denomine esta versión `mySample 1.6 Completed`. Si desea información detallada sobre cómo crear una versión del código, consulte el paso 12 en la página 177.

Creación de un bean de entidad nuevo

Esta sección describe cómo utilizar VisualAge para Java para crear un bean de entidad nuevo. En este escenario de ejemplo, tiene un requisito de negocio para incluir una cuenta de puntos de bonificación para cada usuario en la aplicación de comercio. El esquema de base de datos de WebSphere Commerce no contiene esta información, de modo que es necesario crear una nueva tabla de base de datos para que contenga esta información. De acuerdo con el modelo de programación de WebSphere Commerce, una vez creada la tabla de base de datos, debe crear un bean de entidad (que es un bean enterprise) para acceder a los datos.

En este ejemplo utilizará un SmartGuide de VisualAge para Java para crear este bean de entidad.

Creación de la nueva tabla de base de datos

Como preparación para la creación del bean de entidad, primero debe crear la nueva tabla de base de datos. La tabla que se va a crear se llama Bonus.

DB2 Si utiliza una base de datos DB2, haga lo siguiente para crear la tabla:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Centro de mandatos**) y pulse la pestaña **Scripts**.
2. En la ventana Script, entre lo siguiente:

```
connect to nombre_base_de_datos;
CREATE TABLE Bonus (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (users_id) on delete cascade)
```

   donde *nombre_base_de_datos* es el nombre de su base de datos. Pulse el icono Ejecutar. Se ha creado la tabla Bonus.

   Nota: Deberá emitir el siguiente mandato antes de crear la tabla Bonus si alguien ha realizado anteriormente este ejemplo utilizando esta base de datos:

```
drop table Bonus
```

Oracle Si utiliza una base de datos Oracle, haga lo siguiente para crear la tabla:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. En la ventana de SQL Plus, entre la siguiente sentencia de SQL:

```
CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   y pulse Intro para ejecutar la sentencia de SQL. Se ha creado la tabla BONUS.

   Nota: Deberá emitir el siguiente mandato antes de crear la tabla Bonus si alguien ha realizado anteriormente este ejemplo utilizando esta base de datos:

```
drop table Bonus;
```

6. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia de SQL.

Creación del bean de entidad BonusBean

Una vez haya creado la tabla de base de datos, ya está listo para empezar a crear el nuevo bean de entidad. Los pasos siguientes utilizan VisualAge para Java. Para crear el nuevo bean de entidad, haga lo siguiente:

1. Cree un grupo EJB. Un grupo EJB es un grupo lógico que le permite organizar sus beans enterprise. Puede realizar operaciones globales en un grupo EJB que se repitan en todos los beans enterprise que residan en el grupo. Por ejemplo, si selecciona un grupo EJB para exportarlo a un archivo JAR de EJB, se exportan todos los beans enterprise del grupo.
   En esta guía de aprendizaje, va a crear un grupo EJB para organizar todos los beans enterprise relacionados con la personalización de su tabla Bonus.
   Para crear su grupo EJB, haga lo siguiente:
   a. En el Entorno de trabajo, pulse la pestaña **EJB**.
   b. En el menú **EJB**, seleccione **Añadir > Grupo EJB**.
      Se abre el SmartGuide Añadir grupo EJB.
   c. En el campo **Proyecto**, entre `_WCSamplesEntityBeansProject`.

      Nota: El código del bean de entidad debe estar almacenado dentro de su propio proyecto, para poder desplegarlo.
   d. En el campo **Crear un nuevo grupo EJB denominado:**, entre `WCSSamplesEntityBeans` y pulse **Terminar**.
2. Cree su nuevo bean de entidad.
   Para ello, haga lo siguiente:
   a. En el panel Beans enterprise, pulse con el botón derecho del ratón en el grupo EJB **WCSSamplesEntityBeans** y seleccione **Añadir > Bean enterprise**.
      Se abre el SmartGuide Crear bean enterprise.
   b. Entre la información siguiente

Atributo	Valor
Nombre de bean	`Bonus` **Nota:** El convenio de denominación es llamar al bean de entidad con el mismo nombre que la tabla a la que accede.
Tipo de bean	`Bean de entidad con campos CMP (persistencia gestionada por contenedor)`
Crear una clase de bean nueva	`habilitar`
Proyecto	`_WCSamplesEntityBeansProject`
Paquete	`com.ibm.commerce.sample.objects`
Nombre de clase	`BonusBean`
Superclase	`com.ibm.commerce.base.objects.ECEntityBean`

   y pulse **Siguiente**.
   c. Pulse el botón **Añadir** que hay junto al recuadro de texto **Añadir campos CMP al bean** para añadir un campo para la columna MEMBERID en la tabla BONUS.
      Se abre el SmartGuide Crear campos CMP.
   d. Entre la información siguiente:

Atributo	Valor
Nombre de campo	memberId
Tipo de campo	Long **Nota:** Debe utilizar el tipo de datos *Long*, no *long*.
Campo de clave	habilitar

   y pulse **Terminar**.

e. Pulse de nuevo **Añadir** para añadir un campo para la columna BONUSPOINT de la tabla BONUS.

f. Cree otro campo con la información siguiente:

Atributo	Valor
Nombre de campo	bonusPoint
Tipo de campo	Integer **Nota:** Debe utilizar el tipo de datos *Integer*, no *int*.
Acceso con métodos get y set	habilitar
Promocionar métodos get y set a interfaz remota	habilitar

y luego pulse **Terminar** en la ventana Crear campo CMP.

g. Proteja este bean enterprise mediante control de acceso, de la manera siguiente:

1) Pulse **Añadir** al lado de **¿Qué interfaces debe ampliar la interfaz remota?**.

2) Entre `com.ibm.commerce.security.Protectable` en el campo **Patrón** y pulse **Añadir**. Pulse **Cerrar** para cerrar la ventana.

h. Pulse de nuevo **Terminar**.

La entidad Bonus se crea como un bean enterprise.

3. Establezca el nivel de aislamiento del bean de entidad, haciendo lo siguiente:

a. Pulse con el botón derecho del ratón en el bean **Bonus** y seleccione **Propiedades**.

b. En la lista desplegable de **Nivel de aislamiento**, seleccione **TRANSACTION_READ_COMMITTED** y pulse **Aceptar**.

4. Cuando crea un nuevo bean enterprise, VisualAge para Java genera un campo EntityContext así como sus métodos getEntityContext() y setEntityContext(EntityContext) correspondientes en el bean. Siguiendo el modelo de programación de WebSphere Commerce, su nuevo bean amplía la clase com.ibm.commerce.base.objects.ECEntityBean y ECEntityBean proporciona su propia implementación de este campo y estos métodos. Puesto que no debe modificar EntityContext, getEntityContext ni setEntityContext, ahora debe suprimir el campo y los métodos generados de su bean.
Para suprimir el campo EntityContext generado y sus métodos getter y setter, haga lo siguiente:

a. En el panel Tipos, seleccione la clase **BonusBean**.
El panel Miembros muestra los campos y métodos para esta clase.

Nota: Si no puede visualizar el panel Tipos, pulse el icono C/I (interfaz de clase) en el panel Propiedades. El panel Tipos se abrirá.

b. En el panel Miembros, haga lo siguiente:

1) Pulse con el botón derecho del ratón en el campo **entityContext** y seleccione **Suprimir**.

2) Pulse con el botón derecho del ratón en el método **getEntityContext()** y seleccione **Suprimir**.

3) Pulse con el botón derecho del ratón en el método **setEntityContext(EntityContext)** y seleccione **Suprimir**.

c. Guarde el trabajo (Control+S).

5. Añada un nuevo método getMemberId al bean enterprise, haciendo lo siguiente:

a. Pulse con el botón derecho del ratón en la clase **BonusBean** y seleccione **Añadir > Método**.
Se abre el SmartGuide Crear método.

b. Cree el nuevo método utilizando los valores especificados en la siguiente tabla. Si necesita información más detallada sobre la creación de un nuevo método, consulte "Creación de métodos" en la página 188.

Tabla 11.

Nombre de atributo	**Valor**
Nombre de método	`getMemberId`
Tipo de retorno	`Long`
Parámetros	ninguno
Excepciones	ninguno

c. Cuando se haya generado el nuevo método, observe el código fuente.

d. Por omisión, el método contiene el siguiente código:

```
return null;
```

Cambie este código por

```
return memberId;
```

e. Añada el nuevo método a la interfaz remota pulsando el botón derecho del ratón en el método **getMemberId** y seleccionando **Añadir a > Interfaz remota EJB**.

6. Añada nuevos campos FinderHelper en BonusBeanFinderHelper.
Esta interfaz contiene una cláusula de búsqueda que se corresponde con un método FinderHelper que se creará en el paso siguiente. Para añadir los campos FinderHelper, haga lo siguiente:

a. En el panel Tipos, pulse en la interfaz **BonusBeanFinderHelper**.

b. Modifique el código del panel Fuente para que sea como el siguiente:

```
public interface BonusBeanFinderHelper {
    public static final String
                    findByMemberIdWhereClause = " (MEMBERID = ?) ";
        }
```

Nota: La sintaxis de la cláusula "WhereClause" es muy importante; debe coincidir con el nombre de método utilizado para el método FinderHelper. En este caso, "findByMemberId" de `findByMemberIdWhereClause` coincide exactamente con el nombre del método que va a crear en el paso siguiente (`findByMemberId`).

7. Añada nuevos métodos FinderHelper a la interfaz BonusHome.
Para añadir nuevos métodos FinderHelper, haga lo siguiente:

a. En el panel Tipos, pulse con el botón derecho del ratón en la interfaz **BonusHome** y seleccione **Añadir > Método**.
Se abre el SmartGuide Crear método.

b. Seleccione **Crear un nuevo método** y pulse **Siguiente**.

c. En el campo **Nombre de método**, entre `findByMemberId`.

d. En el campo **Tipo de retorno**, entre `Bonus`.

e. Pulse **Añadir** junto a **¿Qué parámetros debe tener este método?**
Se abre la ventana Parámetros.

f. En el campo **Nombre**, entre `argMemberId`.

g. Seleccione **Tipos de referencia** y entre `Long`. Pulse **Añadir** y luego **Cerrar**.

h. Pulse **Siguiente**.

i. Pulse **Añadir** junto al campo **¿Qué excepciones puede generar este método?**, entre `RemoteException` en el campo **Patrón** y pulse **Añadir**. Esto añade la excepción `java.rmi.RemoteException` en la ventana Atributos. (La ventana Atributos puede encontrarse detrás de la ventana Excepciones.)

j. En el campo **Patrón** de la ventana Excepciones, entre `FinderException`, pulse **Añadir** y luego pulse **Cerrar**. La excepción `javax.ejb.FinderException` aparece listada en la ventana Atributos.

k. Pulse **Terminar**.

8. Añada un nuevo método `ejbCreate` a EJB. Este método se promociona a la interfaz local, de modo que esté disponible en un bean de acceso generado. Para crear este método, haga lo siguiente:

   a. Seleccione la clase **BonusBean** en el panel Tipos.

   b. Pulse **ejbCreate(Long)** en el panel Miembros.

   c. Modifique el código para que coincida con el siguiente:

```
public void ejbCreate(java.lang.Long argMemberId,
  Integer argBonusPoint)
  throws javax.ejb.CreateException, java.rmi.RemoteException {
            _initLinks();
   // All CMP fields should be initialized here.
    memberId=argMemberId;
           bonusPoint=argBonusPoint;
   }
```

   d. Guarde el código. VisualAge para Java crea un nuevo método, llamado **ejbCreate(Long, Integer)**, en el panel Miembros.

   e. Pulse con el botón derecho del ratón en el método **ejbCreate(Long)** original y seleccione **Suprimir**.

9. Añada el nuevo método a la interfaz local. Esto hará que el método esté disponible en la clase de bean de acceso. Para ello, haga lo siguiente:

   a. Pulse con el botón derecho del ratón en el método **ejbCreate(Long, Integer)** en la clase BonusBean y seleccione **Añadir a > Interfaz local EJB**.

10. Actualice el método getOwner de la siguiente manera:

    a. Seleccione la clase **BonusBean** en el panel Tipos.

    b. Pulse el método **getOwner()** en el panel Miembros.

       Nota: Si ha seleccionado ver los métodos heredados, verá dos métodos getOwner(). Uno proviene de la clase ECEntityBean. Este no es el que debe seleccionar en este paso. Asegúrese de seleccionar el método getOwner específico de la clase BonusBean.

    c. El código fuente para el método getOwner aparece así:

```
public Long getOwner()
   throws Exception, java.rmi.RemoteException
   {
      return null;
   }
```

       Debe cambiar el valor que devuelve el método. A continuación se muestra en negrita la parte de código que debe cambiar:

```
public Long getOwner()
   throws Exception, java.rmi.RemoteException
   {
      return getMemberId();
   }
```

Guarde el trabajo.

d. Pulse el método **fulfills(Long, String)** en el panel Miembros.

Nota: Si ha seleccionado ver los métodos heredados, verá dos métodos fulfills(Long, String). Uno proviene de la clase ECEntityBean. Este no es el que debe seleccionar en este paso. Asegúrese de seleccionar el método fulfills(Long, String) específico de la clase BonusBean.

e. El código fuente para el método fulfills(Long, String) tiene este aspecto:

```
public boolean fulfills(Long member, String relationship)
  throws Exception, java.rmi.RemoteException
  {
      return false;
  }
```

Debe especificar la relación que debe satisfacer el usuario. Para ello, debe cambiar la parte de código que se muestra en negrita:

```
public boolean fulfills(Long member, String relationship)
  throws Exception, java.rmi.RemoteException
  {

      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }      return false;
  }
```

Guarde el trabajo.

11. Correlacione la tabla de base de datos BONUS con BonusBean. El primer paso para correlacionar el esquema de base de datos con la entidad BonusBean requiere utilizar las herramientas de VisualAge para Java para crear el esquema de base de datos. Haga lo siguiente para crear el esquema:
    a. En la ventana de Área de trabajo, en el menú **EJB**, seleccione **Abrir en > Esquemas de base de datos**.
    b. En el menú **Esquemas**, seleccione **Importar/Exportar esquema > Importar esquema de base de datos**.
    Se abre la ventana Información necesaria.
    c. En el campo **Nombre de esquema**, entre `WCSSamples` y pulse **Aceptar**.
    Se abre la ventana Información de conexión de base de datos.
    d. Entre la información siguiente:

Atributo	DB2 **Valor DB2**	Oracle **Valor Oracle**
Tipo de conexión	`COM.ibm.db2.jdbc.app.DB2Driver`	`Oracle.jdbc.driver.OracleDriver`
Origen de datos	`jdbc:db2:`*`nombre_bd_wcs`*	`jdbc:oracle:thin:@`*`sistpral`*` :`*`puerto`*`:`*`SID`*
Nombre de usuario	*`nombre_usuario_bd_wcs`*	*`nombre_usuario_bd_wcs`*
Contraseña	*`contraseña_bd_wcs`*	*`contraseña_bd_wcs`*

sustituyendo los valores de la forma siguiente:

- DB2 *`nombre_bd_wcs`* es el nombre de la base de datos de WebSphere Commerce
- Oracle *`sistpral`* es el nombre de sistema principal de Oracle

- Oracle *puerto* es el número de puerto de la base de datos Oracle (por ejemplo, 1521).
- *nombre_usuario_bd_wcs* es el nombre de usuario de la base de datos.
- *contraseña_bd_wcs* es la contraseña de la base de datos.

Pulse **Aceptar**.
Se abre la ventana Seleccionar tablas.

e. En la lista **Calificadores**, seleccione el calificador de su base de datos (puede ser su nombre de usuario de base de datos o el nombre de su máquina) y pulse **Crear lista de tablas**. Se carga una lista de las tablas de base de datos disponibles.

f. Seleccione **Bonus** en el panel Tablas y pulse **Aceptar**. Espere unos minutos.

g. En el Examinador de esquemas, pulse en la tabla recién añadida para ver si aparecen las dos columnas de la tabla.

h. Pulse con el botón derecho del ratón en la tabla **Bonus** y seleccione **Editar tabla**.
Se abre el Editor de tablas.

i. Suprima la información que haya en el campo **Calificador**. Es una buena práctica eliminar la información del calificador de forma que el código pueda desplegarse en otras máquinas utilizando una base de datos diferente.

j. Oracle Modifique los tipos de datos de columna, tal como se indica a continuación:

1) Seleccione la columna **MEMBERID** y, a continuación, pulse **Editar**. En la lista desplegable Tipo, seleccione **BIGINT** y pulse **Aceptar**.

2) Seleccione la columna **BONUSPOINT** y, a continuación, pulse **Editar**. En la lista desplegable Tipo, seleccione **INTEGER** y pulse **Aceptar**.

k. Pulse **Aceptar** para salir del editor de tablas.

l. En el menú **Esquemas**, seleccione **Guardar esquema**.
Se abre la ventana Guardar esquema.

m. Entre la información siguiente:

Atributo	Valor
Proyecto	`_WCSamplesEntityBeansProject`
Paquete	`com.ibm.commerce.sample.objects`
Nombre de clase	`WCSSamplesSchema`

pulse **Terminar** y cierre el Examinador de esquemas.

12. Una vez creado el esquema, puede crear la correlación del esquema. Para crear esta correlación entre la tabla BONUS y la entidad BonusBean, haga lo siguiente:

a. En el menú **EJB**, seleccione **Abrir en > Correlaciones de esquemas**. Se abre el Examinador de correlaciones.

b. En el Examinador de correlaciones, en el menú **Correlaciones de almacenes de datos**, seleccione **Nueva correlación de grupo EJB**.
Se abre la ventana Nueva correlación de almacenes de datos.

c. Entre la información siguiente:

Atributo	Valor
Nombre	`WCSSamples`

Atributo	Valor
Grupo EJB	WCSSamplesEntityBeans
Esquema	WCSSamples

y pulse **Aceptar**.

d. En el panel Correlaciones de almacenes de datos, pulse **WCS Samples**.

e. En el panel Clases persistentes, pulse **Bonus**.

f. En el menú **Correlaciones de tablas**, seleccione **Nueva correlación de tabla > Añadir correlación de tabla sin herencia**.

g. En la lista desplegable **Tabla**, seleccione **Bonus** y pulse **Aceptar**.

h. En el panel Correlaciones de tablas, seleccione **Bonus** y luego pulse con el botón derecho del ratón en él y seleccione **Editar correlaciones de propiedades**.
Se abre el Editor de correlaciones de propiedades.

i. Establezca los atributos de la siguiente manera:

Atributo de clase	Tipo de correlación	Columna de tabla
memberId	Simple	MEMBERID
bonusPoint	Simple	BONUSPOINT

y pulse **Aceptar**.

j. En el menú **Correlaciones de almacenes de datos**, seleccione **Guardar correlación de almacenes de datos**.
Se abre la ventana Guardar correlación de almacenes de datos.

k. Entre la información siguiente:

Atributo	Valor
Proyecto	_WCSamplesEntityBeansProject
Paquete	com.ibm.commerce.sample.objects
Nombre de clase	WCSSamplesMap

y luego pulse **Terminar** y cierre el Examinador de correlaciones.

13. Una vez se haya creado la entidad BonusBean y el esquema esté correlacionado correctamente, debe crear un bean de acceso para el bean de entidad. Este bean de acceso simplifica el acceso de las aplicaciones a la información incluida en el bean de entidad Bonus. Las herramientas de VisualAge para Java se utilizan para generar este bean de acceso, sobre la base de la entidad que ya ha creado (en particular, el bean de acceso sólo utilizará los métodos que se han promocionado a la interfaz remota). Para crear el bean de acceso para su bean de entidad Bonus, haga lo siguiente:

a. En el Entorno de trabajo, con la pestaña EJB seleccionada, pulse con el botón derecho del ratón en el bean enterprise **Bonus** y seleccione **Añadir > Bean de acceso**.
Se abre el SmartGuide Crear bean de acceso (esta ventana puede tardar un poco en abrirse).

b. Asegúrese de entrar la siguiente información:

Atributo	Valor
Grupo EJB	WCSSamplesEntityBeans
Bean enterprise	Bonus

Atributo	Valor
Nombre de bean de acceso	BonusAccessBean
Tipo de bean de acceso	CopyHelper para Bean de entidad

y pulse **Siguiente**.

c. En la lista desplegable **Seleccionar método local para constructor sin argumentos**, seleccione **findByMemberId(Long)**.

d. Para **init_argMemberId**, en la columna Propiedades iniciales, establezca **Conversor** en com.ibm.commerce.base.objects.WCSStringConverter y pulse **Siguiente**.

e. Para bonusPoint, asegúrese de que **CopyHelper** esté seleccionado, establezca el valor de **Conversor** en com.ibm.commerce.base.objects.WCSStringConverter y pulse **Terminar**.

Nota: No es necesario efectuar modificaciones en el campo memberId.

f. Pulse **Aceptar** cuando se visualiza el mensaje "Generación de código completada".

Para ver el código que se acaba de generar, vaya a la pestaña **Proyectos** y expanda el proyecto **_WCSamplesEntityBeansProject** y luego expanda **com.ibm.commerce.sample.objects**. Una nueva clase denominada BonusAccessBean aparece dentro del paquete.

14. El paso siguiente es generar código desplegado.
El programa de utilidad para generación de código analiza los beans para asegurarse de que se cumplan las especificaciones EJB de Sun Microsystems y comprobar que se sigan las normas específicas para el servidor EJB. Además, para cada bean enterprise seleccionado, la herramienta de generación de código genera las implementaciones local y EJBObject (remota) y las clases de implementación para las interfaces local y remota, así como las clases de buscador y persistencia JDBC para los beans CMP. También genera las clases Java ORB, "apéndices" y de relación necesarias para el acceso RMI a través de IIOP, así como "apéndices" para las interfaces local y remota.
Si ha seleccionado un grupo EJB que contiene un bean enterprise CMP, o si ha seleccionado un bean enterprise CMP individual, también se generan los siguientes elementos:
   - Una serie de creación de tabla que se genera en la clase de persistencia.
   - Una implementación de método de persistencia que establece una correlación con la tabla

   Para generar el código desplegado, haga lo siguiente:

   a. Con la pestaña EJB seleccionada, en el panel Beans enterprise, pulse con el botón derecho del ratón en el bean enterprise **Bonus** y seleccione **Generar código desplegado**. La generación del código tarda unos minutos.

15. Antes de probar el bean enterprise Bonus, debe crear un nuevo servidor EJB que contenga todos los grupos EJB de WebSphere Commerce y también el nuevo grupo EJB WCSSamplesEntityBeans. Estos grupos deben estar ejecutándose en el mismo servidor para que se mantenga el ámbito de transacciones. Una vez creado el nuevo servidor, debe iniciarlo.
Para crear e iniciar el nuevo servidor de EJB, haga lo siguiente:

   a. Compruebe que todos los grupos EJB estén contraídos.

   b. En el panel Beans enterprise, seleccione todos los grupos EJB de WebSphere Commerce y su nuevo grupo EJB (es decir, seleccione todos los grupos que empiezan con WCS).

c. Con estos grupos EJB seleccionados, pulse el botón derecho del ratón y seleccione **Añadir a > Configuración de servidor**.
Se abre la ventana Configuración de servidor EJB (esto puede tardar un poco).

d. Pulse con el botón derecho del ratón en el Servidor EJB que acaba de crear (por ejemplo, **Servidor EJB (server2)**), seleccione **Propiedades** y entre la siguiente información:

Atributo	DB2 **Valor DB2**	Oracle **Valor Oracle**
Origen de datos	`WebSphere Commerce DB2 DataSource` *`nombre_instancia`*	`WebSphere Commerce Oracle DataSource` *`nombre_instancia`*
Tipo de conexión	`<OrigenDatos>`	`<OrigenDatos>`
Nombre de usuario	*`nombre_usuario_bd_wcs`*	*`nombre_usuario_bd_wcs`*
Contraseña	*`contraseña_bd_wcs`*	*`contraseña_bd_wcs`*
Tiempo de espera de transacción	`1200`	`1200`
Tiempo de espera de inactividad de transacción	`600000`	`600000`

y pulse **Aceptar**.

Nota: El valor para Origen de datos debe coincidir con el valor para origen de datos que se especifica en el archivo *`nombre_instancia`*`.xml`.

Dependiendo del hardware de su máquina de desarrollo (por ejemplo, la velocidad del procesador), quizá tenga que aumentar los valores de las propiedades del servidor EJB **Transaction Timeout** y **Transaction Inactivity**.

e. Si el Entorno de prueba WebSphere está ejecutándose, deténgalo y detenga también el servidor de nombres persistentes y otro servidor EJB, tal como se describe en el Apéndice A, “Inicio y detención del Entorno de prueba WebSphere” en la página 281.

f. Inicie el servidor de nombres persistentes, tal como se describe en “Inicio y detención del servidor de nombres persistentes” en la página 281.

g. Inicie el nuevo servidor EJB, tal como se describe en “Inicio y detención del servidor EJB” en la página 281.

16. Una vez se haya iniciado el servidor EJB, ya puede iniciar el cliente de prueba. Al utilizar el cliente de prueba, se crea un registro nuevo en la base de datos. Para iniciar el cliente de prueba y crear este registro, haga lo siguiente:

a. Con la pestaña EJB seleccionada, pulse con el botón derecho del ratón en el bean enterprise **Bonus** y seleccione **Ejecutar cliente de prueba**.

b. En la ventana Búsqueda EJB, pulse **Búsqueda**.
Se abre la ventana Bonus.

c. Pulse **create(Long, Integer)**. En el panel Detalles, entre lo siguiente:

Atributo	**Valor**
Long	`-1000`
Integer	`100`

y pulse el icono Invocar en la ventana Cliente de prueba EJB.

Nota: El primer atributo debe coincidir con el ID de miembro (memberId) de cualquier usuario registrado. Puede determinar el ID de miembro mirando la tabla USERS.

17. Compruebe que el registro de base de datos se ha creado correctamente, consultando directamente la base de datos.

    DB2 Si utiliza una base de datos DB2, haga lo siguiente:

    a. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Centro de mandatos**)
    b. Seleccione la pestaña **Interactivo**.
    c. Entre `connect to` *`nombre_base_datos_wcs`* y pulse el icono Ejecutar.
    d. Entre `SELECT * FROM Bonus` y pulse el icono Ejecutar.
    Deberían devolverse los siguientes valores:

Columna	Valor
MEMBERID	`-1000`
BONUSPOINT	`100`

    El registro arriba mostrado ha sido creado por su cliente de prueba EJB.

    Oracle Si utiliza una base de datos Oracle, haga lo siguiente:

    a. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle - OraHome81> Application Development > SQL Plus**).
    b. En el campo **User Name**, entre su nombre de usuario de Oracle.
    c. En el campo **Password**, entre su contraseña de Oracle.
    d. En el campo **Host String**, entre su serie de conexión.
    e. En la ventana de SQL Plus, entre lo siguiente:

```
select * from BONUS;
```

    y pulse Intro para ejecutar la sentencia de SQL.
    Deben devolverse los valores siguientes:

Columna	Valor
MEMBERID	`-1000`
BONUSPOINT	`100`

18. Utilice el cliente de prueba para comprobar que el bean enterprise Bonus puede acceder satisfactoriamente al registro de base de datos, efectuando lo siguiente:

    a. En la ventana Bonus, seleccione la pestaña **Local**.
    b. En el panel Métodos, pulse **findByMemberId(Long)**.
    c. En el campo **Long**, entre `–1000` y pulse el icono Invocar.
    d. Con la pestaña **Remoto** seleccionada, expanda **Métodos**, pulse **getBonusPoint** y luego pulse el icono Invocar.
    El panel Detalles muestra el entero `100`.
    e. Cierre las ventanas Cliente de prueba Bonus y EJB.

Integrar el bean de entidad Bonus con MyNewControllerCmd

En la sección anterior, ha probado el nuevo bean de entidad Bonus utilizando el cliente de prueba que se generó en VisualAge para Java. En esta sección, va a integrar el bean de entidad Bonus con la lógica MyNewControllerCmd. Una vez se

haya actualizado el código Java, se actualizará la plantilla Sample.jsp para crear una interfaz que permita realizar actualizaciones en el saldo de puntos de bonificación de los clientes.

Para integrar el bean de entidad Bonus es necesario realizar los siguientes pasos generales:

1. Modificar el método performExecute de la clase MyNewTaskCmdImpl de modo que calcule los nuevos puntos de bonificación y los guarde en la tabla BONUS.
2. Añadir un método getResources a la clase MyNewControllerCmdImpl para que devuelva una lista de los recursos que utiliza el mandato. Este método se incluye para control de acceso.
3. Crear una nueva política de control de acceso para los nuevos recursos.
4. Modificar DataBeanSampleBean para que se amplíe a partir del bean de acceso para el bean de entidad Bonus. Al hacer que el bean de datos se amplíe a partir del bean de acceso, el bean de datos heredará todos los atributos del bean de acceso.
5. Modificar los métodos de DataBeanSampleBean.
6. Modificar la vía de acceso de clases para el motor de servlets en el Entorno de prueba WebSphere, de modo que incluya el nuevo proyecto _WCSamplesEntityBeansProject.
7. Modificar la plantilla Sample.jsp para permitir a los usuarios entrar puntos de bonificación y visualizar el resultado.

Modificar MyNewTaskCmdImpl para el cálculo de puntos de bonificación: MyNewTaskCmdImpl se utiliza como punto de integración para el bean de entidad Bonus y MyNewControllerCmd (puesto que MyNewControllerCmd invoca MyNewTaskCmd).

Para modificar MyNewTaskCmdImpl de modo que lleve a cabo el cálculo de puntos de bonificación, haga lo siguiente:

1. En la ventana Entorno de trabajo de VisualAge para Java, expanda el proyecto **_WCSamples**.
2. Expanda el paquete **com.ibm.commerce.sample.commands** y, a continuación, seleccione la clase **MyNewTaskCmdImpl** para ver su código fuente.
3. Descomente la siguiente sentencia import:

   ```
   import com.ibm.commerce.sample.objects.*;
   ```

   Guarde el trabajo (**Control + S**).
4. Seleccione el método **performExecute** de la clase **MyNewTaskCmdImpl**.
5. En el código fuente para el método performExecute, descomente la Sección 3. Esto inserta el código siguiente en el método:

   ```
   // use BonusAccessBean to update new bonus point

   String newBonusPoint = null;
   BonusAccessBean bb = new BonusAccessBean();
   try {
        if (refNum != null) {
             bb.setInit_argMemberId(refNum);
             bb.refreshCopyHelper();
             oldBonusPoint = bb.getBonusPoint();
        }
   } catch (javax.ejb.FinderException e) {
        try {
   ```

```
            bb = new BonusAccessBean(new Long(refNum),new Integer(0));
            oldBonusPoint = "0";
        } catch (javax.ejb.CreateException ec) {
            throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
                this.getClass().getName(), "performExecute");
        } catch (javax.naming.NamingException ec) {
            throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
                this.getClass().getName(), "performExecute");
        } catch (java.rmi.RemoteException ec) {
            throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
                this.getClass().getName(), "performExecute");
        }
    } catch (javax.naming.NamingException e) {
        throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
            this.getClass().getName(), "performExecute");
    } catch (java.rmi.RemoteException e) {
        throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
            this.getClass().getName(), "performExecute");
    } catch (javax.ejb.CreateException e) {
        throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
            this.getClass().getName(), "performExecute");
    }


    try {
        if (oldBonusPoint != null) {
            int newBP =  Integer.parseInt(oldBonusPoint) + getTask_input2();
            newBonusPoint = Integer.toString( newBP );
            bb.setBonusPoint( newBonusPoint )   ;
            newBonusPoint=bb.getBonusPoint();
            bb.commitCopyHelper();
        }
    } catch (javax.ejb.FinderException e) {
        throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
            this.getClass().getName(), "performExecute");
    } catch (javax.naming.NamingException e) {
        throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
            this.getClass().getName(), "performExecute");
    } catch (java.rmi.RemoteException e) {
        throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
            this.getClass().getName(), "performExecute");
    } catch (javax.ejb.CreateException e) {
        throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
            this.getClass().getName(), "performExecute");
    }
```

Guarde el trabajo.

Adición del método getResources a la clase MyNewControllerCmdImpl: En esta sección, se añade un nuevo método getResources a la clase MyNewControllerCmdImpl. Este método devuelve una lista de los recursos que el mandato utiliza durante el proceso. Este método es necesario para el control de acceso a nivel de recurso.

Para añadir el método getResources, haga lo siguiente:

1. Con la pestaña **Proyectos** seleccionada, expanda el proyecto **_WCSamples**.
2. Expanda el paquete **com.ibm.commerce.sample.commands**.
3. Seleccione la clase **MyNewControllerCmdImpl** para ver su código fuente.
4. En el código fuente, descomente la sección de control de acceso. Esta sección se muestra en el extracto de código siguiente:

```
public AccessVector getResources() throws ECException{

   // use UserRegistryAccessBean to check member reference number
```

```
   String refNum;
   String methodName="getResources";

   com.ibm.commerce.user.objects.UserRegistryAccessBean rrb =
      new com.ibm.commerce.user.objects.UserRegistryAccessBean();

   try {
      rrb = rrb.findByUserLogonId(getInputString());
      refNum = rrb.getUserId();

   }
   catch (javax.ejb.FinderException e) {

      throw new ECSystemException(ECMessage._ERR_BAD_USER_NAME,
         this.getClass().getName(),methodName);

   }
   catch (javax.naming.NamingException e) {
      throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
         this.getClass().getName(), methodName);
   }
   catch (java.rmi.RemoteException e) {
      throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
         this.getClass().getName(), methodName);
   }
   catch (javax.ejb.CreateException e) {
      throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
      this.getClass().getName(), methodName);
   }


   //find the Bonus bean for this user
   String newBonusPoint = null;
   com.ibm.commerce.sample.objects.BonusAccessBean bb =
      new com.ibm.commerce.sample.objects.BonusAccessBean();
   try {
      if (refNum != null) {
         bb.setInit_argMemberId(refNum);
         bb.refreshCopyHelper();
      }
   }
catch (javax.ejb.FinderException e) {

   // The user doesn't have a Bonus object so return the container that
   // will hold the bonus object when it's created

   return new AccessVector(rrb);

   }
   catch (javax.naming.NamingException e) {
      throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
         this.getClass().getName(), methodName);
   }

   catch (java.rmi.RemoteException e) {
      throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
      this.getClass().getName(), methodName);
   }

   catch (javax.ejb.CreateException e) {
      throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
      this.getClass().getName(), methodName);
   }
```

```
        return new AccessVector(bb);

    }
```

Guarde el trabajo. (Ctrl+S).

Después de guardar la sección de código anterior, VisualAge para Java separa los campos y los métodos de acceso fuera de esta vista concreta. Tenga en cuenta que ahora aparecerán bajo la clase MyNewControllerCmdImpl y el método está marcado con una M.

Nota: Para mayor simplicidad en esta guía de aprendizaje, los objetos de recursos se crean en este método getResources. En una aplicación real, es preferible crear los objetos de recursos en el método validateParameters y guardarlos como variables de instancia. Así, los métodos getResources y performExecute podrían volver a utilizar estos objetos.

Estableciendo la política de control de acceso para el nuevo recurso: Se proporciona una política de control de acceso de ejemplo. Esta política crea los siguientes objetos de control de acceso:

Una acción
La acción que se crea es `com.ibm.commerce.sample.commands.MyNewControllerCmd`

Un grupo de acciones
El grupo de acciones que se crea es `MyNewControllerCmdActionGroup`. Este grupo de acciones sólo contiene una acción: `com.ibm.commerce.sample.commands.MyNewControllerCmd`

Una categoría de recursos
La categoría de recursos que se crea es `com.ibm.commerce.sample.objects.BonusResourceCategory.` Esta categoría de recursos es para el bean de entidad Bonus.

Un grupo de recursos
El grupo de recursos que se crea es `BonusResourceGroup`. Este grupo de recursos sólo contiene la categoría de recursos anterior.

Una política
La política que se crea es `AllUsersUpdateBonusResourceGroup`. Esta política permite a los usuarios efectuar la acción MyNewControllerCmd en el bean Bonus sólo si el usuario es el "propietario" del objeto de bonificación. Por ejemplo, si el usuario se ha conectado como usuario `wcsadmin`, sólo puede modificar los puntos de bonificación para `wcsadmin`.

Para establecer la política `AllUsersUpdateBonusResourceGroup` deben efectuarse los siguientes pasos:

1. Cargar el archivo `SampleACPolicy.xml` utilizando el mandato `acpload`.
2. Cargar la descripción de `SampleACPolicy_`***`entorno_nacional`***`.xml` utilizando el mandato `acpnlsload`.
3. Renovar el registro de políticas. Observe que este paso sólo es necesario si el motor de servlets se está ejecutando en el momento en que se carga la política de control de acceso.

Para establecer la política `AllUsersUpdateBonusResourceGroup`, haga lo siguiente:

1. En un indicador de mandatos, vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServerDev\bin
2. Para cargar el archivo SampleACPolicy.xml, debe emitir el mandato acpload, que tiene el siguiente formato:

   ```
   acpload nombre_bd usuario_bd contraseña_bd archXMLentrada
   ```

   donde
   - *nombre_bd* es el nombre de su base de datos
   - *usuario_bd* es el nombre de usuario de la base de datos
   - *contraseña_bd* es la contraseña de la base de datos
   - *archXMLentrada* es el nombre del archivo XML que contiene la política

   Por ejemplo, puede emitir el mandato siguiente:

   ```
   acpload VAJ_Demo user password SampleACPolicy.xml
   ```
3. Para cargar la descripción de la política, debe emitir el mandato acpnlsload, que tiene el siguiente formato:

   ```
   acpnlsload
   nombre_bd usuario_bd
   contraseña_bd
   archXMLentrada
   ```

   Por ejemplo, puede emitir el mandato siguiente:

   ```
   acpnlsload VAJ_Demo user password SampleACPolicy_es_ES.xml
   ```
4. Si el motor de servlets para el Entorno de prueba WebSphere está en ejecución, deténgalo y vuelva a iniciarlo para renovar el registro de política.

Modificar DataBeanSampleBean para los puntos de bonificación: En esta sección se modifica DataBeanSampleBean para ampliar BonusAccessBean, haciendo lo siguiente:

1. En el Entorno de trabajo, expanda el paquete **com.ibm.commerce.sample.databeans**.
2. Añada un nuevo campo al bean de datos, haciendo lo siguiente:
   a. Pulse con el botón derecho del ratón en la clase **DataBeanSampleBean** y seleccione **Añadir > Campo**. Utilizando el SmartGuide Crear campo, cree nuevos campos tal como se describe en los pasos siguientes. Para obtener información adicional sobre la creación de campos, consulte “Creación de nuevos campos” en la página 181.
   b. Añada un nuevo campo al bean de datos, utilizando los siguientes valores:

Nombre de atributo	**Valor**
Nombre de campo	task_output_oldBonusPoint
Tipo de campo	String
Valor inicial	Déjelo en blanco.
Modificadores de acceso	private
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	Seleccionado
Get	public
Set	public

   Pulse **Terminar**.

3. Pulse la clase **DataBeanSampleBean** para ver su código fuente. En el código fuente, descomente la siguiente sentencia import:

   ```
   import com.ibm.commerce.sample.objects.*;
   ```

   Guarde el trabajo.

4. Sin salir del código fuente de la clase DataBeanSampleBean, descomente la Sección 1 y comente la Sección 2. Esto hace que el bean se amplíe de BonusAccessBean. Después de efectuar estas modificaciones, el código aparece de la siguiente manera:

   ```
   /// Section 1 ////////////////////////////////////////////////

   // Extend the databean to BonusAccessBean

   public class DataBeanSampleBean
      extends com.ibm.commerce.sample.objects.BonusAccessBean
      implements SmartDataBean {


   ////////////////////////////////////////////////////////////

   /// Section 2 ////////////////////////////////////////////////
   /*
   // Extend the databean to BonusAccessBean

        public class DataBeanSampleBean implements SmartDataBean {
   */

   //
   ////////////////////////////////////////////////////////////
   ```

   Guarde el trabajo.

5. Seleccione el método **setTask_output_userId(String)** para ver su código fuente. Localice la línea de código siguiente:

   ```
   public void setTask_output_userId(java.lang.String newTask_output_userId) {
   task_output_userId = newTask_output_userId;
   ```

   Después de la línea anterior, entre el código siguiente para crear una instancia del nuevo BonusAccessBean:

   ```
   ////////////////////////////////////
   // Section A : instantiate BonusAccessBean

   if (task_output_userId != null)
        this.setInit_argMemberId(newTask_output_userId);

   ////////////////////////////////////
   ```

   Guarde el trabajo.

6. Seleccione el método **populate()** para ver su código fuente. Descomente la Sección 2 para crear una instancia de BonusAccessBean. Esto inserta el código siguiente en el método:

   ```
   setTask_output_oldBonusPoint(getRequestProperties().getString(
       "task_output_oldBonusPoint"));
   ```

Modificar la vía de acceso de clases: Antes de poder probar el mandato modificado en el Entorno de prueba WebSphere, debe modificar una vía de acceso de clases de modo que incluya el nuevo proyecto que se ha creado para el nuevo bean de entidad Bonus. Para modificar esta vía de acceso de clases, haga lo siguiente:

1. En el menú **Área de trabajo** de VisualAge para Java, seleccione **Herramientas > Entorno de prueba WebSphere**.
   Se abre el Centro de control del Entorno de prueba WebSphere.
2. Pulse **Motor de servlets**.
3. Si el motor de servlets está en ejecución, pulse **Detener motor de servlets** y luego **Editar vía de acceso de clases**.
4. Pulse **Seleccionar todo** y luego pulse **Aceptar**.

Modificar la plantilla Sample.jsp para puntos de bonificación: Para modificar la plantilla de visualización, haga lo siguiente:

1. Abra los archivos `Sample.jsp` y `Sample_All.jsp` con un editor de texto.
2. Copie la Sección 4 del código de Sample_All.jsp en Sample.jsp, entre los marcadores `<!-- SECTION 4 -->` y `<!-- END OF SECTION 4 -->`. El siguiente código se inserta en la plantilla JSP

```
<!-- SECTION 4 -->

<h1>
Bonus Administration
</h1>

<%
if (userId != null) {
%>

<B>
<UL>
   <LI> The bonus point before update is
      <%=testBean.getTask_output_oldBonusPoint()%>
   <LI> The bonus point after update is
      <%=testBean.getBonusPoint()%>
</UL>
</B>

<%
}
%>

<br>
<B>Input to command:</B><P>

<FORM NAME=Bonus ACTION="MyNewControllerCmd">
<TABLE>
   <TR>
        <TD>
             <B>Logon ID </B>
         </TD>
        <TD>
             <input type=text name=input1
                 value='<%=testBean.getInput1()%>'>
        </TD>
   </TR>
   <TR>
         <TD>
             <B>Bonus Point</B>
         </TD>
         <TD>
              <input type=text name=input2>
         </TD>
     </TR>
     <TR>
          <TD COLSPAN=2>
               <input type=submit>
```

```
        </TD>
    </TR>
</TABLE>
</FORM>
<!-- END OF SECTION 4 -->
```

Guarde el archivo `Sample.jsp`.

3. Puesto que el nuevo bean Bonus está protegido bajo control de acceso y los usuarios sólo pueden ejecutar la acción MyNewControllerCmd en un bean que posean, el usuario debe conectarse. Por esto, debe utilizar la característica de conexión en su tienda de ejemplo para permitir que los usuarios se conecten. Para ello, tiene que copiar el archivo Sample.jsp en la estructura de directorios de la tienda, ya que una vez se conecta a la tienda, el controlador Web buscará el archivo Sample.jsp en el directorio de la tienda. Copie el archivo Sample.jsp de:
   unidad_vaj`:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web`
   a
   unidad_vaj`:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\` *directorio_tienda*.

   Nota: Para la tienda de ejemplo, el valor de *directorio_tienda* es InFashion.
4. Asegúrese de que el Entorno de prueba WebSphere esté en ejecución (consulte el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281).
5. Conéctese como usuario `wcsadmin`, haciendo lo siguiente:
   - Entre el URL siguiente en un navegador:
     ```
     http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
         storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
     ```
   - Pulse el enlace **Registro**.
     Se muestra la página Regístrese o Conéctese.
   - En el campo **Dirección de correo electrónico**, entre `wcsadmin`.
   - En el campo **Contraseña**, entre la contraseña para el usuario `wcsadmin` y luego pulse **Conexión**.

     Nota: La contraseña original era `wcsadmin` pero se cambió al ejecutar el mandato `contractPublish` como parte del proceso de instalación. Este paso se describe en la siguiente publicación:
     - Business *WebSphere Commerce Studio Business Developer Edition, Guía de instalación*
     - Professional *WebSphere Commerce Studio Professional Developer Edition, Guía de instalación*
6. Después de completar la conexión, entre el URL siguiente en el mismo navegador:
   ```
   http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
       input1=wcsadmin&input2=1000
   ```

   Se visualizará una página que contiene todos los parámetros de salida anteriores así como un nuevo formulario que le permite actualizar el saldo de puntos de bonificación de un usuario. La página que se muestra es parecida a la siguiente captura de pantalla:

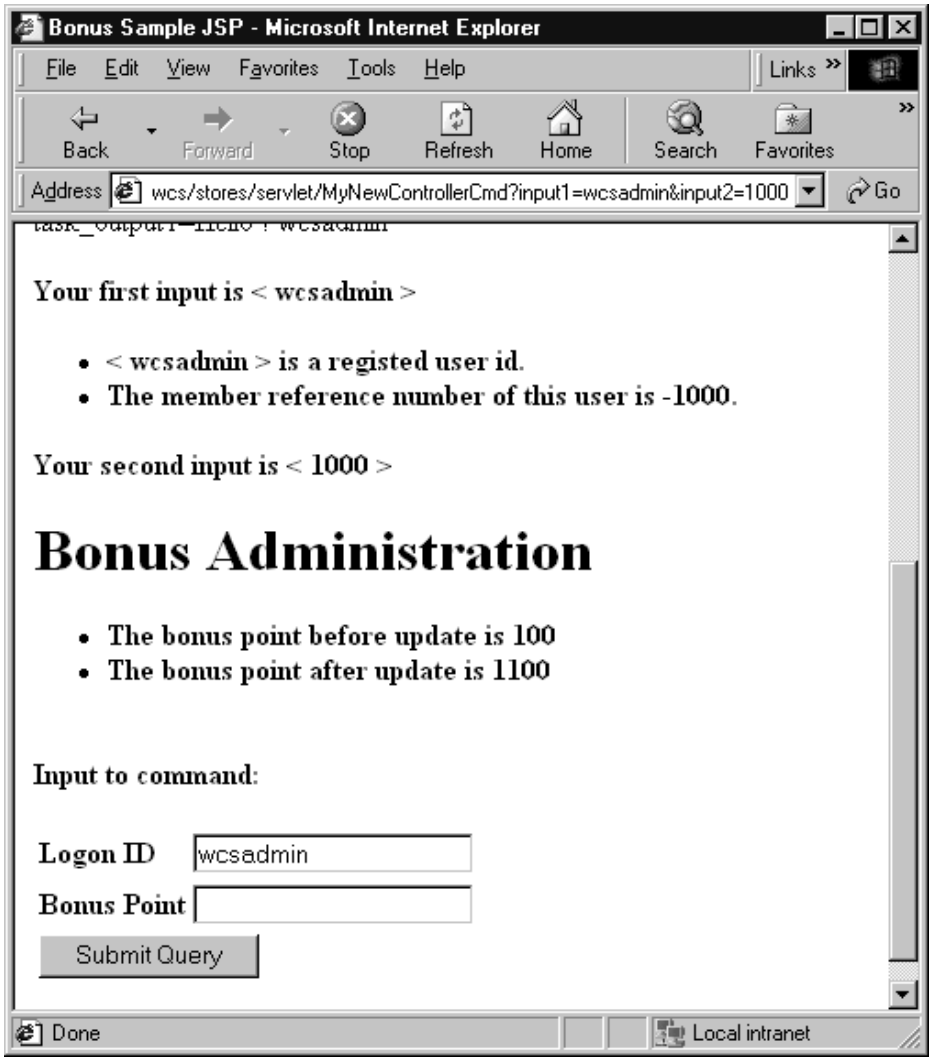

Figura 40.

7. Cree una versión de su código en su estado actual. Denomine esta versión `mySample 1.7 Completed`. Al crear una versión del código, asegúrese de seleccionar sus dos proyectos. Si desea información detallada sobre cómo crear una versión del código, consulte el paso 12 en la página 177.

(Opcional) Utilización del depurador en VisualAge para Java

Esta sección muestra cómo añadir un punto de interrupción en el código y cómo iniciar el componente de depuración de VisualAge para Java. Esta sección se incluye para presentar este componente. Para obtener detalles sobre esta potente característica, consulte la ayuda en línea de VisualAge para Java.

Esta sección es opcional, porque no presenta código nuevo que se necesite en la guía de aprendizaje. En lugar de eso, se crea un punto de interrupción, se comprueban los valores de algunas variables en los puntos de interrupción y se elimina el punto de interrupción.

Adición del punto de interrupción al código

Para añadir el punto de interrupción al código, haga lo siguiente:

1. Asegúrese de que el uso de puntos de interrupción está habilitado en el área de trabajo, haciendo lo siguiente:
   a. En el menú **Ventana**, seleccione **Depurar**. Compruebe que **Habilitación global de puntos de interrupción** esté seleccionado.
2. Seleccione la pestaña **Proyectos**.
3. Expanda el proyecto **_WCSamples**.
4. Expanda el paquete **com.ibm.commerce.sample.commands**.
5. Expanda la clase **MyNewTaskCmdImpl**.
6. Seleccione el método **performExecute**, para ver su código fuente.
7. En el panel Fuente, coloque el cursor (y pulse el botón del ratón) al principio de la siguiente línea de código:
   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```
   Deje el cursor en esta posición.
8. En el menú **Editar**, seleccione **Punto de interrupción**.
9. En la ventana de configuración del punto de interrupción 1, seleccione **En hebra seleccionada** y pulse **Aceptar**.
10. Asegúrese de que el Entorno de prueba WebSphere esté en ejecución (consulte el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281).
11. Conéctese como usuario `wcsadmin`, haciendo lo siguiente:
    - Entre el URL siguiente en un navegador:
      ```
      http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
           storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
      ```
    - Pulse el enlace **Registro**.
      Se muestra la página Regístrese o Conéctese.
    - En el campo **Dirección de correo electrónico**, entre `wcsadmin`.
    - En el campo **Contraseña**, entre la contraseña para el usuario wcsadmin y luego pulse **Conexión**.
12. Después de completar la conexión, entre el URL siguiente:
    ```
    http://localhost:8080/webapp/wcs/stores/servlet/MyNewControllerCmd?
         input1=wcsadmin&input2=1000
    ```
13. Cuando se alcanza el punto de interrupción en el código, se abre la ventana Depurador.

Verificación de los valores de las variables

Esta sección muestra cómo verificar los valores de las variables `task_input1` y `task_output1` en diversos puntos durante la ejecución del mandato.

Cuando se abra el depurador a causa del punto de interrupción en el método performExecute de MyNewTaskCmdImpl, vaya a esa ventana y haga lo siguiente:

1. En el panel Variable, expanda **this**.
2. Pulse **task_input1**.
   En el panel Valor, se muestra el valor para esta variable en este punto durante la ejecución. Debe mostrar `wcsadmin`.

Para verificar que el valor de la variable `task_output1` se establece en el valor esperado, debe hacer que el depurador siga ejecutando el método performExecute hasta alcanzar el punto del código en el que se establece esa variable. Esto puede hacerse de la forma siguiente:

1. Utilice la función de ejecución por pasos para desplazarse por el código paso a paso. Para ello, ejecute una de estas acciones:
   - En el menú **Seleccionado**, seleccione **Recorrer principal**.
   - Pulse F6.
   - Pulse el icono Recorrer principal

   Para este ejemplo, pulse F6 cuatro veces. Esto ejecutará el código que establece la variable task_output1.
2. En el panel Variable, pulse task_output1.
   En el panel Valor, se muestra el valor de esta variable en este punto durante la ejecución. Debe mostrar `Hello ! wcsadmin`.
3. Puede terminar la ejecución del mandato pulsando el icono Reanudar.

Supresión del punto de interrupción

Para eliminar el punto de interrupción del código, haga lo siguiente:

1. Vaya de nuevo a la ventana Entorno de trabajo.
2. Compruebe que puede ver el código fuente del método `performExecute` de la clase `MyNewTaskCmdImpl`.
3. En el panel Fuente, vaya a la siguiente línea de código:
   ```
   setTask_output1( "Hello ! " + getTask_input1() );
   ```

   Observe que hay un punto azul en el margen derecho del panel para marcar el punto de interrupción.
4. Efectúe una doble pulsación en el punto azul para eliminar el punto de interrupción.

Se suprime el punto de interrupción del código.

Integración de MyNewControllerCmd con la tienda de ejemplo en el Entorno de prueba WebSphere

En esta sección, va a añadir un enlace a la página de presentación de la tienda de ejemplo que llama a MyNewControllerCmd. Para realizar este paso de integración, haga lo siguiente:

1. En un editor de texto, abra el archivo `sidebar.jsp`. Este archivo se encuentra en el directorio siguiente:
   unidad_vaj`:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\`*directorio_tienda*`\include`
   donde *unidad_vaj* es la unidad en la que ha instalado VisualAge para Java y *directorio_tienda* es el nombre del directorio para la tienda de ejemplo.
2. Añada otra fila con un enlace a MyNewControllerCmd en la tabla insertando el siguiente código antes del código final `</table>`:
   ```
   <tr>
   <td>
   <a href = "MyNewControllerCmd?input1=wcsadmin&input2=1000">
        MyNewControllerCmd</a>
   </td>
   </tr>
   ```

   Guarde el trabajo.
3. Asegúrese de que el Entorno de prueba WebSphere esté en ejecución.
4. Pruebe la integración entrando el siguiente URL en un navegador:

```
http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
     storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
```

Pulse el enlace **Registro**.
Se muestra la página Regístrese o Conéctese.

5. En el campo **Dirección de correo electrónico**, entre `wcsadmin`.
6. En el campo **Contraseña**, entre la contraseña para el usuario wcsadmin y luego pulse **Conexión**.
7. Después de conectarse, pulse el enlace **MyNewControllerCmd** en el panel lateral de navegación. Se visualiza la JSP de ejemplo.

   Nota: Si el panel de navegación lateral no visualiza el nuevo enlace, es posible que sidebar.jsp esté almacenado en la antememoria. Suprímalo de la antememoria y vuelva a cargar la página. Consulte el Apéndice C, "Consejos para VisualAge para Java" en la página 311 para obtener información sobre la supresión de archivos JSP compilados. Quizá tenga que reiniciar el motor de servlets después de suprimir los archivos JSP compilados.

(Opcional) Despliegue de nueva lógica de negocio en un WebSphere Commerce Server remoto

Esta sección describe cómo desplegar la nueva lógica de negocio en una tienda que se ejecuta en un WebSphere Commerce Server remoto. Antes de efectuar estos pasos de despliegue debe haber creado una tienda (basada en la tienda de ejemplo InFashion) en el WebSphere Commerce Server remoto.

El proceso de despliegue incluye los pasos que se llevan a cabo en la máquina de desarrollo, así como los pasos que se efectúan en el WebSphere Commerce Server de destino.

Creación del archivo JAR para la nueva lógica de mandato

Debe crear un archivo JAR que contenga la nueva lógica de mandato y bean de datos. Para crear este archivo JAR, haga lo siguiente:

1. Detenga el Entorno de prueba WebSphere, tal como se describe en el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281.
2. Con la pestaña Proyectos seleccionada, seleccione el proyecto **_WCSamples**.
3. Con el proyecto resaltado, pulse el botón derecho del ratón y seleccione **Exportar**.
   Se abre el SmartGuide Exportar.
4. Seleccione **Archivo JAR** y pulse **Siguiente.**
5. En el campo del archivo Jar, entre lo siguiente:
   unidad`:\WebSphere\CommerceServerDev\mytemp\wcssamples_1.jar`
   donde *unidad* es la unidad en la que está instalado Commerce Studio.
6. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado

Atributo	Valor
beans	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

Acepte los valores por omisión para los otros atributos.

7. Pulse **Terminar**.
8. Si se le solicita, confirme la creación del nuevo directorio.

Puesto que el archivo JAR creado no contiene toda la información de denominación de paquetes, debe utilizar otro programa de utilidad de empaquetado (fuera de VisualAge para Java) para volver a empaquetar el archivo JAR. Para ello, haga lo siguiente:

1. En una ventana de mandatos, vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServerDev\mytemp
2. Entre `mkdir temp1`.
3. Entre `cd temp1`.
4. Establezca la vía de acceso de la forma siguiente:
   `set PATH=%PATH%;`*unidad*`:\WebSphere\WebSphereStudio4\bin;`
   donde *unidad* es la unidad en la que está instalado WebSphere Studio.
5. Entre `jar xvf ../wcssamples_1.jar`
6. Entre `jar cvf ../wcssamples.jar *` (observe que se ha eliminado _1 del nombre).

Creación del archivo JAR para el nuevo grupo EJB

Debe crear un archivo JAR para el nuevo grupo EJB. Para crear este archivo, haga lo siguiente:

1. Con la pestaña EJB seleccionada, pulse con el botón derecho del ratón en el grupo EJB **WCSSamplesEntityBeans** y seleccione **Exportar > EJB 1.1 JAR**. Se abre el SmartGuide Exportar a un archivo EJB 1.1 JAR.
2. En el campo **Archivo JAR**, entre
   unidad:\WebSphere\CommerceServerDev\mytemp\sampleEntityBeans_DT.jar
3. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
Base de datos de destino	DB2 Si va a desplegar en una base de datos DB2, seleccione **DB2 para NT, V7.1**. Oracle Si va a desplegar en una base de datos Oracle, seleccione **Oracle, V8**.
Incluir atributos de depuración en archivos .class	Seleccionado

Acepte los valores por omisión para los otros atributos.

4. Pulse **Terminar**.

Se crea el archivo JAR.

Al nombre del archivo JAR se le ha añadido el sufijo "_DT" como recordatorio de que debe ejecutar este archivo JAR a través de la herramienta de despliegue de EJB proporcionada con WebSphere Application Server, antes de desplegarlo en su aplicación de WebSphere Commerce.

Creación del archivo JAR de implementación para el nuevo bean enterprise

Debe crear un archivo JAR que contenga el código de implementación para el bean enterprise Bonus. Para crear este archivo, haga lo siguiente:

1. Con la pestaña Proyectos seleccionada, pulse con el botón derecho del ratón en **_WCSSamplesEntityBeansProject** y seleccione **Exportar**.
   Se abre el SmartGuide Exportar.
2. Seleccione **Archivo JAR** y pulse **Siguiente.**
3. En el campo **Archivo JAR**, entre
   unidad:\WebSphere\CommerceServerDev\mytemp\sampleImpl_1.jar
4. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
beans	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
5. Pulse **Terminar**.

Se crea el archivo JAR. Cierre VisualAge para Java.

Puesto que el archivo JAR creado no contiene toda la información de denominación de paquetes, debe utilizar otro programa de utilidad de empaquetado (fuera de VisualAge para Java) para volver a empaquetar el archivo JAR. Para ello, haga lo siguiente:

1. En una ventana de mandatos, vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServerDev\mytemp
2. Entre `mkdir temp2`.
3. Entre `cd temp2`.
4. Establezca la vía de acceso de la forma siguiente:
   `set PATH=%PATH%;`*unidad*`:\WebSphere\WebSphereStudio4\bin;`
   donde *unidad* es la unidad en la que está instalado WebSphere Studio.
5. Entre `jar xvf ../sampleImpl_1.jar`
6. Entre `jar cvf ../sampleImpl.jar *` (observe que se ha eliminado _1 del nombre).

Copia de los archivos JSP en el WebSphere Commerce Server de destino

Debe copiar el archivo `Sample.jsp` y el archivo `sidebar.jsp` actualizado en los directorios correctos para la tienda en la que está desplegando el código.

Antes de copiar las plantillas JSP actualizadas en el WebSphere Commerce Server de destino, quizá desee hacer copias de seguridad de las plantillas JSP originales de esa máquina. En ese caso, cambie el nombre del archivo existente por `sidebar.jsp.bak`.

Para copiar estos archivos, haga lo siguiente:

1. En la máquina de desarrollo, vaya al directorio siguiente:
   unidad_vaj`:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\`*directorio_tienda* donde *unidad_vaj* es la unidad en la que ha instalado VisualAge para Java y *directorio_tienda* es el nombre del directorio de la tienda.
   Copie el archivo `Sample.jsp`.
2. Pegue el archivo `Sample.jsp` en el directorio siguiente del WebSphere Commerce Server de destino:

   unidad`:\WebSphere\AppServer\installedApps\`
   `WC_Enterprise_App_`*nombre_instancia*`.ear\`
   `wcstores.war\`*directorio_tienda*

   donde *unidad* es la unidad en la que está instalado WebSphere Commerce, *directorio_tienda* es el directorio de la tienda y *nombre_instancia* es el nombre de su instancia de WebSphere Commerce.
3. En la máquina de desarrollo, vaya al directorio siguiente:
   unidad_vaj`:\VAJava\Ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\`*directorio_tienda*`\include`
   Copie el archivo `sidebar.jsp`
4. Pegue el archivo `sidebar.jsp` en el directorio siguiente del WebSphere Commerce Server de destino:

   unidad`:\WebSphere\AppServer\installedApps\`
   `WC_Enterprise_App_`*nombre_instancia*`.ear\`
   `wcstores.war\`*directorio_tienda*`\include`

Copia de los archivos JAR en el WebSphere Commerce Server de destino

Debe copiar los archivos JAR de la máquina de desarrollo en el directorio adecuado del WebSphere Commerce Server de destino.

Además, hay dos pasos de proceso adicionales que deben llevarse a cabo en el archivo JAR que contiene el nuevo grupo EJB. En primer lugar, debe ejecutar la herramienta de despliegue de EJB de WebSphere Application Server contra el archivo. A continuación, debe ejecutar el mandato modifyIsolationLevel contra el archivo. Como resultado, en este punto de la copia de los archivos JAR en el WebSphere Commerce Server de destino, este archivo JAR en particular queda almacenado en un directorio temporal en espera de los pasos siguientes.

Para copiar estos archivos, haga lo siguiente:

1. En la máquina de desarrollo, vaya al directorio siguiente:
   unidad`:\WebSphere\CommerceServerDev\mytemp` y localice los archivos siguientes:
   - `wcssamples.jar`

- sampleImpl.jar
- sampleEntityBeans_DT.jar

donde *unidad* es la unidad en la que ha instalado WebSphere Commerce Studio, Business Developer Edition.

Cada uno de los archivos anteriores debe copiarse en un directorio específico del WebSphere Commerce Server de destino. Lea atentamente los siguientes pasos para asegurarse de que cada archivo se almacena en la ubicación correcta.

2. Copie el archivo wcssamples.jar en el siguiente directorio del WebSphere Commerce Server de destino:

   unidad:\WebSphere\AppServer\InstalledApps\
   WC_Enterprise_App_*nombre_instancia*.ear\wcstores.war\WEB-INF\lib

   donde *unidad* es la unidad en la que está instalado WebSphere Commerce Business Edition y *nombre_instancia* es el nombre de la instancia (por ejemplo, demo).
3. Cree un directorio de biblioteca temporal en el que colocará el archivo JAR. Es decir, cree el siguiente directorio:

   unidad:\WebSphere\CommerceServer\temp\lib
4. Copie el archivo sampleImpl.jar en el siguiente directorio del WebSphere Commerce Server de destino:

   unidad:\WebSphere\CommerceServer\temp\lib
5. Copie el archivo sampleEntityBeans_DT.jar en el siguiente directorio del WebSphere Commerce Server de destino:

   unidad:\WebSphere\CommerceServer\temp

Ejecución de la herramienta de despliegue de EJB

Antes de que pueda ejecutar satisfactoriamente sus beans enterprise tanto en un servidor de prueba como en uno de producción, debe generar código de despliegue para los beans enterprise. La herramienta de despliegue de Enterprise JavaBeans (también denominada Herramienta de despliegue de EJB) que se proporciona con WebSphere Application Server, facilita una interfaz de línea de mandatos que puede utilizar para generar código de despliegue de beans enterprise.

Para ejecutar esta herramienta, haga lo siguiente:

1. En un indicador de mandatos, vaya al siguiente directorio:

   unidad:\WebSphere\CommerceServer\temp
2. Añada temporalmente la herramienta a la vía de acceso del sistema entrando el siguiente mandato:

   PATH=*unidad*:\WebSphere\AppServer\deploytool;%PATH%
3. Entre el mandato ejbdeploy de la manera siguiente:

   ejbdeploy *ArchJARGrupoEJB DirTrabajo ArchJARSalida* -nowarn -keep -35 -cp
   ViaClasesDeArchJARDep

   donde:

   - *ArchJARGrupoEJB* es el nombre totalmente calificado del archivo JAR de los beans enterprise para los que desea generar código desplegado. En este caso, es *unidad*:\WebSphere\CommerceServer\temp\sampleEntityBeans_DT.jar.
   - *DirTrabajo* es el nombre del directorio donde se almacenan los archivos temporales necesarios para la generación de código.

- *ArchJARSalida* es el nombre totalmente calificado del archivo JAR de salida. En este caso, entre *unidad*:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar.
- -nowarn es un parámetro opcional para suprimir los mensajes de aviso e información.
- -keep es un parámetro opcional para mantener el directorio de trabajo después de ejecutar el mandato ejbdeploy.
- -35 es un parámetro obligatorio que utilizará las mismas normas de correlación de mayor a menor para los beans de entidad CMP que las que se utilizan en la herramienta de despliegue de EJB que se ha proporcionado con WebSphere Application Server, Versión 3.5.
- -cp *ViaClasesDeArchJARDep* es la vía de clases de los archivos JAR dependientes. En este caso, entre

  ```
  "unidad:\WebSphere\CommerceServer\temp\lib\sampleImpl.jar;
  unidad:\WebSphere\AppServer\installedApps\
     WC_Enterprise_App_nombreInstancia.ear\lib\wcsejbimpl.jar"
  ```

  Nota: Debe poner el valor de la vía de acceso de clases entre comillas (“”).

Modificación del nivel de aislamiento de transacción para los bean de bonificación

En este paso utilizará el mandato modifyIsolationLevel para modificar el nivel de aislamiento de transacción del bean de bonificación. Esta herramienta también establece el nivel de aislamiento de los bean al nivel requerido por su tipo específico de base de datos.

Para ejecutar el mandato modifyIsolationLevel, haga lo siguiente:

1. En el WebSphere Commerce Server de destino, abra una ventana de mandatos.
2. Vaya al directorio siguiente:
   unidad:\WebSphere\CommerceServer\bin
3. Debe emitir el mandato modifyIsolationLevel que tiene la siguiente sintaxis general:

   ```
   modifyIsolationLevel -jarFile arch_jar.jar
      -logFile arch_anotaciones -dbType tipo_bd
   ```

   donde

   - *arch_jar.jar* es el nombre del archivo JAR que contiene el código personalizado
   - *arch_anotaciones* es el nombre totalmente calificado del archivo en el que deben almacenarse las anotaciones cronológicas
   - *tipo_bd* es el tipo de base de datos que utiliza. Entre DB2 u ORACLE

   A continuación se muestra un ejemplo del mandato modifyIsolationLevel con todos los valores especificados:

   ```
   modifyIsolationLevel -jarFile
      D:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar
      -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
      -dbType DB2
   ```

El mandato se ha ejecutado satisfactoriamente si no se muestran excepciones en la ventana de mandatos. Después de la ejecución del mandato, observe que la indicación de la hora del archivo JAR desplegado ha cambiado.

Nota: Los nombres de parámetro son sensibles a las mayúsculas y minúsculas. Es decir, jar**F**ile no es lo mismo que jar**f**ile. Asegúrese de entrar los nombres de parámetro correctamente.

Actualización de la base de datos de destino

Puesto que va a desplegar la nueva lógica de negocio en un WebSphere Commerce Server de destino que utiliza una base de datos diferente de la que utiliza el Entorno de prueba WebSphere, debe actualizar la base de datos de destino, de modo que refleje los cambios realizados en el registro de mandatos y para crear la tabla BONUS.

DB2 Si utiliza una base de datos DB2, haga lo siguiente para actualizar la base de datos de destino:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Centro de mandatos**).
2. En el menú **Herramientas**, seleccione **Valores de herramientas**.
3. Seleccione el recuadro **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (**;**)
4. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla URLREG indicando la siguiente información en la ventana de script:

   ```
   connect to nombre_base_de_datos;
   insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
       DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
       'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
       'Este es un nuevo mandato de controlador con fines de prueba.',
       null)
   ```

   donde *nombre_base_de_datos* es el nombre de su base de datos, y pulse el icono Ejecutar.
   Este mandato lo utilizan todos los comerciantes (indicado por el valor 0 para STOREENT_ID).
5. Cree una entrada en la tabla VIEWREG, entrando lo siguiente en la ventana de script:

   ```
   insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
      CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE) values
      ('SampleViewTask',-1, 0, 'com.ibm.commerce.command.ForwardViewCommand',
      'com.ibm.commerce.command.HttpForwardViewCommandImpl',
      'docname=Sample.jsp','Es una vista de ejemplo para el
      ejercicio de Bonus Point', 0, null)
   ```

   y pulse el icono Ejecutar.
6. Cree la tabla BONUS entrando lo siguiente en la ventana de script:

   ```
   CREATE TABLE Bonus (MEMBERID BIGINT NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (users_id) on delete cascade)
   ```

   Pulse el icono Ejecutar.

Los pasos anteriores registran MyNewControllerCmd y SampleViewTask en el registro de mandatos y también crean la tabla BONUS.

Oracle Si utiliza una base de datos Oracle, haga lo siguiente para actualizar la base de datos de destino:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. Cree la entrada requerida en la tabla URLREG, entrando la siguiente información en la ventana de SQL Plus:

   ```
   insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS,
       DESCRIPTION, AUTHENTICATED) values ('MyNewControllerCmd',0,
       'com.ibm.commerce.sample.commands.MyNewControllerCmd',0,
       'Este es un nuevo mandato de controlador con fines de prueba.',
       null);
   ```

   y pulse Intro para ejecutar la sentencia de SQL.
6. Cree una entrada en la tabla VIEWREG, entrando lo siguiente en la ventana de SQL Plus:

   ```
   insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
      CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE) values
      ('SampleViewTask',-1, 0, 'com.ibm.commerce.command.ForwardViewCommand',
      'com.ibm.commerce.command.HttpForwardViewCommandImpl',
      'docname=Sample.jsp','Es una vista de ejemplo para el
      ejercicio de Bonus Point', 0, null);
   ```

   y pulse Intro para ejecutar la sentencia de SQL.
7. Cree la tabla BONUS entrando lo siguiente en la ventana de SQL Plus:

   ```
   CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (users_id) on delete cascade);
   ```

   y pulse Intro para ejecutar la sentencia de SQL.
8. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia de SQL.

Carga de las políticas de control de acceso para los nuevos recursos

En la guía de aprendizaje, ha creado un nuevo bean enterprise (el bean Bonus) que es un recurso protegible. Por esto, hay una política de control de acceso relacionada con este recurso. También ha creado un nuevo mandato de controlador que pueden ejecutarlo todos los usuarios. Mientras trabajaba en la máquina de desarrollo, ha cargado información de política de control de acceso en esa máquina. Ahora debe cargar la misma información de política de control de acceso en el WebSphere Commerce Server de destino.

Para establecer las políticas de control de acceso, haga lo siguiente:

1. Inserte el siguiente CD en la unidad de CD:
   - Business CD 2 de WebSphere Commerce Business Edition, V5.4
   - Professional CD 2 de WebSphere Commerce Professional Edition, V5.4
2. Vaya al siguiente directorio:

   unidad_CD:\repository\samples\programguide\
3. Localice los siguientes archivos en ese directorio:

- `SampleCmdACPolicy.xml`
  Este archivo XML contiene la política de control de acceso que utiliza el nuevo mandato de controlador.
- `SampleACPolicy.xml`
  Este archivo XML contiene la política de control de acceso que se utiliza al crear un nuevo bean enterprise.
- `SampleACPolicy_`*`entorno_nacional`*`.xml`
  donde *`entorno_nacional`* es el identificador de idioma. Este archivo XML contiene la descripción de la política de control de acceso.

4. Copie los tres archivos anteriores en el siguiente directorio:
   `unidad``:\WebSphere\CommerceServer\xml\policies\xml`
5. Para cargar el archivo `SampleCmdACPolicy.xml`, utilice un indicador de mandatos para ir al siguiente directorio:
   `unidad``:\WebSphere\CommerceServer\bin`
   Debe emitir el mandato `acpload`, que tiene el formato siguiente:
   ```
   acpload nombre_bd usuario_bd contraseña_bd archXMLentrada
   ```
   donde
   - *`nombre_bd`* es el nombre de su base de datos
   - *`usuario_bd`* es el nombre de usuario de la base de datos
   - *`contraseña_bd`* es la contraseña de la base de datos
   - *`archXMLentrada`* es el nombre del archivo XML que contiene la política.

   Por ejemplo, puede emitir el mandato siguiente:
   ```
   acpload mall user password SampleCmdACPolicy.xml
   ```
6. Cargue el archivo `SampleACPolicy.xml`, emitiendo el mandato `acpload` especificando el archivo `SampleACPolicy.xml` como archivo de entrada. Por ejemplo, puede emitir el mandato siguiente:
   ```
   acpload mall user password SampleACPolicy.xml
   ```
7. Para cargar la descripción de la política, debe emitir el mandato `acpnlsload`, que tiene el siguiente formato:
   ```
   acpnlsload
   nombre_bd usuario_bd
   contraseña_bd
   archXMLentrada
   ```
   Por ejemplo, puede emitir el mandato siguiente:
   ```
   acpnlsload mall user password SampleACPolicy_es_ES.xml
   ```

Observe que normalmente será necesario renovar el registro para que la política entre en vigor. En este caso, no es necesario que ejecute este paso ya que detendrá y volverá a iniciar la aplicación WebSphere Commerce Server en WebSphere Application Server, como parte de los pasos de despliegue para el bean enterprise. Si no fuera así, podría utilizar la Consola de administración en WebSphere Commerce para actualizar el registro. Para obtener más información sobre la Consola de administración, consulte la ayuda en línea de WebSphere Commerce.

400 Si despliega en una instancia de WebSphere Commerce en una máquina iSeries, debe usar diferentes mandatos para cargar información de política de control de acceso. Use `LODWCSAC` en lugar de `acpload` y `LODWCSACD` en lugar de `acpnlsload`.

La sintaxis para el mandato `LODWCSAC` es:

```
LODWCSAC DATABASE(nombreBd) SCHEMA(nombreEsquema)
PASSWD(contraseñaInstancia)
INSTROOT('raízInstancia')
INFILE('archivoEntrada')
```

donde

- *nombreBd* es el nombre de la base de datos relacional, tal como se define en el mandato WRKRDBDIRE.
- *nombreEsquema* es el nombre del esquema de la base de datos para la instancia (es el mismo nombre que el de la instancia).
- *contraseñaInstancia* es la contraseña de la instancia.
- *raízInstancia* es el directorio raíz de instancia. Por ejemplo

  `/QIBM/UserData/WebCommerce/instances/nombreInstancia`
- *archivoEntrada* es el nombre totalmente calificado del archivo XML de entrada que tiene las políticas de acceso

La sintaxis para el mandato `LODWCSACD` es:

```
LODWCSACD DATABASE(nombreBd)
SCHEMA(nombreEsquema)
PASSWD(contraseñaInstancia)
INSTROOT('raízInstancia')
INFILE('archivoEntrada')
```

Puede almacenar los archivos XML para sus políticas de control de acceso en el siguiente directorio:

`/QIBM/UserData/WebCommerce/instances/nombreInstancia`

Además, dentro de estos archivos XML, debe utilizar la vía de acceso completa a la DTD de control de acceso. Las DTD para las políticas de control de acceso se almacenan en el directorio `/QIBM/ProdData/WebCommerce/xml/policies/dtd`.

Por ejemplo, si despliega las políticas de control de acceso para las guías de aprendizaje en una instancia de Websphere Commerce que se ejecuta en iSeries, cambie la especificación de DTD en los archivos XML para estas políticas de:

```
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">
```

por

```
<!DOCTYPE Policies SYSTEM "/QIBM/ProdData/WebCommerce/
xml/policies/dtd/accesscontrolpolicies.dtd">
```

Para obtener más información sobre el modelo de control de acceso de WebSphere Commerce, consulte la publicación *WebSphere Commerce, Guía de control de acceso*.

Exportación de la aplicación de empresa actual de WebSphere Application Server

En este paso, exportará la aplicación de empresa actual de WebSphere Application Server de forma que pueda abrirla en la Herramienta de ensamblaje de aplicaciones.

Para exportar la aplicación de empresa actual, haga lo siguiente:

1. Cree el directorio en el que se exportará la aplicación de empresa actual. No denomine a este directorio “temp”, ya que debe evitar el riesgo de que se suprima el archivo durante el mantenimiento rutinario del sistema, hasta que esté seguro de que está satisfecho con la forma en que se comporta el código personalizado, una vez se ha desplegado. Para crear este directorio, haga lo siguiente en un indicador de mandatos:
   a. Vaya al siguiente directorio:
      unidad:\WebSphere\CommerceServer\
   b. Entre el siguiente mandato:
      mkdir working

      Esto crea el directorio *unidad*:\WebSphere\CommerceServer\working.
2. Abra la Consola de administración de WebSphere Application Server.
3. Expanda **Dominio de Administración de WebSphere**.
4. Expanda **Aplicaciones de empresa**.
5. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **demo** y seleccione **Exportar aplicación**.
6. En el campo **Exportar directorio**, entre *unidad*:\WebSphere\CommerceServer\working .
   Esto exportará la aplicación completa, incluidos todos los recursos, al archivo WC_Enterprise_App_*nombreInstancia*.ear (donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce).
7. Pulse **Aceptar**. Este proceso de exportación puede tardar varios minutos.

Exportación de la información de configuración XML para la aplicación de empresa

También debe exportar la información de configuración XML para la aplicación de empresa. Para exportar esta información, utilice el programa de utilidad de línea de mandatos XMLConfig que se proporciona con WebSphere Application Server.

Para exportar esta información de configuración, haga lo siguiente:

1. Copie el archivo was.export.app.xml del directorio siguiente:
   unidad:\WebSphere\CommerceServer\xml\config

   al directorio siguiente:
   unidad:\WebSphere\CommerceServer\working
2. Abra el archivo was.export.app.xml en un editor de texto. En este archivo, sustituya todas las apariciones de $Enterprise_Application_Name$ por
   WebSphere Commerce Enterprise Application - *nombreInstancia*

   donde *nombreInstancia* es el nombre de su instancia de WebSphere Commerce (por ejemplo, demo). Guarde este archivo.

   Nota: El valor que inserte debe coincidir con la información para su instancia que aparece en la Consola de administración del WebSphere avanzado.
3. En un indicador de mandatos, vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServer\working
4. Invoque la herramienta XMLConfig para efectuar una exportación parcial entrando el siguiente mandato:

```
xmlConfig -export OutputFile.xml -partial was.export.app.xml
   -adminNodeName sistpralWas
```

donde *sistpralWas* es el nombre del nodo en el WebSphere Application Server que contiene la aplicación de empresa actual. Además, `OutputFile.xml` es el nombre del archivo que se crea al ejecutar este mandato y `was.export.app.xml` es el archivo que ha modificado en el paso 2.

Después de exportar la información sobre los beans enterprise que están en la aplicación de empresa actual, debe añadir una nueva sección al archivo XML que describa el bean de bonificación.

Para ello, haga lo siguiente:

1. Vaya al siguiente directorio:

   unidad:\WebSphere\CommerceServer\working
2. Abra el archivo `OutputFile.xml` en un editor de texto.
3. Localice el identificador `<ear-file-name>` y sustituya el valor por:

   unidad:\WebSphere\CommerceServer\working\
   WC_Enterprise_App_*nombreInstancia*.ear
4. También debe añadir una nueva sección para el bean de bonificación, tal como se muestra en el siguiente ejemplo:

```
<ejb-module name="WCSSamplesEntityBeans">
   <jar-file>sampleEntityBeans.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         WebSphere Commerce Server - demo/</application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
         <default-user>usuario</default-user>
         <default-password>contraseña</default-password>
      </data-source>
      <enterprise-bean-binding name="Bonus_Binding">
         <jndi-name>democom/ibm/commerce/sample/objects/Bonus</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

   donde

   - *usuario* es el nombre de usuario de la base de datos
   - *contraseña* es la contraseña del usuario de la base de datos.

   Notas:

   a. Los saltos de línea que aparecen en el ejemplo anterior sólo se muestran a efectos de presentación.

   b. Asegúrese de que el valor **$hostName$** coincide con el nombre del servidor de nodos de administración actual. Asegúrese también de que no haya caracteres de retorno de carro en esta línea.

   c. La especificación `<application-server-full-name>` no puede estar en más de una línea.

   d. Si utiliza una base de datos Oracle, debe modificar la información del origen de datos. Cambie la siguiente línea sacada de la sección de código anterior:

      ```
      <jndi-name>jdbc/WebSphere Commerce DB2 DataSource demo</jndi-name>
      ```

por:

```
<jndi-name>jdbc/WebSphere Commerce Oracle DataSource demo</jndi-name>
```

e. Cuando despliegue sus propias aplicaciones (por ejemplo, fuera de esta guía de aprendizaje) asegúrese de que el nombre JNDI para sus beans, que se especifica en el archivo XML, sea el nombre JNDI que se utiliza en VisualAge para Java con el nombre de instancia de WebSphere Commerce añadido al principio.

5. Guarde el archivo `OutputFile.xml`.

Integración del nuevo grupo EJB en la aplicación de empresa

En este paso, abrirá la aplicación de empresa en la herramienta de ensamblaje de aplicaciones. Una vez abierta dentro de esta herramienta, puede hacer lo siguiente para añadir el nuevo bean de bonificación a la aplicación de empresa:

1. Importe el nuevo bean de bonificación. El archivo JAR para el nuevo grupo EJB se almacena dentro de la sección de módulos de EJB de la aplicación de empresa.
2. Establezca la vía de acceso de clases para que el bean de bonificación incluya el archivo JAR de implementación.
3. Añada el archivo JAR de implementación a la aplicación. Este archivo JAR está almacenado dentro de la sección de archivos de la aplicación de empresa.
4. Establezca la seguridad de WebSphere Application Server para los métodos contenidos en el bean de bonificación.

Para integrar el nuevo grupo EJB en la aplicación de empresa, haga lo siguiente:

1. Haga una copia de seguridad de la aplicación de empresa actual, haciendo lo siguiente:
   a. En un indicador de mandatos, vaya al siguiente directorio:

      ```
      unidad:\WebSphere\CommerceServer\working
      ```

   b. Entre el siguiente mandato:

      ```
      copy WC_Enterprise_App_nombreInstancia.ear
         WC_Enterprise_App_nombreInstancia.ear.bak
      ```

2. Abra la Consola de administración de WebSphere Application Server.
3. En el menú **Archivo**, seleccione **Herramientas > Herramienta de ensamblaje de aplicaciones**.
4. Si se abre una ventana de bienvenida, seleccione **Cancelar** para cerrarla.
5. Abra la aplicación de empresa en la que va a trabajar, haciendo lo siguiente:
   a. En el menú **Archivo**, seleccione **Abrir**.
   b. En el campo **Nombre de archivo**, entre:

      ```
      unidad:\WebSphere\CommerceServer\working\
         WC_Enterprise_App_nombreInstancia.ear
      ```

      y pulse **Abrir**. Espere a que la aplicación se abra antes de continuar con los pasos siguientes. Este proceso puede tardar varios minutos.
6. Pulse con el botón derecho del ratón en **Módulos EJB** y seleccione **Importar**.
7. En el campo **Nombre de archivo**, entre

   ```
   unidad:\WebSphere\CommerceServer\temp\sampleEntityBeans.jar
   ```

   y pulse **Abrir**. En la ventana de confirmación, pulse **Aceptar**.

8. Una vez importado el archivo `sampleEntityBeans.jar`, vaya al grupo EJB **WCSSamplesEntityBeans** y seleccione este grupo.
   En el panel derecho se muestra información sobre este grupo.
9. En el campo de la vía de acceso de clases para el nuevo bean enterprise, entre los archivos JAR dependientes. En este caso, entre
   `lib/sampleImpl.jar lib\wcsejbimpl.jar`
10. Pulse **Aplicar**.
11. Añada el archivo `sampleImpl.jar` a la aplicación, haciendo lo siguiente:
    a. Pulse con el botón derecho del ratón en el nodo **Archivos** para la aplicación de empresa y seleccione **Añadir archivos**. El nodo **Archivos** para la aplicación de empresa se encuentra cerca del final del árbol jerárquico. Tenga en cuenta que hay otros nodos Archivos para componentes dentro de la aplicación de empresa, pero debe seleccionar el nodo Archivos para toda la aplicación.)
    b. En la ventana Añadir archivos, pulse **Examinar**.
    c. Vaya al directorio *unidad*`:\WebSphere\CommerceServer\temp`.
    d. Con este directorio resaltado, pulse **Seleccionar**.
    e. Vuelva a la ventana Añadir archivos. Observará que aparece el contenido del directorio*unidad*`:\WebSphere\CommerceServer\temp`. Resalte el directorio `lib`.
       El contenido del directorio `lib` se muestra en el panel de la derecha.
    f. En el panel de la derecha, seleccione el archivo `sampleImpl.jar` y pulse **Añadir**. El archivo aparecerá en el panel Archivos seleccionados.
    g. Pulse **Aceptar**.
12. Configure la seguridad para el bean de bonificación, haciendo lo siguiente:
    a. Con el nodo de módulos EJB expandido, localice y expanda el nodo **WCSSamplesEntityBeans**.
    b. Expanda **Beans de entidad**.
    c. Expanda **Bonus**
    d. Pulse **Extensiones de método** y, a continuación, haga lo siguiente en el panel derecho:
       1) Pulse la pestaña **Avanzadas**.
       2) Compruebe que la opción **Identidad de seguridad** esté seleccionada.
       3) Compruebe que **Utilizar identidad de servidor EJB** esté seleccionado para cada uno de los métodos.
       4) Pulse **Aplicar** (si ha efectuado modificaciones).
    e. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Roles de seguridad** bajo el grupo EJB `WCSamplesEntityBeans` y seleccione **Nuevo**; a continuación, haga lo siguiente:
       1) En el campo **Nombre**, entre `WCSecurityRole` y pulse **Aplicar**. Si este rol ya existe, no es necesario efectuar este paso.
    f. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Permisos de métodos** bajo el grupo EJB `WCSamplesEntityBeans` y seleccione **Nuevo**; a continuación, haga lo siguiente:
       1) En el campo **Nombre del permiso de métodos**, entre `WCMethodPermission`
       2) En el área de selección **Métodos**, pulse **Añadir**.
          Se abre la ventana Añadir métodos.
       3) Expanda **sampleEntityBeans.jar**, luego **Bonus** y finalmente, expanda cada una de las listas **Local** y **Remota** de los métodos.

4) Mantenga pulsada la tecla de desplazamiento y seleccione todos los métodos locales; pulse **Aceptar**.
5) Repita el proceso de selección de métodos para añadir también los métodos remotos, si los hay.
6) En el área de selección de roles, pulse **Añadir**, seleccione `WCSecurityRole` y pulse **Aceptar**.
7) Pulse **Aplicar** para cada actualización.

13. En el menú **Archivo**, seleccione **Guardar**.
14. Cierre la herramienta de ensamblaje de aplicaciones.

Después de completar este paso, ha creado una nueva aplicación de empresa que contiene toda la lógica anterior así como la nueva lógica de negocio. Todo esto se encuentra dentro del archivo modificado recientemente `WC_Enterprise_App_`*`nombreInstancia`*`.ear`.

Importación de la nueva aplicación de empresa a WebSphere Application Server

A continuación se indican los pasos generales que deben realizarse para importar la nueva aplicación de empresa a WebSphere Application Server:

1. Detener la aplicación de empresa que se está ejecutando actualmente en WebSphere Application Server y, a continuación, eliminarla. Estos pasos se llevan a cabo en la Consola de administración de WebSphere Application Server.
2. Importar la nueva aplicación, utilizando el programa de utilidad de línea de mandatos XMLConfig.
3. Renovar la Consola de administración de WebSphere Application Server y luego iniciar la nueva aplicación de empresa.

Cada uno de estos pasos se describe con más detalle en las secciones siguientes.

Detener y eliminar la aplicación de empresa actual

Para detener y eliminar la aplicación de empresa actual de WebSphere Application Server, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**.
3. Expanda **Nodos**.
4. Expanda *nombreNodo* (donde *nombreNodo* es el nombre de su nodo).
5. Expanda **Servidores de aplicaciones**.
6. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en **WebSphere Commerce Server -** *nombreInstancia* y seleccione **Detener**.
7. Expanda **Aplicaciones de empresa**.
8. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Enterprise Application - demo** y seleccione **Detener**.
9. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Enterprise Application - demo** y seleccione **Eliminar**.
10. Cuando se le solicite que indique si la aplicación debe exportarse, seleccione **No**.

Importación de la nueva aplicación de empresa utilizando XMLConfig

Para importar la nueva aplicación de empresa utilizando el programa de utilidad de línea de mandatos XMLConfig, haga lo siguiente:

1. Vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServer\working
2. En el indicador de mandatos, entre el mandato siguiente para importar la aplicación de empresa en WebSphere Application Server:
   ```
   xmlConfig -import OutputFile.xml -adminNodeName nombresistpral_was
   ```
   donde *nombresistpral_was* es el nombre del nodo del WebSphere Application Server que contiene la aplicación actual.

Nota: 400 Si está desplegando en una instancia de WebSphere Commerce que se ejecuta en iSeries, deberá efectuar un paso adicional para modificar los permisos de directorio después de importar la aplicación. Consulte "Importación de una aplicación de empresa" en la página 308 para obtener detalles sobre cómo modificar estos permisos.

Inicio de la nueva aplicación de empresa

Después de importar la nueva aplicación de empresa utilizando el programa de utilidad de línea de mandatos XMLConfig, puede utilizar la Consola de administración de WebSphere Application Server para efectuar una renovación y luego iniciar la nueva aplicación.

Para renovar la consola e iniciar la nueva aplicación, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**
3. Resalte **Nodos**.
4. Pulse el icono **Renovar subárbol seleccionado**.
5. Inicie la aplicación de WebSphere Commerce, haciendo lo siguiente:
   - Expanda **Servidores de aplicaciones**.
   - Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en **WebSphere Commerce Server -** *nombreInstancia* y seleccione **Iniciar**.

Prueba de MyNewControllerCmd

El paso siguiente es probar la nueva lógica en la tienda en ejecución en el entorno de WebSphere Application Server. Para probar MyNewControllerCmd, haga lo siguiente:

1. Pruebe la integración entrando el siguiente URL en un navegador:
   ```
   http://nombresistpral/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
   ```
   donde *ID_tienda* es el identificador de la tienda e *ID_catálogo* es el identificador del catálogo de la tienda.
2. Pulse el enlace **Registro**.
   Se muestra la página Regístrese o Conéctese.
3. En el campo **Dirección de correo electrónico**, entre `wcsadmin`.
4. En el campo **Contraseña**, entre la contraseña para el ID wcsadmin utilizado en este sitio y, a continuación, pulse **Conexión**.

5. Después de conectarse, pulse el enlace **MyNewControllerCmd** en el panel lateral de navegación. Se visualiza la JSP de ejemplo.

Si el archivo `sidebar.jsp` actualizado no aparece, haga lo siguiente para borrar el contenido de la antememoria:

1. Suprima los archivos almacenados en antememoria del directorio siguiente: *unidad*:\WebSphere\CommerceServer\instances*nombreInstancia*\cache
2. Suprima los archivos relacionados con el almacenamiento en antememoria de los directorios siguientes:

   unidad:\WebSphere\AppServer\temp*nombreSistPral*\
   WebSphere_Commerce_Server_*nombreInstancia*\
   WebSphere_Commerce_Enterprise_Application_-_*nombreInstancia*\
   wcstores.war*nombreTienda*

   unidad:\WebSphere\AppServer\temp*nombreSistPral*\
   WebSphere_Commerce_Server_*nombreInstancia*\
   WebSphere_Commerce_Enterprise_Application_-_*nombreInstancia*\
   wcstores.war
3. Borre la antememoria del navegador.

Si la compilación de la página JSP tarda demasiado, es posible que la página no se visualice. En este caso, vuelva a cargar la página.

Capítulo 10. Modificación y ampliación de la lógica de negocio existente

Las siguientes guías de aprendizaje muestran cómo ampliar o modificar lógica de negocio existente de WebSphere Commerce.

Ampliación de un mandato de controlador existente

En esta sección, se ampliará el mandato de controlador OrderProcess existente de modo que el total de puntos de bonificación que se hayan acumulado en la compra se muestren en la página de confirmación de pedido.

Nota: La finalidad de esta guía de aprendizaje es mostrar el proceso de modificación de un mandato de controlador existente. No está pensada para enseñarle la mejor manera de modificar el paso de proceso de pedidos en el flujo de compra. De hecho, WebSphere Commerce proporciona el mandato de tarea `ExtOrderProcess` que se puede utilizar para modificar el paso de proceso de pedidos en el flujo de compra.

Antes de iniciar esta guía de aprendizaje

Ha de haber realizado los pasos del Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" en la página 171.

La lista siguiente resume los pasos necesarios para ampliar el mandato OrderProcess:

1. Crear un paquete nuevo en el que se almacenará el código personalizado. Recordar que se debe almacenar todo el código personalizado (para mandatos y beans de datos) en proyectos y paquetes que están separados del código WebSphere Commerce.
2. Crear una clase `OrderProcessCmdBonusImpl` nueva que amplíe el mandato `OrderProcessCmdImpl` existente.
3. Añadir campos y métodos a la clase `OrderProcessCmdBonusImpl`.
4. Modificar el registro de mandatos para que utilice la clase `OrderProcessCmdBonusImpl`
5. Modificar la plantilla confirmation.jsp para que muestre la lógica de negocio nueva.
6. Comprobar la lógica de negocio nueva en el Entorno de prueba WebSphere.
7. (Opcional) Despliegue de nueva lógica de negocio en una tienda en un WebSphere Commerce Server remoto.

Creación del paquete nuevo para `OrderProcessCmdBonusImpl`

Para crear el paquete nuevo en el que se almacena el mandato `OrderProcessCmdBonusImpl`, haga lo siguiente:

1. En el Entorno de trabajo de VisualAge para Java, compruebe que tiene seleccionada la pestaña Proyectos.
2. Pulse con el botón derecho del ratón en el proyecto **_WCSamples** y seleccione **Añadir > Paquete**.
   Se abrirá el SmartGuide Añadir paquete.

3. Compruebe que el botón de selección **Crear un paquete nuevo llamado** esté habilitado y especifique `com.ibm.commerce.sample.order`.
4. Pulse **Terminar**.

Creación de la clase `OrderProcessCmdBonusImpl`

Para crear la clase `OrderProcessCmdBonusImpl` nueva, haga lo siguiente:

1. Pulse con el botón derecho del ratón en el paquete `com.ibm.commerce.sample.order` y seleccione **Añadir > Clase**.
   Se abrirá el SmartGuide Crear clase.
2. Compruebe que el botón **Crear una clase nueva** esté seleccionado.
3. En el campo **Nombre de clase**, especifique`OrderProcessCmdBonusImpl`.
4. Para especificar la superclase, pulse **Examinar** y luego, en el campo **Patrón**, especifique `com.ibm.commerce.order.commands.OrderProcessCmdImpl` y pulse **Aceptar**.
5. Pulse **Siguiente**.
6. Para especificar los paquetes que deben importarse, pulse **Añadir paquete**. En el campo **Patrón**, entre los paquetes siguientes:
   - `com.ibm.commerce.datatype` y pulse **Añadir**
   - `com.ibm.commerce.exception` y pulse **Añadir**
   - `com.ibm.commerce.order.commands` y pulse **Añadir**
   - `com.ibm.commerce.order.objects` y pulse **Añadir**
   - `com.ibm.commerce.ras` y pulse **Añadir**
   - `com.ibm.commerce.server` y pulse **Añadir**
   - `com.ibm.commerce.sample.objects` y pulse **Añadir**
   - `javax.ejb` y pulse **Añadir**
   - `java.io` y pulse **Añadir**
   - `java.math`, pulse **Añadir** y luego **Cerrar**.
7. Para especificar las interfaces que la clase debe implementar, pulse **Añadir**. En el campo Patrón, entre la interfaz siguiente:
   - `OrderProcessCmd`, pulse **Añadir** y luego **Cerrar**.
8. Pulse **Terminar**.

Adición de campos y métodos a `OrderProcessCmdBonusImpl`

Debe añadir dos campos y un método performExecute a la clase.

Para añadir el campo `theOrder` a la clase `OrderProcessCmdBonusImpl` , haga lo siguiente:

1. Pulse con el botón derecho del ratón en la clase `OrderProcessCmdBonusImpl` y seleccione **Añadir > Campo**.
   Se abre el SmartGuide Crear campo.
2. Añada un campo a la clase, utilizando los atributos siguientes. Para obtener pasos detallados sobre cómo crear un nuevo campo, consulte “Creación de nuevos campos” en la página 181

Nombre de atributo	**Valor**
Nombre de campo	`theOrder`
Tipo de campo	`OrderAccessBean`
Valor inicial	Déjelo en blanco.

Nombre de atributo	Valor
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

y pulse **Terminar**.

Para añadir el campo `bonusPercentAmount` a la clase `OrderProcessCmdBonusImpl`, haga lo siguiente:

1. Pulse con el botón derecho del ratón en la clase `OrderProcessCmdBonusImpl` y vuelva a seleccionar **Añadir > Campo**.
2. Añada un campo a la clase, utilizando los atributos siguientes. Para obtener pasos detallados sobre cómo crear un nuevo campo, consulte "Creación de nuevos campos" en la página 181

Nombre de atributo	Valor
Nombre de campo	`bonusPercentAmount`
Tipo de campo	`double`
Valor inicial	`1000`
Modificadores de acceso	`private`
Otros modificadores	Déjelos sin seleccionar.
Acceso con métodos get y set	`Seleccionado`
Get	`public`
Set	`public`

y pulse **Terminar**.

Para añadir el método `performExecute` a la clase `OrderProcessCmdBonusImpl` , haga lo siguiente:

1. Pulse con el botón derecho del ratón en la clase **`OrderProcessCmdBonusImpl`** y seleccione **Añadir > Método**.
   Se abre el SmartGuide Crear método.
2. Asegúrese de que el botón de selección **Crear un método nuevo** esté seleccionado y pulse **Siguiente**.
3. En el campo **Nombre de método**, entre `performExecute`.
4. En la lista desplegable **Tipo de retorno**, seleccione `void`. Pulse **Siguiente**.
5. Para especificar la excepción que puede generar el método, pulse **Añadir**. En el campo **Patrón**, entre `ECException`, pulse **Añadir** y luego **Cerrar**.
6. Pulse **Finalizar**.
   Se generará el método y se visualizará el código fuente para el método.
7. Debe modificar el código fuente. Localice la siguiente línea en el código fuente del método performExecute:

```
public void performExecute() throws
   com.ibm.commerce.exception.ECException {
```

Después de la línea anterior, entre el código siguiente:

Puede cortar y pegar este código de la versión en PDF de la Guía del programador. Se recomienda copiar inicialmente el código en la ventana del borrador de VisualAge para Java (consulte la ayuda en línea de VisualAge para Java para obtener más información) e inspeccionar el código para asegurarse de que no se ha perdido ni modificado ningún carácter durante la operación de cortar y pegar. Después de validar el código, cópielo en la ubicación destino. Tenga presente que al copiar el texto en otro editor, algunos caracteres pueden modificarse.

```
final String methodName = "performExecute";
ECTrace.entry(ECTraceIdentifiers.COMPONENT_ORDER,
   this.getClass().toString(), methodName);

// efectuar todo el proceso de pedidos de forma normal
super.performExecute();

// *** actualización de la información de puntos de bonificación ***

// obtener información sobre pedidos
theOrder = new OrderAccessBean();
theOrder.setInitKey_orderId(getOrderRn().toString());

int bonusPt;  // puntos de bonificación para este pedido
int bonusTotal;  // puntos de bonificación totales
BigDecimal subtotal;  // subtotal
BigDecimal bonusdeter;  // determinante de bonificación
BigDecimal ans;

// determinar puntos bonif. = subtotal * determinante bonif.
try {
     subtotal = theOrder.getTotalProductPriceInEJBType();
     bonusdeter = new BigDecimal(bonusPercentAmount);
     ans = subtotal.multiply(bonusdeter);
     bonusPt = Math.round(ans.floatValue());

     System.out.println("subtotal is: " + subtotal +
        " deter bonus es: " + bonusdeter + " ans es: " + ans);
     System.out.println("Bonus Percent amount = " +
        bonusPercentAmount);
     System.out.println("Bonus calculated is: "+ bonusPt);
    }


   // Excepciones diversas
   catch (Exception ex) {
   throw new ECSystemException(ECMessage._ERR_GENERIC,
      this.getClass().toString(),methodName,
      ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
   }

//*** Actualización de puntos de bonificación en la tabla BONUS
//    usando el bean creado en el ejemplo anterior ***

BonusAccessBean bonusBean = new BonusAccessBean();
bonusBean.setInit_argMemberId(
   getCommandContext().getUserId().toString());
try {
    // nuevo valor bonif. = puntos bonif. este pedido + puntos antiguos
    bonusTotal = bonusPt + Integer.parseInt(bonusBean.getBonusPoint());
    bonusBean.setBonusPoint(String.valueOf(bonusTotal));
    bonusBean.commitCopyHelper();

    System.out.println("In try, BonusTotal calculated is: "+
           bonusTotal);
```

```
        }

       // Excepciones diversas
      catch (FinderException e) // usuario aún no tiene puntos establecidos
      {
         // crear una fila en la bonificación de tabla
         bonusTotal = bonusPt;

         try {
             BonusAccessBean bonusBeanNew = new
                BonusAccessBean(getCommandContext().getUserId(),
                   new Integer(bonusTotal));

                System.out.println("In catch, BonusTotal calculated is: "+
                   bonusTotal);

      }
      catch (Exception ex) {
         throw new ECSystemException(ECMessage._ERR_GENERIC,
            this.getClass().toString(), methodName,
            ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
         }
   }
      catch (Exception ex) {
          throw new ECSystemException(ECMessage._ERR_GENERIC,
            this.getClass().toString(), methodName,
            ECMessageHelper.generateMsgParms(ex.getMessage()), ex);
      }

      // *** establecer detalles de vista ***

      // Obtener las propiedades de setResponse y añadir los parámetros
     // de bonificación que necesita la página JSP
      TypedProperty resp = getResponseProperties();
      resp.put("bonus", new Integer(bonusPt).toString());
      setResponseProperties(resp);
      ECTrace.exit(ECTraceIdentifiers.COMPONENT_ORDER,
         this.getClass().toString(), methodName);
```

8. Guarde el trabajo.

Modificación del registro de mandatos para utilizar `OrderProcessCmdBonusImpl`

En este ejemplo, querrá utilizar la nueva clase de implementación para procesos siempre que sea necesario el proceso de pedidos. Para poder hacerlo, deberá actualizar el registro de mandatos para asociar la interfaz OrderProcess con la nueva clase de implementación `OrderProcessCmdBonusImpl`.

DB2 Si utiliza una base de datos DB2 nueva, haga lo siguiente para actualizar el registro de mandatos:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Centro de mandatos**).
2. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla CMDREG indicando la siguiente información en la ventana de script:

```
connect to nombre_base_de_datos;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

donde *nombre_base_de_datos* es el nombre de su base de datos, y pulse el icono Ejecutar.
Este mandato lo utilizan todos los comerciantes (indicado por el valor 0 para STOREENT_ID).

Oracle Si utiliza una base de datos Oracle, haga lo siguiente para actualizar el registro de mandatos:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. Cree la entrada requerida en la tabla URLREG, entrando la siguiente información en la ventana de SQL Plus:

   ```
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0;
   ```

   Pulse Intro para ejecutar la sentencia de SQL.
   Este mandato lo utilizan todos los comerciantes (indicado por el valor 0 para STOREENT_ID)
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia de SQL.

Modificación de la plantilla `confirmation.jsp`

Debe modificar la plantilla `confirmation.jsp` de modo que muestre la nueva lógica de negocio que ha añadido al proceso de pedidos del proceso de negocio. Para modificar la plantilla de visualización, haga lo siguiente:

1. Vaya al directorio siguiente:
   `unidad_vaj:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web\`*`directorio_tienda`*.
2. Haga una copia de `confirmation.jsp` y denomínela `confirmation.jsp.bak`
3. Abra `confirmation.jsp` en un editor de texto.
4. Después de las sentencias de importación existentes, entre lo siguiente:

   ```
   <%@  page import="com.ibm.commerce.datatype.*" %>
   ```
5. Después de la línea siguiente en la plantilla JSP:

   ```
   String orderRn = jhelper.getParameter("orderId");
   ```

   añada lo siguiente:

   ```
   String bonus = ((TypedProperty)request.getAttribute(
      ECConstants.EC_REQUESTPROPERTIES)).getString("bonus");
   ```
6. Localice la sección siguiente en la plantilla JSP:

   ```
   <tr>
   <td align="left" valign="middle">
   <font class="product"><%=infashiontext.getString("GRAND_TOTAL")%>
      </font></td>
   <td align="right" valign="middle">
   <font class="strongprice"><%=orderBean.getGrandTotal()  %></font></td>
   ```

   y entonces añada lo siguiente:

```
</tr>
<tr>
<td align="left" valign="middle">
<font class="text">Bonus Points</font></td>
<td align="right" valign="middle">
<font class="strongtext"><%=bonus  %></font></td>
```

7. Guarde el trabajo.

Comprobación de `OrderProcessCmdBonusImpl` en el Entorno de prueba WebSphere

Ahora puede utilizar el Entorno de prueba WebSphere para comprobar la nueva lógica de negocio. Para comprobar el mandato `OrderProcessCmdBonusImpl`, haga lo siguiente:

1. Inicie el Entorno de prueba WebSphere, como se ha descrito en el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281.
2. Abra un navegador y entre el URL de su tienda: Por ejemplo, entre el URL siguiente:

   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=10001&catalogId=10001&langId=-1
   ```

3. Seleccione un producto y cómprelo.
4. Cuando haya comprado el producto, la confirmación del pedido mostrará el número de puntos de bonificación que ha ganado con el pedido.

(Opcional) Despliegue de la lógica de negocio personalizada en un WebSphere Commerce Server remoto

Una vez comprobada por completo la lógica de negocio nueva en el Entorno de prueba WebSphere y si está satisfecho de su código, deberá desplegar el código en una tienda en un WebSphere Commerce Server remoto. En esta guía de aprendizaje, las personalizaciones incluyen lo siguiente:

- La nueva clase OrderProcessCmdBonusImpl.
- El archivo de plantilla confirmation.jsp actualizado.
- El registro de mandatos actualizado.

Como tales, el despliegue de código incluye los pasos siguientes:

1. Creación de un archivo JAR para la lógica de mandatos, con las herramientas de VisualAge para Java.
2. Copia del archivo JAR y de la plantilla JSP en los directorios adecuados del WebSphere Commerce Server de destino.
3. Actualización del registro de mandatos en el servidor WebSphere Commerce Server de destino.

Notas sobre el método de pago de prueba

La tienda de ejemplo que se ejecuta en el Entorno de prueba WebSphere utiliza, por omisión, un método de pago de prueba. Este método se utiliza para que se pueda completar el flujo de compra dentro del Entorno de prueba WebSphere sin necesidad de llamar a Payment Manager. Este método de pago de prueba sólo le permite completar una compra, no permite que los pedidos sometidos con este método estén disponibles para procesos adicionales. Por esta razón, el método de pago de prueba sólo debe utilizarse dentro del Entorno de prueba WebSphere.

Compruebe que puede completar una compra en la tienda en la que está desplegando este código personalizado. El proceso de pago puede llevarse a cabo utilizando un Payment Manager local o remoto.

Para obtener más información sobre el método de pago de prueba, consulte "Método de pago de prueba" en la página 166.

Creación del archivo JAR para la lógica de mandatos

Debe empaquetar la lógica del mandato en un archivo JAR para poder desplegarlo en el WebSphere Commerce Server de destino. Dado que `OrderProcessCmdBonusImpl` se almacena en el proyecto _WCSamples, debe crear un archivo JAR para este proyecto.

Para crear el archivo JAR, haga lo siguiente:

1. Detenga el Entorno de prueba WebSphere, tal como se describe en el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281.
2. Pulse con el botón derecho del ratón en el proyecto **_WCSamples** y seleccione **Exportar**.
   Se abre el SmartGuide Exportar.
3. Seleccione **Archivo JAR** y pulse **Siguiente.**
4. En el campo Archivo jar, entre lo siguiente:
   unidad`:\WebSphere\CommerceServerDev\mytemp_b\wcssamplesb_1.jar`
   donde *unidad* es la unidad en la que se ha instalado Commerce Studio.
5. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
beans	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
6. Pulse **Terminar**.

Puesto que el archivo JAR creado no contiene toda la información de denominación de paquetes, debe utilizar otro programa de utilidad de empaquetado (fuera de VisualAge para Java) para volver a empaquetar el archivo JAR. Para ello, haga lo siguiente:

1. En una ventana de mandatos, vaya al siguiente directorio:
   unidad`:\WebSphere\CommerceServerDev\mytemp_b`
2. Entre `mkdir temp1`.
3. Entre `cd temp1`.
4. Establezca la vía de acceso de la forma siguiente:
   `set PATH=%PATH%;`*unidad*`:\WebSphere\WebSphereStudio4\bin;`
   donde *unidad* es la unidad en la que está instalado WebSphere Studio.
5. Entre `jar xvf ../wcssamplesb_1.jar`
6. Entre `jar cvf ../wcssamplesb.jar *`

Almacenamiento de elementos en el WebSphere Commerce Server de destino

El archivo JAR para la lógica del mandato y la plantilla `confirmation.jsp` modificada se deben colocar en los directorios adecuados del WebSphere Commerce Server de destino.

Para almacenar el archivo JAR en el directorio adecuado del WebSphere Commerce Server remoto, haga lo siguiente:

1. En la máquina de desarrollo, abra una ventana de mandatos y vaya al directorio siguiente:
   unidad:\WebSphere\CommerceServerDev\mytemp_b
   y localice el archivo wcssamplesb.jar.
2. Copie este archivo en el siguiente directorio del WebSphere Commerce Server de destino:

   ```
   unidad:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nombreInstancia.ear\wcstores.war\WEB-INF\lib
   ```

Para almacenar la plantilla confirmation.jsp en el directorio adecuado del WebSphere Commerce Server de destino, haga lo siguiente:

1. En la máquina de desarrollo, vaya al directorio siguiente:
   unidad_vaj:\VAJava\ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*directorio_tienda*
2. Copie la plantilla confirmation.jsp en el directorio siguiente del WebSphere Commerce Server de destino:

   ```
   unidad:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nombreInstancia.ear\wcstores.war\dirTienda
   ```

Actualización del registro de mandatos

Si va a desplegar el mandato OrderProcessCmdBonusImpl en un WebSphere Commerce Server de destino que utiliza una base de datos distinta a la del Entorno de prueba WebSphere, deberá actualizar la base de datos de destino de modo que refleje los cambios que ha realizado en el mandato.

DB2 Si utiliza una base de datos DB2, haga lo siguiente para actualizar la base de datos del WebSphere Commerce Server de destino:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Centro de mandatos**).
2. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla CMDREG indicando la siguiente información en la ventana de script:

   ```
   connect to nombre_bd_destino;
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
   WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
   and storeent_Id=0
   ```

   donde *nombre_bd_destino* es el nombre de su base de datos y pulse el icono Ejecutar

Oracle Si utiliza una base de datos Oracle, haga lo siguiente para actualizar el registro de mandatos:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. Cree la entrada requerida en la tabla CMDREG, entrando la siguiente información en la ventana de SQL Plus:

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.order.OrderProcessCmdBonusImpl'
WHERE INTERFACENAME='com.ibm.commerce.order.commands.OrderProcessCmd'
and storeent_Id=0;
```

Pulse Intro para ejecutar la sentencia de SQL.
Este mandato lo utilizan todos los comerciantes (indicado por el valor 0 para STOREENT_ID)

6. Entre lo siguiente para comprometer los cambios en la base de datos:
   ```
   commit;
   ```
   y pulse Intro para ejecutar la sentencia de SQL.

Reinicio de la aplicación de empresa en WebSphere Application Server

Después de añadir la lógica del mandato a la aplicación de empresa colocando los elementos de archivo en los directorios adecuados y actualizando el registro de mandatos, debe detener y volver a iniciar la aplicación de empresa para que los cambios entren en vigor.

Para detener y reiniciar la aplicación de empresa, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**.
3. Expanda **Nodos**.
4. Expanda ***nombreNodo*** (donde ***nombreNodo*** es el nombre de su nodo).
5. Expanda **Servidores de aplicaciones**.
6. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Server - demo** y seleccione **Detener**.
7. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Server - demo** y seleccione **Iniciar**.

Comprobación de la nueva lógica en InFashion en ejecución en WebSphere Application Server

Ahora puede verificar la nueva lógica de la empresa en la tienda InFashion que se ejecuta en WebSphere Application Server.

Para realizar esta verificación final, haga lo siguiente:

1. Una vez iniciada la instancia de WebSphere Commerce Server, abra un navegador y especifique el URL de la página de presentación de la tienda. Por ejemplo, entre el URL siguiente:
   ```
   http://nombresistpral/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
   ```
   donde *ID_tienda* es el identificador de la tienda e *ID_catálogo* es el identificador del catálogo de la tienda.
2. Seleccione un producto y cómprelo.
3. Cuando haya comprado el producto, la confirmación del pedido mostrará el número de puntos de bonificación que ha ganado con el pedido.

Modificación de un bean de entidad existente y ampliación de un mandato de tarea existente

Nota: La finalidad de esta guía de aprendizaje es mostrar el proceso que se utiliza para modificar beans de entidad existentes y ampliar mandatos de tarea existentes. No está pensada para enseñarle la mejor manera de modificar la fijación de precios de los productos. Para obtener información sobre el descuento en los precios, consulte *WebSphere Commerce Calculation Framework Guide*.

En el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio", se ha creado un conjunto de lógica de negocio totalmente nuevo. Ello incluía la creación de un mandato de controlador, una nueva plantilla JSP, una nueva tabla de base de datos, un nuevo bean enterprise para acceder a la tabla, un bean de acceso correspondiente, así como un bean de acceso a datos. Toda esta lógica se complementa y crea una aplicación de puntos de bonificación simplificada en la que el saldo de puntos de un usuario se puede actualizar utilizando una plantilla JSP que inicia el nuevo mandato del controlador.

La forma en que se utiliza la tabla BONUS en el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio", se muestra en el siguiente diagrama:

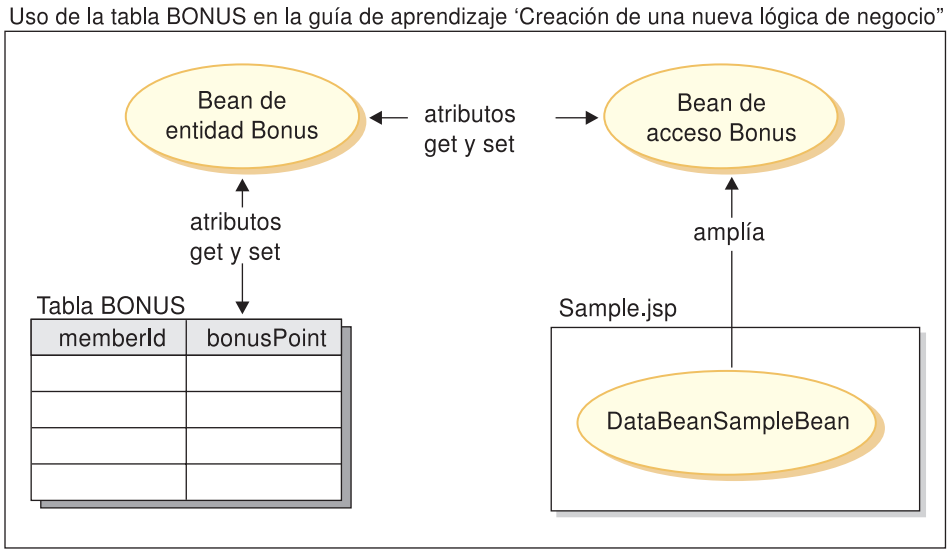

Figura 41.

En la guía de aprendizaje siguiente, la tabla BONUS se utiliza de forma diferente. Específicamente, el bean de entidad User se personaliza de modo que ante las aplicaciones parece que la columna BONUSPOINT de la tabla BONUS es realmente una columna de la tabla USERS. Cuando se crea un nuevo registro en la tabla USERS, automáticamente se inserta un registro correspondiente en la tabla BONUS.

Para crear esta *unión de tablas*, debe añadirse un nuevo campo CMP al bean de entidad User. Este campo CMP se correlaciona con la columna BONUSPOINT de la tabla BONUS utilizando la característica de correlación de tabla secundaria del Examinador de correlaciones de VisualAge para Java.

Para integrar el bean de entidad User en el flujo de compras, se crea un precio nuevo para los productos. Este nuevo precio toma en consideración el saldo actual

de puntos de bonificación del comprador. El nuevo precio se crea ampliando el mandato GetProductContractUnitPriceCmd. Cuando se amplía este mandato, se crea una interfaz nueva que amplía la interfaz GetProductContractUnitPriceCmd. Esta nueva interfaz añade un atributo adicional (para el precio bonificado). También debe crear una nueva clase de implementación que amplíe la clase de implementación GetContractUnitPriceCmdImpl. Esta clase de implementación nueva se denomina "GetNewContractUnitPriceCmdImpl". La nueva clase de implementación llama al método performExecute de su super clase y, a continuación, añade la lógica de negocio para determinar el nuevo precio después de bonificación.

El diagrama siguiente muestra cómo se utiliza la tabla BONUS en la guía de aprendizaje siguiente.

Uso de la tabla BONUS en la guía de aprendizaje 'Personalizar el bean de entidad del usuario"

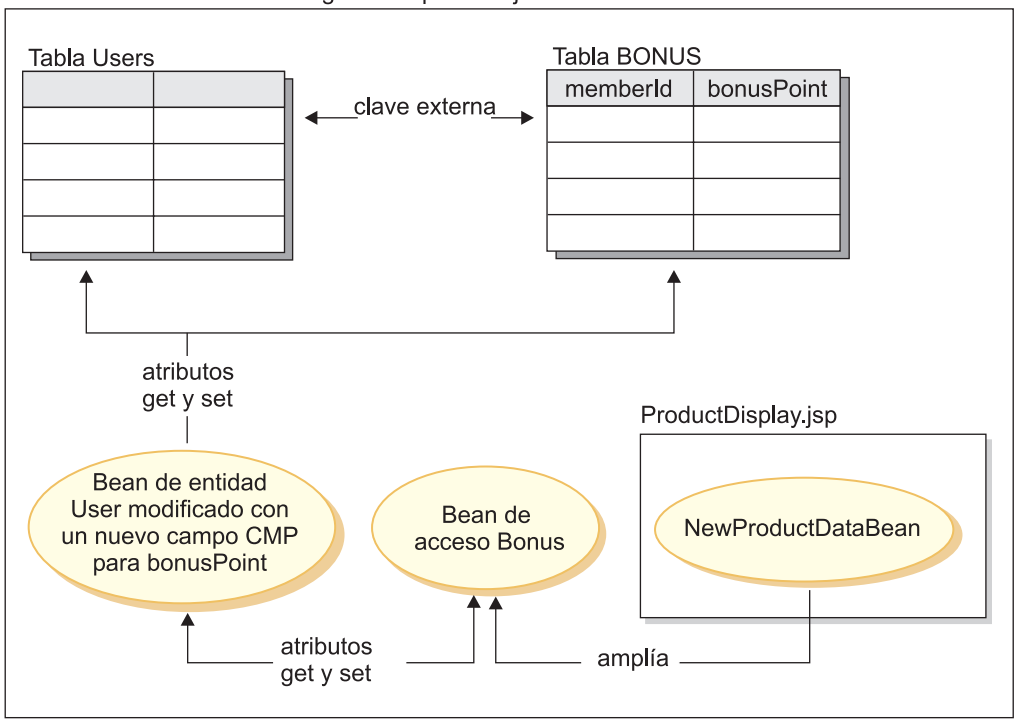

Figura 42.

Esta guía de aprendizaje incluye los pasos siguientes:

1. Añadir un nuevo campo CMP al bean de entidad User.
2. Crear e insertar datos en la tabla BONUS.
3. Actualizar el esquema de base de datos WCSUser y la correlación de tablas de forma que incluya la nueva tabla BONUS.
4. Crear la relación de clave externa entre las tablas BONUS y USER.
5. Crear la correlación de tablas BONUS.
6. Generar el código para desplegar y el bean de acceso para el bean de entidad User.
7. Utilizar el cliente de prueba para las pruebas preliminares del bean de entidad actualizado.
8. Crear una nueva clase de implementación e interfaz de mandatos de tareas. El mandato de tarea nuevo amplía `GetBaseUnitProductPriceCmd`. La característica nueva más importante de este mandato es la lógica del método

performExecute() de la clase de implementación. Este método calcula un nuevo *precio después de bonificación* para el producto. Este precio después de bonificación se crea de forma que el precio disminuya según el saldo de puntos de bonificación del comprador, con una reducción máxima del 20% del precio calculado. La tabla siguiente muestra ejemplos de esta fórmula de reducción de precios:

Precio calculado	Saldo de puntos de bonificación	Deducción máxima (20% del precio calculado)	Nuevo precio después de bonificación
1.000 euros	100	200 euros	900 euros. Puesto que el saldo de puntos de bonificación es menor que la deducción máxima, el precio de catálogo disminuye en el número de puntos de bonificación.
1.000 euros	300	200 euros	800 euros. Puesto que el saldo de puntos de bonificación es mayor que la deducción máxima, el precio de catálogo disminuye en la cifra de deducción máxima de 200 euros.

9. Crear el bean `NewProductDataBean` que amplía `ProductDataBean`. Se añade un nuevo método a este bean de forma que el nuevo precio después de bonificación pueda utilizarse fácilmente en una página de visualización de productos.
10. Actualizar la plantilla JSP de visualización de productos para la tienda InFashion para mostrar el precio después de bonificación.
11. Probar la lógica de negocio en la tienda InFashion ejecutándola dentro del Entorno de prueba WebSphere.
12. (Opcional) Desplegar la lógica de negocio actualizada en el WebSphere Commerce Server remoto y probarla fuera del Entorno de prueba WebSphere.

Antes de iniciar esta guía de aprendizaje

Si no ha completado el Capítulo 9, “Guía de aprendizaje: Creación de nueva lógica de negocio” en la página 171, debe llevar a cabo los pasos descritos en “Preparación del proyecto de ejemplo” en la página 172 antes de iniciar esta guía de aprendizaje.

Adición de un nuevo campo bonusPoint al bean de entidad User

En esta sección, se utilizan las herramientas EJB de VisualAge para Java para añadir el nuevo campo CMP al bean de entidad. El nuevo campo se denomina *bonusPoint* y, eventualmente, se correlacionará con la columna BONUSPOINT de la tabla BONUS.

Para añadir este nuevo campo CMP al bean de entidad User, haga lo siguiente:

1. Si están ejecutándose, detenga el motor de servlets, el servidor EJB y el servidor de nombres persistentes, tal como se describe en el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281.
2. En el Entorno de trabajo, pulse la pestaña **EJB**.
3. Expanda el grupo EJB **WCSUser**.
4. Pulse con el botón derecho del ratón en el bean **User** y seleccione **Añadir > Campo CMP**.
   Se abre el SmartGuide Crear campo CMP.
5. Cree un nuevo campo CMP con las siguientes propiedades:

Propiedad	Valor
Nombre de campo	`bonusPoint`
Tipo de campo	`int`
Valor inicial	`0`
Acceso con métodos get y set	`habilitar`
Promocionar métodos get y set a interfaz remota	`habilitar`
Get	`public`
Set	`public`

   y pulse **Terminar**.

El campo bonusPoint se muestra en el panel Propiedades. Quizá se muestren algunos avisos, pero se solucionan cuando se vuelve a generar el código de bean de entidad.

Creación e inserción de datos en la tabla BONUS

En esta guía de aprendizaje se utiliza una tabla de base de datos que registra los puntos de bonificación de un usuario.

Si ha completado el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" en la página 171, ya está familiarizado con esta tabla. En este caso, deberá actualizar la tabla para añadir una fila para cada usuario de la tabla USERS. Si no ha completado el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" en la página 171, debe crear y rellenar la tabla. Se proporcionan instrucciones para cada uno de estos escenarios.

DB2 Si utiliza una base de datos DB2 y necesita *actualizar* la tabla BONUS, haga lo siguiente:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Centro de mandatos**) y pulse la pestaña **Script**.
2. En el campo **Mandato**, entre
   `connect to nombre_base_de_datos`

   donde *nombre_base_de_datos* es el nombre de su base de datos y pulse el icono Ejecutar.
3. En el campo **Mandato**, entre lo siguiente y, a continuación, pulse el icono Ejecutar:

```
INSERT INTO BONUS
(SELECT USERS_ID, 0
FROM USERS
WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
```

Ahora la tabla BONUS se ha actualizado.

DB2 Si utiliza una base de datos DB2 y necesita *crear e insertar datos* en la tabla BONUS, haga lo siguiente:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Centro de mandatos**) y pulse la pestaña **Script**.
2. En el campo **Mandato**, entre

   ```
   connect to nombre_base_de_datos
   ```

   donde *nombre_base_de_datos* es el nombre de su base de datos y pulse el icono Ejecutar.
3. En el campo **Mandato**, entre lo siguiente y, a continuación, pulse el icono Ejecutar:

   ```
   CREATE TABLE BONUS (MEMBERID BIGINT NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (users_id) on delete cascade)
   ```

   Se ha creado la tabla BONUS.
4. Para insertar datos en la tabla, entre lo siguiente en el campo **Mandato** y, a continuación, pulse el icono Ejecutar:

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```
5. Cierre el Centro de mandatos de DB2.

Oracle Si utiliza una base de datos Oracle y necesita *actualizar* la tabla BONUS, haga lo siguiente:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. Actualice la tabla BONUS, entrando la siguiente información en la ventana de SQL Plus:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

   Pulse Intro para ejecutar la sentencia de SQL.
   La tabla BONUS se ha actualizado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia de SQL.

Oracle Si utiliza una base de datos Oracle y necesita *crear e insertar datos* en la tabla BONUS, haga lo siguiente:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. Cree la tabla BONUS, entrando la siguiente información en la ventana de SQL Plus:

   ```
   CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (users_id) on delete cascade);
   ```

   Pulse Intro para ejecutar la sentencia de SQL.
   La tabla BONUS se ha creado.
6. Inserte datos en la tabla BONUS, entrando lo siguiente:

   ```
   insert into BONUS (select USERS_ID, 0 from USERS);
   ```
7. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia de SQL.

Actualización del esquema y correlación de tablas

En las secciones siguientes, actualizará el esquema WCSUser con la nueva tabla BONUS, creará la relación de clave externa para la nueva tabla y creará una correlación de tablas entre los campos del bean de entidad User y las columnas de la tabla BONUS.

Creación del esquema de la tabla BONUS

Para crear el esquema de la tabla, haga lo siguiente:

1. Pulse con el botón derecho del ratón en el bean de entidad **User** y seleccione **Abrir en > Esquemas de bases de datos**.
   Se abre la ventana Examinador de esquemas.
2. En la lista **Esquemas**, seleccione **Usuario WCS**.
3. En el menú **Tablas**, seleccione **Nueva tabla**.
   Se abre el Editor de tablas.
4. En el campo **Nombre**, entre `BONUS`.
5. Deje los campos **Calificador** y **Nombre físico** en blanco.
6. En la sección Columnas de la tabla, pulse **Nueva**. Se abre el Editor de columnas. Cree una columna, tal como se indica a continuación:

Atributo	**Valor**
Nombre	`memberId`
Nombre físico	Deje este campo en blanco.
Tipo	`BIGINT`
Detalles de tipo	`VapConverter`
Permitir nulos	No seleccionado

   y pulse **Aceptar**.
7. Seleccione **memberId** en la lista de columnas de la tabla y pulse **>>** para utilizar esta columna como clave primaria.

8. Cree otra columna (pulse **Nueva** otra vez), tal como se indica a continuación:

Atributo	Valor
Nombre	`bonusPoint`
Nombre físico	Deje este campo en blanco.
Tipo	`INTEGER`
Detalles de tipo	`VapConverter`
Permitir nulos	Seleccionado

   y pulse **Aceptar**.
9. Pulse **Aceptar** en el Editor de tablas.
   Después de unos instantes, la tabla **BONUS** aparece en la lista **Tablas** de la ventana Examinador de esquemas.
10. Para verificarla, pulse **BONUS** en la lista **Tablas** y compruebe que sus dos columnas aparezcan en la lista **Columnas**.

Creación de la relación de clave externa

En esta sección, establece la relación de clave externa entre las tablas BONUS y USERS.

Para crear la relación de clave externa, haga lo siguiente:

1. En la ventana del Examinador de esquemas, pulse el esquema **Usuario WCS**.
   Se muestran las tablas y las relaciones de clave externa para este esquema.
2. En el menú **Claves externas**, seleccione **Nueva relación de clave externa**.
   Se abre el Editor de relaciones de clave externa.
3. En el campo **Nombre**, entre `F_User_Bonus`.
4. Asegúrese de que el recuadro **Existen restricciones en la base de datos** esté seleccionado.
5. En la lista desplegable **Tabla de clave primaria**, seleccione `USERS`
6. En la lista desplegable **Tabla de clave externa**, seleccione `BONUS`
7. Pulse el campo vacío bajo la cabecera **Clave externa**, de forma que se visualice una lista desplegable. En esta lista, seleccione **memberId** y pulse **Aceptar**.
   Se muestra `F_User_Bonus` en la lista de relaciones de clave externa.
8. En el menú **Esquemas**, seleccione **Guardar esquema** y, a continuación, pulse **Terminar**.
9. Cierre el Examinador de esquemas.

Creación de la correlación de la tabla BONUS

En esta sección, se crea la correlación entre la columna `BONUSPOINT` de la tabla BONUS y el campo `bonusPoint` del bean de entidad User.

Para crear la correlación de la tabla BONUS, haga lo siguiente:

1. Compruebe que el Entorno de trabajo está abierto con la pestaña EJB seleccionada.
2. En el menú **EJB**, seleccione **Abrir en > Correlaciones de esquemas**.
   Se abre el Examinador de correlaciones.
3. En la lista Correlaciones de almacenes de datos, seleccione **Usuario WCS**.
4. En la lista **Clases persistentes**, haga lo siguiente:
   a. Efectúe una doble pulsación en **Miembro**.
   b. Seleccione **User** (Tenga en cuenta que **User** sólo se muestra debajo de **Miembro** después de ampliar Miembro en el paso 4a.)

5. En el menú **Correlaciones de tablas**, seleccione **Nueva correlación de tabla > Añadir correlación de tabla secundaria**.
   Se abre la ventana Correlación de tabla secundaria.
6. En la lista desplegable **Tabla**, seleccione **BONUS**.
7. En la lista desplegable de relaciones de clave foránea, seleccione **F_User_Bonus** y pulse **Aceptar**.
   Después de unos momentos, se muestra la correlación `BONUS (secundaria)` en la lista Correlaciones de tabla.
8. Resalte y pulse con el botón derecho del ratón en la correlación de tabla **BONUS (secundaria)**, y seleccione **Editar correlaciones de propiedades**.
   Se abre el Editor de correlaciones de propiedades.
9. Desplácese hasta la fila que muestra el atributo de clase bonusPoint. Seleccione **bonusPoint**.
10. Pulse en la entrada correspondiente de la columna **Tipo de correlación** y seleccione **Simple** en la lista desplegable.
11. Pulse en la entrada correspondiente en la columna **Tabla** y seleccione **bonusPoint** en la lista desplegable.
12. Deje los demás campos sin cambiar y pulse **Aceptar**.
    Se genera la nueva correlación de propiedades y, al cabo de unos momentos, se visualiza

    ```
    (a) bonusPoint (bonusPoint)
    ```

    en la columna **Correlación de propiedades** del Examinador de correlaciones.
13. Pulse con el botón derecho del ratón en la correlación de almacenes de datos **Usuario WCS**, seleccione **Guardar correlación de almacén de datos** y luego pulse **Finalizar**.
14. Cierre el Examinador de correlaciones.

En este punto, el bean User contiene errores indicando que algunas clases abstractas no se han implementado. Estos errores se arreglan cuando se genera el código desplegado.

Generación del código desplegado y el bean de acceso

Puesto que ha modificado el código del bean de entidad User, debe volver a generar su código desplegado, así como su bean de acceso. Las herramientas en VisualAge para Java hacen que este paso de generación de código sea muy sencillo.

Para efectuar este paso, haga lo siguiente:

1. Compruebe que el Entorno de trabajo está abierto con la pestaña EJB seleccionada.
2. Expanda el grupo EJB **WCSUser**.
3. Pulse con el botón derecho del ratón en el bean **User** y seleccione **Generar código desplegado**.
   Si aparece un mensaje preguntando si desea crear una edición del paquete, pulse **Sí**.
   La generación de código lleva unos minutos.
4. Una vez haya generado el código desplegado, pulse con el botón derecho del ratón en el bean **User** de nuevo y seleccione **Añadir > Bean de acceso**. Se abre el SmartGuide Crear bean de acceso.
5. Acepte los valores por omisión en la página Seleccionar propiedades del bean de acceso y pulse **Siguiente**.

6. Acepte los valores por omisión en la página Definir constructor sin argumentos y pulse **Siguiente**.
7. En la página Seleccionar y personalizar propiedades del bean para CopyHelper, desplácese a **bonusPoint** en la columna Bean Enterprise. Compruebe **Copy Helper** para el bean y establezca el convertidor en `com.ibm.commerce.base.objects.WCSStringConverter`.
8. Pulse **Terminar**.
9. Pulse **Aceptar** cuando se visualiza el mensaje "Generación de código completada".

Comprobación de la modificación utilizando el cliente de prueba

Se puede utilizar un cliente de prueba para probar el bean de entidad User que se acaba de personalizar. Para iniciar el cliente de prueba y probar el bean de entidad, haga lo siguiente:

1. Inicie el servidor de nombres persistentes, tal como se describe en "Inicio y detención del servidor de nombres persistentes" en la página 281.
2. Inicie el servidor EJB que tiene el grupo EJB WCSUser en él, tal como se describe en "Inicio y detención del servidor EJB" en la página 281.
3. En el panel Beans Enterprise, expanda el grupo EJB **WCSUser**. Pulse con el botón derecho del ratón en el bean de entidad **User** y seleccione **Ejecutar cliente de prueba**.
4. En la ventana Búsqueda EJB, pulse **Búsqueda**.
5. En la lista de métodos, seleccione **findByPrimaryKey(MemberKey)**.
6. En el panel de Detalles, pulse **<null>**, luego entre `-1000` en el recuadro de argumentos y pulse el icono Invocar en la ventana del cliente de prueba EJB.

   Observe que el valor entrado aquí debe coincidir con el valor de la columna USERS_ID de la tabla USERS.

   Cuando finalice la ejecución de este método, la información para el usuario con el ID de -1000 se visualiza en una vista de árbol en el panel Métodos. En esta vista, hay un nodo denominado *métodos*.
7. Expanda el nodo **Métodos**.
8. Pulse **getBonusPoint()** y luego el icono Invocar.
   Este método recupera el saldo de puntos de bonificación del cliente. Se devuelve el saldo actual de puntos de bonificación.
9. Pulse **setBonusPoint(int)**, entre `100` en el panel de detalles y pulse el icono Invocar.
10. Pulse **getBonusPoint()** y luego el icono Invocar.
    Se devuelve el valor 100. Esto muestra que el bean enterprise User ha actualizado la base de datos satisfactoriamente.
11. Cierre las ventanas Cliente de prueba User y EJB.

Creación de la interfaz `GetNewProductContractUnitPriceCmd`

Como parte de este ejercicio de personalización, se crea lógica de negocio nueva para calcular un precio de descuento, según el saldo de puntos de bonificación que tiene el cliente. Este calculo lo efectúa un nuevo mandato de tarea. En esta sección, se crea la interfaz para este nuevo mandato de tarea.

Para crear la interfaz del nuevo mandato de tarea, haga lo siguiente:

1. En el Entorno de trabajo, con la pestaña Proyectos seleccionada, expanda el proyecto **_WCSamples**.
2. Pulse con el botón derecho del ratón en el paquete **com.ibm.commerce.sample.commands** y seleccione **Añadir > Interfaz**.
3. Asegúrese de que **Crear una nueva interfaz** esté seleccionada y, en el campo **Nombre de la interfaz**, entre `GetNewProductContractUnitPriceCmd`.
4. Seleccione la interfaz que debe ampliarse pulsando **Añadir** y luego, en el campo **Patrón**, entre `com.ibm.commerce.price.commands.GetProductContractUnitPriceCmd`, pulse **Añadir** y luego **Cerrar**.
5. Pulse **Siguiente**.
6. Para añadir las sentencias de importación correspondientes, pulse **Añadir paquete**; luego, haga lo siguiente:
   a. En el campo **Patrón**, entre `com.ibm.commerce.price.utils` y pulse **Añadir**.
   b. En el campo **Patrón**, entre `com.ibm.commerce.exception` y pulse **Añadir**.
   c. Pulse **Cerrar**.
7. Compruebe que esté seleccionado **Hacer la interfaz public**.
8. Pulse **Terminar**.
   Se muestra el código fuente de la nueva interfaz en el panel Fuente.
9. Cree una firma de método para el método `getBonusPrice()`, haciendo lo siguiente:
   a. Pulse con el botón derecho del ratón en la interfaz **GetNewProductContractUnitPriceCmd** y seleccione **Añadir > Método**. Se abre el SmartGuide Crear método.
   b. Asegúrese de que **Crear un nuevo método** esté seleccionado y pulse **Siguiente**.
   c. En el campo Nombre de método, entre `getBonusPrice`.
   d. Para seleccionar el tipo de retorno del método, pulse **Examinar**. En el campo **Patrón**, entre `MonetaryAmount`, pulse **Aceptar** y luego **Siguiente**.
   e. Para seleccionar la excepción que el método puede generar, pulse **Añadir**. En el campo **Patrón**, entre `ECSystemException`, pulse **Añadir** y luego **Cerrar**.
   f. Pulse **Terminar**.
10. Añada un nuevo campo a la interfaz que especifica la clase de implementación por omisión para el mandato, haciendo lo siguiente:
    a. Pulse con el botón derecho del ratón en la interfaz **GetNewProductContractUnitPriceCmd** y seleccione **Añadir > Campo**. Se abre el SmartGuide Crear campo.
    b. En el campo **Nombre del campo**, especifique `defaultCommandClassName`.
    c. En la lista desplegable **Tipo de campo** seleccione **String**.
    d. En el campo **Valor inicial**, entre

    ```
    "com.ibm.commerce.sample.commands.
       GetNewContractUnitPriceCmdImpl"
    ```

    y pulse **Terminar**.

    Notas:

    1) Debe incluir comillas dobles al entrar este valor.
    2) Si aparece un mensaje de aviso indicándole que un campo en la superclase quedará oculto al crear este nuevo campo, pulse **Sí** para continuar.

11. Añada un nuevo campo a la interfaz que especifique el nombre del mandato, haciendo lo siguiente:
    a. Pulse con el botón derecho del ratón en la interfaz **GetNewProductContractUnitPriceCmd** y seleccione **Añadir > Campo**. Se abre el SmartGuide Crear campo.
    b. En el campo **Nombre del campo**, entre `NAME`.
    c. En la lista desplegable **Tipo de campo** seleccione **String**.
    d. En el campo **Valor inicial**, entre

    ```
    "com.ibm.commerce.sample.commands.
       GetNewProductContractUnitPriceCmd"
    ```

    y pulse **Terminar**.

Creación de la clase de implementación `GetNewContractUnitPriceCmdImpl`

Debe crear la clase de implementación para el nuevo mandato de tarea. Esta clase de implementación implementa la interfaz que acaba de crear y contiene la lógica de negocio para el mandato de tarea.

Para crear la clase de implementación `GetNewContractUnitPriceCmdImpl`, haga lo siguiente:

1. En el Entorno de trabajo, con la pestaña Proyectos seleccionada, expanda el proyecto **_WCSamples**.
2. Pulse con el botón derecho del ratón en el paquete **com.ibm.commerce.sample.commands** y seleccione **Añadir > Clase**. Se abre el SmartGuide Crear clase.
3. Asegúrese de que **Crear una nueva clase** esté seleccionado y cree la clase de la manera siguiente:
    a. En el campo **Nombre de clase**, entre `GetNewContractUnitPriceCmdImpl`.
    b. Para especificar la superclase, pulse **Examinar** y luego, en el campo **Patrón**, entre `com.ibm.commerce.price.commands.GetContractUnitPriceCmdImpl` y pulse **Aceptar**.
    c. Pulse **Siguiente**.
    d. Para especificar los paquetes que deben importarse, pulse **Añadir paquete**. En el campo **Patrón**, entre los paquetes siguientes:
        - `com.ibm.commerce.command` y pulse **Añadir**
        - `com.ibm.commerce.exception` y pulse **Añadir**
        - `com.ibm.commerce.price.commands` y pulse **Añadir**
        - `com.ibm.commerce.price.utils` y pulse **Añadir**
        - `com.ibm.commerce.ras` y pulse **Añadir**
        - `com.ibm.commerce.server` y pulse **Añadir**
        - `java.math`, pulse **Añadir** y luego **Cerrar**.
    e. Para especificar las interfaces que la clase debe implementar, pulse **Añadir**. En el campo Patrón, entre las interfaces siguientes:
        - `GetContractSpecialPriceCmd` y pulse **Añadir**.
        - `GetContractUnitPriceCmd` y pulse **Añadir**.
        - `GetProductContractUnitPriceCmd` y pulse **Añadir**.
        - `GetNewProductContractUnitPriceCmd`, pulse **Añadir** y luego pulse **Cerrar**.
    f. Pulse **Terminar**.

4. Pulse con el botón derecho del ratón en la clase **GetNewContractUnitPriceCmdImpl** y seleccione **Añadir > Campo**. Se abre el SmartGuide Crear campo. Entre la información tal como se indica a continuación:
   a. En el campo **Nombre del campo**, entre `bonusPrice`.
   b. Para seleccionar el tipo de campo, pulse **Examinar**. En el campo Patrón, entre `MonetaryAmount` y pulse **Aceptar**.
   c. Pulse **Terminar**.
5. Añada un nuevo método `performExecute()` a la clase. Este método hace lo siguiente:
   - llama al método `performExecute()` de la superclase (GetContractUnitPriceCmdImpl)
   - establece los valores `thisClass` y `methodName` para que los utilice el mecanismo de manejo de excepciones
   - crea una instancia de StoreAccessBean
   - crea una instancia de UserAccessBean
   - obtiene el precio original del producto
   - calcula el descuento máximo aplicable
   - calcula el nuevo precio después de bonificación (este precio es del tipo *doble*)
   - obtiene el tipo de moneda para la tienda
   - redondea el precio después de bonificación y lo almacena como un importe monetario en la moneda correcta

   Para añadir este método, pulse con el botón derecho del ratón en la clase **GetNewContractUnitPriceCmdImpl** y seleccione **Añadir > Método**. Se abre el SmartGuide Crear método. Entre la información tal como se indica a continuación:
   a. Asegúrese de que **Crear un nuevo método** esté seleccionado y pulse **Siguiente**.
   b. En el campo **Nombre de método**, entre `performExecute`.
   c. En la lista desplegable **Tipo de retorno**, seleccione **void** y pulse **Siguiente**.
   d. Para seleccionar la excepción que el método puede generar, pulse **Añadir**. En el campo **Patrón**, entre `ECException`, pulse **Añadir** y luego **Cerrar**.
   e. Pulse **Terminar**.
   f. Después de esta línea del código fuente:

```
public void performExecute()
   throws com.ibm.commerce.exception.ECException
{
```

   añada el código siguiente:

Puede cortar y pegar este código de la versión en PDF de la Guía del programador. Se recomienda copiar inicialmente el código en la ventana del borrador de VisualAge para Java (consulte la ayuda en línea de VisualAge para Java para obtener más información) e inspeccionar el código para asegurarse de que no se ha perdido ningún carácter durante la operación de cortar y pegar. Después de validar el código, cópielo en la ubicación destino. Tenga presente que al copiar el texto en otro editor, algunos caracteres pueden modificarse.

```
super.performExecute();

// Obtener y establecer este método y nombre de clase
```

```
// para utilizarlo cuando se produzca una excepción.
final String thisClass =
   GetContractUnitPriceCmdImpl.class.getName();
final String methodName = "performExecute";

//obtener el bean de acceso de la tienda
Integer storeId = getStoreId();
com.ibm.commerce.common.objects.StoreAccessBean storeAB =
     getCommandContext().getStore(storeId);

//obtener el bean de acceso del usuario
com.ibm.commerce.user.objects.UserAccessBean bonusAB =
     new com.ibm.commerce.user.objects.UserAccessBean();

// obtener el precio calculado de GetContractUnitPriceCmdImpl
MonetaryAmount priceOrg = super.getPrice();
double dblPriceOrg = priceOrg.getValue().doubleValue();

//calcular la máxima bonificación aplicable a este producto
double dblBonusPrice; // = dblPriceOrg;
double dblMaxBonusPoint = 0;

try {
bonusAB.setInitKey_MemberId(super.getUserId().toString());
bonusAB.refreshCopyHelper();
double dblMaxDed = dblPriceOrg * 0.2;
dblMaxBonusPoint =
     (new java.math.BigDecimal(
        bonusAB.getBonusPoint()).doubleValue());
if (dblMaxBonusPoint > dblMaxDed) dblMaxBonusPoint = dblMaxDed;
} catch (javax.ejb.CreateException ex) {
throw new ECSystemException(
     ECMessage._ERR_CREATE_EXCEPTION, thisClass, methodName, ex);
} catch (javax.ejb.FinderException ex) {

} catch (javax.naming.NamingException ex) {
throw new ECSystemException(
     ECMessage._ERR_GENERIC, thisClass, methodName, ex);
} catch (java.rmi.RemoteException ex) {
throw new ECSystemException(
     ECMessage._ERR_REMOTE_EXCEPTION, thisClass, methodName, ex);
}


//aplicar la máxima bonificación posible a este precio de producto
dblBonusPrice = dblPriceOrg - dblMaxBonusPoint ;

//obtener la moneda de esta tienda
CommandContext context = getCommandContext();
String requestedCurrency = Helper.getCurrency( context, storeAB );

//redondear y devolver el precio con bonificación en tipo MonetoryAmount
bonusPrice = new MonetaryAmount(
     new BigDecimal(dblBonusPrice), requestedCurrency);
CurrencyManager.getInstance().roundCustomized(bonusPrice, storeAB);
```

Guarde el trabajo.

6. Seleccione el método **getBonusPrice()** en la clase **GetNewContractUnitPriceCmdImpl** para ver su código fuente. En el código fuente, cambie

```
return null;
```

por

```
return bonusPrice;
```

Guarde el trabajo.

Creación del bean de datos NewProductDataBean

Debe añadirse el método getCalculatedBonusPrice() al ProductDataBean de WebSphere Commerce existente. Puesto que no debe modificar el código para ProductDataBean, debe crear un nuevo bean de datos que amplíe ProductDataBean y, a continuación, añadir el método al nuevo bean de datos.

Para crear el nuevo bean de datos, haga lo siguiente:

1. En el Entorno de trabajo, con la pestaña Proyectos seleccionada, expanda el proyecto **_WCSamples**.
2. Pulse con el botón derecho del ratón en el paquete **com.ibm.commerce.sample.databeans** y seleccione **Añadir > Clase**. Se abre el SmartGuide Crear clase. Entre la información tal como se indica a continuación:
   a. En el campo **Nombre de clase**, entre NewProductDataBean.
   b. Para especificar la superclase, pulse **Examinar** y luego, en el campo **Patrón**, entre com.ibm.commerce.catalog.beans.ProductDataBean. Pulse **Aceptar** y luego **Siguiente**.
   c. Añada las sentencias de importación correspondientes a la clase pulsando **Añadir paquete**; luego, haga lo siguiente:
      - Entre com.ibm.commerce.beans y pulse **Añadir**.
      - Entre com.ibm.commerce.catalog.objects y pulse **Añadir**.
      - Entre com.ibm.commerce.command y pulse **Añadir**.
      - Entre com.ibm.commerce.datatype y pulse **Añadir**.
      - Entre com.ibm.commerce.exception y pulse **Añadir**.
      - Entre com.ibm.commerce.price.beans y pulse **Añadir**.
      - Entre com.ibm.commerce.ras y pulse **Añadir**.
      - Entre com.ibm.commerce.sample.commands y pulse **Añadir**.
      - Entre com.ibm.commerce.server y pulse **Añadir**.
      - Entre java.util, pulse **Añadir** y luego **Cerrar**.
   d. Pulse **Terminar**.
3. Pulse con el botón derecho del ratón en la clase **NewProductDataBean** y seleccione **Añadir > Método**.
   Se abre el SmartGuide Añadir método.
4. Asegúrese de que **Crear un nuevo método** esté seleccionado y pulse **Siguiente**.
5. En el campo **Nombre de método**, entre getCalculatedBonusPrice.
6. Para seleccionar el tipo de retorno, pulse **Examinar**. En el campo **Patrón**, entre PriceDataBean, pulse **Aceptar** y luego **Siguiente**.
7. Para seleccionar la excepción que el método puede generar, pulse **Añadir**. En el campo **Patrón**, entre ECSystemException, pulse **Añadir** y luego **Cerrar**.
8. Pulse **Terminar**.
9. En el código fuente, sustituya return null; por:

```
PriceDataBean ibnPrice = null;
try {
     GetNewProductContractUnitPriceCmd comm =
        (GetNewProductContractUnitPriceCmd) CommandFactory.createCommand
        (GetNewProductContractUnitPriceCmd.NAME,
             getCommandContext().getStoreId());

     ECTrace.trace(ECTraceIdentifiers.COMPONENT_CATALOG,
     this.getClass().getName(), "getCalculatedBonusPrice",
```

```
        "Getting Price for CatalogEntry: " + getProductID());

        comm.setCatEntryId(new Long(getProductID()));
        comm.setCommandContext(getCommandContext());
        comm.execute();
        ibnPrice = new PriceDataBean(comm.getBonusPrice(),
             getCommandContext().getStore(),
             getCommandContext().getLanguageId());

    } catch (Exception e) {
        throw new ECSystemException(ECMessage._ERR_RETRIEVE_PRICE,
             this.getClass().getName(), "getCalculatedBonusPrice",e);
    }

    return ibnPrice;
```

Guarde el trabajo.

Adición del nuevo precio después de bonificación a la plantilla de visualización de producto

El siguiente paso es añadir el nuevo precio después de bonificación a la plantilla de visualización de producto, para que los compradores puedan ver el precio personalizado. Una vez haya actualizado la plantilla de visualización, se mostrará el nuevo precio con el descuento.

La tienda de ejemplo utiliza la plantilla `ProductDisplay.jsp` para visualizar los productos. Por tanto, debe actualizar esta plantilla con información para visualizar el nuevo precio.

Para actualizar la plantilla de visualización, haga lo siguiente:

1. Vaya al siguiente directorio:
   unidad_vaj:\VAJava\ ide\project_resources\IBM WebSphere Test Environment\hosts\default_host\default_app\web*nombre_tienda*
2. Haga una copia del archivo `ProductDisplay.jsp` y cámbiele el nombre por `ProductDisplay.jsp.bak`.
3. Abra `ProductDisplay.jsp` en un editor de texto.
4. Después de `<%@ page import="com.ibm.commerce.common.beans.*" %>`, añada las siguientes sentencias de importación:
   ```
   <%@ page import="com.ibm.commerce.sample.commands.*" %>
   <%@ page import="com.ibm.commerce.sample.databeans.*" %>
   ```
5. Sustituya todas las apariciones de `ProductDataBean` por `NewProductDataBean`.
6. Localice la línea siguiente:
   ```
   <font class="price"><%=product.getCalculatedContractPrice()%></font>
      <br><br>
   ```
   Después de esta línea, inserte la siguiente línea para que recupere y muestre el precio después de bonificación del producto
   ```
   <font class="price"><%=product.getCalculatedBonusPrice()%>
         Bonus Price </font>
   <br><br>
   ```
7. Guarde este archivo.

Nota: Si efectúa la operación de cortar y pegar secciones en la plantilla de visualización desde la versión en PDF de la publicación *WebSphere Commerce, Guía del programador*, asegúrese de que no se modifica ningún carácter.

Comprobación de la ampliación del bean enterprise

En esta sección, puede probar la ampliación que ha efectuado en el bean enterprise, viendo un producto en la tienda de ejemplo InFashion. Se muestra el nuevo precio después de bonificación.

Tenga en cuenta que para simplificar este ejemplo, el precio de bonificación pueden verlo todos los compradores (los compradores registrados y los compradores invitados). Para los compradores que no tienen puntos de bonificación, el precio después de bonificación es el mismo que el precio normal.

Para probar la ampliación del bean enterprise y ver el precio después de bonificación, haga lo siguiente:

1. Compruebe que el proyecto `_WCSamples` esté incluido en la vía de acceso del motor de servlets, haciendo lo siguiente:
   a. En el menú **Área de trabajo** de VisualAge para Java, seleccione **Herramientas > Entorno de prueba WebSphere**.
   Se abre el Centro de control del Entorno de prueba WebSphere.
   b. Pulse **Motor de servlets**.
   c. Si el motor de servlets está en ejecución, pulse **Detener motor de servlets** y luego **Editar vía de acceso de clases**.
   d. Si todavía no está seleccionado **_WCSamples**, selecciónelo ahora y pulse **Aceptar**.
2. Inicie el Entorno de prueba WebSphere tal como se describe en el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281. El servidor de nombres persistentes y el servidor EJB quizá ya estén ejecutándose; en ese caso, sólo tiene que iniciar el motor de servlets.
3. Abra un navegador y entre el siguiente URL:
   ```
   http://localhost:8080/webapp/wcs/stores/servlet/StoreCatalogDisplay?
       storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
   ```
4. Pulse el enlace **Register** bajo la cabecera Services y, a continuación, pulse **Register** bajo la cabecera New Customer.
   Registre un nuevo cliente utilizando la dirección de correo electrónico `wctester@wc` y la contraseña `wctester1`. Rellene otros campos con valores de prueba y pulse **Someter**. Deje el navegador abierto.
5. DB2 Abra el Centro de mandatos de DB2 y haga lo siguiente:
   a. Pulse la pestaña **Interactivo**
   b. En el campo **Mandato**, haga lo siguiente:
      1) Entre
      ```
      connect to nombre_base_de_datos
      ```
      donde *nombre_base_de_datos* es el nombre de su base de datos de WebSphere Commerce y pulse el icono Ejecutar.
      2) Entre `select users_id from userreg where logonid = 'wctester@wc'` y pulse el icono Ejecutar.
   c. La pestaña con los resultados de la consulta muestra la entrada para el cliente que ha registrado en el paso 4. Anote aquí el valor de USERS_ID del cliente: _____
   d. Actualice el saldo de puntos de bonificación del cliente registrado en los pasos anteriores. Pulse la pestaña Interactivo y, en el campo **Mandato**, entre lo siguiente:
      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = id_usuario
      ```

donde *id_usuario* es el valor del paso 5c. Pulse el icono Ejecutar.

6. Oracle Actualice el saldo de puntos de bonificación del usuario de prueba, haciendo lo siguiente:
   a. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
   b. En el campo **User Name**, entre su nombre de usuario de Oracle.
   c. En el campo **Password**, entre su contraseña de Oracle.
   d. En el campo **Host String**, entre su serie de conexión.
   e. Entre `select users_id from userreg where logonid = 'wctester@wc';`
   f. Se muestra la entrada del cliente que ha registrado en el paso 4. Anote aquí el valor de USERS_ID del cliente: _____
   g. Actualice el saldo de puntos de bonificación del cliente registrado en los pasos anteriores, entrando lo siguiente:

      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = id_usuario;
      ```

      donde *id_usuario* es el valor del paso 6f.
   h. Entre lo siguiente para comprometer los cambios en la base de datos:

      ```
      commit;
      ```

      y pulse Intro para ejecutar la sentencia de SQL.
7. En el navegador, pulse el enlace **Caballero** para ver la sección de ropa de caballero de la tienda.
8. Pulse el enlace de la oferta estelar para ver la página de productos. La página muestra el precio normal y el precio con descuento según el saldo de puntos de bonificación que tiene el cliente.

   Nota: Si se muestra una pila de rastreo en lugar del precio después de bonificación, es posible que necesite inhabilitar el almacenamiento en antememoria. Esto puede llevarse a cabo estableciendo el valor del componente CacheDaemon en false en el archivo *nombre_instancia*.xml, tal como se muestra a continuación:

   ```
   <component compClassName=
         "com.ibm.commerce.cache.daemon.CacheDaemonComponent"
                  enable="false"
                  name="CacheDaemon" />
   ```

   Después de modificar el valor del archivo *nombre_instancia*.xml, debe detener y reiniciar el motor del servlet en el entorno de prueba de WebSphere.

(Opcional) Despliegue de la lógica de negocio personalizada en un WebSphere Commerce Server remoto

Este apartado describe cómo se despliega el bean de entidad modificado y el nuevo mandato de tarea en la tienda que se ejecuta fuera del Entorno de prueba WebSphere.

Para ello, debe crear archivos JAR para la lógica de bean de datos y de mandatos, y los beans enterprise públicos de WebSphere Commerce, colocar los archivos JAR en los directorios correspondientes del servidor de destino, detener la instancia de WebSphere Commerce, modificar las vías de acceso de las clases, desplegar los beans enterprise utilizando el programa de utilidad XMLConfig y volver a iniciar la instancia.

Creación del archivo JAR para el nuevo mandato de precio

Debe crear un archivo JAR para el proyecto **_WCSamples** de modo que se despliegue el nuevo mandato de tarea. Para crear este archivo JAR, haga lo siguiente en la máquina de desarrollo:

1. Detenga el Entorno de prueba WebSphere, tal como se describe en el Apéndice A, "Inicio y detención del Entorno de prueba WebSphere" en la página 281.
2. Con la pestaña Proyectos seleccionada, seleccione el proyecto **_WCSamples**.
3. Con el proyecto resaltado, pulse el botón derecho del ratón y seleccione **Exportar**.
   Se abre el SmartGuide Exportar.
4. Seleccione **Archivo JAR** y pulse **Siguiente.**
5. En el campo **Archivo Jar**, entre lo siguiente:
   unidad:\WebSphere\CommerceServerDev\mytemp_c\wcssamplesc_1.jar
   donde *unidad* es la unidad en la que está instalado WebSphere Commerce.
6. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
beans	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
7. Pulse **Terminar**.

Puesto que el archivo JAR creado no contiene toda la información de denominación de paquetes, debe utilizar otro programa de utilidad de empaquetado (fuera de VisualAge para Java) para volver a empaquetar el archivo JAR. Para ello, haga lo siguiente:

1. En una ventana de mandatos, vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServerDev\ mytemp_c
2. Entre `mkdir temp3`.
3. Entre `cd temp3`.
4. Establezca la vía de acceso de la forma siguiente:
   `set PATH=%PATH%;`*unidad*`:\WebSphere\WebSphereStudio4\bin;`
   donde *unidad* es la unidad en la que está instalado WebSphere Studio.
5. Entre `jar xvf ../wcssamplesc_1.jar`.
6. Entre `jar cvf ../wcssamplesc.jar *` (observe que se ha eliminado _1 del nombre).

Creación de un archivo JAR para el grupo EJB WCSUser

Debe crear un archivo JAR de exportación de EJB 1.1 que contenga el grupo EJB que contiene el bean enterprise modificado. Por tanto, debe seleccionar el grupo siguiente al crear el archivo JAR:

- WCSUser

Para crear el archivo JAR para el grupo EJB WCSUser, haga lo siguiente:

1. Con la pestaña EJB seleccionada, resalte el grupo EJB **WCSUser**.

2. Pulse con el botón derecho del ratón en el grupo EJB **WCSUser** y seleccione **Exportar > EJB 1.1 JAR**.
   Aparece el asistente para exportar un archivo JAR EJB 1.1.
3. En el campo **archivo JAR**, entre
   unidad:\WebSphere\CommerceServerDev\mytemp_c\
   CustomizedWCSUserDeployed_DT.jar
4. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
Base de datos de destino	DB2 Si va a desplegar en una base de datos DB2, seleccione **DB2 para NT, V7.1**. Oracle Si va a desplegar en una base de datos Oracle, seleccione **Oracle, V8**.
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
5. Pulse **Terminar**.

Se crea el archivo JAR.

Al nombre del archivo JAR se le ha añadido el sufijo "_DT" como recordatorio de que debe ejecutar este archivo JAR a través de la herramienta de despliegue de EJB proporcionada con WebSphere Application Server, antes de desplegarlo en su aplicación de WebSphere Commerce.

Creación del archivo wcsejsclient.jar

Para crear el archivo JAR de cliente, haga lo siguiente:

1. Con la pestaña EJB seleccionada, resalte todos los grupos EJB de WebSphere Commerce (los nombres empiezan por WCS). Con todos estos grupos resaltados, pulse el botón derecho y seleccione **Exportar > JAR de cliente**.
   Se abre el SmartGuide Exportar.
2. En el campo **Archivo JAR**, entre
   unidad:\WebSphere\CommerceServerDev\mytemp_c\wcsejsclient.jar
3. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
beans	Seleccionado
class	Seleccionado
java	No seleccionado
resource	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
4. Pulse **Terminar**.

Se crea el archivo JAR.

Copia de la plantilla JSP actualizada en el directorio de la tienda de destino

En "Adición del nuevo precio después de bonificación a la plantilla de visualización de producto" en la página 263, ha actualizado la plantilla de visualización para que reflejara el nuevo precio creado. En este paso copiará la plantilla JSP actualizada de la estructura de directorios del Entorno de prueba WebSphere al directorio utilizado por la tienda cuando se ejecuta fuera del Entorno de prueba WebSphere.

1. En la máquina de desarrollo, vaya al directorio siguiente:
   unidad_vaj:\VAJava\Ide\project_resources\IBM WebSphere Test Environment \hosts\default_host\default_app\web*directorio_tienda*
   donde *unidad_vaj* es la unidad en la que ha instalado VisualAge para Java y *directorio_tienda* es el nombre del directorio de la tienda de ejemplo.
   Copie el archivo ProductDisplay.jsp.
2. Pegue el archivo ProductDisplay.jsp en el siguiente directorio:

   ```
   unidad:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nombre_instancia.ear\
      wcstores.war\directorio_tienda
   ```

   donde *unidad* es la unidad en la que está instalado WebSphere Commerce, *directorio_tienda* es el directorio de la tienda y *nombre_instancia* es el nombre de su instancia de WebSphere Commerce.

Copia de los archivos JAR en el WebSphere Commerce Server de destino

Debe copiar los archivos JAR de la máquina de desarrollo en el directorio adecuado del WebSphere Commerce Server de destino. Para copiar estos archivos, haga lo siguiente:

1. En la máquina de desarrollo, vaya al directorio siguiente:
   unidad:\WebSphere\CommerceServerDev\mytemp_c
   y localice los archivos siguientes:
   - wcssamplesc.jar
   - CustomizedWCSUserDeployed_DT.jar
   - wcsejsclient.jar

   donde *unidad* es la unidad en la que ha instalado WebSphere Commerce Studio, Business Developer Edition.

   Cada uno de los archivos anteriores debe copiarse en un directorio específico del WebSphere Commerce Server de destino. Lea atentamente los siguientes pasos para asegurarse de que cada archivo se almacena en la ubicación correcta.
2. Copie el archivo wcssamplesc.jar en el directorio siguiente del WebSphere Commerce Server de destino:

   ```
   unidad:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nombre_instancia.ear\
      wcstores.war\WEB-INF\lib
   ```

   donde *unidad* es la unidad en la que está instalado WebSphere Commerce Business Edition y *nombre_instancia* es el nombre de la instancia (por ejemplo, demo).
3. Copie el archivo wcsejsclient.jar en el siguiente directorio del WebSphere Commerce Server de destino:

   ```
   unidad:\WebSphere\CommerceServer\temp\lib
   ```

4. Copie el archivo CustomizedWCSUserDeployed_DT.jar en el siguiente directorio del WebSphere Commerce Server de destino:

   unidad:\WebSphere\CommerceServer\temp

Ejecución de la herramienta de despliegue de EJB

Debe ejecutar la herramienta de despliegue de EJB contra el archivo JAR que contiene el nuevo grupo EJB. Esta herramienta se incluye con WebSphere Application Server.

Para ejecutar esta herramienta, haga lo siguiente:

1. En un indicador de mandatos, vaya al siguiente directorio:

   unidad:\WebSphere\CommerceServer\temp

2. Añada temporalmente la herramienta a la vía de acceso del sistema entrando el siguiente mandato:

   PATH=*unidad*:\WebSphere\AppServer\deploytool;%PATH%

3. Entre el mandato ejbdeploy de la manera siguiente:

   ejbdeploy *ArchJARGrupoEJB DirTrabajo ArchJARSalida* -nowarn -keep -35 -cp *ViaClasesDeArchJARDep*

   donde:

   - *ArchJARGrupoEJB* es el nombre del archivo JAR para su grupo EJB. En este caso, es CustomizedWCSUserDeployed_DT.jar.
   - *DirTrabajo* es el directorio de trabajo.
   - *ArchJARSalida* es el nombre del archivo JAR de salida. En este caso, entre CustomizedWCSUserDeployed.jar.
   - -nowarn es un parámetro opcional para suprimir los mensajes de aviso e información.
   - -keep es un parámetro opcional para mantener el directorio de trabajo después de ejecutar el mandato ejbdeploy.
   - -35 es un parámetro obligatorio que utilizará las mismas normas de correlación de mayor a menor para los beans de entidad CMP que las que se utilizan en la herramienta de despliegue de EJB que se ha proporcionado con WebSphere Application Server, Versión 3.5.
   - -cp *ViaClasesDeArchJARDep* es la vía de clases de los archivos JAR dependientes. Cuando modifica un bean enterprise de WebSphere Commerce existente, debe incluir los archivos wcsejsclient.jar, wcsejbimpl.jar y xml4j.jar en la vía de acceso de clases de los archivos JAR dependientes. Para ello, entre lo siguiente:

     "*unidad*:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
     unidad:\WebSphere\AppServer\InstalledApps\
     WC_Enterprise_App_*nombreInstancia*.ear\lib\wcsejbimpl.jar;
     unidad:\WebSphere\AppServer\InstalledApps\
     WC_Enterprise_App_*nombreInstancia*.ear\lib\xml4j.jar;"

Modificación del nivel de aislamiento de transacción para los beans de entidad

En este paso utilizará el mandato modifyIsolationLevel para modificar el nivel de aislamiento de transacción de los beans de entidad al nivel requerido por su tipo específico de base de datos.

Para ejecutar el mandato modifyIsolationLevel, haga lo siguiente:

1. En el WebSphere Commerce Server de destino, utilice un indicador de mandatos para navegar al directorio siguiente:

   unidad:\WebSphere\CommerceServer\bin

2. Debe emitir el mandato `modifyIsolationLevel` que tiene la siguiente sintaxis general:

   ```
   modifyIsolationLevel -jarFile arch_jar.jar
      -logFile arch_anotaciones -dbType tipo_bd
   ```

   donde

   - *arch_jar.jar* es el nombre del archivo JAR que contiene el código personalizado
   - *arch_anotaciones* es el nombre totalmente calificado del archivo en el que deben almacenarse las anotaciones cronológicas
   - *tipo_bd* es el tipo de base de datos que utiliza. Entre DB2 u ORACLE

   A continuación se muestra un ejemplo del mandato `modifyIsolationLevel` con todos los valores especificados:

   ```
   modifyIsolationLevel -jarFile
      D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
      -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
      -dbType DB2
   ```

Nota: Los nombres de parámetro son sensibles a las mayúsculas y minúsculas. Es decir, `jarFile` no es lo mismo que `jarfile`. Asegúrese de entrar los nombres de parámetro correctamente.

El mandato se ha ejecutado satisfactoriamente si no se muestran excepciones en la ventana de mandatos. Después de la ejecución del mandato, observe que la indicación de la hora del archivo JAR desplegado ha cambiado.

Actualización de la base de datos de destino

Si va a desplegar en un WebSphere Commerce Server de destino que utiliza una base de datos distinta a la que utiliza el Entorno de prueba WebSphere, deberá actualizar la base de datos de destino, de la forma siguiente:

- Si ha completado el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" en la página 171, debe actualizar la tabla para añadir una fila para cada usuario en la tabla USERS.
- Si no ha completado el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" en la página 171, debe crear y rellenar la tabla.

Si no ha completado el Capítulo 9, "Guía de aprendizaje: Creación de nueva lógica de negocio" en la página 171, debe crear y rellenar la tabla. Se proporcionan instrucciones para cada uno de estos escenarios.

DB2 Si utiliza una base de datos DB2 y necesita *actualizar* la tabla BONUS, haga lo siguiente:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Centro de mandatos**) y pulse la pestaña **Script**.
2. En el campo **Script**, entre

   ```
   connect to nombre_base_de_datos
   ```

   donde *nombre_base_de_datos* es el nombre de su base de datos y pulse el icono Ejecutar.
3. En el campo **Script**, entre lo siguiente y, a continuación, pulse el icono Ejecutar:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS))
   ```

Ahora la tabla BONUS se ha actualizado.

DB2 Si utiliza una base de datos DB2 y necesita *crear e insertar datos* en la tabla BONUS, haga lo siguiente:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Centro de mandatos**) y pulse la pestaña **Script**.
2. En el campo **Script**, entre

   ```
   connect to nombre_base_de_datos
   ```

   donde *nombre_base_de_datos* es el nombre de su base de datos y pulse el icono Ejecutar.
3. En el campo **Script**, entre lo siguiente y, a continuación, pulse el icono Ejecutar:

   ```
   CREATE TABLE BONUS (MEMBERID BIGINT NOT NULL,
       BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
       constraint f_memberid foreign key (MEMBERID)
       references users (users_id) on delete cascade)
   ```

   Se ha creado la tabla BONUS.
4. Para insertar datos en la tabla, entre lo siguiente en el campo **Script** y, a continuación, pulse el icono Ejecutar:

   ```
   insert into BONUS (select USERS_ID, 0 from USERS)
   ```
5. Cierre el Centro de mandatos de DB2.

Oracle Si utiliza una base de datos Oracle y necesita *actualizar* la tabla BONUS, haga lo siguiente:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.
5. Actualice la tabla BONUS, entrando la siguiente información en la ventana de SQL Plus:

   ```
   INSERT INTO BONUS
   (SELECT USERS_ID, 0
   FROM USERS
   WHERE USERS_ID NOT IN (SELECT MEMBERID FROM BONUS));
   ```

   Pulse Intro para ejecutar la sentencia de SQL.
   La tabla BONUS se ha actualizado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia de SQL.

Oracle Si utiliza una base de datos Oracle y necesita *crear e insertar datos* en la tabla BONUS, haga lo siguiente:

1. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
2. En el campo **User Name**, entre su nombre de usuario de Oracle.
3. En el campo **Password**, entre su contraseña de Oracle.
4. En el campo **Host String**, entre su serie de conexión.

5. Cree la tabla BONUS, entrando la siguiente información en la ventana de SQL Plus:

```
CREATE TABLE Bonus (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_memberid primary key (MEMBERID),
    constraint f_memberid foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   Pulse Intro para ejecutar la sentencia de SQL.
   La tabla BONUS se ha creado.
6. Inserte datos en la tabla BONUS, entrando lo siguiente:

```
insert into BONUS (select USERS_ID, 0 from USERS);
```

7. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia de SQL.

Exportación de la aplicación de empresa actual de WebSphere Application Server

En este paso, exportará la aplicación de empresa actual de WebSphere Application Server de forma que, posteriormente, pueda abrirla en la Herramienta de ensamblaje de aplicaciones.

Para exportar la aplicación de empresa actual de WebSphere Application Server, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**.
3. Expanda **Aplicación de empresa**.
4. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, expanda la aplicación **demo** y seleccione **Exportar aplicación**.
5. En el campo **Exportar directorio**, entre
   unidad:\WebSphere\CommerceServer\working .
   Esto exportará la aplicación completa, incluidos todos los recursos, al archivo WC_Enterprise_App_*nombreInstancia*.ear (donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce).

   Nota: Si ya tiene un archivo WC_Enterprise_App_*nombreInstancia*.ear, puede renombrar el archivo antiguo o sobregrabarlo.

Exportación de la información de configuración XML para la aplicación de empresa

También debe exportar la información de configuración XML para la aplicación de empresa. Para exportar esta información, utilice el programa de utilidad de línea de mandatos XMLConfig que se proporciona con WebSphere Application Server.

Para exportar esta información de configuración, haga lo siguiente:

1. En un indicador de mandatos, vaya al siguiente directorio:

```
unidad:\WebSphere\CommerceServer\working
```

2. Invoque la herramienta XMLConfig para efectuar una exportación parcial entrando el siguiente mandato:

```
xmlConfig -export OutputFileB.xml -partial was.export.app.xml
   -adminNodeName sistpralWas
```

donde *sistpralWas* es el nombre del nodo en el WebSphere Application Server que contiene la aplicación de empresa actual. Además, OutputFileB.xml es el nombre del archivo que se crea al ejecutar este mandato.

Después de exportar la información sobre los beans enterprise en la aplicación de empresa actual, debe actualizar el archivo OutputFileB.xml de forma que dirija al archivo JAR que contiene el código para el bean User modificado.

Para actualizar el archivo OutputFileB.xml, haga lo siguiente:

1. Vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServer\working
2. Abra el archivo OutputFileB.xml en un editor de texto.
3. Localice el identificador <ear-file-name> y sustituya el valor por:
   unidad:\WebSphere\CommerceServer\working\
   WC_Enterprise_App_*nombreInstancia*.ear
4. Localice la sección <ejb-module> para el grupo EJB WCSUser y cambie el valor que contienen los códigos <jar-file> por CustomizedWCSUserDeployed.jar
5. Guarde el archivo OutputFileB.xml.

Integración del grupo EJB modificado en la aplicación de empresa

En este paso, abrirá la aplicación de empresa en la herramienta de ensamblaje de aplicaciones. Una vez abierta, puede hacer lo siguiente para incluir el bean de usuario modificado en la aplicación de empresa:

1. Haga una copia de la vía de acceso de clases para la versión existente del grupo EJB WCSUser.
2. Elimine la versión existente del grupo EJB WCSUser.
3. Importe la nueva versión del grupo EJB WCSUser. El archivo JAR para el nuevo grupo EJB se almacena dentro de la sección de módulos de EJB de la aplicación de empresa.
4. Establezca la vía de acceso de clases para el grupo EJB WCSUser.

Para integrar el nuevo grupo EJB en la aplicación de empresa, haga lo siguiente:

1. Haga una copia de seguridad de la aplicación de empresa actual, haciendo lo siguiente:
   a. En un indicador de mandatos, vaya al siguiente directorio:
      unidad:\WebSphere\CommerceServer\working
   b. Entre el siguiente mandato:
      copy WC_Enterprise_App_*nombreInstancia*.ear
      WC_Enterprise_App_*nombreInstancia*.ear.bak2
2. Abra la Consola de administración de WebSphere Application Server.
3. En el menú **Herramientas**, seleccione **Herramienta de ensamblaje de aplicaciones**. (Si se abre una ventana de bienvenida, seleccione **Cancelar** para acceder a la consola.)
4. Abra la aplicación de empresa en la que va a trabajar, haciendo lo siguiente:
   a. En el menú **Archivo**, seleccione **Abrir**.
   b. En el campo **Nombre de archivo**, entre:
      unidad:\WebSphere\CommerceServer\working\
      WC_Enterprise_App_*nombreInstancia*.ear

      y pulse **Abrir**. Espere a que la aplicación se abra antes de continuar con los pasos siguientes. Este proceso puede tardar varios minutos.

5. Pulse **Módulos EJB**. El panel derecho muestra los módulos EJB en la aplicación de empresa.
6. Pulse el módulo EJB **WCSUser**.
7. Pulse la pestaña General para ver la información de vía de acceso de clases para el módulo EJB WCSUser existente. Copie esta información de vía de acceso de clases en un archivo de texto (por ejemplo, `WCSUser_path.txt`).
8. Pulse con el botón derecho del ratón en el módulo EJB **WCSUser** y seleccione **Suprimir**.
9. Pulse con el botón derecho del ratón en **Módulos EJB** y seleccione **Importar**.
10. En el campo **Nombre de archivo**, entre
    unidad`:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar`

    y pulse **Abrir**. En la ventana de confirmación, pulse **Aceptar**.
11. Una vez importado el archivo `CustomizedWCSUserDeployed.jar`, vaya al grupo EJB **WCSUser** y seleccione este grupo.
    En el panel derecho se muestra información sobre este grupo.
12. Abra el archivo de texto que contiene la información de vía de acceso de clases para la versión anterior del grupo EJB WCSUser. Seleccione y copie la vía de acceso de clases.
13. En el campo de la vía de acceso de clases para el nuevo grupo EJB WCSUser, pegue esta información de vía de acceso.
14. Pulse **Aplicar**.
15. En el menú **Archivo**, seleccione **Cerrar**.
16. Espere a que el archivo se cierre y, a continuación, en el menú **Archivo**, seleccione **Abrir** y vuelva a abrir el archivo
    unidad`:\WebSphere\CommerceServer\working\`
    `WC_Enterprise_App_`*nombreInstancia*`.ear`.
17. Configure la seguridad para el bean de usuario, haciendo lo siguiente:
    a. Con el nodo de módulos EJB expandido, localice y expanda el nodo **WCSUser**.
    b. Expanda **Beans de entidad**.
    c. Expanda **User**
    d. Pulse **Extensiones de método** y, a continuación, haga lo siguiente en el panel derecho:
       1) Pulse la pestaña **Avanzadas**.
       2) Compruebe que la opción **Identidad de seguridad** esté seleccionada.
       3) Compruebe que **Utilizar identidad de servidor EJB** esté seleccionado para cada uno de los métodos.
       4) Pulse **Aplicar** (si ha efectuado modificaciones).
    e. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Roles de seguridad** bajo el grupo EJB `WCSUser` y seleccione **Nuevo**; a continuación, haga lo siguiente:
       1) En el campo **Nombre**, entre `WCSecurityRole` y pulse **Aplicar**. Si este rol ya existe, no es necesario efectuar este paso.
    f. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Permisos de métodos** bajo el grupo EJB `WCSUser` y seleccione **Nuevo**; a continuación, haga lo siguiente:
       1) En el campo **Nombre del permiso de métodos**, entre `WCMethodPermission`

2) En el área de selección **Métodos**, pulse **Añadir**.
   Se abre la ventana Añadir métodos.
3) Expanda **CustomizedWCSUserDeployed.jar** y seleccione todos los beans enterprise (mantenga pulsada la tecla de desplazamiento mientras efectúa la selección). Pulse **Aceptar**. A continuación, bajo la columna Bean enterprise aparecen todos los beans enterprise y bajo la columna Tipos aparece **Todos los métodos**.
4) En el área de selección de roles, pulse **Añadir**, seleccione `WCSecurityRole` y pulse **Aceptar**.
5) Pulse **Aplicar** y, a continuación, **Aceptar**.

18. En el menú **Archivo**, seleccione **Guardar**.
19. Cierre la herramienta de ensamblaje de aplicaciones.

Después de completar este paso, ha creado una nueva aplicación de empresa que contiene toda la lógica anterior así como la nueva lógica de negocio. Todo esto se encuentra dentro del archivo modificado recientemente `WC_Enterprise_App_`*`nombreInstancia`*`.ear`.

Importación de la nueva aplicación de empresa a WebSphere Application Server

A continuación se indican los pasos generales que deben realizarse para importar la nueva aplicación de empresa a WebSphere Application Server:

1. Detener la aplicación de empresa que se está ejecutando actualmente en WebSphere Application Server y, a continuación, eliminarla. Estos pasos se llevan a cabo en la Consola de administración de WebSphere Application Server.
2. Importar la nueva aplicación, utilizando el programa de utilidad de línea de mandatos XMLConfig.
3. Renovar la Consola de administración de WebSphere Application Server y luego iniciar la nueva aplicación de empresa.

Cada uno de estos pasos se describe con más detalle en las secciones siguientes.

Detener y eliminar la aplicación de empresa actual: Para detener y eliminar la aplicación de empresa actual de WebSphere Application Server, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**.
3. Expanda **Nodos**.
4. Expanda *nombreNodo* (donde *nombreNodo* es el nombre de su nodo).
5. Expanda **Servidores de aplicaciones**.
6. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en **WebSphere Commerce Server -** *nombreInstancia* y seleccione **Detener**.
7. Expanda **Aplicaciones de empresa**.
8. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Enterprise Server - demo** y seleccione **Detener**.
9. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Enterprise Server - demo** y seleccione **Eliminar**.
10. Cuando se le solicite que indique si la aplicación debe exportarse, seleccione **No**.

Importación de la nueva aplicación de empresa utilizando XMLConfig: Para importar la nueva aplicación de empresa utilizando el programa de utilidad de línea de mandatos XMLConfig, haga lo siguiente:

1. Vaya al siguiente directorio:
   unidad:\WebSphere\CommerceServer\working
2. En el indicador de mandatos, entre el mandato siguiente para importar la aplicación de empresa en WebSphere Application Server:
   ```
   xmlConfig -import OutputFileB.xml -adminNodeName nombresistpral_was
   ```
   donde *nombresistpral_was* es el nombre del nodo del WebSphere Application Server que contiene la aplicación actual.

Nota: 400 Si está desplegando en una instancia de WebSphere Commerce que se ejecuta en iSeries, deberá efectuar un paso adicional para modificar los permisos de directorio después de importar la aplicación. Consulte "Importación de una aplicación de empresa" en la página 308 para obtener detalles sobre cómo modificar estos permisos.

Inicio de la nueva aplicación de empresa: Después de importar la nueva aplicación de empresa utilizando el programa de utilidad de línea de mandatos XMLConfig, puede utilizar la Consola de administración de WebSphere Application Server para efectuar una renovación y luego iniciar la nueva aplicación.

Para renovar la consola e iniciar la nueva aplicación, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**
3. Resalte **Nodos**.
4. Pulse el icono **Renovar subárbol seleccionado**.
5. Inicie la aplicación de WebSphere Commerce, haciendo lo siguiente:
   - Expanda **Servidores de aplicaciones**.
   - Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en **WebSphere Commerce Server -** *nombreInstancia* y seleccione **Iniciar**.

Prueba del código nuevo en la tienda de destino

Para probar el bean de entidad modificado y el nuevo mandato de tarea en la tienda que se ejecuta en WebSphere Application Server, haga lo siguiente:

1. Abra un navegador y entre el siguiente URL:
   ```
   http://nombresistpral/webapp/wcs/stores/servlet/StoreCatalogDisplay?
        storeId=ID_tienda&catalogId=ID_catálogo&langId=-1
   ```
   donde *ID_tienda* es el identificador de la tienda e *ID_catálogo* es el identificador del catálogo de la tienda.

   Nota: Si recibe el Error 500, quizá tenga que reiniciar WebSphere Application Server para renovar la aplicación de empresa.
2. Pulse el enlace **Registrar** bajo la cabecera de Servicios y luego pulse **Registrar** bajo la cabecera Nuevo cliente.
   Registre un nuevo cliente utilizando la dirección de correo electrónico `wctester@wc` y la contraseña `wctester1`. Rellene otros campos con valores de prueba y pulse **Someter**. Deje el navegador abierto.

3. DB2 Abra el Centro de mandatos de DB2 y haga lo siguiente:
   a. Pulse la pestaña **Interactivo**
   b. En el campo **Mandato**, haga lo siguiente:
      1) Entre
         ```
         connect to nombre_base_de_datos
         ```
         donde *nombre_base_de_datos* es el nombre de su base de datos de WebSphere Commerce y pulse el icono Ejecutar.
      2) Entre `select users_id from USERREG where LOGONID = 'wctester@wc'` y pulse el icono Ejecutar.
   c. La pestaña con los resultados de la consulta muestra la entrada para el cliente que ha registrado en el paso 2. Anote aquí el valor de USERS_ID del cliente: _____
   d. Actualice el saldo de puntos de bonificación del cliente registrado en los pasos anteriores. Pulse la pestaña Interactivo y, en el campo **Mandato**, entre lo siguiente:
      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = id_usuario
      ```
      donde *id_usuario* es el valor del paso 3c. Pulse el icono Ejecutar.
4. Oracle Actualice el saldo de puntos de bonificación del usuario de prueba, haciendo lo siguiente:
   a. Abra la ventana de mandatos de SQL Plus de Oracle (**Inicio > Programas > Oracle > Application Development > SQL Plus**).
   b. En el campo **User Name**, entre su nombre de usuario de Oracle.
   c. En el campo **Password**, entre su contraseña de Oracle.
   d. En el campo **Host String**, entre su serie de conexión.
   e. Entre `select users_id from USERREG where LOGONID = 'wctester@wc';` para determinar el ID de usuario.
   f. Se muestra la entrada del cliente que ha registrado en el paso 2. Anote aquí el valor de USERS_ID del cliente: _____
   g. Actualice el saldo de puntos de bonificación del cliente registrado en los pasos anteriores, entrando lo siguiente:
      ```
      update BONUS set BONUSPOINT = 1000 where MEMBERID = id_usuario;
      ```
      donde *id_usuario* es el valor del paso 4f.
   h. Entre lo siguiente para comprometer los cambios en la base de datos:
      ```
      commit;
      ```
      y pulse Intro para ejecutar la sentencia de SQL.
5. En el navegador, pulse el enlace **Moda para caballero** para ver la sección de ropa de caballero de la tienda.
6. Pulse el enlace de la oferta estelar para ver la página de productos. La página muestra el precio normal y el precio con descuento según el saldo de puntos de bonificación que tiene el cliente.

Parte 5. Apéndices

Apéndice A. Inicio y detención del Entorno de prueba WebSphere

En esta sección se describen los pasos necesarios para iniciar el Entorno de prueba WebSphere dentro de VisualAge para Java. En general, para iniciar este entorno es necesario efectuar los pasos siguientes:

1. Iniciar el servidor de nombres persistentes.
2. Iniciar el servidor EJB.
3. Iniciar el motor de servlets.

En las siguientes secciones encontrará una descripción de cada uno de estos pasos.

Para detener el entorno de prueba, efectúe los pasos en orden inverso.

Nota: Para poder utilizar todas las características del Entorno de prueba WebSphere, debe asegurarse de que WebSphere Application Server no esté en ejecución antes de iniciar el Entorno de prueba WebSphere. Para obtener información sobre cómo detener WebSphere Application Server, consulte la publicación *WebSphere Commerce, Guía de instalación*.

Inicio y detención del servidor de nombres persistentes

El servidor de nombres persistentes recibe de los clientes peticiones de búsqueda de beans enterprise. Todos los beans enterprise están registrados en el servidor de nombres persistentes.

Para iniciar o detener el servidor de nombres persistentes, haga lo siguiente:

1. En el menú **Área de trabajo** de VisualAge para Java, seleccione **Herramientas > Entorno de prueba WebSphere**.
   Se abre el Centro de control del Entorno de prueba WebSphere.
2. Pulse **Servidor de nombres persistentes** y pulse en uno de los siguientes elementos:
   - **Iniciar servidor de nombres**.
     Espere a que aparezca el mensaje "Server open for business" en la ventana Consola. Este servidor puede tardar varios minutos en iniciarse.
   - **Detener servidor de nombres**.

Inicio y detención del servidor EJB

El servidor EJB es una aplicación o un proceso de alto nivel que proporciona un entorno de ejecución para dar soporte a la ejecución de aplicaciones de servidor que utilizan beans enterprise.

Para iniciar o detener el servidor EJB, haga lo siguiente:

1. Abra la ventana de configuración del servidor EJB (pulse la pestaña EJB y en el menú **EJB** seleccione **Abrir en > Configuración de servidor**).
2. Pulse con el botón derecho del ratón en el servidor EJB que desea iniciar o detener (por ejemplo, **EJB Server {Server 1}**) y efectúe una de las siguientes acciones:

- Seleccione **Iniciar servidor**
  Espere a que aparezca el mensaje “Server open for business” en la ventana Consola (es posible que tenga que seleccionar **Servidor EJB** en la Consola para ver el estado de este servidor). La operación de inicio de este servidor puede tardar de 10 a 15 minutos, según el sistema.
- Seleccione **Detener servidor**

La Consola es el dispositivo de entrada y salida estándar para los programas Java en el IDE. Para ver la salida de un programa concreto, primero seleccione el programa en el panel Todos los programas; la correspondiente salida se visualizará en el panel Salida.

Inicio y detención del motor de servlets

El motor de servlets integra la funcionalidad del servidor Web y de WebSphere Application Server para crear un entorno con fines de prueba.

Para iniciar o detener el motor de servlets, haga lo siguiente:

1. En el menú **Área de trabajo** de VisualAge para Java, seleccione **Herramientas > Entorno de prueba WebSphere**.
   Se abre el Centro de control del Entorno de prueba WebSphere.
2. Pulse **Motor de servlets** y después uno de los siguientes:
   - **Iniciar motor de servlets**.
     Cuando el motor de servlets se ha iniciado, la consola muestra el mensaje `***El motor de servlets se ha iniciado***`.
   - **Detener motor de servlets**.

Apéndice B. Información detallada sobre el despliegue

Después de crear código personalizado en VisualAge para Java y de probarlo dentro del Entorno de prueba WebSphere, debe desplegarlo en una instancia de WebSphere Commerce que se ejecute fuera del Entorno de prueba WebSphere. Esta instancia de WebSphere Commerce puede ejecutarse localmente en su máquina de desarrollo o puede estar en otra máquina (utilizando el mismo sistema operativo u otro distinto).

Este apéndice describe los pasos necesarios para el despliegue de código personalizado en una instancia de WebSphere Commerce que se ejecute fuera del Entorno de prueba WebSphere. Puede consultar los pasos generales del proceso de despliegue en el apartado "Despliegue del código" en la página 159 y, a continuación, volver a este apéndice para obtener información más detallada.

Correlación con el sistema de archivos integrado (iSeries)

400 Esta sección sólo es aplicable si su WebSphere Commerce Server de destino está ejecutándose en una plataforma iSeries.

Si su WebSphere Commerce Server de destino está ejecutándose en una plataforma iSeries, debe correlacionar una unidad local de su máquina de desarrollo con el sistema de archivos integrado (IFS) del servidor iSeries. En secciones posteriores de este documento, se utiliza *unidad_iSeries* para hacer referencia a esta unidad local que está correlacionada con IFS. Además, se utiliza *unidad* para una unidad local de su máquina de desarrollo (una que no esté correlacionada con IFS).

Archivos JAR para mandatos y beans de datos personalizados

El código de beans de datos y mandatos personalizados debe almacenarse en un proyecto aparte del código de WebSphere Commerce. Cuando haya completado la prueba dentro del Entorno de prueba WebSphere, debe crear un archivo JAR para el proyecto que contenga el código de los beans de datos y mandatos personalizados y, a continuación, colocar el archivo JAR en el directorio correspondiente del WebSphere Commerce Server de destino.

Para crear un archivo JAR para beans de datos y mandatos personalizados, haga lo siguiente:

1. En el Entorno de trabajo de VisualAge para Java, seleccione la pestaña **Proyecto**.
2. Pulse con el botón derecho del ratón en el proyecto que contiene el código de los mandatos y beans de datos personalizados y seleccione **Exportar**.
   Se abre el SmartGuide Exportar.
3. Seleccione **Archivo JAR** y pulse **Siguiente**.
4. En el campo **Archivo Jar**, entre lo siguiente:
   unidad:\WebSphere\CommerceServerDev*dir_temp*\ *nombre_arch_jar*_1.jar
   donde *unidad*:\WebSphere\CommerceServerDev*dir_temp*\ es un directorio temporal que tiene suficiente espacio libre para el archivo JAR y *nombre_arch_jar*_1.jar es el nombre del archivo JAR añadiéndole _1 al final.

5. Seleccione los atributos para el archivo JAR, de la forma siguiente:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
beans	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
6. Pulse **Terminar**.

Puesto que el archivo JAR de implementación creado no contiene toda la información de denominación de paquetes, debe utilizar otro programa de utilidad de empaquetado (fuera de VisualAge para Java) para volver a empaquetar el archivo JAR. Para ello, haga lo siguiente:

1. En una ventana de mandatos, vaya al directorio donde ha guardado *nombre_arch_jar*_1.jar en los pasos anteriores.
2. Entre `mkdir temp1`.
3. Entre `cd temp1`.
4. Establezca la vía de acceso de la forma siguiente:
   `set PATH=%PATH%;`*`unidad`*`:\WebSphere\WebSphereStudio4\bin;`
   donde *unidad* es la unidad en la que está instalado WebSphere Studio.
5. Entre `jar xvf ../`*`nombre_arch_jar`*`_1.jar`.
6. Entre `jar cvf ../`*`nombre_arch_jar`*`.jar *` (observe que se ha eliminado _1 del nombre).
7. Vaya al directorio *unidad*:\WebSphere\CommerceDev*dir_temp*.
8. Si está desplegando en una instancia local de WebSphere Commerce, copie *nombre_arch_jar*.jar en el siguiente directorio:

   ```
   unidad:\WebSphere\AppServer\installedApps\
      WC_Enterprise_App_nombreInstancia.ear\wcstores.war\WEB-INF\lib
   ```

   De lo contrario, deje el archivo JAR en el directorio temporal hasta que tenga que transferir todos los elementos de archivo al WebSphere Commerce Server de destino.

Creación de archivos JAR para beans de entidad nuevos

Al crear beans de entidad nuevos, debe almacenar el código en un proyecto separado de todo el código de WebSphere Commerce. Además, también debe estar separado del código de los mandatos y beans de datos personalizados. Debe crear un bean de entidad nuevo en un grupo EJB distinto de los grupos EJB que contienen los beans de entidad públicos de WebSphere Commerce.

Al desplegar los beans de entidad nuevos, deberá crear dos archivos JAR. El primero es el archivo JAR de exportación de EJB 1.1 y se crea mediante las herramientas disponibles al seleccionar la pestaña EJB. El segundo contiene el código de implementación para el bean de entidad y se crea a partir del proyecto que contiene el código del bean de entidad. Este último archivo JAR se crea utilizando las herramientas disponibles al seleccionar la pestaña Proyectos en el entorno de trabajo de VisualAge para Java.

Creación del archivo JAR de exportación de EJB 1.1

Para crear el archivo JAR de exportación de EJB 1.1 para los beans de entidad nuevos, haga lo siguiente:

1. En el Entorno de trabajo de VisualAge para Java, seleccione la pestaña **EJB**.
2. Pulse con el botón derecho del ratón en el grupo EJB que contiene el código de bean de entidad personalizado y seleccione **Exportar > EJB 1.1 JAR**.
   Se abre el SmartGuide de exportación a archivo JAR de EJB 1.1.
3. En el campo **Archivo Jar**, entre lo siguiente:
   unidad:\WebSphere\CommerceServerDev\dir_temp*nombreArchJar*_DT.jar
   donde *unidad*:\WebSphere\CommerceServerDev\dir_temp es un directorio temporal que tiene suficiente espacio libre para su archivo JAR y *nombreArchJar*_DT.jar es el nombre de su archivo JAR añadiéndole _DT al final.

Al nombre del archivo JAR se le ha añadido el sufijo "_DT" como recordatorio de que debe ejecutar este archivo JAR a través de la herramienta de despliegue de EJB proporcionada con WebSphere Application Server, antes de desplegarlo en su aplicación de WebSphere Commerce.

4. Seleccione los atributos para el archivo JAR, de la forma siguiente:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
Base de datos de destino	DB2 Si va a desplegar en una base de datos DB2, seleccione uno de los siguientes: • Windows AIX Solaris **DB2 para NT, V7.1** • 400 **DB2 para AS/400, V4** Oracle Si va a desplegar en una base de datos Oracle, seleccione **Oracle, V8**
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
5. Pulse **Terminar**.

Se crea el archivo JAR.

Creación del archivo JAR de implementación

Para crear el archivo JAR de implementación para los beans de entidad nuevos, haga lo siguiente:

1. En el Entorno de trabajo de VisualAge para Java, seleccione la pestaña **Proyecto**.
2. Pulse con el botón derecho del ratón en el proyecto que contiene el código personalizado del bean de entidad y seleccione **Exportar**.
   Se abre el SmartGuide Exportar.
3. Seleccione **Archivo JAR** y pulse **Siguiente**.

4. En el campo **Archivo Jar**, entre lo siguiente:
   unidad:\WebSphere\CommerceServerDev*dir_temp**nombreArchJar*_1.jar
   donde *unidad*:\WebSphere\CommerceServerDev*dir_temp* es un directorio temporal que tiene suficiente espacio libre para el archivo JAR y *nombreArchJar*_1.jar es el nombre del archivo JAR añadiéndole _1 al final.
5. Seleccione los atributos para el archivo JAR, de la forma siguiente:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado
beans	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

   Acepte los valores por omisión para los otros atributos.
6. Pulse **Terminar**.

Puesto que el archivo JAR de implementación creado no contiene toda la información de denominación de paquetes, debe utilizar otro programa de utilidad de empaquetado (fuera de VisualAge para Java) para volver a empaquetar el archivo JAR. Para ello, haga lo siguiente:

1. En una ventana de mandatos, vaya al directorio donde ha guardado *nombreArchJar*_1.jar en los pasos anteriores.
2. Entre `mkdir temp1`.
3. Entre `cd temp1`.
4. Establezca la vía de acceso de la forma siguiente:
   `set PATH=%PATH%;`*unidad*`:\WebSphere\WebSphereStudio4\bin;`
   donde *unidad* es la unidad en la que está instalado WebSphere Studio.
5. Entre `jar xvf ../`*nombreArchJar*`_1.jar`.
6. Entre `jar cvf ../`*nombreArchJar*`.jar *` (observe que se ha eliminado _1 del nombre).
7. Vaya al directorio *unidad*:\WebSphere\CommerceServerDev*dir_temp*.
8. Si está desplegando en una instancia local de WebSphere Commerce, copie *nombreArchJar*_1.jar en el siguiente directorio:
   unidad:\WebSphere\CommerceServer\temp\lib
   De lo contrario, deje el archivo JAR en el directorio temporal hasta que tenga que transferir todos los elementos de archivo al WebSphere Commerce Server de destino.

Creación de archivos JAR para beans de entidad de WebSphere Commerce personalizados

Puede ampliar cualquiera de los beans de entidad públicos de WebSphere Commerce. Estos beans se encuentran en los siguientes grupos EJB:

- WCSActrlEJBGroup
- WCSApproval
- WCSAuction
- WCSCatalog
- WCSCommon

- WCSContract
- WCSCoupon
- WCSFulfillment
- WCSInventory
- WCSMessageExtensions
- WCSOrder
- WCSOrderManagement
- WCSOrderStatus
- WCSPayment
- WCSPVCDevices
- WCSTaxation
- WCSUserTraffic
- WCSUser
- WCSUTF

Si amplía alguno de los beans de entidad públicos de WebSphere Commerce, debe crear un archivo JAR de exportación de EJB 1.1 para el grupo EJB que contiene el bean de entidad púbico de WebSphere Commerce modificado.

Creación del archivo JAR de exportación de EJB 1.1

Para crear el archivo JAR de exportación de EJB 1.1 para el grupo EJB que contiene el bean de entidad modificado, haga lo siguiente:

1. En el Entorno de trabajo de VisualAge para Java, seleccione la pestaña **EJB**.
2. Pulse con el botón derecho del ratón en el grupo EJB que contiene el código de bean de entidad personalizado y seleccione **Exportar > EJB 1.1 JAR**.
   Se abre el SmartGuide de exportación a archivo JAR de EJB 1.1.
3. En el campo **Archivo Jar**, entre lo siguiente:
   unidad:\WebSphere\CommerceServerDev*dir_temp*\ Cust_*nombreGrupoEJB*-ejb_DT.jar
   donde *unidad*:\WebSphere\CommerceServerDev*dir_temp* es un directorio temporal que tiene suficiente espacio libre para su archivo JAR y Cust_*nombreGrupoEJB*-ejb_DT.jar es el nombre de su archivo JAR añadiéndole _DT al final.

Al nombre del archivo JAR se le ha añadido el sufijo "_DT" como recordatorio de que debe ejecutar este archivo JAR a través de la herramienta de despliegue de EJB proporcionada con WebSphere Application Server, antes de desplegarlo en su aplicación de WebSphere Commerce.

4. Seleccione los atributos para el archivo JAR, de la forma siguiente:

Atributo	Valor
class	Seleccionado
java	No seleccionado
resource	Seleccionado

Atributo	Valor
Base de datos de destino	DB2 Si va a desplegar en una base de datos DB2, seleccione uno de los siguientes: • Windows AIX Solaris **DB2 para NT, V7.1** • 400 **DB2 para AS/400, V4** Oracle Si va a desplegar en una base de datos Oracle, seleccione **Oracle, V8**
Incluir atributos de depuración en archivos .class	Seleccionado

Acepte los valores por omisión para los otros atributos.

5. Pulse **Terminar**.

Creación del archivo JAR de cliente

Para crear el archivo JAR de cliente, haga lo siguiente:

1. Con la pestaña EJB seleccionada, resalte todos los grupos EJB de WebSphere Commerce (los nombres empiezan por `WCS`). Con todos estos grupos resaltados, pulse el botón derecho y seleccione **Exportar > JAR de cliente**.
   Se abre el SmartGuide Exportar.
2. En el campo **Archivo JAR**, entre lo siguiente:
   unidad`:\WebSphere\CommerceServerDev\`*dir_temp*`\wcsejsclient.jar`
3. Seleccione los atributos de la forma indicada a continuación:

Atributo	Valor
beans	Seleccionado
class	Seleccionado
java	No seleccionado
resource	Seleccionado
Incluir atributos de depuración en archivos .class	Seleccionado

Acepte los valores por omisión para los otros atributos.

4. Pulse **Terminar**.

Se crea el archivo JAR.

Almacenamiento de elementos en el WebSphere Commerce Server de destino

Los elementos relacionados con el código personalizado deben copiarse en el WebSphere Commerce Server de destino. Estos elementos incluyen archivos JAR para beans de entidad, beans de datos y mandatos personalizados. También puede tener nuevas plantillas JSP y gráficos que dan soporte a la personalización.

AIX Solaris Debe realizar todos los pasos de despliegue en el WebSphere Commerce Server de destino utilizando el usuario que se creó cuando se llevaron a cabo los pasos de la sección *"Ejecución del script posterior a la instalación"* de la publicación *WebSphere Commerce, Guía de instalación*. Por omisión, este usuario es `wasuser`. Además, compruebe que este usuario tiene autorización de lectura,

grabación y ejecución de sus elementos de archivo (por ejemplo, los archivos JAR) y los directorios en los que se encuentran.

400 Asegúrese de que las autorizaciones para los directorios `/QIBM/UserData/WebCommerce/instances/`*`nombreInstancia`* y `/QIBM/UserData/WebCommerce/instances/ `*`nombreInstancia`*`/working` incluyen al usuario `QEJB`. Añada este usuario en ambos directorios, estableciendo la autorización de datos en `*RWX`.

La tabla siguiente resume los directorios estándar en los que se almacenan los elementos en un WebSphere Commerce Server de destino ejecutándose en Windows NT o Windows 2000.

Tabla 12.

Tipo de elemento	Ubicación del directorio en WebSphere Commerce Server de destino
Archivos JAR para código de mandatos y beans de datos	*unidad*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*nombreInstancia*`.ear\wcstores.war\WEB-INF\lib`
JAR de exportación de EJB 1.1 creado con VisualAge para Java	*unidad*`:\WebSphere\CommerceServer\temp` **Nota:** Este archivo se pasa como entrada a la herramienta EJBDeploy para generar código desplegado que se pueda utilizar en WebSphere Application Server V4.0.
Archivo JAR de código cliente de EJB	*unidad*`:\WebSphere\CommerceServer\temp\lib` **Nota:** Este archivo sólo se utiliza en la vía de acceso de clases para la herramienta EJBDeploy.
Archivo JAR de código de implementación de EJB	*unidad*`:\WebSphere\CommerceServer\temp\lib` **Nota:** Este archivo se añade a la aplicación de empresa como elemento de archivo.
plantillas JSP	*unidad*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*nombreInst*`.ear\wcstores.war\`*dirTienda*
Imágenes	*unidad*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*nombreInst*`.ear\wcstores.war\`*dirTienda*`\images`

Para almacenar elementos en el WebSphere Commerce Server de destino, haga lo siguiente:

1. Localice los archivos JAR para su código de mandatos y beans de datos. Deberían estar en uno de los siguientes directorios de la máquina de desarrollo:
   - Windows *unidad*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*nombreInstancia*`.ear\wcstores.war\WEB-INF\lib`
   - Windows *unidad*`:\WebSphere\CommerceServerDev\`*dir_temp*
2. Copie los archivos JAR para su código de mandatos y bean de datos en uno de los siguientes directorios del WebSphere Commerce Server de destino:
   - Windows *unidad*`:\WebSphere\AppServer\installedApps\WC_Enterprise_App_`*nombreInstancia*`.ear\wcstores.war\WEB-INF\lib`
   - AIX `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_`*nombreInstancia*`.ear/wcstores.war/WEB-INF/lib`

- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/WEB-INF/lib
- 400 /QIBM/UserData/WebASAdv4/*nombreInstaciaAdmin_WAS*/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/WEB-INF/lib

3. Localice los archivos JAR de exportación de EJB 1.1 para los grupos EJB nuevos así como cualquier bean de entidad de WebSphere Commerce modificado en el siguiente directorio de la máquina de desarrollo:
   Windows *unidad*:\WebSphere\CommerceServerDev*dir_temp*
4. Copie los archivos JAR de exportación de EJB 1.1 en uno de los siguientes directorios del WebSphere Commerce Server de destino:
   - Windows *unidad*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
   - 400 /QIBM/UserData/WebCommerce/instances/*nombreInst*/temp
5. Si ha creado beans enterprise nuevos, localice el archivo JAR de implementación para estos beans en el siguiente directorio de la máquina de desarrollo:
   Windows *unidad*:\WebSphere\CommerceServerDev*dir_temp*
6. Copie el archivo JAR de implementación para sus nuevos beans enterprise en el siguiente directorio del WebSphere Commerce Server de destino:
   - Windows *unidad*:\WebSphere\CommerceServer\temp\lib
   - AIX /usr/WebSphere/CommerceServer/temp/lib
   - Solaris /opt/WebSphere/CommerceServer/temp/lib
   - 400 /QIBM/UserData/WebCommerce/instances/*nombreInstancia*/temp/lib
7. Si ha modificado beans de entidad de WebSphere Commerce existentes, localice el archivo JAR cliente en el siguiente directorio de la máquina de desarrollo:
   Windows *unidad*:\WebSphere\CommerceServerDev*dir_temp*
8. Copie el archivo JAR de cliente en uno de los siguientes directorios del WebSphere Commerce Server de destino:
   - Windows *unidad*:\WebSphere\CommerceServer\temp\lib
   - AIX /usr/WebSphere/CommerceServer/temp/lib
   - Solaris /opt/WebSphere/CommerceServer/temp/lib
   - 400 /QIBM/UserData/WebCommerce/instances/*nombreInstancia*/temp/lib
9. Copie las plantillas JSP en los siguientes directorios del WebSphere Commerce Server de destino:
   - Windows *unidad*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*nombreInstancia*.ear\wcstores.war*dir_tienda*
   - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/*dir_tienda*

- Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/*dir_tienda*
- 400 /QIBM/UserData/WebASAdv4/*nombreInstanciaAdmin_WAS*/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/*directorio_tienda*

donde *directorio_tienda* es el directorio para su tienda.

10. Copie las imágenes en los siguientes directorios del WebSphere Commerce Server de destino:
    - Windows *unidad*:\WebSphere\AppServer\installedApps\WC_Enterprise_App_*nombreInstancia*.ear\wcstores.war*directorio_tienda*\images
    - AIX /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/*directorio_tienda*/images
    - Solaris /opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/*directorio_tienda*/images
    - 400 /QIBM/UserData/WebASAdv4/*nombreInstanciaAdmin_WAS*/installedApps/WC_Enterprise_App_*nombreInstancia*.ear/wcstores.war/*directorio_tienda*/images

    donde *directorio_tienda* es el directorio para su tienda.

Actualización de la base de datos de destino

Cuando el WebSphere Commerce Server de destino utiliza una base de datos distinta a la que utiliza la máquina de desarrollo, todas las actualizaciones que se llevaron a cabo en la base de datos de desarrollo deben también efectuarse en la base de datos que utiliza el WebSphere Commerce Server de destino. Esto incluye las actualizaciones para el registro de mandatos nuevos o modificados, tablas adicionales que se hayan creado y la creación de políticas de control de acceso para cualquier recurso nuevo que se haya creado.

Para obtener información sobre la carga de políticas de control de acceso (incluida la sintaxis de mandatos para varias plataformas y los requisitos de permisos de directorio), consulte la publicación *WebSphere Commerce Guía de control de acceso*.

400 Usted es el responsable de poseer un programa de utilidad para ejecutar sentencias SQL. Una forma de hacerlo es utilizar Client Access Express V5R1 (instalación completa). Para abrir este programa de utilidad, haga lo siguiente:

1. Abra **Operations Navigator**.
2. Una vez abierto, se le solicitará que se conecte a un sistema específico. Asegúrese de seleccionar la máquina de iSeries de destino y de utilizar el perfil de usuario y la contraseña de la instancia de WebSphere Commerce. Así se asegura que el perfil de usuario de la instancia de WebSphere Commerce es el propietario de todas las nuevas tablas que se crean.
3. Pulse en su sistema en el panel de la izquierda, a continuación, pulse con el botón derecho del ratón en en **BASE DE DATOS** y seleccione **Ejecutar scripts SQL** en la lista desplegable.
   Se abre la ventana Ejecutar scripts SQL. Utilizando esta ventana, puede cortar y

pegar sentencias SQL o abrir un script SQL. Puede establecer el esquema por omisión utilizando las opciones de configuración de Connection/JDBC.

Generación de código desplegado

Esta sección describe cómo utilizar la herramienta de despliegue de EJB para generar el código desplegado para los beans enterprise contenidos en el archivo JAR de exportación de EJB 1.1. Esta herramienta la proporciona WebSphere Application Server.

Para generar el código desplegado, efectúe lo siguiente:

1. En un indicador de mandatos del WebSphere Commerce Server de destino, vaya al directorio siguiente:
   - Windows *unidad*:\WebSphere\CommerceServer\temp
   - AIX /usr/WebSphere/CommerceServer/temp
   - Solaris /opt/WebSphere/CommerceServer/temp
2. 400 En un indicador de mandatos del WebSphere Commerce Server de destino, entre los mandatos siguientes:

   ```
   STRQSH
   cd /QIBM/UserData/WebCommerce/instances/nombreInstancia/temp
   ```
3. Añada temporalmente la herramienta a la vía de acceso del sistema entrando el siguiente mandato:
   - Windows PATH=*unidad*:\WebSphere\AppServer\deploytool;%PATH%
   - AIX PATH=/usr/WebSphere/AppServer/deploytool:$PATH
   - Solaris PATH=/opt/WebSphere/AppServer/deploytool:$PATH
   - 400 PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH

     CP=*ViaClasesDeArchJARDep*

     donde *ViaClasesDeArchJARDep* es la vía de clases de los archivos JAR dependientes. Tenga en cuenta que si modifica un bean enterprise de WebSphere Commerce existente, debe incluir los archivos wcsejsclient.jar, wcsejbimpl.jar y xml4j.jar en la vía de acceso de clases de los archivos JAR dependientes. A continuación se muestra un ejemplo de vía de acceso de clases para cuando se ha modificado un bean enterprise de WebSphere Commerce existente:

     ```
     CP=/QIBM/UserData/WebCommerce/instances/nombreInstancia/temp/lib/
     wcsejsclient.jar:
     /QIBM/UserData/WebASAdv4/nombreInstanciaAdmin_WAS/installedApps/
     WC_Enterprise_App_nombreInstancia.ear/lib/wcsejbimpl.jar:
     /QIBM/UserData/WebASAdv4/nombreInstanciaAdmin_WAS/installedApps/
     WC_Enterprise_App_nombreInstancia.ear/lib/xml4j.jar
     ```

     Nota: Los saltos de línea que aparecen en el ejemplo de vía de acceso de clases anterior sólo se muestran a efectos de presentación. Debe entrar la vía de acceso en una sola línea.
4. Solaris Para no sobrepasar el límite de número de caracteres en la interfaz de línea de mandatos, se utiliza un script para invocar el mandato ejbdeploy. Haga lo siguiente para crear el script e invocar el mandato:
   a. Vaya al siguiente directorio:

      ```
      /opt/WebSphere/AppServer/bin
      ```

b. Cree un nuevo archivo y póngale un nombre de script adecuado. Por ejemplo, denomínelo `myejbd.sh`.

c. Incluya la siguiente información en el archivo `myejbd.sh`:

```
#!/bin/sh
./ejbdeploy.sh ArchJARGrupoEJB DirTrabajo ArchJARSalida
-nowarn -keep -35 -cp ViaClasesDeArchJARDep
```

donde las variables tienen la misma definición que en el paso 5 (observe que no efectúa ese paso). A continuación se muestra un ejemplo de lo que debe incluir en el archivo script, incluyendo valores para las variables:

```
#!/bin/sh
./ejbdeploy.sh
/opt/WebSphere/CommerceServer/temp/CustomizedWCSUserDeployed_DT.jar
. /opt/WebSphere/CommerceServer/temp/CustomizedWCSUserDeployed.jar
-nowarn -keep -35 -cp "/opt/WebSphere/CommerceServer/temp/lib/
wcsejsclient.jar:/opt/WebSphere/AppServer/installedApps/
WC_Enterprise_App_demo.ear/lib/wcsejbimpl.jar:/opt/WebSphere/
AppServer/installedApps/WC_Enterprise_App_demo.ear/lib/xml4j.jar"
```

Nota: Después del mandato `./ejbdeploy.sh`, algunas líneas se muestran partidas sólo a efectos de visualización. En su archivo, no debería haber saltos de línea después del mandato `./ejbdeploy.sh`.

Guarde el script y haga que sea ejecutable.

d. Invoque el script entrando lo siguiente:

```
./myejbd.sh
```

5. Windows AIX 400 Entre el mandato `ejbdeploy` de la manera siguiente:

Windows 400

```
ejbdeploy ArchJARGrupoEJB_DT DirTrabajo ArchJARSalida -nowarn -keep -35 -cp
   ViaClasesDeArchJARDep
```

AIX

```
/usr/WebSphere/AppServer/bin/ejbdeploy.sh ArchJARGrupoEJB_DT DirTrabajo
ArchJARSalida -nowarn -keep -35
-cp ViaClasesDeArchJARDep
```

donde:

- *ArchJARGrupoEJB*_DT es el nombre del archivo JAR para su grupo EJB. Por ejemplo, si ha modificado un bean en el grupo EJB WCSUser, un ejemplo de uso en una plataforma basada en Windows es

  ```
  unidad:\WebSphere\CommerceServer\temp\CustomizedWCSUser_DT.jar
  ```

  400 Un ejemplo de uso para la plataforma iSeries es

  ```
  /QIBM/UserData/WebCommerce/instances/nombreInstancia/temp/
  CustomizedWCSUser_DT.jar
  ```

- *DirTrabajo* es el nombre del directorio donde se almacenan los archivos temporales necesarios para la generación de código.
- *ArchJARSalida* es el nombre totalmente calificado del archivo JAR de salida. Por ejemplo, puede entrar `CustomizedWCSUserDeployed.jar`.
- `-nowarn` es un parámetro opcional para suprimir los mensajes de aviso e información.
- `-keep` es un parámetro opcional para mantener el directorio de trabajo después de ejecutar el mandato ejbdeploy.

- `-35` es un parámetro obligatorio que utilizará las mismas normas de correlación de mayor a menor para los beans de entidad CMP que las que se utilizan en la herramienta de despliegue de EJB que se ha proporcionado con WebSphere Application Server, Versión 3.5.
- `-cp` *`ViaClasesDeArchJARDep`* es la vía de clases de los archivos JAR dependientes. Cuando modifica un bean enterprise de WebSphere Commerce existente, debe incluir los archivos `wcsejsclient.jar` , `wcsejbimpl.jar` y `xml4j.jar` en la vía de acceso de clases de los archivos JAR dependientes. A continuación se muestra un ejemplo de vía de acceso de clases para cuando se ha modificado un bean enterprise de WebSphere Commerce existente:

```
"unidad:\WebSphere\CommerceServer\temp\lib\wcsejsclient.jar;
unidad:\WebSphere\AppServer\installedApps\
WC_Enterprise_App_nombreInstancia.ear \lib\wcsejbimpl.jar;
unidad:\WebSphere\AppServer\installedApps\
WC_Enterprise_App_nombreInstancia.ear\lib\xml4j.jar;"
```

Nota: 400 Para los valores de vía de acceso de clases, entre `-cp $CP` donde `CP` se ha definido anteriormente con las entradas de vía de acceso de clases adecuadas. Además, las comillas no son obligatorias.

Modificación del nivel de aislamiento de transacción de los beans de entidad

Esta sección describe cómo utilizar el programa de utilidad de línea de mandatos `modifyIsolationLevel` para establecer el nivel de aislamiento de transacción de los beans de entidad en el nivel apropiado según el tipo de base de datos.

Para ejecutar el mandato `modifyIsolationLevel`, haga lo siguiente:

1. En el WebSphere Commerce Server de destino, utilice un indicador de mandatos para navegar al directorio siguiente:
   - Windows *`unidad`*`:\WebSphere\CommerceServer\bin`
   - AIX `/usr/WebSphere/CommerceServer/bin`
   - Solaris `/opt/WebSphere/CommerceServer/bin`
2. 400 Entre los siguientes mandatos:

```
STRQSH
cd /QIBM/ProdData/WebCommerce/bin
```

3. Debe emitir el mandato `modifyIsolationLevel`, que tiene la siguiente sintaxis general:

   Windows AIX 400

```
modifyIsolationLevel -jarFile arch_jar.jar
   -logFile arch_anotaciones -dbType tipo_bd
```

   Solaris

```
./modifyIsolationLevel.sh -jarFile arch_jar.jar
   -logFile arch_anotaciones -dbType tipo_bd
```

   donde

   - *`arch_jar.jar`* es el nombre del archivo JAR que contiene el código personalizado
   - *`arch_anotaciones`* es el nombre totalmente calificado del archivo en el que deben almacenarse las anotaciones cronológicas

- *tipo_bd* es el tipo de base de datos que utiliza. Entre DB2 u ORACLE

A continuación se muestra un ejemplo del mandato `modifyIsolationLevel` con todos los valores especificados para su uso en una plataforma Windows:

```
modifyIsolationLevel -jarFile
   D:\WebSphere\CommerceServer\temp\CustomizedWCSUserDeployed.jar
   -logFile D:\WebSphere\CommerceServer\instances\demo\logs\output.log
   -dbType DB2
```

400 A continuación se muestra un ejemplo del mandato `modifyIsolationLevel` con todos los valores especificados para su uso en una plataforma iSeries:

```
modifyIsolationLevel -jarFile
   /QIBM/UserData/WebCommerce/instances/nombreInstancia/temp/
   CustomizedWCSUserDeployed.jar -logFile /QIBM/UserData/WebCommerce/
   instances/nombreInstancia/logs/output.log -dbType DB2
```

En los ejemplos anteriores, las líneas se muestran partidas sólo a efectos de visualización.

Nota: Los nombres de parámetro son sensibles a las mayúsculas y minúsculas. Es decir, `jarFile` no es lo mismo que `jarfile`. Asegúrese de entrar los nombres de parámetro correctamente.

El mandato se ha ejecutado satisfactoriamente si no se muestran excepciones en la ventana de mandatos. Después de la ejecución del mandato, observe que la indicación de la hora del archivo JAR desplegado ha cambiado.

Exportación de la aplicación de empresa actual de WebSphere Commerce

Esta sección describe cómo utilizar la Consola de administración de WebSphere Application Server para exportar la aplicación de empresa actual de WebSphere Commerce.

Para exportar la aplicación de empresa actual de WebSphere Application Server, haga lo siguiente:

1. Asegúrese de que en el WebSphere Commerce Server de destino existe el siguiente directorio:
   - Windows *unidad*`:\WebSphere\CommerceServer\working`
   - AIX `/usr/WebSphere/CommerceServer/working`
   - Solaris `/opt/WebSphere/CommerceServer/working`
   - 400 `/QIBM/UserData/WebCommerce/instances/` *nombreInstancia*`/working`

   Si todavía no existe este directorio, créelo ahora.

   Nota: AIX Solaris Compruebe que el permiso para el directorio `/working` esté establecido en el usuario que se creó al llevar a cabo los pasos de la sección *"Ejecución del script posterior a la instalación"* de la publicación *WebSphere Commerce, Guía de instalación*.

   400 Asegúrese de que las autorizaciones para los directorios `/QIBM/UserData/WebCommerce/instances/`*nombreInstancia* y `/QIBM/UserData/WebCommerce/instances/`*nombreInstancia*`/working`

incluyen al usuario QEJB. Añada este usuario en ambos directorios, estableciendo la autorización de datos en *RWX.

2. Abra la Consola de administración de WebSphere Application Server.
3. Expanda **Dominio de Administración de WebSphere**.
4. Expanda **Aplicación de empresa**.
5. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, expanda la aplicación **demo** y seleccione **Exportar aplicación**.
6. En el campo **Exportar directorio**, entre lo siguiente:
   - Windows *unidad*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/ *nombreInstancia*/working

   Esto exporta toda la aplicación, incluidos todos los recursos en el archivo WC_Enterprise_App_*nombreInstancia*.ear (donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce).

Exportación de la información de configuración para los beans enterprise

Esta sección describe cómo utilizar la opción -export del programa de utilidad de línea de mandatos XMLConfig para exportar la información de configuración para los beans enterprise contenidos en la aplicación de empresa existente.

Después de exportar esta información, debe añadir información manualmente para cualquier bean nuevo que esté añadiendo a la aplicación.

Para exportar esta información de configuración, haga lo siguiente:

1. Copie el archivo was.export.app.xml del directorio siguiente del WebSphere Commerce Server de destino:
   - Windows *unidad*:\WebSphere\CommerceServer\xml\config
   - AIX /usr/WebSphere/CommerceServer/xml/config
   - Solaris /opt/WebSphere/CommerceServer/xml/config
   - 400 /QIBM/ProdData/WebCommerce/xml/config

   al directorio siguiente del WebSphere Commerce Server de destino:
   - Windows *unidad*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
   - 400 /QIBM/UserData/WebCommerce/instances/ *nombreInstancia*/working

   donde
   - *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.
2. Windows AIX Solaris Abra el archivo was.export.app.xml en un editor de texto. En este archivo, sustituya todas las apariciones de $Enterprise_Application_Name$ por

   WebSphere Commerce Enterprise Application - *nombreInstancia*

donde *nombreInstancia* es el nombre de su instancia de WebSphere Commerce (por ejemplo, demo). Guarde este archivo.

Nota: El valor que inserte debe coincidir con la información para su instancia que aparece en la Consola de administración de WebSphere Application Server.

3. 400 Abra el archivo was.export.app.xml en un editor de texto. En este archivo, sustituya todas las apariciones de $Enterprise_Application_Name$ por

   nombreInstancia - WebSphere Commerce Enterprise Application

   donde *nombreInstancia* es el nombre de su instancia de WebSphere Commerce (por ejemplo, demo). Guarde este archivo.

   Nota: El valor que inserte debe coincidir con la información para su instancia que aparece en la Consola de administración de WebSphere Application Server.

4. Windows AIX Solaris En un indicador de mandatos, vaya al siguiente directorio:
   - Windows *unidad*:\WebSphere\CommerceServer\working
   - AIX /usr/WebSphere/CommerceServer/working
   - Solaris /opt/WebSphere/CommerceServer/working
5. 400 En un indicador de mandatos, entre lo siguiente:

   ```
   STRQSH
   PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
   cd /QIBM/UserData/WebCommerce/instances/nombreInstancia/working
   ```

6. Windows AIX Solaris Invoque la herramienta XMLConfig para efectuar una exportación parcial entrando el siguiente mandato:

   Windows

   ```
   xmlConfig -export OutputFile.xml -partial was.export.app.xml
      -adminNodeName sistpralWas
   ```

   AIX

   ```
   /usr/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
      -partial was.export.app.xml -adminNodeName sistpralWas
      -nameServiceHost sistpralWas -nameServicePort puertoAdminWas
   ```

   Solaris

   ```
   /opt/WebSphere/AppServer/bin/XMLConfig.sh  -export OutputFile.xml
      -partial was.export.app.xml -adminNodeName sistpralWas
      -nameServiceHost sistpralWas -nameServicePort puertoAdminWas
   ```

   donde
   - *sistpralWas* es el nombre del nodo en el WebSphere Application Server que contiene la aplicación de empresa actual.
   - OutputFile.xml es el nombre del archivo que se crea al ejecutar este mandato y was.export.app.xml es el archivo que ha modificado en el paso 2.
   - *puertoAdminWas* es el puerto de administración de WebSphere Application Server.

7. 400 Invoque la herramienta XMLConfig para efectuar una exportación parcial entrando el siguiente mandato:

   ```
   xmlConfig -export OutputFile.xml -partial was.export.app.xml
   -adminNodeName sistpralWas -nameServiceHost sistpralWas
   -nameServicePort puertoAdminWas -instance nombreInstanciaWas
   ```

   donde

   - *sistpralWas* es el nombre del nodo en el WebSphere Application Server que contiene la aplicación de empresa actual.

     Nota: El valor para *sistpralWas* es sensible a las mayúsculas y minúsculas y debe coincidir con el valor que hay en la configuración de TCP/IP. (Utilice un procesador de línea de mandatos para acceder a CFGTCP, opción 12 para verificar el nombre de sistema principal.)
   - `OutputFile.xml` es el nombre del archivo que se crea al ejecutar este mandato y `was.export.app.xml` es el archivo que ha modificado en el paso 3.
   - *puertoAdminWas* es el puerto de administración de WebSphere Application Server.
   - *nombreInstanciaWas* es el nombre de la instancia de WebSphere Application Server.

Después de exportar la información de configuración para cada uno de los beans contenidos en la aplicación de empresa actual, debe añadir una nueva sección que describa cada bean enterprise que añadirá a su aplicación. Por ejemplo, si tiene un nuevo bean de entidad denominado “Bonus”, debe añadir una sección que describa este bean Bonus. Además, debe sustituir una variable en el archivo de configuración para especificar el nombre exacto del archivo .ear para su aplicación de empresa.

Para incorporar estas actualizaciones en el archivo `OutputFile.xml`, haga lo siguiente:

1. Vaya al siguiente directorio del WebSphere Commerce Server de destino:
   - Windows *unidad*`:\WebSphere\CommerceServer\working`
   - AIX `/usr/WebSphere/CommerceServer/working`
   - Solaris `/opt/WebSphere/CommerceServer/working`
   - 400 `/QIBM/UserData/WebCommerce/instances/` *nombreInstancia*`/working`
2. Abra el archivo `OutputFile.xml` en un editor de texto.
3. Localice el identificador `<ear-file-name>` y sustituya el valor por:
   - Windows *unidad*`:\WebSphere\CommerceServer\working\WC_Enterprise_App_`*nombreInstancia*`.ear`
   - AIX `/usr/WebSphere/CommerceServer/working/WC_Enterprise_App_`*nombreInstancia*`.ear`
   - Solaris `/opt/WebSphere/CommerceServer/working/WC_Enterprise_App_`*nombreInstancia*`.ear`
   - 400 `/QIBM/UserData/WebCommerce/instances/`*nombreInstancia*`/working/WC_Enterprise_App_`*nombreInstancia*`.ear`

4. Asimismo, debe añadir una nueva sección para cada nuevo bean que añada a la aplicación de empresa. El ejemplo siguiente muestra cómo añadir un nuevo bean, denominado “Bonus”.

Windows AIX Solaris

```
<ejb-module name="suGrupoEJB">
   <jar-file>suArchJarDesplegado.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         WebSphere Commerce Server - nombreInstancia/
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/WebSphere Commerce DB2 DataSource nombreInstancia
            </jndi-name>
         <default-user>usuario</default-user>
         <default-password>contraseña</default-password>
      </data-source>
      <enterprise-bean-binding name="nombreEnlaceBean">
         <jndi-name>nombreInstanciaNombreJNDIDeBean</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

400

```
<ejb-module name="suGrupoEJB">
   <jar-file>suArchJarDesplegado.jar</jar-file>
   <module-install-info>
      <application-server-full-name>/NodeHome:$hostName$/EJBServerHome:
         nombreInstancia - WebSphere Commerce Server/
      </application-server-full-name>
   </module-install-info>
   <ejb-module-binding>
      <data-source>
         <jndi-name>jdbc/nombreInstancia WebSphere Commerce DB2 DataSource
            </jndi-name>
         <default-user>usuario</default-user>
         <default-password>contraseña</default-password>
      </data-source>
      <enterprise-bean-binding name="nombreEnlaceBean">
         <jndi-name>nombreInstanciaNombreJNDIDeBean</jndi-name>
      </enterprise-bean-binding>
   </ejb-module-binding>
</ejb-module>
```

donde

- *suGrupoEJB* es el nombre del grupo EJB que contiene el bean que añade a la aplicación de empresa.
- *suArchJarDesplegado* es el nombre del archivo JAR que contiene el código desplegado para su grupo EJB.
- *nombreInstancia* es el nombre de la instancia de WebSphere Commerce en la que está desplegando el código.
- *usuario* es el nombre de usuario de la base de datos
- *contraseña* es la contraseña del usuario de la base de datos.
- *nombreEnlaceBean* es el nombre de enlace para su bean enterprise. Por ejemplo, para el bean denominado Bonus, es `Bonus_Binding`.
- *nombreInstanciaNombreJNDIDeBean* es el nombre JNDI del bean enterprise con la instancia de WebSphere Commerce añadida al principio. Este nombre

JNDI debe coincidir exactamente con el del bean enterprise. Un valor de ejemplo es democom/ibm/commerce/sample/objects/Bonus. En este ejemplo, el nombre de la instancia es "demo" y el nombre JNDI del bean enterprise es "com/ibm/commerce/sample/objects/Bonus". Este valor debe entrarse en una sola línea.
Puede comprobar el nombre JNDI utilizando VisualAge para Java o la Herramienta de ensamblaje de aplicaciones.
Para comprobar el nombre JNDI utilizando VisualAge para Java, haga lo siguiente:

a. Pulse con el botón derecho del ratón en el bean y seleccione **Propiedades**. Se visualiza el nombre JNDI.

Para comprobar el nombre JNDI utilizando la Herramienta de ensamblaje de aplicaciones es necesario que el nuevo bean enterprise ya se haya incorporado a la aplicación de empresa. A la Herramienta de ensamblaje de aplicaciones se accede desde el menú Herramientas de la Consola de administración de WebSphere Application Server.

Para comprobar el nombre JNDI utilizando la Herramienta de ensamblaje de aplicaciones, haga lo siguiente:

a. Abra el archivo .ear que contiene el bean.
b. Expanda el módulo EJB que contiene el bean.
c. Seleccione el bean.
d. Pulse la pestaña **Enlaces**.
   Se visualiza el nombre JNDI.

Recuerde que, aunque estos dos métodos muestran el nombre JNDI, cuando añada la información al archivo XML debe añadir el nombre de la instancia de WebSphere Commerce al principio.

Notas:

a. Los saltos de línea en las secciones anteriores sólo se muestran a efectos de presentación.
b. Asegúrese de que el valor **$hostName$** coincide con el nombre del servidor de nodos de administración actual. Asegúrese también de que no haya caracteres de retorno de carro en esta línea.
c. La especificación <application-server-full-name> no puede estar en más de una línea.
d. Si utiliza una base de datos Oracle, debe modificar la información del origen de datos. Cambie la siguiente línea sacada de la sección de código anterior:

```
<jndi-name>jdbc/WebSphere Commerce DB2 DataSource nombreInstancia
   </jndi-name>
```

por:

```
<jndi-name>jdbc/WebSphere Commerce Oracle DataSource nombreInstancia
   </jndi-name>
```

5. Si está desplegando beans de entidad de WebSphere Commerce modificados, debe actualizar las secciones correspondientes a estos beans para que reflejen los nombres de los archivos JAR que contienen el código desplegado para los beans modificados.
6. Guarde el archivo OutputFile.xml.

Integración de nuevos beans enterprise en una aplicación de empresa

Esta sección describe cómo utilizar la Herramienta de ensamblaje de aplicaciones para integrar nuevos beans enterprise en una aplicación de empresa existente.

400 Puesto que la Herramienta de ensamblaje de aplicaciones se ejecuta en la plataforma Windows, debe hacer referencia a la unidad que ha correlacionado con su IFS de iSeries cuando se le soliciten nombres de vía de acceso totalmente calificados. Se le denomina *unidad_iSeries*.

AIX Solaris Es aconsejable aumentar el valor del tamaño del almacenamiento dinámico de memoria en el archivo `assembly.sh` para evitar quedarse sin memoria al guardar archivos `.ear` nuevos o modificados.

Este archivo se encuentra en el siguiente directorio:

- AIX `/usr/WebSphere/AppServer/bin`
- Solaris `/opt/WebSphere/AppServer/bin`

Para aumentar el tamaño del almacenamiento dinámico de memoria, modifique la siguiente línea:

```
$JAVA_HOME/jre/bin/java
```

de forma que aparezca como:

```
$JAVA_HOME/jre/bin/java -mx512M
```

En este paso, se abre el archivo `.ear` de la aplicación de empresa, que se ha creado en la sección Exportación de la aplicación de empresa actual de WebSphere Commerce en la herramienta de ensamblaje de aplicaciones. Una vez abierto, efectúe las siguientes tareas para añadir el nuevo bean de entidad a la aplicación de empresa:

1. Importe el grupo EJB que contiene el nuevo bean de entidad. El archivo JAR para el nuevo grupo EJB se almacenará dentro de la sección de módulos de EJB de la aplicación de empresa.
2. Establezca la vía de acceso de clases para que el bean de entidad incluya el archivo JAR de implementación.
3. Añada el archivo JAR de implementación a la aplicación. Este archivo JAR se almacenará dentro de la sección de archivos de la aplicación de empresa.
4. Establezca la seguridad de WebSphere Application Server para los métodos contenidos en el nuevo bean de entidad.

Para integrar el nuevo grupo EJB en la aplicación de empresa, haga lo siguiente:

1. Windows AIX Solaris Haga una copia de seguridad de la aplicación de empresa actual, haciendo lo siguiente en el WebSphere Commerce Server de destino:
   a. En un indicador de mandatos, vaya al siguiente directorio:
      - Windows *unidad*`:\WebSphere\CommerceServer\working`
      - AIX `/usr/WebSphere/CommerceServer/working`
      - Solaris `/opt/WebSphere/CommerceServer/working`

b. Haga una copia del archivo WC_Enterprise_App_*nombreInstancia*.ear existente y denomínelo WC_Enterprise_App_*nombreInstancia*.ear.bak.

2. 400 Haga una copia de seguridad de la aplicación de empresa actual, haciendo lo siguiente:
   a. En un indicador de mandatos, entre lo siguiente:
   ```
   STRQSH
   cd /QIBM/UserData/WebCommerce/instances/nombreInstancia/working
   cp WC_Enterprise_App_nombreInstancia.ear
   WC_Enterprise_App_nombreInstancia.ear.bak
   ```
   b. Para evitar tiempos de espera innecesariamente largos debidos a la transferencia de datos entre la máquina cliente local y la máquina iSeries que ejecuta WebSphere Application Server, cree el siguiente directorio en la máquina cliente local y, a continuación, copie el archivo WC_Enterprise_App_*nombreInstancia*.ear en este directorio:
   unidad:\WebSphere\CommerceServer\working
   donde *unidad* es una unidad local.

   Nota: Si el directorio *unidad*:\WebSphere\CommerceServer\working no existe en su máquina local, créelo ahora.
3. Abra la Consola de administración de WebSphere Application Server.
4. En el menú **Herramientas**, seleccione **Herramienta de ensamblaje de aplicaciones**.
5. Si se abre una ventana de bienvenida, seleccione **Cancelar** para cerrarla.
6. Abra la aplicación de empresa en la que va a trabajar, haciendo lo siguiente:
   a. En el menú **Archivo**, seleccione **Abrir**.
   b. En el campo **Nombre de archivo**, entre:
      - Windows *unidad*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*nombreInstancia*.ear
      - AIX /usr/WebSphere/CommerceServer/working/WC_Enterprise_App_*nombreInstancia*.ear
      - Solaris /opt/WebSphere/CommerceServer/working/WC_Enterprise_App_*nombreInstancia*.ear
      - 400 *unidad*:\WebSphere\CommerceServer\working\WC_Enterprise_App_*nombreInstancia*.ear

      y pulse **Abrir**. Espere a que la aplicación se abra antes de continuar con los pasos siguientes. Este proceso puede tardar varios minutos.
7. Pulse con el botón derecho del ratón en **Módulos EJB** y seleccione **Importar**.
8. En el campo **Nombre de archivo**, entre lo siguiente:
   - Windows *unidad*:\WebSphere\CommerceServer\temp*suArchJarDesplegado*.jar
   - AIX /usr/WebSphere/CommerceServer/temp/ *suArchJarDesplegado*.jar
   - Solaris /opt/WebSphere/CommerceServer/temp/ *suArchJarDesplegado*.jar
   - 400 *unidad_iSeries*:\QIBM\UserData\WebCommerce\instances*nombreInstancia*\temp*suArchJarDesplegado*.jar

   donde
   - *suArchJarDesplegado* es el nombre del archivo JAR que contiene el código desplegado para su grupo EJB

- *unidad_iSeries* es la unidad local correlacionada con IFS de iSeries.

Pulse **Abrir** y, a continuación, en la ventana de confirmación de valores, pulse **Aceptar**.

9. Una vez se haya importado el archivo *suArchJarDesplegado*.jar, desplácese hasta el grupo EJB *suGrupoEJB* (donde *suGrupoEJB* es el nombre de su grupo EJB) y seleccione este grupo.
   La información sobre este grupo se muestra en el panel de la derecha.
10. En el campo de la vía de acceso de clases para el nuevo bean enterprise, entre los archivos JAR dependientes. Por ejemplo, puede entrar el archivo JAR de implementación correspondiente y el archivo JAR de implementación para los beans de entidad de WebSphere Commerce, tal como se muestra a continuación:

    lib/*suArchJarImpl*.jar lib/wcsejbimpl.jar
11. Pulse **Aplicar**.
12. Añada el archivo JAR de implementación para su grupo EJB a la aplicación, haciendo lo siguiente:
    a. Pulse con el botón derecho del ratón en el nodo **Archivos** para la aplicación de empresa y seleccione **Añadir archivos**. (El nodo **Archivos** para la aplicación de empresa se encuentra cerca del final del árbol jerárquico. Tenga en cuenta que hay otros nodos Archivos para componentes dentro de la aplicación de empresa, pero debe seleccionar el nodo Archivos para toda la aplicación.)
    b. En la ventana Añadir archivos, pulse **Examinar**.
    c. Vaya al siguiente directorio:
       - Windows *unidad*:\WebSphere\CommerceServer\temp
       - AIX /usr/WebSphere/CommerceServer/temp
       - Solaris /opt/WebSphere/CommerceServer/temp
       - 400 *unidad_iSeries*:\QIBM\UserData\WebCommerce\instances*nombreInstancia*\temp
    d. Con este directorio resaltado, pulse **Seleccionar**.
    e. Vuelva a la ventana Añadir archivos. Observará que aparece el contenido del directorio temporal. Resalte el directorio lib.
       El contenido del directorio lib se muestra en el panel de la derecha.
    f. En el panel de la derecha, seleccione el archivo *suArchJarImpl* y pulse **Añadir**. El archivo aparecerá en el panel Archivos seleccionados.
    g. Pulse **Aceptar**.
13. Configure la seguridad para su bean de entidad, haciendo lo siguiente:
    a. Con el nodo de módulos EJB expandido, localice y expanda el nodo *suGrupoEJB*.
    b. Expanda **Beans de entidad**.
    c. Expanda *suBeanEntidad* donde *suBeanEntidad* es el nombre de su bean de entidad.
    d. Pulse **Extensiones de método** y, a continuación, haga lo siguiente en el panel derecho:
       1) Pulse la pestaña **Avanzadas**.
       2) Compruebe que la opción **Identidad de seguridad** esté seleccionada.
       3) Compruebe que **Utilizar identidad de servidor EJB** esté seleccionado para cada uno de los métodos.

4) Pulse **Aplicar** (si ha efectuado modificaciones).

e. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Roles de seguridad** bajo el grupo EJB *suGrupoEJB* y seleccione **Nuevo**; a continuación, haga lo siguiente:

1) En el campo **Nombre**, entre `WCSecurityRole` y pulse **Aplicar**.Si este rol ya existe, no es necesario efectuar este paso.

f. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Permisos de métodos** bajo el grupo EJB *suGrupoEJB* y seleccione **Nuevo**; a continuación, haga lo siguiente:

1) En el campo **Nombre del permiso de métodos**, entre `WCMethodPermission`
2) En el área de selección **Métodos**, pulse **Añadir**.
Se abre la ventana Añadir métodos.
3) Expanda *suArchJarDesplegado***.jar**, luego **Bonus** y, a continuación, expanda cada una de las listas de métodos **Local** y **Remota**.
4) Mantenga pulsada la tecla de desplazamiento y seleccione todos los métodos locales; pulse **Aceptar**.
5) Repita el proceso de selección de métodos para añadir también los métodos remotos, si los hay.
6) En el área de selección de roles, pulse **Añadir**, seleccione `WCSecurityRole` y pulse **Aceptar**.
7) Pulse **Aplicar**.

14. En el menú **Archivo**, seleccione **Guardar**.
15. Cierre la Herramienta de ensamblaje de aplicaciones.
16. 400 Copie el archivo `WC_Enterprise_App_`*`nombreInstancia`*`.ear` que acaba de modificar de la máquina local a la máquina iSeries que ejecuta WebSphere Application Server. Es decir, copie el archivo del directorio siguiente:
`unidad``:\WebSphere\CommerceServer\working`

al directorio siguiente:
`unidad_iSeries``:\QIBM\UserData\WebCommerce\instances\`*`nombInstancia`*`\working`

donde *`unidad_iSeries`* es la letra de la unidad que ha correlacionado con IFS de iSeries.

Después de completar este paso, ha creado una nueva aplicación de empresa que contiene toda la lógica anterior así como la nueva lógica de negocio. Todo esto se encuentra dentro del archivo modificado recientemente `WC_Enterprise_App_`*`nombreInstancia`*`.ear`.

Integración de beans enterprise modificados en una aplicación de empresa

Esta sección describe cómo utilizar la Herramienta de ensamblaje de aplicaciones para integrar beans enterprise de WebSphere Commerce modificados en una aplicación de empresa.

En este paso, abrirá la aplicación de empresa en la herramienta de ensamblaje de aplicaciones. Una vez abierta, puede hacer lo siguiente para incluir un bean enterprise de WebSphere Commerce modificado en la aplicación de empresa:

1. Haga una copia de la vía de acceso de clases para la versión existente del grupo EJB que ha modificado.

2. Elimine la versión existente del grupo EJB que ha modificado.
3. Importe la nueva versión del grupo EJB que ha modificado. El archivo JAR para el nuevo grupo EJB se almacena dentro de la sección de módulos de EJB de la aplicación de empresa.
4. Establezca la vía de acceso de clases para el grupo EJB modificado.
5. Establezca la seguridad de WebSphere Application Server para los métodos contenidos en el bean de entidad modificado.

AIX Solaris Es aconsejable aumentar el valor del tamaño del almacenamiento dinámico de memoria en el archivo `assembly.sh` para evitar quedarse sin memoria al guardar archivos `.ear` nuevos o modificados.

Este archivo se encuentra en el siguiente directorio:

- AIX `/usr/WebSphere/AppServer/bin`
- Solaris `/opt/WebSphere/AppServer/bin`

Para aumentar el tamaño del almacenamiento dinámico de memoria, modifique la siguiente línea:

```
$JAVA_HOME/jre/bin/java
```

de forma que aparezca como:

```
$JAVA_HOME/jre/bin/java -mx512M
```

Para integrar el grupo EJB modificado en la aplicación de empresa, haga lo siguiente:

1. Windows AIX Solaris Haga una copia de seguridad de la aplicación de empresa actual, haciendo lo siguiente en el WebSphere Commerce Server de destino:
   a. En un indicador de mandatos, vaya al siguiente directorio:
      - Windows *unidad*`:\WebSphere\CommerceServer\working`
      - AIX `/usr/WebSphere/CommerceServer/working`
      - Solaris `/opt/WebSphere/CommerceServer/working`
   b. Haga una copia del archivo `WC_Enterprise_App_`*nombreInstancia*`.ear` existente y denomínelo `WC_Enterprise_App_`*nombreInstancia*`.ear.bak`.
2. 400 Haga una copia de seguridad de la aplicación de empresa actual, haciendo lo siguiente:
   a. En un indicador de mandatos, entre lo siguiente:

      ```
      STRQSH
      cd /QIBM/UserData/WebCommerce/instances/nombreInstancia/working
      cp WC_Enterprise_App_nombreInstancia.ear
      WC_Enterprise_App_nombreInstancia.ear.bak
      ```

   b. Para evitar tiempos de espera innecesariamente largos debidos a la transferencia de datos entre la máquina cliente local y la máquina iSeries que ejecuta WebSphere Application Server, cree el siguiente directorio en la máquina cliente local y, a continuación, copie el archivo `WC_Enterprise_App_`*nombreInstancia*`.ear` en este directorio:
      unidad`:\WebSphere\CommerceServer\working`
      donde *unidad* es una unidad local.

Nota: Si el directorio *unidad*:\WebSphere\CommerceServer\working no existe en su máquina local, créelo ahora.

3. Abra la Consola de administración de WebSphere Application Server.
4. En el menú **Herramientas**, seleccione **Herramienta de ensamblaje de aplicaciones**.
5. Abra la aplicación de empresa en la que va a trabajar, haciendo lo siguiente:
   a. En el menú **Archivo**, seleccione **Abrir**.
   b. En el campo **Nombre de archivo**, entre:
      - Windows *unidad*:\WebSphere\CommerceServer\working\ WC_Enterprise_App_*nombreInstancia*.ear
      - AIX /usr/WebSphere/CommerceServer/working/ WC_Enterprise_App_*nombreInstancia*.ear
      - Solaris /opt/WebSphere/CommerceServer/working/ WC_Enterprise_App_*nombreInstancia*.ear
      - 400 *unidad*:\WebSphere\CommerceServer\working\ WC_Enterprise_App_*nombreInstancia*.ear

      y pulse **Abrir**. Espere a que la aplicación se abra antes de continuar con los pasos siguientes. Este proceso puede tardar varios minutos.
6. Pulse **Módulos EJB**. El panel derecho muestra los módulos EJB en la aplicación de empresa.
7. Pulse el módulo de EJB para el grupo EJB que ha modificado. Por ejemplo, si ha modificado un bean en el grupo EJB WCSUser, debe pulsar **WCSUser**.
8. Pulse la pestaña General para ver la información de vía de acceso de clases. Copie esta información de vía de acceso de clases existente en un archivo de texto (por ejemplo, WCSUser_path.txt).
9. Pulse con el botón derecho del ratón en el módulo EJB y seleccione **Suprimir**.
10. Pulse con el botón derecho del ratón en **Módulos EJB** y seleccione **Importar**.
11. En el campo **Nombre de archivo**, entre lo siguiente:
    - Windows *unidad*:\WebSphere\CommerceServer\temp\ Cust_*nombreGrupoEJB*-ejb.jar
    - AIX /usr/WebSphere/CommerceServer/temp/Cust_*nombreGrupoEJB*-ejb.jar
    - Solaris /opt/WebSphere/CommerceServer/temp/Cust_*nombreGrupoEJB*-ejb.jar
    - 400 *unidad_iSeries*:\QIBM\UserData\WebCommerce\instances\ *nombreInstancia*\temp\Cust_*nombreGrupoEJB*-ejb.jar

    y pulse **Abrir**. En la ventana de confirmación, pulse **Aceptar**.
12. Una vez importado el archivo *archJARGrupoEJB*.jar, desplácese hasta el grupo EJB modificado y seleccione este grupo.
    En el panel derecho se muestra información sobre este grupo.
13. Abra el archivo de texto que contiene la información de vía de acceso de clases para la versión anterior del grupo EJB. Seleccione y copie la vía de acceso de clases.
14. En el campo **Vía acceso de clases** para el grupo EJB modificado, pegue la información de vía de acceso de clases.
15. Pulse **Aplicar**.

16. En el menú **Archivo**, seleccione **Cerrar**.
17. Espere a que el archivo se cierre y, a continuación, en el menú **Archivo**, seleccione **Abrir** y vuelva a abrir el archivo `WC_Enterprise_App_nombreInstancia.ear`.
18. Configure la seguridad para el bean modificado, haciendo lo siguiente:
    a. Con el nodo de módulos EJB expandido, localice y expanda el nodo para el grupo EJB modificado.
    b. Expanda **Beans de entidad**.
    c. Expanda el grupo EJB modificado.
    d. Pulse **Extensiones de método** y, a continuación, haga lo siguiente en el panel derecho:
        1) Pulse la pestaña **Avanzadas**.
        2) Compruebe que la opción **Identidad de seguridad** esté seleccionada.
        3) Compruebe que **Utilizar identidad de servidor EJB** esté seleccionado para cada uno de los métodos.
        4) Pulse **Aplicar** (si ha efectuado modificaciones).
    e. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Roles de seguridad** bajo el grupo EJB modificado y seleccione **Nuevo**; a continuación, haga lo siguiente:
        1) En el campo **Nombre**, entre `WCSecurityRole` y pulse **Aplicar**.Si este rol ya existe, no es necesario efectuar este paso.
    f. En el panel de navegación izquierdo, pulse con el botón derecho del ratón en **Permisos de métodos** bajo el grupo EJB modificado y seleccione **Nuevo**; a continuación, haga lo siguiente:
        1) En el campo **Nombre del permiso de métodos**, entre `WCMethodPermission`
        2) En el área de selección **Métodos**, pulse **Añadir**. Se abre la ventana Añadir métodos.
        3) Expanda el *grupoEJBmodificado* y seleccione todos los beans enterprise (mantenga pulsada la tecla de desplazamiento mientras efectúa la selección). Pulse **Aceptar**. A continuación, bajo la columna Bean enterprise aparecen todos los beans enterprise y bajo la columna Tipos aparecen todos los métodos.
        4) En el área de selección de roles, pulse **Añadir**, seleccione `WCSecurityRole` y pulse **Aceptar**.
        5) Pulse **Aplicar**.
19. En el menú **Archivo**, seleccione **Guardar**.
20. Cierre la Herramienta de ensamblaje de aplicaciones.
21. 400 Copie el archivo `WC_Enterprise_App_nombreInstancia.ear` que acaba de modificar de la máquina local a la máquina iSeries que ejecuta WebSphere Application Server. Es decir, copie el archivo del directorio siguiente:

    `unidad:\WebSphere\CommerceServer\working`

    al directorio siguiente:

    `unidad_iSeries:\QIBM\UserData\WebCommerce\instances\nombInstancia\working`

    donde *unidad_iSeries* es la letra de la unidad que ha correlacionado con IFS de iSeries.

Después de completar este paso, ha creado una nueva aplicación de empresa que contiene toda la lógica anterior así como la nueva lógica de negocio. Todo esto se encuentra dentro del archivo modificado recientemente `WC_Enterprise_App_nombreInstancia.ear`.

Detención y supresión de una aplicación de empresa

Esta sección describe cómo utilizar la Consola de administración de WebSphere Application Server para detener una aplicación de empresa que se esté ejecutando actualmente y eliminarla.

Para detener y eliminar la aplicación de empresa, haga lo siguiente:

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**.
3. Expanda **Nodos**.
4. Expanda ***nombreNodo*** (donde ***nombreNodo*** es el nombre de su nodo).
5. Expanda **Servidores de aplicaciones**.
6. Windows AIX Solaris Pulse con el botón derecho del ratón en su servidor de aplicaciones de WebSphere Commerce.Por ejemplo, pulse con el botón derecho del ratón en **WebSphere Commerce Server - *nombreNodo*** y seleccione **Detener**.
7. 400 Pulse con el botón derecho del ratón en su servidor de aplicaciones de WebSphere Commerce.Por ejemplo, pulse con el botón derecho del ratón en ***nombreInstancia*- WebSphere Commerce Server** y seleccione **Detener**.
8. Expanda **Aplicaciones de empresa**.
9. Windows AIX Solaris Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Enterprise Application *-nombreInstancia*** y seleccione **Detener**.
10. 400 Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación ***nombreInstancia* - WebSphere Commerce Enterprise Application** y seleccione **Detener**.
11. Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación **WebSphere Commerce Enterprise Application *-nombreInstancia*** (o ***nombreInstancia* - WebSphere Commerce Enterprise Application** para iSeries) y seleccione **Eliminar**.
12. Cuando se le solicite que indique si la aplicación debe exportarse, seleccione **No**.
13. Cuando se le solicite que indique si la aplicación debe eliminarse, seleccione **Sí**.

Importación de una aplicación de empresa

Esta sección describe cómo utilizar el programa de utilidad de línea de mandatos XMLConfig para importar una aplicación de empresa.

Para importar la nueva aplicación de empresa, haga lo siguiente:

1. Windows AIX Solaris Invoque la herramienta XMLConfig para importar la aplicación de empresa a WebSphere Application Server entrando el siguiente mandato:

   Windows

   ```
   xmlConfig -import OutputFile.xml -adminNodeName sistpralWas
   ```

   AIX

   ```
   /usr/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
    -adminNodeName sistpralWas
    -nameServiceHost sistpralWas -nameServicePort
   puertoAdminWas
   ```

   Solaris

   ```
   /opt/WebSphere/AppServer/bin/XMLConfig.sh -import OutputFile.xml
    -adminNodeName sistpralWas
    -nameServiceHost sistpralWas -nameServicePort
   puertoAdminWas
   ```

   donde

   - *sistpralWas* es el nombre del nodo en el WebSphere Application Server que contiene la aplicación de empresa actual.
   - *OutputFile.xml* es el archivo XML que describe todos sus beans enterprise.
   - *puertoAdminWas* es el puerto de administración de WebSphere Application Server.

2. 400 Invoque la herramienta XMLConfig para importar la aplicación de empresa a WebSphere Application Server entrando el siguiente mandato:

   ```
   STRQSH
   PATH=/QIBM/ProdData/WebASAdv4/bin:$PATH
   xmlConfig -import
   /QIBM/UserData/WebCommerce/instances/nombreInstancia/working/OutputFile.xml
   -adminNodeName sistpralWas
    -nameServiceHost sistpralWas -nameServicePort puertoAdminWas
   -instance nombreInstanciaWas
   ```

   donde

   - *OutputFile.xml* es el nombre totalmente calificado del archivo XML que describe todos sus beans enterprise.
   - *sistpralWas* es el nombre del nodo en el WebSphere Application Server que contiene la aplicación de empresa actual.

     Nota: 400 El valor para *sistpralWas* es sensible a las mayúsculas y minúsculas y debe coincidir con el valor que hay en la configuración de TCP/IP. (Utilice un procesador de línea de mandatos para acceder a CFGTCP, opción 12 para verificar el nombre de sistema principal.)
   - *puertoAdminWas* es el puerto de administración de WebSphere Application Server.
   - *nombreInstanciaWas* es el nombre de la instancia de WebSphere Application Server.

400 Mientras ejecuta el mandato `XMLConfig -import`, quizá reciba el siguiente mensaje de error:"No se puede expandir el archivo ear bajo `/QIBM/UserData/WebAsAdv4/nombreInstanciaWas/installedApps/WC_Enterprise_App_nombreInstancia.ear`". Si recibe este mensaje, elimine o renombre el directorio anterior y ejecute el mandato de nuevo.

3. 400 Después de importar la aplicación de empresa, debe ejecutar un script para modificar los permisos de directorio. Para ejecutar este script, haga lo siguiente:
   - En un indicador de mandatos, entre lo siguiente:

     ```
     STRSQH
     cd /QIBM/ProdData/WebCommerce/bin
     changeAuthority nombreInstanciaAdminWas
     nombreInstancia
     ```

     donde
     - *nombreInstanciaAdminWas* es el nombre de la instancia de administración de WebSphere Application Server.
     - *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.

Inicio de una aplicación de empresa

Esta sección describe cómo utilizar la Consola de administración de WebSphere Application Server para renovar la vista y, a continuación, iniciar una aplicación de empresa.

1. Abra la Consola de administración de WebSphere Application Server.
2. Expanda **Dominio de Administración de WebSphere**, luego **Nodos** y, finalmente, *nombreNodo*
3. Resalte el nodo *nombreNodo*.
4. Pulse el icono **Renovar subárbol seleccionado**.
5. Inicie la aplicación de WebSphere Commerce, haciendo lo siguiente:
   - Expanda **Servidores de aplicaciones**.
   - Windows AIX Solaris Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en **WebSphere Commerce Server -** *nombreInstancia* y seleccione **Iniciar**.
   - 400 Pulse con el botón derecho del ratón en la aplicación de WebSphere Commerce. Por ejemplo, pulse con el botón derecho del ratón en la aplicación *nombreInstancia* **- WebSphere Commerce Server** y seleccione **Iniciar**.

Apéndice C. Consejos para VisualAge para Java

En esta sección se ofrecen algunos consejos relacionados con la resolución de problemas, las mejoras en el rendimiento y la simplificación dentro del entorno de desarrollo.

Cambio de las propiedades del motor de servlets en el Entorno de prueba WebSphere

Las propiedades del motor de servlets en el Entorno de prueba WebSphere las controla el archivo de propiedades `default.servlet_engine`. En este archivo, puede modificar el directorio raíz de documentos para el servidor Web así como cambiar el puerto utilizado por el Entorno de prueba WebSphere.

Puede que desee cambiar el puerto en el caso de que el Entorno de prueba WebSphere haya dejado de funcionar y rearrancar el sistema no sea una opción válida. Para cambiar el puerto, haga lo siguiente:

1. Abra el archivo *vía_instalación_VAJ*`\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine` con un editor de texto.
2. En la sección `<transport>`, sustituya el valor `8080`, que aparece en la línea siguiente, por un valor de puerto disponible.

   ```
   <arg name="port" value="8080"/>
   ```

3. Sustituya el valor `8080`, que aparece en las líneas siguientes, por el mismo puerto que ha especificado anteriormente.

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```

4. Guarde el archivo.
5. Abra el Centro de control del Entorno de prueba WebSphere e inicie el motor de servlets.

Por omisión, el directorio raíz de documentos para el Entorno de prueba WebSphere es *vía_instalación_VAJ*`\IDE\ProjectResources\IBM WebSphere Test Environment\hosts\default_host \default_app\web`. Para cambiarlo por otro directorio, haga lo siguiente:

1. :NONE.Abra el Centro de control del Entorno de prueba WebSphere y detenga el motor de servlets.:NONE.
2. Abra el archivo *vía_instalación_VAJ*`\IDE\ProjectResources\IBM WebSphere Test Environment\properties\default.servlet_engine` con un editor de texto.
3. En la sección `<websphere-webgroup name="default_app">`, sustituya `<document-root>$approot$/web</document-root>` por lo siguiente:

   ```
   <document-root>su_directorio_raíz_de_documentos</document-root>
   ```

   donde *su_directorio_raíz_de_documentos* es el directorio raíz de documentos que desea.
4. Sustituya el valor `8080`, que aparece en las líneas siguientes, por el mismo puerto que ha especificado anteriormente.

   ```
   <hostname-binding hostname="localhost:8080" servlethost="default_host"/>
      <hostname-binding hostname="127.0.0.1:8080" servlethost="default_host"/>
   ```

5. Guarde el archivo.

6. Abra el Centro de control del Entorno de prueba WebSphere e inicie el motor de servlets.

Resolución de problemas del servidor de nombres persistentes

Si está experimentando problemas con el servidor de nombres persistentes, es posible que tenga que descartar y volver a crear la base de datos del servidor de nombres persistentes. Si desea información detallada sobre cómo crear esta base de datos, consulte la publicación *Commerce Studio, Guía de instalación*.

De forma alternativa, si aparece un mensaje indicando que el puerto está actualmente en uso, asegúrese de que WebSphere Application Server no se esté ejecutando.

Supresión de archivos JSP compilados

Por motivos de rendimiento, VisualAge para Java mantiene una carpeta de proyecto en la que se almacenan los archivos JSP compilados. Es posible que a veces sea necesario suprimir archivos JSP compilados. Por ejemplo, si suprime un bean de datos de un archivo JSP, la próxima vez que llame al archivo JSP es posible que se produzcan errores. En este caso, puede suprimir el archivo JSP compilado.

Para suprimir archivos JSP compilados, haga lo siguiente:

1. En VisualAge para Java, seleccione la pestaña Proyectos.
2. Desplácese hacia abajo hasta el proyecto **JSP Page Compile Generate Code**.
3. Seleccione todo el proyecto (si necesita suprimir muchos archivos JSP compilados) o expanda el proyecto y suprima sólo los que se deben volver a compilar.

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en los Estados Unidos. Es posible que IBM no proporcione los productos, servicios o características a los que hace referencia este documento en otros países. Póngase en contacto con su representante de IBM local para obtener información acerca de los productos y servicios disponibles actualmente en su área. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implica que sólo pueda utilizarse ese producto, programa o servicio de IBM. En su lugar puede utilizarse cualquier producto, programa o servicio funcionalmente equivalente que no vulnere ninguno de los derechos de propiedad intelectual de IBM. No obstante, 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 patente pendientes que cubran temas tratados en este documento. La entrega de este documento no le otorga ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Para realizar consultas sobre la licencia en relación a información de doble byte (DBCS), póngase en contacto con el departamento de propiedad intelectual de IBM en su país o envíe sus consultas, por escrito, a:

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

El párrafo siguiente no es aplicable al Reino Unido ni a ningún otro país donde las disposiciones en él expuestas sean incompatibles con la legislación local:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍAS DE NINGUNA CLASE, NI EXPLÍCITAS NI IMPLÍCITAS, INCLUIDAS, PERO SIN LIMITARSE A, LAS GARANTÍAS IMPLÍCITAS DE NO INFRACCIÓN, COMERCIALIZACIÓN O IDONEIDAD PARA UNA FINALIDAD DETERMINADA. Algunas legislaciones no contemplan la exclusión de garantías, ni implícitas ni explícitas, en determinadas transacciones, por lo que puede haber usuarios a los que no les afecte dicha norma.

Esta información puede contener imprecisiones técnicas o errores tipográficos. La información aquí contenida está sometida a cambios periódicos; tales cambios se irán incorporando en nuevas ediciones de la publicación. IBM se reserva el derecho de realizar cambios y/o mejoras, cuando lo considere oportuno y sin previo aviso, en los productos y/o programas descritos en esta publicación.

Todas las referencias hechas en este documento a sitios Web que no son de IBM se proporcionan únicamente para su información y no representan en modo alguno

una recomendación de dichos sitios Web. El contenido de esos sitios Web no forma parte del contenido de este producto de IBM, por lo que la utilización de dichos sitios es responsabilidad del usuario.

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

Los licenciatarios de este programa que deseen obtener información sobre el mismo con el fin de permitir: (i) el intercambio de información entre programas creados independientemente y otros programas (incluido éste) y (ii) el uso mutuo de la información que se ha intercambiado, deberán ponerse en contacto con:

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue, Markham, Ontario L6G 1C7
Canadá

Dicha información puede estar disponible sujeta a los términos y condiciones apropiados, incluyendo, en algunos casos, el pago de una cantidad.

IBM proporciona el programa bajo licencia descrito en esta información, y todo el material bajo licencia disponible para el mismo, bajo los términos del Contrato de cliente IBM, el Acuerdo Internacional de Programas bajo Licencia IBM, o de cualquier acuerdo equivalente entre IBM y el cliente.

Todos los datos de rendimiento incluidos en este documento han sido determinados en un entorno controlado. Por consiguiente, los resultados obtenidos en otros entornos operativos pueden variar de forma significativa. Algunas mediciones pueden haberse realizado en sistemas de nivel de desarrollo y no hay ninguna garantía de que estas mediciones sean las mismas en sistemas de uso general. Asimismo, algunas mediciones se pueden haber estimado mediante extrapolación. Los resultados reales pueden variar. Los usuarios de este documento deben verificar qué datos son aplicables a su entorno específico.

La información sobre productos que no son de IBM se ha obtenido de los distribuidores de dichos productos, de los anuncios publicados o de otras fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la precisión del rendimiento, la compatibilidad ni ninguna otra afirmación relacionada con productos que no son de IBM. Las preguntas sobre las prestaciones de productos no de IBM deben dirigirse a los distribuidores de dichos productos.

Todas las declaraciones sobre futuras tendencias o intenciones de IBM están sujetas a modificación o retirada sin previo aviso y representan únicamente metas y objetivos.

Todos los precios IBM mostrados son precios al por menor sugeridos por IBM, son actuales y están sujetos a cambios sin previo aviso. Los precios del comerciante pueden variar.

Esta información se proporciona únicamente con fines de planificación. Está sujeta a posibles cambios antes de que los productos que en ella se describen estén disponibles.

Esta información contiene ejemplos de datos e informes que se utilizan en operaciones comerciales cotidianas. Para ilustrar los ejemplos de la forma más

completa posible, éstos incluyen nombres de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier parecido con nombres y direcciones utilizados en empresas reales es pura coincidencia.

LICENCIA DE COPYRIGHT:

Esta información contiene programas de aplicación de ejemplo en lenguaje fuente que ilustran las técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir libremente estos programas de ejemplo, sin pagar por ello a IBM, con la finalidad de desarrollar, utilizar, comercializar o distribuir programas de aplicación conformes a la interfaz de programas de aplicación para la plataforma operativa para la cual están escritos los programas de ejemplo. Estos ejemplos no han sido probados en profundidad bajo todas las condiciones. En consecuencia, IBM no puede garantizar ni afirmar la fiabilidad, solidez o funcionalidad de estos programas. Puede copiar, modificar y distribuir libremente estos programas de ejemplo, sin pagar por ello a IBM, con la finalidad de desarrollar, utilizar, comercializar o distribuir programas de aplicación conformes a las interfaces de programas de aplicación de IBM.

Cada copia, parcial o completa, de estos programas de ejemplo, o cualquier trabajo obtenido a partir de los mismos, debe incluir el siguiente aviso de copyright:

Si examina esta información en formato de copia software, es posible que algunas fotografías o ilustraciones a color no aparezcan.

Marcas registradas y marcas de servicio

Los siguientes términos son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países:

400	iSeries
AIX	MQSeries
AS/400	Net.Commerce
CICS	Net.Data
DB2	VisualAge
IBM	WebSphere

Windows y Windows NT son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Oracle es una marca registrada de Oracle Corporation en los Estados Unidos y/o en otros países.

Solaris, Java y todas las marcas y logotipos basados en Java son marcas comerciales o marcas registradas de Sun Microsystems, Inc., en los Estados Unidos y/o en otros países.

Otros nombres de empresas, productos y servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice

A

acuerdos de comercio 125
adaptadores 8
ámbito de transacción 115
arquitectura de aplicación 4
arquitectura de ejecución 6

B

bean de datos invocador de mandato de controlador 37
beans de datos
  activar 37
  BeanInfo 37
  descripción 12
  interfaces 35
    bean de datos de entrada 36
    bean de datos de mandato 36
    bean de datos inteligente 35
  personalizar existentes 123
  tipos 34
beans de entidad
  ampliación 44
  antememoria 66
  descripción 11
  descriptores de despliegue 42
  transacciones 65
  utilizar 68
  visión general 41
beans de sesión
  escribir nuevos 63
  uso recomendado 47
bloqueos de base de datos 65

C

ciclos de vida de objetos 64
CMDREG 26
código personalizado
  crear paquetes 107
  despliegue 159
componentes de software 3
compromisos de base de datos 115
consideraciones sobre la base de datos
  denominación 69
  tipo de datos 71
control de acceso 75
  a nivel de mandato 84
  a nivel de recurso 84
  interfaz de agrupación 88
  interfaz protegible 88
  políticas 77
  protección de recursos 89
Controlador Web 10
crear paquetes de código personalizado 107

D

depósito de código 158
descriptores de despliegue 42
despliegue de
  beans de entidad modificados 163
  beans de entidad nuevos 161
  mandatos y beans de datos modificados 163
  mandatos y beans de datos nuevos 160
diferencias de la Versión 4.1 14

E

EJB, código desplegado 159
entorno de desarrollo 157
escuchas de protocolo 8

F

flujo de mandatos 23

G

grupos de relaciones 81

M

mandatos
  contexto de mandatos 108
  escribir nuevos mandatos de controlador 109
  escribir nuevos mandatos de políticas de negocio 130
  escribir nuevos mandatos de tareas 117
  fábrica 23
  implementación 105
  infraestructura 21
  interfaces 21
  personalizar existentes 118
  registro 25
  tipos 10
mandatos de controlador
  escribir nuevos 109
  larga ejecución 112
  personalizar existentes 118
mandatos de tarea
  escribir nuevos 117
  personalizar existentes 122
mandatos de vista
  formato de propiedades de entrada 112
  propiedades necesarias 40
manejo de errores 95
  en código personalizado 97
  flujo 96
  JSP 103
  mandato 95
manejo de errores *(continuación)*
  rastreo 102
  tipos de excepción 95
mensajes
  archivos de propiedades 96
  crear mensajes 99
método flushRemote 66
metodologías para la ampliación del modelo de objeto 44
modifyIsolationLevel, mandato 294
motor de servlets 7

N

niveles de aislamiento de transacción 43

P

patrón de diseño de mandatos 20
patrón de diseño de modelo-vista-controlador 19
patrón de diseño de visualización 33
patrones de diseño 19
  de mandatos 20
  de modelo-vista-controlador 19
  de visualización 33
persistencia 41
plantillas JSP 12
  establecer atributos 38

R

rastreo del flujo de ejecución 102
registro de mandatos 25

T

términos y condiciones 135

U

URLREG 25

V

VIEWREG 29

Número Pieza: CT024ES

(1P) P/N: CT024ES

GC10-3831-00