IBM WebSphere Commerce

Guías de programación y aprendizaje

Versión 5.5

IBM WebSphere Commerce

Guías de programación y aprendizaje

Versión 5.5

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

Primera edición, primera revisión (septiembre de 2003)

Esta edición se aplica a IBM WebSphere Commerce Business Edition Versión 5.5, IBM WebSphere Commerce - Express Versión 5.5 e IBM WebSphere Commerce Professional Edition Versión 5.5 (número de producto 5724-A18), y a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones.

También se aplica a todos los releases y modificaciones posteriores hasta que se indique lo contrario en nuevas ediciones. Asegúrese de que está utilizando la edición correcta para el nivel del producto.

Puede solicitar publicaciones a través del representante de IBM o de la sucursal local de IBM.

IBM agradece sus comentarios. Puede enviar sus comentarios utilizando el formulario de opinión del lector sobre la documentación de IBM WebSphere Commerce, que está disponible en el siguiente URL:

http://www.ibm.com/software/webservers/commerce/rcf.html

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

Antes de empezar

La publicación *WebSphere Commerce, Guías de programación y aprendizaje* 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 lógica de negocio nueva
- Modificación de un mandato de controlador existente
- Ampliación del modelo de objeto y modificación de un mandato de tarea existente
- Ampliación de un bean de entidad de WebSphere Commerce existente

Resumen de cambios

La versión más reciente de este documento está disponible como archivo PDF en la biblioteca técnica de WebSphere Commerce.

Las actualizaciones realizadas desde la última versión de este documento se identifican mediante esta imagen: 

para indicar información específica de IBM WebSphere Commerce - Express.

Actualizaciones realizadas en este manual

Las copias de este manual así como cualquier versión actualizada del mismo están disponibles como archivos PDF en la sección de Biblioteca técnica del sitio Web de WebSphere Commerce:

http://www.ibm.com/software/commerce/library/

Para obtener información de soporte adicional, consulte el sitio de Soporte de WebSphere Commerce:

http://www.ibm.com/software/commerce/support/

Las versiones actualizadas de este manual también están disponibles en la Zona de de WebSphere Commerce en el dominio del desarrollador de WebSphere que está en el siguiente sitio Web:

http://www.ibm.com/websphere/developer/zones/commerce

Convenios utilizados en este manual

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 un consejo: información adicional que puede ayudarle a realizar una tarea.

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

AIX indica información específica para WebSphere Commerce para AIX.

Linux indica información específica para WebSphere Commerce para Linux.

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

Windows indica información específica para WebSphere Commerce para Windows 2000.

DB2 indica información específica para DB2 Universal Database

Oracle indica información específica para Oracle. Si utiliza WebSphere Commerce Business Edition o WebSphere Commerce Professional Edition, puede utilizar Oracle como sistema de gestión de base de datos.

Business indica información específica para IBM WebSphere Commerce Business Edition.

Express indica información específica para IBM WebSphere Commerce - Express.

Professional indica información específica para IBM WebSphere Commerce Professional Edition.

Developer indica información específica para el entorno de desarrollo de WebSphere Commerce. Para WebSphere Commerce Business Edition y WebSphere Commerce Professional Edition, el entorno de desarrollo es WebSphere Commerce Studio, Versión 5.5.

Para WebSphere Commerce - Express, el entorno de desarrollo es WebSphere Commerce - Express Developer Edition, Versión 5.5

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 EnterpriseJavaBeans
- Tecnología JavaServer Pages
- HTML
- Tecnología de base de datos
- Business Professional WebSphere Studio Application Developer, Versión 5

  Express WebSphere Studio Application Developer, Versión 5.1

Variables de vía de acceso

Esta guía utiliza las variables siguientes para representar vías de acceso de directorio:

dir_instal_WC
: Es el directorio de instalación de WebSphere Commerce. A continuación, se indican los directorios de instalación por omisión para WebSphere Commerce en diversos sistemas operativos:

  - 400 /QIBM/ProdData/CommerceServer55
  - AIX /usr/WebSphere/CommerceServer55
  - Linux /opt/WebSphere/CommerceServer55
  - Solaris /opt/WebSphere/CommerceServer55
  - Windows C:\Archivos de programa\WebSphere\CommerceServer55

dirusuario_WC
: Directorio para todos los datos utilizados por WebSphere Commerce que se pueden modificar o que necesitan que el usuario los configure.

  - 400 /QIBM/UserData/CommerceServer55

dir_instal_WAS
: Es el directorio de instalación de WebSphere Application Server. A continuación, se indican los directorios de instalación por omisión para WebSphere Application Server en diversos sistemas operativos:

  - 400 /QIBM/ProdData/WebAs5/Base
  - AIX /usr/WebSphere/AppServer
  - Linux /opt/WebSphere/AppServer
  - Solaris /opt/WebSphere/AppServer
  - Windows C:\Archivos de programa\WebSphere\AppServer

dirusuario_WAS
: Directorio para todos los datos utilizados por WebSphere Application Server que se pueden modificar o que necesitan que el usuario los configure.

- 400 QIBM/UserData/WebAS5/Base/*nombreInstancia_WAS* y *nombre_instancia_WAS* representan el nombre del WebSphere Application Server con el que está asociada la instancia de WebSphere Commerce.

dir_instal_WCDE
Directorio de instalación de el Entorno de desarrollo de WebSphere Commerce. Para WebSphere Commerce Business Edition y WebSphere Commerce Professional Edition, el entorno de desarrollo es WebSphere Commerce Studio, Versión 5.5. El directorio de instalación por omisión es el siguiente:
C:\WebSphere\CommerceStudio55.

Para WebSphere Commerce - Express, el entorno de desarrollo es WebSphere Commerce - Express Developer Edition, Versión 5.5. El directorio de instalación por omisión es el siguiente:
C:\WebSphere\CommerceDev55

Dónde encontrar más información

Para obtener más información relacionada con WebSphere Commerce, consulte el sitio Web siguiente:

http://www.ibm.com/software/commerce/library/

Contenido

Parte 1. Conceptos y arquitectura

Capítulo 1. Visión general

Componentes de software de WebSphere Commerce

Antes de examinar cómo funciona WebSphere Commerce Server, es útil consultar la ilustración de los componentes de software que están relacionados con WebSphere Commerce. El diagrama siguiente muestra una vista simplificada de estos productos de software:

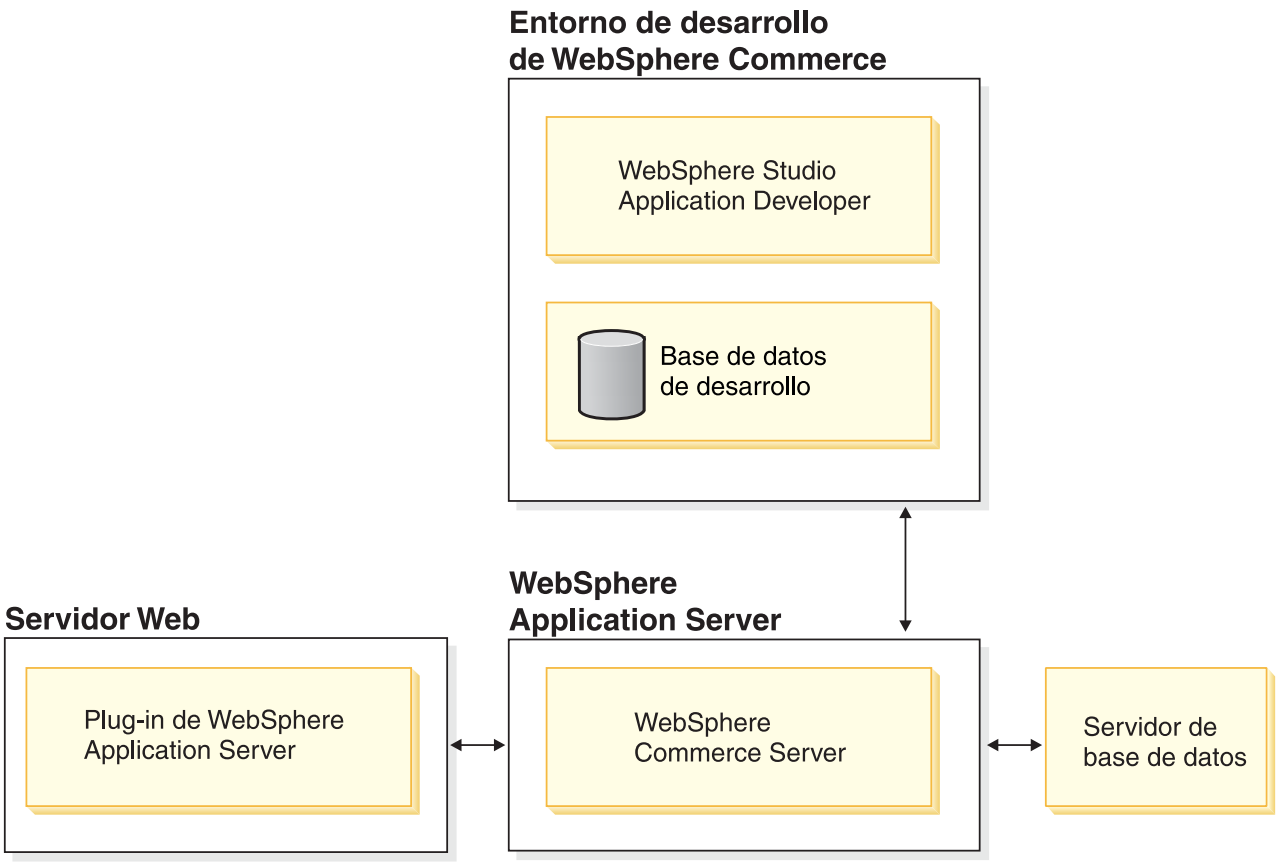


Figura 1.

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 eficiente con WebSphere Application Server, utiliza el plug-in de WebSphere Application Server.

WebSphere Commerce Server se ejecuta en WebSphere Application Server, lo que le permite aprovechar muchas de las características 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 extensiones de 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 WebSphere Studio Application Developer para realizar las tareas siguientes:

- crear y personalizar elementos de escaparate, por ejemplo plantillas JSP y páginas HTML
- crear una nueva lógica de negocio en Java
- modificar la lógica de negocio existente en Java
- probar los elementos de escaparate y el código

El entorno de desarrollo de WebSphere Commerce utiliza una base de datos de desarrollo. Los desarrolladores pueden utilizar sus herramientas de base de datos preferidas (incluido WebSphere Studio Application Developer) para realizar modificaciones de base de datos.

Arquitectura de la aplicación WebSphere Commerce

Ahora que ha visto cómo se ajustan entre sí los diferentes componentes de software relacionados con WebSphere Commerce, 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:

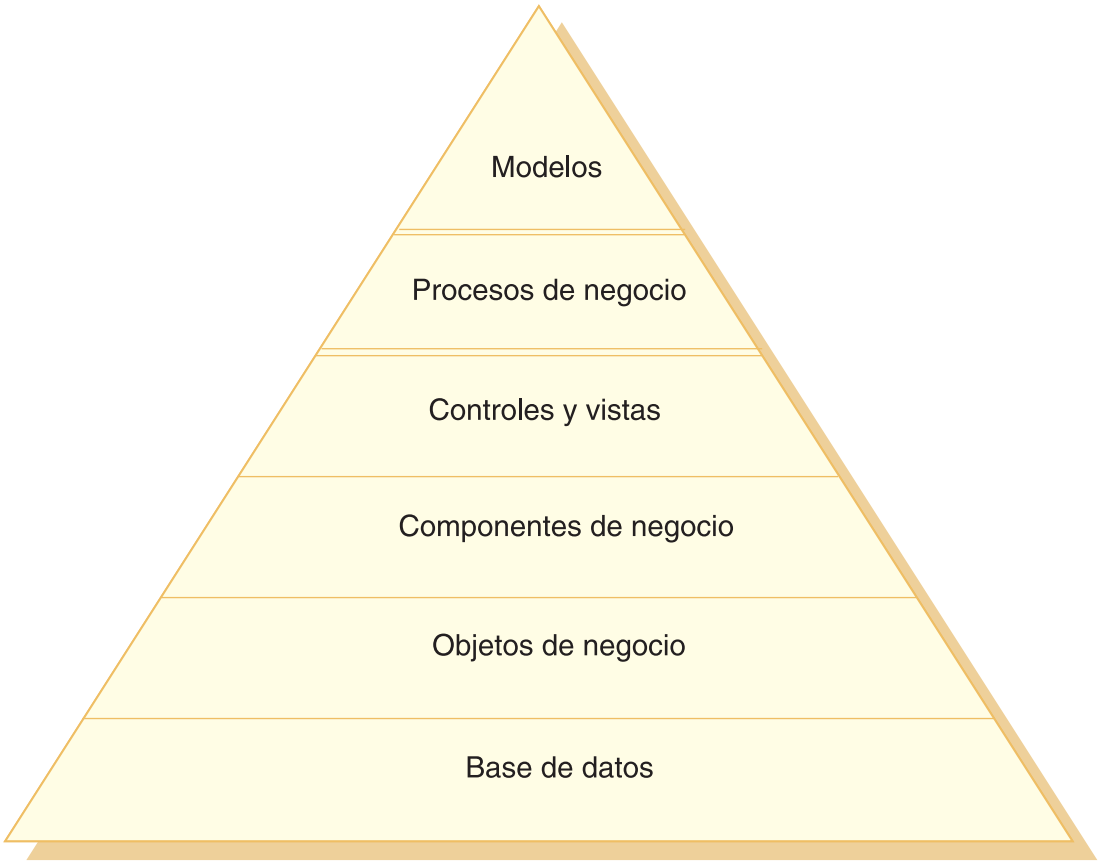


Figura 2.

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:

- USERS
- ORDERS
- INVENTORY

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 interfaz entre los componentes de negocio 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 ProductDisplayView (devuelve una página de producto que muestra la información relativa al producto seleccionado por el comprador) y OrderCancelView.

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:

Crear una campaña de correo electrónico

Este proceso de negocio incluye los componentes de negocio y las vistas relacionadas con todos los pasos implicados en el proceso de creación de campañas de correo electrónico.

Preparar un catálogo en línea

Este proceso de negocio incluye los componentes de negocio y los subprocesos relacionados con la creación de un catálogo en línea. Esto incluye el diseño del catálogo, la carga de datos del catálogo,

la creación de asociaciones de comercialización y el establecimiento de la información de fijación de precios.

Modelos

El conjunto de capas inferiores del diagrama conforman los modelos de negocio de comercio electrónico. Un ejemplo de modelo de negocio de comercio electrónico es el modelo directo al consumidor que se visualiza en la tienda de ejemplo FashionFlow. Business Otro ejemplo es el modelo directo a B2B que se visualiza 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

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

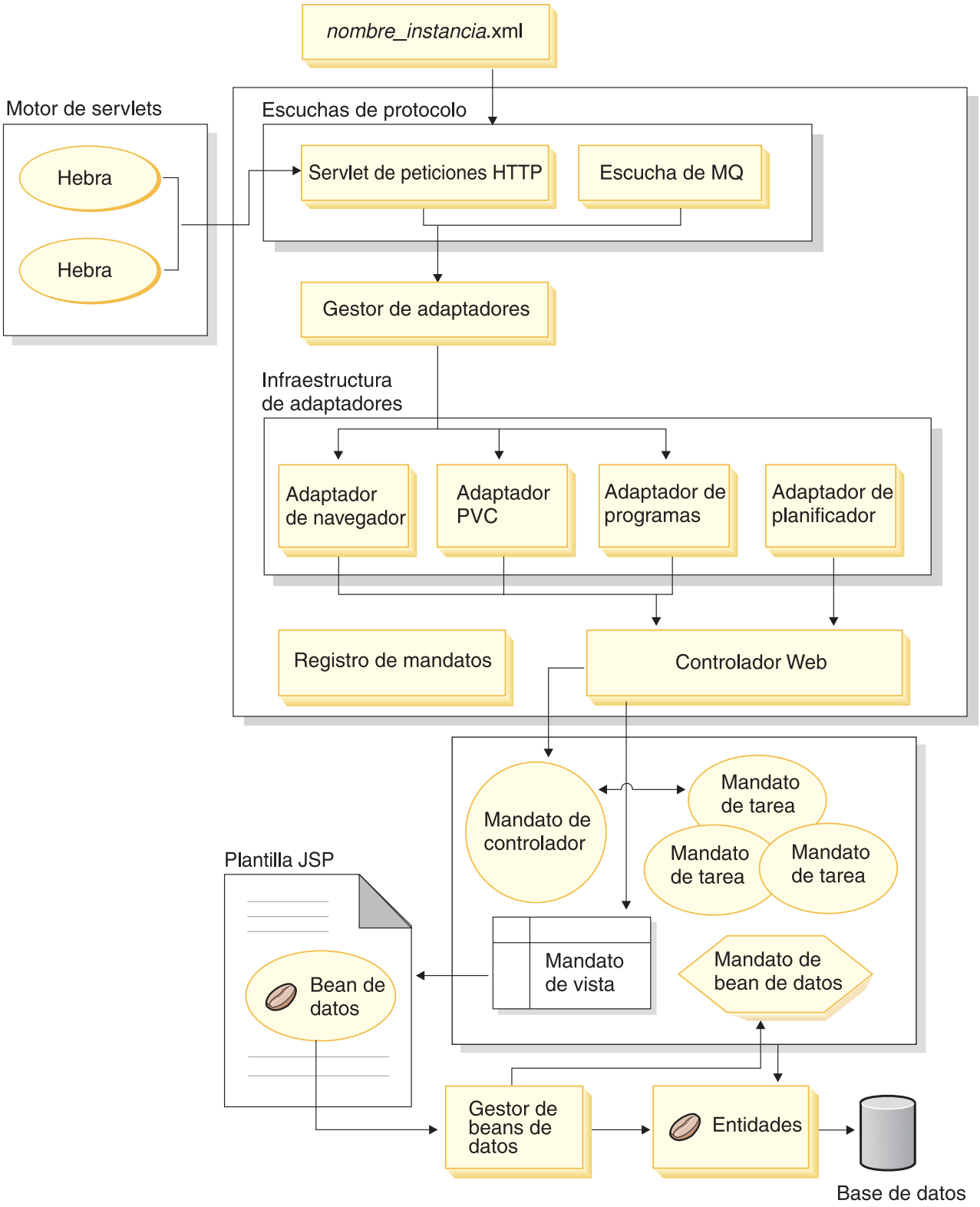

nombre_instancia.xml
Motor de servlets
Escuchas de protocolo
Hebra
Hebra
Servlet de peticiones HTTP
Escucha de MQ
Gestor de adaptadores
Infraestructura
de adaptadores
Adaptador
de navegador
Adaptador
PVC
Adaptador de
programas
Adaptador de
planificador
Registro de mandatos
Controlador Web
Mandato de
controlador
Mandato
de tarea
Mandato
de tarea
Mandato
de tarea
Plantilla JSP
Bean de
datos
Mandato
de vista
Mandato de
bean de datos
Gestor de
beans de
datos
Entidades
Base de datos

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.

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 entonces asigna las peticiones a los adaptadores apropiados, basándose en 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 (donde *nombre_instancia* es el nombre de la instancia de WebSphere Commerce). 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.

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.

Adaptadores

Los adaptadores de WebSphere Commerce son componentes específicos de 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.

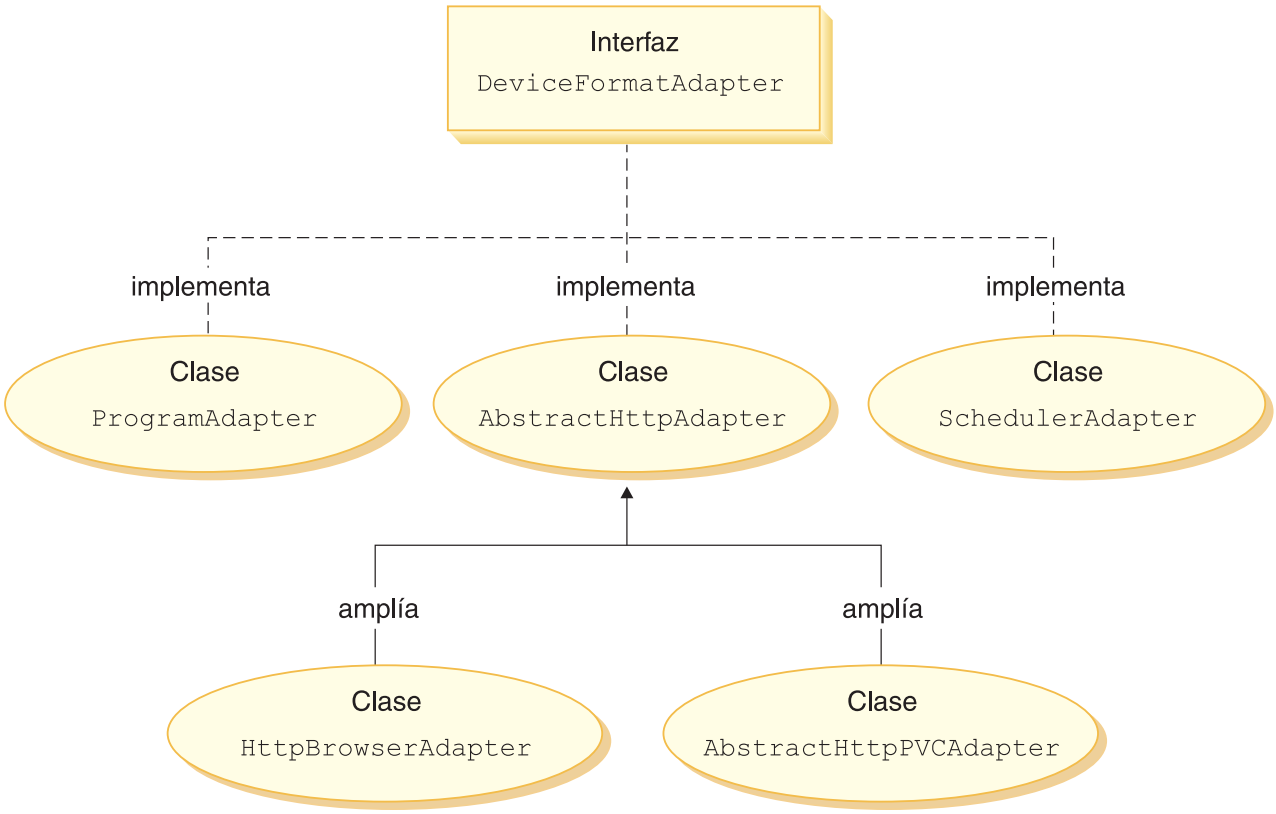


Figura 4.

Tal como se muestra en el diagrama anterior, todos los adaptadores implementan la interfaz DeviceFormatAdapter. A continuación se describen los adaptadores utilizados por 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. Los mandatos de controlador incluyen, por ejemplo, el mandato OrderProcessCmd para el proceso de pedidos y el mandato LogonCmd que permite a los usuarios iniciar la sesión. 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
El gestor de beans de datos invoca un mandato de bean de datos cuando se 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 45.

Beans de datos

Los beans de datos son beans Java utilizados principalmente por los diseñadores Web. Generalmente proporcionan acceso a una entidad de WebSphere Commerce. Un diseñador 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 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 Web comprenda el funcionamiento del bean.

Gestor de beans de datos

Los beans de datos de WebSphere Commerce insertados en las plantillas JSP permiten incluir contenido dinámico en la página. El gestor de beans de datos activa el bean de datos para que sus valores se llenen de datos cuando se inserte la siguiente línea de código en la página:

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

donde *bean_datos* es el bean de datos que se debe activar y *petición* es un objeto HTTPServletRequest.

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 cómo proteger las plantillas JSP y otros recursos, consulte el Capítulo 4, “Control de acceso”, en la página 81.

Archivo de configuración *nombre_instancia*.xml

El archivo de configuración *nombre_instancia*.xml (donde *nombre_instancia* es el nombre de la instancia de WebSphere Commerce) establece la información de configuración para la instancia de WebSphere Commerce. 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.

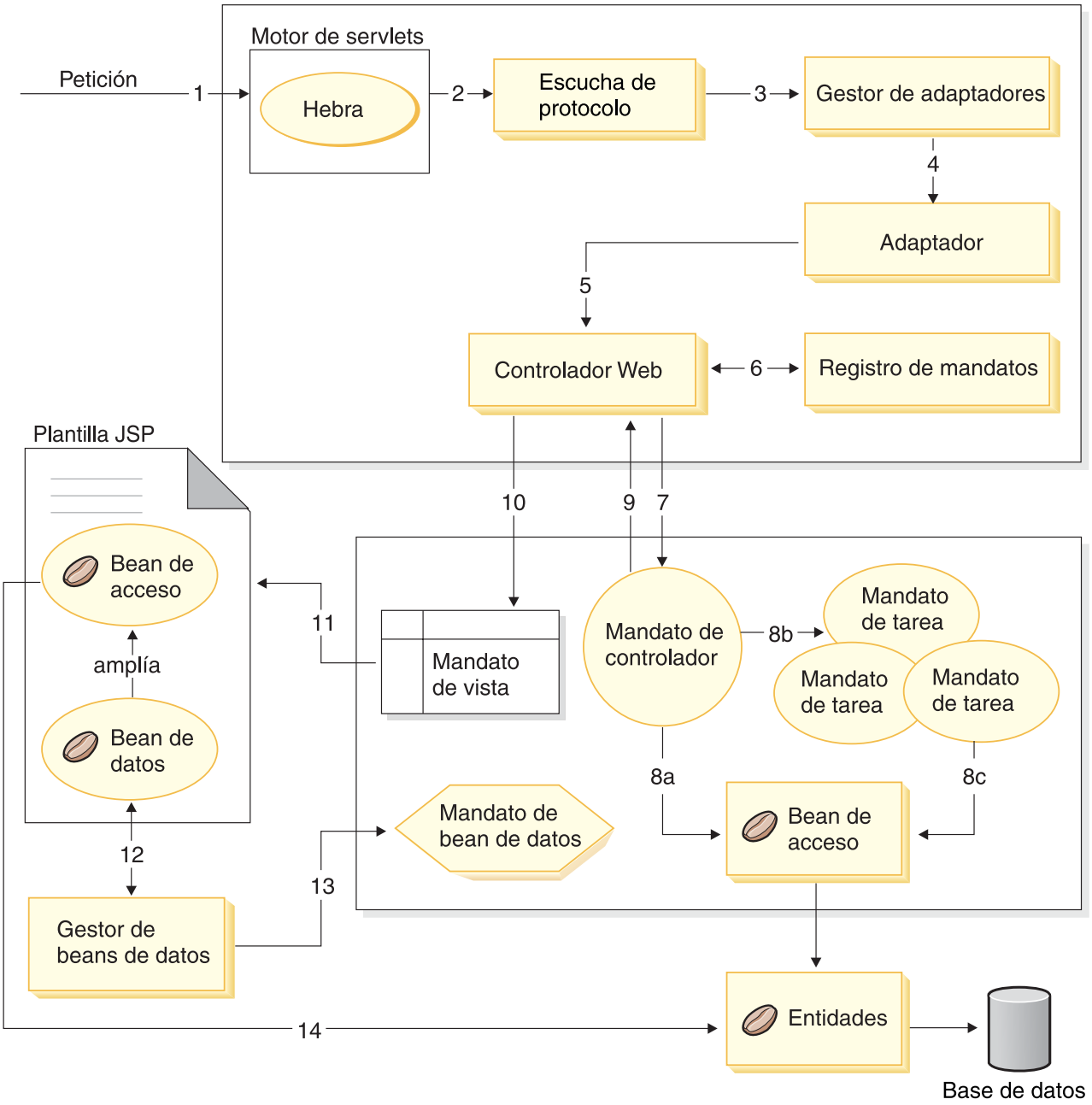


Figura 5.

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 ha empezado su ejecución, hay varias vías de acceso 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 8c en la figura 5).
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.

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 consta de un modelo de datos, de información de presentación y de 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.

El diagrama siguiente muestra cómo se aplica el patrón de diseño MVC en WebSphere Commerce.

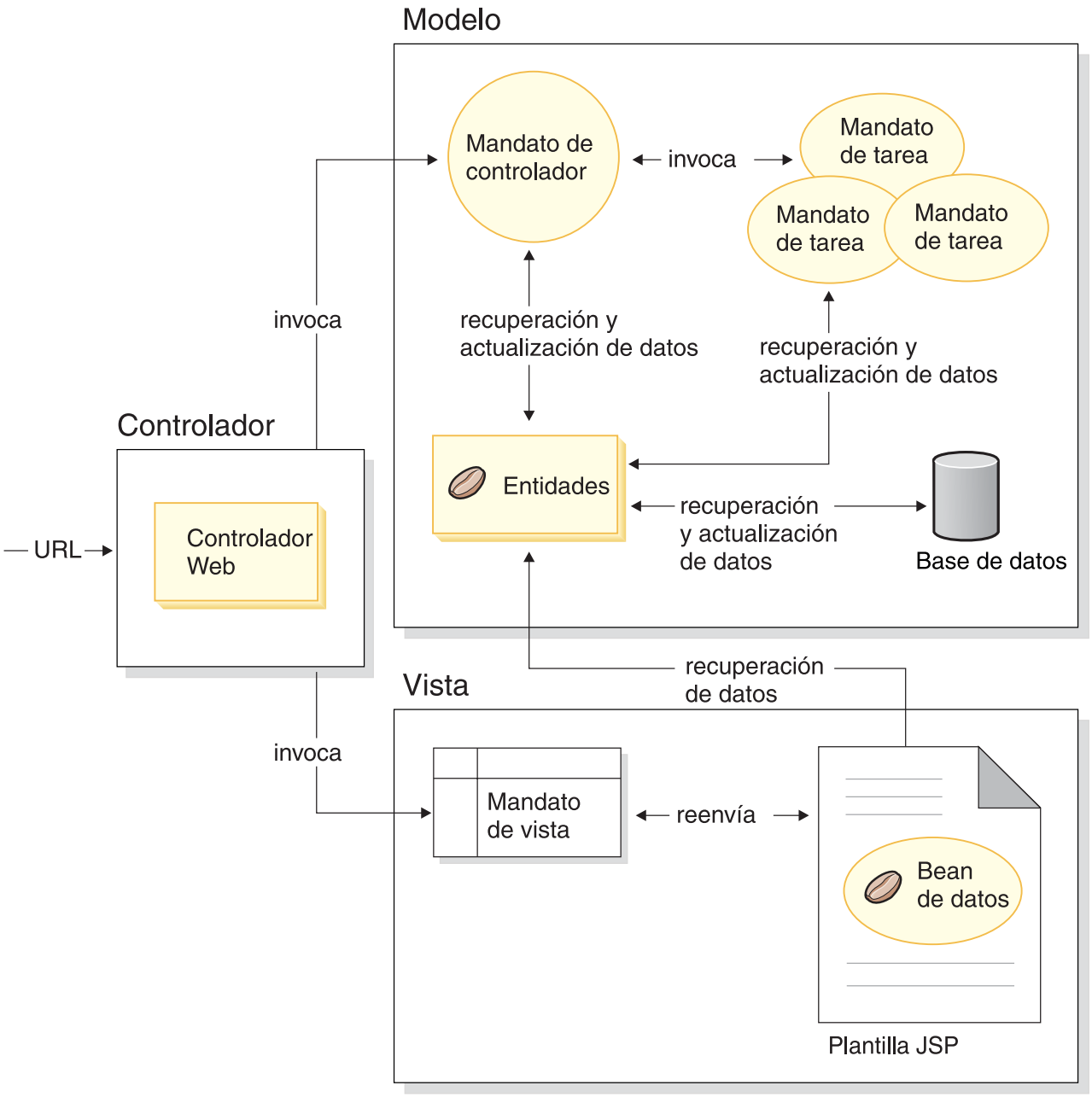

Modelo
Mandato de controlador
invoca
Mandato de tarea
Mandato de tarea
Mandato de tarea
invoca
recuperación y actualización de datos
recuperación y actualización de datos
Controlador
Entidades
URL
Controlador Web
recuperación y actualización de datos
Base de datos
recuperación de datos
Vista
invoca
Mandato de vista
reenvía
Bean de datos
Plantilla JSP

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:

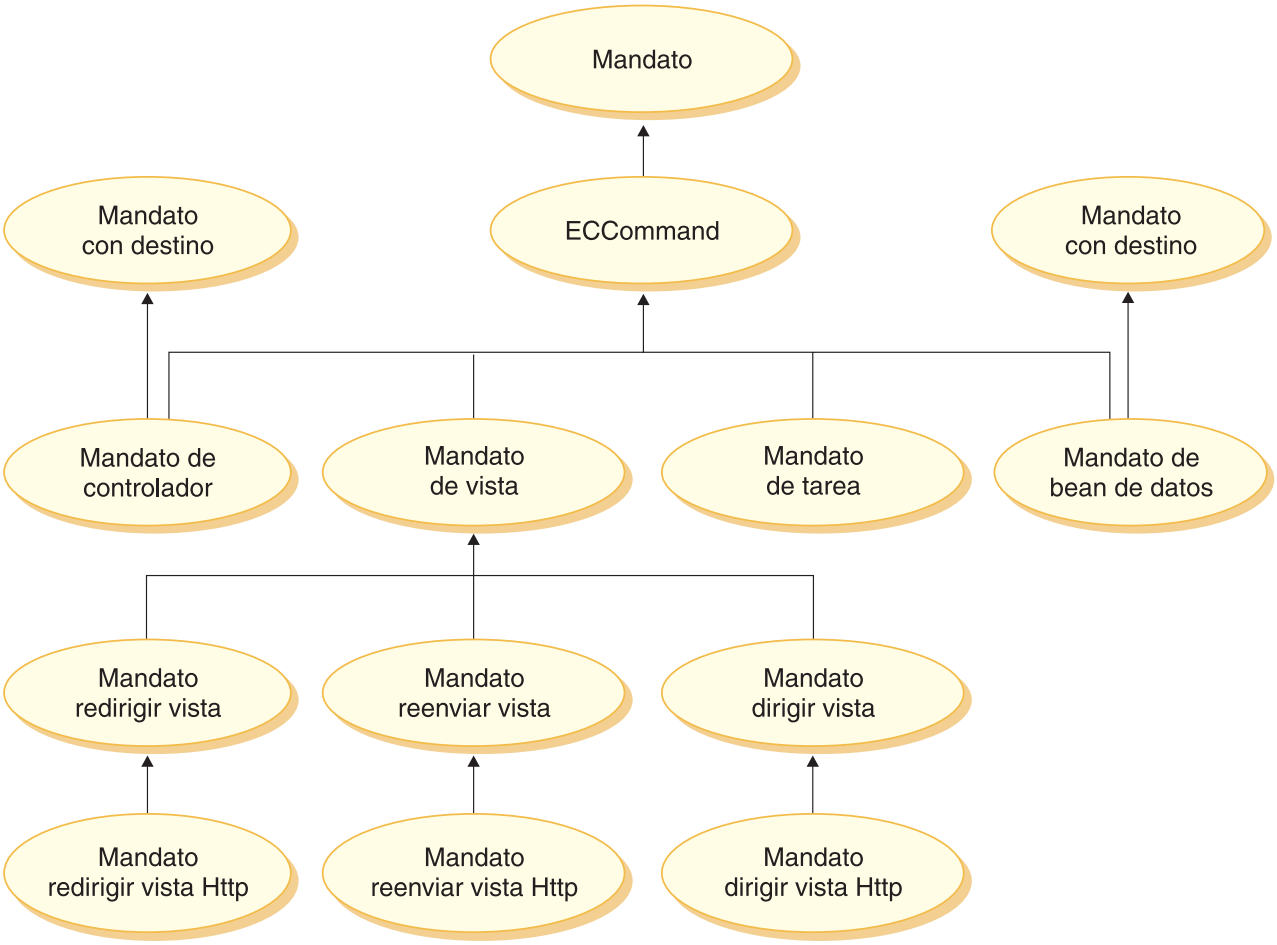


Figura 7.

Mandato de controlador

Un mandato de controlador encapsula la lógica relacionada con un proceso de negocio determinado. Los mandatos de controlador incluyen, por ejemplo, el mandato *OrderProcessCmd* para el proceso de pedidos y el mandato LogonCmd que permite a los usuarios iniciar la sesión. 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, basándose en el nombre de vista, el identificador de tienda y el tipo de dispositivo, el controlador Web determina la clase de implementación apropiada para el mandato de vista y ejecuta dicho mandato.

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 invocador 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 27 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, idTienda)
```

donde *nombreInterfaz* es el nombre de interfaz para el nuevo bean de mandato e *idTienda* es el identificador de la tienda para la que se debe implementar el mandato. Normalmente, el ID de tienda puede recuperarse utilizando el método commandContext.getStoreId().

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 164.

Mandatos de controlador anidados

Aunque la fábrica de mandatos se utiliza la mayoría de las veces para crear instancias de mandatos de tarea, también se puede utilizar en un mandato de controlador para crear una instancia de otro mandato de controlador. En otras palabras, se utiliza cuando se llama a un mandato de controlador desde dentro de otro.

La sintaxis para crear instancias de mandatos de tarea y de mandatos de controlador es la misma. Es decir, se especifica el nombre de la interfaz del mandato y el ID de tienda en ambos escenarios.

Si anida un mandato de controlador dentro de otro, tenga en cuenta los puntos siguientes:

- Una vez que haya creado la instancia del mandato anidado, llame al método setCommandContext y pásele el contexto de mandato actual. Tenga en cuenta que si está pasando un conjunto diferente de propiedades de petición al mandato anidado y estos parámetros van a afectar al contexto del mandato, deberá clonar el contexto de mandato antes de crear la instancia del mandato anidado. Esto conservará la información de contexto de mandato para el mandato externo.
- Idealmente, llame al método setRequestProperties del mandato anidado y pásele un objeto TypedProperties que contenga propiedades de entrada. De lo contrario, puede utilizar los métodos set individuales definidos en la interfaz del mandato para establecer las propiedades necesarias.
- Después de que se hayan establecido las propiedades de entrada, llame al método execute del mandato anidado.

- Dado que todos los mandatos de controlador deben devolver una vista cuando el proceso se ha completado, el mandato externo no puede hacer nada con la vista devuelta por el mandato anidado.
- El mandato anidado se ejecutará dentro del ámbito de transacción del mandato externo.

Examine el fragmento de código siguiente como ejemplo de mandato de controlador anidado. Este ejemplo muestra un método en el mandato externo y cómo éste puede utilizar la fábrica de mandatos para crear una instancia de un segundo mandato de controlador.

```
yourControllerCmd ctrlCmd = null;

public void processAndCallOtherCommand()
   throws ECException
   {
      ctrlCmd = (yourControllerCmd)CommandFactory.createCommand(
         com.yourcompany.commands.yourControllerCmd, this.getStoreId());
      ctrlCmd.setCommandContext(this.getCommandContext());
      ctrlCmd.setRequestProperties(this.getRequestProperties());
      ctrlCmd.execute();
    }
```

Examine este otro caso de ejemplo en el que se está ejecutando el mandato anidado para una tienda diferente de la primera. En este caso, el contexto de mandato del mandato externo debe conservarse para que el mandato interno no lo sobregrabe.

```
// Cree un clónico para conservar el contexto del mandato más externo
CommandContext cloneCmdCtx = (CommandContext)this.getCommandContext().clone();

//Ahora pásele un nuevo conjunto de propiedades de petición al contexto de
mandato clonado cloneCmdCtx.setRequestProperties(reqProp);

yourControllerCmd ctrlCmd = null;

public void processAndCallOtherCommandForOtherStore(int aStoreId)
   throws ECException
   {
      ctrlCmd = (yourControllerCmd)CommandFactory.createCommand(
         com.yourcompany.commands.yourControllerCmd, aStoreId;
      ctrlCmd.setCommandContext(cloneCmdCtx);
      ctrlCmd.setRequestProperties(reqProp);
      ctrlCmd.execute();
    }
```

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.

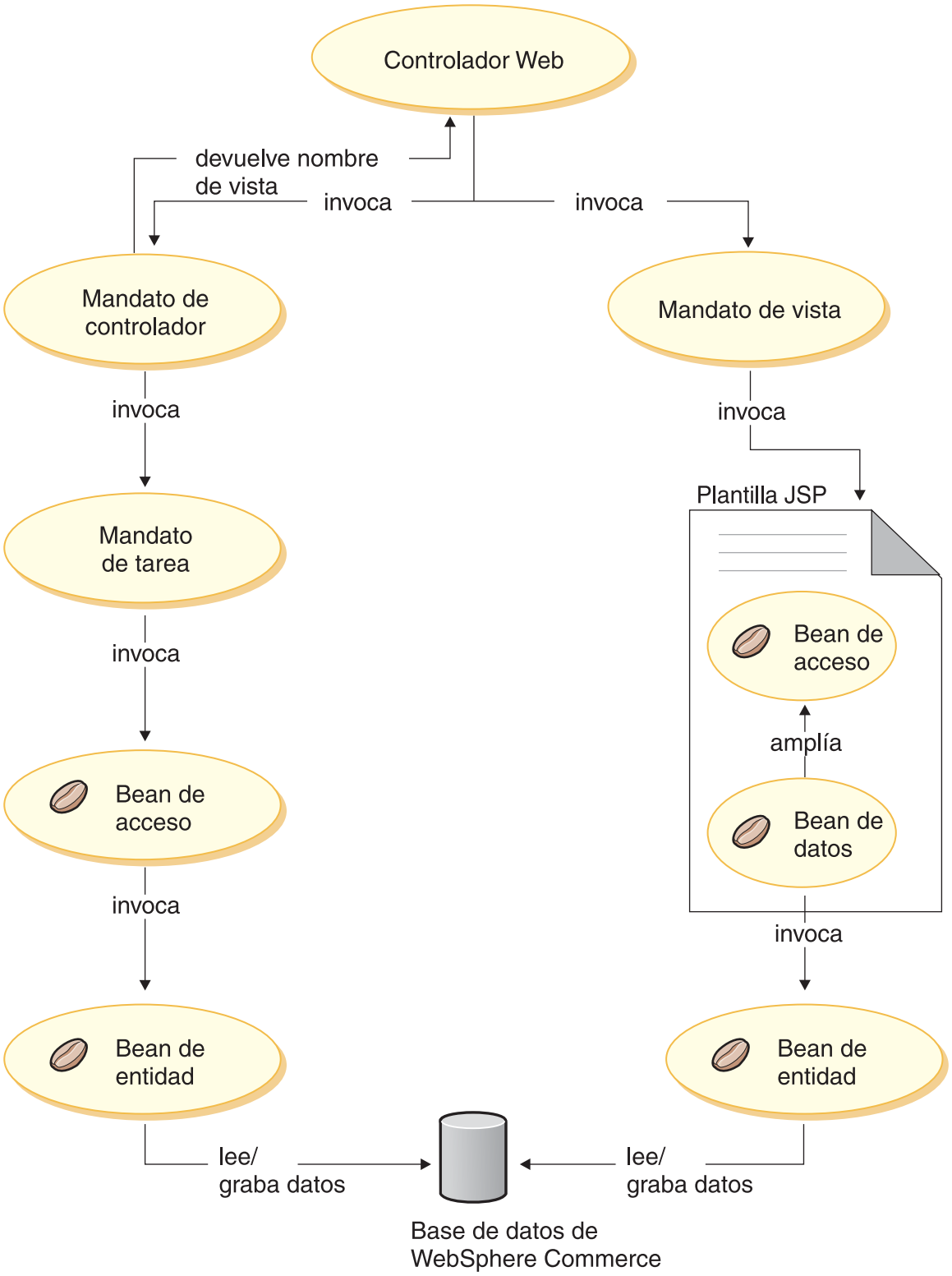


Figura 8.

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 150.

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&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 `com.ibm.commerce.catalog.commands.ProductDisplayCmd`
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.ProductDisplayCmdImpl`
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 necesaria la 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.
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 determinada (cuyo identificador de tienda es 5):

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

Los valores de serie deben escribirse entre comillas simples.

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:

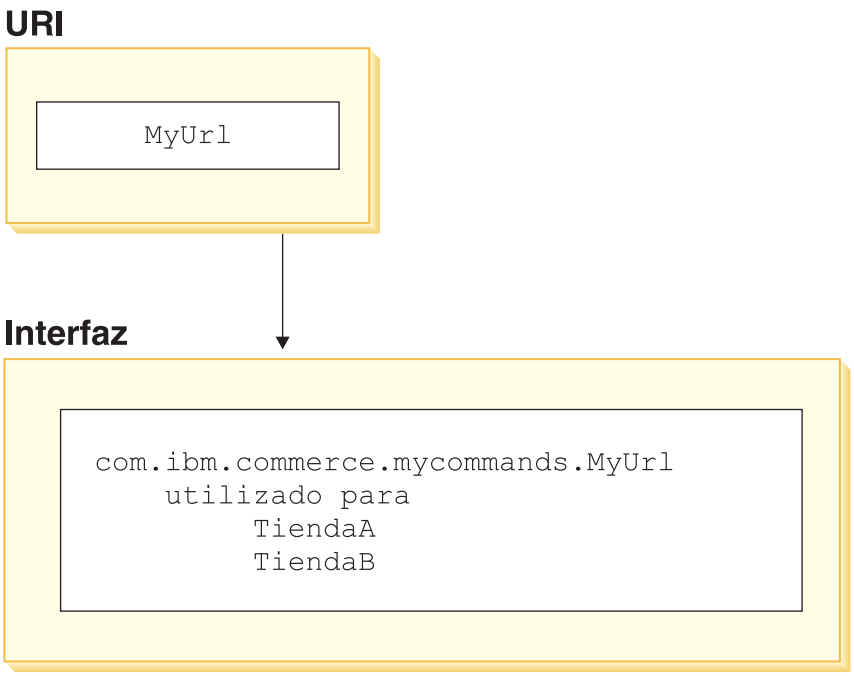


Figura 9.

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
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:

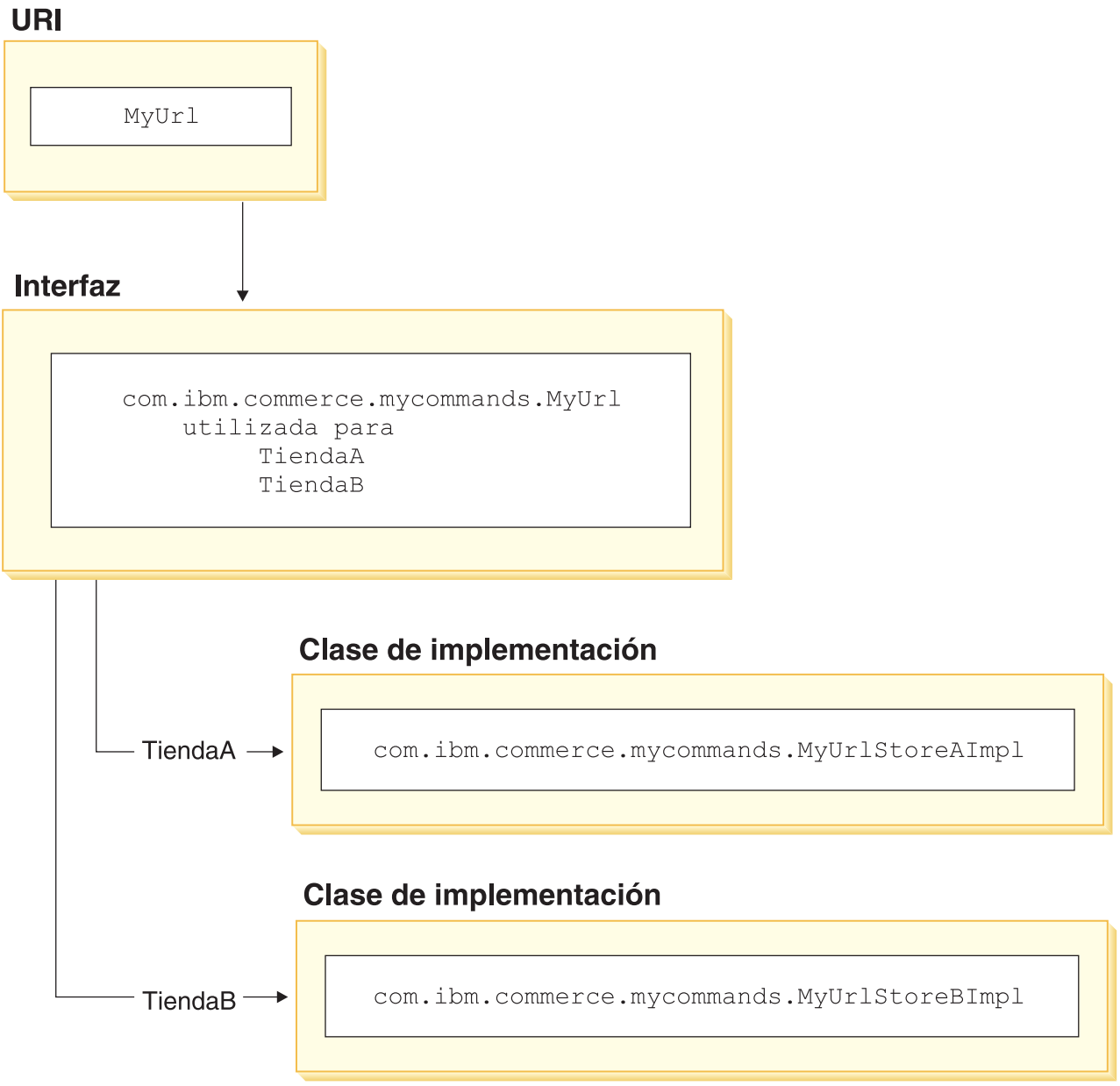


Figura 10.

Tabla VIEWREG

La tabla VIEWREG permite el registro de implementaciones de vista específicas de dispositivo y específicas de tienda. Utilizando esta tabla, pueden registrarse varias implementaciones de una vista. La infraestructura de mandatos podrá entonces devolver distintas vistas a diversos clientes.

Cuando se devuelve un nombre de vista desde un mandato de controlador o cuando se especifica un nombre de vista en una excepción, el controlador Web determina la implementación de vista a partir de la tabla VIEWREG. Se pueden correlacionar varios nombres 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 una vista nueva, es posible que necesite crear una entrada correspondiente en la tabla VIEWREG. Si se cumple una de las condiciones siguientes, se debe registrar la vista en la tabla VIEWREG:

- La 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 las vistas registradas mediante el registro de vista 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

Dado que MyView es una vista registrada, un cliente puede acceder a la vista utilizando el nombre de vista o sustituyendo el nombre de vista 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 un cliente invoque directamente una vista registrada (utilizando el nombre de archivo de visualización), deberá registrar la vista utilizando el mismo nombre para la vista que el nombre de archivo de visualización real, 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 *MyNewView* que utiliza una tienda determinada:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
CLASSNAME, PROPERTIES, DESCRIPTION,HTTPS, LASTUPDATE, INTERNAL) values
('MyNewView', -1, 5, 'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commmerce.command.HttpForwardViewCommandImpl',
'docname=MyNewView.jsp', 'Una vista de prueba.', 0, '0000-12-01', 0)
```

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

VIEW NAME	INTERFACE - NAME	CLASSNAME	PROPERTIES
Vista ProductDisplay	Forward View Command	HttpForwardView CommandImpl	docname= UserArea/ ServiceSection/ InterestItemList Subsection/ WishListDisplay.jsp
Generic Application Error	Forward View Command	HttpForwardView CommandImpl	docname =Generic Application Error.jsp&storeDir=no
GenericSystem Error	Forward View Command	HttpForwardView CommandImpl	docname = Generic System Error.jsp &storeDir=no
LogonForm	Forward View Command	HttpForwardView CommandImpl	docname = LoginForm.jsp &generic=true &storeDir=no

Nota: Los espacios que se muestran en VIEWNAME, INTERFACENAME, CLASSNAME y PROPERTIES sólo existen 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.
- 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 utiliza la tabla anterior como ejemplo, se visualizará la plantilla `GenericApplicationError.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 utiliza la tabla anterior como ejemplo, se visualizará la plantilla `GenericSystemError.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. Normalmente un bean de datos proporciona una representación simple 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 incorporación de 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(bean_datos, petición)
```

donde *bean_datos* es el bean de datos que se debe activar y *petición* es un objeto HTTPServletRequest.

Los desarrolladores de tienda deben tener en cuenta las propiedades de la tienda y los temas de globalización cuando desarrollan plantillas JSP. Para obtener más información sobre la globalización, consulte la publicación *WebSphere Commerce, Guía para el desarrollo de tiendas*.

Consideraciones sobre la seguridad de los beans de datos

Una práctica de codificación determinada para el uso de los beans de datos minimiza la posibilidad de que usuarios mal intencionados puedan acceder a la 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 una 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 WebSphere Studio Application Developer). Tan pronto como se haya vuelto a generar el bean de acceso, todos los atributos nuevos estarán automáticamente disponibles para el bean de datos inteligente.
- 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:

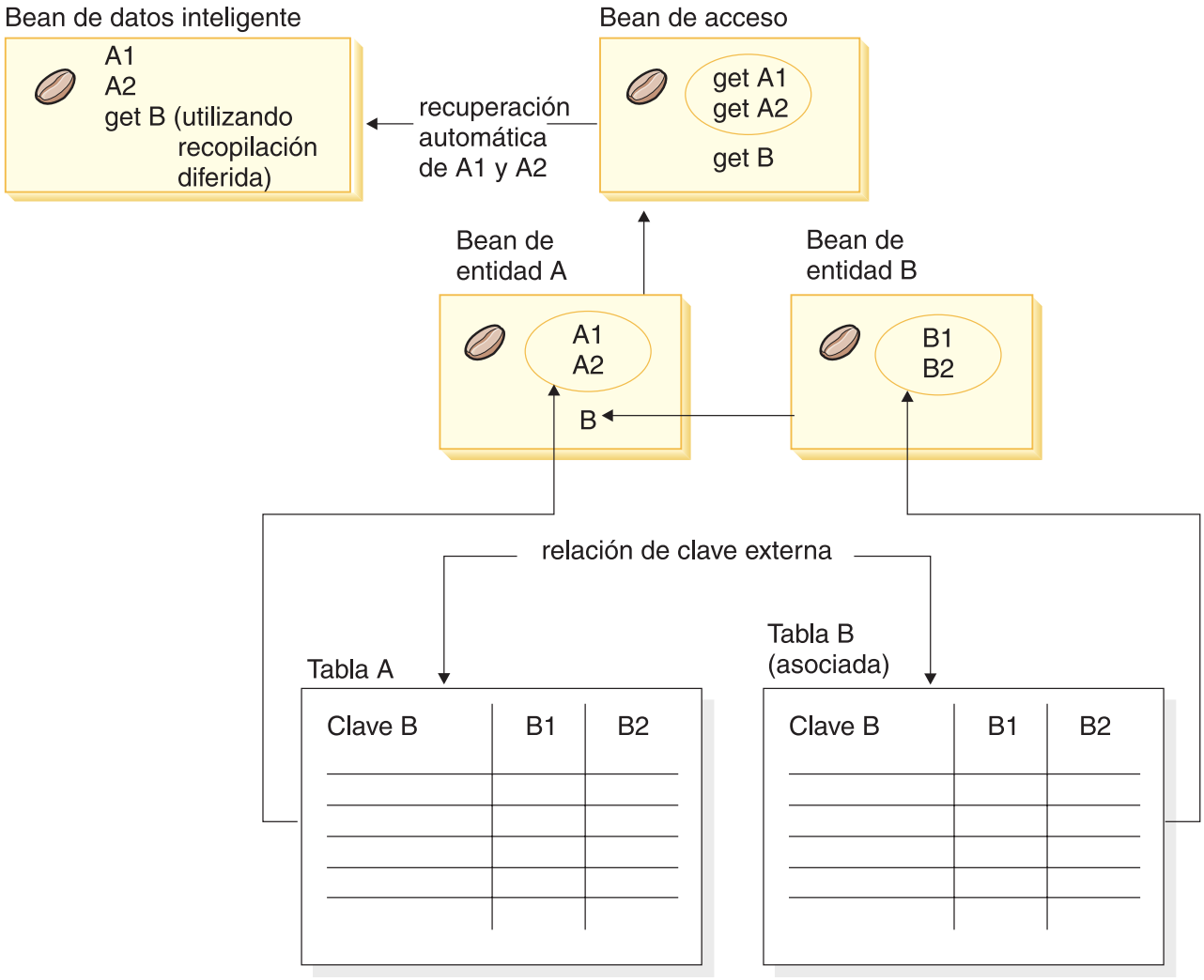


Figura 11.

Para obtener más información sobre cómo implementar una recuperación de recopilación diferida, consulte "Recuperación de datos de recopilación diferida" en la página 41.

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. Se deberá tener en cuenta que es necesario el código siguiente para que se active el bean de datos:

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

donde *bean_datos* es el bean de datos que se debe activar y *petición* es un objeto 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 la invocación de mandatos de controlador desde dentro de una plantilla JSP no es coherente con la separación de lógica de la visualización, es posible que encuentre una situación en la que sea necesario realizar dichas acciones. 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.

Si no utiliza el método setRequestProperties antes de activar este bean de datos, los parámetros del objeto de petición se pasarán al bean y, en consecuencia, también se pasarán al mandato de controlador. Sin embargo, si llama al método setRequestProperties antes de activar este bean de datos, sólo quedarán disponibles para el mandato las propiedades especificadas (las que se han pasado en el método setRequestProperties) y las propiedades por omisión especificadas en la columna PROPERTIES de la tabla CMDREG.

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

No deberá volver a utilizar la misma instancia del bean de datos para invocar otros mandatos de controlador, porque ésta contendrá datos e información de estado de su uso original.

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 programación 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.

CMREG

INTERFACENAME		PROPERTIES
com.ibm.xxx. NewCommand		parm1=1&parm2=2

CCPd: parm1=1&parm2=2

VIEWREG

INTERFACENAME		PROPERTIES
com.ibm.command.ForwardViewCommand		docname=NewView.jsp

VPd: docName=NewView.jsp

URL: http://nombre_sistpral/webapp/wcs/stores/servlet/NewCommand?storeId=1&...

CCPu: storeID=1&...

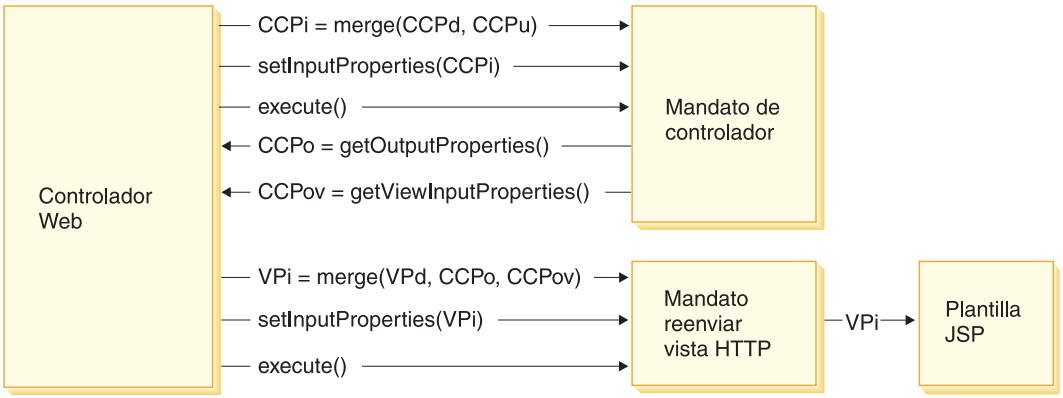


Figura 12.

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 el tema "Referencias" de la Ayuda en línea a la producción y al desarrollo 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=SearchResult.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.

  Cuando se utiliza el mandato DirectView, es opcional establecer `contentType = ttt` donde *ttt* es el tipo de contenido de 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 numerosas 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 como capa de objetos persistentes beans de entidad que se basan en la arquitectura de componentes Enterprise JavaBeans (EJB) Versión 1.1. 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.

WebSphere Studio Application Developer proporciona unas sofisticadas herramientas EJB y un entorno de prueba de unidad que soporta el desarrollo de esta infraestructura.

Las secciones siguientes están dentro del contexto de la implementación del modelo de objetos persistentes de WebSphere Commerce, que está en la especificación de EJB 1.1.

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 utilizar las herramientas EJB proporcionadas en WebSphere Studio Application Developer. 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 permanezcan en la base de datos y que llenan los campos Java con datos de la base de datos.

WebSphere Studio Application Developer proporciona dos ampliaciones en la especificación de EJB 1.1: 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 WebSphere Studio Application Developer. Al crear sus propios beans de entidad, se recomienda no utilizar la característica de asociación de WebSphere Studio Application Developer. Esta recomendación tiene por finalidad minimizar la complejidad del modelo de objeto. En vez de utilizar la característica de asociación que proporciona WebSphere Studio Application Developer, 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. Las herramientas y el entorno de ejecución de WebSphere Commerce utilizan beans enterprise privados. *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 módulos EJB:

- Catalog-ProductManagementData
- Enablement-RelationshipManagementData
- Marketing-CampaignsAndScenarioMarketingData
- Marketing-CustomerProfilingAndSegmentationData
- Member-MemberManagementData
- Merchandising-PromotionsAndDiscountsData
- Order-OrderCaptureData
- Order-OrderManagementData
- Trading-AuctionsAndRFQsData

Algunos de los módulos EJB de la lista anterior 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 bean de sesión nuevo en el módulo EJB WebSphereCommerceServerExtensionsData. Para obtener más información sobre la creación de beans de sesión nuevos, consulte “Creación de nuevos beans de sesión” en la página 70.

Descriptores de despliegue para beans enterprise de WebSphere Commerce

Un descriptor de despliegue EJB contiene valores de despliegue para los beans enterprise. WebSphere Studio Application Developer proporciona un editor de descriptor de despliegue EJB que se puede utilizar para modificar esta información de despliegue.

Al crear nuevos beans enterprise (beans de entidad o sesión), la información del descriptor de despliegue se establece dentro de la vista de Jerarquía J2EE de la perspectiva de J2EE de WebSphere Studio Application Developer. Puede ver un descriptor de despliegue EJB para los beans de WebSphere Commerce realizando lo siguiente:

1. Abra WebSphere Studio Application Developer y conmute a la perspectiva de J2EE.
2. Mediante la utilización de la Jerarquía J2EE, localice el módulo EJB para el que desea ver la información de descriptor de despliegue.

3. Pulse con el botón derecho del ratón en el módulo EJB *nombreMódulo_EJB* y seleccione **Abrir con > Editor del Descriptor de despliegue**.
   Se abrirá el Editor del Descriptor de despliegue.
4. Seleccione la pestaña Beans y observe lo siguiente:
   a. En la lista de beans, seleccione un bean. Los demás campos se llenan de información para dicho bean.
   b. El bean debe visualizarse como un bean de Entidad gestionado por contenedor 1.x.
   c. El recuadro de selección Reentrante no debe estar seleccionado.
   d. En la sección de Enlaces de WebSphere, se utiliza el valor por omisión para el nombre JNDI.
   e. En la sección de Extensiones de WebSphere, Habilitar bloqueo optimista no está habilitado para el control de simultaneidad.
   f. Observe también que puede ver los buscadores del bean en la sección de Buscadores.
5. Seleccione la pestaña Descriptor de ensamblado y observe lo siguiente:
   a. En la sección Permisos de métodos, se ha asignado WCSecurityRole a todos los métodos del bean enterprise.
   b. En la sección de Transacciones de contenedor, se ha especificado "Obligatorio" para todos los métodos del bean enterprise.
6. Seleccione la pestaña Acceso y observe lo siguiente:
   a. En la sección Intención de acceso para entidades 1.x, están definidos todos los métodos de sólo lectura. Por ejemplo, los métodos get codificados manualmente y _copyFromEJB() están asignados para ser métodos de sólo lectura. Cuando cree sus propios beans de entidad, asegúrese de marcar que los métodos apropiados sean 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.
   b. El Nivel de aislamiento se establece del modo siguiente:
      - DB2 Lectura repetible
      - Oracle Comprometido para lectura

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, encontrará que el módulo EJB Member-MemberManagementData 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:

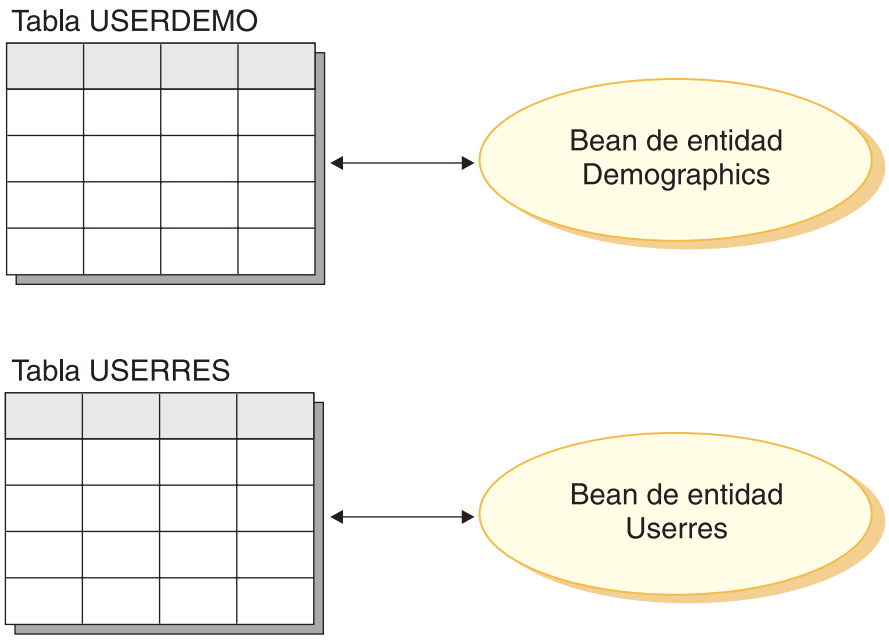


Figura 13.

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:

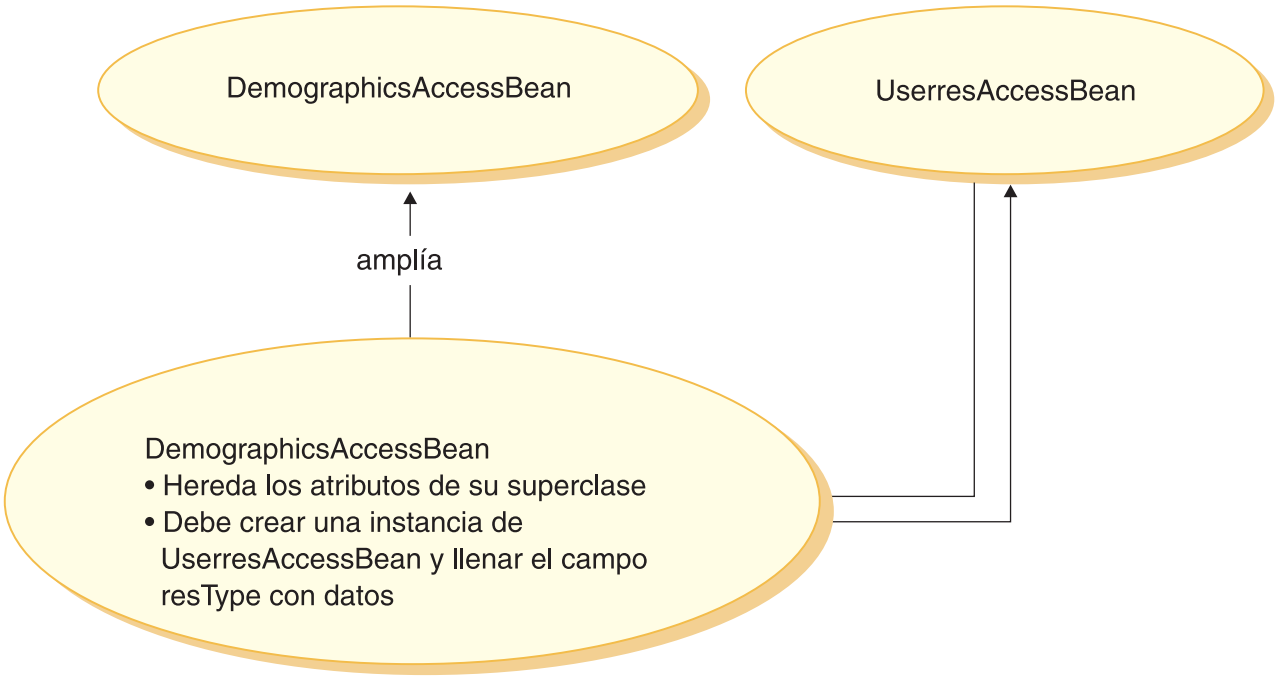


Figura 14.

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:

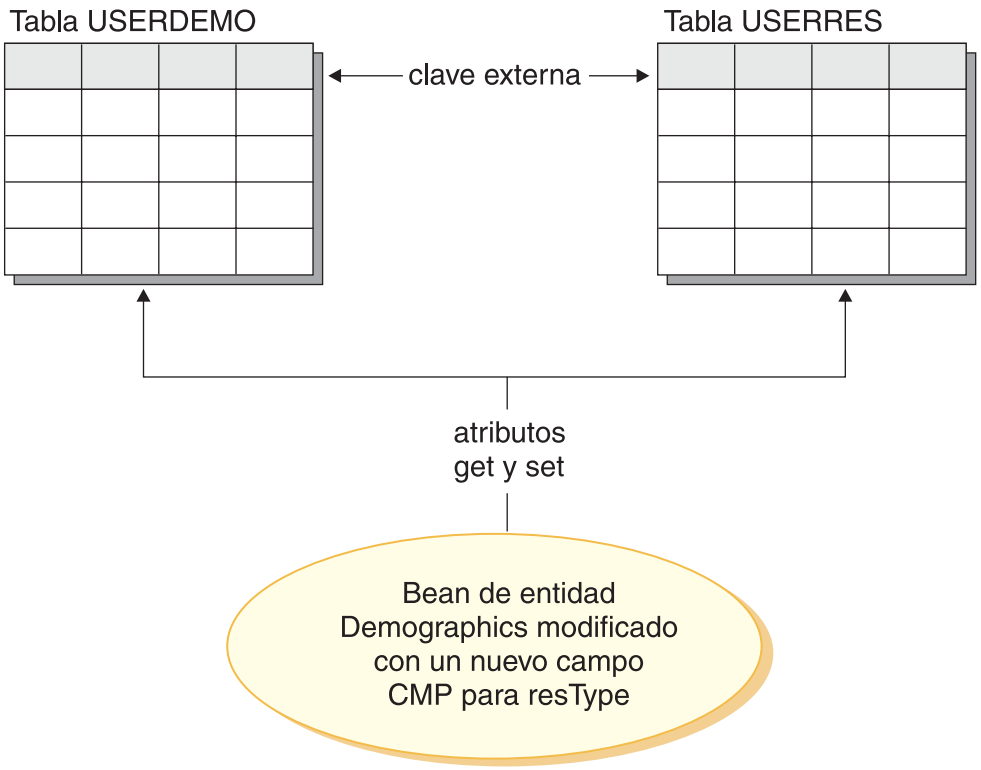


Figura 15.

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 tiene un rendimiento mejor 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 esté disponible una nueva versión de WebSphere Commerce. Deberá fusionar el código personalizado con el nuevo código, pero cuando importe el nuevo espacio de

trabajo de WebSphere Commerce, no se conservará la información de correlación entre los campos que ha añadido al bean enterprise y la nueva tabla. Por lo tanto, al migrar a un nuevo release de código de WebSphere Commerce, se deben realizar los pasos siguientes:

1. Cree una versión de su código EJB personalizado.
2. Importe la nueva versión del código de WebSphere Commerce.
3. Mediante la utilización de las herramientas de WebSphere Studio Application Developer, 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 en el 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 de la parte del servidor que se pueden generar con las herramientas proporcionadas por WebSphere Studio Application Developer. 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 *caso de entidad agregada*.
- Un caso en el que una sentencia SQL realiza 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, establecer 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 crear una sentencia SQL sencilla para recuperar toda la información de las diversas 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 "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 WebSphere Studio Application Developer para los beans enterprise en la especificación EJB 1.1 no soportan esta noción de recopilación detallada. En su lugar, se realiza una recopilación diferida que produce una selección (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.

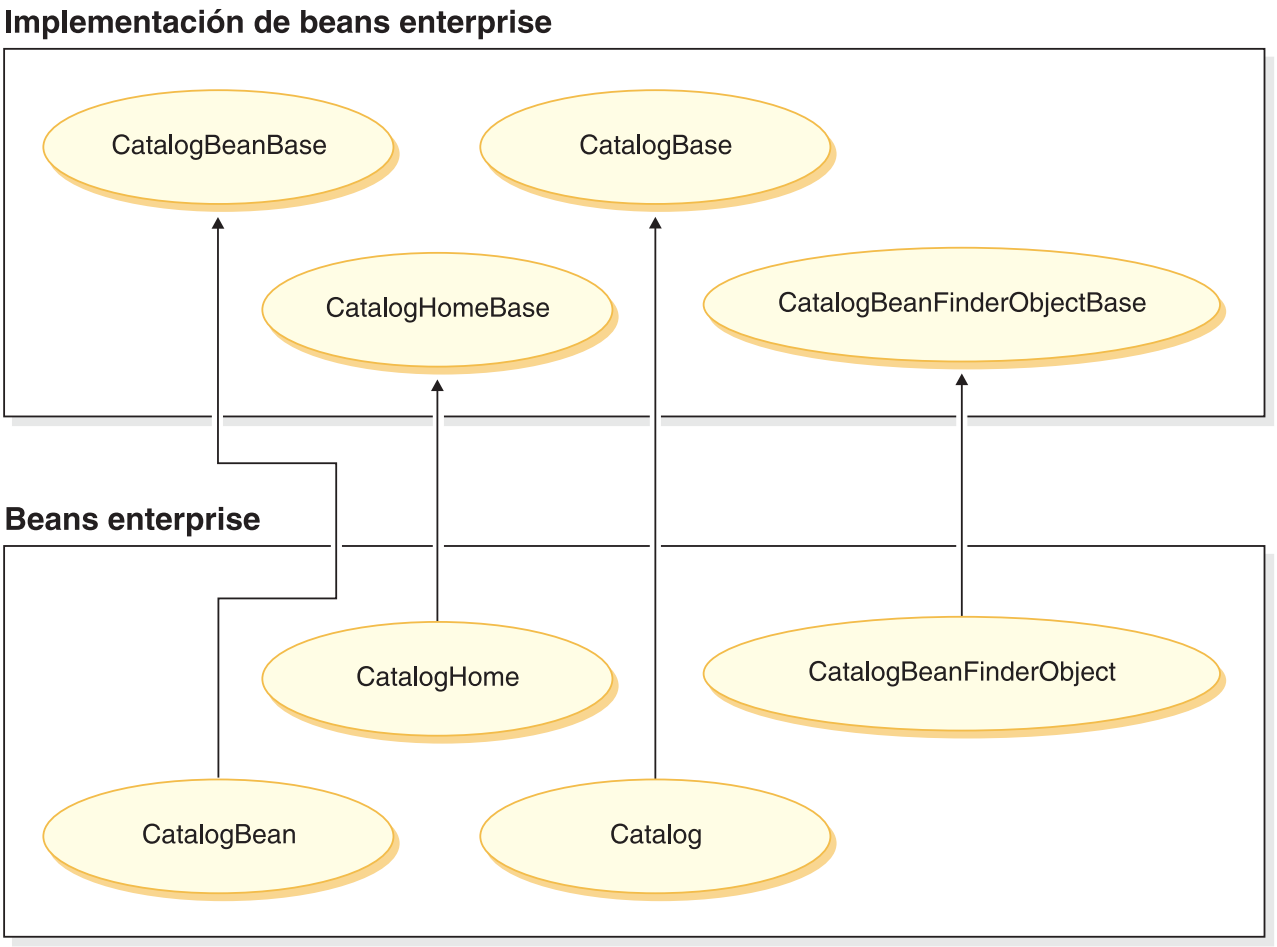


Figura 17.

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 InterestItemBeanBase 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 Java independientes de las clases y las interfaces hijo.

El espacio de trabajo de WebSphere Commerce contiene código binario para todas estas superclases e interfaces. Las modificaciones pueden realizarse en las interfaces y clases hijo. En general, se pueden realizar modificaciones en los paquetes com.ibm.commerce.*xxx*.objects y com.ibm.commerce.*xxx*.objsrc (donde *xxx* es un nombre de componente).

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. El editor de descriptor de despliegue se utiliza al añadir buscadores nuevos.

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 de búsqueda en el bean. Si la parte de la cláusula `where` de los métodos de búsqueda contienen algunos alias de base de datos (por ejemplo, T1. o T2.), deben eliminarse estos alias.

Beans de entidad públicos que contienen tipos de buscadores "findForUpdate": Si un bean de entidad público de WebSphere Commerce contiene algún tipo de buscador "findForUpdate", no podrá añadir campos nuevos al bean creando una correlación secundaria con una nueva tabla que haya creado. Esto se debe al hecho de que si crea una correlación secundaria, las sentencias SQL generadas no serán válidas y el bean ya no funcionará como se desea. Si desea ampliar la parte del modelo de objeto representado por un bean de este tipo, deberá crear un nuevo bean de entidad y utilizar en el código personalizado el bean original junto con el nuevo bean.

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, cree un campo que corresponda al atributo de la nueva tabla de base de datos.

Debido al hecho de que el espacio de trabajo de WebSphere Commerce proporciona un proyecto EJB predefinido para los nuevos beans enterprise, no necesita crear un proyecto EJB adicional. Los nuevos beans enterprise deben colocarse en el proyecto EJB WebSphereCommerceServerExtensionsData. Posteriormente, cuando despliegue los beans personalizados, cree un archivo JAR WebSphereCommerceServerExtensionsData.jar y sustituya el archivo JAR existente de la aplicación de empresa de WebSphere Commerce que se ejecuta en WebSphere Application Server. Si se utiliza este convenio de empaquetado, se simplifica mucho el despliegue.

Para crear un bean enterprise CMP nuevo, deberá realizar los siguientes pasos en WebSphere Studio Application Developer:

1. Cree el nuevo bean enterprise CMP, utilizando el asistente para la Creación de bean Enterprise. Para cada columna de la tabla de base de datos correspondiente, añada un nuevo campo CMP al bean.

2. Establezca el nivel de aislamiento de transacción para el nuevo bean.
3. Establezca la identidad de seguridad del nuevo bean.
4. Modifique los métodos de contexto de entidad.
5. Si fuera necesario, defina buscadores nuevos utilizando el editor de descriptor de despliegue EJB.
6. 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.
7. Si el bean está protegido por el sistema de control de acceso de WebSphere Commerce, implemente los métodos de control de acceso necesarios en el bean. Consulte el Capítulo 4, "Control de acceso", en la página 81 para obtener más detalles sobre cómo implementar el control de acceso en los beans enterprise. Opcionalmente, puede implementar el control de acceso después de haber creado el bean de acceso.
8. Correlacione los campos del bean enterprise con las columnas de la tabla de base de datos.
9. Genere el bean de acceso correspondiente para el bean enterprise.
10. Genere el código desplegado para el bean enterprise.
11. Pruebe el bean utilizando el Cliente de prueba universal de WebSphere Studio Application Developer.

Para obtener más detalles sobre cada uno de estos pasos, consulte los apartados siguientes. Cuando lea estas secciones, suponga que tiene una tabla nueva denominada XUSERRES que especifica información sobre el tipo de residencia 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 del bean enterprise CMP nuevo: Para crear el nuevo bean enterprise CMP, puede utilizar el asistente para la Creación de bean enterprise, como se indica a continuación:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Pulse con el botón derecho del ratón en el módulo **WebSphereCommerceServerExtensionsData** y seleccione **Nuevo > Bean enterprise**.
   Se abrirá el asistente para la Creación de bean enterprise.
3. En la lista desplegable **Proyecto EJB** seleccione **WebSphereCommerceServerExtensionsData** y pulse **Siguiente**.
4. En la ventana Crear un bean enterprise, realice lo siguiente:
   a. Seleccione **Bean de entidad con campos CMP (Persistencia gestionada por contenedor)**
   b. En el campo **Nombre de bean**, entre un nombre apropiado para el bean. Normalmente, el nombre de bean coincide con el nombre de la tabla de base de datos correspondiente. Por ejemplo, el bean se denominará XUserRes para que se corresponda con la tabla XUSERRES.
   c. En el campo **Carpeta fuente**, deje el valor por omisión que está especificado (ejbModule).
   d. En el campo **Paquete por omisión**, entre `com.mycompany.mycomponent.objects`.
   e. Pulse **Siguiente**.
5. En la ventana Detalles del bean enterprise, realice lo siguiente:

a. Pulse **Añadir** para añadir un nuevo atributo CMP para las columnas de la tabla de base de datos.
   Se abrirá la ventana Crear atributo CMP. En esta ventana, realice lo siguiente:
   1) En el campo **Nombre**, entre un nombre apropiado para el nuevo campo CMP. Tenga en cuenta que si posteriormente desea utilizar la función Emparejar por nombre cuando correlacione este campo con su columna correspondiente de la tabla de base de datos, deberá denominar el campo con un nombre exacto (no es sensible a las mayúsculas y minúsculas) al de la columna.
   2) En el campo **Tipo**, entre el tipo de datos apropiado para el campo. Tenga en cuenta que deberá utilizar las clases de reiniciador para tipos de datos primitivos (por ejemplo, utilice el tipo de datos *java.lang.Long*, no el tipo de datos *long*
   3) Si el campo es la clave primaria, marque el recuadro de selección **Campo de clave** y pulse **Aplicar**.
   4) Si el campo no es la clave primaria, seleccione **Acceso con métodos get y set**.
   5) Si el campo no es la clave primaria, borre la selección del recuadro **Promocionar métodos get y set a interfaz remota**. El recuadro de selección **Hacer que el método get sea de sólo lectura** dejará de estar disponible y, a continuación, pulse **Aplicar**.
   6) Pulse **Añadir** otra vez y repita los pasos para añadir campos nuevos para cada columna de la tabla de base de datos que necesite un campo CMP.
   7) Pulse **Cerrar** para cerrar esta ventana.

b. Elimine la marca del recuadro de selección **Utilizar el tipo de atributo de clave única para la clase de clave** y, a continuación, pulse **Siguiente**.

6. En la ventana Detalles de clase Java EJB, realice lo siguiente:
   a. Para seleccionar la superclase del bean, pulse **Examinar**.
      Se abrirá la ventana Selección de tipo.
   b. En el campo **Seleccionar una clase utilizando: (cualquiera)**, entre `ECEntityBean` y pulse **Aceptar**. Esto selecciona com.ibm.commerce.base.objects.ECEntityBean como superclase.
   c. Si el nuevo bean enterprise debe protegerse bajo la infraestructura de control de acceso de WebSphere Commerce, especifique las interfaces que la interfaz remota debe ampliar pulsando **Añadir**. Se abrirá la ventana Selección de tipo.
   d. En el campo **Seleccionar una clase utilizando: (cualquiera)**, entre `Protectable` y pulse **Aceptar**. Esto selecciona com.ibm.commerce.security.Protectable. Esta interfaz es necesaria para proteger el nuevo recurso bajo el control de acceso.
   e. Pulse **Finalizar**.

En la clase de bean, WebSphere Studio Application Developer crea el campo privado denominado `EntityContext`. WebSphere Commerce proporciona su propio campo de contexto de entidad en ECEntityBean, y el nuevo bean de entidad deberá utilizar dicho campo en lugar del campo generado. Por ello, debe eliminar el `EntityContext` generado de su nuevo bean de entidad. Esto se describe en una sección subsiguiente.

Cuando especifique `com.ibm.commerce.base.objects.ECEntityBean` como superclase, el bean heredará determinadas funciones. El siguiente ejemplo de código muestra estas funciones:

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

Establecimiento del nivel de aislamiento de transacción: Debe establecer el nivel de aislamiento de transacción para el bean en el valor correcto para el tipo de base de datos de desarrollo. Para establecer el nivel de aislamiento de transacción, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Efectúe una doble pulsación en el proyecto **WebSphereCommerceServerExtensionsData** para abrirlo con el Editor de descriptor de despliegue.
3. Pulse la pestaña **Acceso**.
4. Pulse **Añadir** junto al recuadro de texto Nivel de aislamiento.
   Se abrirá la ventana Añadir nivel de aislamiento.
5. DB2 Seleccione **Lectura repetible** y, a continuación, pulse **Siguiente**.
   Oracle Seleccione **Comprometido para lectura** y, a continuación, pulse **Siguiente**.
6. En la lista **Beans encontrados**, seleccione el bean *suNuevoBean* y, a continuación, pulse **Siguiente**.
7. En la lista **Métodos encontrados**, seleccione *suNuevoBean* para seleccionar todos los métodos y pulse **Finalizar**.
8. Guarde el trabajo (Control + S) y mantenga el editor abierto.

Establecimiento de la identidad de seguridad del bean: A continuación, establezca la identidad de seguridad del bean realizando lo siguiente:

1. En el editor de Descriptor de despliegue, asegúrese de tener seleccionada la pestaña Acceso.
2. Pulse **Añadir** junto al recuadro de texto Identidad de seguridad.
   Se abrirá la ventana Añadir identidad de seguridad.
3. Seleccione **Utilizar identidad de servidor EJB** y, a continuación, pulse **Siguiente**.
4. En la lista **Beans encontrados**, seleccione el bean *suNuevoBean* y, a continuación, pulse **Siguiente**.
5. En la lista **Métodos encontrados**, seleccione *suNuevoBean* para seleccionar todos los métodos y pulse **Finalizar**.
6. Guarde el trabajo (Control+S). Mantenga abierto el editor.

Establecimiento del rol de seguridad del bean: A continuación, establezca el rol de seguridad para los métodos del bean, realizando lo siguiente:

1. En el editor del Descriptor de despliegue, seleccione la pestaña Descriptor de ensamblado.
2. En la sección Permisos de método, pulse **Añadir**.

3. Seleccione **WCSecurityRole** como el rol de seguridad y pulse **Siguiente**.
4. En la lista de beans encontrados, seleccione *suNuevoBean* y pulse **Siguiente**.
5. En la página Elementos de método, pulse **Aplicar a todo** y, a continuación, pulse **Finalizar**.
6. Guarde el trabajo (Control + S) y, a continuación, cierre el editor de Descriptor de despliegue.

Supresión de los campos y métodos de contexto de entidad: El siguiente paso consiste en suprimir algunos de los campos y métodos relacionados con el contexto de entidad que WebSphere Studio Application Developer genera. La razón por la que es necesario suprimir estos campos es que la clase base ECEntityBean proporciona su propia implementación de estos métodos. Para suprimir los campos y métodos de contexto de entidad generados, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda el proyecto **WebSphereCommerceServerExtensionsData**.
2. Expanda **Beans de entidad**, a continuación, *suNuevoBean* y, a continuación, efectúe una doble pulsación en la clase *suNuevoBean***Bean**.
3. En la vista de Esquema, realice lo siguiente:

   Nota: Express WebSphere Studio Application Developer 5.1 le solicita si desea suprimir getEntityContext() y setEntityContext(EntityContext), con lo que no tendrá que suprimirlos manualmente, tal como se ha mencionado anteriormente. Asegúrese de seleccionar su supresión.

   a. Pulse con el botón derecho del ratón en el campo **myEntityCtx** y seleccione **Suprimir**.
   b. Pulse con el botón derecho del ratón en el método **getEntityContext()** y seleccione **Suprimir**.
   c. Pulse con el botón derecho del ratón en el método **setEntityContext(EntityContext)** y seleccione **Suprimir**.
   d. Pulse con el botón derecho del ratón en el método **unsetEntityContext()** y seleccione **Suprimir**.
4. Guarde el trabajo (Control+S).

Adición de buscadores nuevos:

Introducción para trabajar con buscadores: Se desaprueba la utilización de interfaces de ayuda de buscadores. Para cualquier trabajo de desarrollo nuevo, es necesario utilizar el editor de descriptor de despliegue EJB en lugar de la interfaz de ayuda de buscadores a fin de definir las consultas y las declaraciones de método.

Si desea información detallada sobre la utilización del lenguaje de consulta EJB para crear buscadores, las ventajas de este planteamiento respecto a otros planteamientos para los buscadores y más detalles sobre las herramientas relacionadas, consulte la ayuda en línea de WebSphere Studio Application Developer.

Adición de un buscador nuevo en el nuevo bean: Si necesita añadir un nuevo buscador en el bean enterprise, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Efectúe una doble pulsación en el proyecto **WebSphereCommerceServerExtensionsData** para abrir el Editor del descriptor de despliegue de EJB.

3. Pulse la pestaña **Beans**.
4. En el panel Beans, seleccione el bean *suNuevoBean* y, a continuación, en el panel de la derecha, desplácese hacia abajo y expanda **Extensiones de WebSphere**.
5. Pulse **Añadir** junto al recuadro de texto **Buscadores**.
   Se abrirá la ventana Añadir descriptor de buscador.
6. Seleccione **Nuevo** y, a continuación, en el campo **Nombre**, entre `findByX`*`suArg`* (donde *suArg* es el nombre del argumento por el que desea realizar la búsqueda). 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.
7. Pulse **Añadir** junto al recuadro de texto **Parámetros** y, a continuación, realice lo siguiente:
   a. En el campo **Nombre**, entre *suArg*.
   b. En el campo **Tipo**, entre el tipo de datos apropiado.
   c. Pulse **Aceptar**.
8. En el campo **Tipo de retorno**, entre uno de los siguientes y pulse **Siguiente**:
   - 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`.
9. En la lista desplegable **Tipo de buscador**, seleccione **WhereClauseFinderDescriptor**.
10. En el campo **Sentencia de buscador**, entre un buscador apropiado. Por ejemplo, entre `T1.MEMBERID = ?` y, a continuación, pulse **Finalizar**.
11. Guarde el trabajo y, a continuación, cierre el editor del Descriptor de despliegue de EJB.

Por razones de seguridad, cuando cree métodos FinderHelper para un nuevo bean de entidad, deberá utilizar inserciones de parámetros como se muestra en los pasos anteriores. 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:

```
T1.MEMBERID = "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 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 la vista de Jerarquía J2EE, efectúe una doble pulsación en la clase ***suNuevoBean*Bean** para abrirla y ver el código fuente.
2. Debe modificar el código fuente para que cada campo CMP no anulable se incluya como parámetro de entrada en el método y, de este modo, se cree una instancia de cada campo CMP con el valor apropiado. 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 {
      _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 {
      _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 73 para obtener detalles.
3. Debe añadir el nuevo método ejbCreate e la interfaz inicial. Esto hará que el método esté disponible en el bean de acceso generado. Para añadir el método a la interfaz inicial, realice lo siguiente:
   a. Pulse con el botón derecho del ratón en el método **ejbCreate(***susParámetros***)** de la vista de Esquema y seleccione **Bean enterprise > Promocionar a interfaz inicial**.

Creación de un método ejbPostCreate nuevo: A continuación, deberá crear un nuevo método ejbPostCreate que tenga los mismos parámetros de entrada que el nuevo método ejbCreate. Para crear este método nuevo, realice lo siguiente:

1. Efectúe una doble pulsación en la clase ***suNuevoBean*Bean** para abrirla y ver el código fuente.
2. Cree un nuevo método ejbPostCreate con los mismos parámetros de entrada que se han utilizado en el método ejbCreate nuevo. Para continuar con el ejemplo de residencia de usuario, añadiremos el código siguiente en la clase:

   ```
   public void ejbPostCreate(int argUserId,
      String argHome, byte Rooms)
      {
      }
   ```

Guarde los cambios de código.

Adición de métodos de control de acceso en el bean: Si el nuevo bean tiene que estar protegido bajo el control de acceso, deberá añadir el método getOwner. Otro método que es opcional para el control de acceso es el método fulfills. Para obtener detalles sobre los métodos necesarios y opcionales, consulte el apartado "Implementación del control de acceso en los beans enterprise" en la página 96. Para añadir métodos de control de acceso al nuevo bean, realice lo siguiente:

1. En la vista de Jerarquía J2EE, efectúe una doble pulsación en la clase ***suNuevoBean*Bean** para abrirla y ver el código fuente.
2. En el código fuente, añada un método getOwner que incluya lógica para devolver el propietario de este recurso. Por ejemplo, para devolver el propietario del bean UserRes, deberá devolver el ID de miembro del usuario:

```
public java.lang.Long getOwner()
   throws java.lang.Exception {
      return getMemberId();
   }
```

3. Para este ejemplo, deberá añadir un método fulfills que especifique qué relación debe satisfacer el usuario antes de que se le permita actuar en este recurso. En este caso, deberá especificar que sólo el creador de este objeto UserRes está autorizado a actuar. En otras palabras, cada usuario sólo está autorizado a actuar en su propio objeto UserRes. En el siguiente fragmento de código, se muestra este requisito de relación:

```
public boolean fulfills(Long member, String relationship)
   throws java.lang.Exception {
if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      return false;
   }
```

4. Guarde el trabajo.

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. Cuando existen el bean enterprise y la tabla de base de datos correspondiente, se utiliza un tipo de correlación de Encuentro a medio camino. WebSphere Studio Application Developer proporciona herramientas para simplificar esta tarea.

Para crear la correlación, realice lo siguiente:

1. En la vista de Jerarquía J2EE, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Correlación entre EJB y RDB**.
   Se abrirá la ventana Correlación entre EJB y RDB.
2. Seleccione **Encuentro a medio camino** y pulse **Siguiente**.
3. En la ventana Conexión de base de datos, realice lo siguiente:
   a. En el campo **Nombre de conexión**, entre `WebSphereCommerceServerExtensionsData`
   b. En el campo **Base de datos**, entre *`BDdesarrollo`*
   c. En el campo **ID de usuario**, entre *`usuariobd`*
   d. En el campo **Contraseña**, entre *`contraseñabd`*
   e. En la lista desplegable de base de datos, seleccione el tipo de proveedor de base de datos para la base de datos de desarrollo.

- DB2 DB2 Universal Database 8.1
- Oracle Oracle 9i

f. Oracle En el campo **Sistema principal**, entre el nombre de sistema principal totalmente calificado del servidor de bases de datos. Por ejemplo, entre `dbserver.yourcompany.com`

g. Oracle En el campo Ubicación de clase, entre la ubicación del archivo classes12.zip. Por ejemplo, entre `D:\oracle\ora92\jdbc\lib\classes12.zip`

h. Pulse **Siguiente**. Una vez que se haya establecido la conexión, se visualizará la lista de tablas de la base de datos. Posteriormente también puede ver el Documento de conexión examinando la vista de Servidores de bases de datos en la perspectiva de datos.

4. Seleccione la tabla ***suNuevaTabla*** y pulse **Siguiente**.
5. Seleccione **Emparejar por nombre y tipo** y, a continuación, pulse **Finalizar**. Ahora se abrirá el Editor de correlaciones.
6. Oracle Si alguna columna tiene un tipo de datos de "NUMBER", deberá modificar el tipo de datos. Pulse con el botón derecho del ratón en la tabla ***suNuevaTabla*** y seleccione **Abrir editor de tablas**. En el editor de tablas, realice lo siguiente:
   a. Seleccione la pestaña **Columna**.
   b. Seleccione la columna cuyo tipo de datos necesite cambiarse y cambie el tipo de columna NUMBER por un tipo más específico. Por ejemplo, cámbielo por INTEGER.
   c. Guarde los cambios.
7. En el panel Beans enterprise, expanda el bean ***suNuevoBean***. En el panel Tablas, expanda la tabla ***suNuevaTabla***.
8. Correlacione los campos del bean *suNuevoBean* con las columnas de la tabla *suNuevaTabla*, realizando lo siguiente:
   a. Pulse con el botón derecho del ratón en el bean ***suNuevoBean*** y seleccione **Emparejar por nombre**.
9. Guarde los cambios realizados en el archivo Map.mapxmi y, a continuación, cierre el archivo.
10. Oracle Deberá editar la definición de tabla utilizando un editor de texto, como se indica a continuación:
    a. Abra el archivo *suNuevoBean*.xmi con un editor de texto.
    b. Sustituya todas las apariciones de `SQLNumeric6` por `SQLNumeric3`.
    c. Guarde los cambios y, a continuación, cierre el archivo.

Modificación del nombre de esquema: El siguiente paso consiste en modificar el nombre de esquema para que el bean se pueda portar a otras bases de datos. El valor especial para permitir que un bean se pueda portar de este modo es NULLID. Para modificar el nombre de esquema, realice lo siguiente:

1. En la Perspectiva de J2EE, conmute a la vista de Jerarquía J2EE.
2. Expanda **Bases de datos** y, a continuación, expanda **WebSphereCommerceServerExtensionsData**.
3. Pulse con el botón derecho del ratón en el nodo de esquema (por ejemplo db2user) y seleccione **Redenominar**.
4. Establezca el valor en `NULLID`.

Creación del bean de acceso: 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. Las herramientas de WebSphere Studio Application Developer 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, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** y, a continuación, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Nuevo > Bean de acceso**.
   Se abrirá la ventana Añadir un bean de acceso.
2. Seleccione **Ayudante de copia** y pulse **Siguiente**.
3. Seleccione el bean ***suNuevoBean*** y pulse **Siguiente**.
4. En la lista desplegable Método constructor, seleccione **findByPrimaryKey(***nombreSuPaquete***.** *suNuevoBean***Key)** como método constructor.
5. Seleccione todos los atributos en la sección Ayudantes de atributo.
6. Pulse **Finalizar**.
7. Guarde el trabajo.

Generación de código desplegado: El programa de utilidad de generación de código analiza los beans para asegurar que se cumplan las especificaciones de EJB de Sun Microsystems y comprueba que se sigan las normas específicas del 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 inicial y remota, así como las clases de buscador y persistencia JDBC para los beans CMP. También genera las clases Java ORB, de ″apéndices″ y de relación necesarias para el acceso RMI a través de IIOP, así como ″apéndices″ para las interfaces inicial y remota.

Para generar el código desplegado, efectúe lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** y, a continuación, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Código de despliegue y RMIC**.
   Se abrirá la ventana Código de despliegue y RMIC.
2. Seleccione el bean ***suNuevoBean*** y pulse **Finalizar**.

Puede ver el código que se acaba de generar yendo a la vista del Business Professional Navegador J2EE Express Navegador de proyectos. Encontrará lo siguiente:

Tabla 1.

Tipo de código	Nombre de clase
Código generado por implementación de contenedor	EJSCMP*suNuevoBean*HomeBean.java
	EJSRemoteCMP*suNuevoBean*.java
	EJSRemoteCMP*suNuevoBean*Home.java
	EJSFinder*suNuevoBean*Bean.java
Código de acceso JDBC	EJSJDBCPersisterCMP*suNuevoBean*Bean.java

Tabla 1. (continuación)

Tipo de código	Nombre de clase
Código de apéndices y relación RMI	_EJSRemoteCMP*suNuevoBean*_Tie.java
	*_suNuevoBean*_Stub.java
	_EJSRemoteCMP*suNuevoBean*Home_Tie.java
	*_suNuevoBean*Home_Stub.java

Utilización del cliente de prueba para probar el bean enterprise: WebSphere Studio Application Developer 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. Conmute a la perspectiva de Servidores.
2. Efectúe una doble pulsación en el servidor **WebSphereCommerceServer** y pulse la pestaña **Configuración**.
3. Seleccione **Habilitar el cliente de prueba universal**. Guarde los cambios.
4. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar**.
5. Pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Ejecutar en servidor**.
   Se abrirá el navegador Web con el cliente de prueba universal. El lugar inicial para probar el bean enterprise es el Explorador JNDI donde podrá encontrar los nombres de los beans que se ejecutan en el contenedor EJB de este servidor.
6. Pulse con el botón derecho del ratón en **Explorador JNDI**
7. A continuación, deberá navegar hasta la interfaz *suNuevoBean*Home, expandiendo la jerarquía del modo siguiente: **célula > nodos > localhost > servidores > server1 > ejb >** *suEstructuraPaquete*.
8. Pulse la interfaz ***suNuevoBean*Home**. En el panel Referencias, seleccione **Referencias EJB > *suNuevoBean* > *suNuevoBean*Home**. Pulse el método de creación.
9. En los campos del panel derecho que corresponden a los parámetros de entrada necesarios para el método de creación, entre los valores apropiados.
10. Pulse **Invocar** y se mostrará el resultado en el panel inferior.
11. Pulse **Trabajar con objeto** para añadir la interfaz remota al panel Referencias y verá los valores que ha entrado bajo Referencias de objeto. Se ha creado un registro nuevo en la nueva tabla.
12. Pruebe otros métodos, según sea apropiado.
13. Cierre el cliente de prueba y detenga el servidor.

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

- No utilice el tipo de datos BLOB ni CLOB.
- El código de beans enterprise no debe hacer referencia a ningún elemento externo a los módulos de beans enterprise. Por ejemplo, no debe hacer referencia a mandatos ni beans de datos
- Las secciones anteriores describen cómo incluir el control de acceso en un nuevo bean cuando se crea inicialmente el bean. También se puede añadir después de haber creado el bean añadiendo la interfaz

com.ibm.commerce.security.Protectable. Si es necesario, añada también la interfaz com.ibm.commerce.security.Groupable en la interfaz remota del bean enterprise. Asimismo, se deben implementar en el bean determinados métodos. Después de añadir estas interfaces y de añadir los métodos necesarios, vuelva a generar el código desplegado del bean y el bean de acceso. Para obtener más información, consulte "Implementación del control de acceso en los beans enterprise" en la página 96.

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 WebSphere Studio Application Developer.

Los beans de datos nuevos se almacenan en el proyecto WebSphereCommerceServerExtensionsLogic.

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

1. Cree un paquete para almacenar el código de 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.

Casa paso se describe más detalladamente en las secciones subsiguientes.

Creación del paquete para el código de bean de datos: Al crear un paquete, se crea un lugar en el que se puede almacenar el código de bean de datos.

Para crear un nuevo paquete, haga lo siguiente:

1. Abra WebSphere Studio Application Developer y conmute a la perspectiva Java.
2. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Nuevo > Paquete**. Se abrirá el asistente para Paquete Java nuevo.
3. El valor para **Carpeta fuente** es previamente lleno con `WebSphereCommerceServerExtensionsLogic/src`. Conserve este valor.
4. En el campo **Nombre**, entre un nombre apropiado para el nuevo paquete. Por ejemplo, entre `com.mycompany.mydatabeans`.
5. Pulse **Finalizar**.

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 el botón derecho del ratón en el paquete en el que almacenará el bean de datos y seleccione **Nuevo > Clase**.
   Se abrirá el asistente para Clase Java nueva.
2. Los campos de nombre de proyecto y paquete ya están rellenados.

3. En el campo **Nombre**, entre un nombre para el nuevo bean de datos. Por ejemplo, para crear un bean de datos que amplíe UserResAccessBean, entre `UserResDataBean`.
4. En la lista **Modificadores**, seleccione **public**.
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. Para especificar las interfaces que el bean de datos debe implementar, 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 **Aceptar**.
7. Pulse **Finalizar**.

Adición de campos obligatorios al bean de datos: Esta sección describe cómo modificar los campos necesarios en el bean de datos nuevo. Los dos campos necesarios son para los siguientes tipos de información:

- contexto de mandatos
- propiedades de petición

Para modificar el campo iCommandContext, realice lo siguiente:

1. Efectúe una doble pulsación en el bean de datos nuevo (por ejemplo UserResDataBean) para ver el código fuente.
2. Localice el método getCommandContext. Aparece de este modo:

   ```
   public CommandContext getCommandContext() {
   return null;
   ```

   Añada lo siguiente en el código fuente:

   ```
   private CommandContext iCommandContext = null;
    public com.ibm.commerce.command.CommandContext getCommandContext()
   {
    return iCommandContext;
   }
   ```

3. Modifique el método *set*CommandContext. Inicialmente aparece de este modo:

   ```
   public void setCommandContext(CommandContext arg0) {
    }
   ```

   Modifique el código para que aparezca del modo siguiente:

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

4. Modifique el método *get*CommandContext. Inicialmente aparece de este modo

   ```
   public CommandContext getCommandContext() {
      return null;
    }
   ```

   Modifique el código para que aparezca del modo siguiente:

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

5. Guarde el trabajo.

Para modificar el campo iRequestProperties, realice lo siguiente:

1. Efectúe una doble pulsación en el bean de datos nuevo (por ejemplo UserResDataBean) para ver el código fuente.
2. Localice el método *get*RequestProperties. Inicialmente aparece de este modo:

   ```
   public TypedProperty getRequestProperties() {
      return null;
    }
   ```

   Modifique el código fuente para que quede así:

   ```
   private com.ibm.commerce.datatype.TypedProperty
      requestProperties;

   public TypedProperty getRequestProperties() {
    return requestProperties;
   }
   ```

3. Modifique el método *set*RequestProperties. Inicialmente aparece de este modo:

   ```
   public void setRequestProperties(TypedProperty arg0) throws Exception {
    }
   ```

   Modifique el código fuente para que quede así:

   ```
   public void setRequestProperties(com.ibm.commerce.datatype.TypedProperty
      aParam)
      throws Exception
   {
   // copiar TypedProperteis de entrada en local

     requestProperties = aParam;
   }
   ```

4. Modifique el método getRequestProperties. Inicialmente aparece de este modo:

   ```
   public TypedProperty getRequestProperties() {
      return null;
    }
   ```

   Modifique el código fuente para que quede así:

   ```
   public TypedProperty getRequestProperties() {
    return requestProperties;
   }
   ```

5. Guarde el trabajo.

Cómo llenar de datos la clave primaria del bean de acceso correspondiente: Tenga en cuenta que es posible que desee modificar el código fuente para llenar de datos 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. En el ejemplo siguiente, observe que 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)
           {
           }
      }
```

Tenga en cuenta que el procedimiento recomendado para establecer la clave primaria es el método indirecto.

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. En la vista de Esquema, seleccione el método **`populate()`** para ver el código fuente.
   Inicialmente aparece de este modo:
   ```
   public void populate () throws Exception {}
   ```
3. Modifique el código de forma que el método quede así:
   ```
         try {
      super.refreshCopyHelper();
        } catch (javax.ejb.FinderException e) {
         throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        "UserResDataBean", "populate");
   ```

Guarde el trabajo (Control+S).

Tenga en cuenta que si el nuevo bean de datos está ampliando un bean de acceso que requiere parámetros de entrada adicionales cuando se crea una instancia del mismo, también deberá establecer esos valores en el método ″populate″ del bean de datos.

Creación de nuevos beans de sesión

Cuando cree beans de sesión nuevos, créelos en el proyecto WebSphereCommerceServerExtensionsData.

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 de origen de datos utilizado por WebSphere Commerce Server, para que el bean de sesión participe 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,
      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 al origen 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 Versión 1.1 especifica tres opciones alternativas en el momento del compromiso respecto al estado de la instancia. Se describen como opciones A, B y C en el documento de especificaciones. Para obtener detalles completos sobre estas opciones, consulte el documento de especificación de la versión 1.1 de Enterprise JavaBean 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 “find or update”, se aplica un *bloqueo de grabación* (también conocido como bloque 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, REGISTRATIONUPDATE de la tabla USERS no se habrá actualizado 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 está utilizando WebSphere Application Server para proteger los beans enterprise, deberá asignar el rol `WCSecurityRole` a los métodos de cualquier bean enterprise nuevo. Para realizar esta tarea se utiliza Application Assembly Tool de WebSphere Application Server. Realice este paso cuando esté desplegando los nuevos beans enterprise. Adicionalmente, si modifica beans de entidad de WebSphere Commerce existentes, deberá asignar el rol `WCSecurityRole` a cada método de cada uno de los beans de entidad del proyecto EJB modificado.

Para obtener una descripción del proceso de despliegue del código personalizado, consulte el Capítulo 9, “Información detallada sobre el despliegue”, en la página 179.

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 de 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 {
      //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 la clave primaria está definida como un tipo de datos largo (long) (BIGINT para DB2 o NUMBER(38, 0) 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. WebSphere Studio Application Developer utiliza el convenio de denominación `setInitKey_xxx` (donde *xxx* es el nombre de campo de clave primaria) para dar nombre a los métodos set para 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 iniciales 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 WebSphere Studio Application Developer 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 inicial 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.

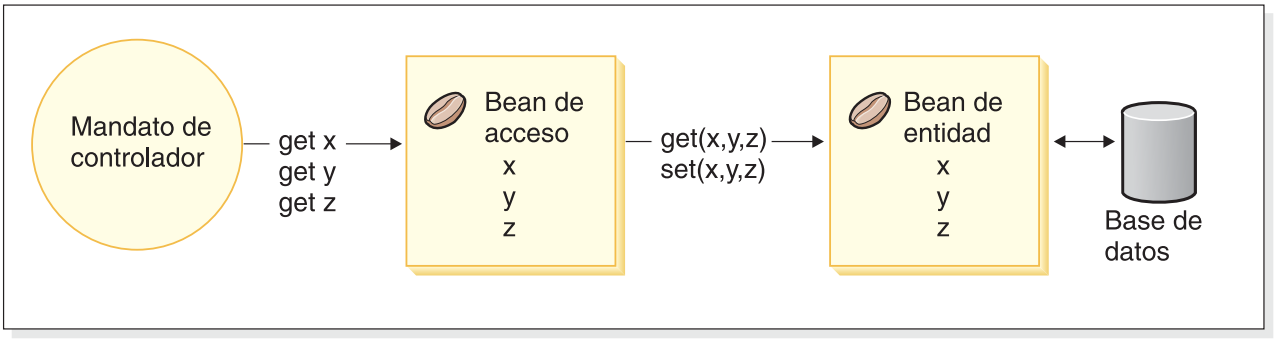


Figura 18.

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

En general, cuando cree tablas nuevas que siguen los convenios anteriores para los nombres de tabla, puede implementar su propio convenio de denominación para las columnas de dichas tablas. Se da por supuesto que siempre utiliza nombres de columna totalmente calificados en las sentencias SQL. Sin embargo, si desea realizar uniones con tablas de WebSphere Commerce existentes y no desea utilizar los nombres de columna totalmente calificados, deberá seguir los convenios de denominación de columna descritos en esta sección.

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 79 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 fraccionaria 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	AIX Linux Solaris Windows **DB2**	AIX Solaris Windows **Oracle**	400 **DB2**
Hashtable	BLOB()	BLOB	BLOB()
Timestamp	TIMESTAMP	DATE	TIMESTAMP
Integer	INTEGER	INTEGER	INTEGER
BigDecimal	DECIMAL(,)	DECIMAL(,)	DECIMAL(,)
Long	BIGINT	NUMBER(38,0)	BIGINT
Double	FLOAT	NUMBER(38,0)	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

Notas:

1. 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).
2. 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.

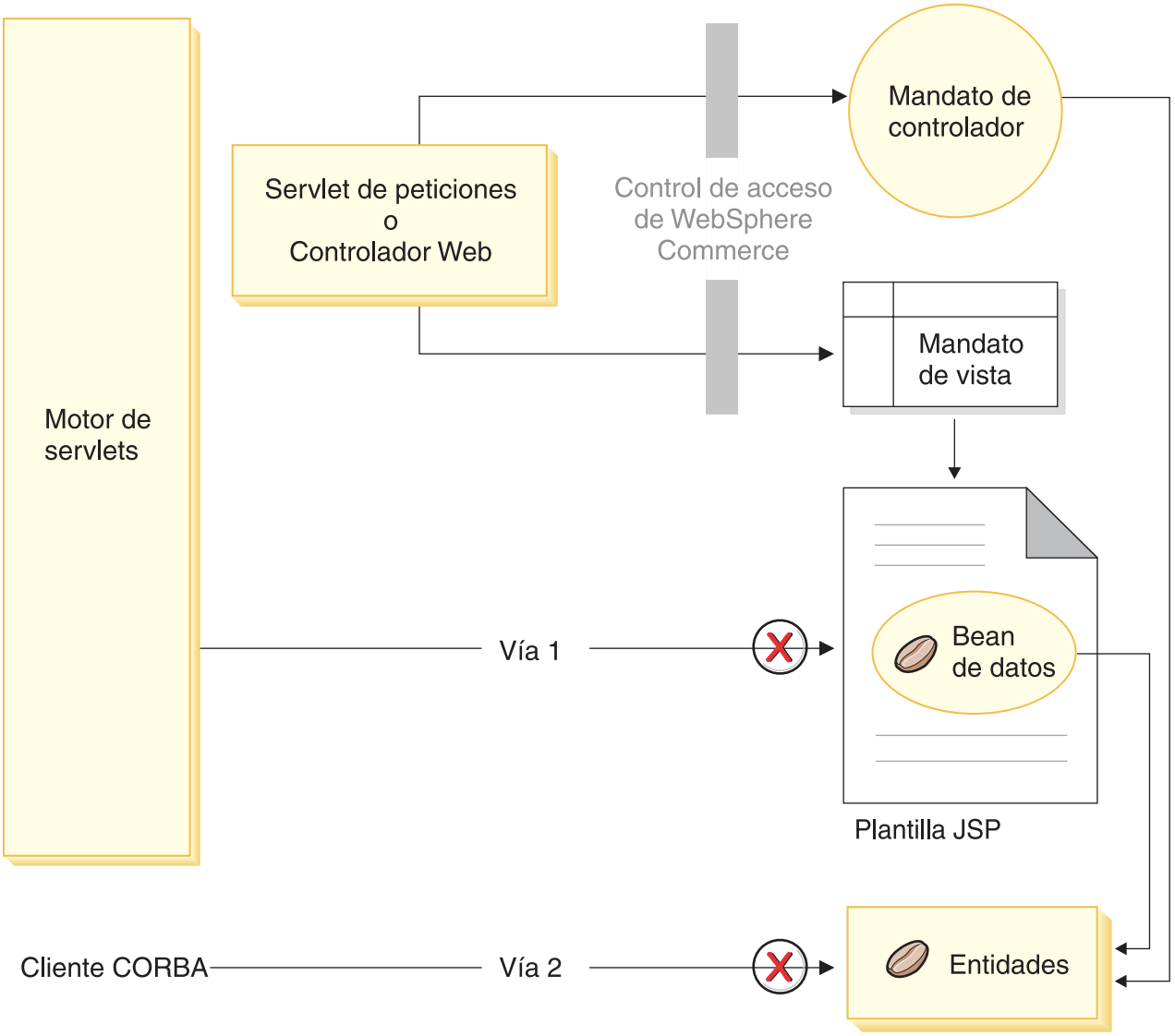


Figura 19.

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 diversas características de seguridad. WebSphere Commerce utiliza una de estas características para

asegurar que todos los métodos de bean enterprise y todas las plantillas JSP se configuren para que sólo los invoque una entidad elegida. Para acceder a estos recursos de WebSphere Commerce, se debe direccionar una petición de URL al servlet de peticiones. El servlet de peticiones establece la identidad elegida 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).

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

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.

Consideraciones acerca de la seguridad para los parámetros de URL

Los parámetros de URL proporcionan un modo útil para pasar información específica de petición. Para minimizar la posibilidad de que un usuario malicioso obtenga el acceso a la base de datos sin autorización, se deberá seguir una práctica de codificación determinada. Las partes de inserción, selección, actualización y supresión de las sentencias SQL deberán crearse en el momento del desarrollo. Se deberán utilizar inserciones 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, deberá evitar utilizar series de entrada como procedimiento para componer la sentencia SQL. A continuación se muestra un ejemplo de utilización de una serie de entrada:

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

La razón por la que se debe evitar utilizar la serie de entrada para componer la selección consiste en que la serie de entrada es fácil de manipular. Un usuario malicioso puede establecer una serie de entrada de este tipo en un valor inesperado y, potencialmente, obtener el acceso no autorizado o realizar operaciones no deseadas en la base de datos.

Introducción a las políticas de control de acceso de WebSphere Commerce

Esta sección proporciona una introducción muy breve a los principales componentes de la infraestructura de control de acceso de WebSphere Commerce. Se proporciona para que algunas de las tareas de programación relacionadas con el control de acceso se visualicen en el contexto correcto. Si necesita información sobre cómo configurar el control de acceso en un sistema de producción o si desea más detalles sobre la infraestructura de control de acceso, consulte la publicación *WebSphere Commerce, Guía de seguridad*.

El modelo de control de acceso de WebSphere Commerce se basa en la implantación de políticas de control de acceso. Estas políticas permiten exteriorizar 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, puede haber una política de control de acceso que permita a todos los usuarios que tienen el rol de Jefe de

producto realizar 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 a la producción y al desarrollo 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: Se pueden encontrar detalles sobre la información de XML para la columna Condiciones de la tabla ACRESGRP en la publicación *WebSphere Commerce, Guía de seguridad*.

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>

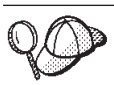
ACRESGRPES

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.ContractUpdateCmd
10248	com.ibm.commerce.contract. commands.ContractDeleteCmd
10249	com.ibm.commerce.contract. commands.ContractCancelCmd
10250	com.ibm.commerce.contract. commands.ContractCloseCmd

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 concepto importante relacionado con las políticas de control de acceso es el de los grupos de políticas de control de acceso. WebSphere Commerce Versión 5.5 soporta diversos modelos de negocio y cada uno de ellos tiene su propio conjunto de políticas de control de acceso. Para agrupar los conjuntos de políticas dentro de los modelos, se utilizan grupos de políticas. Las políticas se asignan explícitamente a los grupos de políticas apropiados y, a continuación, las organizaciones pueden suscribirse a uno o varios de estos grupos de políticas. En las versiones anteriores de WebSphere Commerce, una política se aplicaba a todos los recursos que eran propiedad de los descendientes de la organización propietaria de dicha política.

En WebSphere Commerce Versión 5.5, si una organización se suscribe a uno o más grupos de políticas, sólo se aplicarán a los recursos de dicha organización las políticas contenidas en esos grupos de políticas. Si un recurso es propiedad de una organización que no se suscribe a ningún grupo de políticas, el gestor de políticas de control de acceso buscará en la jerarquía de organizaciones hasta que encuentre la organización predecesora más próxima que esté suscrita como mínimo a un grupo de políticas; una vez que la haya encontrado, aplicará las políticas que pertenecen a dichos grupos de políticas.

Examine el diagrama siguiente:

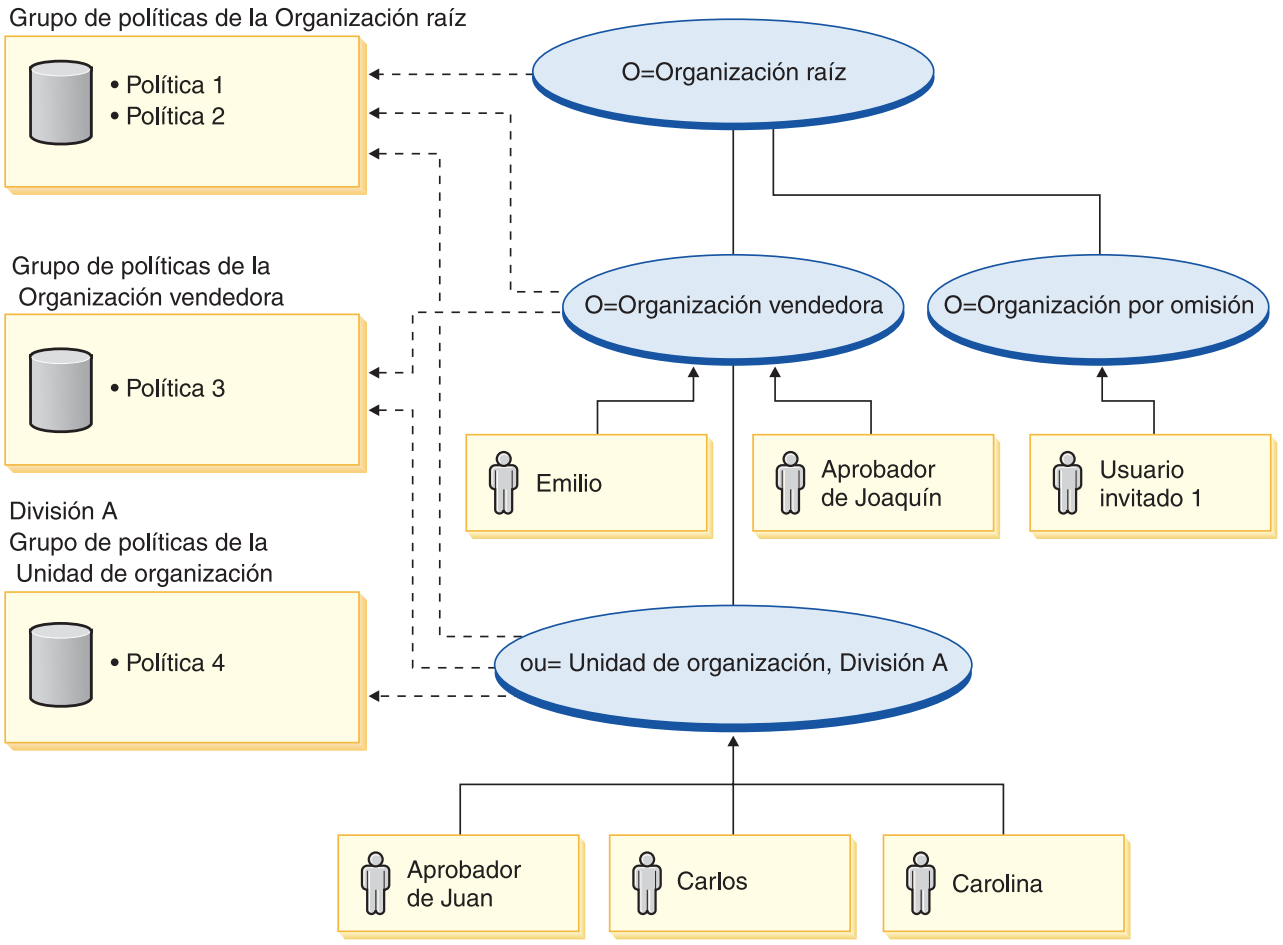


Figura 21.

Para los recursos que son propiedad de la Organización vendedora, se aplican las políticas del grupo de políticas de la Organización vendedora y las del grupo de políticas de la Organización raíz. Estas políticas son aplicables porque la Organización vendedora se suscribe explícitamente a esos dos grupos de políticas (las flechas de guiones muestran la suscripción). En total, se aplican tres políticas para esta organización. En este ejemplo la organización que era propietaria del recurso se suscribe explícitamente a los grupos de políticas. El diagrama también muestra un ejemplo en el que la organización no se suscribe explícitamente a ningún grupo de políticas, de modo que la infraestructura de control de acceso debe buscar más arriba en el jerarquía. Para los recursos que son propiedad de la Organización por omisión, se debe subir por la jerarquía hasta la Organización raíz. La Organización raíz sólo se suscribe a un grupo de políticas, de modo que ésas son las únicas políticas (políticas 1 y 2) que se aplican a los recursos de la Organización por omisión.

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: Express Professional Para las siguientes secciones sobre los grupos de relaciones, es importante tener en cuenta que las únicas organizaciones disponibles en WebSphere Commerce Professional Edition y WebSphere Commerce - Express son la organización raíz, la organización por omisión y

la organización vendedora. Business 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.

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

Cuando una cadena de relaciones tiene una longitud de 2, el primer elemento `<parameter name="unNombre" value="unValor" />` está entre un usuario y una entidad de organización. El último elemento `<parameter name="unNombre" value="unValor" />` está 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 crear su propio archivo XML basándose en el archivo defaultAccessControlPolicies.xml. Para obtener más información sobre cómo crear esta información basada en XML, consulte la publicación *WebSphere Commerce, Guía de seguridad*.

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. Si desea obtener información sobre la utilización de la Consola de administración para configurar grupos de acceso, consulte la Ayuda en línea a la producción de WebSphere Commerce.

ACPOLICY

PolicyName	Member_Id	MbrGrp_Id	AcActGrp_id	AcResGrp_Id	AcRelGrp_Id
StoreAdministrators ExecuteStoreAdministrators 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	StoreAdministratorsCmdResourceGroup

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 un diagrama de interacciones que muestra cómo funciona el control de acceso en la infraestructura de política de control de acceso de WebSphere Commerce.

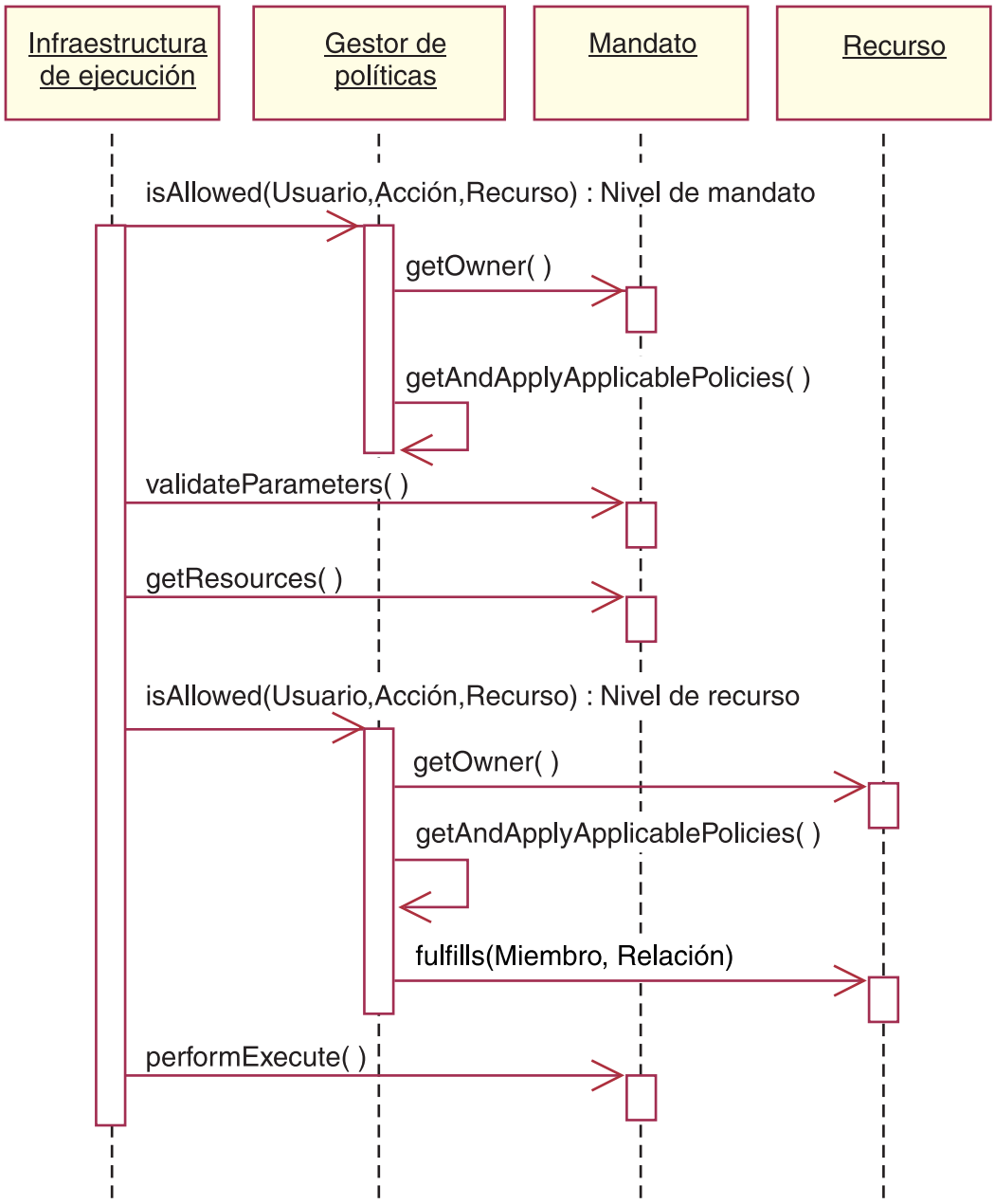


Figura 23.

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 lo determina buscando en las políticas de los grupos a los que está suscrito el propietario del recurso. Si el propietario del recurso no está suscrito a ningún grupo de políticas, busca en las políticas de los grupos a los que está suscrito el predecesor más cercano del propietario del recurso. 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 mandatos 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. `getAndApplyApplicablePolicies()`
   El Gestor de políticas de control de acceso busca y procesa las políticas aplicables, basándose en el usuario, la acción y el recurso especificados. Si como mínimo una política aplicable otorga el acceso, se pasa la comprobación de acceso a nivel de mandato y el gestor de políticas continuará en el siguiente paso para empezar a comprobar la autorización a nivel de recurso. Y, a la inversa, si ninguna de las políticas aplicables otorga acceso a nivel de mandato, el gestor de políticas vuelve a este punto y rechaza el acceso.
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 los beans enterprise” en la página 96.

   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.
8. `getAndApplyApplicablePolicies()`
   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 especificado una relación o un grupo de relaciones, se realiza una comprobación en el recurso para ver si el miembro satisface la relación especificada respecto al recurso.
10. `performExecute()`
    La lógica de negocio del mandato.

Interfaz Protectable

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 Protectable, 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 Groupable

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 seguridad*. 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 (Protectable). Normalmente, la cuestión de qué se debe proteger surge cuando se crean beans enterprise y beans de datos nuevos. 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 los 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 un recurso va a agruparse por un atributo distinto de su nombre de clase Java con el fin de aplicar políticas de control de acceso, la interfaz remota del bean también debe ampliar la interfaz `com.ibm.commerce.grouping.Groupable`.

4. La clase de bean enterprise hereda las implementaciones por omisión para los siguientes métodos de com.ibm.commerce.base.objects.ECEntityBean:
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

   Modifique los métodos que necesite. Como mínimo, deberá alterar el método `getOwner`.

   El método fulfills debe implementarse si existe una política de control de acceso que incluya este recurso en su grupo de recursos y también especifica una relación o un grupo de relaciones. El método getGroupingAttributeValue debe implementarse si existe una política de control de acceso con un grupo de recursos implícito que incluye determinadas instancias de este recurso, basándose en valores de atributo específicos (por ejemplo, si hay una política de control de acceso que sólo pertenece a los Pedidos en estado = 'P' (pendiente)).

   Tenga en cuenta que si la única relación necesaria es "propietario", no necesita alterar el método fulfills. En este caso, el gestor de políticas utilizará el resultado del método getOwner().

   En los fragmentos de código siguientes se muestran las implementaciones por omisión de estos métodos. Estas implementaciones provienen de la clase ECEntityBean.

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

   A continuación, se muestran implementaciones de ejemplo de estos métodos que se basan en las implementaciones utilizadas en el bean OrderBean:

   - Para el método getOwner, la lógica del método proporcionado es:

```
*************************************************************************
   com.ibm.commerce.common.objects.StoreEntityAccessBean storeEntAB =
      new com.ibm.commerce.common.objects.StoreEntityAccessBean();
   storeEntAB.setInitKey_storeEntityId(getStoreEntityId().toString());
   return storeEntAB.getMemberIdInEJBType();
*************************************************************************
```

   - Para el método fulfills, la lógica del método proporcionado es:

```
*************************************************************************
if ("creator".equalsIgnoreCase(relationship))
{
   return member.equals(bean.getMemberId());
}
else if ("BuyingOrganizationalEntity".equalsIgnoreCase(relationship))
{
   return (member.equals(bean.getOrganizationId()));
}
else if ("sameOrganizationalEntityAsCreator".
   equalsIgnoreCase(relationship))
{
   com.ibm.commerce.user.objects.UserAccessBean creator =
     new com.ibm.commerce.user.objects.UserAccessBean();
   creator.setInitKey_MemberId(bean.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;
*************************************************************************
```

- Para el método getGroupingAttributeValue, la lógica del método proporcionado es:

```
*************************************************************************
      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.

Tenga en cuenta que si examina otros beans de entidad públicos de WebSphere Commerce para comprender cómo se implementan los métodos getOwner, fulfills y getGroupingAttributeValue, observará que estos métodos se implementan en la clase de ayuda de acceso para los beans. Como consecuencia del hecho que los métodos se implementan en las clases de ayuda de acceso en lugar de hacerlo directamente en la clase de bean, las firmas de método son ligeramente diferentes. En particular, para que los métodos tomen un parámetro de entrada adicional para el objeto propiamente dicho que se debe pasar en la ayuda de acceso.

Cuando cree beans nuevos, *deberá* asegurarse de implementar estos métodos directamente en la clase de bean. Además, no deberá modificar ninguno de dichos métodos de las clases de ayuda de acceso de los beans de entidad públicos de WebSphere Commerce.

Implementación del control de acceso en los 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.

Para determinar si se debe proteger un bean de datos, tenga en cuenta lo siguiente:

1. ¿La información contenida en el bean de datos es información que necesita protección? Por ejemplo, ¿es información de naturaleza privada? Si no es así, no es necesaria la protección directa mediante el control de acceso. Si la respuesta es afirmativa, continúe.

2. Las instancias de los beans de datos las crean las vistas. ¿Creará la vista la instancia del bean de datos utilizando información del contexto de mandato (u otra información predeterminada diferente)? Si la respuesta es afirmativa y el control de acceso ya ha determinado que se permite el acceso, no será necesario la protección directa de este bean de datos. De lo contrario, continúe.
3. ¿Creará la vista la instancia del bean de datos utilizando información de algún parámetro de entrada? En este caso, debido al hecho de que no está seguro de si el control de acceso ya ha determinado si se permite o no se permite al usuario acceder a esta información, deberá proteger el nuevo bean de datos.

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)`.

   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 apropiados, basándose en las políticas de control de acceso existentes. El método `isAllowed` se describe con la siguiente sección de código:

   ```
   isAllowed(Context, "Display", beandatos_protegible);
   ```

   donde *beandatos_protegible* es el bean de datos que se debe proteger.
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 los 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`.

En el caso de las políticas de control de acceso 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. De acuerdo con 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 ampliando la implementación del mandato a partir de `com.ibm.commerce.command.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 `Protectable`. 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, altere el método `getResources` dado que la implementación por omisión de este método devuelve un valor nulo y, por consiguiente, 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.

Sólo es necesario especificar la acción cuando un mandato está realizando una operación de lectura y otra de grabación en instancias diferentes de la misma clase de recurso. Por ejemplo, en el caso del mandato OrderCopy, puede leer en un pedido de origen y grabar en un pedido de destino. En este caso, se debe realizar una diferenciación entre las dos acciones. Esto se logra especificando la acción "-Read" para el pedido de origen y la acción "-Write" para el pedido de destino. Cuando la infraestructura de control de acceso detecta estas acciones, las añade automáticamente al principio con el nombre de interfaz del mandato antes de buscar las políticas aplicables. En este caso, las acciones que se utilizarán finalmente en las políticas son "com.ibm.commerce.order.commands.OrderCopyCmd-Read" y "com.ibm.commerce.order.commands.OrderCopyCmd-Write".

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. Entonces, si es

necesario, puede utilizar la misma variable de instancia en el método performExecute del nuevo mandato de controlador.

Para determinar si debe alterar el método getResources, tenga en cuenta lo siguiente:

- Si obtiene el recurso basándose en una fuente predefinida de información, por ejemplo el contexto de mandato, no necesitará alterar el método getResources. Por ejemplo, el mandato de WebSphere Commerce UserRegistrationUpdate obtiene el ID del usuario del contexto de mandato. En este caso, el usuario ya está autorizado a actuar en su propia información de registro, de modo que no es necesario alterar el método getResources.
- Si el mandato está especificando de forma arbitraria un recurso nuevo (y este recurso es de naturaleza privada), deberá alterar el método getResources. Por ejemplo, el mandato de WebSphere Commerce OrderItemUpdate toma un ID de pedido como parámetro de entrada. En este caso, cuando se cree la instancia de recurso de pedido, no sabrá si el usuario tiene autorización para realizar acciones en ese recurso en particular. En este caso, se altera el método getResources.

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 uno o más objetos de pedido (que son protegibles), cuando está actualizando los pedidos existentes. Dado que no se especifica la acción, la acción toma por omisión 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 tiene acceso a dos de tres recursos, la acción no podrá continuar (es necesario tener acceso a los tres).

Si tiene que efectuar comprobación adicional de parámetros o resolución de parámetros en el mandato de controlador, puede utilizar el método validateParameters(). La utilización de este método 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 este caso, el método getResources debe devolver el contenedor del nuevo recurso. Por ejemplo, si se está creando un pedido, esto se realiza dentro del recurso de tienda y se crea un nuevo usuario dentro de un recurso de 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.

Modificación del control de acceso en los recursos de WebSphere Commerce existentes

Esta sección proporciona información para guiarle durante la modificación del control de acceso en los recursos de WebSphere Commerce existentes. En particular, se revisan los siguientes escenarios:

- Adición de una nueva relación a un bean de entidad de WebSphere Commerce existente que ya está protegido bajo el control de acceso.
- Adición de protección de control de acceso a un bean de entidad de WebSphere Commerce existente que aún *no* está protegido bajo el control de acceso.
- Interpretación de las implicaciones del control de acceso cuando se amplía un mandato de controlador existente.

Adición de una nueva relación a un bean de entidad de WebSphere Commerce existente

Los beans de entidad de WebSphere Commerce que implementan la interfaz Protectable ya están protegidos bajo el control de acceso. Los términos de los requisitos de control de acceso se determinan por el modo en que se utiliza el bean en las características y las funciones de WebSphere Commerce tal como se suministran. Es posible que encuentre situaciones en las que necesite añadir relaciones adicionales al control de acceso para dichos beans. Por ejemplo, si utiliza un bean existente en alguna parte del código personalizado o si modifica un bean de entidad público de WebSphere Commerce existente, es posible que necesite añadir relaciones adicionales al bean.

La lista siguiente proporciona los pasos de alto nivel para añadir relaciones nuevas en un bean de entidad de WebSphere Commerce existente que ya está protegido por el control de acceso:

1. Examine el método fulfills existente para el bean de entidad. Éste está ubicado en la clase de ayuda de acceso del bean. No modifique esta clase; utilícela sólo para determinar si necesita añadir una o más relaciones nuevas a esta lógica o si necesita alterar este método. Por ejemplo, el siguiente método fulfills aparece en la clase com.ibm.commerce.fulfillment.objsrc.FulfillmentCenterBeanAccessHelper:

```
public boolean fulfills(Object obj, Long member, String relationship)
  throws Exception {

 FulfillmentCenterBean bean = (FulfillmentCenterBean) obj;

 if ("ShippingArrangementOrganizationalEntity".
   equalsIgnoreCase(relationship))
 {
      FulfillmentJDBCHelperAccessBean ffmJDBCAB =
         new FulfillmentJDBCHelperAccessBean();
      int count = ffmJDBCAB.
         checkFulfillmentCenterByMemberIdAndFulfillmentCenterId(
         member,bean.getFulfillmentCenterId());
      if(count>0)
         return true;
 }
      return false;
}
```

2. El paso siguiente es crear un nuevo método fulfills en la clase de bean. Por ejemplo, puede crear un nuevo método fulfills en la clase com.ibm.commerce.fulfillment.objects.FulfillmentBean.java. La declaración para el método debe aparecer como:

```
public boolean fulfills(Long member, String relationship)
  throws Exception {
      // Espacio reservado para información de relación
 }
```

3. Si está añadiendo una relación adicional a las relaciones existentes, la primera línea del método debe ser para llamar al método fulfills desde la superclase. Entonces, si dicho método devuelve 'false', compruebe las nuevas relaciones como se indica a continuación:

```
public boolean fulfills(Long member, String relationship)
  throws Exception {
      if (super.fulfills().equals(false))
      {
      // Comprobar si se cumple la nueva relación
         return true;
      }

      return false;
 }
```

4. Si está sustituyendo enteramente las relaciones a partir de la implementación original, no deberá llamar al método super.fulfills, como se indica a continuación:

```
public boolean fulfills(Long member, String relationship)
  throws Exception {
      // Comprobar si se cumple la nueva relación
      // Si se cumple, devolver verdadero;

     // Si no se cumple la relación, devolver falso;
 }
```

5. Guarde los cambios. Vuelva a generar el código desplegado y RMIC para el bean, así como el bean de acceso correspondiente.

Adición de control de acceso a un bean de entidad de WebSphere Commerce existente que aún no está protegido

Si está utilizando un bean de entidad de WebSphere Commerce existente y la aplicación necesita que el bean esté protegido mediante el control de acceso, puede añadir esta protección.

La lista siguiente proporciona los pasos de alto nivel para proteger un bean de entidad de WebSphere Commerce existente bajo el sistema de control de acceso de WebSphere Commerce:

1. Abra la clase *NombreBean*.java. Ésta es la interfaz remota. Modifíquela para que amplíe la interfaz `com.ibm.commerce.security.Protectable`.
2. Si se va a agrupar un recurso por un atributo distinto de su nombre de clase Java con el fin de aplicar políticas de control de acceso, la interfaz remota del bean también debe ampliar la interfaz `com.ibm.commerce.grouping.Groupable`.
3. Guarde los cambios efectuados en la interfaz remota.
4. Abra la clase *NombreBean*Bean.java, donde *NombreBean* es el nombre del bean de entidad al que está añadiendo protección de control de acceso.
5. La clase de bean enterprise hereda de com.ibm.commerce.base.objects.ECEntityBean las implementaciones por omisión para los métodos siguientes:
   - `getOwner`
   - `fulfills`
   - `getGroupingAttributeValue`

   Modifique los métodos que necesite. Como mínimo, deberá alterar el método `getOwner`. Consulte el apartado "Implementación del control de acceso en los beans enterprise" en la página 96 para obtener más información sobre estos métodos.
6. Guarde los cambios. Vuelva a generar el código desplegado y RMIC para el bean, así como el bean de acceso correspondiente.

Interpretación de las implicaciones del control de acceso cuando se amplía un mandato de controlador

De acuerdo con el modelo de programación de WebSphere Commerce, puede crear sus propias implementaciones de los mandatos de controlador existentes. En este caso, cree una nueva clase de implementación y, a continuación, asocie la nueva clase de implementación a la interfaz existente actualizando el registro de mandato.

La realización de una ampliación de este tipo tiene tres implicaciones potenciales relacionadas con el control de acceso:

1. Efecto en el método getResources.
2. Efecto en las políticas de control de acceso a nivel de mandato
3. Efecto en las políticas de control de acceso a nivel de recurso

En las secciones subsiguientes, se describe cada uno de estos puntos más detalladamente.

Efecto en el método getResources

Si está ampliando un mandato de controlador existente, lo que significa que se ejecutará la lógica existente del mandato así como la nueva lógica personalizada, el nuevo mandato será una subclase del mandato existente. El método performExecute de la nueva implementación llama al método performExecute de la superclase. Si el nuevo mandato no accede a ningún recurso nuevo que requiera protección, no tendrá que alterar el método getResources. Sin embargo, si se accede a nuevos recursos protegibles, el nuevo mandato deberá implementar su propio método getResources y en este método llamar al método getResources de la superclase, antes de implementar la lógica getResources propia.

Los resultados del método getResources de la superclase deben almacenarse en una variable de instancia privada de tipo AccessVector. Entonces se deberán añadir los resultados del método getResources local al final de este vector. Por ejemplo, la nueva clase de implementación contendrá código similar al pseudocódigo siguientes:

```
private AccessVector resources = null;

public AccessVector getResources() throws ECException {
   // Primero, obtener los recursos de la implementación original
   resources = super.getResources();

   // Ahora, añadir los nuevos recursos

   ////////////////////////////////////
   // Lógica para obtener nuevos recursos //
   // y añadirlos al vector.             //
  ////////////////////////////////////

   return resources;
}
```

Si no está ampliando la lógica de un mandato de controlador existente, sino que está sustituyendo completamente la lógica, no deberá llamar al método super.performExecute en la nueva implementación. Entonces no necesitará llamar al método super.getResources desde dentro de su propia implementación. En lugar de ello, simplemente implemente su propio método getResources, como sea apropiado.

Efecto en las políticas de control de acceso a nivel de mandato

Las políticas a nivel de mandato para los mandatos de controlador constan de la acción “Execute” que se actúa en el recurso de mandato. El recurso de mandato se especifica mediante el nombre de interfaz. Al ampliar un mandato existente, el mandato sigue implementando la interfaz original y, en sí, la política existente a nivel de mandato es suficiente para mantener el control de acceso anterior a nivel de mandato. Por consiguiente, no es necesario ningún cambio.

Efecto en las políticas de control de acceso a nivel de recurso

Si ha creado una nueva implementación de un mandato existente y ha asociado esta implementación a la interfaz de ese mandato existente, no es necesario realizar cambios en las políticas de control de acceso a nivel de recurso.

Si, en lugar de ello, crea una nueva clase de implementación que amplía una implementación existente y en esta clase llama al método super.performExecute y también implementa una nueva interfaz, será necesario realizar cambios en las políticas de control de acceso a nivel de recurso.

En este último caso, si el nuevo mandato implementa el método getResources o hereda una implementación no trivial del método getResources del mandato base, será necesario realizar cambios de política a nivel de recurso. Normalmente, las políticas a nivel de recurso constan de la acción de mandato que actúa en el recurso de objeto de negocio. Dado que la acción es simplemente una representación de serie de la interfaz del mandato, generalmente es necesario añadir el nuevo mandato a los grupos de acciones del mandato base. Sin embargo, si el nuevo mandato altera la implementación base del método getResources devolviendo simplemente nulo (null), no se realizará ninguna comprobación de control de acceso para este nuevo mandato. Tenga en cuenta que si esto no se realiza cuidadosamente, el nuevo mandato se podría abrir para usuarios mal intencionados.

Cuando se modifican las políticas de control de acceso a nivel de recurso, los pasos de alto nivel a realizar son los siguientes:

1. Localice el archivo defaultAccessControlPolicies.xml, que se encuentra en el directorio siguiente:
   - Developer *dir_instal_WCDE*\Commerce\xml\policies\xml
2. Realice una copia de este archivo. Por ejemplo, denomine el archivo *myDefaultAccessControlPolicies.xml*.
3. En este nuevo archivo, debe definir la nueva interfaz como una acción nueva. Por ejemplo, añada lo siguiente:

   ```
   <Action Name="yourNewInterface"
      CommandName="yourNewInterface">
   </Action>
   ```

   donde *yourNewInterface* es el nombre de la nueva interfaz.
4. A continuación, deberá buscar en el archivo para localizar todos los grupos de acciones a los que pertenecía la acción original y, a continuación, añadir la nueva acción.
5. Si el nuevo mandato está actuando en recursos que el mandato base no especificaba anteriormente, el grupo de recursos de las políticas de control de acceso correspondientes también se deberán cambiar para adaptar los nuevos recursos.
6. Una vez que haya completado los cambios en el archivo XML, podrá eliminar las secciones que no haya modificado. Asegúrese de conservar el texto que se encuentra antes del código `<Policies>`.
7. Cargue la nueva información de política en la base de datos, de acuerdo con las instrucciones contenidas en la publicación *WebSphere Commerce, Guía de seguridad*.

Políticas de control de acceso de ejemplo para el desarrollo

Esta sección proporciona algunas políticas de control de acceso muy simples que se pueden utilizar en el entorno de desarrollo para que pueda probar rápidamente los recursos nuevos. No están diseñadas para utilizarse en cualquier entorno de producción de WebSphere Commerce porque no proporcionan la protección de recursos adecuada.

Para obtener información sobre cómo cargar estas políticas, consulte la publicación *WebSphere Commerce, Guía de seguridad*.

Política de control de acceso de ejemplo para vistas nuevas

Si crea una vista nueva, puede utilizar la siguiente política de control de acceso para poder probar la nueva vista en el entorno de desarrollo (modifique la política para el entorno y cárguela utilizando el mandato acpload):

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="YourNewView"
        CommandName="YourNewView">
   </Action>
   <ActionGroup Name="AllSiteUsersViews"
      OwnerID="RootOrganization">
      <ActionGroupAction Name="YourNewView"/>
   </ActionGroup>
</Policies>
```

donde *YoutNewView* es el nombre de la vista recién creada. La política de control de acceso anterior añade la nueva vista en el grupo de acciones AllSiteUsersViews existente. Esta política permite a cualquier usuario acceder a la nueva vista.

Política de control de acceso a nivel de mandato de ejemplo para mandatos de controlador nuevos

Los mandatos de controlador necesitan políticas de control de acceso para satisfacer los requisitos de la infraestructura de control de acceso. Si crea un nuevo mandato de controlador, el nombre de la interfaz del mandato se especifica como un recurso. La sección de código XML siguiente se puede modificar para el nuevo mandato y se puede cargar utilizando el mandato acpload:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>

   <Action Name="ExecuteCommand"
         CommandName="Execute">
   </Action>

   <ResourceCategory Name="com.yourcompany.yourpackage.commands.
         YourControllerCmdResourceCategory"
         ResourceBeanClass="com.yourcompany.yourpackage.commands.
         YourControllerCmd">
        <ResourceAction Name="ExecuteCommand"/>
   </ResourceCategory>

   <ResourceGroup Name="AllSiteUserCmdResourceGroup"
       OwnerID="RootOrganization">
      <ResourceGroupResource Name="com.yourcompany.yourpackage.commands.
         YourControllerCmdResourceCategory" />
   </ResourceGroup>

</Policies>
```

donde:

- *com.yourcompany.yourpackage.commands* representa la estructura de empaquetado
- *YourControllerCmd* representa el nombre del nuevo mandato de controlador

Por ejemplo, el archivo XML siguiente se utiliza para cargar la política de control de acceso para un nuevo mandato de controlador que se crea en una guía de aprendizaje contenida en este manual.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="ExecuteCommand"
   CommandName="Execute">
   </Action>

   <ResourceCategory Name="com.ibm.commerce.sample.commands.
          MyNewControllerCmdResourceCategory"
          ResourceBeanClass="com.ibm.commerce.sample.commands.
          MyNewControllerCmd">
         <ResourceAction Name="ExecuteCommand"/>
   </ResourceCategory>

   <ResourceGroup Name="AllSiteUserCmdResourceGroup"
       OwnerID="RootOrganization">
      <ResourceGroupResource Name="com.ibm.commerce.sample.commands.
```

```
        MyNewControllerCmdResourceCategory" />
   </ResourceGroup>

</Policies>
```

Política de control de acceso a nivel de recurso de ejemplo para un mandato y un bean enterprise nuevos

El siguiente archivo XML se toma de una guía de aprendizaje contenida en este manual. Puede actuar como plantilla para los requisitos de control de acceso al crear nuevos beans de entidad. En el caso del archivo siguiente, el nuevo bean de entidad se denomina bean Bonus, corresponde a la tabla de base de datos XBONUS y lo utiliza el mandato de controlador MyNewControllerCmd. En esta política de control de acceso, sólo el creador de un objeto bean bonus puede realizar la acción MyNewControllerCmd en dicho objeto.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no" ?>
<!DOCTYPE Policies SYSTEM "../dtd/accesscontrolpolicies.dtd">

<Policies>
   <Action Name="MyNewControllerCmd"
      CommandName="com.ibm.commerce.sample.commands.MyNewControllerCmd">
   </Action>

   <ResourceCategory Name="com.ibm.commerce.extension.objects.
       BonusResourceCategory"
       ResourceBeanClass="com.ibm.commerce.extension.objects.Bonus" >

         <ResourceAction Name="MyNewControllerCmd" />
         </ResourceCategory>

   <ActionGroup Name="MyNewControllerCmdActionGroup"
      OwnerID="RootOrganization">
   <ActionGroupAction Name="MyNewControllerCmd"/>
   </ActionGroup>

   <ResourceGroup Name="BonusResourceGroup"  OwnerID="RootOrganization" >
      <ResourceGroupResource Name="com.ibm.commerce.extension.objects.
            BonusResourceCategory" />
   </ResourceGroup>
   <Policy Name="AllUsersUpdateBonusResourceGroup"
      OwnerID="FashionFlowMemberId"
      UserGroup="AllUsers"
      UserGroupOwner="RootOrganization"
      ActionGroupName="MyNewControllerCmdActionGroup"
      ResourceGroupName="BonusResourceGroup"
      RelationName="creator"
      PolicyType="groupableStandard">
   </Policy>

   <PolicyGroup Name="ManagementAndAdministrationPolicyGroup"
         OwnerID="RootOrganization">
   <!-- Define policies in this policy group -->
   <PolicyGroupPolicy  Name="AllUsersUpdateBonusResourceGroup"
            PolicyOwnerID="FashionFlowMemberId" />

   </PolicyGroup>

</Policies>
```

donde *FashionFlowMemberId* es el ID de miembro de la tienda en la que se está utilizando el nuevo recurso.

En la política de control de acceso anterior, el nombre de interfaz del mandato de controlador se especifica como la acción, sin calificarlo totalmente con el nombre

de paquete. Si la aplicación tiene varias interfaces con el mismo nombre, deberá calificarlas totalmente con los nombres de paquete cuando las especifique como acciones en las políticas de control de acceso. Por ejemplo, si hubiera ambigüedad con los nombres de interfaz, sería necesario realizar los cambios siguientes en la política de control de acceso anterior (tenga en cuenta que sólo se visualizan las líneas cambiadas y que las modificaciones se muestran en negrita):

```
<Action Name="com.ibm.commerce.sample.commands.MyNewControllerCmd"
CommandName="com.ibm.commerce.sample.commands.MyNewControllerCmd">
.
.
.
<ResourceAction Name="com.ibm.commerce.sample.commands.MyNewControllerCmd" />
.
.
.
<ActionGroupAction Name="com.ibm.commerce.sample.commands.MyNewControllerCmd"/>
```

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 del sistema, ésta se anota cronológicamente de forma automática.

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.

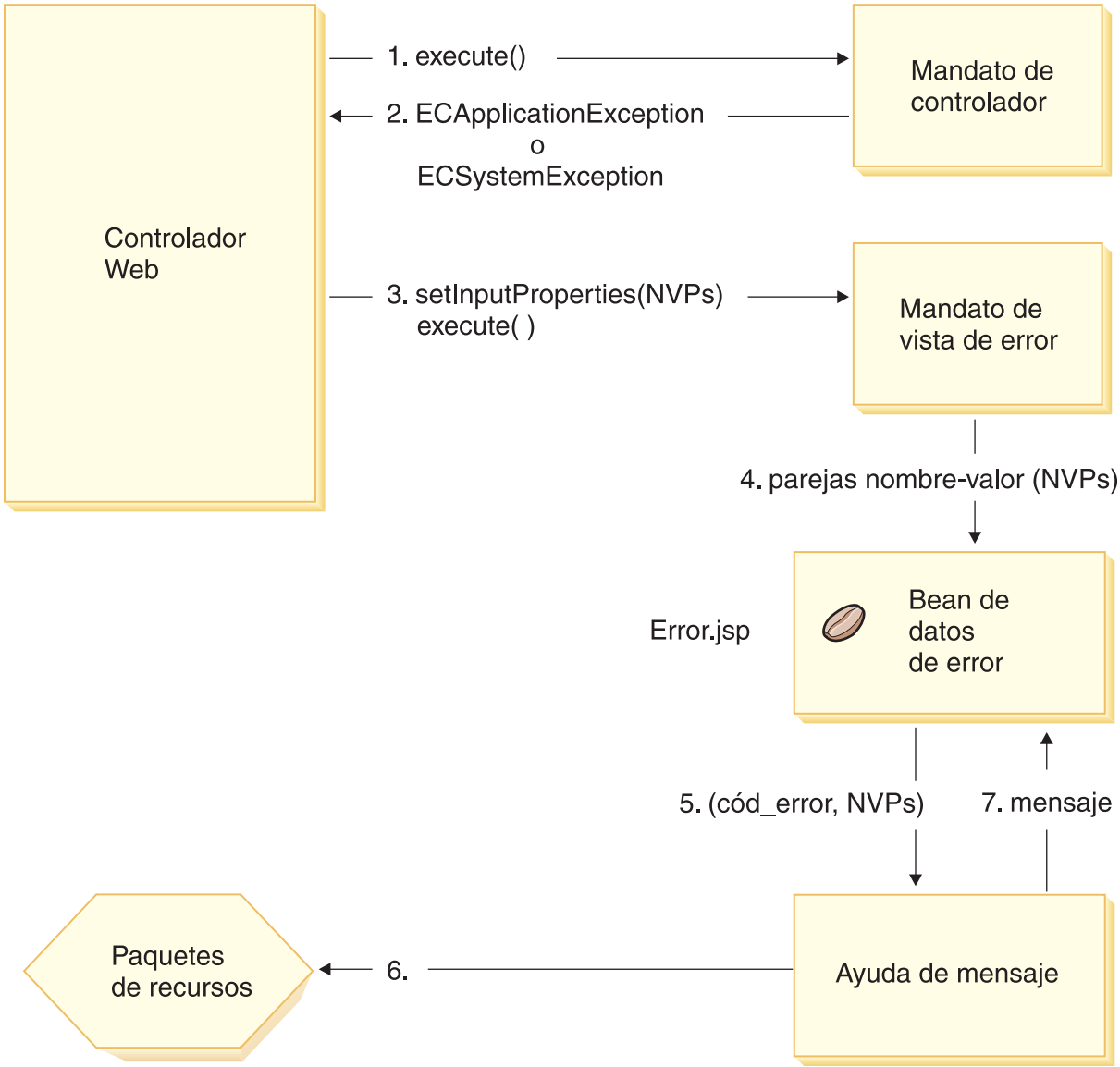


Figura 24.

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 de 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.

Objeto de mensaje	Descripción
_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.

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 del paquete propio y se debe almacenar en el proyecto WebSphereCommerceServerExtensionsLogic.

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 contenido 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 directorio siguiente:

Developer *dir_espaciotrabajo*\Stores\Web Content\WEB-INF\classes*dirTienda*
donde *dirTienda* es el nombre de la tienda.

Si los mensajes los va a utilizar WebSphere Commerce Accelerator, coloque este archivo en el directorio siguiente:

Developer *dir_espaciotrabajo*\CommerceAccelerator\Web Content\WEB-INF\classes

Si los mensajes los va a utilizar la Consola de administración, coloque este archivo en el directorio siguiente:

Developer *dir_espaciotrabajo*\SiteAdministration\Web Content\WEB-INF\classes

Business Si los mensajes los va a utilizar la Consola de administración de la Organización, coloque este archivo en el directorio siguiente:

Developer *dir_espaciotrabajo*\OrganizationAdministration\Web Content\WEB-INF\classes

Si los mensajes los va a utilizar globalmente cualquier servlet de la aplicación de empresa, coloque este archivo en el directorio siguiente:

Developer *dir_espaciotrabajo*\WebSphereCommerceServer\properties

Los directorios anteriores se especifican dentro del contexto del entorno de desarrollo. Una vez que haya realizado las pruebas en dicho entorno, consulte el Capítulo 9, “Información detallada sobre el despliegue”, en la página 179 para obtener información sobre cómo desplegar en el WebSphere Commerce Server de destino.

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}".
```

Rastreo del flujo de ejecución

Cuando necesite hacer el seguimiento del flujo de ejecución en la aplicación de comercio, deberá utilizar el recurso JRas de WebSphere Application Server. Se trata de una API de rastreo de diagnóstico y anotación cronológica de mensajes que las aplicaciones pueden utilizar.

Para obtener información sobre cómo utilizar este recurso en el código personalizado, consulte el InfoCenter de WebSphere Application Server.

Para obtener información sobre cómo configurar el rastreo de componentes en el entorno de desarrollo, consulte el Apéndice A, “Configuración del rastreo de componentes de WebSphere Commerce en el Entorno de desarrollo de WebSphere Commerce”, en la página 331.

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 negocio. Para obtener información sobre estos mandatos, consulte el Capítulo 7, "Acuerdos comerciales y políticas de negocio (Business Edition)", en la página 141.

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.

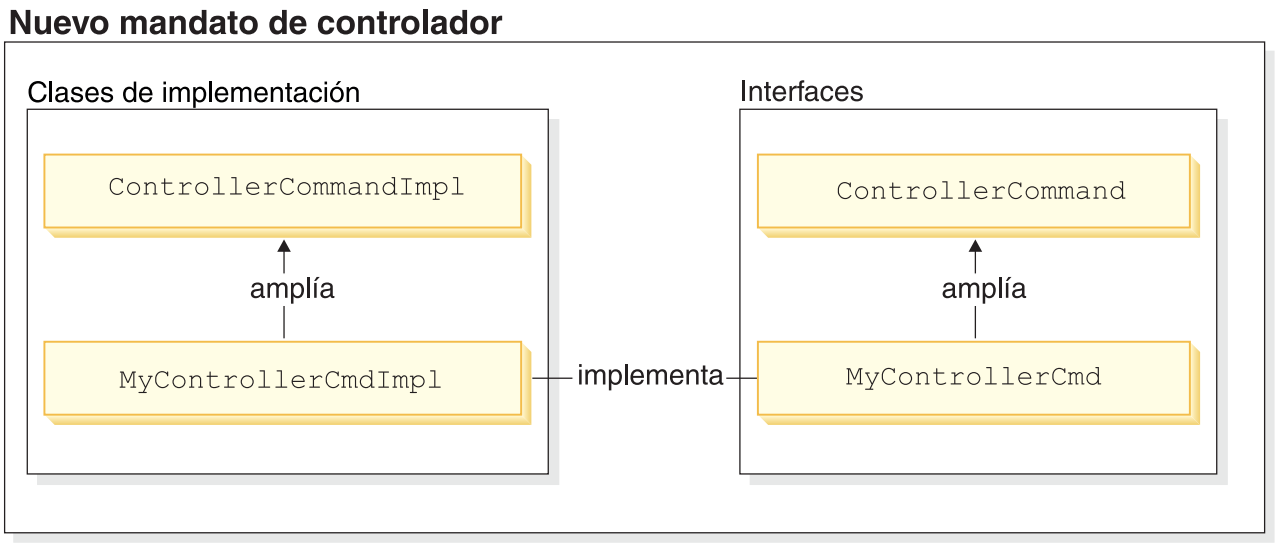


Figura 25.

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.

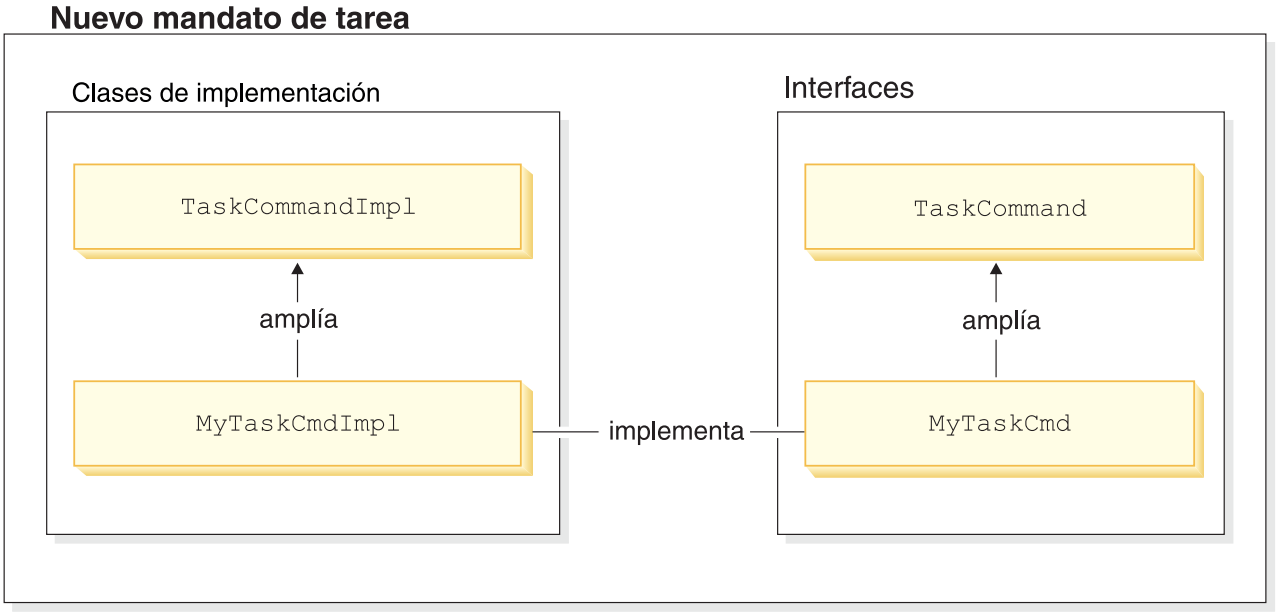


Figura 26.

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.

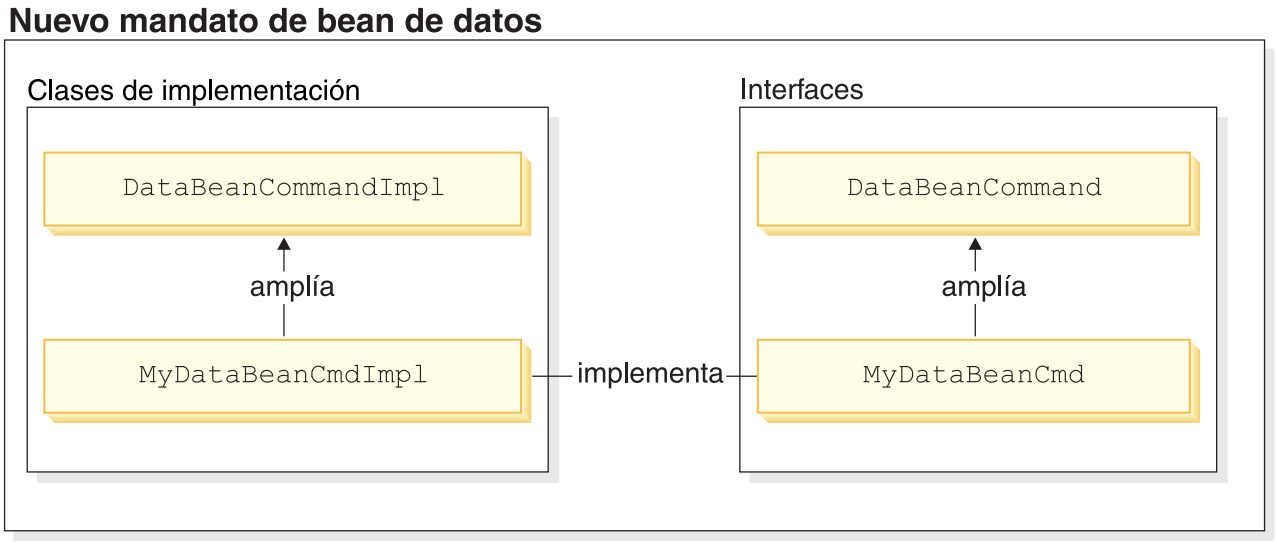


Figura 27.

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 proyectos del espacio de trabajo de WebSphere Commerce que están predefinidos para el código personalizado. Se proporcionan dos proyectos predefinidos; el proyecto WebSphereCommerceServerExtensionsLogic y el proyecto WebSphereCommerceServerExtensionsData. El primer objeto es para la lógica de beans de datos y de mandatos y el segundo es para los beans enterprise que se crean.

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.

Al crear beans de datos nuevos, éstos deben mantenerse en un paquete que sea independiente de la lógica de mandatos, aunque este paquete debe mantenerse dentro del proyecto (WebSphereCommerceServerExtensionsLogic) que almacena los paquetes de mandatos. Siguiendo el ejemplo anterior, puede poner el paquete `com.bigbusiness.databeans` en el proyecto WebSphereCommerceServerExtensionsLogic.

Al crear beans de entidad nuevos, éstos deben almacenarse en el proyecto WebSphereCommerceServerExtensionsData. Por consiguiente, puede tener el proyecto WebSphereCommerceServerExtensionsData que contiene el paquete `com.bigbusiness.objects`.

Esta estrategia de creación de paquetes es necesaria a efectos de despliegue del código.

Contexto de mandatos

Los mandatos pueden obtener información del controlador Web utilizando el contexto de mandatos. El ID del usuario, el objeto de usuario, el identificador de idioma y el identificador de tienda son ejemplos de la información disponible.

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. Dado que la moneda forma parte de la Infraestructura de globalización, la lógica que hay detrás de este método es similar 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 el uso del entorno de ejecución de WebSphere Commerce y es posible que se dejen de utilizar en los releases futuros.

Para obtener detalles completos sobre la API (Interfaz de programas de aplicación) de contexto de mandatos, consulte el tema "Referencias" de la Ayuda en línea a la producción y al desarrollo de WebSphere Commerce.

Cambios temporales en la información contextual para los mandatos de URL

Es posible alterar parte de la información de contexto de mandatos y ejecutar los mandatos de URL dentro del contexto de otra tienda o en nombre de otro usuario. Los mandatos de URL tienen los siguientes parámetros de entrada de URL que permiten esta conmutación temporal en el contexto de mandatos:

- forStoreId
- forUser
- forUserId

El parámetro de entrada de URL forStoreId le permite especificar el ID de tienda que se debe utilizar para esa petición de URL en particular. En realidad, éste cambia temporalmente el valor de storeId en el contexto de mandatos por el de la tienda especificada, pero dicho cambio sólo es válido durante el tiempo que dura el mandato de URL.

Los parámetros de entrada de URL forUser y forUserId le permiten especificar que el mandato se ejecute para el usuario especificado, incluso si el usuario que ha iniciado la sesión actualmente sea diferente. Esto es especialmente útil cuando un representante de servicio al cliente necesita ayudar a un cliente. Por ejemplo, el representante de servicio al cliente puede actualizar la información de dirección de un cliente en nombre de dicho cliente, especificando el nombre de usuario o el ID de usuario del cliente mediante el uso de los parámetros de entrada de URL. Este

cambio en la información de usuario sólo es válido durante el tiempo que dura la petición de URL para la que se ha especificado.

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 92 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 81.

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                     //
//////////////////////////////////////////////

// Create a new TypedProperty for response properties.
TypedProperty rspProp = new TypedProperty();

// set response properties
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);

////////////////////////////////////////////////////////////////////////////
//   Si utiliza dirigir vista, puede establecer las propiedades de        //
//   respuesta de la sigiente manera:                                     //
//   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:

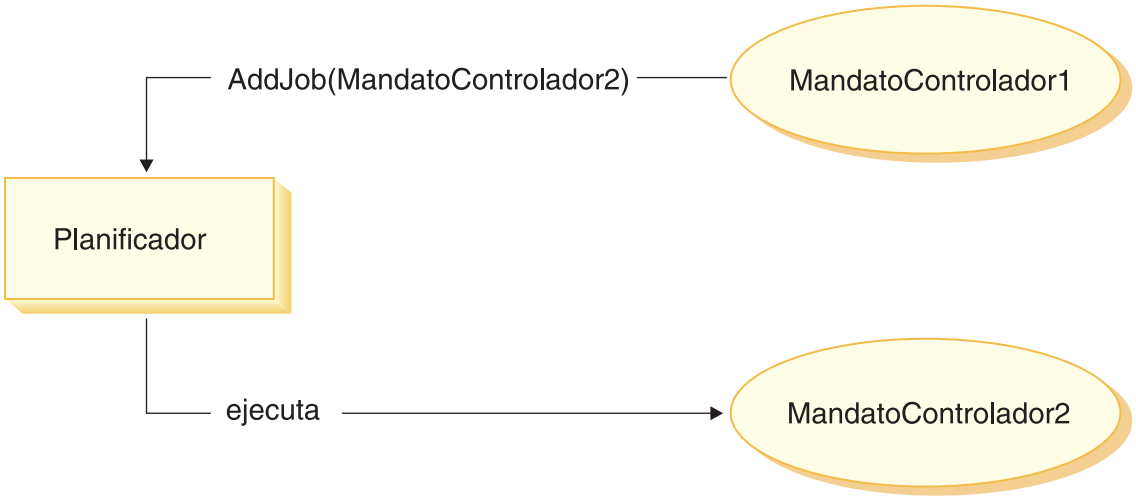


Figura 28.

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 41. 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 lo tanto, puede producirse un problema si hay un gran número de parámetros de entrada o si está utilizando el cifrado, porque una serie cifrada es normalmente dos o tres veces más larga que una serie 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 infraestructura 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 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 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 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 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. Entonces esto se implementa como un reenvío de vista genérico.

Nuevos mandatos de tarea

Un mandato de tarea nuevo debe ampliarse de la clase de mandato de tarea abstracta (com.ibm.commerce.command.TaskCommandImpl) e implementar una interfaz que amplíe la interfaz com.ibm.commerce.command.TaskCommand. Tal como se muestra en el diagrama de la página 120, 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 los mandatos de controlador existentes

Un mandato de controlador encapsula la lógica de negocio para un proceso de negocio. Las 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 implementación del mandato de tarea que desea alterar por la implementación de un mandato de tarea nuevo que realice la tarea del modo deseado.

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 136 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 utilizada por un mandato de controlador existente, consulte el tema “Referencias” de la Ayuda en línea a la producción y al desarrollo 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 fragmento de código siguiente muestra cómo añadir lógica de negocio nueva 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 el tema "Referencias" de la Ayuda en línea a la producción y al desarrollo de WebSphere Commerce para asegurarse de que la nueva clase de implementación proporcione 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 WebSphere Studio Application Developer, 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
{

}
```

WebSphere Studio Application Developer 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 un proceso en línea de RFQ o negociarlo fuera de línea y, a continuación, capturarlo utilizando las interfaces de Gestión de relaciones comerciales de 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:

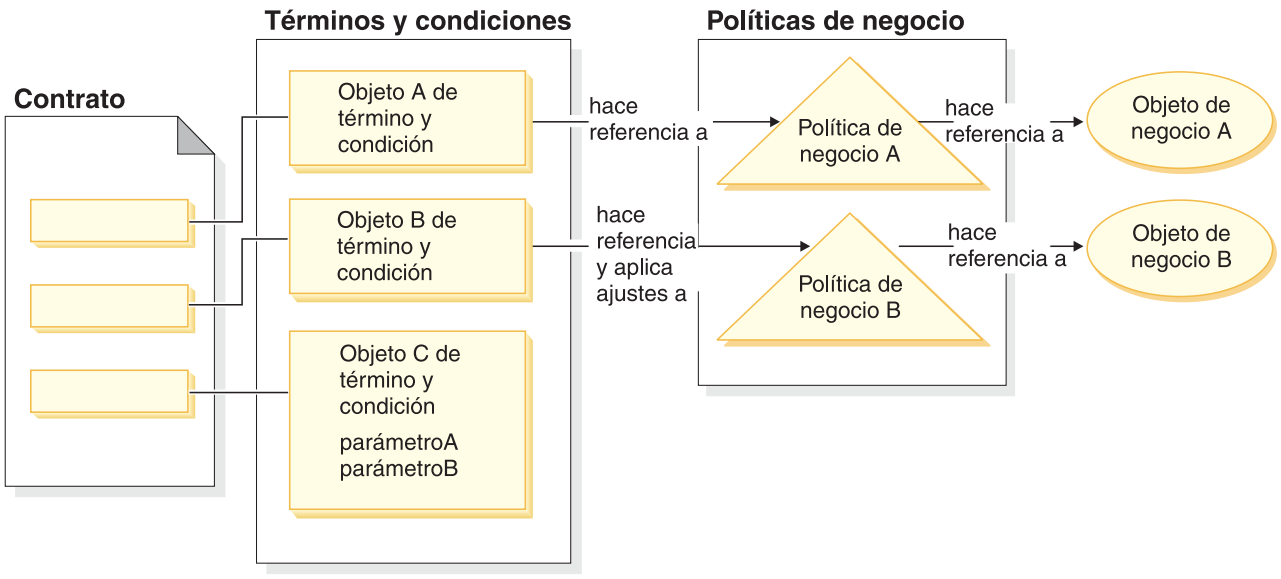


Figura 29.

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, en el diagrama 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.
- Otro ejemplo, el objeto de término y condición B del diagrama 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 el catálogo maestro.

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 soportan estos ejemplos, consulte el apartado "Datos del contrato de la tienda de ejemplo ToolTech" en la página 144.

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 precio para un término y condición. 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:

Nuevo mandato de política de negocio

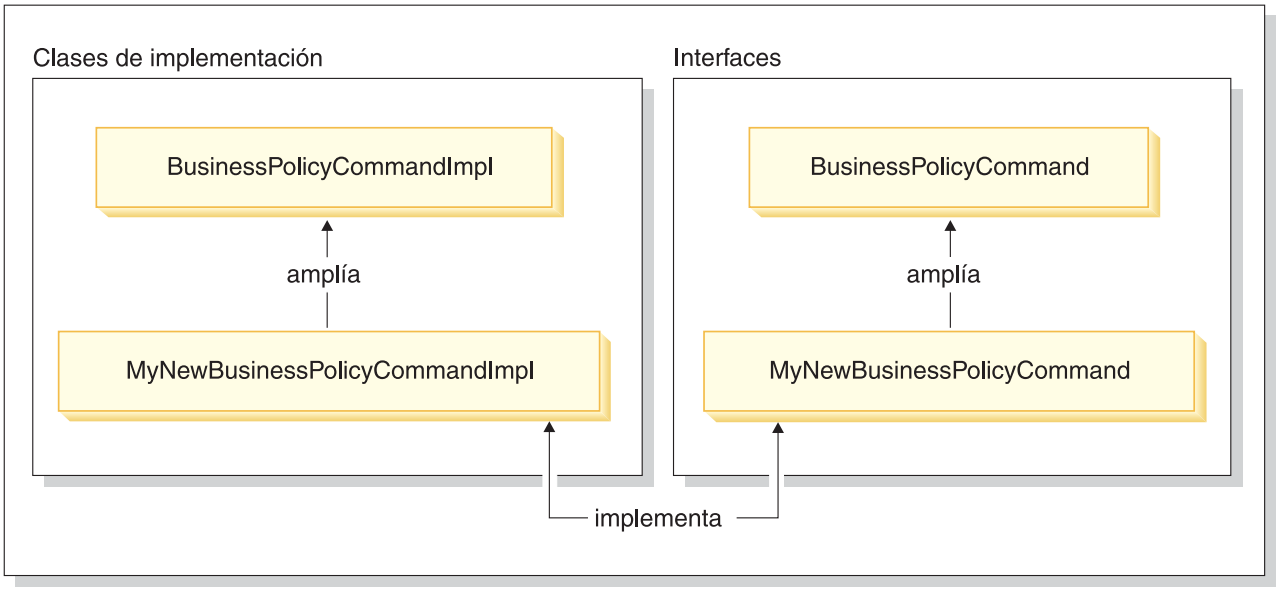


Figura 30.

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 datos de ejemplo pertinentes de la tabla de base de datos CONTRACT. Tenga en cuenta que, para mostrarlo, las cabeceras de columna de base de datos aparecen en la primera columna y la fila de datos de ejemplo de la tabla aparecen 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 datos de ejemplo pertinentes de la tabla de base de datos TERMCOND. 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 datos de ejemplo pertinentes de la tabla de base de datos POLICYTC. 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
Fila 2 de datos de ejemplo	10056	10030

Datos de ejemplo de la tabla POLICY

La tabla siguiente muestra datos de ejemplo pertinentes de la tabla de base de datos POLICY.

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 datos de ejemplo pertinentes de la tabla de base de datos TRADEPOSCN.

	Nombre de columna			
	TRADEEPOSCN_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 datos de ejemplo pertinentes de la tabla de base de datos SHIPMODE.

	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 se puede realizar relacionando un objeto de términos y condiciones existente con la nueva política de negocio o creando un objeto de términos y condiciones nuevo. 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 XSD (definición de esquema XML) del contrato
   c. Crear un nuevo bean enterprise CMP para el término y condición
   d. Actualizar WebSphere Commerce Accelerator para reflejar el nuevo término y condición
3. Invocar la nueva política de negocio durante el flujo de negocio.

WebSphere Commerce Versión 5.5 presenta nuevos tipos de contratos. Para obtener información sobre los tipos de contratos disponibles, consulte el subtema “Cuentas de negocio” del tema “Conceptos” de la Ayuda en línea a la producción de WebSphere Commerce.

Las secciones siguientes utilizan BuyerContract como el tipo de contrato en el ejemplo de ampliación. Se utilizan metodologías de ampliación similares para los demás tipos de contrato.

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 existente con la nueva política de negocio (esto se realiza capturando la relación entre el objeto de términos y condiciones existente y la nueva política de negocio en los archivos XSD (definición de esquema XML)) o puede crear un objeto de términos y condiciones nuevo 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 archivos XSD para reflejar 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 registrar el nuevo término y condición en la base de datos:

```
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');
```

Registrar el nuevo término y condición en la XSD del contrato

Para que el nuevo término y condición esté disponible en los contratos, deberá crear un nuevo archivo XSD que defina el nuevo término y condición. También deberá actualizar el archivo Package.xsd para incluir el nuevo archivo XSD.

Para crear el nuevo archivo XSD, realice lo siguiente:

1. Vaya al siguiente directorio:
   - 400 *dirusuario_WC*/instances/*nombreInstancia*/xml/trading/xsd
   - AIX Linux Solaris *dir_instal_WC*/xml/trading/xsd
   - Windows *dir_instal_WC*\xml\trading\xsd

   donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.
2. En este directorio, cree un archivo XSD nuevo. A continuación se muestra el XSD que se utilizará para el ejemplo de MyTC. El archivo es CustomizedBuyerContract.xsd:

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.ibm.com/WebSphereCommerce"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wc="http://www.ibm.com/WebSphereCommerce"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <!-- include basic trading agreement xsd -->

  <include schemaLocation="BuyerContract.xsd" />
  <complexType name="MyTCType">
        <complexContent>
            <extension base="wc:TermConditionType"/>
        </complexContent>
    </complexType>
    <element name="MySubTC" substitutionGroup="wc:AbstractCustomizedTC">
        <complexType>
            <complexContent>
                <extension base="wc:MyTCType">
                    <sequence>
                        <element ref="wc:ProductSetPolicyRef"/>
                    </sequence>
                    <attribute name="attr1" type="normalizedString"
                         use="required"/>
                    <attribute name="attr2" type="int" use="required"/>
                </extension>
            </complexContent>
        </complexType>
    </element>
</schema>
```

3. Guarde el nuevo archivo.

A continuación, deberá actualizar el archivo Package.xsd para eliminar BuyerContract.xsd y, en su lugar, incluir CustomizedBuyerContract.xsd, del modo siguiente:

1. Vaya al siguiente directorio:
   - Windows *dir_instal_WC*\xml\trading\xsd
   - AIX Solaris Linux *dir_instal_WC*/xml/trading/xsd
   - 400 *dirusuario_WC*/instances/*nombreInstancia*/xml/trading/xsd

   donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.
2. Abra el archivo Package.xsd en un editor de texto.
3. Localice la sección acerca de BuyerContract.xsd y modifíquelo como se indica a continuación:

```
<!--include schemaLocation="BuyerContract.xsd"/-->
<include schemaLocation="CustomizedBuyerContract.xsd"/>
```

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 beans enterprise nuevos, colocará los beans en el proyecto WebSphereCommerceServerExtensionsData, en lugar de incluirlos en uno de los grupos EJB que contienen los beans de entidad de WebSphere Commerce. Sin embargo, en este caso, dado que todos los beans de entidad nuevos para los términos y condiciones deben heredar del bean TermCondition de WebSphere Commerce, deberá colocar los nuevos beans de términos y condiciones en el proyecto Enablement-RelationshipManagementData. Las secciones siguientes describen cómo crea el nuevo bean enterprise, utilizando las herramientas de WebSphere Studio Application Developer.

Crear un bean enterprise nuevo: Para crear el nuevo bean enterprise CMP para el nuevo término y condición, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Pulse con el botón derecho del ratón en el módulo **Enablement-RelationshipManagementData** y seleccione **Nuevo > Bean Enterprise**. Se abrirá el asistente para la Creación de bean enterprise.
3. En la lista desplegable **Proyecto EJB**, **Enablement-RelationshipManagementData** ya está seleccionado. Pulse **Siguiente**.
4. En la ventana Crear un bean enterprise, realice lo siguiente:
   a. Seleccione **Bean de entidad con campos CMP (Persistencia gestionada por contenedor)**
   b. En el campo **Nombre de bean**, entre un nombre apropiado para el bean. Para este ejemplo, entre `MySubTC`
   c. En el campo **Carpeta fuente**, deje el valor por omisión que está especificado (ejbModule).
   d. En el campo **Paquete por omisión**, entre `com.ibm.commerce.contract.objects`.
   e. Pulse **Siguiente**.
5. En la ventana Detalles del bean enterprise, realice lo siguiente:
   a. En la lista desplegable **Supertipo de bean**, seleccione **TermCondition**.

b. Pulse **Añadir** para añadir un nuevo atributo CMP.
Se abrirá la ventana Crear atributo CMP. En esta ventana, realice lo siguiente:
   1) En el campo **Nombre**, entre un nombre apropiado para el nuevo campo CMP. Para este ejemplo, entre `attr1`.
   2) En el campo **Tipo**, entre el tipo de datos apropiado para el campo. Para este ejemplo, entre `String`.
   3) Marque el recuadro de selección **Acceso con método get y set**.
   4) Marque el recurso de selección **Promocionar métodos get y set a interfaz remota**.
   5) Elimine la marca del recuadro de selección **Hacer que el método get sea de sólo lectura**.
   6) Pulse **Aplicar**.
   7) Cree otro atributo. En el campo **Nombre**, entre `attr2`.
   8) En el campo **Tipo**, entre el tipo de datos apropiado para el campo. Para este ejemplo, entre `Integer`.
   9) Elimine la marca del recuadro de selección **Hacer que el método get sea de sólo lectura**.
   10) Pulse **Aplicar**.
   11) Pulse **Cerrar** para cerrar esta ventana.

6. Pulse **Finalizar**.

Correlacionar los campos del nuevo bean en la tabla TERMCOND: El paso siguiente es correlacionar los campos del nuevo bean con columnas de la tabla TERMCOND. Para crear esta correlación, realice lo siguiente:

1. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda las carpetas siguientes: **Enablement-RelationshipManagementData > ejbModule > META-INF**.
3. Efectúe una doble pulsación en el archivo **Map.mapxmi**.
4. En el panel Beans enterprise, expanda el bean **TermCondition** y, a continuación, expanda el bean **MySubTC** a fin de poder ver sus atributos.
5. En el panel Tablas, expanda la tabla **TERMCOND** para que se puedan ver las columnas.
6. Arrastre el campo **attr1** de MySubTC hasta la columna **STRINGFIELD3** de la tabla TERMCOND.
7. Arrastre el campo **attr2** de MySubTC hasta la columna **INTEGERFIELD3** de la tabla TERMCOND.
8. Guarde los cambios.

Adición de un método ejbCreate nuevo: En este paso añadirá un método ejbCreate nuevo al bean MySubTC, realizando lo siguiente:

1. En la vista de Jerarquía J2EE, efectúe una doble pulsación en la clase **MySubTCBean** para abrirla y ver el código fuente.
2. Cree un nuevo método ejbCreate(Long, Element), añadiendo el código siguiente en la clase:

```
public com.ibm.commerce.contract.objects.TermConditionKey
   ejbCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException,
     javax.ejb.FinderException,
```

```
        javax.naming.NamingException,
        javax.ejb.RemoveException {
             _initLinks();
             super.ejbCreate (argTradingId, argElement);
              this.attr1= null;
              this.attr2 = null;
              return null;
}
```

Guarde los cambios de código.

3. Deberá añadir el nuevo método ejbCreate(Long, Element) a la interfaz inicial. Esto hará que el método esté disponible en el bean de acceso generado. Para añadir el método a la interfaz inicial, realice lo siguiente:
   a. Pulse con el botón derecho del ratón en el método **ejbCreate(Long, Element)** de la vista de Esquema y seleccione **Bean enterprise > Promocionar a interfaz inicial**.

Adición de un método ejbPostCreate: A continuación, cree un método ejbPostCreate(Long, Element) nuevo para que tenga los mismos parámetros que el método ejbCreate(Long, Element), realizando lo siguiente:

1. Efectúe una doble pulsación en la clase **MySubTCBean** para abrirla y ver el código fuente.
2. Cree un método ejbPostCreate(Long, Element) nuevo, añadiendo el código siguiente en la clase:

```
public void ejbPostCreate(java.lang.Long argTradingId,
   org.w3c.dom.Element argElement)
   throws javax.ejb.CreateException,
     javax.ejb.FinderException,
     javax.naming.NamingException,
     javax.ejb.RemoveException
        {
           parseXMLElement(argElement);
        }
```

Guarde los cambios de código.

Adición de un método parseXMLElement: En este paso deberá crear el método parseXMLElement en MySubTCBean, como se indica a continuación:

1. Efectúe una doble pulsación en la clase **MySubTCBean** para abrirla y ver el código fuente.
2. Actualice el método del modo siguiente:

```
public void parseXMLElement(org.w3c.dom.Element argElement) throws
    javax.ejb.CreateException,
    javax.ejb.FinderException,
    javax.naming.NamingException,
    javax.ejb.RemoveException
{
 super.parseXMLElement(argElement);
 if (argElement == null)
  return;
 String nodeName = argElement.getNodeName();
 if (nodeName.equals("TCCopy"))
  return;

 this.attr1 = argElement .getAttribute("attr1").trim();
 this.attr2 = new Integer (argElement.
     getAttribute("attr2").trim());
 // obtener elemento "ProductSetPolicyRef" de "MySubTC"
 Element ePolicyReference = null;
 ePolicyReference = ContractUtil.getElementByTag(
```

```
        argElement, "ProductSetPolicyRef");

        parseElementPolicyReference(ePolicyReference);

    }
```

3. Guarde el trabajo.

Adición de un método createNewVersion: En este paso deberá crear un método createNewVersion nuevo en MySubTCBean, como se indica a continuación:

1. Efectúe una doble pulsación en la clase **MySubTCBean** para abrirla y ver el código fuente.
2. Actualice el método del modo siguiente:

```
public Long createNewVersion(Long argNewTradingId) throws
  javax.ejb.CreateException,
  javax.ejb.FinderException,
  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);
   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;
}
```

3. Guarde el trabajo.

Adición de un método getXMLString: En este paso deberá crear un método getXMLString nuevo en MySubTCBean, como se indica a continuación:

1. Efectúe una doble pulsación en la clase **MySubTCBean** para abrirla y ver el código fuente.
2. Altere el método del modo siguiente:

```
public String getXMLString() throws javax.ejb.CreateException,
   javax.ejb.FinderException,
   javax.naming.NamingException
{
 return getXMLString(false);
}


public String getXMLString(boolean tcdata) throws
 javax.ejb.CreateException,
 javax.ejb.FinderException,
 javax.naming.NamingException
{
 String xmlTC = " <MySubTC %TC_DATA% " +
  "  attr1=\"" + this.attr1 +
  "\" attr2=\"" + this.attr2.toString() + "\">" +
  "%TC_DESC%" +
  "%PARTICIPANT%" +
  "%XML_POLICYREFERENCE%" +
  " </MySubTC>";
```

```
xmlTC = ContractUtil.replace( xmlTC, "%TC_DATA%",
     getXMLStringForTCData(tcdata));
String xmlPolicy = getXMLStringForElementPolicyReference(
    "ProductSet");
xmlTC = ContractUtil.replace( xmlTC, "%XML_POLICYREFERENCE%",
    xmlPolicy);
xmlTC = ContractUtil.replaceAll( xmlTC, "%POLICY_REF_TYPE%",
    "ProductSetPolicyRef");
return xmlTC;
}
```

3. Guarde el trabajo.

Adición de un método markForDelete: En este paso deberá crear un método markForDelete nuevo en MySubTCBean, como se indica a continuación:

1. Efectúe una doble pulsación en la clase **MySubTCBean** para abrirla y ver el código fuente.
2. Altere el método del modo siguiente:

```
public void markForDelete() throws
 javax.ejb.CreateException,
 javax.ejb.FinderException,
 javax.naming.NamingException
{
 // código: elimine entradas de las tablas asociadas que
 // no se pueden suprimir mediante supresión en cascada
}
```

3. Guarde el trabajo.

Actualización de la interfaz remota: Deberá asegurarse de que los métodos siguientes se han añadido a la interfaz remota, como se indica a continuación:

1. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda el proyecto **Enablement-RelationshipManagementData**.
3. Expanda el paquete **com.ibm.commerce.contract.objects**.
4. Efectúe una doble pulsación en el bean **MySubTCBean**.
5. En la vista de Esquema, pulse con el botón derecho del ratón en el método **getXMLString()** y seleccione **Bean enterprise > Promocionar a interfaz remota**
6. En la vista de Esquema, pulse con el botón derecho del ratón en el método **getXMLString(boolean tcdata)** y seleccione **Bean enterprise > Promocionar a interfaz remota**
7. En la vista de Esquema, pulse con el botón derecho del ratón en el método **parseXMLElement(org.w3c.dom.Element argElement)** y seleccione **Bean enterprise > Promocionar a interfaz remota**
8. En la vista de Esquema, pulse con el botón derecho del ratón en el método **createNewVersion(Long argNewTradingId)** y seleccione **Bean enterprise > Promocionar a interfaz remota**
9. En la vista de Esquema, pulse con el botón derecho del ratón en el método **markForDelete()** y seleccione **Bean enterprise > Promocionar a interfaz remota**
10. Guarde los cambios.

Creación de un bean de acceso para MySubTC: Para crear el bean de acceso, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** y, a continuación, pulse con el botón derecho del ratón en **Enablement-RelationshipManagementData** y seleccione **Nuevo > Bean de acceso**.
   Se abrirá la ventana Bean de acceso.
2. Seleccione **Ayudante de copia** y pulse **Siguiente**.
3. Seleccione el bean ***MySubTC*** y pulse **Siguiente**.
4. En la lista desplegable de método Constructor, seleccione **findByPrimaryKey(com.ibm.commerce.contract.objects.***MySubTC***Key)** como método constructor.
5. Seleccione todos los atributos en la sección Ayudantes de atributo.
6. Pulse **Finalizar**.
7. Guarde el trabajo.

Adición de conversores de serie nuevos: Debe añadir conversores de serie nuevos del modo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB > Enablement-RelationshipManagementData > ejbModule > META-INF**.
2. Efectúe una doble pulsación en el archivo **ibm-ejb-access-bean.xmi**.
3. Localice la sección acerca de MySubTC.
4. Para cada elemento copyHelperProperties, añada el atributo siguiente:
   ```
   converterClassName="com.ibm.commerce.base.objects.WCSStringConverter"
   ```
5. Guarde el trabajo.
6. A continuación, debe volver a generar el bean de acceso para MySubTC, del modo siguiente:
   a. Pulse con el botón derecho del ratón en **Enablement-RelationshipManagementData** y seleccione **Beans de acceso > Regenerar beans de acceso**.
   b. Seleccione **MySubTC** y pulse **Finalizar**.

Generación de código desplegado: Deberá generar el código desplegado para el bean MySubTC así como para el bean TermCondition, del modo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Pulse con el botón derecho del ratón en **Enablement-RelationshipManagementData** y seleccione **Generar > Código de despliegue y RMIC**.
3. Seleccione **MySubTC** y pulse **Finalizar**.
4. Pulse con el botón derecho del ratón en **Enablement-RelationshipManagementData** y seleccione **Generar > Código de despliegue y RMIC**.
5. Pulse **Seleccionar todo** y pulse **Finalizar**.

   Nota: Técnicamente, deberá volver a generar el código desplegado para el bean padre (el bean TermCondition) y para todos los beans hermano (todos los demás beans del grupo Enablement-RelationshipManagementData 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 simplificar el ejemplo, las instrucciones anteriores seleccionan todos los beans de este proyecto.

Alteración de métodos en el mandato de tarea validateContract: 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.

Para obtener detalles sobre cómo realizar esta modificación, consulte el apartado "Personalización de los mandatos de tarea existentes" en la página 136.

Creación de un mandato de despliegue nuevo: 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:

1. 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');
   ```

2. 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.

3. 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 que haya creado nuevos términos y condiciones, podrá actualizar WebSphere Commerce Accelerator con el fin de poderlo utilizar para crear contratos nuevos que incluyan esos nuevos términos y condiciones. La actualización de WebSphere Commerce Accelerator con esta finalidad incluye los pasos siguientes:

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:

- 400 AIX Linux Solaris *dir_instal_WC*/samples/contract/Extensions.js
- Windows *dir_instal_WC*\samples\contract\Extensions.js
- Developer *dir_instal_WCDE*\samples\contract\Extensions.js

Para utilizar este archivo de ejemplo, cópielo en el directorio siguiente:

- 400 *dirusuario_WAS*/installedApps/*nombre_célula*/WC_*nombreInstancia*.ear/CommerceAccelerator.war/tools/contract
- AIX Linux Solaris *dir_instal_WAS*/installedApps/*nombre_célula*/WC_*nombreInstancia*.ear/CommerceAccelerator.war/tools/contract
- Windows *dir_instal_WAS*\installedApps*nombre_célula*\WC_*nombreInstancia*.ear\CommerceAccelerator.war\tools\contract
- Developer *dir_espaciotrabajo*\CommerceAccelerator\Web Content\tools\contract

donde *nombreInst* es el nombre de la instancia de WebSphere Commerce y *nombre_célula* es el nombre de la célula de WebSphere Application Server.

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. Esta tarea debe hacerse de forma coherente con las extensiones que ha realizado en los archivos XSD. Esto se muestra en la sección de código siguiente:

```
function submitMyTC(contract) {

   var tcModel = get("ContractMyTCModel");

   if (tcModel != null) {

      var myTC = new Object();
 myTC.attr1 = tcModel.attr1;
 myTC.attr2 = tcModel.attr2;

 myTC.ProductSetPolicyRef = new Object();
 myTC.ProductSetPolicyRef.policyName = tcModel.policyList[
      tcModel.selectedPolicyIndex].policyName;
 myTC.ProductSetPolicyRef.StoreRef = new Object();
 myTC.ProductSetPolicyRef.StoreRef.name = tcModel.policyList[
      tcModel.selectedPolicyIndex].storeIdentity;
 myTC.ProductSetPolicyRef.StoreRef.Owner = new Object();
 myTC.ProductSetPolicyRef.StoreRef.Owner = 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";
 }

 contract.MySubTC = 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:

- 400 AIX Linux Solaris *dir_instal_WC*/samples/contract/ContractMyTC.jsp
- Windows *dir_instal_WC*\samples\contract\ContractMyTC.jsp
- Developer *dir_instal_WCDE*\samples\contract\ContractMyTC.jsp

Para utilizar este archivo de ejemplo, cópielo en el directorio siguiente:

- 400 *dirusuario_WAS*/installedApps/*nombre_célula*/WC_*nombreInstancia*.ear/CommerceAccelerator.war/javascript/tools/contract

- AIX Linux Solaris *dir_instal_WAS*/installedApps/*nombre_célula*/WC_*nombreInstancia*.ear/CommerceAccelerator.war/ javascript/tools/contract
- Windows *dir_instal_WAS*\installedApps*nombre_célula*\WC_*nombreInstancia*.ear\CommerceAccelerator.war\ javascript\tools\contract
- Developer *dir_espaciotrabajo*\CommerceAccelerator\Web Content\tools\contract

donde *nombreInst* es el nombre de la instancia de WebSphere Commerce y *nombre_célula* es el nombre de la célula de WebSphere Application Server.

La sección de código siguiente muestra un ejemplo de sección HTML de una plantilla JSP que se puede utilizar para MyTC.

```
<!--
////////////////////////////////////
// HTML SECTION
////////////////////////////////////
-->

<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:

- 400 *dirusuario_WAS*/installedApps/*nombre_célula*/WC_*nombreInstancia*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*entorno_nacional*.properties
- AIX Linux Solaris *dir_instal_WAS*/installedApps/*nombre_célula*/WC_*nombreInstancia*.ear/properties/com/ibm/commerce/tools/contract/properties/ContractRB_*entorno_nacional*.properties
- Windows *dir_instal_WAS*\installedApps*nombre_célula*\WC_*nombreInstancia*.ear\properties\com\ibm\commerce\tools\contract\properties\ContractRB_*entorno_nacional*.properties
- Developer *dir_espaciotrabajo*\WebSphereCommerceServer\properties\com\ibm\commerce\tools\contract\properties\ContractRB_*entorno_nacional*.properties

donde *nombreInst* es el nombre de la instancia de WebSphere Commerce y *nombre_célula* es el nombre de la célula de WebSphere Application Server.

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.

- 400 *dir_instal_WC*/xml/tools/contract/ContractNotebook.xml
- AIX Linux Solaris *dir_instal_WC*/xml/tools/contract/ContractNotebook.xml
- Windows *dir_instal_WC*\xml\tools\contract\ContractNotebook.xml
- Developer *dir_instal_WCDE*\Commerce\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:

```
<MySubTC attr1="abc" attr2="123">
       <ProductSetPolicyRef policyName = "Product Set 1">
           <StoreRef name = "StoreGroup1">
               <Owner>
                   <OrganizationRef distinguishName = "o=Root Organization"/>
               </Owner>
           </StoreRef>
       </ProductSetPolicyRef>
   </MySubTC>
```

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 paso siguiente para integrar totalmente la ampliación del modelo de contrato en el proceso de negocio consiste en 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 cómo crear contratos, consulte la Ayuda en línea a la producción y al desarrollo 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 los archivos XSD para reflejar el nuevo RebateTC.
   c. Crear un nuevo bean enterprise para RebateTC.
   d. Actualizar WebSphere Commerce Accelerator para reflejar 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 información detallada sobre cómo crear este bean, consulte el apartado “Creación de un bean enterprise CMP nuevo” en la página 55.

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. Ésta es la interfaz `RebatePolicyCmd` que `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 pertinentes 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 los archivos XSD para reflejar 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

Registrar el término y condición de rebaja en la definición de tipo de documento del contrato: Para que el término y condición de rebaja esté disponible en los contratos, deberá crear un nuevo archivo XSD que defina el nuevo término y condición. También deberá actualizar el archivo Package.xsd para incluir el nuevo archivo XSD.

Para crear el nuevo archivo XSD, realice lo siguiente:

1. Vaya al siguiente directorio:
   - Windows *dir_instal_WC*\xml\trading\xsd
   - AIX Solaris Linux *dir_instal_WC*/xml/trading/xsd
   - 400 *dirusuario_WC*/instances/*nombreInstancia*/xml/trading/xsd

   donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.
2. En este directorio, cree un archivo XSD nuevo. A continuación se muestra el XSD que se utilizará para el ejemplo de RebateTC. El archivo es RebateCustomizedBuyerContract.xsd:

```
<?xml version="1.0"?>
<schema targetNamespace="http://www.ibm.com/WebSphereCommerce"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:wc="http://www.ibm.com/WebSphereCommerce"
        elementFormDefault="qualified"
        attributeFormDefault="unqualified">

  <!-- include basic trading agreement xsd -->

  <include schemaLocation="BuyerContract.xsd" />
  <complexType name="RebateTCType">
            <complexContent>
            <extension base="wc:TermConditionType"/>
            </complexContent>
        </complexType>
    <element name="RebateTC" substitutionGroup="wc:AbstractCustomizedTC">
        <complexType>
            <complexContent>
                <extension base="wc:RebateTCType">
                    <sequence>
                        <element ref="wc:RebatePolicyRef"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
```

```
        <element name="RebatePolicyRef" type="wc:BusinessPolicyRef" />
    </schema>
```

3. Guarde el nuevo archivo.

A continuación, debe actualizar Package.xsd para eliminar BuyerContract.xsd y, en su lugar, incluir RebateCustomizedBuyerContract.xsd, del modo siguiente:

1. Vaya al siguiente directorio:
   - Windows *dir_instal_WC*\xml\trading\xsd
   - AIX Solaris Linux *dir_instal_WC*/xml/trading/xsd
   - 400 *dirusuario_WC*/instances/*nombreInstancia*/xml/trading/xsd

   donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.
2. Abra el archivo Package.xsd en un editor de texto.
3. Localice la sección acerca de BuyerContract.xsd y modifíquelo como se indica a continuación:

```
<!--include schemaLocation="BuyerContract.xsd"/-->
<include schemaLocation="RebateCustomizedBuyerContract.xsd"/>
```

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 153.

Tabla 8.

Atributo	Valor
Proyecto EJB	Enablement-RelationshipManagementData
Tipo de bean	Bean de entidad con campos persistencia gestionada por contenedor
Nombre del bean	RebateTC
Paquete	com.ibm.commerce.contract.objects
Supertipo de bean	TermCondition
Clase de bean	RebateTCBean

En el nuevo bean, cree tres campos CMP que se utilizarán para los valores siguientes:

- ID de código de rebaja
- importe
- moneda

Actualización de WebSphere Commerce Accelerator para incluir RebateTC: Una vez que haya creado nuevos términos y condiciones, podrá actualizar WebSphere

Commerce Accelerator con el fin de poderlo utilizar para crear contratos nuevos 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 160.

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 contratos nuevos se describe en la Ayuda en línea a la producción y al desarrollo 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. Entorno de desarrollo

Este capítulo presenta las principales herramientas de desarrollo utilizadas para personalizar una aplicación de WebSphere Commerce.

Entorno de desarrollo típico

Business Para crear código personalizado para WebSphere Commerce Business Edition, se recomienda el paquete de desarrollo WebSphere Commerce Studio, Business Developer Edition. Professional Para crear código personalizado para WebSphere Commerce Professional Edition, se recomienda el paquete de desarrollo WebSphere Commerce Studio, Professional Developer Edition. Express Para crear código personalizado para WebSphere Commerce - Express, se recomienda el paquete de desarrollo WebSphere Commerce - Express Developer Edition. Todos estos paquetes incluyen las herramientas necesarias para crear código personalizado y llevar a cabo tareas de desarrollo de la Web. En general, en este manual se hace referencia colectivamente a estos productos denominándolos el Entorno de desarrollo de WebSphere Commerce.

Existen cuatro componentes principales en el Entorno de desarrollo de WebSphere Commerce:

1. El espacio de trabajo de WebSphere Commerce que se utiliza en WebSphere Studio Application Developer
2. La base de datos de desarrollo
3. Elementos de sistema de archivos
4. Plug-ins de WebSphere Commerce en WebSphere Studio Application Developer

Con este entorno de desarrollo, puede crear código personalizado y probarlo dentro del contexto del entorno de prueba de WebSphere.

Para crear tiendas en el entorno de desarrollo, simplemente inicie en WebSphere Studio Application Developer la Consola de administración que se ejecuta en el servidor de prueba WebSphereCommerceServer definido localmente y utilice dicha herramienta para publicar una tienda basada en una de las tiendas de ejemplo. Alternativamente, puede crear su propia tienda.

Además del espacio de trabajo de WebSphere Commerce que se proporciona, el Entorno de desarrollo de WebSphere Commerce proporciona herramientas y plug-ins adicionales. Se proporcionan los plug-ins siguientes:

- Un plug-in del Gestor de configuración que ayuda con la gestión de las instancias de WebSphere Commerce (y WebSphere Commerce Payments).
- Un plug-in de ayuda en línea de WebSphere Commerce le permite acceder a la ayuda en línea de WebSphere Commerce desde dentro de WebSphere Studio Application Developer. Adicionalmente, con este plug-in, se puede iniciar la ayuda en línea sensible al contexto de WebSphere Commerce pulsando F1 cuando se ejecutan las herramientas de WebSphere Commerce (por ejemplo, la Consola de administración).
- Un plug-in de información de consulta de la API de WebSphere Commerce que le permite acceder a la información de consulta de la API de WebSphere Commerce desde dentro de WebSphere Studio Application Developer.

También se proporciona una herramienta de conversión de bean enterprise de WebSphere Commerce. Con esta herramienta, puede desarrollar beans enterprise utilizando una base de datos de desarrollo diferente de la base de datos de producción de destino. Por ejemplo, esta herramienta le permite utilizar un base de datos DB2 local para el desarrollo, pero la base de datos de producción puede estar en la plataforma iSeries o incluso ser una base de datos Oracle.

WebSphere Studio Application Developer

El paquete de el Entorno de desarrollo de WebSphere Commerce incluye WebSphere Studio Application Developer que es el entorno de desarrollo esencial de IBM. Le ayuda a optimizar y simplificar el desarrollo de servicios Web y Java2 Enterprise Edition (J2EE) ofreciendo las mejores prácticas, las plantillas, la generación de código y el entorno de desarrollo más completo de su clase. Este sofisticado entorno de desarrollo integrado (IDE) incluye soporte integrado para componentes Java, beans enterprise, servlets, archivos JSP, HTML, XML y servicios Web, todos ellos en un solo entorno de desarrollo.

Entre muchas otras características importantes, también incluye herramientas de prueba local que le permiten generar rápidamente clientes de prueba. También incluye un entorno de prueba de WebSphere Application Server completo que le permite probar el código de un extremo a otro en un entorno nacional.

Entorno de desarrollo para iSeries

400 En resumen, no es necesario un entorno de desarrollo especial para crear código personalizado para iSeries. La estación de trabajo de desarrollo se configura siguiendo el mismo procedimiento que se describe en la publicación Business Professional *WebSphere Commerce Studio, Guía de instalación* o Express *WebSphere Commerce - Express Developer Edition, Guía de instalación*. La base de datos local de desarrollo utilizada debe ser DB2. Si se utiliza esta configuración, se puede crear y probar código personalizado utilizando la base de datos DB2 local y el servidor de prueba WebSphereCommerceServer local.

Después de que las pruebas determinen que el código personalizado funciona de forma satisfactoria en el contexto del servidor de prueba, deberá desplegar dicho código en un WebSphere Commerce Server de destino que se ejecute en la plataforma iSeries. Con el fin de poder tratar las diferencias entre la base de datos de la plataforma Windows y la de la plataforma iSeries, se proporciona una herramienta de conversión de bean enterprise de WebSphere Commerce. En el apartado “Visión general de la herramienta de conversión de bean enterprise de WebSphere Commerce” en la página 177, se proporciona más información acerca de esta herramienta.

Utilización de una base de datos DB2 local para el desarrollo cuando el entorno de producción utiliza una base de datos Oracle

Es posible que los desarrolladores utilicen una base de datos DB2 local en las máquinas de desarrollo aunque la base de datos para el entorno de producción vaya a ser una base de datos Oracle. En este caso, se utiliza la herramienta de conversión de bean enterprise de WebSphere Commerce para convertir los metadatos de los beans del formato de DB2 al formato de Oracle. En el apartado “Visión general de la herramienta de conversión de bean enterprise de WebSphere Commerce” en la página 177, se proporciona más información acerca de esta herramienta.

Visión general de la herramienta de conversión de bean enterprise de WebSphere Commerce

En general, existen dos casos en los que se utiliza la herramienta de conversión de bean enterprise:

- Si el entorno de producción de destino se ejecuta en la plataforma iSeries.
- Si la base de datos de producción de destino es una base de datos Oracle aunque las máquinas de desarrollo utilicen bases de datos DB2 locales.

Esta herramienta específica de WebSphere Commerce le permite desarrollar en un tipo de base de datos y, a continuación, desplegar en otro tipo de base de datos. Utilizando esta herramienta, los metadatos para los beans enterprise se convierten al formato apropiado y en la información para la base de datos de destino y el código desplegado también se genera utilizando estos nuevos metadatos. Para obtener detalles paso a paso sobre cómo utilizar esta herramienta, consulte el apartado “Creación de un archivo JAR EJB con conversión” en la página 181.

Opciones de pago en el entorno de desarrollo

En las versiones anteriores de el Entorno de desarrollo de WebSphere Commerce, se proporcionaba un método de pago de prueba para permitir a los desarrolladores realizar una compra dentro de la tienda del entorno de prueba, sin llamar a un proveedor de pago remoto. Ahora, con el Entorno de desarrollo de WebSphere Commerce Versión 5.5, el componente WebSphere Commerce Payments puede ejecutarse dentro del entorno de prueba.

Esto significa que ahora tiene la opción de utilizar la instancia de WebSphere Commerce Payments local o puede configurar la instancia de desarrollo de WebSphere Commerce para utilizar una instancia de WebSphere Commerce Payments remota.

Por omisión, la instancia de WebSphere Commerce Payments local se crea al instalar el Entorno de desarrollo de WebSphere Commerce. Adicionalmente, si esta instancia está en ejecución en el momento en que publica una tienda de ejemplo, la tienda se configura automáticamente para utilizar dicha instancia de WebSphere Commerce Payments local.

En la publicación *el Entorno de desarrollo de WebSphere Commerce, Guía de instalación*, se proporciona información sobre cómo configurar las opciones de pago.

Capítulo 9. Información detallada sobre el despliegue

Después de haber creado código personalizado en el Entorno de desarrollo de WebSphere Commerce y de haberlo probado en el entorno de prueba de WebSphere, deberá desplegarlo en un WebSphere Commerce Server de destino que se ejecute fuera del entorno de prueba de WebSphere. Este WebSphere Commerce Server de destino puede ejecutarse localmente en la máquina de desarrollo o puede estar en otra máquina (utilizando el mismo sistema operativo u otro diferente).

Este capítulo describe los pasos necesarios para el despliegue de código personalizado en un WebSphere Commerce Server de destino que se ejecuta fuera del entorno de prueba de WebSphere.

Este capítulo está dividido en secciones que describen cómo desplegar los diversos tipos de código que la aplicación personalizada puede contener. Se describen las tareas siguientes:

- Despliegue de beans enterprise
- Despliegue de mandatos y beans de datos
- Despliegue de elementos de tienda
- Actualización de la base de datos de destino

Cada una de las tareas anteriores se describe más detalladamente en las secciones siguientes.

Requisitos de permiso de usuario para los pasos de despliegue

AIX Linux Solaris Deberá realizar todos los pasos de despliegue (excepto las actualizaciones de control de acceso) en el WebSphere Commerce Server de destino utilizando el ID de usuario no root que ha creado en la preparación del proceso de instalación de WebSphere Commerce. 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.

Para obtener información sobre los permisos de usuario que son necesarios para realizar actualizaciones de control de acceso, consulte el tema “Carga de cambios XML en la base de datos” en la publicación *WebSphere Commerce, Guía de seguridad*.

Despliegue incremental

El tipo de despliegue descrito en este capítulo es un despliegue *incremental*. En un despliegue incremental, ya debe tener instalada una aplicación de empresa WebSphere Commerce en el WebSphere Commerce Server de destino. Entonces, en el proceso de despliegue, sólo desplegará esos elementos (mandatos, beans de datos, beans enterprise, etc) en la aplicación de empresa existente.

Para crear la aplicación de empresa inicial en el WebSphere Commerce Server de destino, deberá instalar WebSphere Commerce y, a continuación, utilizar el Gestor de configuración para crear la instancia de WebSphere Commerce (y, por lo tanto, crear la aplicación de empresa WebSphere Commerce).

Despliegue de beans enterprise

Esta sección describe cómo desplegar beans enterprise. Estos beans pueden ser beans enterprise nuevos que ha creado para la aplicación de comercio electrónico o pueden ser beans de entidad de WebSphere Commerce que ha modificado. En cualquiera de los casos, los pasos de despliegue son básicamente los mismos.

Uno de los factores clave para comprender el proceso de despliegue de la aplicación de WebSphere Commerce consiste en conocer el esquema de empaquetado que se utiliza para el código personalizado de WebSphere Commerce. En particular, no necesita crear un proyecto EJB nuevo en el espacio de trabajo de WebSphere Commerce. Los beans enterprise nuevos se colocan en el proyecto WebSphereCommerceServerExtensionsData y el código personalizado para los beans de entidad de WebSphere Commerce modificados permanecen en el proyecto EJB original de WebSphere Commerce.

En este proceso de despliegue hay dos pasos principales:

- Creación del archivo JAR EJB
- Actualización del archivo JAR EJB en el WebSphere Commerce Server de destino

Creación del archivo JAR EJB

En función del escenario de despliegue, existen dos planteamientos diferentes para crear el archivo JAR EJB, como se indica a continuación:

- Si está creando un archivo JAR EJB que se está desplegando en un WebSphere Commerce Server de destino que utiliza el mismo tipo de base de datos que el entorno de despliegue, siga las instrucciones contenidas en el apartado "Creación de un archivo JAR EJB sin conversión" en la página 180.
- Si está creando un archivo JAR EJB que se está desplegando en un WebSphere Commerce Server de destino que utiliza un tipo diferente de base de datos que el entorno de despliegue, siga las instrucciones contenidas en el apartado "Creación de un archivo JAR EJB con conversión" en la página 181.

Creación de un archivo JAR EJB sin conversión

Para crear el archivo JAR EJB, realice lo siguiente:

1. Abra el Entorno de desarrollo de WebSphere Commerce (**Inicio > Programas > IBM el Entorno de desarrollo de WebSphere Commerce > Entorno de desarrollo de WebSphere Commerce**) y conmute a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda el proyecto EJB que contiene el bean o los beans que está desplegando, como se indica a continuación:
   - Si está creando beans enterprise nuevos, expanda el proyecto EJB **WebSphereCommerceServerExtensionsData**.
   - Si ha modificado beans de entidad de WebSphere Commerce, expanda el proyecto que contiene el bean modificado. Por ejemplo, si ha modificado el bean de usuario, expanda el proyecto EJB **Member-MemberManagementData**.
3. Efectúe una doble pulsación en **Descriptor de despliegue EJB**.
4. Con la pestaña Visión general seleccionada, desplácese a la parte inferior del panel para localizar la sección **Enlaces de WebSphere**.
5. En el campo **Nombre JNDI de origen de datos**, entre el nombre JNDI de origen de datos del WebSphere Commerce Server de destino. A continuación se muestra un valor de ejemplo:

DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
donde el WebSphere Commerce Server de destino está utilizando un base de datos DB2 y el nombre de instancia de WebSphere Commerce es “demo”

Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
donde el WebSphere Commerce Server de destino está utilizando una base de datos Oracle y el nombre de instancia de WebSphere Commerce es “demo”.

El valor para el nombre JNDI de origen de datos se crea añadiendo “`jdbc/`” al nombre de origen de datos del WebSphere Commerce Server de destino. Puede verificar el nombre de origen de datos abriendo el archivo *nombreInstancia*.xml del WebSphere Commerce Server de destino y buscando `DatasourceName=` en el archivo.

6. Guarde los cambios del descriptor de despliegue (Control+S).
7. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, pulse con el botón derecho del ratón en el proyecto EJB (WebSphereCommerceServerExtensionsData o el proyecto que contiene el bean de entidad de WebSphere Commerce modificado) y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
8. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR EJB** y pulse **Siguiente**.
   b. Business Professional El valor para **¿Qué recursos desea exportar?** está previamente lleno con el nombre del proyecto EJB.
      Express El nombre del proyecto EJB ya está rellenado.
      Deje el valor que aparece.
   c. Business Professional En el campo **¿Dónde desea exportar los recursos?** entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      Express Para el destino, entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      Por ejemplo, entre `C:\ExportTemp\`*`NombreArchivoJar`*`.jar` donde *NombreArchivoJar* es el nombre del archivo JAR. Si ha creado beans enterprise nuevos, deberá entrar *`suDir`*`\WebSphereCommerceServerExtensionsData.jar`. Si ha modificado un bean de entidad público de WebSphere Commerce existente, deberá utilizar el nombre de archivo JAR predefinido para este grupo EJB. Por ejemplo, si la modificación ha sido en el módulo EJB Member-MemberManagementData, entre *`suDir`*`\Member-MemberManagementData.jar`.
   d. Asegúrese de que **Exportar archivos fuente** no esté seleccionado.
   e. Pulse **Finalizar**.
9. Después de haber creado el archivo JAR, abra el descriptor de despliegue EJB y vuelva a restaurar las modificaciones que se han realizado en el paso 5 al valor que sea necesario para el servidor de prueba local. Guarde los cambios.

Creación de un archivo JAR EJB con conversión

Para convertir los metadatos y crear el archivo JAR EJB, realice lo siguiente:

1. Abra el Entorno de desarrollo de WebSphere Commerce (**Inicio > Programas > IBM el Entorno de desarrollo de WebSphere Commerce > Entorno de desarrollo de WebSphere Commerce**) y conmute a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda el proyecto EJB que contiene el bean o los beans que está desplegando, como se indica a continuación:

- Si ha creado beans enterprise nuevos, expanda el proyecto **WebSphereCommerceServerExtensionsData**.
- Si ha modificado beans de entidad de WebSphere Commerce, expanda el proyecto que contiene el bean modificado. Por ejemplo, si ha modificado el bean de usuario, expanda el proyecto EJB **Member-MemberManagementData**.

3. Efectúe una doble pulsación en **Descriptor de despliegue EJB**.
4. Con la pestaña Visión general seleccionada, desplácese a la parte inferior del panel para localizar la sección **Enlaces de WebSphere**.
5. En el campo **Nombre JNDI de origen de datos**, entre el nombre JNDI de origen de datos del WebSphere Commerce Server de destino. A continuación se muestra un valor de ejemplo:

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   donde el WebSphere Commerce Server de destino está utilizando una base de datos DB2 y el nombre de instancia de WebSphere Commerce es "demo"

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   donde el WebSphere Commerce Server de destino está utilizando una base de datos Oracle y el nombre de instancia de WebSphere Commerce es "demo".

El valor para el nombre JNDI de origen de datos se crea añadiendo "`jdbc/`" al nombre de origen de datos del WebSphere Commerce Server de destino. Puede verificar el nombre de origen de datos abriendo el archivo *nombreInstancia*.xml del WebSphere Commerce Server de destino y buscando `DatasourceName=` en el archivo.

6. Guarde los cambios del descriptor de despliegue (Control+S).
7. Cierre WebSphere Studio Application Developer.
8. En un indicador de mandatos, vaya al siguiente directorio:

   dir_instal_WCStudio`\Commerce\bin`
9. Entre el siguiente mandato:

   `ejbDeploy.bat` *nombreProy nombreJarSalida archivoCorrelación dir_espaciotrabajo dirEclipse archivoEjbDeployXml dirCommerce_WCStudio*

   donde:

   - *nombreProy* es el nombre del proyecto EJB que se debe convertir
   - *nombreJarSalida* es el nombre totalmente calificado del archivo JAR de salida. Tenga en cuenta que deberá estar utilizando el nombre del archivo JAR que ya existe en la aplicación de empresa de WebSphere Commerce. A continuación se muestra un ejemplo:

     `C:\ExportTemp\WebSphereCommerceServerExtensionsData.jar`
   - *archivoCorrelación* es el nombre del archivo de correlación. Los archivos siguientes son ejemplos de ello:

     dir_instal_WCDE`\Commerce\properties\com\ibm\commerce\metadata\conversion\oracle.mapping`
     dir_instal_WCDE`\Commerce\properties\com\ibm\commerce\metadata\conversion\as400.mapping`
   - *dir_espaciotrabajo* es el directorio para el espacio de trabajo de desarrollo actual.
   - *dirEclipse* es la vía de acceso al directorio eclipse. Por omisión, es

     dir_instal_WCDE`\Studio5`
   - *archivoEjbDeployXml* es el archivo ejbDeploy.xml totalmente calificado. Por omisión, es

dir_instal_WCDE\Commerce\xml\ejbDeploy.xml

Nota: Cuando ejecute este mandato, es posible que vea errores que indican que no se han podido expandir algunos archivos. Esto no es ningún problema.

- *dirCommerce_WCStudio* es la vía de acceso al directorio Commerce que se crea al instalar el Entorno de desarrollo de WebSphere Commerce. Por omisión, es

  dir_instal_WCDE\Commerce

  o más específicamente

  C:\WebSphere\CommerceDev55

A continuación, se muestra un ejemplo de uso de este mandato en el que se han especificado todos los valores:

```
ejbDeploy.bat WebSphereCommerceServerExtensionsData
WebSphereCommerceServerExtensionsData.jar
C:\WebSphere\CommerceStudio55\Commerce\properties\com\ibm\
   commerce\metadata\conversion\oracle.mapping
C:\WebSphere\workspace_db2 C:\WebSphere\Studio5
C:\WebSphere\CommerceStudio55\Commerce\xml\ejbDeploy.xml
C:\WebSphere\CommerceStudio55\Commerce
```

Tenga en cuenta que los saltos de línea sólo se muestran a efectos de presentación.

10. Abra el Entorno de desarrollo de WebSphere Commerce y abra el descriptor de despliegue EJB y vuelva a restaurar las modificaciones que se han realizado en el paso 5 al valor que sea necesario para el servidor de prueba local. Guarde los cambios.
11. Si está reuniendo todos los elementos que se deben desplegar en un directorio individual (por ejemplo C:\ExportTemp), copie el archivo JAR EJB recién creado en dicho directorio.

Actualización del archivo JAR EJB en el WebSphere Commerce Server de destino

El siguiente paso es copiar el archivo JAR EJB recién creado en el lugar apropiado del WebSphere Commerce Server de destino.

Para actualizar el archivo JAR EJB en el WebSphere Commerce Server de destino, realice lo siguiente:

1. Detenga la instancia de WebSphere Commerce que se está ejecutando en WebSphere Application Server. Consulte la publicación *WebSphere Commerce, Guía de instalación* para su plataforma y base de datos si desea obtener detalles sobre cómo detener esta instancia.
2. Localice el archivo JAR EJB original en la instancia de WebSphere Commerce. Por ejemplo, localice el archivo siguiente:

   - 400 *dirusuario_WAS*/installedApps/*nombre_nodo_WAS*/WC_*nombre_instancia*.ear/*nombreArchivoJar*.jar
   - AIX Linux Solaris *dir_instal_WAS*/installedApps/*nombreCélula*/WC_*nombre_instancia*.ear/*nombreArchivoJar*.jar
   - Windows *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombre_instancia*.ear*nombreArchivoJar*.jar

   donde

- *nombre_instancia* es el nombre de la instancia de WebSphere Commerce.
- 400 *nombre_nodo_WAS* representa el sistema iSeries donde está instalado el producto WebSphere Application Server.
- *nombreArchivoJar* es el nombre del archivo JAR que contiene el código personalizado

3. Realice una copia del archivo JAR EJB original.
4. Copie el archivo JAR EJB nuevo de la máquina de desarrollo en la ubicación del paso 2.
5. Si ha modificado los descriptores de despliegue EJB, realice lo siguiente:
   a. Localice el depósito de despliegue (directorio META-INF) para esta célula de WebSphere Application Server. Éste toma normalmente el formato siguiente:
      - 400 *dirusuario_WAS*/config/cells/*nombre_nodo_WAS*/applications/WC_*nombre_instancia*.ear/deployments/WC_*nombre_instancia*/*nombreMóduloEJB*.jar/META-INF
      - AIX Linux Solaris *dir_instal_WAS*/config/cells/*nombreCélula*/applications/WC_*nombre_instancia*.ear/deployments/WC_*nombre_instancia*/*nombreMóduloEJB*.jar/META-INF
      - Windows *dir_instal_WAS*\config\cells*nombreCélula*\applications\WC_*nombre_instancia*.ear\deployments\WC_*nombre_instancia**nombreMóduloEJB*.jar\META-INF

      donde
      - *nombre_instancia* es el nombre de la instancia de WebSphere Commerce.
      - 400 *nombre_nodo_WAS* representa el sistema iSeries en el que está instalado el producto WebSphere Application Server.
      - *nombreMóduloEJB* es el nombre del módulo EJB que se ha modificado

      A continuación, se proporciona ejemplos específicos de plataforma del directorio META-INF:
      - 400 *dirusuario_WAS*/config/cells/myNode/applications/WC_demo.ear/deployments/WC_demo/Member-MemberManagementData.jar/META-INF
      - AIX Linux Solaris *dir_instal_WAS*/config/cells/myCell/applications/WC_demo.ear/deployments/WC_demo/Member-MemberManagementData.jar/META-INF
      - Windows *dir_instal_WAS*\config\cells\myCell\applications\WC_demo.ear\deployments\WC_demo\Member-MemberManagementData.jar\META-INF

      donde
      - myCell es el nombre de la célula de WebSphere Application Server
      - demo es el nombre de la instancia de WebSphere Commerce
      - Member-MemberManagementData es el nombre del módulo EJB que se ha modificado
      - 400 myNode es el nombre del nodo de WebSphere Application Server

   b. Haga una copia de seguridad de todos los archivos del directorio anterior.
   c. Utilice una herramienta para abrir el archivo *nombreArchivoJar*.jar y ver su contenido.

d. Extraiga el contenido del directorio META-INF del archivo *nombreArchivoJar*.jar en el directorio del paso 5a.

6. Reinicie la instancia de WebSphere Commerce tal como se describe en la publicación *WebSphere Commerce, Guía de instalación* para la plataforma y la base de datos.

Despliegue de mandatos y beans de datos

Cuando realice cualquiera de las tareas siguientes, el código personalizado deberá estar empaquetado en el proyecto WebSphereCommerceServerExtensionsLogic:

- Crear mandatos nuevos
- Crear beans de datos nuevos
- Modificar mandatos de WebSphere Commerce existentes
- Modificar beans de datos de WebSphere Commerce existentes

No se le permite realizar modificaciones directamente en una clase existente (o en el proyecto que contiene la clase existente) para mandatos o beans de datos.

El despliegue de mandatos y beans de datos personalizados implica los pasos siguientes:

- Creación del archivo JAR
- Actualización del archivo JAR en el WebSphere Commerce Server de destino

Los pasos anteriores se describen más detalladamente en las secciones siguientes.

Asegúrese de consultar el apartado "Actualización de la base de datos de destino" en la página 188 si el código personalizado necesita actualizaciones en el registro de mandatos, políticas de control de acceso nuevas u otras actualizaciones de base de datos.

Creación del archivo JAR

Para crear el archivo JAR, haga lo siguiente:

1. Abra el Entorno de desarrollo de WebSphere Commerce (**Inicio > Programas > IBM el Entorno de desarrollo de WebSphere Commerce > Entorno de desarrollo de WebSphere Commerce**) y conmute a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Exportar**. Se abrirá el asistente para Exportar.
3. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR** y pulse **Siguiente**.
   b. El panel izquierdo bajo **¿Seleccionar los recursos para exportar?** está previamente lleno con el nombre del proyecto. Deje este valor como está.
   c. En el panel derecho bajo **¿Seleccionar los recursos para exportar?**, asegúrese de que sólo estén seleccionados los recursos siguientes:
      - .classpath
      - .project
      - .serverPreference
   d. Asegúrese de que **Exportar archivos de clases y recursos generados** esté seleccionado.
   e. *No* seleccione **Exportar archivos fuente y recursos Java**.

f. En el campo **Seleccionar el destino de exportación**, entre el nombre de archivo JAR totalmente calificado que se debe utilizar. Por ejemplo, entre `C:\ExportTemp\WebSphereCommerceServerExtensionsLogic.jar`.
Tenga en cuenta que el nombre de archivo JAR debe ser WebSphereCommerceServerExtensionsLogic.jar.

g. Pulse **Finalizar**.

Una vez que se ha creado satisfactoriamente el archivo JAR, el siguiente paso consiste en transferir el archivo a la ubicación apropiada del WebSphere Commerce Server de destino.

Actualización del archivo JAR en el WebSphere Commerce Server de destino

El siguiente paso es copiar el archivo JAR recién creado en el lugar apropiado del WebSphere Commerce Server de destino.

Para actualizar el archivo JAR, realice lo siguiente:

1. Detenga la instancia de WebSphere Commerce que se está ejecutando en WebSphere Application Server. Consulte la publicación *WebSphere Commerce, Guía de instalación* para su plataforma y base de datos si desea obtener detalles sobre cómo detener esta instancia.
2. Localice el archivo JAR EJB original en la instancia de WebSphere Commerce. Por ejemplo, localice el archivo siguiente:
   - 400 *dirusuario_WAS*/installedApps/*nombre_nodo_WAS*/WC_*nombre_instancia*.ear/WebSphereCommerceServerExtensionsLogic.jar
   - AIX Linux Solaris *dir_instal_WAS*/installedApps/*nombreCélula*/WC_*nombre_instancia*.ear/WebSphereCommerceServerExtensionsLogic.jar
   - Windows *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombre_instancia*.ear\WebSphereCommerceServerExtensionsLogic.jar

   donde
   - *nombre_instancia* es el nombre de la instancia de WebSphere Commerce.
   - 400 *nombre_nodo_WAS* representa el sistema iSeries en el que está instalado el producto WebSphere Application Server.
3. Realice una copia del archivo JAR original en una ubicación de copia de seguridad.
4. Copie el archivo JAR de la máquina de desarrollo en la ubicación del paso 2.
5. Reinicie la instancia de WebSphere Commerce tal como se describe en la publicación *WebSphere Commerce, Guía de instalación* para la plataforma y la base de datos.

Despliegue de elementos de tienda

Los elementos de tienda incluyen elementos como los siguientes:

- Plantillas JSP
- Archivos HTML
- Archivos de imagen
- Archivos XML
- Archivos de propiedades y paquetes de recursos

Estos elementos deben desplegarse desde el entorno de desarrollo en el WebSphere Commerce Server de destino. Esta tarea incluye los pasos siguientes:

- Exportación de elementos de tienda de WebSphere Studio Application Developer
- Transferencia de elementos al WebSphere Commerce Server de destino

Los pasos anteriores se describen más detalladamente en las secciones siguientes.

Exportación de elementos de tienda

Para exportar los elementos de tienda del entorno de desarrollo, realice lo siguiente:

1. Abra el Entorno de desarrollo de WebSphere Commerce (**Inicio > Programas > IBM el Entorno de desarrollo de WebSphere Commerce > Entorno de desarrollo de WebSphere Commerce**) y conmute a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda la carpeta **Stores**.
3. Pulse con el botón derecho del ratón en la carpeta **Web Content** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Sistema de archivos** y pulse **Siguiente**.
   b. Seleccione todos los recursos que desea desplegar. Es decir, seleccione la totalidad de plantillas JSP, archivos HTML, imágenes, archivos de propiedades y otros elementos de tienda que es necesario desplegar.
   c. Seleccione **Crear estructura de directorios para archivos seleccionados**.
   d. En el campo **Directorio**, entre un directorio temporal en el que se se colocarán estos recursos. Por ejemplo, entre `C:\ExportTemp\StoreAssets`
   e. Pulse **Finalizar**.

El paso siguiente es copiar estos recursos en la ubicación apropiada del WebSphere Commerce Server de destino.

Transferencia de elementos de tienda

Para transferir elementos de tienda de la máquina de desarrollo al WebSphere Commerce Server de destino, realice lo siguiente:

1. En función de los tipos de elementos que esté desplegando y de los detalles de configuración específicos, es posible que necesite detener la instancia de WebSphere Commerce que se ejecuta en WebSphere Application Server. Si no está seguro de si será necesario realizar un reinicio para el escenario de despliegue en particular, deberá detener la instancia. Consulte la publicación *WebSphere Commerce, Guía de instalación* para su plataforma y base de datos si desea obtener detalles sobre cómo detener esta instancia.
2. En la máquina de destino, localice el directorio Stores.war. A continuación se muestra un ejemplo de este directorio:
   - 400 *dirusuario_WAS*/installedApps/*nombre_nodo_WAS*/WC_*nombre_instancia*.ear/Stores.war
   - AIX Linux Solaris *dir_instal_WAS*/installedApps/*nombreCélula*/WC_*nombre_instancia*.ear/Stores.war
   - Windows *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombre_instancia*.ear\Stores.war

   donde
   - *nombre_instancia* es el nombre de la instancia de WebSphere Commerce.

- 400 *nombre_nodo_WAS* representa el sistema iSeries en el que está instalado el producto WebSphere Application Server.

3. Copie los archivos exportados en el apartado “Exportación de elementos de tienda” en el directorio Stores.war.
4. Si ha detenido anteriormente la instancia de WebSphere Commerce, inicie la instancia. Consulte la publicación *WebSphere Commerce, Guía de instalación* para la plataforma y la base de datos si desea obtener detalles sobre cómo iniciar esta instancia.

Actualización de la base de datos de destino

Dado que el WebSphere Commerce Server de destino utiliza una base de datos diferente de la utilizada por máquina de desarrollo, deberá realizar todas las actualizaciones que se han efectuado en la base de datos de desarrollo en la base de datos utilizada por el WebSphere Commerce Server de destino. Esto incluye las actualizaciones para el registro de vistas o 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.

400 Es responsabilidad suya tener un programa de utilidad para ejecutar sentencias SQL. Una forma de hacerlo consiste en utilizar IBM iSeries Access para Windows. Para abrir este programa de utilidad, haga lo siguiente:

1. Abra **iSeries Navigator**.
2. Una vez abierto, se le solicitará que se conecte a un sistema específico. Asegúrese de seleccionar la máquina 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. En el panel de la izquierda, expanda el sistema iSeries y, a continuación, **BASES DE DATOS**. Pulse con el botón derecho del ratón en la Base de datos relacional y seleccione **Ejecutar scripts SQL** en la lista desplegable.
   Se abrirá 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 la opción **Configuración JDBC** bajo la selección **Conexión**.

Actualizaciones de control de acceso

Mientras está contenida en la base de datos, la información de control de acceso es un tipo especial de información que puede no ser necesariamente una duplicación directa del entorno de desarrollo en el entorno de destino. En particular, es posible que decida utilizar en el entorno de desarrollo políticas de control de acceso muy liberales que no son apropiadas para un entorno de producción (o del siguiente nivel de prueba). Por ejemplo, dentro de los límites del entorno de desarrollo, las políticas para mandatos nuevos pueden establecerse de forma que todos los usuarios puedan ejecutar el mandato, pero puede que esto no sea apropiado en cualquier otro lugar.

Por consiguiente, antes de copiar información de control de acceso del entorno de desarrollo en el entorno de destino, deberá tener en cuenta los requisitos de control de acceso en el nuevo entorno y ajustar las políticas como corresponda.

Para obtener información sobre cómo cargar políticas de control de acceso (incluyendo la sintaxis de mandatos para diversas plataformas y los requisitos de permisos de directorio), consulte la publicación *WebSphere Commerce, Guía de seguridad*.

Parte 4. Guías de aprendizaje

Las siguientes guías de aprendizaje están destinadas a presentar las diversas tareas relacionadas con la creación de código personalizado para las aplicaciones de WebSphere Commerce. Los pasos relacionados con el desarrollo deben realizarse en un entorno de desarrollo de WebSphere Commerce Business Edition o de WebSphere Commerce - Express, o en WebSphere Commerce Studio, Professional Developer Edition. Los pasos de despliegue se deben realizar en WebSphere Commerce (Business, Express o Professiona Edition, según su entorno de desarrollo) ejecutándose en Windows 2000.

Capítulo 10. Guía de aprendizaje: Creación de lógica de negocio nueva

Esta guía de aprendizaje está destinada a mostrarle los pasos incluidos en la creación de lógica de negocio nueva. Los tipos de elementos creados incluyen una vista nueva, un mandato de controlador nuevo, un mandato de tarea nuevo, beans de datos nuevos y un bean de entidad nuevo. Esta guía de aprendizaje utiliza el escenario de desarrollo de una pequeña interfaz para permitir a los usuarios modificar un saldo de puntos de bonificación. Sólo sirve para realizar una demostración y no refleja la lógica necesaria para crear una aplicación de programa fiel. En cambio, en esta guía de aprendizaje, aprenderá los pasos de desarrollo que son comunes para crear cada uno de los tipos de elementos de código listados anteriormente.

Esta guía de aprendizaje está dividida en las subtareas siguientes:

1. Preparación del espacio de trabajo
2. Creación de una vista nueva
3. Creación de un mandato de controlador nuevo
4. Paso de información del mandato de controlador a la vista
5. Análisis y validación de parámetros de URL en el mandato de controlador
6. Creación de un mandato de tarea nuevo
7. Modificación del mandato de tarea nuevo
8. Creación de un bean enterprise nuevo:
   a. Creación de una tabla nueva
   b. Creación de un bean enterprise CMP nuevo
   c. Correlación del nuevo bean con la tabla y creación del esquema
   d. Creación de un bean de acceso asociado
   e. Generación de código desplegado
   f. Comprobación del bean con el cliente de prueba universal
9. Integración del bean Bonus en MyNewControllerCmd
   a. Modificación del método performExecute de la clase MyNewTaskCmdImpl para calcular los nuevos puntos de bonificación y guardar los puntos en la tabla XBONUS.
   b. Adición de un método getResources en la clase MyNewControllerCmdImpl para devolver una lista de recursos que el mandato utiliza. Este método se incluye para control de acceso.
   c. Creación de BonusDataBean para que los puntos de bonificación se puedan visualizar fácilmente en una plantilla JSP.
   d. Creación de una política de control de acceso nueva para los recursos nuevos.
   e. Modificación del archivo MyNewJSPTemplate.jsp para permitir a los usuarios entrar puntos de bonificación y visualizar los resultados.
   f. Comprobación del código integrado.
10. Despliegue de todo el código anterior, las políticas de control de acceso, las plantillas JSP, las imágenes y los paquetes de recursos en un WebSphere Commerce Server de destino que se ejecuta en WebSphere Application Server.

Cada uno de los pasos anteriores se describe detalladamente en las secciones subsiguientes.

Localización del código de ejemplo

Antes de empezar esta guía de aprendizaje, baje el paquete WC_SAMPLE_55.zip que contiene el punto de partida para estas guías de aprendizaje de programación. Guarde este archivo en la máquina de desarrollo. Por ejemplo, puede guardar el archivo en el directorio *dir_instal_WCDE*:

Este paquete está ubicado con la publicación *WebSphere Commerce, Guías de programación y aprendizaje* en el siguiente sitio Web:

http://www.ibm.com/software/commerce/library/

Preparación del espacio de trabajo

En este paso, importará el código de ejemplo al espacio de trabajo de WebSphere Commerce. Este código de ejemplo es el punto de partida de la guía de aprendizaje.

Para preparar el espacio de trabajo, realice lo siguiente:

1. Asegúrese de tener instalado el Entorno de desarrollo de WebSphere Commerce Versión 5.5 y de haber completado la configuración del entorno de desarrollo. También tiene que haber publicado una tienda basada en la tienda de ejemplo FashionFlow (que es un ejemplo del modelo directo al consumidor) en el entorno de desarrollo. En la Ayuda en línea a la producción y al desarrollo de WebSphere Commerce, encontrará instrucciones sobre cómo publicar una tienda.
2. Importe el código de ejemplo al espacio de trabajo, realizando lo siguiente:
   a. Inicie entorno de desarrollo de WebSphere Commerce de este modo:
      - Business Professional **Inicio > Programas > IBM WebSphere Commerce Studio > entorno de desarrollo de WebSphere Commerce**
      - Express Express **Inicio > Programas > IBM WebSphere - Express Developer Edition > entorno de desarrollo de WebSphere Commerce**
   b. Cambie a la perspectiva J2EE (**Ventana > Abrir perspectiva > J2EE**) y, a continuación, seleccione la vista Business Professional Navegador J2EE Express Navegador de proyectos.
   c. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
   d. Pulse el botón derecho del ratón en la carpeta **src** y seleccione **Importar**. Se abrirá el asistente para Importar.
   e. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
   f. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
      suDirectorio\WC_SAMPLE_55.zip
      donde *suDirectorio* es el directorio en el que ha bajado el paquete.
   g. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione importar los archivos siguientes:
      - com\ibm\commerce\sample\commands\ MyNewControllerCmd.java

- com\ibm\commerce\sample\commands\MyNewControllerCmdImpl.java
- com\ibm\commerce\sample\commands\MyNewTaskCmd.java
- com\ibm\commerce\sample\commands\MyNewTaskCmdImpl.java
- com\ibm\commerce\sample\databeans\MyNewDataBean.java

h. En el campo **Carpeta**, la carpeta `WebSphereCommerceServerExtensionsLogic/src` ya está especificada. Conserve este valor.

i. Pulse **Finalizar**.

3. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Reconstruir proyecto**.
4. Importe la plantilla JSP para las guías de aprendizaje al directorio apropiado, realizando lo siguiente:

   a. En la vista Business Professional Navegador J2EE Express Navegador de proyectos, expanda el proyecto **Stores**. A continuación, expanda **Web Content > *nombre_FashionFlow*** donde *nombre_FashionFlow* es el nombre de la tienda basada en la tienda de ejemplo FashionFlow.

Si acaba de publicar la tienda, es posible que el directorio *nombre_FashionFlow* no se visualice. Si es así, pulse con el botón derecho del ratón en el proyecto Stores y seleccione **Renovar**. Ahora el directorio *nombre_FashionFlow* estará disponible en el espacio de trabajo.

   b. Pulse con el botón derecho del ratón en el directorio ***nombre_FashionFlow*** y seleccione **Importar**.
   Se abrirá el asistente para Importar.

   c. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.

   d. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
   suDirectorio\WC_SAMPLE_55.zip
   donde *suDirectorio* es el directorio en el que ha bajado el paquete.

   e. Pulse **Deseleccionar todo** y, a continuación, seleccione el archivo MyNewJSPTemplate_All.jsp.

   f. En el campo **Carpeta**, la carpeta `Stores/Web Content/`*`nombre_FashionFlow`* ya está especificada. Conserve este valor.

   g. Pulse **Finalizar**.

5. Importe el archivo de propiedades para la guía de aprendizaje al directorio apropiado, realizando lo siguiente:

   a. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, expanda los siguientes directorios:
   Stores > Web Content> WEB-INF >classes > *nombre_FashionFlow*.

   b. Pulse con el botón derecho del ratón en el directorio ***nombre_FashionFlow*** y seleccione **Importar**.
   Se abrirá el asistente para Importar.

   c. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.

   d. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:

suDirectorio\WC_SAMPLE_55.zip
donde *suDirectorio* es el directorio en el que ha bajado el paquete.

e. Pulse **Deseleccionar todo** y, a continuación, seleccione el archivo Tutorial_All_en_US.properties.

f. En el campo **Carpeta**, la carpeta `Stores/Web Content/WEB-INF/classes/`*`nombre_FashionFlow`* ya está especificada. Conserve este valor.

g. Pulse **Finalizar**.

6. Hay cuatro archivos que se utilizan con el fin de las políticas de control de acceso para los recursos nuevos que se crean en las guías de aprendizaje. Son:
   - MyNewViewACPolicy.xml — Este archivo XML contiene la política de control de acceso utilizada al crear una vista nueva.
   - MyNewControllerCmdACPolicy.xml— Este archivo XML contiene la política de control de acceso utilizada al crear un nuevo mandato de controlador.
   - SampleACPolicy_template.xml— Este archivo XML contiene la política de control de acceso utilizada al crear un nuevo bean enterprise.
   - SampleACPolicy_template_en_US.xml— Este archivo XML contiene la descripción de política de control de acceso utilizada al crear un nuevo bean enterprise.

   Copie los archivos anteriores en el directorio apropiado, realizando lo siguiente:

   a. En el sistema de archivos, vaya al directorio siguiente: *suDirectorio*\WC_SAMPLE_55.zip.

   b. Expanda el archivo ZIP y extraiga los cuatro archivos anteriores en el directorio siguiente: *dir_instal_WCDE*\Commerce\xml\policies\xml

7. Pruebe el entorno para asegurarse de que está listo para iniciar las guías de aprendizaje, realizando lo siguiente:

   a. En WebSphere Studio Application Developer, conmute a la perspectiva de Servidor.

   b. Inicie el servidor de pagos. Si está ejecutando un servidor de pagos local, pulse con el botón derecho del ratón en **WebSphereCommercePaymentsServer** y seleccione **Iniciar** (o **Reiniciar**).

   c. Pulse con el botón derecho del ratón en **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).

   d. Observe la consola para ver cuándo finaliza el proceso de arranque del servidor WebSphereCommerceServer. Cuando el servidor se haya iniciado, verá información similar a la siguiente:

```
[4/2/03 12:56:06:286 EST] 66adf8d1 ApplicationMg A WSVR0221I:
   Aplicación iniciada: WebSphereCommerceServer
[4/2/03 12:56:06:777 EST] 66adf8d1 HttpTransport A SRVE0171I:
   El transporte http está escuchando en el puerto 9,080.
[4/2/03 12:56:10:742 EST] 66adf8d1 HttpTransport A SRVE0171I:
   El transporte https está escuchando en el puerto 9,443.
[4/2/03 12:56:10:762 EST] 66adf8d1 HttpTransport A SRVE0171I:
   El transporte http está escuchando en el puerto 8,080.
[4/2/03 12:56:10:762 EST] 66adf8d1 HttpTransport A SRVE0171I:
   El transporte http está escuchando en el puerto 80.
[4/2/03 12:56:11:123 EST] 66adf8d1 HttpTransport A SRVE0171I:
   El transporte https está escuchando en el puerto 443.
[4/2/03 12:56:11:183 EST] 66adf8d1 HttpTransport A SRVE0171I:
   El transporte https está escuchando en el puerto 9,043.
[4/2/03 12:56:11:653 EST] 66adf8d1 RMIConnectorC A ADMC0026I:
   Conector RMI disponible en el puerto 2809
[4/2/03 12:56:12:575 EST] 66adf8d1 WsServer
   A WSVR0001I: Servidor server1 abierto para e-business
```

e. En la vista Business Professional Navegador J2EE Express Navegador de proyectos, expanda el proyecto **Stores**. A continuación, expanda **Web Content** > ***nombre_FashionFlow***.

f. Pulse con el botón derecho del ratón en el archivo **index.jsp** y seleccione **Ejecutar en servidor**.
Se abrirá la tienda de ejemplo.

g. Seleccione un producto y asegúrese de que puede comprarlo.

Ahora ya está preparado para continuar con las guías de aprendizaje.

Creación de una vista nueva

El primer paso de esta guía de aprendizaje es crear una vista nueva. Esta vista nueva se denomina MyNewView y tiene una plantilla JSP correspondiente, MyNewJSPTemplate.jsp.

En esta sección de la guía de aprendizaje, aprenderá lo siguiente:

- Dónde se deben poner las plantillas JSP y los archivos de gráficos que se aplican a una tienda específica.
- Cómo utilizar WebSphere Studio Application Developer para crear la plantilla JSP.
- Cómo crear el archivo de propiedades que contiene el texto para la plantilla JSP.
- Cómo actualizar el registro de vista (tabla VIEWREG) con la nueva "MyNewView"
- Cómo configurar el control de acceso para la nueva vista.
- Cómo utilizar el entorno de prueba de WebSphere (WTE) para probar la nueva vista.

En general, la creación de una vista nueva incluye los pasos siguientes:

1. Denominar y registrar la vista en el registro de vista.
2. Crear archivos de propiedades nuevos en los que se almacena el texto que se puede traducir para las plantillas JSP.
3. Crear una plantilla JSP nueva para la vista nueva.
4. Crear y cargar políticas de control de acceso para la vista.

Registro de MyNewView

En este escenario, la vista que se crea se denomina MyNewView. Esta vista debe estar registrada en la tabla VIEWREG, que forma parte del registro de mandatos. Para registrar una vista nueva, sólo necesita utilizar una sentencia SQL simple para crear una nueva entrada en la tabla VIEWREG.

Antes de continuar con este paso, deberá conocer el identificador exclusivo de la tienda. Puede determinarlo ejecutando la siguiente consulta de SQL en la base de datos de desarrollo:

```
select STOREENT_ID from STOREENT where IDENTIFIER = 'nombre_FashionFlow'
```

donde *nombre_FashionFlow* es el nombre de la tienda. Tome nota del valor que encuentre aquí, ya que es necesario en la sección siguiente:

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyNewView:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**).
2. En el menú **Herramientas**, seleccione **Valores de herramientas**.

3. Marque el recuadro de selección **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (;)
4. Cierre los valores de herramientas.
5. Con la pestaña Script seleccionada, cree la entrada necesaria en la tabla VIEWREG, entrando la información siguiente en la ventana de script:

```
connect to BDdesarrollo user usuariobd using contraseñabd;
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, ID_enttienda_FF,
'com.ibm.commerce.command.ForwardViewCommand',
'com.ibm.commerce.command.HttpForwardViewCommandImpl',
'docname=MyNewJSPTemplate.jsp', 'This is my new view for tutorial one',
0, null)
```

   donde

   - *BDdesarrollo* es el nombre de la base de datos de desarrollo
   - *usuariobd* es el usuario de base de datos
   - *contraseñabd* es la contraseña para el usuario de base de datos
   - *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

   Pulse el icono **Ejecutar**.

   Verá un mensaje que indica que el mandato de SQL se ha completado satisfactoriamente.

Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar la vista en la base de datos:

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 SQL:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, ID_enttienda_FF,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

   donde

   - *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

   Pulse Intro para ejecutar la sentencia SQL.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

MyNewView ya se ha registrado.

Creación de un archivo de propiedades para la guía de aprendizaje

En este paso creará el archivo de propiedades nuevo que contendrá el texto que se puede traducir utilizado en la plantilla JSP. La separación del texto que se puede traducir de la propia plantilla JSP simplifica mucho la tarea de traducción y constituye la clave para tener un sitio Web globalizado.

Para crear el archivo de propiedades, realice lo siguiente en WebSphere Studio Application Developer:

1. Abra la perspectiva de Web (**Ventana > Abrir perspectiva > Web**).
2. En el proyecto Web **Stores**, expanda las carpetas **Web Content > WEB-INF > classes > *nombre_FashionFlow***. Encontrará el archivo Tutorial_All_en_US.properties.
3. Pulse con el botón derecho del ratón en **Tutorial_All_en_US.properties** y seleccione **Abrir con > Editor de archivo de propiedades**.
4. Pulse con el botón derecho del ratón en la carpeta ***nombre_FashionFlow*** y seleccione **Nuevo > Otro > Simple > Archivo > Siguiente** para crear un archivo de propiedades nuevo.
   Se abrirá la ventana Nuevo archivo.
5. En el campo **Nombre de archivo**, entre `TutorialNLS_en_US.properties` y, a continuación, pulse **Finalizar**.
   Se abrirá el nuevo archivo vacío.
6. Copie la sección 1 del archivo Tutorial_All_en_US.properties en el nuevo archivo TutorialNLS_en_US.properties. Esto introduce las siguientes parejas de nombre y valor en el archivo TutorialNLS_en_US.properties:

```
# -- SECTION 1 -- #

ProgrammerGuide=Programmer's Guide
Tutorial=Tutorial: Creating new business logic
ParametersFromCmd= List of parameter-value pairs sent from the
   controller command
CalledByControllerCmd=MyNewView was called by a controller command
CalledByWhichControllerCmd=MyNewView was called by the controller
   command which is -
ControllerParm1=ControllerParm1=
ControllerParm2=ControllerParm2=
Example=This is an example of using the <if> tag from JSP Standard
   Tag Library (JSTL)
UserName=UserName=
Points=Points=
Greeting=Greeting=
UserId=UserId=
FirstInput=Your first input parameter
RegisteredUser=is a registered user
ReferenceNumber=The member refernce number of this user is
NotRegisteredUser=is not a registered user
BonusAdmin=Bonus Administration
PointBeforeUpdate=The bonus point before update is
PointAfterUpdate=The bonus point after update is
EnterPoint=Please enter the points, then submit it to the controller
   command

# -- END OF SECTION 1 -- #
```

   Tenga en cuenta que los saltos de línea en un valor sólo se muestran a efectos de presentación.
7. Guarde el archivo TutorialNLS_en_US.properties (Control+S).

El nuevo archivo TutorialNLS_en_US.properties se guarda bajo el directorio Stores\Web Content\WEB-INF\classes*nombre_FashionFlow* y la plantilla JSP nueva lo utilizará como un paquete de recursos.

Nota: Cuando se modifique el contenido de un archivo de propiedades, se deberá reiniciar el servidor para que se puedan ver los cambios en la comprobación.

Creación de MyNewJSPTemplate

En este paso, utilizará la herramienta Page Designer de WebSphere Studio para crear una nueva plantilla JSP. En particular, debe crear MyNewJSPTemplate.jsp que se utiliza con MyNewView. Al crear esta plantilla, se crea una nueva plantilla JSP en blanco y, a continuación, se añaden las secciones apropiadas de otra plantilla JSP (MyNewJSPTemplate_All.jsp).

Para crear la nueva plantilla JSP, realice lo siguiente:

1. En la perspectiva de Web, cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Pulse con el botón derecho del ratón en el proyecto Web **Stores** y seleccione **Propiedades**.
3. Business Professional Seleccione **Web** en el panel izquierdo y, a continuación, en la lista Características de proyecto Web disponibles, seleccione **Incluir la biblioteca de códigos estándar JSP**.

   Express Seleccione **Características de proyecto Web** en el panel izquierdo y, a continuación, en la lista Características de proyecto Web disponibles, seleccione **Biblioteca de códigos estándar JSP**.

   Pulse **Aplicar**. Cuando la actualización se haya completado, pulse **Aceptar** para cerrar el editor de propiedades.
4. Expanda el directorio **Web Content*nombre_FashionFlow***.
5. Pulse con el botón derecho del ratón en el archivo **MyNewJSPTemplate_All.jsp** y seleccione **Abrir con > Page Designer**.
6. Pulse con el botón derecho del ratón en la carpeta ***nombre_FashionFlow*** y seleccione **Nuevo > Archivo JSP** para crear la nueva plantilla JSP en esta carpeta.
   Se abrirá la ventana Archivo JSP nuevo.
7. Especifique valores para el nuevo archivo, como se indica a continuación:
   a. En el campo **Nombre de archivo**, entre `MyNewJSPTemplate.jsp`.
   b. En la lista desplegable **Lenguaje de códigos**, seleccione **XHTML** y pulse **Siguiente**.
   c. Business Professional Pulse **Añadir biblioteca de códigos**.
      Se abrirá la ventana Seleccionar una biblioteca de códigos.

      Express Seleccione **Configurar opciones avanzadas** > pulse **Siguiente** > pulse **Añadir** (al lado de la tabla Bibliotecas de códigos).
   d. Seleccione las siguientes bibliotecas de códigos:
      - http://java.sun.com/jstl/core
      - http://java.sun.com/jstl/fmt

      Pulse **Aceptar** y luego **Siguiente**.
   e. Pulse **Siguiente**.
   f. Elimine la marca del recuadro de selección **(Utilizar el valor por omisión del entorno de trabajo) Codificación del entorno de trabajo**.

g. En la lista desplegable **Codificación**, seleccione **ISO Latin -1**.
h. En la lista desplegable **Tipo de documento**, seleccione **XHTML 1.0 Transitional**.
i. Pulse **Finalizar**.
Se abrirá el archivo MyNewJSPTemplate.jsp. Pulse las pestañas Diseño, Fuente y Vista previa para obtener diferentes vistas del archivo.

8. Con la pestaña Diseño seleccionada, pulse en el texto **Sitúe aquí el contenido de MyNewJSPTemplate.jsp**. Sustituya este texto por `Hello world!`.
9. Conmute a las pestañas Fuente y, a continuación, Vista previa. Observe que el texto ha cambiado.
10. Ahora deberá copiar una sección de preparación del archivo MyNewJSPTemplate_All.jsp en el nuevo archivo MyNewJSPTemplate.jsp. Esta sección establece los espacios reservados para las actualizaciones que realizará en el archivo. Copie el texto entre los marcadores `<%--PREPARATION SECTION` y `END OF PREPARATION SECTION --%>` en la nueva plantilla JSP. Cuando copie este texto en la plantilla JSP, grabe encima del texto siguiente:

```
<title> MyNewJSPTemplate.jsp </title>
</head>
<body>
<p> Hello World! </p>
</body>
</html>
```

Nota: No copie los marcadores `<%--PREPARATION SECTION` y `END OF PREPARATION SECTION --%>` en la nueva plantilla JSP. Copie sólo el texto contenido entre dichos marcadores.

11. Copie las secciones 1A y 2 del archivo MyNewJSPTemplate_All.jsp en el nuevo archivo MyNewJSPTemplate.jsp. Coloque el nuevo texto entre los marcadores `<!-- SECTION 1A -->`, `<!-- END OF SECTION 1A -->`, `<!-- SECTION 2 -->` y `<!-- END OF SECTION 2 -->`. Esto inserta el texto siguiente en MyNewJSPTemplate.jsp:

```
<!-- SECTION 1A -->

    <%@ include file="include/EnvironmentSetup.jsp"%>

<!-- END OF SECTION 1A -->

<!-- SECTION 2 -->

<fmt:setLocale value="${CommandContext.locale}" />
<fmt:setBundle basename="${sdb.directory}/TutorialNLS" var="tutorial" />

<!-- END OF SECTION 2-->
```

La primera sección incluye el archivo EnvironmentSetup.jsp que se utiliza para configurar variables de entorno. La segunda sección se utiliza para crear el objeto de paquete de recursos que se utiliza para recuperar información del archivo de propiedades y establece el entorno nacional.

12. Ahora añada un gráfico y texto en la plantilla JSP. De nuevo, este paso se realiza copiando texto del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esta vez, copie la sección 3 del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esto inserta el texto siguiente en la plantilla JSP:

```
<!-- SECTION 3 -->

<table cellpadding="0" cellspacing="0" border="0">
 <tr>
```

```
  <td bgcolor="#ff2d2d" >
   <img src="<c:out value="${sdb.jspStoreDirFilePath}
                           images/male_blueshirt.gif" />" border="0"/>
  </td>
 </tr>
</table>

<h1><fmt:message key="ProgrammerGuide" bundle="${tutorial}" /> </h1>

<h2><fmt:message key="Tutorial" bundle="${tutorial}" /> </h2>

<!-- END OF SECTION 3 -->
```

La sección 3 inserta una imagen que está ubicada en la subcarpeta de imágenes específicas de la tienda (Stores\Web Content*nombre_FashionFlow*\images). También recupera texto del archivo de propiedades.

13. Guarde los cambios que ha realizado en el archivo MyNewJSPTemplate.jsp (Control+S).

Creación y carga de políticas de control de acceso para MyNewView

Se debe especificar control de acceso a nivel de mandatos para la nueva vista. En este caso, la política de control de acceso a nivel de mandatos especifica que todos los usuarios pueden ejecutar la vista. Tenga en cuenta que este tipo de política de control de acceso es aceptable para el entorno de desarrollo, pero es posible que no sea adecuado en otras circunstancias. Para conocer los requisitos de control de acceso más avanzados, consulte la publicación *WebSphere Commerce, Guía de seguridad*.

La política de control de acceso se define mediante el archivo MyNewViewACPolicy.xml, que se coloca en el directorio siguiente, como parte de los pasos de preparación:
dir_instal_WCDE\Commerce\xml\policies\xml

Para cargar la nueva política, realice lo siguiente:

1. En un indicador de mandatos, vaya al directorio siguiente:
   dir_instal_WCDE\Commerce\bin
2. 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 la base de datos de desarrollo.
   - *usuario_bd* es el nombre del usuario de base de datos.
   - *contraseña_bd* es la contraseña del usuario de base de datos.
   - *archXMLentrada* es el archivo XML que contiene la especificación de política de control de acceso. En este caso, especifique `MyNewViewACPolicy.xml`.

   A continuación, se muestra un ejemplo del mandato en el que se especifican variables:
   ```
   acpload Demo_Dev db2user db2user MyNewViewACPolicy.xml
   ```

Comprobación de MyNewView

El paso final para crear una vista nueva es comprobarla en el entorno de prueba de WebSphere. Tenga en cuenta que cuando se comprueba la vista nueva (y,

posteriormente, cuando se comprueban los nuevos mandatos) se debe iniciar primero la página de presentación de la tienda. La razón por la que es necesario iniciar la tienda es porque la nueva vista está registrada específicamente para la tienda. Por lo tanto, antes de intentar acceder a la vista nueva debe iniciar la página de presentación de la tienda para establecer el valor de ID de tienda en el contexto de mandato. De forma similar, el mandato de controlador que se crea posteriormente también está registrado específicamente para la tienda y necesita el ID de tienda del contexto de mandato.

Para probar la nueva vista, realice lo siguiente:

1. En WebSphere Studio Application Developer, abra la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. En el navegador Web, entre el URL siguiente:
   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewView
   ```

   Después de unos segundos, se visualizará la nueva plantilla JSP, como se muestra en la captura de pantalla siguiente:

Figura 31.

Creación de un nuevo mandato de controlador

En este paso, creará un nuevo mandato de controlador, denominado MyNewControllerCmd. Inicialmente, este mandato sólo devuelve la vista MyNewView.

En esta sección de la guía de aprendizaje, aprenderá lo siguiente:

- Los requisitos mínimos para el código contenido en un mandato de controlador
- Cómo crear la interfaz de mandato de controlador y la clase de implementación nuevas
- Cómo configurar un mandato de controlador para devolver una vista
- Cómo registrar un mandato de controlador en el registro de mandatos
- Cómo configurar el control de acceso para un mandato de controlador

En general, la creación de un nuevo mandato de controlador incluye los pasos siguientes:

1. Registro del nuevo mandato en el registro de mandatos.
2. Creación de una interfaz para el mandato.
3. Creación de una clase de implementación para el mandato.
4. Creación y carga de políticas de control de acceso para el mandato.
5. Comprobación del mandato.

Registro de MyNewControllerCmd

En esta parte de la guía de aprendizaje, crear un nuevo mandato de controlador denominado MyNewControllerCmd. Este mandato debe registrarse en el registro de mandatos. En particular, se debe registrar la interfaz en la tabla URLREG y la asociación entre la interfaz y su clase de implementación se registra en la tabla CMDREG.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyNewControllerCmd:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**).
2. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla URLREG indicando la siguiente información en la ventana de script:

```
connect to BDdesarrollo user usuariobd using contraseñabd;
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl',
   'local');
```

donde

- *BDdesarrollo* es el nombre de la base de datos de desarrollo
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña para el usuario de base de datos
- *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

Pulse el icono **Ejecutar**.

Verá un mensaje que indica que el mandato de SQL se ha completado satisfactoriamente.

Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyNewControllerCmd:

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 SQL:

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl','local');
```

   donde

   - *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

   Pulse Intro para ejecutar la sentencia SQL.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

Tenga en cuenta que la segunda sentencia 'insert' en la tabla CMDREG no es absolutamente necesaria. En este escenario, la interfaz utilizará la implementación por omisión y, de este modo, esta asociación entre la interfaz y la clase de implementación no necesita realmente especificarse en el registro de mandatos. Aquí se incluye para que el ejemplo sea más completo.

Creación de la interfaz MyNewControllerCmd

De acuerdo con el modelo de programación de WebSphere Commerce, todos los nuevos mandatos de controlador deben tener una interfaz, así como una clase de implementación. Para esta guía de aprendizaje, se proporciona una base para la interfaz en el código de ejemplo. Se divide en varias secciones diferentes que actualmente están como comentarios en el código. A medida que avance por la guía de aprendizaje, deberá eliminar las marcas de comentario de diversas secciones del código.

Para crear la interfaz MyNewControllerCmd, realice lo siguiente:

1. En WebSphere Studio Application Developer, abra la perspectiva Java (**Ventana > Abrir perspectiva > Java**).
2. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.

3. Vaya al directorio **src** y, a continuación, expanda el paquete **com.ibm.commerce.sample.commands**.
4. Efectúe una doble pulsación en la interfaz **MyNewControllerCmd.java** para abrir el archivo.
5. En el código fuente, elimine la marca de comentario de la sección 1 (suprima "/* " antes de la sección y "*/" después de la sección). Esto inserta el siguiente código en la interfaz:

```
/// Section 1 ////////////////////////////////////////////

// set default command implement class

 static final String defaultCommandClassName =
     "com.ibm.commerce.sample.commands.MyNewControllerCmdImpl";


/// End of section 1/////////////////////////////////////
```

   Esta sección de código especifica que, por omisión, la interfaz deberá utilizar la clase de implementación MyNewControllerCmdImpl.
6. Guarde el cambio realizado en la interfaz (Control+S).

Creación de la clase de implementación MyNewControllerCmdImpl

Una vez creada la interfaz, el siguiente paso es crear la clase de implementación para el mandato. Para esta guía de aprendizaje, se proporciona una base para la clase de implementación en el código de ejemplo. Se divide en varias secciones diferentes que actualmente están como comentarios en el código. A medida que avance por la guía de aprendizaje, deberá eliminar las marcas de comentario de diversas secciones del código.

Para crear la clase de implementación MyNewControllerCmdImpl, realice lo siguiente:

1. Efectúe una doble pulsación en la clase **MyNewControllerCmdImpl.java** para abrirla.
2. En la vista de Esquema, seleccione el método **performExecute** para ver el código fuente.
3. En el código fuente para el método performExecute, elimine la marca de comentario de la Sección 1. Esto inserta el código siguiente en el método:

```
/// Section 1 ////////////////////////////////////////////

    /// create a new TypedProperties for output purpose.

 TypedProperty rspProp = new TypedProperty();

/// End of section 1 /////////////////////////////////////
```

   Esto crea un objeto TypedProperty nuevo que se utiliza para incluir las propiedades de respuesta del mandato.
4. En el código fuente para el método performExecute, elimine la marca de comentario de la sección 5. Esto inserta el código siguiente en el método:

```
/// Section 5 ////////////////////////////////////////////

    /// see how controller command call a JSP

 rspProp.put(ECConstants.EC_VIEWTASKNAME, "MyNewView");
```

```
setResponseProperties(rspProp);

/// End of section 5/////////////////////////////////////////
```

Esta sección de código lleva a cabo dos tareas principales. En primer lugar, es un requisito del modelo de programación de WebSphere Commerce que todos los mandatos de controlador devuelvan una vista. En esta sección, especifica que la vista que se debe devolver es MyNewView, que se ha creado anteriormente. Adicionalmente, establece las propiedades de respuesta del mandato para que sean el nuevo objeto rspProp.

5. Guarde los cambios (Control+S).
6. Para compilar los cambios que ha realizado en el código, pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Crear proyecto**.

Creación y carga de políticas de control de acceso para el mandato

Se debe especificar el control de acceso a nivel de mandatos para el nuevo mandato. En este caso, la política de control de acceso a nivel de mandatos especifica que todos los usuarios pueden ejecutar el mandato. Tenga en cuenta que este tipo de política de control de acceso es aceptable para el entorno de desarrollo, pero es posible que no sea adecuado en otras circunstancias. Para conocer los requisitos de control de acceso más avanzados, consulte la publicación *WebSphere Commerce, Guía de seguridad*.

La política de control de acceso se define mediante el archivo MyNewControllerCmdACPolicy.xml, que se coloca en el directorio siguiente, como parte de los pasos de preparación:
dir_instal_WCDE\Commerce\xml\policies\xml

Para cargar la nueva política, realice lo siguiente:

1. En un indicador de mandatos, vaya al directorio siguiente:
   dir_instal_WCDE\Commerce\bin
2. 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 la base de datos de desarrollo.
   - *usuario_bd* es el nombre del usuario de base de datos.
   - *contraseña_bd* es la contraseña del usuario de base de datos.
   - *archXMLentrada* es el archivo XML que contiene la especificación de política de control de acceso. En este caso, especifique `MyNewControllerCmdACPolicy.xml`.

   A continuación, se muestra un ejemplo del mandato en el que se especifican variables:
   ```
   acpload Demo_Dev db2user db2user MyNewControllerCmdACPolicy.xml
   ```

Comprobación de MyNewControllerCmd

Ahora que se han creado la interfaz, la clase de implementación, el registro de mandatos y la información de control de acceso, puede probar el nuevo mandato de controlador.

Cuando se ha modificado código Java, se debe reiniciar el servidor de prueba para que se reconozcan los cambios.

Para probar el nuevo código, realice lo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. En el navegador Web, entre el URL siguiente:
   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   Después de unos segundos, se visualizará la nueva plantilla JSP, como se muestra en la captura de pantalla siguiente:

Figura 32.

Cómo pasar información de MyNewControllerCmd a MyNewView

En este paso, modificará MyNewControllerCmd para que pase información a MyNewView. Se muestran dos modos diferentes para pasar información a la vista. En primer lugar, aprenderá a utilizar el objeto TypedProperties para las propiedades de respuesta y a extraer información de este objeto en la plantilla JSP. En segundo lugar, aprenderá a crear un bean de datos nuevo que se utiliza para pasar información a la plantilla JSP.

Cómo pasar información utilizando un objeto TypedProperties

En esta sección, modificará MyNewControllerCmdImpl para pasar información a la plantilla JSP. En particular, modificará el mandato para añadir parejas de nombre y valor adicionales en el objeto rspProp TypedProperties existente que se utiliza para las propiedades de respuesta del mandato. En la plantilla JSP, utilizará el lenguaje de expresión JSTL para extraer la información del archivo de respuestas.

En esta sección de la guía de aprendizaje, aprenderá lo siguiente:

- Cómo modificar el mandato de controlador para incluir propiedades de respuesta adicionales en TypedProperty
- Cómo modificar la plantilla JSP para recuperar información de las propiedades de respuesta utilizando el lenguaje de expresión JSTL

Para habilitar la visualización de información del objeto TypedProperties en la plantilla JSP, realice lo siguiente:

1. El primer paso es modificar la clase MyNewControllerCmdImpl, del modo siguiente:
   a. Conmute a la perspectiva Java.
   b. Expanda estos directorios: **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands**.
   c. Efectúe una doble pulsación en **MyNewControllerCmdImpl.java** y seleccione el método **performExecute** en la vista de Esquema.
   d. En el código fuente para el método performExecute, elimine la marca de comentario de la Sección 2. Esto inserta el código siguiente en el método:

```
/// Section 2 ////////////////////////////////////////////////

    /// see how the controller command pass in variables to JSP

 /// add additional parameters in controller command to rspProp
  ///  for response
    String message1 = "Hello from IBM!";

 rspProp.put("controllerParm1", message1);
 rspProp.put("controllerParm2", "Have a nice day!");

/// End of section 2//////////////////////////////////////////
```

   El fragmento de código anterior crea dos parámetros nuevos que se ponen en el objeto de propiedades de respuesta. Finalmente este objeto se pasa a la vista.
   e. Guarde los cambios.
   f. Compile el mandato pulsando el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccionando **Crear proyecto**.
2. A continuación, deberá actualizar el archivo MyNewJSPTemplate.jsp, realizando lo siguiente:

a. Si los archivos de plantilla JSP aún no están abiertos, realice lo siguiente:
   1) En la perspectiva de Web, cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos y expanda el proyecto Web **Stores**.
   2) Vaya al directorio **Web Content*nombre_FashionFlow***.
   3) Pulse con el botón derecho del ratón en **MyNewJSPTemplate_All.jsp** y seleccione **Abrir con > Page Designer**.
   4) Pulse con el botón derecho del ratón en **MyNewJSPTemplate.jsp** y seleccione **Abrir con > Page Designer**.

b. Copie la sección 4 del archivo MyNewJSPTemplate_All.jsp en el nuevo archivo MyNewJSPTemplate.jsp. Coloque el nuevo texto entre los marcadores `<!-- SECTION 4 -->` y `<!-- END OF SECTION 4 -->`. Esto inserta el texto siguiente en MyNewJSPTemplate.jsp:

```
<!-- SECTION 4 -->

<h3><fmt:message key="ParametersFromCmd" bundle="${tutorial}" /> </h3>

<fmt:message key="ControllerParm1" bundle="${tutorial}" />
<c:out value="${controllerParm1}"/> <br />

<fmt:message key="ControllerParm2" bundle="${tutorial}" />
<c:out value="${controllerParm2}"/> <br /> <br />

<!-- END OF SECTION 4 -->
```

   Esta sección utiliza el lenguaje de expresión JSTL para obtener y visualizar los valores que se han pasado desde el mandato de controlador.

c. Guarde los cambios.

3. El siguiente paso es comprobar las modificaciones en el mandato de controlador y en la plantilla JSP, realizando lo siguiente:
   a. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
   b. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
   c. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**. Se visualizará la página de presentación de la tienda en el navegador Web.
   d. En el navegador Web, entre el URL siguiente:

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
```

      Después de unos segundos, se visualizará la nueva plantilla JSP, como se muestra en la captura de pantalla siguiente:

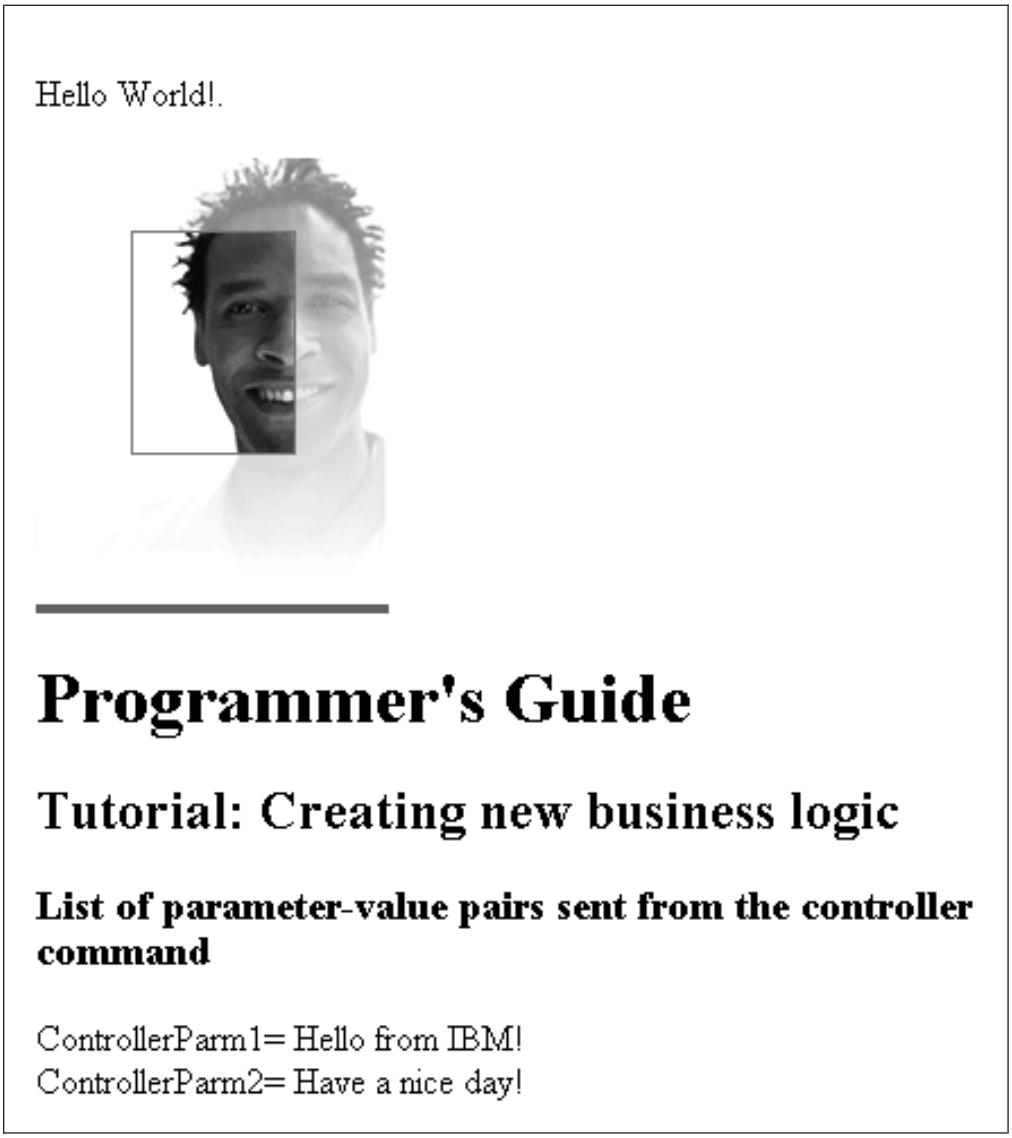


Figura 33.

Cómo pasar información utilizando un bean de datos

En esta sección, añadirá código para determinar si la vista se ha llamado directamente o si la ha llamado el mandato de controlador. En el último caso, la plantilla JSP también visualizará el nombre del mandato que la ha llamado.

Para pasar esta información a la vista, se crea un nuevo bean de datos denominado MyNewDataBean. MyNewJSPTemplate también se modifica para poder visualizar la nueva información.

MyNewDataBean se utiliza estrictamente para dejar la información del mandato de controlador disponible en la plantilla JSP. Compare esto con la tarea de dejar disponible la información de la base de datos. Más adelante en la guía de aprendizaje, aprenderá a crear un nuevo bean de datos cuya finalidad es dejar la información de la base de datos disponible en la plantilla JSP.

Es posible que se pregunte por qué esta guía de aprendizaje utiliza un bean de datos sólo para dejar disponible la información del mandato de controlador. Hay dos razones para crear este bean: la primera es que es una buena práctica de

programación que permite una agrupación lógica de atributos y la segunda es que simplifica a los desarrolladores de páginas Web la tarea de añadir información a la página Web utilizando un bean de datos, en lugar de utilizando el objeto de propiedades de respuesta TypedProperties.

En esta sección de la guía de aprendizaje, aprenderá lo siguiente:

- Cómo crear un bean de datos nuevo
- Cómo modificar el mandato de controlador para crear una instancia de un bean de datos
- Cómo establecer atributos en un bean de datos utilizando el mandato de controlador
- Cómo pasar a la plantilla JSP el bean de datos del que se ha creado una instancia
- Cómo modificar la plantilla JSP para recuperar información del bean de datos
- Consulte un ejemplo de utilización del código <if> en la plantilla JSP

Creación de MyNewDataBean

MyNewDataBean se utiliza para pasar información a la página MyNewJSPTemplate.jsp. Como con otras secciones de esta guía de aprendizaje, se proporciona una base para el bean de datos en el código de ejemplo. Se divide en varias secciones diferentes que actualmente están como comentarios en el código. A medida que avance por la guía de aprendizaje, deberá eliminar las marcas de comentario de diversas secciones del código.

Para crear MyNewDataBean, realice lo siguiente:

1. Abra la perspectiva Java y utilice la vista de Explorador de paquetes.
2. Expanda estos directorios: **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.databeans**.
3. Efectúe una doble pulsación en **MyNewDataBean.java** para ver el código fuente.
4. En el código fuente para la clase principal, elimine la marca de comentario de la Sección 1. Esto inserta el código siguiente en la clase:

```
 /// Section 1 ////////////////////////////////////////////
/// create fields and accessors (setter/getter methods)

 private java.lang.String callingCommandName = null;
 private boolean calledByControllerCmd = false;


 public java.lang.String getCallingCommandName() {
  return callingCommandName;
 }

 public void setCallingCommandName(java.lang.String newCallingCommandName)
  {
  callingCommandName = newCallingCommandName;
 }

 public boolean getCalledByControllerCmd() {
  return calledByControllerCmd;
 }

 public void setCalledByControllerCmd(boolean newCalledByControllerCmd)
  {
  calledByControllerCmd = newCalledByControllerCmd;
 }

/// End of Section 1 ////////////////////////////////////
```

El código anterior inserta dos variables que se utilizan para visualizar información cuando la vista la ha devuelto un mandato de controlador, en lugar de que la haya llamado directamente el URL para la vista.

5. Guarde los cambios.

Creación de una instancia de MyNewDataBean y establecimiento de los atributos utilizando MyNewControllerCmd

En este paso modificará MyNewControllerCmdImpl para crear una instancia de MyNewDataBean y establecer los atributos de este bean.

Para modificar MyNewControllerCmdImpl, realice lo siguiente:

1. En la perspectiva Java, expanda estos directorios: **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands**.
2. Efectúe una doble pulsación en **MyNewControllerCmdImpl.java** para ver el código fuente.
3. En el código para la clase principal, elimine la marca de comentario de la Sección 1 de importación para que el nuevo bean de datos esté disponible en esta clase. Esto inserta el código siguiente en la clase:

```
/// Import Section 1 ////////////////////////////
import com.ibm.commerce.sample.databeans.*;
/// End of Import Section 1 ////////////////////////
```

4. En la vista de Esquema, seleccione el método **performExecute**.
5. En el código fuente para el método performExecute, elimine la marca de comentario de las Secciones 3A y 3B. Esto inserta el código siguiente en el método:

```
/// Section 3A/////////////////////////////////////

 /// instantiate the MyNewDataBean databean and set the properties,
  ///  then add the instance to resProp for response

 MyNewDataBean mndb = new MyNewDataBean();
 mndb.setCallingCommandName(this.getClass().getName());
 mndb.setCalledByControllerCmd(true);

/// end of section 3A/////////////////////////////////////

/// Section 3B/////////////////////////////////////
 rspProp.put("mndbInstance", mndb);

/// end of section 3B/////////////////////////////////////
```

   El fragmento de código anterior crea una instancia del objeto MyNewDataBean, establece dos parámetros en el objeto (que indican que lo ha llamado un mandato de controlador y qué mandato lo ha llamado) y, a continuación, pone el objeto de bean de datos en las propiedades de respuesta para que quede disponible en la plantilla JSP.

6. Guarde los cambios.
7. Compile los cambios de código pulsando el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccionando **Crear proyecto**.

Utilización de MyNewDataBean en MyNewJSPTemplate

En esta sección modificará MyNewJSPTemplate para indicar si la vista la ha devuelto un mandato de controlador. Si la ha devuelto un mandato de controlador, también deberá visualizar el nombre de dicho de mandato de controlador. Para

determinar esto, la plantilla JSP utiliza códigos JSTL para la lógica condicional, basándose en los valores de MyNewDataBean.

Para modificar la plantilla JSP, realice lo siguiente:

1. Si los archivos de plantilla JSP aún no están abiertos, realice lo siguiente:
   a. En la perspectiva de Web, cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos y expanda el proyecto Web **Stores**.
   b. Vaya al directorio **Web Content*nombre_FashionFlow***.
   c. Resalte los archivos **MyNewJSPTemplate_All.jsp** y **MyNewJSPTemplate.jsp**, pulse con el botón derecho del ratón y seleccione **Abrir con > Page Designer**.
2. Copie la Sección 5 del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esto inserta el texto siguiente en la plantilla JSP:

```
<!-- SECTION 5 -->

<c:if test="${mndbInstance.calledByControllerCmd}">
   <fmt:message key="Example" bundle="${tutorial}" /> <br />
   <fmt:message key="CalledByControllerCmd" bundle="${tutorial}" />
      <br />
   <fmt:message key="CalledByWhichControllerCmd" bundle="${tutorial}" />
   <b><c:out value="${mndbInstance.callingCommandName}" /></b> <br />
      <br />
</c:if>

<!-- END OF SECTION 5 -->
```

   Esta sección de código utiliza el código JSTL <if> para determinar si se debe visualizar información acerca del mandato de controlador que realiza la llamada. También recupera texto traducible del paquete de recursos para la guía de aprendizaje.
3. Guarde los cambios.

Comprobación de la plantilla JSP modificada

Para comprobar la plantilla JSP modificada, realice lo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. En el navegador Web, entre el URL siguiente:

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
```

   Después de unos segundos, se visualizará la plantilla JSP, como se muestra en la captura de pantalla siguiente:

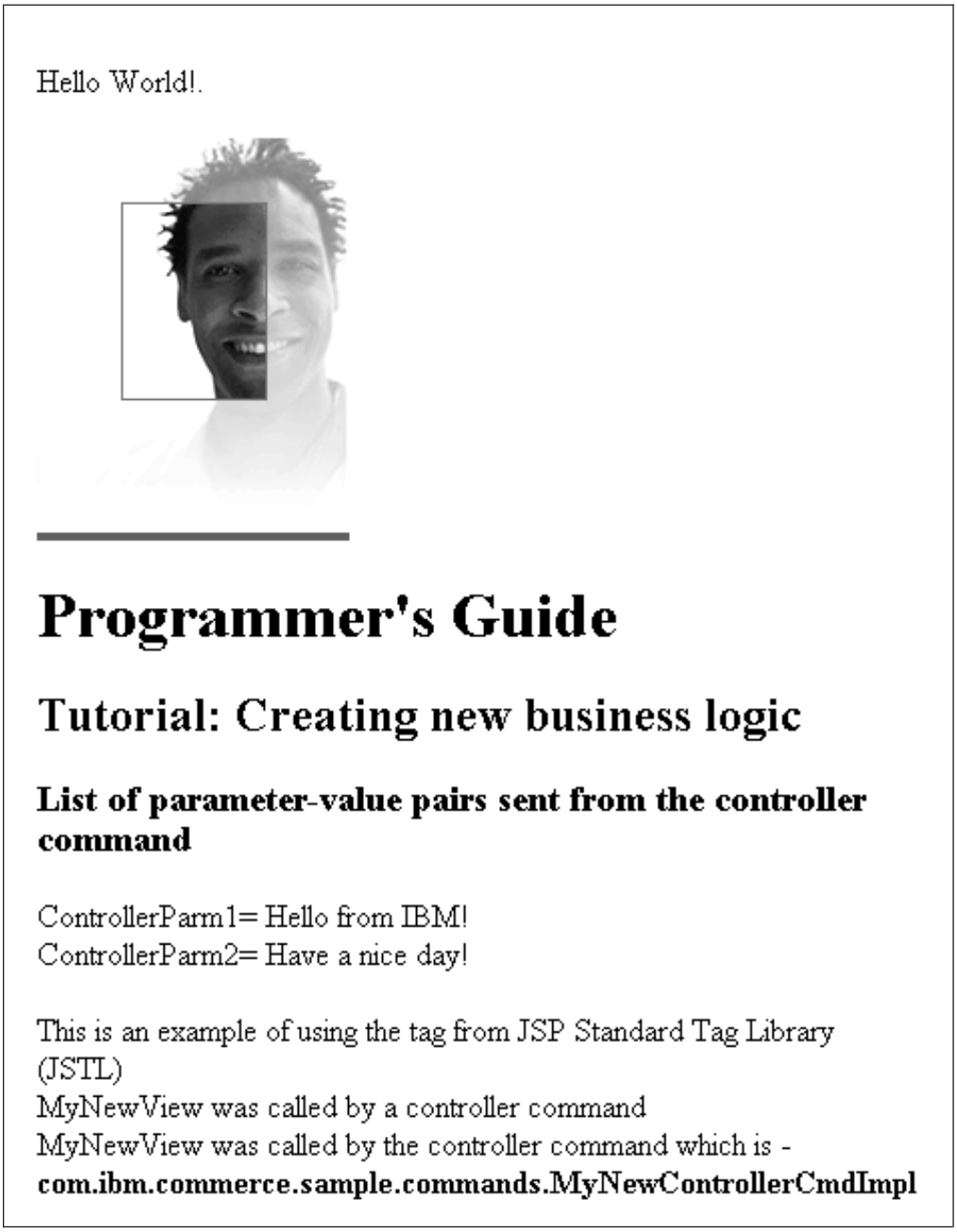


Figura 34.

5. A continuación, entre el URL para llamar a la vista directamente y observe la diferencia en la información visualizada:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewView
   ```

Análisis y validación de parámetros de URL en MyNewControllerCmd

En este paso, modificará el mandato de controlador para utilizar parámetros que se pasan a través del URL que llama al mandato de controlador. En el mandato también se incluye la lógica de validación para asegurar que se incluyen los parámetros necesarios, así como para asegurar que se utilizan los valores apropiados para estos parámetros.

El método validateParameters que hay actualmente en su nuevo mandato es simplemente un ″apéndice″ del mandato. Consta del siguiente código:

```
public void validateParameters() throws ECApplicationException {
}
```

Ahora deberá añadir al mandato la comprobación de parámetros personalizados y pasar los parámetros de URL a la plantilla JSP. Al modificar el método validateParameters, se añaden campos nuevos que corresponden a los parámetros de URL.

En esta sección de la guía de aprendizaje, aprenderá lo siguiente:

- Cómo añadir en un mandato campos nuevos que se correspondan con los parámetros de URL
- Cómo utilizar el método getRequestProperties para permitir que estos campos se llenen con los parámetros de entrada de URL
- Cómo obtener excepciones de parámetro que faltan
- El modo apropiado para pasar parámetros de URL a una vista
- Ver ejemplos de varios casos de prueba en los que los valores de parámetro son incorrectos o se han omitido

Adición de campos nuevos en MyNewControllerCmd

En este paso, creará dos campos nuevos para los parámetros de URL. Éstos se añaden en la interfaz de mandato y en la clase de implementación. Uno de los parámetros de URL es un valor de serie que contiene un nombre de usuario y el segundo es un entero que se utilizará para aceptar un valor de entrada de puntos de bonificación.

Para añadir estos campos nuevos, realice lo siguiente:

1. Conmute a la perspectiva Java.
2. Expanda estos directorios: **WebSphereCommerceServerExtensionsLogic > src > com.ibm.commerce.sample.commands**.
3. Efectúe una doble pulsación en la interfaz **MyNewControllerCmd.java** para ver el código fuente.
4. Elimine la marca de comentario de la Sección 2 para añadir los campos nuevos y los método get correspondientes en la interfaz. Esto inserta el código siguiente en la clase:

```
/// Section 2 ////////////////////////////////////////////////

// set interface methods


 public java.lang.Integer getPoints() ;

 public java.lang.String getUserName() ;

 public void setPoints(java.lang.Integer newPoints) ;

 public void setUserName(java.lang.String newUserName)  ;


/// End of section 2/////////////////////////////////////////
```

5. Guarde los cambios.
6. Efectúe una doble pulsación en la clase **MyNewControllerCmdImpl.java** para ver el código fuente.

7. Elimine la marca de comentario de la Sección 1 de la clase principal para añadir los campos nuevos así como los métodos get y set correspondientes en la clase. Esto inserta el código siguiente en la clase:

```
/// Section 1 ////////////////////////////////////////////

/// create and implement controller command's fields and accessors
/// (setter/getter methods)

 private java.lang.String userName = null;
 private java.lang.Integer points;


 public java.lang.Integer getPoints() {
  return points;
 }

 public java.lang.String getUserName() {
  return userName;
 }

 public void setPoints(java.lang.Integer newPoints) {
  points = newPoints;
 }

 public void setUserName(java.lang.String newUserName) {
  userName = newUserName;
 }

/// End of Section 1 /////////////////////////////////////////
```

8. Guarde los cambios.

Cómo pasar parámetros de URL a la vista

En este paso, incluirá el código para pasar los parámetros de entrada a la plantilla JSP. Esto se realiza estableciendo campos en el bean de datos con los valores de los parámetros de entrada.

Para pasar los parámetros de URL, realice lo siguiente:

1. Efectúe una doble pulsación en **MyNewControllerCmdImpl.java**.
2. En la vista de Esquema, seleccione el método **performExecute**.
3. En el código fuente del método performExecute, elimine la marca de comentario de la Sección 3C. Esto inserta el código siguiente en el método:

```
/// Section 3C/////////////////////////////////////////

 // pass the input information to the databean
 mndb.setUserName(this.getUserName());
 mndb.setPoints(this.getPoints());

/// end of section 3C/////////////////////////////////////////
```

   Este código establece los valores en el objeto de bean de datos para que estén disponibles para la plantilla JSP.

4. Guarde los cambios.

Cómo obtener parámetros que faltan y validar valores

En este paso, modificará el método validateParameters para insertar la lógica de comprobación de errores y de validación de parámetros. Una vez modificado, el código comprueba lo siguiente:

- Si no se proporciona el primer parámetro de entrada, se emite una excepción de "parámetro no encontrado". Esto se debe al hecho de que el primer parámetro de entrada es un parámetro necesario. En este caso, se muestra al cliente la página de error genérico.
- El segundo parámetro de entrada es opcional. De este modo, si no se proporciona el segundo parámetro de entrada, *no* se emite ninguna "excepción de parámetro no encontrado". En cambio, el valor para el segundo parámetro de entrada toma por omisión cero y el cliente no se ve afectado por el error. El proceso continúa.

Para añadir esta comprobación de error, realice lo siguiente:

1. Efectúe una doble pulsación en **MyNewControllerCmdImpl.java**.
2. En la vista de Esquema, seleccione el método **validateParameters**.
3. En el código fuente del método validateParameters, elimine la marca de comentario de la Sección 1. Esto inserta el código siguiente en el método:

```
/// Section 1 ////////////////////////////////////////////////

/// uncomment to check parameters

    final String strMethodName = "validateParameters";

 TypedProperty prop = getRequestProperties();

 /// retrieve required parameters
  try {
  setUserName(prop.getString("input1"));

 } catch (ParameterNotFoundException e) {
  /// the next exception uses _ERR_CMD_MISSING_PARAM ECMessage object
     /// defined in ECMessage class
  throw new ECApplicationException(ECMessage._ERR_CMD_MISSING_PARAM,
        this.getClass().getName(), strMethodName,
   ECMessageHelper.generateMsgParms(e.getParamName()));
 }

 /// retrieve optional Integer
 // set input2 = 0 if no input value
 setPoints(prop.getInteger("input2",0));

/// End of section 1/////////////////////////////////////////////
```

   El fragmento de código anterior comprueba los dos parámetros de entrada. El bloque encabezado por 'try' determina si existe el primer parámetro; si no existe, se emite una excepción. Puesto que el segundo parámetro es opcional, este código establece el valor en cero si dicho parámetro se ha omitido o tiene un valor incorrecto.
4. Guarde los cambios.

Adición de campos nuevos en MyNewDataBean

En este paso, añadirá campos nuevos y los métodos get asociados en el bean de datos MyNewDataBean, para que los parámetros de URL estén disponibles en la plantilla JSP.

Para modificar MyNewDataBean, realice lo siguiente:

1. Efectúe una doble pulsación en **MyNewDataBean.java** para ver el código fuente.
2. Elimine la marca de comentario de la Sección 2 para insertar el código siguiente en la clase:

```
/// Section 2 ////////////////////////////////////////

 private java.lang.String userName = null;
 private java.lang.Integer points;


 public String getUserName() {
  return userName;
 }

 public void setUserName(java.lang.String newUserName) {
  userName = newUserName;
 }


 public Integer getPoints() {
  return points;
 }

 public void setPoints(java.lang.Integer newPoints) {
  points = newPoints;
 }


/// End of Section 2 ////////////////////////////////////
```

3. Guarde los cambios.
4. Compile los cambios de código pulsando el botón derecho del ratón en **WebSphereCommerceServerExtensionsLogic** y seleccionando **Crear proyecto**.

Modificación de MyNewJSPTemplate para visualizar los parámetros de URL

En este paso modificará el archivo MyNewJSPTemplate.jsp para añadir una sección nueva que visualice los parámetros de entrada de URL, realizando lo siguiente:

1. Si los archivos de plantilla JSP aún no están abiertos, realice lo siguiente:
   a. En la perspectiva de Web, cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos y expanda el proyecto Web **Stores**.
   b. Navegue hasta la subcarpeta Web Content*nombre_FashionFlow*.
   c. Resalte los archivos **MyNewJSPTemplate_All.jsp** y **MyNewJSPTemplate.jsp**, pulse con el botón derecho del ratón y seleccione **Abrir con > Page Designer**.
2. Copie la Sección 6 del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esto inserta el texto siguiente en la plantilla JSP:

```
<!-- SECTION 6 -->

<fmt:message key="UserName" bundle="${tutorial}" />
<c:out value="${mndbInstance.userName}"/> <br>

<fmt:message key="Points" bundle="${tutorial}" />
<c:out value="${mndbInstance.points}"/> <br>

<!-- END OF SECTION 6 -->
```

3. Guarde los cambios.

Comprobación de valores de parámetros de URL

El siguiente paso es comprobar si la nueva comprobación de errores funciona correctamente, realizando lo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. Caso 1: El primer caso de prueba será excluir ambos parámetros del URL. Después de que se visualice la página de presentación de la tienda, entre el siguiente URL:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd
   ```

   Puesto que no se pasa ningún parámetro al mandato, se muestra un error genérico de aplicación.

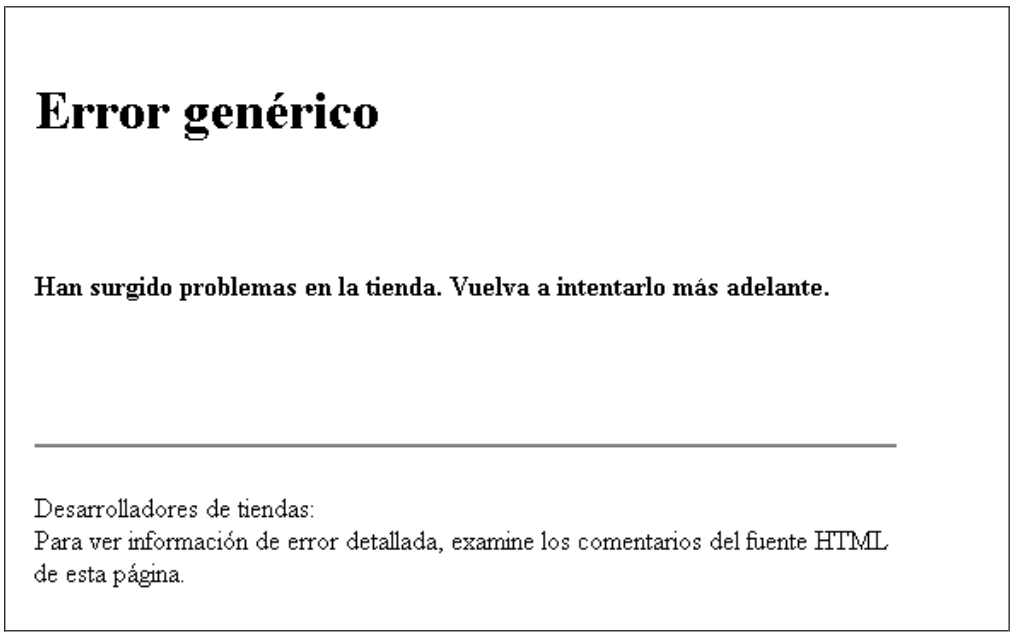


Figura 35.

   La consola de WebSphere Studio Application Developer muestra información similar a la siguiente:

   ```
   indicaciónHora 6730e546 CommerceSrvr  E
   com.ibm.commerce.sample.commands.MyNewControllerCmdImpl
   validateParameters CMN0206E Por favor, compruebe todos los campos.
   "input1" es un campo obligatorio.
   ```

5. Caso 2: El siguiente caso de prueba consiste en utilizar un primer parámetro válido, pero omitir el segundo parámetro. En este caso, se espera que no se detecte ningún error puesto que, por omisión, se utilizará un valor de cero para el segundo parámetro omitido. Entre el siguiente URL:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?input1=abc
   ```

   El resultado de este mandato es que se visualiza la página MyNewJSPTemplate y se muestra un valor de cero para input2.

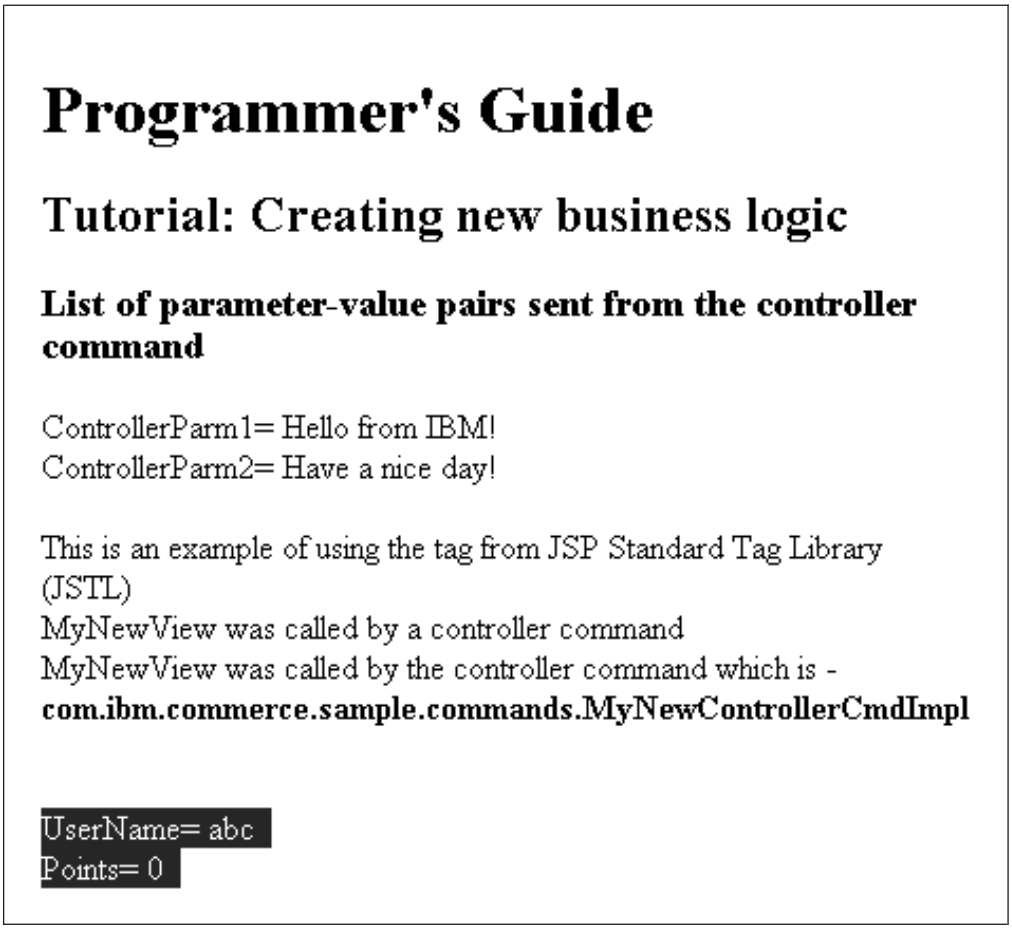


Figura 36.

6. Caso 3: En este caso de prueba, se proporciona un parámetro válido para el primer parámetro de entrada y se proporciona un parámetro no válido para el segundo parámetro (se utiliza una serie en lugar de un entero). De forma similar al último caso, no deberá ver ningún error, puesto que la línea de manejo de errores cambia el segundo parámetro de entrada por un cero. Entre el siguiente URL:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc&input2=abc
   ```

   El resultado de este mandato es que se visualiza la página MyNewJSPTemplate y se muestra un valor de cero para input2.

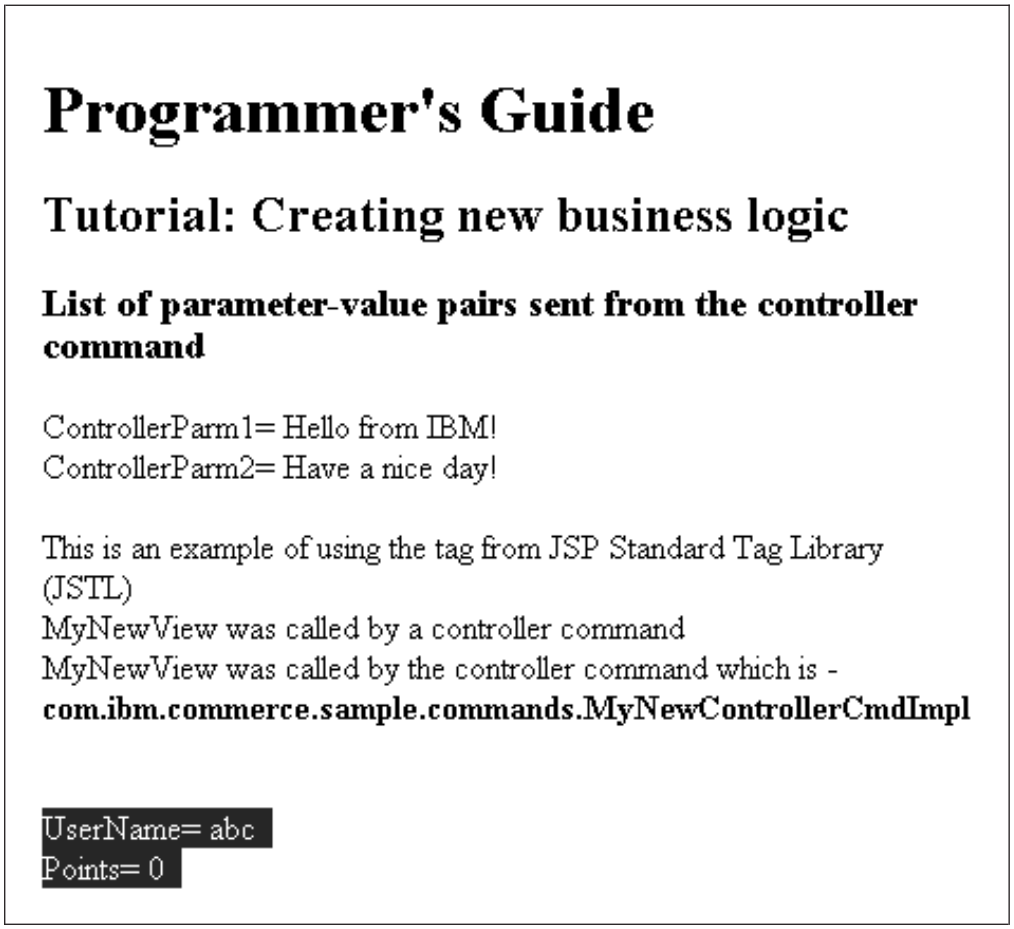


Figura 37.

7. Caso 4: En este caso, se utilizan parámetros válidos para ambos parámetros de entrada de URL. Entre el siguiente URL:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc&input2=1000
   ```

   El resultado de este mandato es que se visualiza la página MyNewJSPTemplate y se visualizan 1000 puntos para el usuario.

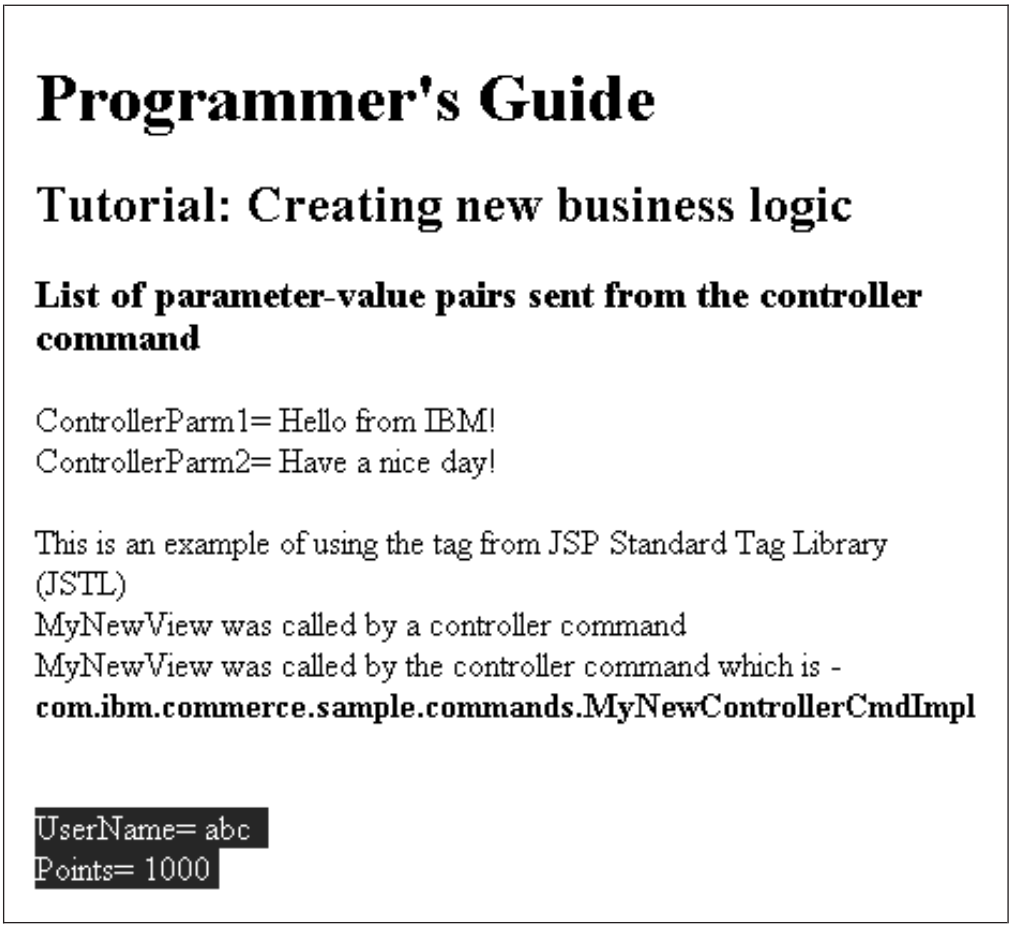


Figura 38.

Creación de un mandato de tarea nuevo

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. Normalmente un proceso de negocio se puede dividir en tareas más 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.

Actualmente, MyNewControllerCmdImpl no llama a ningún mandato de tarea. Esta sección se divide en dos pasos. En el primer paso, creará el nuevo mandato de tarea. En el segundo paso, modificará el método performExecute del mandato de controlador para llamar al nuevo mandato de tarea.

En este paso, creará una nueva interfaz de mandato de tarea y su clase de implementación asociada. Inicialmente, el nuevo mandato de tarea tiene poca actividad, excepto manejar los parámetros de vista. Sólo tiene campos para defaultCommandClassName, los parámetros de URL y el valor actual de puntos de bonificación. Tiene un método para obtener el valor actual de los puntos de bonificación.

En este paso de la guía de aprendizaje, obtendrá información sobre lo siguiente:

- Cómo crear una nueva interfaz de mandato de tarea y su clase de implementación asociada
- La cantidad mínima de código que se necesita en un mandato de tarea
- Cómo añadir campos y métodos en el mandato de tarea

Creación de MyNewTaskCmd

Este paso le 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, tendrá un nuevo mandato de tarea, denominado MyNewTaskCmd. Este mandato lo utiliza la tienda que está basada en el ejemplo FashionFlow.

Para crear MyNewTaskCmd, realice lo siguiente:

1. Conmute a la perspectiva Java y seleccione el proyecto **WebSphereCommerceServerExtensionsLogic**.
2. Expanda el directorio **src** y, a continuación, el directorio **com.ibm.commerce.sample.commands**.
3. Efectúe una doble pulsación en la interfaz **MyNewTaskCmd.java** para ver el código fuente.
4. En el código fuente de esta interfaz, elimine la marca de comentario de la Sección 1 para crear un campo que especifique la clase de implementación por omisión que la interfaz debe utilizar. Esto inserta el siguiente código en la interfaz:

   ```
   /// Section 1 ////////////////////////////////////////////

   // set default command implement class

    static final String defaultCommandClassName=
     "com.ibm.commerce.sample.commands.MyNewTaskCmdImpl";


   /// End of section 1/////////////////////////////////////////
   ```

   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.
5. A continuación, elimine la marca de comentario de la Sección 2 para crear los métodos get y set que se utilizarán en la clase de implementación MyNewTaskCmdImpl. Estos métodos son para los campos correspondientes a los siguientes tipos de información:
   - El ID de usuario de un cliente.
   - Un valor de puntos de bonificación
   - Un mensaje de saludo

Al eliminar la marca de comentario de la Sección 2, se inserta el código siguiente en la interfaz:

```
/// Section 2 ////////////////////////////////////////////////
// set interface methods

 public void setInputUserName(java.lang.String inputUserName);
 public void setInputPoints(Integer inputPoints);
 public void setGreetings(java.lang.String greeting);

 public java.lang.String getInputUserName();
 public java.lang.Integer getInputPoints();
 public java.lang.String getGreetings();


/// End of section 2/////////////////////////////////////////
```

6. Guarde los cambios.
7. Efectúe una doble pulsación en la clase de implementación **MyNewTaskCmdImpl.java** para ver el código fuente.
8. Elimine la marca de comentario de las Secciones 1A y 1B para crear campos y sus métodos get y set correspondientes en la clase de implementación. Esto inserta el código siguiente en la clase:

```
//// Section 1A ////////////////////////////////////////////

 private java.lang.String inputUserName;
 private java.lang.String greetings;
 private java.lang.Integer inputPoints;


////End of Section 1A ////////////////////////////////////////


//// Section 1B ////////////////////////////////////////////

 public void setInputUserName(java.lang.String newInputUserName) {
  inputUserName = newInputUserName;
 }

 public void setInputPoints(Integer newInputPoints) {
  inputPoints = newInputPoints;
 }

 public void setGreetings(java.lang.String newGreetings) {
  greetings = newGreetings;
 }

 public java.lang.String getInputUserName() {
  return inputUserName;
 }

 public Integer getInputPoints() {
  return inputPoints;
 }

 public java.lang.String getGreetings() {
  return greetings;
 }


////End of Section 1B ////////////////////////////////////////
```

Guarde el trabajo.

9. En la vista de Esquema, seleccione el método **performExecute** de la clase MyNewTaskCmdImpl.

10. En el código fuente de este método performExecute, elimine la marca de comentario de la Sección 1 para insertar el código siguiente en el método:

```
/// Section 1 ////////////////////////////////////////////////

/// modify the greetings and see it in the NVP list

 setGreetings( "Hello ! " + getInputUserName() );

/// End of section 1 ////////////////////////////////////////
```

    Esto actualiza el valor de saludos (greetings). Entonces el valor de saludos queda disponible para otros objetos a través del método getGreetings(). Se añadirá a la lista pareja nombre-valor (NVP).
11. Guarde el trabajo.

Llamada del mandato de tarea

Una vez ha creado el mandato de tarea, tiene que llamarlo desde el mandato de controlador. Los pasos siguientes muestran cómo modificar el mandato de controlador de este modo:

1. En la perspectiva Java, efectúe una doble pulsación en la clase **MyNewControllerCmdImpl.java**
2. En la vista de Esquema, seleccione el método **performExecute**.
3. En el código fuente para el método performExecute, navegue hasta el Área 4.
4. Elimine la marca de comentario de las Secciones 4A, 4B y 4C para incluir el código siguiente en el método:

```
/// Section 4A
    /// see how the controller command call a task command

 MyNewTaskCmd cmd = null;

  try {

  cmd = (MyNewTaskCmd) CommandFactory.createCommand(
         "com.ibm.commerce.sample.commands.MyNewTaskCmd", getStoreId());

        // this is required for all commands
          cmd.setCommandContext(getCommandContext());

  /// set input parameters to task command
  cmd.setInputUserName(getUserName());
  cmd.setInputPoints(getPoints()); // cambiar a entero

/// End Section 4A ////////////////////////////////////////

/// Section 4B ///////////////////////////////////

  /// invoke the command's performExecute method
   cmd.execute();

  /// retrieve output parameter from task command, then put it to
     /// response properties
  rspProp.put("taskOutputGreetings", cmd.getGreetings());

/// End Section 4B ///////////////////////////////////

/// Start Section 4C ///////////////////////////////////
 } catch (ECException ex) {
  /// throw the exception as is
  throw (ECException) ex;
```

```
 }

/// End Section 4C ////////////////////////////////////
```

La Sección 4A crea el nuevo objeto de mandato de tarea utilizando la fábrica de mandatos. Entonces establece el contexto de mandato y los parámetros de entrada del mandato de tarea. La Sección 4B llama al método validateParameters para el control de acceso, antes de invocar el método de ejecución (execute) del mandato de tarea. Entonces recupera el valor de saludos del mandato de tarea. La Sección 4C es un simple bloque de captura para las excepciones.

5. Guarde los cambios.
6. Compile los cambios de código pulsando el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccionando **Crear proyecto**.

Modificación de MyNewJSPTemplate para añadir el mensaje de saludo

En este paso modificará el archivo MyNewJSPTemplate.jsp para añadir una sección nueva que visualiza el mensaje de saludo, realizando lo siguiente:

1. Si los archivos de plantilla JSP aún no están abiertos, realice lo siguiente:
   a. En la perspectiva de Web, cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos y expanda el proyecto Web **Stores**.
   b. Navegue hasta la subcarpeta Web Content*nombre_FashionFlow*.
   c. Resalte los archivos **MyNewJSPTemplate_All.jsp** y **MyNewJSPTemplate.jsp**, pulse con el botón derecho del ratón y seleccione **Abrir con > Page Designer**.
2. Copie la Sección 7 del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esto inserta el texto siguiente en la plantilla JSP:

```
<!-- SECTION 7 -->

<fmt:message key="Greeting" bundle="${tutorial}" />
<c:out value="${taskOutputGreetings}"/> <br /> <br />

<!-- END OF SECTION 7 -->
```

3. Guarde los cambios.

Comprobación de MyNewTaskCmd

El siguiente paso es comprobar si el nuevo mandato de tarea funciona correctamente, realizando lo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. A continuación, entre el URL siguiente:

```
http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
     input1=abc&input2=1000
```

Se visualiza MyNewJSPTemplate. Ésta incluye el mensaje de saludo creado por el mandato de tarea.

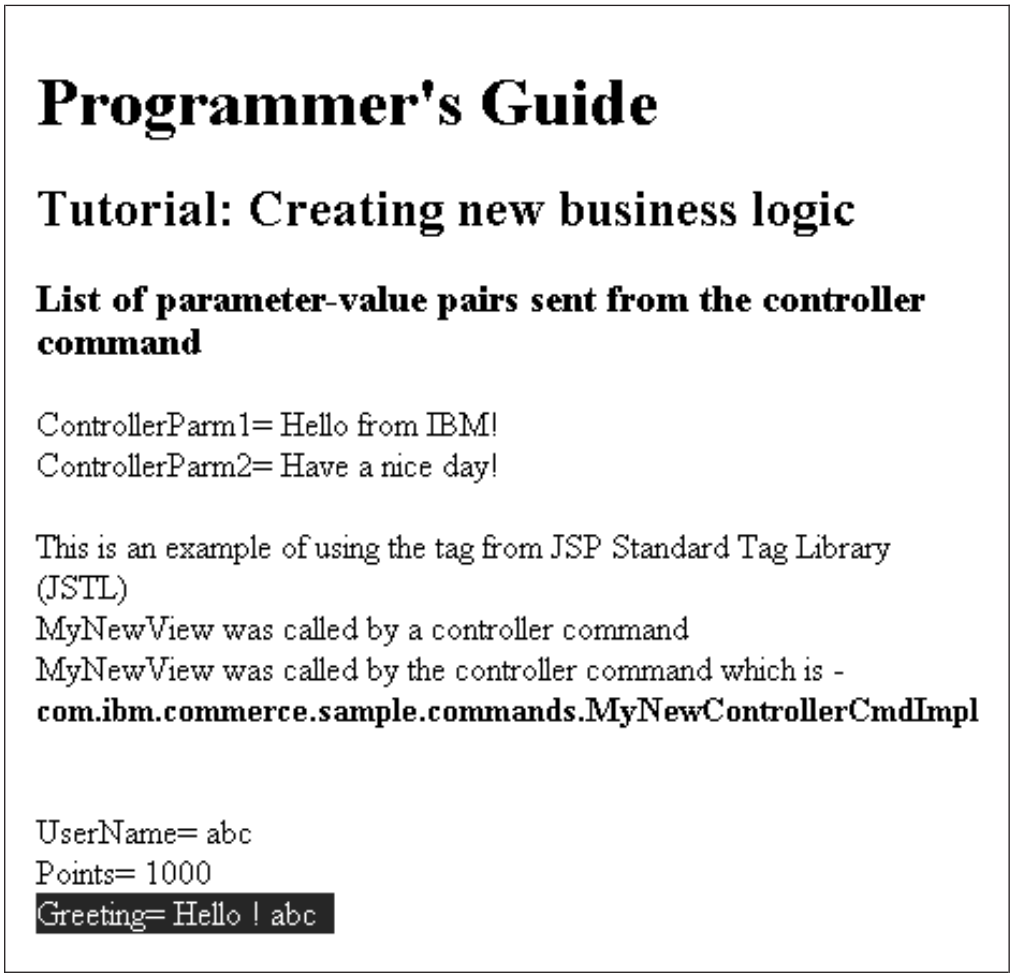


Figura 39.

Modificación de MyNewTaskCmd

En este paso de la guía de aprendizaje, modificará MyNewTaskCmd para determinar si el nombre de usuario incluido en el URL es el de un usuario registrado. MyNewDataBean también se modifica para manejar campos del mandato de tarea (por ejemplo el nombre de usuario). Correspondientemente, se modifica MyNewJSPTemplate para visualizar si el usuario está registrado o no lo está.

En esta sección de la guía de aprendizaje, aprenderá lo siguiente:

- Cómo utilizar los parámetros de URL y acceder a la información de WebSphere Commerce existente desde dentro del propio código personalizado

Modificación de MyNewControllerCmdImpl para crear un objeto para el mandato de tarea

A fin de promover el uso eficiente de los objetos, el mandato de controlador crea una variable de instancia de objeto UserRegistryAccessBean. En consecuencia, este objeto también está disponible para el mandato de tarea. Si se sigue esta propuesta, el mandato de tarea no necesita crear una instancia independiente del objeto. Este

objeto UserRegistryAccessBean se utiliza posteriormente (en el mandato de tarea) para determinar si el comprador es un usuario registrado.

Para modificar MyNewControllerCmdImpl, realice lo siguiente:

1. En la perspectiva Java, efectúe una doble pulsación en la clase **MyNewControllerCmdImpl.java** para ver el código fuente.
2. En el cuerpo principal de esta clase, elimine la marca de comentario de la Sección 2, para insertar el código siguiente en la clase:

```
/// Section 2 ////////////////////////////////////////////
/// create a user registry accessbean resource instance variable

    private UserRegistryAccessBean rrb = null;

/// End of Section 2 /////////////////////////////////////
```

3. En la vista de Esquema, seleccione el método **performExecute**.
4. En el código fuente para el método performExecute, elimina la marca de comentario de las Secciones 4D y 4F para pasar la variable de instancia al mandato de tarea y luego dejar disponible en las propiedades de respuesta el ID de usuario devuelto. Esto inserta el código siguiente en el método:

```
// Section 4D ///////////////////////////////////
/// pass rrb instance variable to the task command

  cmd.setUserRegistryAccessBean(rrb);

// End of section 4D ////////////////////////////


// Section 4F ///////////////////////////////////
///using access bean to get information from database
  if (cmd.getFoundUserId() != null) {
   rspProp.put("taskOutputUserId", cmd.getFoundUserId());
  }
// End of section 4F ////////////////////////////
```

   Recibirá un error que indica que algunos de los métodos no están definidos, pero esto se solucionará en el siguiente paso cuando modifique el mandato de tarea.
5. Guarde el trabajo.

Modificación del mandato de tarea nuevo para la validación de nombre de usuario

A continuación, deberá modificar el nuevo mandato de tarea para validar si la entrada de nombre de usuario del URL es la de un usuario registrado, del modo siguiente:

1. Efectúe una doble pulsación en la interfaz **MyNewTaskCmd.java** para ver el código fuente.
2. Elimine la marca de comentario de la Sección 3, para insertar el código siguiente en la interfaz:

```
/// Section 3 ////////////////////////////////////////////

 public void setFoundUserId(java.lang.String inputUserId);
 public java.lang.String getFoundUserId();

 public void setUserRegistryAccessBean(UserRegistryAccessBean rrb);

/// End of section 3/////////////////////////////////////
```

3. Guarde el trabajo.

4. Efectúe una doble pulsación en la clase **MyNewTaskCmdImpl.java** para ver el código fuente. Elimine la marca de comentario de la sección 1 de importación para insertar las dos sentencias import siguientes en el código:

```
/// Import section 1 ////////////////////////////////////
import com.ibm.commerce.user.objects.*;
import com.ibm.commerce.sample.databeans.*;
///  End of Import section 1 ////////////////////////////
```

5. Elimine la marca de comentario de las Secciones 2A y 2B para crear los nuevos campos y los métodos get y set que corresponden a los métodos añadidos en la interfaz. Esto inserta el código siguiente en la clase:

```
//// Section 2A ////////////////////////////////////////////

 private java.lang.String foundUserId = null;

 private UserRegistryAccessBean rrb = null;

////End of Section 2A ////////////////////////////////////////

//// Section 2B ////////////////////////////////////////////

 public void setUserRegistryAccessBean(UserRegistryAccessBean newRRB) {
  rrb = newRRB;
 }

 public void setFoundUserId(java.lang.String newFoundUserId) {
  foundUserId = newFoundUserId;
 }

 public java.lang.String getFoundUserId() {
  return foundUserId;
 }


/// End of section 2B ////////////////////////////////////////
```

6. En la vista de Esquema, seleccione el método **validateParameters** y examine el código fuente. Elimine la marca de comentario de la Sección 1, para insertar el código siguiente en el método:

```
// section 1 ////////////////////////////////////////////////

// use UserRegistryAccessBean to check user Id

  try {

   if (rrb!=null){
    setFoundUserId(rrb.getUserId());
   } else {
     rrb =new UserRegistryAccessBean();
     rrb=rrb.findByUserLogonId(getInputUserName());
     setFoundUserId(rrb.getUserId());
   }

 }  catch (javax.ejb.FinderException e) {
    return;

 } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
 } catch (javax.naming.NamingException e) {
   throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(), "validateParameters");
 } catch (javax.ejb.CreateException e) {
   throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(), "validateParameters");
```

```
    }

// end of section 1 ////////////////////////////////////////////
```

7. Guarde el trabajo.
8. Compile los cambios de código pulsando el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccionando **Crear proyecto**.

Modificar MyNewJSPTemplate para la validación de nombre de usuario

La plantilla JSP actual debe modificarse para visualizar la información de validación de nombre de usuario. Para modificar este archivo, realice lo siguiente:

1. Conmute a la perspectiva de Web.
2. Abra los archivos **MyNewJSPTemplate_All.jsp** y **MyNewJSPTemplate.jsp**.
3. Copie la Sección 8 del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esto inserta el texto siguiente en la plantilla JSP:

```
<!-- SECTION 8 -->

<c:if test="${!empty taskOutputUserId}">
   <fmt:message key="UserId" bundle="${tutorial}" />
   <c:out value="${taskOutputUserId}"/> <br />
   <fmt:message key="FirstInput" bundle="${tutorial}" />
   <b><c:out value="${userName}"/></b>
   <fmt:message key="RegisteredUser" bundle="${tutorial}" /> <br />
   <fmt:message key="ReferenceNumber" bundle="${tutorial}" />
   <b><c:out value="${taskOutputUserId}"/></b> <br /> <br />
</c:if>

<c:if test="${empty taskOutputUserId}">
   <fmt:message key="FirstInput" bundle="${tutorial}" />
   <b><c:out value="${userName}"/></b>
   <fmt:message key="NotRegisteredUser" bundle="${tutorial}" /> <br />
</c:if>


<!-- END OF SECTION 8 -->
```

4. Guarde los cambios realizados en el archivo MyNewJSPTemplate.jsp.

Comprobación de la validación de nombre de usuario

Para comprobar la lógica de validación de nombre de usuario, realice lo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. Cree un nuevo usuario registrado, realizando lo siguiente:
   a. Pulse **Regístrese**.
   b. Pulse **Regístrese** otra vez para crear un cliente nuevo.
   c. En el formulario de registro, entre los valores apropiados en todos los campos obligatorios. Por ejemplo, en el campo de correo electrónico, entre `tester@mycompany`. Tome nota del valor para la dirección de correo electrónico: _____.
   d. Una vez entrados los valores, pulse **Someter**.

5. A continuación, entre un nombre de usuario válido como valor para input1. Entre el siguiente URL:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=nuevo_correo_electrónico&input2=1000
   ```

   donde *nuevo_correo_electrónico* es la dirección de correo electrónico para el usuario creado en el paso 4. Se visualiza MyNewJSPTemplate. Ahora deberá indicar que el valor para input1 es un nombre de usuario válido.

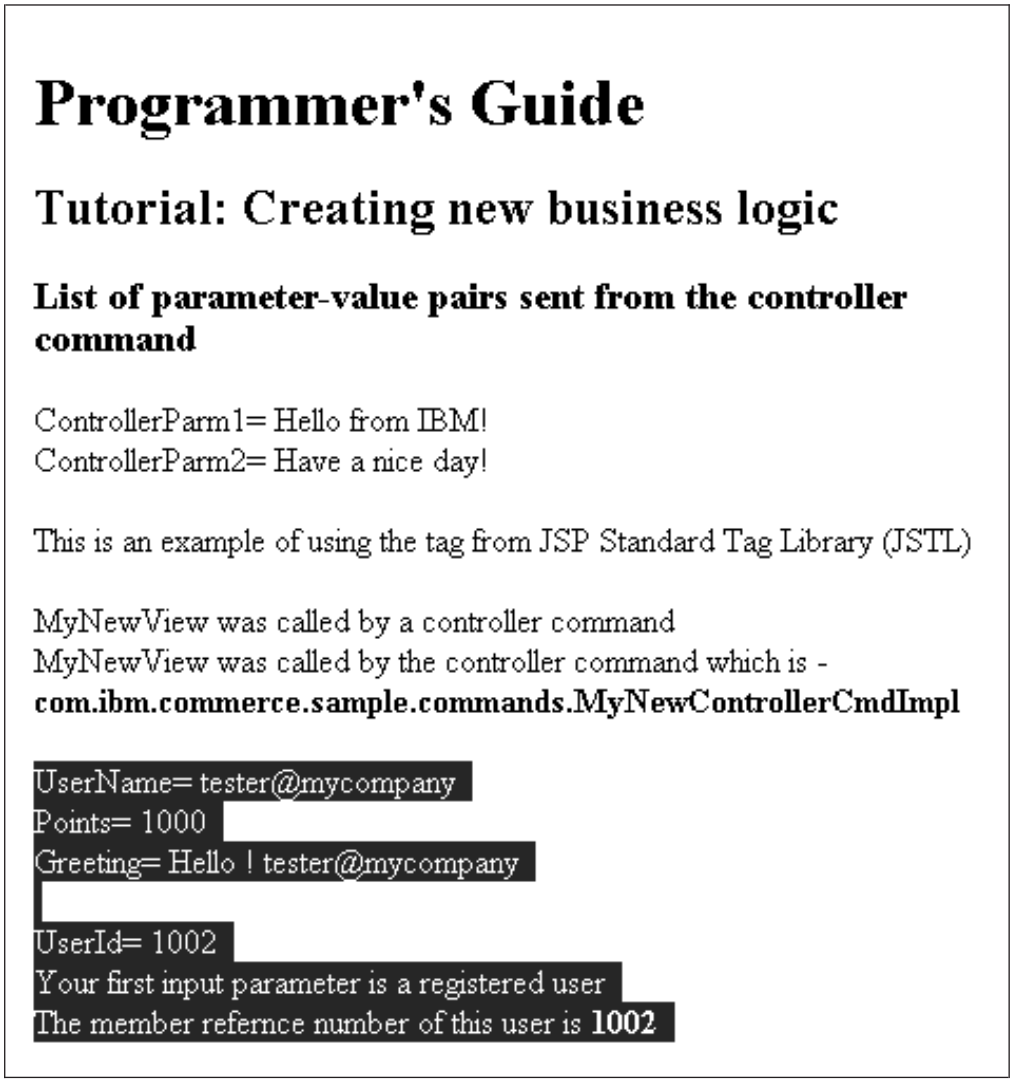


Figura 40.

6. A continuación, entre el URL siguiente en el que el valor para el nombre de usuario no es válido:

   ```
   http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
        input1=abc&input2=1000
   ```

   Se visualiza una excepción genérica. Si visualiza el código fuente de la página, encontrará que se ha producido una excepción _ERR_FINDER_EXCEPTION. Esto se debe al hecho de que el nombre de usuario proporcionado no corresponde a un usuario registrado.

Creación de un bean de entidad nuevo

Esta sección describe cómo crear un nuevo bean de entidad. En este escenario de ejemplo, un negocio requiere 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, deberá crear un bean de entidad para acceder a los datos.

Creación de la tabla XBONUS

Como preparación para la creación del bean de entidad, primero debe crear la nueva tabla de base de datos. La tabla que se debe crear se denomina XBONUS.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para crear la tabla:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**) y pulse la pestaña **Scripts**.
2. En la ventana Script, entre lo siguiente:

   ```
   connect to BDdesarrollo user usuariobd using contraseñabd;
   create table XBONUS (MEMBERID BIGINT NOT NULL,
       BONUSPOINT INTEGER NOT NULL,
              constraint p_xbonus primary key (MEMBERID),
       constraint f_xbonus foreign key (MEMBERID)
       references users (users_id) on delete cascade)
   ```

   donde

   - *BDdesarrollo* es el nombre de la base de datos de desarrollo
   - *usuariobd* es el usuario de base de datos
   - *contraseñabd* es la contraseña para el usuario de base de datos

   Pulse el icono Ejecutar.
   Verá un mensaje que indica que la sentencia SQL se ha completado satisfactoriamente.

Oracle Si está utilizando una base de datos Oracle, realice 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 SQL:

   ```
   create table XBONUS (MEMBERID NUMBER NOT NULL,
       BONUSPOINT INTEGER NOT NULL,
              constraint p_xbonus primary key (MEMBERID),
       constraint f_xbonus foreign key (MEMBERID)
       references users (users_id) on delete cascade);
   ```

   y pulse Intro para ejecutar la sentencia SQL. La tabla XBONUS ya se ha creado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

y pulse Intro para ejecutar la sentencia SQL.

Creación del bean de entidad BonusBean

Una vez que se haya creado la tabla, estará preparado para empezar a crear el nuevo bean de entidad. Los pasos siguientes utilizan WebSphere Studio Application Developer para crear este bean.

A continuación, creará el nuevo bean Bonus, realizando lo siguiente:

1. En WebSphere Studio Application Developer, conmute a la perspectiva de J2EE.
2. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
3. Pulse con el botón derecho del ratón en el módulo **WebSphereCommerceServerExtensionsData** y seleccione **Nuevo > Bean enterprise**.
   Se abrirá el asistente para la Creación de bean enterprise.
4. En la lista desplegable **Proyecto EJB** seleccione **WebSphereCommerceServerExtensionsData** y pulse **Siguiente**.
5. En la ventana Crear un bean enterprise, realice lo siguiente:
   a. Seleccione **Bean de entidad con campos CMP (Persistencia gestionada por contenedor)**
   b. En el campo **Nombre de bean**, entre `Bonus`.
   c. En el campo **Carpeta fuente**, deje el valor por omisión que está especificado (ejbModule).
   d. En el campo **Paquete por omisión**, entre `com.ibm.commerce.extension.objects`.
   e. Pulse **Siguiente**.
6. En la ventana Detalles del bean enterprise, realice lo siguiente:
   a. Pulse **Añadir** para añadir atributos CMP nuevos para las columnas MEMBERID y BONUSPOINT de la tabla BONUS.
      Se abrirá la ventana Crear atributo CMP. En esta ventana, realice lo siguiente:
      1) En el campo **Nombre**, entre `memberId`.
      2) En el campo **Tipo**, entre `java.lang.Long`.

         Nota: Deberá utilizar el tipo de datos *java.lang.Long*, no el tipo de datos *long*.
      3) Marque el recuadro de selección **Campo de clave**.
      4) Pulse **Aplicar**.
      5) En el campo **Nombre**, entre `bonusPoint`.
      6) En el campo **Tipo**, entre `java.lang.Integer`.

         Nota: Deberá utilizar el tipo de datos *java.lang.Integer*, no el tipo de datos *integer*.
      7) Marque el recuadro de selección **Acceso con método get y set**.
      8) Elimine la marca del recuadro de selección **Promocionar métodos get y set a interfaz remota**. El recuadro de selección **Hacer que el método get sea de sólo lectura** dejará de estar disponible.
      9) Pulse **Aplicar**.
      10) Pulse **Cerrar** para cerrar esta ventana.
   b. Elimine la marca del recuadro de selección **Utilizar el tipo de atributo de clave única para la clase de clave** y, a continuación, pulse **Siguiente**.

7. En la ventana Detalles de clase Java EJB, realice lo siguiente:
   a. Para seleccionar la superclase del bean, pulse **Examinar**.
   Se abrirá la ventana Selección de tipo.
   b. En el campo **Seleccionar una clase utilizando: (cualquiera)**, entre `ECEntityBean` y pulse **Aceptar**. Esto selecciona com.ibm.commerce.base.objects.ECEntityBean como superclase.
   c. Especifique las interfaces que la interfaz remota debe ampliar pulsando **Añadir**. Se abrirá la ventana Selección de tipo.
   d. En el campo **Seleccionar una clase utilizando: (cualquiera)**, entre `Protectable` y pulse **Aceptar**. Esto selecciona com.ibm.commerce.security.Protectable. Esta interfaz es necesaria para proteger el nuevo recurso bajo el control de acceso.
   e. Pulse **Finalizar**.

Establezca el nivel de aislamiento para el nuevo bean, realizando lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Efectúe una doble pulsación en el proyecto **WebSphereCommerceServerExtensionsData** para abrirlo con el Editor de descriptor de despliegue. Como procedimiento alternativo para abrir este archivo en este editor, puede realizar lo siguiente:
   a. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, expanda **WebSphereCommerceServerExtensionsData > ejbModule > META-INF**.
   b. Pulse con el botón derecho del ratón en **ejb-jar.xml** y seleccione **Abrir con > Editor del Descriptor de despliegue**)
3. Pulse la pestaña **Acceso**.
4. Pulse **Añadir** junto al recuadro de texto Nivel de aislamiento.
   Se abrirá la ventana Añadir nivel de aislamiento.
5. DB2 Seleccione **Lectura repetible** y, a continuación, pulse **Siguiente**.
   Oracle Seleccione **Comprometido para lectura** y, a continuación, pulse **Siguiente**.
6. En la lista **Beans encontrados**, seleccione el bean **Bonus** y, a continuación, pulse **Siguiente**.
7. En la lista **Métodos encontrados**, seleccione **Bonus** para seleccionar la totalidad de los métodos y pulse **Finalizar**.
8. Guarde el trabajo (Control + S) y mantenga el editor abierto.

A continuación, establezca la identidad de seguridad del bean, realizando lo siguiente:

1. En el editor de Descriptor de despliegue, asegúrese de tener seleccionada la pestaña Acceso.
2. Pulse **Añadir** junto al recuadro de texto Identidad de seguridad.
   Se abrirá la ventana Añadir identidad de seguridad.
3. Seleccione **Utilizar identidad de servidor EJB** y, a continuación, pulse **Siguiente**.
4. En la lista **Beans encontrados**, seleccione el bean **Bonus** y, a continuación, pulse **Siguiente**.
5. En la lista **Métodos encontrados**, seleccione **Bonus** para seleccionar la totalidad de los métodos y pulse **Finalizar**.
6. Guarde el trabajo (Control + S) y mantenga abierto el editor.

A continuación, establezca el rol de seguridad para los métodos del bean, realizando lo siguiente:

1. En el editor del Descriptor de despliegue, seleccione la pestaña Descriptor de ensamblado.
2. En la sección Permisos de método, pulse **Añadir**.
3. Seleccione **WCSecurityRole** como el rol de seguridad y pulse **Siguiente**.
4. En la lista de beans encontrados, seleccione **Bonus** y pulse **Siguiente**.
5. En la página Elementos de método, pulse **Aplicar a todo** y, a continuación, pulse **Finalizar**.
6. Guarde el trabajo (Control + S) y cierre el editor de descriptor de despliegue.

El siguiente paso es eliminar algunos de los campos y métodos relacionados con el contexto de entidad que WebSphere Studio Application Developer genera. La razón por la que es necesario suprimir estos campos es que la clase base ECEntityBean proporciona su propia implementación de estos métodos. Para suprimir los campos y métodos de contexto de entidad generados, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda el proyecto **WebSphereCommerceServerExtensionsData**.
2. Expanda el bean **Bonus** y, a continuación, efectúe una doble pulsación en la clase **BonusBean**.
3. En la vista de Esquema, realice lo siguiente:
   a. Pulse con el botón derecho del ratón en el campo **myEntityCtx** y seleccione **Suprimir**.
   b. Pulse con el botón derecho del ratón en el método **getEntityContext()** y seleccione **Suprimir**.
   c. Pulse con el botón derecho del ratón en el método **setEntityContext(EntityContext)** y seleccione **Suprimir**.
   d. Pulse con el botón derecho del ratón en el método **unsetEntityContext()** y seleccione **Suprimir**.
4. Guarde el trabajo (Control+S). Mantenga abierta la clase BonusBean.

A continuación, añada un nuevo método getMemberId al bean enterprise, realizando lo siguiente:

1. Visualice el código fuente de la clase **BonusBean**.
2. Añada el código siguiente al final de esta clase (aún dentro de la clase):

```
public java.lang.Long getMemberId() {
   return memberId;
}
```

3. Deberá añadir el nuevo método en la interfaz remota, realizando lo siguiente:
   a. En la vista de Esquema, pulse con el botón derecho del ratón en el método **getMemberId** y seleccione **Bean enterprise > Promocionar a interfaz remota**. Una vez que esto se haya completado, se visualizará un pequeño icono R junto al método, indicando que éste se ha promocionado a la interfaz remota.
4. Guarde el trabajo.
5. Cierre el editor de BonusBean.

A continuación, añada nuevos métodos FinderHelper en la interfaz BonusHome, realizando lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.

2. Efectúe una doble pulsación en el proyecto **WebSphereCommerceServerExtensionsData** para abrir el Editor del descriptor de despliegue de EJB.
3. Pulse la pestaña **Beans**.
4. En el panel Beans, seleccione el bean **Bonus** y, a continuación, en el panel de la derecha, desplácese hacia abajo y expanda **WebSphere Extensiones**.
5. Pulse **Añadir** junto al recuadro de texto **Buscadores**.
   Se abrirá la ventana Añadir descriptor de buscador.
6. Seleccione **Nuevo** y, a continuación, en el campo **Nombre**, entre `findByMemberId`.
7. Pulse **Añadir** junto al recuadro de texto **Parámetros** y, a continuación, realice lo siguiente:
   a. En el campo **Nombre**, entre `memberId`.
   b. En el campo **Tipo**, entre `java.lang.Long`.
   c. Pulse **Aceptar**.
8. En la lista desplegable **Tipo de retorno**, seleccione **com.ibm.commerce.extension.objects.Bonus** y, a continuación, pulse **Siguiente**.
9. En la lista desplegable **Tipo de buscador**, seleccione **WhereClauseFinderDescriptor**.
10. En el campo **Sentencia de buscador**, entre `T1.MEMBERID = ?` y, a continuación, pulse **Finalizar**.
11. Guarde el trabajo y, a continuación, cierre el editor del Descriptor de despliegue de EJB.

A continuación, añada un nuevo método ejbCreate en el bean Bonus, realizando lo siguiente:

1. En la vista de Jerarquía J2EE, efectúe una doble pulsación en la clase **BonusBean** para abrirla y ver el código fuente.
2. Cree un nuevo método ejbCreate(Long, Integer), añadiendo el código siguiente en la clase:

```
public com.ibm.commerce.extension.objects.BonusKey ejbCreate(
   java.lang.Long memberId,java.lang.Integer bonusPoint)
   throws javax.ejb.CreateException {
      _initLinks();
      this.memberId=memberId;
      this.bonusPoint=bonusPoint;
      return null;
}
```

3. Guarde los cambios de código.
4. Deberá añadir el nuevo método ejbCreate(Long, Integer) en la interfaz inicial. Esto hará que el método esté disponible en el bean de acceso generado. Para añadir el método a la interfaz inicial, realice lo siguiente:
   a. En la vista de Esquema, pulse con el botón derecho del ratón en el método **ejbCreate(Long, Integer)** y seleccione **Bean enterprise > Promocionar a interfaz inicial**.

A continuación, cree un método ejbPostCreate(Long, Integer) nuevo de forma que tenga los mismos parámetros que el método ejbCreate(Long, Integer), realizando lo siguiente:

1. Efectúe una doble pulsación en la clase **BonusBean** para abrirla y ver el código fuente.

2. Cree un método ejbPostCreate(Long, Integer) nuevo, añadiendo el código siguiente en la clase:

```
public void ejbPostCreate(java.lang.Long memberId,
   java.lang.Integer bonusPoint)
   throws javax.ejb.CreateException
   {
   }
```

3. Guarde los cambios de código.

Los pasos siguientes añaden métodos nuevos en el bean Bonus para que el sistema de control de acceso de WebSphere Commerce pueda protegerlo. Los métodos getOwner y fulfills se añaden en el bean, realizando lo siguiente:

1. Efectúe una doble pulsación en la clase **BonusBean** para abrirla y ver el código fuente.
2. Añada el nuevo método getOwner en la clase BonusBean, añadiendo el código siguiente al final de la clase:

```
public java.lang.Long getOwner()
   throws java.lang.Exception {
      return getMemberId();
   }
```

3. Guarde el trabajo.
4. Añada el nuevo método fulfills, añadiendo el código siguiente al final de la clase:

```
public boolean fulfills(Long member, String relationship)
   throws java.lang.Exception {
      if (relationship.equalsIgnoreCase("creator"))
      {
         return member.equals(getMemberId());
      }
      return false;
   }
```

5. Guarde el trabajo y cierre el editor de bean de bonificación.

El paso siguiente consiste en correlacionar la tabla XBONUS con el bean de entidad BonusBean. Se utiliza la correlación de encuentro a medio camino. Para crear la correlación, haga lo siguiente: Business Professional

1. En la vista de Jerarquía J2EE, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Correlación entre EJB y RDB**.
   Se abrirá la ventana Correlación entre EJB y RDB.
2. Seleccione **Encuentro a medio camino** y pulse **Siguiente**.
3. En la ventana Conexión de base de datos, realice lo siguiente:
   a. En el campo **Nombre de conexión**, entre `WebSphereCommerceServerExtensionsData`
   b. En el campo **Base de datos**, entre el nombre de la base de datos de desarrollo.
   c. En el campo **ID de usuario**, entre el ID de usuario de base de datos.
   d. En el campo **Contraseña**, entre la contraseña del usuario de base de datos.
   e. En la lista desplegable **Tipo de proveedor de base de datos**, seleccione el tipo de proveedor de base de datos para la base de datos de desarrollo.
      - DB2 DB2 Universal Database 8.1
      - Oracle Oracle 9i

f. Oracle En el campo **Sistema principal**, entre el nombre de sistema principal totalmente calificado del servidor de bases de datos. Por ejemplo, entre `dbserver.yourcompany.com`

g. Oracle En el campo Ubicación de clase, entre la ubicación del archivo classes12.zip. Por ejemplo, entre `D:\oracle\ora92\jdbc\lib\classes12.zip`

h. Pulse **Siguiente**. Una vez que se haya establecido la conexión, se visualizará la lista de tablas de la base de datos. Posteriormente también puede ver el Documento de conexión examinando la vista de Servidores de bases de datos en la perspectiva de datos.

4. Seleccione la tabla **XBONUS** y pulse **Siguiente**.
5. Seleccione **Emparejar por nombre y tipo** y, a continuación, pulse **Finalizar**. Ahora se abrirá el Editor de correlaciones.
6. Oracle Pulse con el botón derecho del ratón en la tabla XBONUS y seleccione **Abrir editor de tablas**. En el editor de tablas, realice lo siguiente:
   a. Seleccione la pestaña **Columna**.
   b. Seleccione la columna BONUSPOINT y cambie el tipo de columna de NUMBER a INTEGER
   c. Guarde los cambios.
7. En el panel Beans enterprise, expanda el bean **Bonus**. En el panel Tablas, expanda la tabla **XBONUS**.
8. Correlacione los campos del bean Bonus con las columnas de la tabla XBONUS, realizando lo siguiente:
   a. Pulse con el botón derecho del ratón en el bean Bonus y seleccione **Emparejar por nombre**.
9. Guarde el archivo Map.mapxmi (Control+S) y cierre el archivo.
10. Oracle Deberá editar la definición de tabla utilizando un editor de texto, como se indica a continuación:
    a. Abra el archivo XBONUS.xmi con un editor de texto.
    b. Sustituya todas las apariciones de `SQLNumeric6` por `SQLNumeric3`.
    c. Guarde los cambios.

Express

1. Abra la perspectiva de Datos y cambie a la vista de Definición de datos.
2. Vaya al directorio siguiente: **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF**.
3. Pulse el botón derecho del ratón en **META-INF** y seleccione **Definición de base de datos nueva**. Se abrirá el asistente de Definición de base de datos nueva.
4. En el campo **Nombre de base de datos**, entre el nombre de la base de datos de desarrollo. Por ejemplo, entre `Demo_Dev`.
5. En la lista desplegable **Tipo de proveedor de base de datos**, seleccione **DB2 Universal Database Express V8.1** y pulse **Finalizar**. La nueva definición de base de datos ya se ha creado.
6. Vaya al siguiente directorio: **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF** > **Esquema** > *BDdesarrollo*.
7. Pulse el botón derecho del ratón en **BDdesarrollo** y seleccione **Nuevo** > **Definición de esquema nuevo**. Aparece la ventana Definición de esquema nuevo.

8. En el campo **Nombre de esquema**, entre `NULLID` y pulse **Finalizar**.
9. Vaya al siguiente directorio: **WebSphereCommerceServerExtensionsData** > **ejbModule** > **META-INF** > **Esquema** > *BDdesarrollo* > **NULLID** y seleccione **Nuevo** > **Definición de base de datos nueva**. Se abrirá el asistente de Definición de base de datos nueva.
10. En el campo **Nombre de tabla**, entre `XBONUS` y pulse **Siguiente**.
11. Añada la columna de clave en la definición de tabla, del modo siguiente:
    a. Pulse **Añadir otra** para añadir la columna MEMBERID.
    b. En el campo **Nombre de columna**, entre `MEMBERID`.
    c. Seleccione **Columna de clave**.
    d. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:
    DB2 `BIGINT`
    Oracle `NUMBER`
12. Pulse **Añadir otra** para añadir la columna BONUSPOINT.
13. En el campo **Nombre de columna**, entre `BONUSPOINT`.
14. En la lista desplegable **Tipo de columna**, seleccione **INTEGER** y pulse **Finalizar**.
15. Vaya a la Perspectiva J2EE y en la vista Jerarquía J2EE, seleccione **Módulos EJB** > **WebSphereCommerceServerExtensionsData**.
16. Pulse el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Nuevo** > **Bean de acceso**. Aparece la ventana Añadir un bean de acceso.
17. Seleccione **Ayudante de copia** y pulse **Siguiente**.
18. Seleccione el bean **Bonus** y pulse **Siguiente**.
19. En el recuadro de selección **Método constructor**, seleccione **findByPrimaryKey(com.ibm.commerce.extension.objects.Bonus)** y pulse **Finalizar**.
20. En la vista de Jerarquía J2EE, seleccione **Módulos EJB** > **WebSphereCommerceServerExtensionsData**.
21. Pulse el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Abrir con** > **Editor del descriptor de despliegue**.
22. Vaya a la sección JNDI - Por omisión y compruebe que el valor de **Nombre JNDI de origen de datos** esté vacío. Si el valor es **jdbc/Default**, suprima **jdbc/Default** y pulse Control+S para guardar los cambios.
23. En la vista de Jerarquía J2EE, expanda **Módulos EJB**, a continuación pulse con el botón derecho del ratón **WebSphereCommerceServerExtensionsData** y seleccione **Generar** > **Código de despliegue y RMIC**.
24. Seleccione el bean **Bonus** y pulse **Finalizar**.

El siguiente paso consiste en arreglar el nombre de esquema para que el nuevo bean se pueda portar a otras bases de datos. Para efectuar esta modificación, haga lo siguiente:

1. En la Perspectiva de J2EE, conmute a la vista de Jerarquía J2EE.
2. Expanda **Bases de datos** y, a continuación, expanda **WebSphereCommerceServerExtensionsData**.
3. Pulse con el botón derecho del ratón en el nodo de esquema (por ejemplo DB2USER) y seleccione **Redenominar**.
4. Establezca el valor en `NULLID`.

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 WebSphere Studio Application Developer se utilizan para generar este bean de acceso, basándose en el bean de entidad que ya se ha creado (en particular, el bean de acceso sólo utilizará los métodos que se hayan promocionado a la interfaz remota). Para crear el bean de acceso para el bean de entidad Bonus, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** y, a continuación, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Nuevo > Bean de acceso**.
   Se abrirá la ventana Añadir un bean de acceso.
2. Seleccione **Ayudante de copia** y pulse **Siguiente**.
3. Seleccione el bean **Bonus** y pulse **Siguiente**.
4. En la lista desplegable Método constructor, seleccione **findByPrimaryKey(com.ibm.commerce.extension.objects.BonusKey)** como método constructor.
5. Seleccione todos los atributos en la sección Ayudantes de atributo.
6. Pulse **Finalizar**.

Puede ver el código que se acaba de generar yendo a la pestaña del Business Professional Navegador J2EE Express Navegador de proyectos y expandiendo lo siguiente:
WebSphereCommerceServerExtensionsData > ejbModule > com.ibm.commerce.extension.objects
Se crea una nueva clase denominada BonusAccessBean y una nueva interfaz denominada BonusAccessBeanData, que se muestran dentro del paquete.

El siguiente paso consiste en generar el código desplegado.

El programa de utilidad de generación de código analiza los beans para asegurar que se cumplan las especificaciones de EJB de Sun Microsystems y comprueba que se sigan las normas específicas del 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 inicial y remota, así como las clases de buscador y persistencia JDBC para los beans CMP. También genera las clases Java ORB, de ″apéndices″ y de relación necesarias para el acceso RMI a través de IIOP, así como ″apéndices″ para las interfaces inicial y remota.

Para generar el código desplegado, efectúe lo siguiente: Business Professional

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**, a continuación pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Código de despliegue y RMIC**.
2. Seleccione el bean **Bonus** y pulse **Finalizar**.

Express

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** > **WebSphereCommerceServerExtensionsData**.
2. Pulse el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Abrir con** > **Editor del descriptor de despliegue**.

3. Vaya a la sección JNDI - Por omisión y compruebe que el valor de **Nombre JNDI de origen de datos** esté vacío. Si el valor es **jdbc/Default**, suprima **jdbc/Default** y pulse Control+S para guardar los cambios.

Puede ver el código que se acaba de generar yendo a la vista del Business Professional Navegador J2EE Express Navegador de proyectos. Encontrará lo siguiente:

Tabla 11.

Tipo de código	Nombre de clase
Código generado por implementación de contenedor	EJSCMPBonusHomeBean.java
	EJSRemoteCMPBonus.java
	EJSRemoteCMPBonusHome.java
	EJSFinderBonusBean.java
Código de acceso JDBC	EJSJDBCPersisterCMPBonusBean.java
Código de apéndices y relación RMI	_EJSRemoteCMPBonus_Tie.java
	_Bonus_Stub.java
	_EJSRemoteCMPBonusHome_Tie.java
	_BonusHome_Stub.java

El siguiente paso es utilizar el cliente de prueba universal para probar el nuevo bean enterprise, realizando lo siguiente:

1. Conmute a la perspectiva de Servidores.
2. En la vista Configuración de servidor, efectúe una doble pulsación en el servidor **WebSphereCommerceServer** y pulse la pestaña **Configuración**.
3. Seleccione **Habilitar el cliente de prueba universal**. Guarde los cambios.
4. En la vista de Servidores, pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar**.
5. En la Jerarquía J2EE, expanda **Módulos EJB > WebSphereCommerceServerExtensionsData**.
6. Pulse con el botón derecho del ratón en el bean **Bonus** y seleccione **Ejecutar en servidor**.
   Se abrirá el Cliente de prueba universal de IBM.
7. En el panel izquierdo, pulse **Bonus** y, a continuación, pulse **Bonus Home**.
8. Pulse el método **Bonus create(Long, Integer)**.
9. En el campo **Largo** del panel derecho, entre `-1000` y en el campo **Entero**, entre `1000`.
10. Pulse **Invocar** y se mostrará el resultado en el panel inferior.
11. Pulse **Trabajar con objeto** para añadir la interfaz remota al panel Referencias y verá los valores que ha entrado bajo Referencias de EJB. Se ha creado un registro nuevo en la tabla BONUS.
12. Seleccione el método **getMemberId** y pulse **Invocar** y se visualizará el resultado de -1000 en el panel inferior.
13. Cierre el cliente de prueba y detenga el servidor.

Integración del 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 WebSphere Studio Application Developer. De este modo ha determinado que puede actualizar la información de base de datos satisfactoriamente. Ahora, integrará el bean de entidad Bonus con la lógica MyNewControllerCmd. Una vez que se haya actualizado el código Java, se actualizará el archivo MyNewJSPTemplate.jsp para crear una interfaz que permita actualizar el saldo de puntos de bonificación de un cliente.

Para integrar el bean de entidad Bonus es necesario realizar los siguientes pasos generales:

1. Modificar el mandato de tarea MyNewTaskCmd para incluir campos y métodos para puntos de bonificación, actualizar el método validateParameters y añadir lógica a fin de actualizar el saldo de puntos de bonificación de un usuario.
2. Añadir un método getResources a la clase MyNewControllerCmdImpl para devolver una lista de los recursos que el mandato utiliza. Este método se incluye para control de acceso.
3. Crear un BonusDataBean nuevo para que los puntos de bonificación se puedan visualizar en una plantilla JSP.
4. Crear una política de control de acceso nuevo para los nuevos recursos.
5. Modificar la plantilla MyNewJSPTemplate.jsp para poder entrar puntos de bonificación para un usuario y, a continuación, visualizar el nuevo saldo de puntos de bonificación de dicho usuario.

Modificación de la interfaz MyNewTaskCmd para incluir puntos de bonificación

En este paso, modificará la interfaz MyNewTaskCmd para especificar los campos y métodos necesarios para los puntos de bonificación, realizando lo siguiente:

1. Conmute a la perspectiva Java y expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
2. Expanda el directorio **com.ibm.commerce.sample.commands\src**.
3. Efectúe una doble pulsación en la interfaz **MyNewTaskCmd** para ver el código fuente.
4. Elimine la marca de comentario de la sección 2 de importación para incluir el paquete siguiente:

```
/// Import section 2 ////////////////////////////////////
import com.ibm.commerce.extension.objects.*;
/// End of import section 2 /////////////////////////////
```

5. Elimine la marca de comentario de la Sección 4 para insertar el código siguiente en el método:

```
/// Section 4 ///////////////////////////////////////////////

 public java.lang.Integer getOldBonusPoints();
 public Integer getTotalBonusPoints();

 public void setBonusAccessBean(BonusAccessBean bb);
 public BonusAccessBean getBonusAccessBean();


/// End of section 4/////////////////////////////////////////
```

6. Guarde los cambios.

Modificación de MyNewTaskCmdImpl para calcular los puntos de bonificación

MyNewTaskCmdImpl se utiliza como punto de integración entre el bean de entidad Bonus y MyNewControllerCmd (puesto que MyNewControllerCmd invoca MyNewTaskCmd).

Para modificar MyNewTaskCmdImpl a fin de calcular los puntos de bonificación, realice lo siguiente:

1. Seleccione la clase **MyNewTaskCmdImpl** para ver el código fuente.
2. Elimine la marca de comentario de la sección 2 de importación para insertar el paquete siguiente:

```
/// Import section 2 ////////////////////////////////////
import com.ibm.commerce.extension.objects.*;
///  End of Import section 2 ////////////////////////////////
```

3. Elimine la marca de comentario de las Secciones 3A y 3B para insertar el código siguiente en la clase:

```
//// Section 3A ////////////////////////////////////////////

 private java.lang.Integer oldBonusPoints;
 private java.lang.Integer totalBonusPoints;


 private BonusAccessBean bb = null;

////End of Section 3A ////////////////////////////////////////

//// Section 3B ////////////////////////////////////////////

 public void setBonusAccessBean(BonusAccessBean newBB) {
  bb = newBB;
 }

 public BonusAccessBean getBonusAccessBean(){
  return bb;
 }

    public java.lang.Integer getOldBonusPoints() {
  return oldBonusPoints;
 }

 public Integer getTotalBonusPoints(){
  return totalBonusPoints;
 }


/// End of section 3B ////////////////////////////////////////
```

4. En la vista de Esquema, seleccione el método **validateParameters** y elimine la marca de comentario de la Sección 2, a fin de insertar el código siguiente en el método:

```
// section 2 ////////////////////////////////////////////////

 try {
    oldBonusPoints = bb.getBonusPoint();
 }  catch (javax.ejb.FinderException e) {
  try {
   // If bb is null, create a new instance
    bb = new BonusAccessBean(new Long(foundUserId), new Integer(0));
    oldBonusPoints = new Integer(0);
   } catch (javax.ejb.CreateException ec) {
    throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     this.getClass().getName(), "validateParameters");
   } catch (javax.naming.NamingException ec) {
```

```
        throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
         this.getClass().getName(), "validateParameters");
       } catch (java.rmi.RemoteException ec) {
        throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
         this.getClass().getName(), "validateParameters");
       }
     } catch (javax.naming.NamingException e) {
       throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
        this.getClass().getName(), "validateParameters");
     } catch (java.rmi.RemoteException e) {
       throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
        this.getClass().getName(), "validateParameters");
     } catch (javax.ejb.CreateException e) {
       throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        this.getClass().getName(), "validateParameters");
      }

    // end of section 2 ////////////////////////////////////////////
```

5. En la vista de Esquema, seleccione el método **performExecute**.
6. En el código fuente para este método performExecute, elimine la marca de comentario de la Sección 2. Esto inserta el código siguiente en el método:

```
/// use BonusAccessBean to update new bonus point
/// Section 2 ////////////////////////////////////////////////

 int newBP =  oldBonusPoints.intValue() + getInputPoints().intValue();
 totalBonusPoints = new Integer (newBP);
 bb.setBonusPoint(totalBonusPoints)   ;

 try {
   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");
 }

/// End of section 2 ////////////////////////////////////////////
```

7. Guarde los cambios.

Adición de un método getResources a la clase MyNewControllerCmdImpl

En esta sección, añadirá un método getResources nuevo a MyNewControllerCmdImpl. Este método devuelve una lista de 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. Efectúe una doble pulsación en la clase **MyNewControllerCmdImpl** para abrir y ver el código fuente.
2. En el código fuente, elimine la marca de comentario de la Sección 2 de importación para insertar el paquete siguiente en la clase:

```
/// Import Section 2 ////////////////////////////
import com.ibm.commerce.extension.objects.*;
/// End of Import Section 2 ////////////////////////
```

3. Elimine la marca de comentario de la Sección, para insertar el código siguiente en la clase:

```
/// Section 3 ////////////////////////////////////////////
/// Create an instance variable of type AccessVector to hold
/// the resources and a BonusAccessBean instance variable for
/// access control purposes.

    private AccessVector resources = null;
    private BonusAccessBean bb = null;

/// End of Section 3 /////////////////////////////////////
```

4. En el código fuente, elimine la marca de comentario de la Sección de Control de acceso. Esta sección se muestra en el extracto de código siguiente:

```
/// AccessControl Section ///////////////////////////////////

public AccessVector getResources() throws ECException{

   if (resources == null) {

    /// use UserRegistryAccessBean to check user reference number

 String refNum = null;
 String methodName = "getResources";

  rrb = new UserRegistryAccessBean();

 try {
   rrb = rrb.findByUserLogonId(getUserName());
   refNum = rrb.getUserId();
 }  catch (javax.ejb.FinderException e) {
   throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
        this.getClass().getName(),methodName,e);
 } catch (javax.naming.NamingException e) {
   throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
        this.getClass().getName(), methodName,e);
 } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
        this.getClass().getName(), methodName,e);
 } catch (javax.ejb.CreateException e) {
   throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
        this.getClass().getName(), methodName,e);
  }


 /// find the bonus bean for this registered user

    bb = new com.ibm.commerce.extension.objects.BonusAccessBean();
 try {
   if (refNum != null) {
    bb.setInitKey_memberId(new Long(refNum));
    bb.refreshCopyHelper();
    resources = new AccessVector(bb);
   }
 }  catch (javax.ejb.FinderException e) {

   ///doesn't have a bonus object so return the container that
     ///will hold the bonus object when it's created
  resources = new AccessVector(rrb);
  return resources;

 } 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 resources;
       }

       /// End of AccessControl Section ////////////////////////////////
```

5. Guarde los cambios.

Modificación del método performExecute de la clase MyNewControllerCmdImpl

En este paso modificará el código del método performExecute de MyNewControllerCmdImpl para incluir código relacionado con el nuevo bean Bonus, del modo siguiente:

1. Efectúe una doble pulsación en la clase **MyNewControllerCmdImpl**.
2. En la vista de Esquema, seleccione el método **performExecute**.
3. En el código fuente, elimine la marca de comentario de las Secciones 4E, 4G y 4H para insertar el código siguiente en el método:

```
// Section 4E ////////////////////////////////////
/// pass bb instance variable to the task command
    cmd.setBonusAccessBean(bb);
// End of section 4E ////////////////////////////

// Section 4G ////////////////////////////////////
if (cmd.getOldBonusPoints() != null) {
      rspProp.put("oldBonusPoints", cmd.getOldBonusPoints());
}
// End of section 4G ////////////////////////////

// Section 4H ////////////////////////////////////
///Instantiate the bonus data bean , then put it to response properties
   BonusDataBean bdb  =
      new com.ibm.commerce.sample.databeans.BonusDataBean(
         cmd.getBonusAccessBean());
  rspProp.put("bdbInstance", bdb );
// End of section 4H ////////////////////////////////
```

4. Guarde los cambios.

Nota: Verá errores porque BonusDataBean aún no se ha definido. Éstos se arreglarán en la sección siguiente.

Creación del bean de datos BonusDataBean

De acuerdo con el modelo de programación, deberá crear un nuevo bean de datos que corresponda al nuevo bean de entidad Bonus. Mientras que no todos los beans de entidad necesitan tener un bean de datos correspondiente, si desea poder visualizar información del bean de entidad en una plantilla JSP, deberá crear un nuevo bean de datos con esta finalidad.

En este escenario, es necesario que cree un nuevo bean de datos BonusDataBean que amplíe el BonusAccessBean. Como sucede con otras partes de la guía de aprendizaje, se proporciona el código base y es necesario eliminar los signos de comentario de diversas secciones de código.

Para insertar BonusDataBean en el código, realice lo siguiente:

1. El primer paso es importar el código base para el nuevo bean de datos, del modo siguiente:
   a. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos en la perspectiva de Java.
   b. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
   c. Pulse con el botón derecho del ratón en la carpeta **src** y seleccione **Importar**.
      Se abrirá el asistente para Importar.
   d. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
   e. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
      suDirectorio\WC_SAMPLE_55.zip
      donde *suDirectorio* es el directorio en el que ha bajado el paquete.
   f. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione el archivo siguiente para importarlo.
      - com\ibm\commerce\sample\databeans\BonusDataBean.java
   g. En el campo **Carpeta**, la carpeta `WebSphereCommerceServerExtensionsLogic/src` ya está especificada. Conserve este valor.
   h. Pulse **Finalizar**.
2. Efectúe una doble pulsación en la clase **BonusDataBean** para ver el código fuente.
3. En el código fuente, elimine la marca de comentario de la Sección 1, para insertar el código siguiente en el bean:

```
/// Section 1 ////////////////////////////////////////////

// create fields and accessors (setter/getter methods)

 private java.lang.String userId;
 private java.lang.Integer totalBonusPoints;


 public java.lang.String getUserId() {
  return userId;
 }

 public void setUserId(java.lang.String newUserId) {
  userId = newUserId;

  ////////////////////////////////////
  /// Section A : instantiate BonusAccessbean

  if (userId != null)
   this.setInitKey_memberId(new Long(newUserId));

  ////////////////////////////////////
 }


    public java.lang.Integer getTotalBonusPoints() {
  return totalBonusPoints;
 }
 public void setTotalBonusPoints(java.lang.Integer newTotalBonusPoints) {
  totalBonusPoints= newTotalBonusPoints;
 }
```

```
//// End of section 1 ////////////////////////////////////////////
```

4. A continuación, elimine la marca de comentario de la Sección 2, para insertar la siguiente sección de código en el bean:

```
/// Section 2/////////////////////////////////////////////

// create a new constructor for passing access bean into databean
// so that JSP can work with the access bean

public BonusDataBean(BonusAccessBean bb)
   throws com.ibm.commerce.exception.ECException {
 try {
  super.setEJBRef(bb.getEJBRef());
 }  catch (javax.ejb.FinderException e) {
     throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
    "BonusDataBean", "BonusDataBean(bb)");
 } catch (javax.naming.NamingException e) {
   throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
           "BonusDataBean", "BonusDataBean(bb)");
 } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
           "BonusDataBean", "BonusDataBean(bb)");
 } catch (javax.ejb.CreateException e) {
   throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
           "BonusDataBean", "BonusDataBean(bb)");
  }
}

//// End of section 2 ////////////////////////////////////////////
```

5. A continuación, elimine la marca de comentario de la Sección 3, para insertar el código siguiente en el bean:

```
//// Section 3 ////////////////////////////////////////////////

// set additional data field that is used for instantiating BonusAccessbean

 try
 {

  setUserId(getRequestProperties().getString("taskOutputUserId"));

  try {
   super.refreshCopyHelper();
  } catch (javax.ejb.FinderException e) {
      throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
     "BonusDataBean", "populate");
 } catch (javax.naming.NamingException e) {
   throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
           "BonusDataBean", "populate");
 } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
           "BonusDataBean", "populate");
 } catch (javax.ejb.CreateException e) {
   throw new ECSystemException(ECMessage._ERR_CREATE_EXCEPTION,
           "BonusDataBean", "populate");
  }


 }
 catch (ParameterNotFoundException e){}


///// End of Section 3 ////////////////////////////////////////
}
```

6. A continuación, elimine la marca de comentario de la Sección 4 para insertar el código siguiente en el bean:

```
/// Section 4 ////////////////////////////////////////

// copy input TypedProperteis to local

 requestProperties = aParam;

/// End of section 4 ////////////////////////////////
```

7. Guarde los cambios.
8. Compile el código cambiado pulsando el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccionando **Reconstruir proyecto**.

Creación de la política de control de acceso para el nuevo bean de entidad

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 `tester@mycompany`, dicho usuario sólo puede modificar sus propios puntos de bonificación.

Para establecer la política `AllUsersUpdateBonusResourceGroup` deben efectuarse los siguientes pasos:

1. Modificar las políticas de control de acceso para reflejar el entorno.
2. Cargar el archivo SampleACPolicy.xml utilizando el mandato `acpload`.
3. Cargar la descripción de SampleACPolicy_en_US.xml utilizando el mandato `acpnlsload`.

Si desea personalizar las políticas de control de acceso para el entorno, realice lo siguiente:

1. Determine el valor de ID de miembro para la tienda FashionFlow, del modo siguiente:
   - DB2 Si está utilizando una base de datos DB2, realice lo siguiente:
     a. Conéctese a la base de datos de desarrollo.

b. Emita la siguiente sentencia SQL:

```
select member_id from storeent where storeent_id=ID_enttienda_FF
```

donde *ID_enttienda_FF* es el ID de entidad de tienda para la tienda FashionFlow. Por ejemplo, puede entrar:

```
select member_id from storeent where storeent_id=10001
```

Anote el valor de ID de miembro: _____

- Oracle Si está utilizando una base de datos Oracle, realice lo siguiente:

a. Abra la ventana de mandatos de Oracle SQL Plus y conéctese a la base de datos de desarrollo.

b. Entre lo siguiente:
`column member_id format 999999999999999999999999999999999999` (treinta y cinco veces 9)
Esto establece la visualización de la columna member_id en una anchura suficiente para ver el valor entero.

c. Emita la siguiente sentencia SQL:

```
select member_id from storeent where storeent_id=ID_enttienda_FF
```

donde *ID_enttienda_FF* es el ID de entidad de tienda para la tienda FashionFlow. Por ejemplo, puede entrar:

```
select member_id from storeent where storeent_id=10001
```

Anote el valor de ID de miembro: _____

d. Restablezca la anchura de visualización de la columna member_id entrando lo siguiente:

```
column member_id clear
```

2. Utilizando un editor de texto, abra el archivo SampleACPolicy_template.xml (en el directorio *dir_instal_WCDE*\Commerce\xml\policies\xml) y sustituya *IdMiembroFashionFlow* por el valor de ID de miembro de la tienda Fashion Flow, tal como se ha determinado en el paso 1. Guarde el archivo modificado como SampleACPolicy.xml (en el mismo directorio).
3. Utilizando un editor de texto, abra el archivo SampleACPolicy_template_en_US.xml (en el directorio *dir_instal_WCDE*\Commerce\xml\policies\xml) y sustituya *IdMiembroFashionFlow* por el valor de ID de miembro de la tienda Fashion Flow, tal como se ha determinado en el paso 1. Guarde el archivo modificado como SampleACPolicy_en_US.xml (en el mismo directorio).
4. En un indicador de mandatos, vaya al directorio siguiente: *dir_instal_WCDE*\commerce\bin
5. Para cargar el archivo SampleACPolicy.xml, deberá 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. En este caso, entre `SampleACPolicy.xml`

Por ejemplo, puede emitir el mandato siguiente:

```
acpload Demo_dev db2user db2user SampleACPolicy.xml
```

6. Para cargar la descripción de política, deberá emitir el mandato acpnlsload, que tiene el formato siguiente:

   ```
   acpnlsload nombre_bd usuario_bd contraseña_bd archXMLentrada
   ```

   Por ejemplo, puede emitir el mandato siguiente:

   ```
   acpnlsload Demo_dev db2user db2user SampleACPolicy_en_US.xml
   ```

7. Si el servidor para el entorno de prueba está actualmente en ejecución, puede utilizar la opción de renovación de registro de la Consola de administración de WebSphere Commerce para actualizar el registro de control de acceso, como se indica a continuación:
   a. Abra un navegador Web y entre el URL siguiente:

      ```
      https://localhost/webapp/wcs/admin/servlet/ToolsLogon?XMLFile=
         adminconsole.AdminConsoleLogon
      ```

   b. Cuando se le solicite, conéctese utilizando un ID de administrador de sitio.
   c. Seleccione trabajar en el **Sitio** y pulse **Aceptar**.
   d. En el menú **Configuración**, seleccione **Registro**.
   e. Pulse **Actualizar todo** y, al cabo de un momento, pulse **Renovar** para verificar que se han realizado las actualizaciones.
   f. Desconéctese y cierre las ventanas de la Consola de administración.

Modificación de la plantilla MyNewJSPTemplate.jsp para incluir puntos de bonificación

Para modificar la página de visualización, realice lo siguiente:

1. En WebSphere Studio Application Developer, conmute a la perspectiva de Web.
2. Abra los archivos **MyNewJSPTemplate_All.jsp** y **MyNewJSPTemplate.jsp**.
3. Copie la Sección 9 del archivo MyNewJSPTemplate_All.jsp en el archivo MyNewJSPTemplate.jsp. Esto inserta el texto siguiente en la plantilla JSP:

```
<!-- SECTION 9 -->

<h2><fmt:message key="BonusAdmin" bundle="${tutorial}" /> </h2>

<c:if test="${!empty taskOutputUserId}">
 <ul>
  <li>
   <b>
   <fmt:message key="PointBeforeUpdate" bundle="${tutorial}" />
   <c:out value="${oldBonusPoints}"/>
   </b>
  </li>
  <li>
   <b>
   <fmt:message key="PointAfterUpdate" bundle="${tutorial}" />
      <c:out value="${bdbInstance.bonusPoint}" />
      </b>
     </li>
 </ul>
</c:if>


      <br />
<b><fmt:message key="EnterPoint" bundle="${tutorial}" /></b><p />


<form name="Bonus" action="MyNewControllerCmd">
<table>
 <tr>
  <td>
```

```
  <b>Logon ID </b>
 </td>
 <td>
  <input type="text" name="input1" value="<c:out
   value="${userName}"/>" />
 </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 9 -->
```

4. Guarde el archivo MyNewJSPTemplate.jsp.

Comprobación del bean Bonus integrado

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 comprobar la nueva lógica, realice lo siguiente:

1. Conmute a la vista de Servidor.
2. Pulse con el botón derecho del ratón en el servidor **WebSpherCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en el archivo **index.jsp** para la tienda y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda.
4. Conéctese como usuario registrado, realizando lo siguiente:
   a. Pulse el enlace **Registro**.
      Se visualizará la página Regístrese.
   b. En el campo **Dirección de correo electrónico**, entre la dirección de correo electrónico para el usuario que ha creado en el apartado "Comprobación de la validación de nombre de usuario" en la página 229.
   c. En el campo **Contraseña**, entre la contraseña para este usuario y, a continuación, pulse **Conexión**.
   d. Una vez que se haya completado la conexión, entre el URL siguiente en el mismo navegador:

      ```
      http://localhost/webapp/wcs/stores/servlet/MyNewControllerCmd?
         input1=correo_electrónico_usuario&input2=1000
      ```

      donde *correo_electrónico_usuario* es la dirección de correo electrónico del usuario que ha creado en el apartado "Comprobación de la validación de nombre de usuario" en la página 229. Se le mostrará una página que contiene todos los parámetros de salida anteriores así como un nuevo

formulario que le permitirá actualizar el saldo de los puntos de bonificación para el usuario.

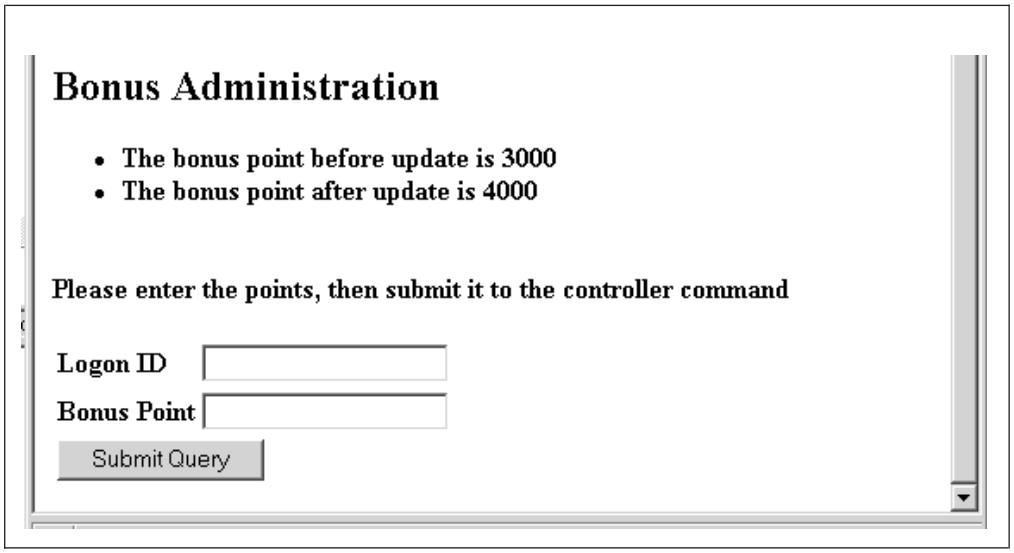


Figura 41.

e. Ahora, en el campo **Logon ID**, entre la dirección de correo electrónico del usuario y en el campo **Bonus Point**, entre `500`. Pulse **Submit Query** . Se visualizará una página similar a la siguiente que muestra que el saldo de puntos de bonificación se ha actualizado.

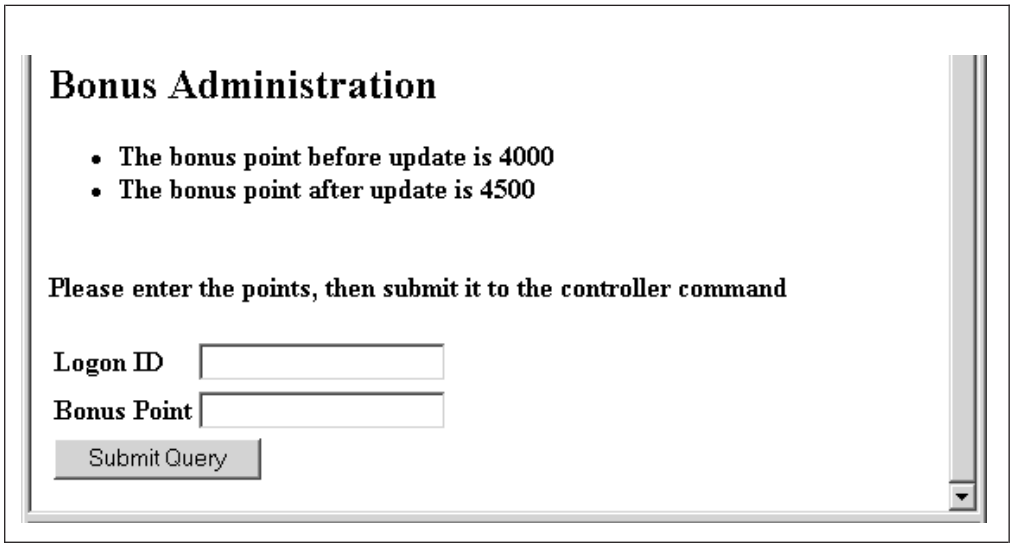


Figura 42.

Despliegue de la lógica de puntos de bonificación

Esta sección describe cómo desplegar la nueva lógica de negocio en una tienda que se ejecuta en un WebSphere Commerce Server remoto. Antes de empezar estos pasos de despliegue, tiene que haber creado una tienda (basada en la tienda de ejemplo FashionFlow) 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.

Existen varios tipos diferentes de elementos que debe desplegar en el WebSphere Commerce Server de destino. Éstos incluyen:

- Lógica de mandatos de controlador, mandatos de tarea y beans de datos
- Lógica de beans enterprise
- Una plantilla JSP y un archivo de imagen
- Un archivo de propiedades y paquetes de recursos
- Actualizaciones de base de datos que incluye actualizaciones de esquema (tabla nueva) así como actualizaciones de registro de mandatos
- Actualizaciones de control de acceso

Esta sección describe cómo desplegar todos estos elementos, *de forma incremental* en el WebSphere Commerce Server de destino. Esto se realiza como despliegue incremental, a diferencia de un despliegue del archivo EAR entero.

Creación del archivo JAR de mandatos y beans de datos

Esta sección describe cómo crear el archivo JAR que contiene la lógica de mandatos de controlador, mandatos de tarea y beans de datos.

Para crear este archivo JAR, realice los pasos siguientes en la máquina de desarrollo:

1. Cree un directorio en el sistema de archivos local denominado *unidad*:\ExportTemp.
2. En WebSphere Studio Application Developer, cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
3. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Exportar**. Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR** y pulse **Siguiente**.
   b. El panel izquierdo bajo **Seleccionar los recursos para exportar** está previamente lleno con el nombre del proyecto. Deje este valor como está.
   c. En el panel derecho, asegúrese de que sólo estén seleccionados los recursos siguientes:
      - .classpath
      - .project
      - .serverPreference
   d. Asegúrese de que **Exportar archivos de clases y recursos generados** esté seleccionado.
   e. *No* seleccione **Exportar archivos fuente y recursos Java**.
   f. En el campo **Seleccionar el destino de exportación**, entre el nombre de archivo JAR totalmente calificado que se debe utilizar. En este caso, entre *unidad*:\ExportTemp\WebSphereCommerceServerExtensionsLogic.jar. Tenga en cuenta que el nombre de archivo JAR debe ser WebSphereCommerceServerExtensionsLogic.jar.
   g. Pulse **Finalizar**.

Creación del archivo JAR EJB

Para crear el archivo JAR EJB, realice lo siguiente:

1. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda el proyecto **WebSphereCommerceServerExtensionsData**.
3. Efectúe una doble pulsación en **Descriptor de despliegue EJB**.
4. Con la pestaña Visión general seleccionada, desplácese a la parte inferior del panel para localizar la sección **Enlaces de WebSphere**.
5. En el campo **Nombre JNDI de origen de datos**, entre el nombre JNDI de origen de datos del WebSphere Commerce Server de destino. A continuación se muestra un valor de ejemplo:

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   donde el WebSphere Commerce Server de destino está utilizando una base de datos DB2 y el nombre de instancia de WebSphere Commerce es "demo"

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   donde el WebSphere Commerce Server de destino está utilizando una base de datos Oracle y el nombre de instancia de WebSphere Commerce es "demo".

El valor para el nombre JNDI de origen de datos se crea añadiendo "`jdbc/`" al nombre de origen de datos del WebSphere Commerce Server de destino. Puede verificar el nombre de origen de datos abriendo el archivo *nombreInstancia*.xml del WebSphere Commerce Server de destino y buscando `DatasourceName=` en el archivo.

6. Guarde los cambios del descriptor de despliegue (Control+S).
7. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsData** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
8. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR EJB** y pulse **Siguiente**.
   b. Business Professional El valor para **¿Qué recursos desea exportar?** está previamente lleno con el nombre del proyecto EJB.

      Express El nombre del proyecto EJB ya está rellenado.
      Deje el valor que aparece.
   c. Business Professional En el campo **¿Dónde desea exportar los recursos?** entre el nombre de archivo JAR totalmente calificado que se debe utilizar.

      Express Para el destino, entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      En este caso, entre
      unidad`:\ExportTemp\WebSphereCommerceServerExtensionsData.jar`.
   d. Pulse **Finalizar**.
9. Después de que se haya creado el archivo JAR, deshaga los cambios efectuados en el descriptor de despliegue local en el paso 5, para restaurar el valor que es necesario para el servidor de prueba local.

Nota: Esta guía de aprendizaje supone que la base de datos de desarrollo y la base de datos utilizada por el WebSphere Commerce Server de destino son del

mismo tipo. Si estaba desplegando en un tipo de base de datos diferente, deberá seguir las instrucciones contenidas en “Creación de un archivo JAR EJB con conversión” en la página 181.

Exportación de elementos de tienda

Para exportar los elementos de tienda, realice lo siguiente:

1. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda la carpeta **Stores**.
3. Pulse con el botón derecho del ratón en la carpeta **Web Content** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Sistema de archivos** y pulse **Siguiente**.
   b. Pulse **Deseleccionar todo**.
   c. Seleccione exportar los recursos siguientes:
      - Web Content*nombre_FashionFlow*\MyNewJSPTemplate.jsp
      - Web Content*nombre_FashionFlow*\images\male_blueshirt.gif
      - Web Content\WEB-INF\classes*nombre_FashionFlow*\Tutorial_NLS_en_US.properties
      - Web Content\WEB-INF\lib\jstl.jar
      - Web Content\WEB-INF\lib\standard.jar
   d. Seleccione **Crear estructura de directorios para archivos**.
   e. En el campo Directorio, entre un directorio temporal en el que se colocarán estos recursos. Por ejemplo, entre `C:\ExportTemp`
   f. Pulse **Finalizar**.

Empaquetado de políticas de control de acceso

En esta sección copiará las políticas de control de acceso que se han creado para los recursos nuevos en el directorio *unidad*:\ExportTemp, como se indica a continuación:

1. Vaya al directorio *dir_instal_WCDE*\Commerce\xml\policies\xml.
2. Copie los archivos siguientes en el directorio *unidad*:\ExportTemp\ACPolicies:
   - MyNewViewACPolicy.xml
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy_template.xml
   - SampleACPolicy_template_en_US.xml

Transferencia de elementos al WebSphere Commerce Server de destino

En este paso creará un directorio temporal en el WebSphere Commerce Server de destino y, a continuación, copiará los elementos de puntos de bonificación en este directorio. En los pasos subsiguientes, colocará los diferentes tipos de código en el lugar apropiado de la aplicación WebSphere Commerce.

Para copiar los archivos de la máquina de desarrollo en el WebSphere Commerce Server de destino, realice lo siguiente:

1. En el WebSphere Commerce Server de destino, cree un directorio temporal denominado *unidad*:\ImportTemp.
2. Determine cómo copiará los archivos de un sistema a otro. Puede realizar esta tarea correlacionando una unidad del WebSphere Commerce Server de destino con la máquina de desarrollo o utilizando una aplicación FTP, si la tiene configurada.
3. En la máquina de desarrollo, copie el contenido de *unidad*:\ExportTemp en *unidad*:\ImportTemp en el WebSphere Commerce Server de destino.

Detención del WebSphere Commerce Server de destino

Antes de iniciar los pasos de despliegue, debe detener el WebSphere Commerce Server de destino. Para obtener detalles sobre cómo detener el WebSphere Commerce Server, consulte la publicación Business Professional *WebSphere Commerce Studio, Guía de instalación* o Express *WebSphere Commerce - Express Developer Edition, Guía de instalación*.

Actualización de la base de datos en el WebSphere Commerce Server de destino

Antes de actualizar la base de datos de destino, verifique el ID de entidad de la tienda en la que está desplegando la lógica personalizada. Se puede utilizar la siguiente sentencia SQL para determinar este valor:

```
select STOREENT_ID from STOREENT where IDENTITY='nombre_FashionFlow'
```

donde *nombre_FashionFlow* es el nombre de la tienda en la que está desplegando el código.

Registro de la vista

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyNewView:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**).
2. En el menú **Herramientas**, seleccione **Valores de herramientas**.
3. Marque el recuadro de selección **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (**;**)
4. Cierre los valores de herramientas.
5. Con la pestaña Script seleccionada, cree la entrada necesaria en la tabla VIEWREG, entrando la información siguiente en la ventana de script:

```
connect to BDdestino user usuariobd using contraseñabd;
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, ID_enttienda_FF,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

donde

- *BDdestino* es el nombre de la base de datos de destino
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña del usuario de base de datos
- *ID_enttienda_FF* es el identificador exclusivo de la tienda que está basada en la tienda de ejemplo FashionFlow

Pulse el icono **Ejecutar**. Mantenga abierto el Centro de mandatos.

Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar la vista en la base de datos:

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 SQL:

```
insert into VIEWREG (VIEWNAME, DEVICEFMT_ID, STOREENT_ID, INTERFACENAME,
   CLASSNAME, PROPERTIES, DESCRIPTION, HTTPS, LASTUPDATE)
values ('MyNewView',-1, ID_enttienda_FF,
   'com.ibm.commerce.command.ForwardViewCommand',
   'com.ibm.commerce.command.HttpForwardViewCommandImpl',
   'docname=MyNewJSPTemplate.jsp','This is my new view for tutorial 1',
   0, null);
```

   donde

   - *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

   Pulse Intro para ejecutar la sentencia SQL.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

MyNewView ya se ha registrado.

Registro del nuevo mandato de controlador

Para registrar MyNewControllerCmd, realice lo siguiente:

1. DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyNewControllerCmd:
   a. En el Centro de mandatos, con la pestaña Script seleccionada, cree la entrada necesaria en la tabla URLREG, entrando la información siguiente en la ventana de script

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl',
   'local');
```

      donde

      - *BDdestino* es el nombre de la base de datos de destino
      - *usuariobd* es el usuario de base de datos
      - *contraseñabd* es la contraseña del usuario de base de datos
      - *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

Pulse el icono **Ejecutar**. Mantenga abierto el Centro de mandatos.

2. Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyNewControllerCmd:
   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 la siguiente sentencia SQL:

```
insert into URLREG (URL, STOREENT_ID, INTERFACENAME, HTTPS, DESCRIPTION,
   AUTHENTICATED) values ('MyNewControllerCmd',ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   0, 'This is a new controller command for tutorial one.',null);
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FF,
   'com.ibm.commerce.sample.commands.MyNewControllerCmd',
   'This is a new controller command for tutorial one.',
   'com.ibm.commerce.sample.commands.MyNewControllerCmdImpl','local');
```

   donde

   - *ID_enttienda_FF* es el identificador exclusivo para la tienda que está basada en la tienda de ejemplo FashionFlow.

   Pulse Intro para ejecutar la sentencia SQL.

   f. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

Creación de la tabla XBONUS

En este paso, creará la tabla XBONUS en la base de datos utilizada por el WebSphere Commerce Server de destino.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para crear la tabla:

1. En la ventana Script, entre lo siguiente:

```
create table XBONUS (MEMBERID BIGINT NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_xbonus
           primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade)
```

   donde

   - *BDdestino* es el nombre de la base de datos de destino
   - *usuariobd* es el usuario de base de datos
   - *contraseñabd* es la contraseña del usuario de base de datos

   Pulse el icono Ejecutar.
   La tabla XBONUS ya se ha creado.

Oracle Si está utilizando una base de datos Oracle, realice 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 SQL:

```
create table XBONUS (MEMBERID NUMBER NOT NULL,
    BONUSPOINT INTEGER NOT NULL, constraint p_xbonus
           primary key (MEMBERID),
    constraint f_xbonus foreign key (MEMBERID)
    references users (users_id) on delete cascade);
```

   y pulse Intro para ejecutar la sentencia SQL. La tabla XBONUS ya se ha creado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

Carga de políticas de control de acceso en el WebSphere Commerce Server de destino

En este paso cargará las políticas de control de acceso para los nuevos recursos en el WebSphere Commerce Server de destino.

Para cargar la nueva política, realice lo siguiente:

1. Actualice las políticas de control de acceso para reflejar los valores específicos del WebSphere Commerce Server de destino, como se indica a continuación:
   a. Determine el valor de ID de miembro para la tienda FashionFlow, del modo siguiente:
      - DB2 Si está utilizando una base de datos DB2, realice lo siguiente:
        1) Conéctese a la base de datos de desarrollo.
        2) Emita la siguiente sentencia SQL:

```
select member_id from storeent where storeent_id=ID_enttienda_FF
```

           donde *ID_enttienda_FF* es el ID de entidad de tienda para la tienda FashionFlow. Por ejemplo, puede entrar:

```
select member_id from storeent where storeent_id=10001
```

           Anote el valor de ID de miembro: _____
      - Oracle Si está utilizando una base de datos Oracle, realice lo siguiente:
        1) Abra la ventana de mandatos de Oracle SQL Plus y conéctese a la base de datos de desarrollo.
        2) Entre lo siguiente:
           `column member_id format 99999999999999999999999999999999999`
           (treinta y cinco veces 9)
           Esto establece la visualización de la columna member_id en una anchura suficiente para ver el valor entero.
        3) Emita la siguiente sentencia SQL:

```
select member_id from storeent where storeent_id=ID_enttienda_FF
```

           donde *ID_enttienda_FF* es el ID de entidad de tienda para la tienda FashionFlow. Por ejemplo, puede entrar:

```
select member_id from storeent where storeent_id=10001
```

           Anote el valor de ID de miembro: _____

4) Restablezca la anchura de visualización de la columna member_id entrando lo siguiente:

```
column member_id clear
```

b. Vaya al directorio siguiente:
unidad:\ImportTemp\ACPolicies

c. Utilizando un editor de texto, abra el archivo SampleACPolicy_template.xml y sustituya *IdMiembroFashionFlow* por el valor de ID de miembro de la tienda Fashion Flow, que se ha determinado en el paso 1a. Guarde como SampleACPolicy.xml el archivo modificado.

d. Utilizando un editor de texto, abra el archivo SampleACPolicy_template_en_US.xml y sustituya *IdMiembroFashionFlow* por el valor de ID de miembro de la tienda Fashion Flow, que se ha determinado en el paso 1a. Guarde como SampleACPolicy_en_US.xml el archivo modificado.

2. Copie las siguientes políticas de control de acceso en el directorio *dir_instal_WC*\xml\policies\xml:
   - MyNewViewACPolicy.xml
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy.xml
   - SampleACPolicy_en_US.xml
3. En un indicador de mandatos, vaya al directorio siguiente:
dir_instal_WC\bin
4. Debe emitir el mandato acpload, que tiene el formato siguiente:

```
acpload BDdestino usuariobd contraseñabd archivoXMLentrada
```

   donde
   - *BDdestino* es el nombre de la base de datos de desarrollo.
   - *usuariobd* es el nombre del usuario de base de datos.
   - *contraseñabd* es la contraseña del usuario de base de datos.
   - *archXMLentrada* es el archivo XML que contiene la especificación de política de control de acceso. En este caso, especifique `MyNewViewACPolicy.xml`.

   A continuación, se muestra un ejemplo del mandato en el que se especifican variables:

```
acpload Demo_Dev db2admin db2admin MyNewViewACPolicy.xml
```

5. Repita el paso 4 para cada unas de las políticas de control de acceso siguientes:
   - MyNewControllerCmdACPolicy.xml
   - SampleACPolicy.xml
6. Para cargar la descripción de política (que está contenida en el archivo SampleACPolicy_en_US.xml), deberá emitir el mandato acpnlsload, que tiene el formato siguiente:

```
acpnlsload nombre_bd usuario_bd contraseña_bd archXMLentrada
```

   Por ejemplo, puede emitir el mandato siguiente:

```
acpnlsload Demo_dev user password SampleACPolicy_en_US.xml
```

7. Compruebe en el directorio *dir_instal_WC*\xml\policies\xml si hay anotaciones cronológicas de errores que se puedan haber generado mientras se cargaban las políticas de control de acceso.

Actualización de elementos de tienda en el WebSphere Commerce Server de destino

En este paso, actualice la tienda con los elementos de tienda modificados, como se indica a continuación:

1. Haga una copia de seguridad del directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInst*.ear\Stores.war (donde *nombreCélula* es normalmente el nombre de sistema principal de la máquina y *nombreInst* es el nombre de la instancia de WebSphere Commerce).
2. Vaya al directorio *unidad*:\ImportTemp\Stores\Web Content.
3. Copie las carpetas *nombre_FashionFlow* y WEB-INF en el directorio siguiente: *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInst*.ear\Stores.war

Actualización del archivo JAR de mandatos y beans de datos en el WebSphere Commerce Server de destino

En este paso actualizará el WebSphere Commerce Server de destino para utilizar el nuevo archivo JAR de mandatos y beans de datos, como se indica a continuación:

1. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.
   b. Haga una copia del archivo WebSphereCommerceServerExtensionsLogic.jar y guárdela en una ubicación de copia de seguridad.
2. Copie el nuevo archivo WebSphereCommerceServerExtensionsLogic.jar del directorio *unidad*:\ImportTemp en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.

donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.

Actualización del archivo JAR EJB en el WebSphere Commerce Server de destino

En este paso actualizará el WebSphere Commerce Server de destino para utilizar el nuevo archivo JAR EJB, como se indica a continuación:

1. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.
   b. Haga una copia del archivo WebSphereCommerceServerExtensionsData.jar y guárdela en una ubicación de copia de seguridad.
2. Copie el nuevo archivo WebSphereCommerceServerExtensionsData.jar del directorio *unidad*:\ImportTemp en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear
3. A continuación, deberá modificar la información de descriptor de despliegue EJB, del modo siguiente:
   a. Localice el depósito de despliegue (directorio META-INF) para esta célula de WebSphere Application Server. Éste toma normalmente el formato siguiente:
   dir_instal_WAS\config\cells*nombreCélula*
   \applications\WC_*nombre_instancia*.ear\deployments\
   WC_*nombre_instancia**nombreMóduloEJB*.jar\META-INF.
   A continuación se muestra un ejemplo específico de esto:
   D:\WebSphere\AppServer\config\cells\myCell\applications\

WC_demo.ear\deployments\WC_demo\
WebSphereCommerceServerExtensionsData.jar\META-INF

b. El directorio contiene los archivos siguientes:
   - ejb-jar.xml
   - ibm-ejb-access-bean.xmi
   - ibm-ejb-jar-bnd.xmi
   - ibm-ejb-jar-ext.xmi
   - MANIFEST.MF

   Haga una copia de seguridad de todos estos archivos.

c. Utilice una herramienta para abrir el nuevo archivo WebSphereCommerceServerExtensionsData.jar y ver su contenido.

d. Extraiga el contenido del directorio meta-inf (los archivos anteriores listados) de este archivo WebSphereCommerceServerExtensionsData.jar en el directorio del paso 3a. Asegúrese de que la estructura de directorios sigue siendo correcta después de haber extraído los archivos.

4. Utilizando el mandato startServer de WebSphere Application Server en la línea de mandatos, reinicie la instancia de WebSphere Commerce. Consulte la publicación *WebSphere Commerce, Guía de instalación* para la plataforma y base de datos si desea más información sobre cómo iniciar y detener esta instancia.

Verificación de la lógica de puntos de bonificación en el WebSphere Commerce Server de destino

En este paso verificará que la lógica de puntos de bonificación se haya desplegado satisfactoriamente en el WebSphere Commerce Server de destino, realizando lo siguiente:

1. Abra un navegador Web y entre el URL para iniciar la tienda que está basada en la tienda de ejemplo FashionFlow.
2. Cree un nuevo usuario registrado, realizando lo siguiente:
   a. Pulse **Regístrese**.
   b. Pulse **Regístrese** otra vez para crear un cliente nuevo.
   c. En el formulario de registro, entre los valores apropiados en todos los campos obligatorios. Por ejemplo, en el campo de correo electrónico, entre `tester@mycompany`. Tome nota del valor para la dirección de correo electrónico: _____.
   d. Una vez entrados los valores, pulse **Someter**.
3. Una vez que se haya completado la conexión, entre el URL siguiente en el mismo navegador:

   ```
   http://nombresistpral/webapp/wcs/stores/servlet/MyNewControllerCmd?
      input1=correo_electrónico_usuario&input2=1000
   ```

   donde *nombresistpral* es el nombre de sistema principal y *correo_electrónico_usuario* es la dirección de correo electrónico para el usuario que ha creado en el paso 2. Se le mostrará una página que contiene todos los parámetros de salida anteriores así como un nuevo formulario que le permitirá actualizar el saldo de los puntos de bonificación para el usuario.

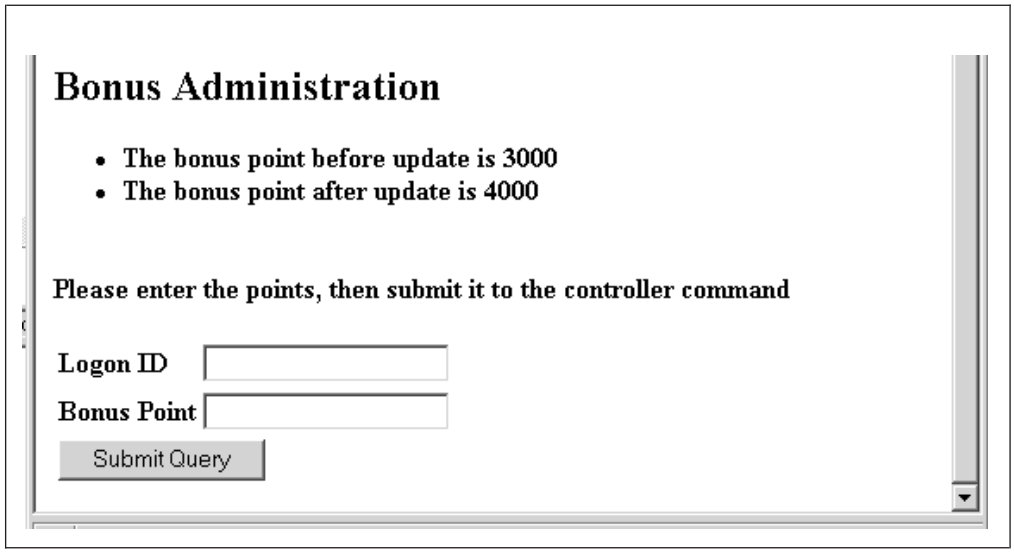


Figura 43.

4. Ahora, en el campo **Logon ID**, entre la dirección de correo electrónico del usuario y en el campo **Bonus Point**, entre `500`. Pulse **Submit Query** . Se visualizará una página similar a la siguiente que muestra que el saldo de los puntos de bonificación se ha actualizado.

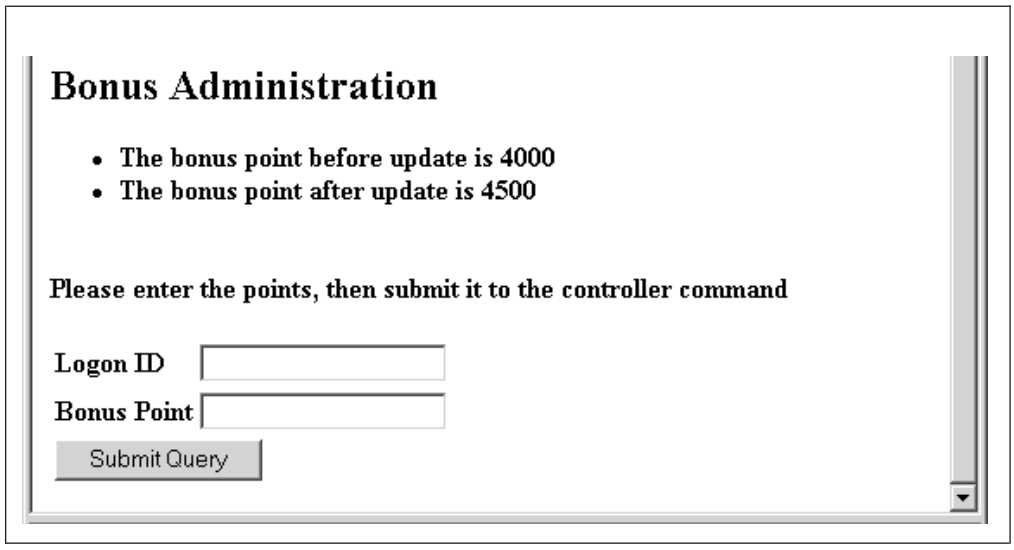


Figura 44.

Capítulo 11. Guía de aprendizaje: Modificación de un mandato de controlador existente

La finalidad de esta guía de aprendizaje es mostrar el proceso utilizado para modificar un mandato de controlador existente.

En esta guía de aprendizaje, restringirá a cinco o menos el número de artículos del carro de la compra de un cliente. Para implementar esta solución, alterará OrderItemAddCmdImpl con su propia implementación que incluirá la lógica para comprobar el número de artículos del carro. Si un cliente intenta añadir un sexto artículo al carro de la compra, se emitirá una excepción. Esta excepción utiliza un nuevo mensaje de error.

Tenga en cuenta que la finalidad de esta guía de aprendizaje es mostrar el proceso de desarrollo utilizado para modificar la lógica de mandatos existente. No está destinada a ser un ejemplo exhaustivo de restricción de artículos en el carro de la compra. Para esclarecer la guía de aprendizaje, se ha simplificado la lógica utilizada.

En esta guía de aprendizaje, aprenderá lo siguiente:

- Cómo crear una nueva implementación para un mandato de controlador existente
- Cómo actualizar el registro de mandatos para que se utilice la nueva implementación en la aplicación
- Cómo desplegar un mandato de controlador modificado en una aplicación WebSphere Commerce existente

Requisitos previos

Esta guía de aprendizaje no requiere que haya completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191. Si ha completado dicha guía de aprendizaje, no hay ningún peligro en dejar el código en el espacio de trabajo, puesto que dicho código no está en conflicto con esta guía de aprendizaje.

Antes de empezar esta guía de aprendizaje, tendrá que haber publicado una tienda basada en la tienda de ejemplo FashionFlow. En esta tienda, debe poder realizar una compra (por ejemplo, examinar el catálogo, añadir artículos al carro de la compra, pasar por caja y ver la confirmación del pedido).

Para poder realizar los pasos de despliegue, la tienda también tiene que existir en el WebSphere Commerce Server de destino.

Creación de la nueva clase MyOrderItemAddCmdImpl

En este paso de la guía de aprendizaje, creará una clase MyOrderItemAddCmdImpl nueva. Para crear esta clase, realice lo siguiente:

1. En WebSphere Studio Application Developer, abra la perspectiva Java (**Ventana > Abrir perspectiva > Java**).
2. Navegue hasta el proyecto **WebSphereCommerceServerExtensionsLogic**.
3. Vaya al directorio **src**.

4. Si no ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, pulse con el botón derecho del ratón en el directorio **src** y seleccione **Nuevo > Paquete**.
   Se abrirá el asistente para el Paquete Java nuevo. En este asistente, realice lo siguiente:
   a. En el campo **Nombre**, entre `com.ibm.commerce.sample.commands`.
   b. Pulse **Finalizar**.
5. Pulse con el botón derecho del ratón en el paquete **com.ibm.commerce.sample.commands**. Seleccione **Nueva > Clase**.
   Se abrirá el asistente para la Clase Java nueva.
6. En el asistente para Clase Java nueva, realice lo siguiente:
   a. En el campo **Nombre**, entre `MyOrderItemAddCmdImpl`.
   b. Para especificar la superclase, pulse el botón **Examinar** junto al campo de Superclase y, a continuación, entre `OrderItemAddCmdImpl`. Pulse **Aceptar**.
   c. Para especificar qué interfaz se debe implementar, pulse **Añadir** y, a continuación, entre `OrderItemAddCmd` y pulse **Aceptar**.
   d. Pulse **Finalizar**.

Se visualiza el código fuente para la clase MyOrderItemAddCmdImpl.

El siguiente paso consiste en actualizar las sentencias de importación en esta clase, como se indica a continuación:

1. En la vista de Esquema, seleccione y abra **declaraciones de importación**. Verá que ya hay dos sentencias de importación creadas.
2. En el código fuente para las sentencias de importación, añada las siguientes sentencias de importación (import):

```
import com.ibm.commerce.exception.ECApplicationException;
import com.ibm.commerce.exception.ECException;
import com.ibm.commerce.exception.ECSystemException;
import com.ibm.commerce.order.objects.OrderAccessBean;
import com.ibm.commerce.ras.ECMessage;
import com.ibm.commerce.sample.messages.MyNewMessages;
```

3. Guarde los cambios.

El siguiente paso consiste en añadir la lógica de negocio y el manejo de excepciones para determinar si hay más de cinco artículos en el carro de la compra del cliente antes de añadir más artículos de pedido. Actualice el código del modo siguiente:

1. En la vista de Esquema, seleccione la clase MyOrderItemAddCmdImpl y visualice el código fuente. Actualmente sólo contiene lo siguiente:

```
public class MyOrderItemAddCmdImpl
 extends OrderItemAddCmdImpl
 implements OrderItemAddCmd {

}
```

2. Deberá añadir un nuevo método performExecute en esta clase. Este método contiene la lógica para comprobar el número de artículos del carro de la compra y, si el número es inferior a cinco, se llamará al método performExecute corriente de la superclase (OrderItemAddCmdImpl), como es habitual. Si hay cinco artículos o más, se emite una excepción y el usuario no puede añadir más artículos al carro. Para añadir este método, copie el código fuente siguiente en la clase (asegúrese de que se incluye antes de la última llave de cierre "}" indicando el final de la clase):

```
public void performExecute() throws ECException {
  // Get a list of order ids
  String[] orderIds = getOrderId();

  // Check to make sure that an id exists at all
  // if order id exists then get number of items in the order
  // else if no order id exists then execute normal code
  if (orderIds != null && orderIds.length > 0) {
   // An exception should be thrown when trying to add a sixth item
   // to the cart.  Since this code is run before any items are added
   // throw and exception if there are 5 or more items in the cart
   if (itemsInOrder(orderIds[0]) >= 5) {
          throw new ECApplicationException(
     MyNewMessages._ERR_TOO_MANY_ITEMS,
    this.getClass().getName(),
     "performExecute");
   }
   //else perform normal flow
  }
  super.performExecute();
 }
 //get number of items in the order
 protected int itemsInOrder(String orderId) throws ECException {
  try {
   OrderAccessBean order = new OrderAccessBean();
   order.setInitKey_orderId(orderId);
   order.refreshCopyHelper();
   return order.getOrderItems().length;
  } catch (javax.ejb.FinderException e) {
   throw new ECSystemException(
    ECMessage._ERR_FINDER_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
  } catch (javax.naming.NamingException e) {
   throw new ECSystemException(
    ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
  } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(
    ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
  } catch (javax.ejb.CreateException e) {
   throw new ECSystemException(
    ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(),
    "itemsInOrder");
  }
 }
```

Después de haber pegado el nuevo código en la clase, pulse el botón derecho del ratón en el código fuente y seleccione **Formatear** para formatear el código.

3. Guarde el trabajo.

Nota: Hay avisos que indican que falta información de mensaje. Esto se corregirá en los pasos subsiguientes.

Creación de información de mensaje

La nueva implementación de mandato utiliza un nuevo mensaje de error: _ERR_TOO_MANY_ITEMS. En esta sección, creará el código para ese mensaje nuevo y el archivo de propiedades asociado. Para importar este código, realice lo siguiente:

1. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
2. Pulse con el botón derecho del ratón en el directorio **src** y seleccione **Nuevo > Paquete**.
   Se abrirá el asistente para el Paquete Java nuevo. En este asistente, realice lo siguiente:
   a. En el campo **Nombre**, entre com.ibm.commerce.sample.messages.
   b. Pulse **Finalizar**.
3. Pulse con el botón derecho del ratón en el paquete **com.ibm.commerce.sample.messages**. Seleccione **Nueva > Clase**.
   Se abrirá el asistente para la Clase Java nueva.
4. En el asistente para Clase Java nueva, realice lo siguiente:
   a. En el campo **Nombre**, entre MyNewMessages.
   b. Pulse **Finalizar**.
5. Efectúe una doble pulsación en la clase **MyNewMessages** para ver el código fuente.
6. Inmediatamente antes de la línea public class MyNewMessages del código, añada las siguientes sentencias de importación:

```
import com.ibm.commerce.ras.ECMessage;
import com.ibm.commerce.ras.ECMessageSeverity;
import com.ibm.commerce.ras.ECMessageType;
```

7. En la clase, añada el código siguiente:

```
// Resouce bundle used to extract the text for an exception
 static final String errorBundle = "MyNewErrorMessages";

 // An ECMessage is used to describe an ECException and is passed
 // into the ECException when thrown
 public static final ECMessage _ERR_TOO_MANY_ITEMS =
      new ECMessage(ECMessageSeverity.ERROR, ECMessageType.USER,
      MyNewMessageKeys._ERR_TOO_MANY_ITEMS, errorBundle);
```

8. Guarde los cambios.
9. Pulse con el botón derecho del ratón en el paquete **com.ibm.commerce.sample.messages**. Seleccione **Nueva > Clase**.
   Se abrirá el asistente para la Clase Java nueva.
10. En el asistente para Clase Java nueva, realice lo siguiente:
    a. En el campo **Nombre**, entre MyNewMessageKeys.
    b. Pulse **Finalizar**.
11. Defina el siguiente código en la clase:

```
public class MyNewMessageKeys {
// This class defines the keys used to create new exceptions that are
// thrown by customized code.
 public static final String _ERR_TOO_MANY_ITEMS = "_ERR_TOO_MANY_ITEMS";
}
```

12. Guarde los cambios.
13. Compile el código pulsando el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccionando **Crear proyecto**.
14. Cree el nuevo archivo de propiedades que contendrá la información de mensaje, como se indica a continuación:

a. Abra la perspectiva de Web (**Ventana > Abrir perspectiva > Web**).
b. En el proyecto Web **Stores**, expanda las carpetas **Web Content > WEB-INF > classes**.
c. Pulse con el botón derecho del ratón en la carpeta **classes** y seleccione **Nuevo > Otro > Simple > Archivo > Siguiente** para crear un nuevo archivo de propiedades.
Se abrirá la ventana Nuevo archivo.
d. En el campo **Nombre de archivo**, entre `MyNewErrorMessages.properties` y, a continuación, pulse **Finalizar**.
Se abrirá el nuevo archivo vacío.
e. En el nuevo archivo, copie el texto siguiente:

```
_ERR_TOO_MANY_ITEMS=Está intentando poner demasiados artículos diferentes
  en un carro de la compra.
```

Nota: El salto de línea en el código anterior es sólo para ofrecer una mejor presentación. Entre el texto en una sola línea.

f. Guarde los cambios.

Modificación del registro de mandatos

En este paso, modificará el registro de mandatos para que se utilice la nueva clase de implementación MyOrderItemAddCmdImpl en lugar de la clase de implementación OrderItemAddCmdImpl original. La única tabla del registro de mandatos que es necesario modificar es la tabla CMDREG. En este caso, se utiliza la nueva clase de implementación para todas las tiendas.

Para modificar el registro de mandatos, realice lo siguiente:

1. DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyOrderItemAddCmdImpl:
   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. Marque el recuadro de selección **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 BDdesarrollo user usuariobd using contraseñabd;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME=
'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

   donde
   - *BDdesarrollo* es el nombre de la base de datos de desarrollo
   - *usuariobd* es el usuario de base de datos
   - *contraseñabd* es la contraseña para el usuario de base de datos

   Pulse el icono **Ejecutar**.
2. Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyOrderItemAddCmdImpl:
   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 la siguiente sentencia SQL:

```
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME=
'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

Pulse Intro para ejecutar la sentencia SQL.

f. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

y pulse Intro para ejecutar la sentencia SQL.

Comprobación del mandato MyOrderItemAddCmdImpl

El siguiente paso es comprobar que la nueva lógica está funcionando correctamente. Para realizar esta comprobación, tiene que poder añadir de forma satisfactoria cinco artículos al carro de la compra, pero esperar que se emita un error cuando intente añadir un sexto artículo al carro.

Para probar la nueva lógica de negocio, realice lo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
   Se visualizará la página de presentación de la tienda en el navegador Web.
4. Compre en la tienda y añada cinco artículos al carro de la compra. Después de añadir el quinto artículo, tendrá un carro de la compra que será similar al siguiente:

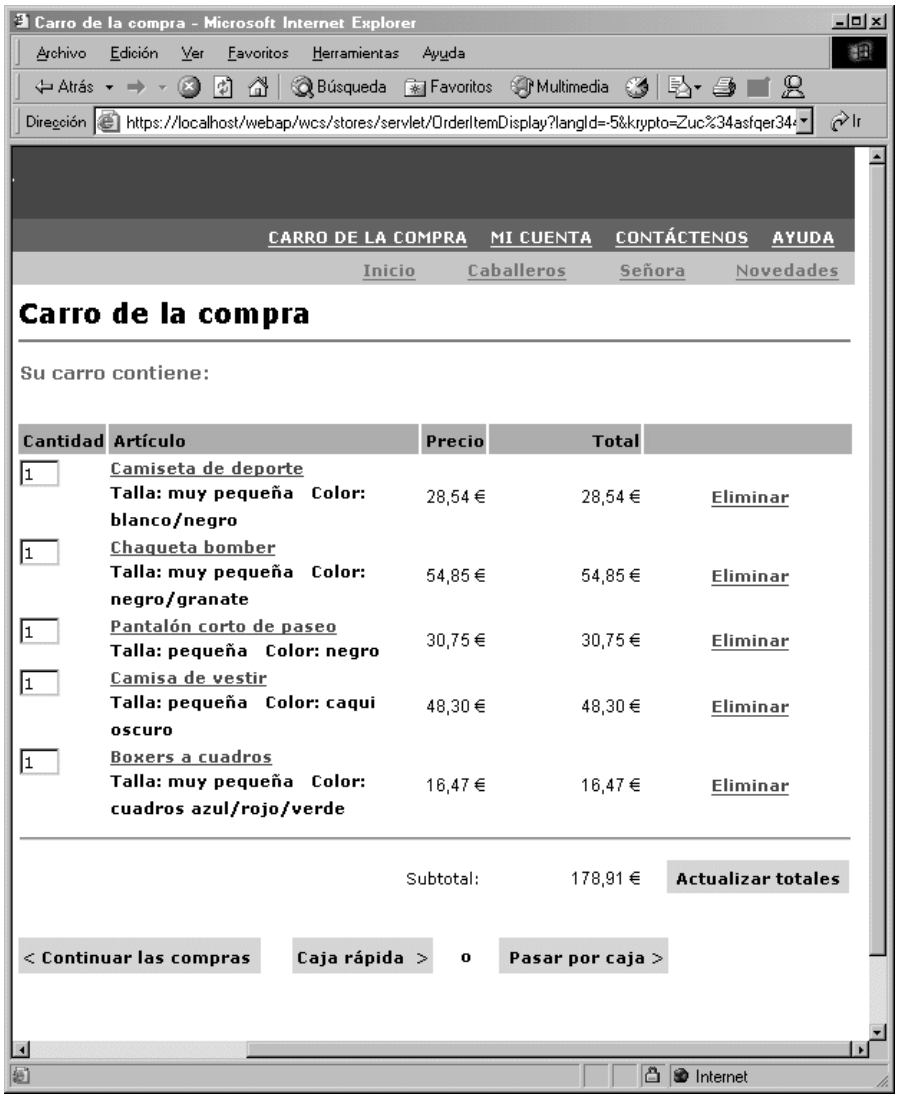


Figura 45.

5. Pulse **Continuar las compras** y seleccione otro artículo. Para este artículo seleccionado, pulse **Añadir al carro de la compra**. Aparecerá la siguiente página de error:

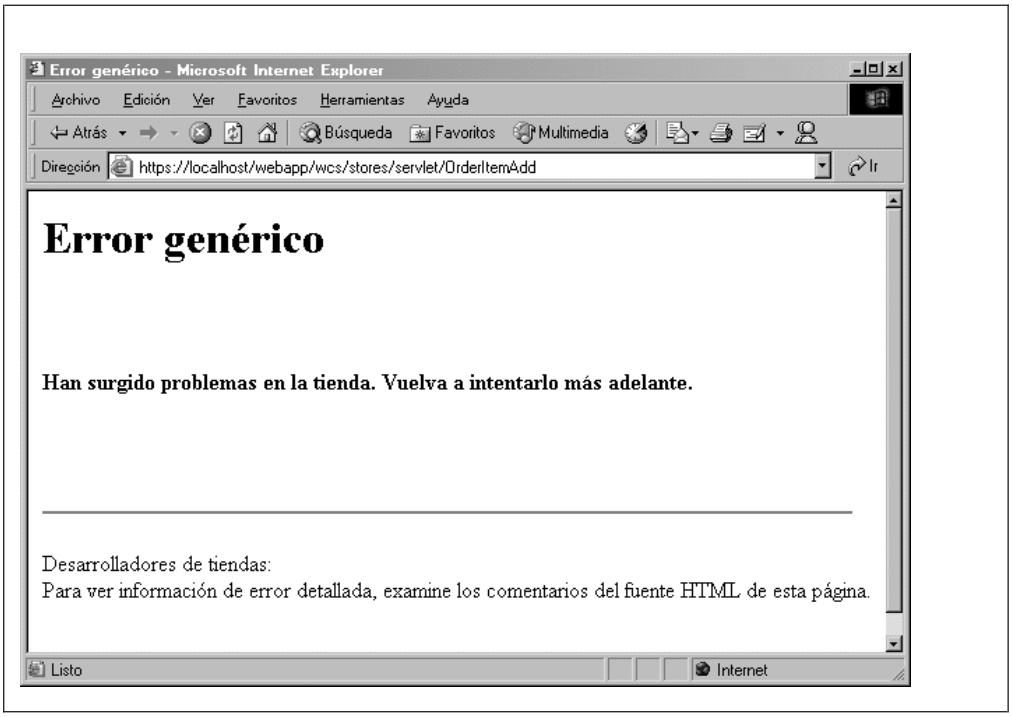


Figura 46.

Visualice el código fuente para la página de error. En el código fuente, desplácese hacia abajo para buscar la siguiente información de error:

```
Message Key: _ERR_TOO_MANY_ITEMS
Message: Está intentando poner demasiados artículos diferentes
en un carro de la compra.
```

Despliegue de MyOrderItemAddCmdImpl

En este paso, desplegará la lógica de negocio modificada en un WebSphere Commerce Server de destino. En este caso, el despliegue consta de los siguientes pasos de alto nivel:

1. Creación del archivo JAR que contiene la lógica de mandato y las clases de error.
2. Exportación del archivo de propiedades de mensaje de error.
3. Transferencia de los elementos al WebSphere Commerce Server de destino.
4. Actualización del registro de mandatos en el servidor WebSphere Commerce Server de destino.
5. Validación de la nueva lógica en el WebSphere Commerce Server de destino.

Creación del archivo JAR de mandatos

A continuación de la estrategia de personalización de código de WebSphere Commerce, se colocan los mandatos personalizados y los beans de datos en el proyecto WebSphereCommerceServerExtensionsLogic. Por lo tanto, cuando llega el momento de desplegar el código personalizado, observará que se incluye el código anteriormente personalizado en el archivo JAR. Por ejemplo, si ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, observará que cuando cree el archivo JAR para desplegar la clase MyOrderItemAddCmdImpl, se incluirá el MyNewControllerCmd creado anteriormente y otras clases.

Para crear el archivo JAR que contiene la clase MyOrderItemAddCmdImpl, realice los pasos siguientes en la máquina de desarrollo:

1. En el sistema de archivos local, cree un directorio denominado *unidad*:\ExportTemp2.
2. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
3. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR** y pulse **Siguiente**.
   b. El panel izquierdo bajo **Seleccionar los recursos para exportar** está previamente lleno con el nombre del proyecto. Deje este valor como está.
   c. En el panel derecho, asegúrese de que sólo estén seleccionados los recursos siguientes:
      - .classpath
      - .project
      - .serverPreference
   d. Asegúrese de que **Exportar archivos de clases y recursos generados** esté seleccionado.
   e. *No* seleccione **Exportar archivos fuente y recursos Java**.
   f. En el campo **Seleccionar el destino de exportación**, entre el nombre de archivo JAR totalmente calificado que se debe utilizar. En este caso, entre *unidad*:\ExportTemp2\WebSphereCommerceServerExtensionsLogic.jar. Tenga en cuenta que el nombre de archivo JAR debe ser WebSphereCommerceServerExtensionsLogic.jar.
   g. Pulse **Finalizar**.

Exportación del archivo de propiedades de mensaje

En esta sección, exportará el archivo de propiedades que contiene el texto para el nuevo mensaje, como se indica a continuación:

1. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda la carpeta **Stores**.
3. Pulse con el botón derecho del ratón en la carpeta **Web Content** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.

4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Sistema de archivos** y pulse **Siguiente**.
   b. Pulse **Deseleccionar todo**.
   c. Seleccione exportar el recurso siguiente:
      Web Content\WEB-INF\classes\MyNewErrorMessages.properties
   d. Seleccione **Crear estructura de directorios para archivos**.
   e. En el campo Directorio, entre un directorio temporal en el que se colocarán estos recursos. Por ejemplo, entre `C:\ExportTemp2`
   f. Pulse **Finalizar**.

Transferencia de elementos al WebSphere Commerce Server de destino

En este paso, creará un directorio temporal en el WebSphere Commerce Server de destino y, a continuación, copiará los elementos MyOrderItemAddCmdImpl en este directorio.

Para copiar los archivos de la máquina de desarrollo en el WebSphere Commerce Server de destino, realice lo siguiente:

1. En el WebSphere Commerce Server de destino, cree un directorio temporal denominado *unidad*:\ImportTemp2.
2. Determine cómo copiará los archivos de un sistema a otro. Puede realizar esta tarea correlacionando una unidad del WebSphere Commerce Server de destino con la máquina de desarrollo o utilizando una aplicación FTP, si la tiene configurada.
3. En la máquina de desarrollo, copie el contenido de *unidad*:\ExportTemp2 en *unidad*:\ImportTemp2 del servidor WebSphere Commerce Server de destino.

Detención del WebSphere Commerce Server de destino

Antes de empezar los pasos de despliegue, deberá detener el WebSphere Commerce Server de destino. Para obtener detalles sobre cómo detener el WebSphere Commerce Server de destino, consulte la publicación Business Professional *WebSphere Commerce Studio, Guía de instalación* o Express *WebSphere Commerce - Express Developer Edition, Guía de instalación*.

Actualización de la base de datos en el WebSphere Commerce Server de destino

En este paso, modificará el registro de mandatos para que utilice la nueva clase de implementación MyOrderItemAddCmdImpl.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyOrderItemAddCmdImpl:

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. Marque el recuadro de selección **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (**;**)
4. Cierre los valores de herramientas.
5. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla CMDREG indicando la siguiente información en la ventana de script:

```
connect to BDdestino user usuariobd using contraseñabd;
update CMDREG
set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
WHERE INTERFACENAME=
'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
and storeent_Id=0;
```

donde

- *BDdestino* es el nombre de la base de datos utilizada por el WebSphere Commerce Server de destino
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña de base de datos

Pulse el icono **Ejecutar**.

Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyOrderItemAddCmdImpl:

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 SQL:

   ```
   update CMDREG
   set CLASSNAME='com.ibm.commerce.sample.commands.MyOrderItemAddCmdImpl'
   WHERE INTERFACENAME=
   'com.ibm.commerce.orderitems.commands.OrderItemAddCmd'
   and storeent_Id=0;
   ```

   Pulse Intro para ejecutar la sentencia SQL.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia SQL.

Actualización del archivo JAR de mandatos en el WebSphere Commerce Server de destino

En este paso, actualizará el WebSphere Commerce Server de destino para utilizar el archivo JAR que contiene la nueva MyOrderItemAddCmdImpl, como se indica a continuación:

1. Utilizando el mandato stopServer de WebSphere Application Server en una línea de mandatos, detenga la instancia de WebSphere Commerce. Si es necesario, consulte la publicación *WebSphere Commerce, Guía de instalación* correspondiente a la plataforma y la base de datos para obtener información sobre cómo iniciar y detener esta instancia.
2. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear, donde *nombreCélula* es generalmente el nombre de sistema principal de la máquina
   b. Haga una copia del archivo WebSphereCommerceServerExtensionsLogic.jar y guárdela en una ubicación de copia de seguridad.

3. Copie el nuevo archivo WebSphereCommerceServerExtensionsLogic.jar del directorio *unidad*:\ImportTemp2 en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.

donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.

Actualización de las propiedades de mensaje en el WebSphere Commerce Server de destino

En este paso añadirá el nuevo archivo de propiedades de mensaje a la aplicación, como se indica a continuación:

1. Haga una copia de seguridad del directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInst*.ear\Stores.war (donde *nombreCélula* es normalmente el nombre de sistema principal de la máquina y *nombreInst* es el nombre de la instancia de WebSphere Commerce).
2. Vaya al directorio *unidad*:\ImportTemp\Stores\Web Content.
3. Copie la carpeta WEB-INF en el directorio siguiente: *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInst*.ear\Stores.war
4. Utilizando el mandato startServer de WebSphere Application Server en una línea de mandatos, reinicie la instancia de WebSphere Commerce.

donde *nombreInst* es el nombre de la instancia de WebSphere Commerce.

Verificación de la lógica MyOrderItemAddCmdImpl en el WebSphere Commerce Server de destino

En este paso, realizará una comprobación rápida para verificar que el código está funcionando correctamente, una vez que se ha desplegado.

Para verificar el código, realice lo siguiente:

1. Abra un navegador Web e inicie la tienda FashionFlow. Por ejemplo, entre el URL siguiente para iniciar la tienda:

   ```
   http://nombresistpral/webapp/wcs/stores/servlet/FashionFlow/index.jsp
   ```

   donde *nombresistpral* es el nombre de sistema principal para la instancia.
2. Compre en la tienda y añada cinco artículos al carro de la compra. Después de añadir el quinto artículo, tendrá un carro de la compra que será similar al siguiente:

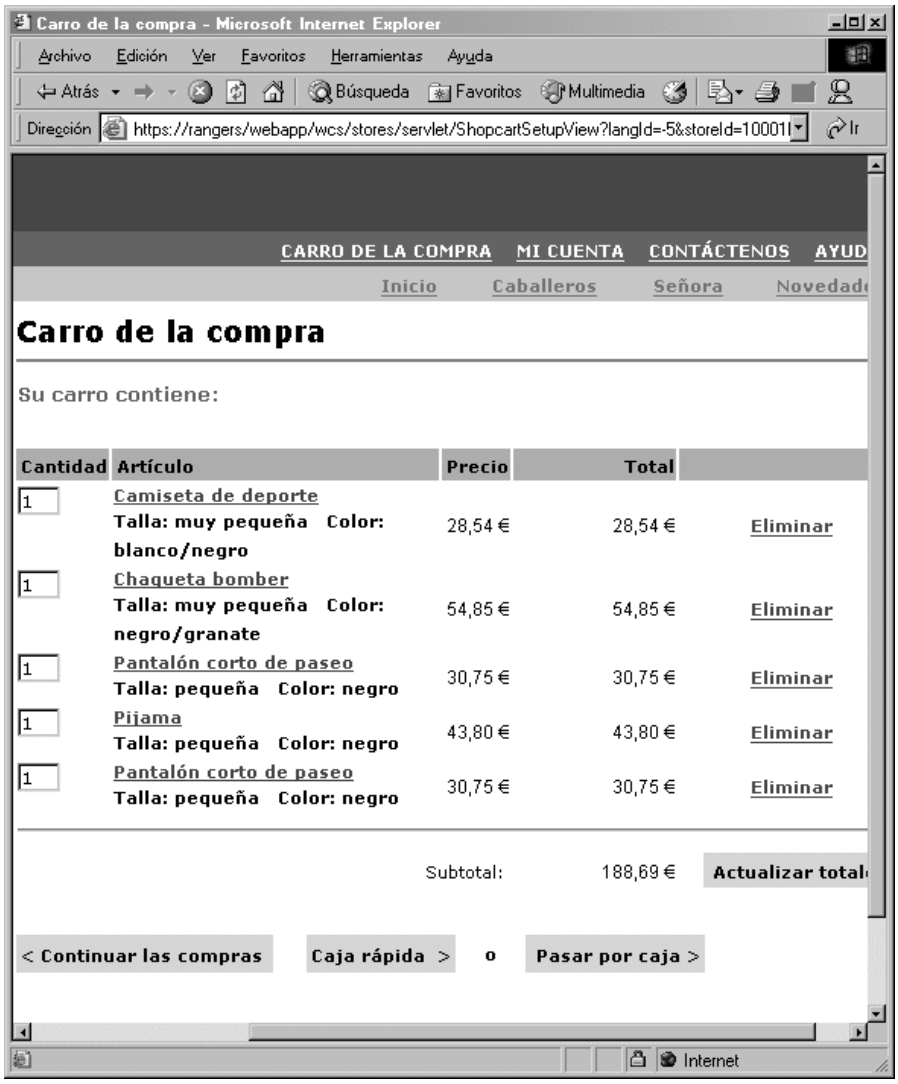


Figura 47.

3. Pulse **Continuar las compras** y seleccione otro artículo. Para este artículo seleccionado, pulse **Añadir al carro de la compra**. Aparecerá la siguiente página de error:

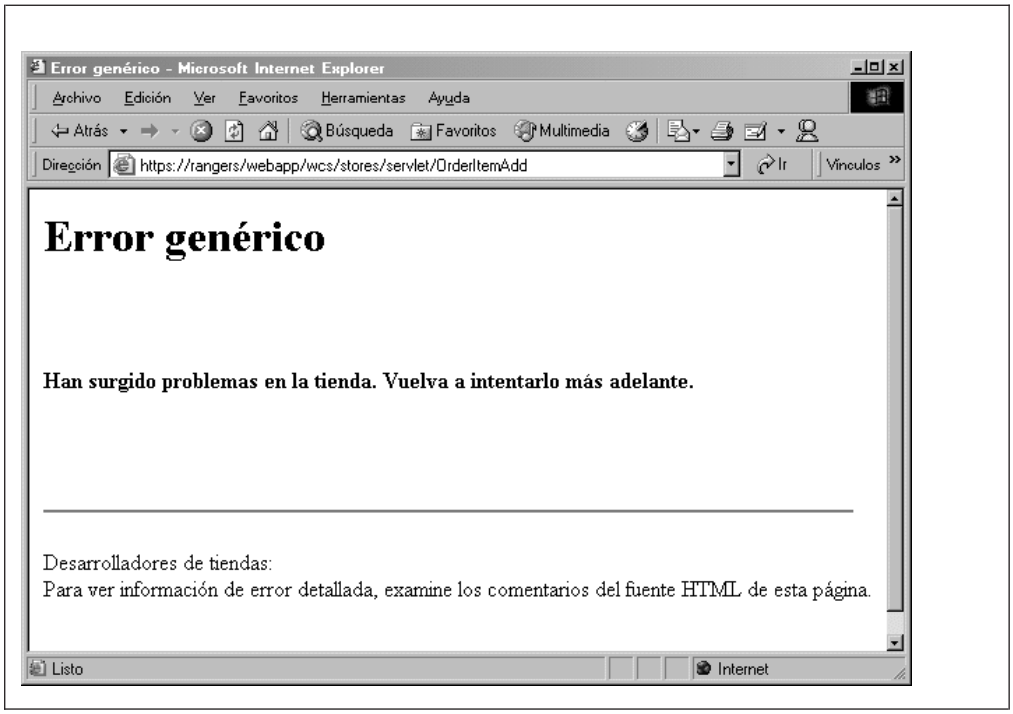


Figura 48.

Visualice el código fuente para la página de error. Desplácese hacia abajo en el código fuente y encontrará que la clave de mensaje para el error es _ERR_TOO_MANY_ITEMS.

Capítulo 12. Guía de aprendizaje: Ampliación del modelo de objeto y modificación de un mandato de tarea existente

En esta guía de aprendizaje, se encargará de un requisito para reunir información sobre las tarjetas de regalo para los pedidos. La información que se debe reunir incluye el nombre del destinatario, el nombre del remitente y dos mensajes. La información debe reunirse cuando el cliente someta el pedido.

Dado que la información para la tarjeta de regalo debe almacenarse en la base de datos, está claro que se necesita una nueva tabla de base de datos. Puesto que esto es una aplicación del proceso de pedido, existen dos modos posibles de modificar el modelo de objeto. Normalmente, puede modificar un bean de entidad público de WebSphere Commerce existente y correlacionar los campos nuevos del bean modificado con la nueva tabla o puede crear un bean de entidad nuevo que se correlacione directamente con la nueva tabla. Dado que este planteamiento necesita que se amplíe el bean de entidad de pedido y este bean utiliza un tipo de consulta de SQL "find for update", no puede ampliar el bean de esta manera. De este modo, en este caso deberá crear un bean de entidad nuevo que se correlacione con la nueva tabla.

Además del nuevo bean de entidad, se amplía el mandato de tarea ExtOrderProcessCmdImpl existente. La extensión se utiliza para crear una instancia de un nuevo bean de datos que corresponda a la nueva tabla y se utiliza para actualizar la información sobre regalo en la base de datos.

Esta guía de aprendizaje incluye las tareas de alto nivel siguientes:

1. Creación de la nueva tabla XORDGIFT e inserción de datos en ella
2. Creación del nuevo bean de entidad OrderGift
3. Creación del esquema XORDGIFT
4. Creación de la definición de tabla y correlación de los campos del bean de entidad OrderGift con las columnas de la tabla XORDGIFT
5. Generación del código desplegado y bean de acceso para el bean OrderGift
6. Creación del nuevo OrderGiftDataBean
7. Creación de la nueva implementación de mandato de tarea MyExtOrderProcessCmdImpl
8. Modificación de OrderSubmitForm.jsp para reunir la información de mensaje y modificación de OrderDetailDisplayForm.jsp para visualizar la información de mensaje.
9. Comprobación del código modificado

Requisitos previos

Esta guía de aprendizaje no requiere que haya completado las guías de aprendizaje. Si ha completa dichas guías de aprendizaje, no hay ningún peligro en dejar el código en el espacio de trabajo, puesto que dicho código no está en conflicto con esta guía de aprendizaje.

Antes de empezar esta guía de aprendizaje, tendrá que haber publicado una tienda basada en la tienda de ejemplo FashionFlow. En esta tienda, debe poder realizar

una compra (por ejemplo, examinar el catálogo, añadir artículos al carro de la compra, pasar por caja y ver la confirmación del pedido).

Creación de la nueva tabla XORDGIFT e inserción de datos en ella

En este paso, creará la tabla XORDGIFT.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para crear la tabla:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**) y pulse la pestaña **Script**.
2. En la ventana Script, entre lo siguiente:

   ```
   connect to BDdesarrollo user usuariobd using contraseñabd;
   create table XORDGIFT (ORDERSID bigint not null, RECEIPTNAME varchar(50),
   SENDERNAME varchar(50), MSGFIELD1 varchar(50), MSGFIELD2 varchar(50),
   constraint p_xordgift primary key (ORDERSID),
   constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
   on delete cascade);
   insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
   ```

   donde

   - *BDdesarrollo* es el nombre de la base de datos de desarrollo
   - *usuariobd* es el usuario de base de datos
   - *contraseñabd* es la contraseña para el usuario de base de datos

   Pulse el icono Ejecutar.
   La tabla XORDGIFT ya se ha creado.

Oracle Si está utilizando una base de datos Oracle, realice 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 SQL Plus, entre las sentencias SQL siguientes:

   ```
   create table XORDGIFT (ORDERSID NUMBER NOT NULL, RECEIPTNAME VARCHAR(50),
   SENDERNAME VARCHAR(50), MSGFIELD1 VARCHAR(50), MSGFIELD2 VARCHAR(50),
   constraint p_xordgift primary key (ORDERSID),
   constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
   on delete cascade);
   insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
   ```

   y pulse Intro para ejecutar la sentencia SQL. La tabla XORDGIFT ya se ha creado.

   Nota: Si alguien ha ejecutado anteriormente este ejemplo utilizando esta base de datos, deberá emitir el mandato siguiente antes de crear la tabla XORDGIFT:

   ```
   drop table XORDGIFT;
   ```

6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

y pulse Intro para ejecutar la sentencia SQL.

Creación del bean de entidad OrderGift

Una vez que se haya creado la tabla de base de datos, estará preparado para empezar a crear el nuevo bean de entidad. Los pasos siguientes utilizan WebSphere Studio Application Developer para crear este bean.

A continuación, cree el nuevo bean OrderGift, realizando lo siguiente:

1. Inicie entorno de desarrollo de WebSphere Commerce de este modo:
   - Business Professional **Inicio > Programas > IBM WebSphere Commerce Studio > entorno de desarrollo de WebSphere Commerce**
   - Express Express **Inicio > Programas > IBM WebSphere - Express Developer Edition > entorno de desarrollo de WebSphere Commerce**
2. Abra la perspectiva J2EE.
3. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
4. Pulse con el botón derecho del ratón en el módulo **WebSphereCommerceServerExtensionsData** y seleccione **Nuevo > Bean enterprise**.
   Se abrirá el asistente para la Creación de bean enterprise.
5. En la lista desplegable **Proyecto EJB** seleccione **WebSphereCommerceServerExtensionsData** y pulse **Siguiente**.
6. En la ventana Crear un bean enterprise, realice lo siguiente:
   a. Seleccione **Bean de entidad con campos CMP (Persistencia gestionada por contenedor)**
   b. En el campo **Nombre de bean**, entre `OrderGift`.
   c. En el campo **Carpeta fuente**, deje el valor por omisión que está especificado (ejbModule).
   d. En el campo **Paquete por omisión**, entre `com.ibm.commerce.extension.objects`.
   e. Pulse **Siguiente**.
7. En la ventana Atributos CMP, realice lo siguiente:
   a. Pulse **Añadir** para añadir campos nuevos para las columnas siguientes de la tabla:
      - ORDERSID
      - RECEIPTNAME
      - SENDERNAME
      - MSGFIELD1
      - MSGFIELD2

      Se abrirá la ventana Crear atributo CMP. En esta ventana, realice lo siguiente:
      1) Cree el campo ordersId, del modo siguiente:

Tabla 12.

Nombre de parámetro	Valor de parámetro
Nombre	`ordersId`
Tipo	`java.lang.Long` **Nota:** Debe utilizar el tipo de datos *java.lang.Long*, no el tipo de datos *long*.

Tabla 12. (continuación)

Nombre de parámetro	Valor de parámetro
Campo de clave	Select

Pulse **Aplicar**.

2) Cree el campo receiptName, del modo siguiente:

Tabla 13.

Nombre de parámetro	Valor de parámetro
Nombre	`receiptName`
Tipo	`java.lang.String`
Acceso con métodos get y set	Select
Promocionar métodos get y set a interfaz remota	Clear

Pulse **Aplicar**.

3) Cree el campo senderName, del modo siguiente:

Tabla 14.

Nombre de parámetro	Valor de parámetro
Nombre	`senderName`
Tipo	`java.lang.String`
Acceso con métodos get y set	Select
Promocionar métodos get y set a interfaz remota	Clear

Pulse **Aplicar**.

4) Cree el campo msgField1, del modo siguiente:

Tabla 15.

Nombre de parámetro	Valor de parámetro
Nombre	`msgField1`
Tipo	`java.lang.String`
Acceso con métodos get y set	Select
Promocionar métodos get y set a interfaz remota	Clear

Pulse **Aplicar**.

5) Cree el campo msgField2, del modo siguiente:

Tabla 16.

Nombre de parámetro	Valor de parámetro
Nombre	`msgField2`
Tipo	`java.lang.String`
Acceso con métodos get y set	Select
Promocionar métodos get y set a interfaz remota	Clear

Pulse **Aplicar**.

6) Pulse **Cerrar** para cerrar esta ventana.

b. Elimine la marca del recuadro de selección **Utilizar el tipo de atributo de clave única para la clase de clave** y, a continuación, pulse **Siguiente**.

8. En la ventana Detalles de clase Java EJB, realice lo siguiente:

a. Para seleccionar la superclase del bean, pulse **Examinar**.
   Se abrirá la ventana Selección de tipo.
b. En el campo **Seleccionar una clase utilizando: (cualquiera)**, entre `ECEntityBean` y pulse **Aceptar**. Esto selecciona com.ibm.commerce.base.objects.ECEntityBean como superclase.
c. Pulse **Finalizar**.

Establezca el nivel de aislamiento para el nuevo bean, realizando lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Efectúe una doble pulsación en el proyecto **WebSphereCommerceServerExtensionsData** para abrirlo con el Editor de descriptor de despliegue.
3. Pulse la pestaña **Acceso**.
4. Pulse **Añadir** junto al recuadro de texto Nivel de aislamiento.
   Se abrirá la ventana Añadir nivel de aislamiento.
5. DB2 Seleccione **Lectura repetible** y, a continuación, pulse **Siguiente**.
   Oracle Seleccione **Comprometido para lectura** y, a continuación, pulse **Siguiente**.
6. Seleccione el bean **OrderGift** y, a continuación, pulse **Siguiente**.
7. Seleccione **OrderGift** para seleccionar todos sus métodos y pulse **Finalizar**.
8. Guarde el trabajo (Control+S).

A continuación, establezca la identidad de seguridad del bean realizando lo siguiente:

1. En el Editor del descriptor de despliegue, seleccione la pestaña Acceso.
2. Pulse **Añadir** junto al recuadro de texto Identidad de seguridad (Nivel de método).
   Se abrirá la ventana Añadir identidad de seguridad.
3. Seleccione **Utilizar identidad de servidor EJB** y, a continuación, pulse **Siguiente**.
4. Seleccione el bean **OrderGift** y, a continuación, pulse **Siguiente**.
5. Seleccione **OrderGift** para seleccionar todos sus métodos y pulse **Finalizar**.
6. Guarde el trabajo (Control + S) y mantenga abierto el editor.

A continuación, establezca el rol de seguridad para los métodos del bean, realizando lo siguiente:

1. En el editor del Descriptor de despliegue, seleccione la pestaña Descriptor de ensamblado.
2. En la sección Permisos de método, pulse **Añadir**.
3. Seleccione **WCSecurityRole** como el rol de seguridad y pulse **Siguiente**.
4. En la lista de beans encontrados, seleccione **OrderGift** y pulse **Siguiente**.
5. En la página Elementos de método, pulse **Aplicar a todo** y, a continuación, pulse **Finalizar**.
6. Guarde el trabajo (Control + S) y cierre el editor de descriptor de despliegue.

El siguiente paso consiste en eliminar los campos y métodos relacionados con el contexto de entidad que WebSphere Studio Application Developer genera. La razón por la que es necesario suprimir estos campos es que la clase base

ECEntityBean proporciona su propia implementación de estos métodos. Para suprimir los campos y métodos de contexto de entidad generados, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda el proyecto **WebSphereCommerceServerExtensionsData**.
2. Expanda el módulo EJB **OrderGift** y, a continuación, efectúe una doble pulsación en **OrderGiftBean**.
3. En la vista de Esquema, realice lo siguiente:

   Nota: Express WebSphere Studio Application Developer 5.1 le solicita si desea suprimir getEntityContext() y setEntityContext(EntityContext), con lo que no tendrá que suprimirlos manualmente, tal como se ha mencionado anteriormente. Asegúrese de seleccionar su supresión.

   a. Pulse con el botón derecho del ratón en el campo **myEntityCtx** y seleccione **Suprimir**.
   b. Pulse con el botón derecho del ratón en el método **getEntityContext()** y seleccione **Suprimir**.
   c. Pulse con el botón derecho del ratón en el método **setEntityContext(EntityContext)** y seleccione **Suprimir**.
   d. Pulse con el botón derecho del ratón en el método **unsetEntityContext()** y seleccione **Suprimir**.
4. Guarde el trabajo (Control+S).

Deberá modificar el método ebjCreate generado para que todos los parámetros se establezcan explícitamente al crear un nuevo bean OrderGift, del modo siguiente:

1. En la vista de Jerarquía J2EE, efectúe una doble pulsación en la clase **OrderGiftBean** para abrirla y ver el código fuente.
2. En la vista de Esquema, seleccione el método ejbCreate(Long). El código fuente aparece de la siguiente manera:

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId)
      throws javax.ejb.CreateException {
      _initLinks();
      this.ordersId = ordersId;
      return null;
}
```

3. Modifique el código para que aparezca del modo siguiente:

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId)
      throws javax.ejb.CreateException {
      _initLinks();
      this.ordersId = ordersId;
      this.senderName = null;
      this.receiptName = null;
      this.msgField1 = null;
      this.msgField2 = null;
      return null;
}
```

4. Guarde los cambios de código.

A continuación, añada un nuevo método ejbCreate al bean OrderGift, realizando lo siguiente:

1. En la vista de Jerarquía J2EE, efectúe una doble pulsación en la clase **OrderGiftBean** para abrirla y ver el código fuente.

2. Cree un nuevo método ejbCreate(Long, String, String, String, String), añadiendo el código siguiente en la clase:

```
public com.ibm.commerce.extension.objects.OrderGiftKey
   ejbCreate(java.lang.Long ordersId,
   java.lang.String receiptName,
   java.lang.String senderName,
   java.lang.String msgField1,
   java.lang.String msgField2)
      throws javax.ejb.CreateException {
      _initLinks();
      this.ordersId = ordersId;
      this.senderName = senderName;
      this.receiptName = receiptName;
      this.msgField1 = msgField1;
      this.msgField2 = msgField2;
      return null;
 }
```

3. Guarde los cambios de código.
4. Debe añadir este nuevo método ejbCreate en la interfaz inicial. Esto hará que el método esté disponible en el bean de acceso generado. Para añadir el método a la interfaz inicial, realice lo siguiente:
   a. En la vista de Esquema, pulse con el botón derecho del ratón en el método **ejbCreate(Long, String, String, String, String)** y seleccione **Bean enterprise > Promocionar a interfaz inicial**.

A continuación, cree un método ejbPostCreate(Long, String, String, String, String) nuevo de forma que tenga los mismos parámetros que el método ejbCreate(Long, String, String, String, String), realizando lo siguiente:

1. Efectúe una doble pulsación en la clase **OrderGiftBean** para abrirla y ver el código fuente.
2. Cree un método ejbPostCreate(Long ordersId, String receiptName, String senderName, String msgField1, String msgField2) nuevo, añadiendo el código siguiente en la clase:

```
public void ejbPostCreate(
   java.lang.Long ordersId,
   java.lang.String receiptName,
   java.lang.String senderName,
   java.lang.String msgField1,
   java.lang.String msgField2)
      throws javax.ejb.CreateException {
   }
```

3. Guarde los cambios de código.

A continuación, debe crear la definición para la tabla XORDGIFT. Si *no* ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, realice lo siguiente para crear la definición de tabla XORDGIFT

1. Abra la perspectiva de Datos y conmute la vista de Definición de datos.
2. Vaya al directorio siguiente:
   WebSphereCommerceServerExtensionsData > ejbModule > META-INF
3. Pulse con el botón derecho del ratón en **META-INF** y seleccione **Definición de base de datos nueva**.
   Se abrirá el asistente para la Definición de base de datos nueva.
4. En el campo **Nombre de base de datos**, entre el nombre de la base de datos de desarrollo. Por ejemplo, entre `Demo_Dev`.

5. En la lista desplegable de tipo de proveedor de base de datos, seleccione el tipo de base de datos apropiado para el entorno de desarrollo:

   DB2

   - Business Professional **DB2 Universal Database V8.1**
   - Express **DB2 Universal Database V8.1 Express**

   Oracle **Oracle 9i**
6. Pulse **Finalizar**. La nueva definición de base de datos ya se ha creado.
7. Vaya al directorio siguiente: **WebSphereCommerceServerExtensionsData > ejbModule > META-INF > Esquema >** *BDdesarrollo*.
8. Pulse con el botón derecho del ratón en *BDdesarrollo* y seleccione **Nuevo > Definición de esquema nuevo**.
   Se abrirá el asistente para la Definición de esquema nuevo.
9. En el campo **Nombre de esquema**, entre `NULLID` y pulse **Finalizar**.
10. Vaya al directorio **WebSphereCommerceServerExtensionsData > ejbModule > META-INF > Esquema >** *BDdesarrollo* **> NULLID > Tablas**.
11. Pulse con el botón derecho del ratón en **Tablas** y seleccione **Definición de tabla nueva**.
    Se abrirá el asistente para la Definición de tabla nueva.
12. En el campo **Nombre de tabla**, entre `XORDGIFT` y pulse **Siguiente**.
13. Añada la columna de clave en la definición de tabla, del modo siguiente:

    Business Professional

    a. Pulse **Añadir otra**.
    b. En el campo **Nombre de columna**, entre `ORDERSID`.
    c. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:

       DB2 `BIGINT`

       Oracle `NUMBER`
    d. Seleccione **Columna de clave**.
    e. Oracle En el campo **Precisión numérica**, entre `38`.
    f. Oracle Deje el valor para **Escala numérica** en `0`.

    Express

    a. Pulse **Añadir otra**.
    b. En el campo **Nombre de columna**, entre `ORDERSID`.
    c. Seleccione **Columna de clave**.
    d. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:

       DB2 `BIGINT`

       Oracle `NUMBER`
    e. Oracle En el campo **Precisión numérica**, entre `38`.
    f. Oracle Deje el valor para **Escala numérica** en `0`.
14. Añada columnas adicionales en la definición de tabla, del modo siguiente:
    a. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 17.

Propiedad	Valor
Nombre de columna	`RECEIPTNAME`

Tabla 17. (continuación)

Propiedad	Valor
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

b. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 18.

Propiedad	Valor
Nombre de columna	SENDERNAME
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

c. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 19.

Propiedad	Valor
Nombre de columna	MSGFIELD1
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

d. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 20.

Propiedad	Valor
Nombre de columna	MSGFIELD2
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

e. Pulse **Finalizar**.

f. Oracle Deberá editar la definición de tabla utilizando un editor de texto, como se indica a continuación:

1) Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.

2) Expanda el proyecto **WebSphereCommerceServerExtensionsData**.
3) Expanda lo siguiente: **ejbModule > META-INF >Esquema**.
4) Pulse con el botón derecho del ratón en el archivo **WebSphereCommerceServerExtensionsData_NULL_XORDGIFT. xmi** y seleccione **Abrir con > Editor de texto**.
5) Sustituya todas las apariciones de `SQLNumeric_6` por `SQLNumeric_3`.
6) Guarde los cambios y cierre el editor de texto.

Si ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, realice lo siguiente para crear la definición de tabla XORDGIFT:

1. Abra la perspectiva de Datos y conmute la vista de Definición de datos.
2. Vaya al directorio siguiente: **WebSphereCommerceServerExtensionsData > ejbModule > META-INF > Schema >** *BDdesarrollo* **> NULLID > Tablas**
3. Pulse con el botón derecho del ratón en el directorio **Tablas** y seleccione **Nueva > Definición de tabla nueva**.
   Se abrirá el asistente para la Definición de tabla nueva.
4. En el campo **Nombre de tabla**, entre `XORDGIFT` y pulse **Siguiente**.
5. Añada la columna de clave en la definición de tabla, del modo siguiente:

   Business Professional

   a. Pulse **Añadir otra**.
   b. En el campo **Nombre de columna**, entre `ORDERSID`.
   c. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:

      DB2 `BIGINT`

      Oracle `NUMBER`

   d. Seleccione **Columna de clave**.
   e. Oracle En el campo **Precisión numérica**, entre 38.
   f. Oracle Deje el valor para **Escala numérica** en `0`.

   Express

   a. Pulse **Añadir otra**.
   b. En el campo **Nombre de columna**, entre `ORDERSID`.
   c. Seleccione **Columna de clave**.
   d. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:

      DB2 `BIGINT`

      Oracle `NUMBER`

   e. Oracle En el campo **Precisión numérica**, entre 38.
   f. Oracle Deje el valor para **Escala numérica** en `0`.
6. Añada columnas adicionales en la definición de tabla, del modo siguiente:
   a. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 21.

Propiedad	Valor
Nombre de columna	`RECEIPTNAME`
Tipo de columna	DB2 `VARCHAR` Oracle `VARCHAR2`

Tabla 21. (continuación)

Propiedad	Valor
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

b. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 22.

Propiedad	Valor
Nombre de columna	SENDERNAME
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

c. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 23.

Propiedad	Valor
Nombre de columna	MSGFIELD1
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

d. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 24.

Propiedad	Valor
Nombre de columna	MSGFIELD2
Tipo de columna	DB2 VARCHAR Oracle VARCHAR2
Con posibilidad de nulos	Select
Longitud de serie	50
Para datos de bit	Clear

e. Pulse **Finalizar**.

7. Oracle Deberá editar la definición de tabla utilizando un editor de texto, como se indica a continuación:
   a. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
   b. Expanda el proyecto **WebSphereCommerceServerExtensionsData**.
   c. Expanda lo siguiente: **ejbModule > META-INF > Esquema**.

d. Pulse con el botón derecho del ratón en el archivo **WebSphereCommerceServerExtensionsData_NULL_XORDGIFT.xmi** y seleccione **Abrir con > Editor de texto**.
e. Sustituya todas las apariciones de `SQLNumeric_6` por `SQLNumeric_3`.
f. Guarde los cambios y cierre el editor de texto.

El paso siguiente consiste en correlacionar la tabla XORDGIFT con el bean de entidad OrderGiftBean. Dado que la base de datos de desarrollo y la tabla XORDGIFT ya existen, se utiliza la correlación de Encuentro a medio camino.

Si no ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, realice lo siguiente para crear la correlación:

1. En la vista de Jerarquía J2EE, pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Correlación entre EJB y RDB**.
   Se abrirá el asistente para Crear una correlación EJB/RDB nueva.
2. Seleccione **Encuentro a medio camino** y pulse **Siguiente**.
3. Seleccione **Emparejar por nombre y tipo** y pulse **Finalizar**.
   Se abrirá el editor Map.mapxmi.
4. En el panel Beans enterprise, expanda el bean **OrderGift**. En el panel Tablas, expanda la tabla **XORDGIFT**.
5. Correlacione los campos del bean Bonus con las columnas de la tabla XORDGIFT, realizando lo siguiente:
   a. Pulse con el botón derecho del ratón en el bean **OrderGift** y seleccione **Emparejar por nombre**.
6. Guarde el archivo Map.mapxmi pulsando Control + S. Cierre el archivo.

Si ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, realice lo siguiente para crear la correlación:

1. En la vista de Jerarquía J2EE, pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Correlación entre EJB y RDB**.
   Se abrirá el editor Map.mapxmi.
2. En el panel Beans enterprise, expanda el bean **OrderGift**. En el panel Tablas, expanda la tabla **XORDGIFT**.
3. Correlacione los campos del bean Bonus con las columnas de la tabla XORDGIFT, realizando lo siguiente:
   a. Pulse con el botón derecho del ratón en el bean **OrderGift** y seleccione **Emparejar por nombre**.
4. Guarde el archivo Map.mapxmi pulsando Control + S. Cierre el archivo.

Una vez que se haya creado la entidad OrderGiftBean y que el esquema esté correlacionado correctamente, podrá crear un bean de acceso para el bean de entidad. Este bean de acceso simplifica el acceso de las aplicaciones a la información contenida en el bean de entidad OrderGift. Las herramientas de WebSphere Studio Application Developer 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 el bean de entidad OrderGift, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** y, a continuación, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData**

y seleccione **Nuevo > Bean de acceso**.
Se abrirá la ventana Añadir un bean de acceso.

2. Seleccione **Ayudante de copia** y pulse **Siguiente**.
3. Seleccione el bean **OrderGift** y pulse **Siguiente**.
4. En la lista desplegable **Método constructor**, seleccione **findByPrimaryKey(com.ibm.commerce.extension.objects.OrderGiftKey)**.
5. Seleccione todos los atributos en la sección Ayudantes de atributo.
6. Pulse **Finalizar**.

Para ver el código que se acaba de generar, vaya a la vista de Business Professional Navegador J2EE Express Navegador de proyectos, expanda el proyecto **WebSphereCommerceServerExtensionsData**, expanda la subcarpeta **ejbModule** y, a continuación, expanda **com.ibm.commerce.extension.objects**. En el interior del paquete se crean y se visualizan una nueva clase denominada OrderGiftAccessBean y una nueva interfaz denominada OrderGiftAccessBeanData.

El siguiente paso consiste en generar el código desplegado.

El programa de utilidad de generación de código analiza los beans para asegurar que se cumplan las especificaciones de EJB de Sun Microsystems y comprueba que se sigan las normas específicas del 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 inicial y remota, así como las clases de buscador y persistencia JDBC para los beans CMP. También genera las clases Java ORB, de ″apéndices″ y de relación necesarias para el acceso RMI a través de IIOP, así como ″apéndices″ para las interfaces inicial y remota.

Para generar el código desplegado, efectúe lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB** y, a continuación, pulse con el botón derecho del ratón en **WebSphereCommerceServerExtensionsData** y seleccione **Generar > Código de despliegue y RMIC**.
Se abrirá la ventana Código de despliegue y RMIC.
2. Seleccione **OrderGift** y pulse **Finalizar**.

Puede ver el código que se acaba de generar yendo a la vista del Business Professional Navegador J2EE Express Navegador de proyectos. Encontrará lo siguiente:

Tabla 25.

Tipo de código	Nombre de clase
Código generado por implementación de contenedor	EJSCMPOrderGiftHomeBean.java
	EJSRemoteCMPOrderGift.java
	EJSRemoteCMPOrderGiftHome.java
	EJSFinderOrderGiftBean.java
Código de acceso JDBC	EJSJDBCPersisterCMPOrderGiftBean.java
Código de apéndices y relación RMI	_EJSRemoteCMPOrderGift_Tie.java
	_OrderGift_Stub.java
	_EJSRemoteCMPOrderGiftHome_Tie.java
	_OrderGiftHome_Stub.java

Integración del bean de entidad OrderGift en el flujo de compra

En esta sección, integrará el bean de entidad OrderGift en el flujo de compra normal de la tienda de ejemplo, realizando las acciones siguientes:

1. Creación de un nuevo bean de datos OrderGiftDataBean
2. Creación de un nuevo mandato de tarea MyExtOrderProcessCmdImpl
3. Modificación de la página de visualización OrderSubmitForm.jsp

Cada uno de los pasos anteriores se describe detalladamente en las secciones siguientes.

Creación de OrderGiftDataBean

Debe crear OrderGiftDataBean para que los atributos del bean de entidad OrderGift puedan visualizarse en una página de visualización JSP. Como sucede con otras partes de la guía de aprendizaje, se proporciona el código base y es necesario eliminar los signos de comentario de diversas secciones de código.

Para crear OrderGiftDataBean, realice lo siguiente:

1. El primer paso es importar el código base para el nuevo bean de datos, del modo siguiente:
   a. Asegúrese de que ha completado los pasos del apartado "Localización del código de ejemplo" en la página 192.
   b. Cambie a la perspectiva de J2EE y, a continuación, seleccione la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
   c. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
   d. Pulse con el botón derecho del ratón en la carpeta **src** y seleccione **Importar**.
      Se abrirá el asistente para Importar.
   e. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
   f. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
      suDirectorio\WC_SAMPLE_55.zip
      donde *suDirectorio* es el directorio en el que ha bajado el paquete.
   g. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione el archivo siguiente para importarlo.
      - com\ibm\commerce\sample\databeans\OrderGiftDataBean.java
   h. En el campo **Carpeta**, la carpeta `WebSphereCommerceServerExtensionsLogic/src` ya está especificada. Conserve este valor.
   i. Pulse **Finalizar**.

Una vez que haya importado el código para el bean, examine el código.

Creación de la clase MyExtOrderProcessCmdImpl

En esta sección, añadirá lógica nueva al final del proceso de negocio OrderProcess para que la información relacionada con el pedido de regalo se actualice en la tabla de base de datos XORDGIFT. Se proporciona el mandato ExtOrderProcessCmdImpl como punto de ampliación a este proceso de negocio. Siguiendo el modelo de programación, para ampliar esta lógica, deberá crear una nueva clase de implementación del mandato de tarea e incluir la nueva lógica en esta clase.

Entonces deberá actualizar el registro de mandatos para asociar la nueva clase de implementación con la interfaz ExtOrderProcessCmd.

Para crear la clase MyExtOrderProcessCmdImpl, realice lo siguiente:

1. El primer paso es importar el código base para el nuevo mandato, como se indica a continuación:
   a. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
   b. Pulse con el botón derecho del ratón en la carpeta **src** y seleccione **Importar**.
      Se abrirá el asistente para Importar.
   c. Como fuente de importación, seleccione **Archivo zip** y pulse **Siguiente**.
   d. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
   e. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
      suDirectorio\WC_SAMPLE_55.zip
      donde *suDirectorio* es el directorio en el que ha bajado el paquete.
   f. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione el archivo siguiente para importarlo.
      - com\ibm\commerce\sample\commands\MyExtOrderProcessCmdImpl.java
   g. En el campo **Carpeta**, la carpeta `WebSphereCommerceServerExtensionsLogic/src` ya está especificada. Conserve este valor.
   h. Pulse **Finalizar**.

Una vez que haya importado el código para este nuevo mandato, puede examinar el código fuente para ver qué hace el mandato. Observe que el mandato llama al método performExecute() de la superclase para asegurar que se ejecute cualquier proceso del mandato. Entonces, incluye la lógica para establecer la información de pedido de regalo en el nuevo bean de datos.

En este paso, deberá modificar el registro de mandatos para que se utilice la nueva clase de implementación MyExtOrderProcessCmdImpl en lugar de la clase de implementación ExtOrderProcessCmdImpl original. La única tabla del registro de mandatos que es necesario modificar es la tabla CMDREG. En este caso, se utiliza la nueva clase de implementación para todas las tiendas.

Para modificar el registro de mandatos, realice lo siguiente:

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyExtOrderProcessCmdImpl:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**).
2. En el menú **Herramientas**, seleccione **Valores de herramientas**.
3. Marque el recuadro de selección **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (**;**)
4. Cierre los valores de herramientas.
5. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla URLREG indicando la siguiente información en la ventana de script:

```
connect to BDdesarrollo user usuariobd using contraseñabd;
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FashionFlow,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

donde

- *BDdesarrollo* es el nombre de la base de datos de desarrollo
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña del usuario de base de datos
- *ID_enttienda_FashionFlow* es el identificador de tienda para la tienda de ejemplo

Pulse el icono **Ejecutar**.

Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyOrderItemAddCmdImpl:

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 SQL:

   ```
   insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
      CLASSNAME, TARGET)
   values (ID_enttienda_FashionFlow,
      'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
      'This is a new task command for tutorial two.',
      'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
      'local');
   ```

   Pulse Intro para ejecutar la sentencia SQL.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia SQL.

Compilación de cambios

Importante: Asegúrese de que no vuelve a crear el proyecto web Stores.

En esta sección, compilará los cambios que ha realizado en el código, como se indica a continuación:

1. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, seleccione los siguientes proyectos:
   - WebSphereCommerceServerExtensionsData
   - WebSphereCommerceServerExtensionsLogic
2. Con los proyectos anteriores resaltados, pulse con el botón derecho del ratón en **Crear proyecto** y selecciónelo.

Modificación de las páginas de visualización para los mensajes de regalo

En este paso, modificará las plantillas OrderSubmitForm y OrderDetailDisplay de modo que el cliente pueda entrar información de mensaje para el pedido de regalo así como ver la información en una página de resumen. La estrategia para modificar estas páginas es incluir plantillas JSP adicionales que especifiquen la nueva información para la página. Estas páginas nuevas (OrderSubmitFormInclude.jsp y OrderDetailDisplayInclude.jsp) utilizan JSTL para visualizar la nueva información.

Para modificar las páginas de visualización, realice lo siguiente:

1. Cambie a la perspectiva de Web y utilice la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Si no ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, deberá modificar las propiedades del proyecto web Stores, del modo siguiente:
   a. Pulse con el botón derecho del ratón en el proyecto Web **Stores** y seleccione **Propiedades**.
   b. Business Professional Seleccione **Web** en el panel izquierdo y, a continuación, en la lista Características de proyecto Web disponibles, seleccione **Incluir la biblioteca de códigos estándar JSP**.

      Express Seleccione **Características de proyecto Web** en el panel izquierdo y, a continuación, en la lista Características de proyecto Web disponibles, seleccione **Biblioteca de códigos estándar JSP**.

      Pulse **Aplicar**. Cuando la actualización se haya completado, pulse **Aceptar** para cerrar el editor de propiedades.
3. Expanda el directorio siguiente:
   Stores\Web Content*nombre_FashionFlow*.
4. Cree una copia de seguridad del archivo OrderSubmitForm.jsp, realizando lo siguiente:
   a. Expanda los directorios **ShoppingArea>CheckoutSection>StandardCheckoutSubsection**.
   b. Pulse con el botón derecho del ratón en el archivo **OrderSubmitForm.jsp** y seleccione **Redenominar**.
   c. En la ventana Redenominar, entre `OrderSubmitForm_bak.jsp` y pulse **Aceptar**.
   d. Cuando se le solicite si desea actualizar los enlaces a este archivo, pulse **No**.
5. Cree una copia de seguridad del archivo OrderDetailDisplay.jsp, realizando lo siguiente:
   a. Expanda los directorios **UserArea>ServiceSection>TrackOrderStatusSubsection**.
   b. Pulse con el botón derecho del ratón en el archivo **OrderDetailDisplay.jsp** y seleccione **Redenominar**.
   c. En la ventana Redenominar, entre `OrderDetailDisplay_bak.jsp` y pulse **Aceptar**.
   d. Cuando se le solicite si desea actualizar los enlaces a este archivo, pulse **No**.

6. Pulse con el botón derecho del ratón en el directorio *nombre_FashionFlow* y seleccione **Importar**.
   Se abrirá el asistente para Importar.
7. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
8. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
   suDirectorio\WC_SAMPLE_55.zip
   donde *suDirectorio* es el directorio en el que ha bajado el paquete.
9. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione los archivos siguientes para importarlos.
   - ShoppingArea\CheckoutSection\StandardCheckoutSubsection\OrderSubmitForm.jsp
   - ShoppingArea\CheckoutSection\StandardCheckoutSubsection\OrderSubmitFormInclude.jsp
   - UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplay.jsp
   - UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplayInclude.jsp
10. En el campo **Carpeta**, la carpeta `Stores/Web Content/`*`nombre_FashionFlow`* ya está especificada. Conserve este valor.
11. Pulse **Finalizar**.

Si examina el archivo OrderSubmitForm.jsp, encontrará que incluye lo siguiente:

```
<!-- tutorial start-->
   <jsp:include page="OrderSubmitFormInclude.jsp" flush="true" />
<!-- tutorial done -->
```

Éste es el modo en que el nuevo archivo OrderSubmitFormInclude.jsp se incorpora en la plantilla JSP existente. Examine OrderSubmitFormInclude.jsp para ver un ejemplo de cómo utilizar JSTL en las plantillas. De forma similar, examine los archivos OrderDetailDisplay.jsp y OrderDetailDisplayInclude.jsp.

También deberá importar el archivo de propiedades que contiene los valores de serie utilizados en las plantillas JSP modificadas. Este archivo se denomina ordergift.properties. Para importar este archivo, realice lo siguiente:

1. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, expanda los siguientes directorios:
   Stores > Web Content> WEB-INF >classes > *nombre_FashionFlow*.
2. Pulse con el botón derecho del ratón en el directorio *nombre_FashionFlow* y seleccione **Importar**.
   Se abrirá el asistente para Importar.
3. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
4. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
   suDirectorio\WC_SAMPLE_55.zip
   donde *suDirectorio* es el directorio en el que ha bajado el paquete.
5. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione el archivo siguiente para importarlo.
   - ordergift.properties

6. En el campo **Carpeta**, la carpeta `Stores/Web Content/WEB-INF/classes/`*`nombre_FashionFlow`* ya está especificada. Conserve este valor.
7. Pulse **Finalizar**.

Comprobación de la nueva funcionalidad del mensaje de regalo

En esta sección, comprobará la nueva funcionalidad del mensaje de regalo, del modo siguiente:

1. Conmute a la perspectiva de Servidor (**Ventana > Abrir perspectiva > Servidor**).
2. Inicie el servidor de pagos.
3. Pulse con el botón derecho del ratón en el servidor **WebSphereCommerceServer** y seleccione **Iniciar** (o **Reiniciar**).
4. Abra un navegador Web y entre el URL siguiente:
   `http://localhost/webapp/wcs/stores/servlet/FashionFlow/index.jsp`
5. Inicie la sesión como usuario nuevo. Por ejemplo, pulse **Regístrese** y, a continuación, cree un nuevo usuario o conéctese como un usuario existente.
6. Como usuario nuevo registrado, examine la tienda, añada un artículo al carro de la compra y, a continuación, realice la compra. Podrá añadir un mensaje de regalo al pedido, como se muestra en la siguiente instantánea de pantalla (versión en inglés):

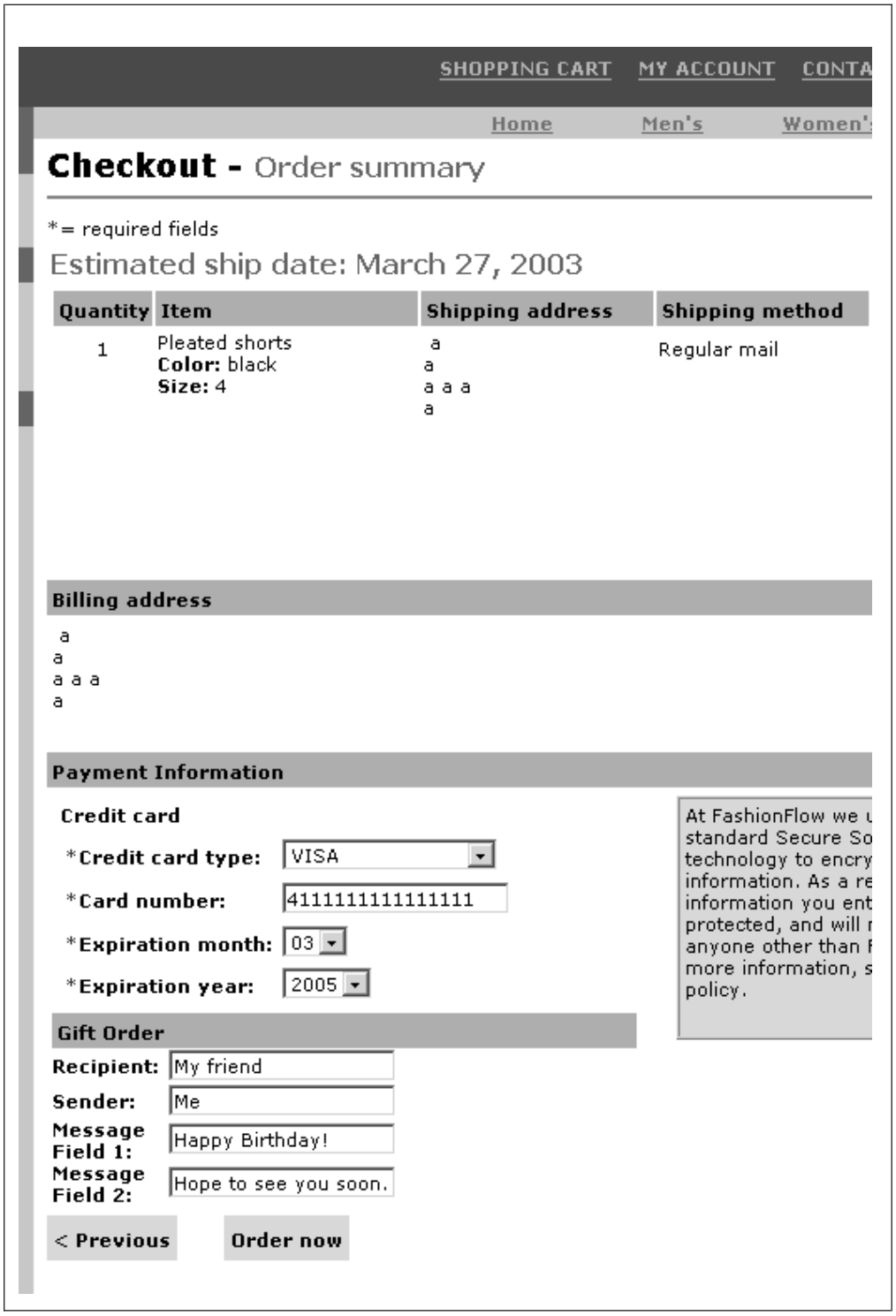


Figura 49.

Anote el número de pedido asociado con este pedido.

7. Pulse **Mi cuenta**.
8. Pulse **Ver pedidos**.
9. Seleccione el pedido que ha creado en el paso 6. Verá una pantalla similar a la siguiente:

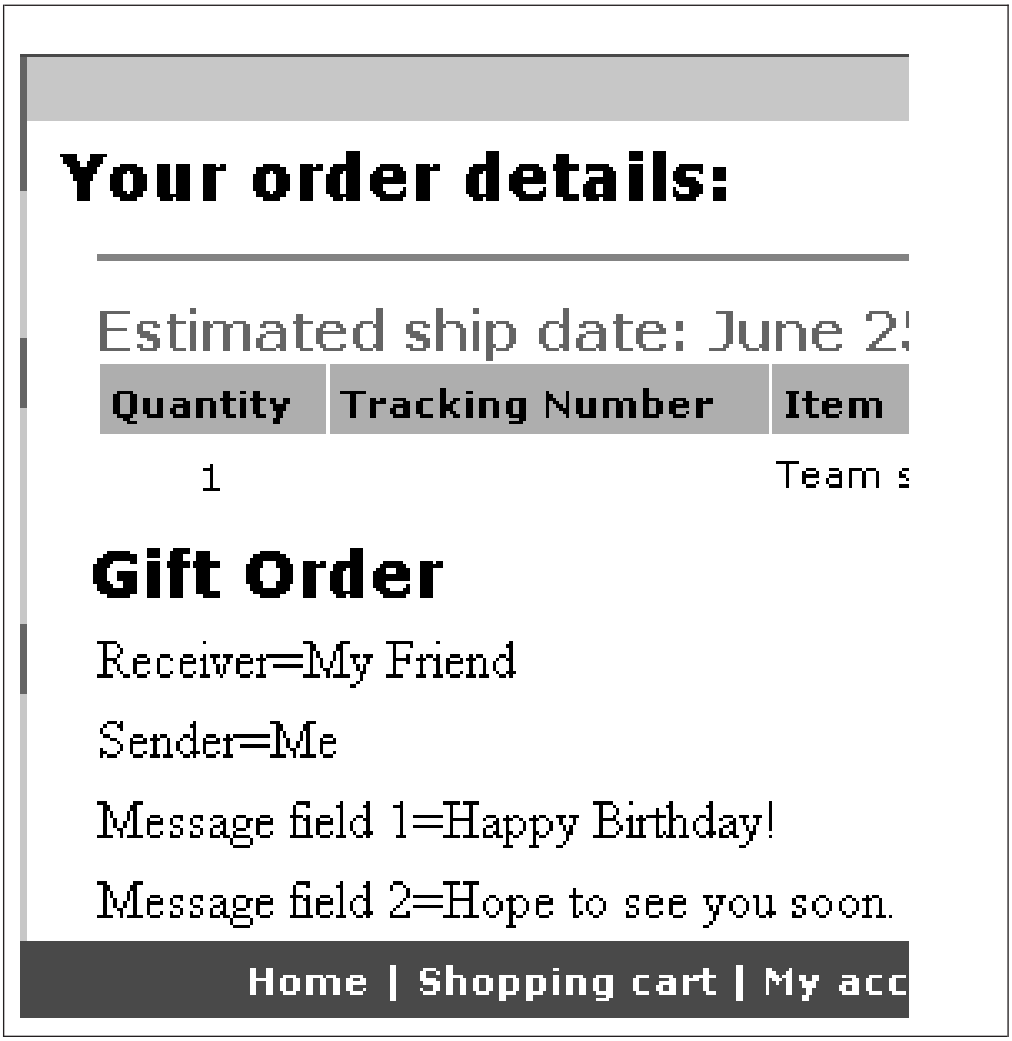


Figura 50.

Despliegue de la funcionalidad del mensaje de regalo

Esta sección describe cómo desplegar la nueva lógica de negocio en una tienda que se ejecuta en un WebSphere Commerce Server remoto. Antes de empezar estos pasos de despliegue, tiene que haber creado una tienda (basada en la tienda de ejemplo FashionFlow) en el WebSphere Commerce Server remoto. Dentro de esa tienda, debe poder completar una compra.

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.

Existen varios tipos diferentes de elementos que debe desplegar en el WebSphere Commerce Server de destino. Éstos incluyen:

- Lógica de beans de datos y mandatos de tarea
- Lógica de beans enterprise
- Plantillas JSP actualizadas

- Actualizaciones de base de datos que incluyen actualizaciones de esquema (tabla nueva) así como actualizaciones de registro de mandatos

Esta sección describe cómo desplegar todos estos elementos, *de forma incremental* en el WebSphere Commerce Server de destino. Esto se realiza como despliegue incremental, a diferencia de un despliegue del archivo EAR entero.

Creación del archivo JAR de mandatos y beans de datos

Esta sección describe cómo crear el archivo JAR que contiene la lógica de mandatos de controlador, mandatos de tarea y beans de datos.

Para crear este archivo JAR, realice los pasos siguientes en la máquina de desarrollo:

1. En el sistema de archivos local, cree un directorio denominado \ExportTemp3.
2. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
3. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Exportar**. Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR** y pulse **Siguiente**.
   b. El panel izquierdo bajo **Seleccionar los recursos para exportar** está previamente lleno con el nombre del proyecto. Deje este valor como está.
   c. En el panel derecho, asegúrese de que sólo estén seleccionados los siguientes recursos:
      - .classpath
      - .project
      - .serverPreference
   d. Asegúrese de que **Exportar archivos de clases y recursos generados** esté seleccionado.
   e. *No* seleccione **Exportar archivos fuente y recursos Java**.
   f. En el campo **Seleccionar el destino de exportación**, entre el nombre de archivo JAR totalmente calificado que se debe utilizar. En este caso, entre *unidad*:\ExportTemp3\WebSphereCommerceServerExtensionsLogic.jar. Tenga en cuenta que el nombre de archivo JAR debe ser WebSphereCommerceServerExtensionsLogic.jar.
   g. Pulse **Finalizar**.

Creación del archivo JAR EJB

Para crear el archivo JAR EJB, realice lo siguiente:

1. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda el proyecto **WebSphereCommerceServerExtensionsData**.
3. Efectúe una doble pulsación en **Descriptor de despliegue EJB**.
4. Con la pestaña Visión general seleccionada, desplácese a la parte inferior del panel para localizar la sección **Enlaces de WebSphere**.
5. En el campo **Nombre JNDI de origen de datos**, entre el nombre JNDI de origen de datos del WebSphere Commerce Server de destino. A continuación se muestra un valor de ejemplo:

DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
donde el WebSphere Commerce Server de destino está utilizando una base de datos DB2 y el nombre de instancia de WebSphere Commerce es “demo”

Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
donde el WebSphere Commerce Server de destino está utilizando una base de datos Oracle y el nombre de instancia de WebSphere Commerce es “demo”.

El valor para el nombre JNDI de origen de datos se crea añadiendo “`jdbc/`” al nombre de origen de datos del WebSphere Commerce Server de destino. Puede verificar el nombre de origen de datos abriendo el archivo *nombreInstancia*.xml del WebSphere Commerce Server de destino y buscando `DatasourceName=` en el archivo.

6. Guarde los cambios del descriptor de despliegue (Control+S).
7. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsData** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
8. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR EJB** y pulse **Siguiente**.
   b. Business Professional El valor para **¿Qué recursos desea exportar?** está previamente lleno con el nombre del proyecto EJB.
      Express El nombre del proyecto EJB ya está rellenado.
      Deje el valor que aparece.
   c. Business Professional En el campo **¿Dónde desea exportar los recursos?** entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      Express Para el destino, entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      En este caso, entre
      unidad`:\ExportTemp3\WebSphereCommerceServerExtensionsData.jar`.
   d. Pulse **Finalizar**.
9. Después de que se haya creado el archivo JAR, deshaga los cambios efectuados en el descriptor de despliegue local en el paso 5, para restaurar el valor que es necesario para el servidor de prueba local.

Exportación de elementos de tienda

Para exportar las plantillas de visualización OrderSubmitForm y OrderDetailDisplay de WebSphere Studio Application Developer, realice lo siguiente:

1. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda la carpeta **Stores**.
3. Pulse con el botón derecho del ratón en la carpeta **Web Content** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Sistema de archivos** y pulse **Siguiente**.
   b. Seleccione los recursos siguientes para desplegarlos:
      - Web Content*nombre_FashionFlow*\ShoppingArea\ CheckoutSection\StandardCheckoutSubsection\ OrderSubmitForm.jsp

- Web Content*nombre_FashionFlow*\ShoppingArea\CheckoutSection\StandardCheckoutSubsection\OrderSubmitFormInclude.jsp
- Web Content*nombre_FashionFlow*\UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplay.jsp
- Web Content*nombre_FashionFlow*\UserArea\ServiceSection\TrackOrderStatusSubsection\OrderDetailDisplayInclude.jsp
- Web Content\WEB-INF\lib\jstl.jar
- Web Content\WEB-INF\lib\standard.jar
- Web Content\WEB-INF\classes*nombre_FashionFlow*\ordergift.properties

c. Seleccione **Crear estructura de directorios para archivos**.

d. En el campo Directorio, entre un directorio temporal en el que se colocarán estos recursos. Por ejemplo, entre `C:\ExportTemp3`

e. Pulse **Finalizar**.

Transferencia de elementos al WebSphere Commerce Server de destino

En este paso, creará un directorio temporal en el WebSphere Commerce Server de destino y, a continuación, copiará los elementos de pedido de regalo en este directorio. En los pasos subsiguientes, colocará los diferentes tipos de código en el lugar apropiado de la aplicación WebSphere Commerce.

Para copiar los archivos de la máquina de desarrollo en el WebSphere Commerce Server de destino, realice lo siguiente:

1. En el WebSphere Commerce Server de destino, cree un directorio temporal denominado *unidad*:\ImportTemp3.
2. Determine cómo copiará los archivos de un sistema a otro. Puede realizar esta tarea correlacionando una unidad del WebSphere Commerce Server de destino con la máquina de desarrollo o utilizando una aplicación FTP, si la tiene configurada.
3. En la máquina de desarrollo, copie el contenido de \ExportTemp3 en \ImportTemp3 del WebSphere Commerce Server de destino.

Detención del WebSphere Commerce Server de destino

Antes de empezar los pasos de despliegue, deberá detener el WebSphere Commerce Server de destino emitiendo el mandato stopServer en la línea de mandatos. Para obtener detalles sobre este mandato, consulte la publicación Business Professional *WebSphere Commerce Studio, Guía de instalación* o Express *WebSphere Commerce - Express Developer Edition, Guía de instalación*.

Actualización de la base de datos en el WebSphere Commerce Server de destino

Registro de la implementación de mandatos de tarea

Para modificar el registro de mandatos, realice lo siguiente:

1. DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyExtOrderProcessCmdImpl:
   a. Abra el Centro de mandatos de DB2 (**Inicio > Programas >IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**).

b. En el menú **Herramientas**, seleccione **Valores de herramientas**.
c. Marque el recuadro de selección **Utilizar carácter terminador de sentencia** y asegúrese de que el carácter especificado sea un punto y coma (**;**)
d. Cierre los valores de herramientas.
e. Con la pestaña Script seleccionada, cree la entrada requerida en la tabla URLREG indicando la siguiente información en la ventana de script:

```
connect to BDdestino user usuariobd using contraseñabd;
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FashionFlow,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

donde

- *BDdestino* es el nombre de la base de datos de destino
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña del usuario de base de datos
- *ID_enttienda_FashionFlow* es el identificador de tienda para la tienda de ejemplo

Pulse el icono **Ejecutar**. Mantenga abierto el Centro de mandatos.

2. Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyOrderItemAddCmdImpl:
   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 la siguiente sentencia SQL:

```
insert into CMDREG (STOREENT_ID, INTERFACENAME, DESCRIPTION,
   CLASSNAME, TARGET)
values (ID_enttienda_FashionFlow,
   'com.ibm.commerce.order.commands.ExtOrderProcessCmd',
   'This is a new task command for tutorial two.',
   'com.ibm.commerce.sample.commands.MyExtOrderProcessCmdImpl',
   'local');
```

   Pulse Intro para ejecutar la sentencia SQL.
   f. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

Creación de la tabla XORDGIFT

En este paso, creará la tabla XORDGIFT.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para crear la tabla:

1. En la ventana Script, entre lo siguiente:

```
create table XORDGIFT (ORDERSID bigint not null, RECEIPTNAME varchar(50),
SENDERNAME varchar(50), MSGFIELD1 varchar(50), MSGFIELD2 varchar(50),
constraint p_xordgift primary key (ORDERSID),
```

```
constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
on delete cascade);
insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
```

donde

- *BDdestino* es el nombre de la base de datos de destino
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña del usuario de base de datos

Pulse el icono Ejecutar.
La tabla XORDGIFT ya se ha creado.

Oracle Si está utilizando una base de datos Oracle, realice 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 SQL Plus, entre las sentencias SQL siguientes:

   ```
   create table XORDGIFT (ORDERSID NUMBER NOT NULL, RECEIPTNAME VARCHAR(50),
   SENDERNAME VARCHAR(50), MSGFIELD1 VARCHAR(50), MSGFIELD2 VARCHAR(50),
   constraint p_xordgift primary key (ORDERSID),
   constraint f_xordgift foreign key (ORDERSID) references ORDERS(ORDERS_ID)
   on delete cascade);
   insert into XORDGIFT (ORDERSID) (select ORDERS_ID from ORDERS);
   ```

   y pulse Intro para ejecutar la sentencia SQL. La tabla XORDGIFT ya se ha creado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia SQL.

Actualización de elementos de tienda en el WebSphere Commerce Server de destino

En este paso, actualice la tienda con los elementos de tienda modificados, como se indica a continuación:

1. Haga una copia de seguridad del directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear\Stores.war.
2. Vaya al directorio \ImportTemp3\Stores\Web Content.
3. Copie la carpeta *nombre_FashionFlow* en el directorio siguiente: *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInst*.ear\Stores.war

donde *nombreInst* es el nombre de la instancia de WebSphere Commerce.

Actualización del archivo JAR de mandatos y beans de datos en el WebSphere Commerce Server de destino

En este paso actualizará el WebSphere Commerce Server de destino para utilizar el nuevo archivo JAR de mandatos y beans de datos, como se indica a continuación:

1. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.
   b. Haga una copia del archivo WebSphereCommerceServerExtensionsLogic.jar y guárdela en una ubicación de copia de seguridad.
2. Copie el nuevo archivo WebSphereCommerceServerExtensionsLogic.jar del directorio \ImportTemp3 en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear

donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.

Actualización del archivo JAR EJB en el WebSphere Commerce Server de destino

En este paso actualizará el WebSphere Commerce Server de destino para utilizar el nuevo archivo JAR EJB, como se indica a continuación:

1. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.
   b. Haga una copia del archivo WebSphereCommerceServerExtensionsData.jar y guárdela en una ubicación de copia de seguridad.

   donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.
2. Copie el nuevo archivo WebSphereCommerceServerExtensionsData.jar del directorio \ImportTemp3 en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear
3. A continuación, deberá modificar la información de descriptor de despliegue EJB, del modo siguiente:
   a. Localice el depósito de despliegue (directorio META-INF) para esta célula de WebSphere Application Server. Éste toma normalmente el formato siguiente:
   dir_instal_WAS\config\cells*nombreCélula*
   \applications\WC_*nombre_instancia*.ear\deployments\
   WC_*nombre_instancia**nombreMóduloEJB*.jar\META-INF
   A continuación se muestra un ejemplo específico de esto:
   D:\WebSphere\AppServer\config\cells\myCell\applications\
   WC_demo.ear\deployments\WC_demo\
   WebSphereCommerceServerExtensionsData.jar\META-INF
   donde
      - mycell es el nombre de la célula de WebSphere Application Server
      - demo es el nombre de la instancia de WebSphere Commerce
   b. El directorio contiene los archivos siguientes:
      - ejb-jar.xml
      - ibm-ejb-access-bean.xmi
      - ibm-ejb-jar-bnd.xmi
      - ibm-ejb-jar-ext.xmi

- MANIFEST.MF

Haga una copia de seguridad de todos estos archivos.

c. Utilice una herramienta para abrir el nuevo archivo WebSphereCommerceServerExtensionsData.jar y ver su contenido.

d. Extraiga el contenido del directorio meta-inf de este archivo WebSphereCommerceServerExtensionsData.jar en el directorio del paso 3a.

4. Utilizando el mandato startServer de WebSphere Application Server en la línea de mandatos, reinicie la instancia de WebSphere Commerce.

Verificación de la funcionalidad del mensaje de regalo en el WebSphere Commerce Server de destino

En esta sección, verificará que la lógica del mensaje de regalo esté funcionando correctamente en el WebSphere Commerce Server de destino, realizando lo siguiente:

1. Abra un navegador y entre el URL para la tienda que está basada en la tienda de ejemplo FashionFlow.
2. Inicie la sesión como usuario nuevo. Por ejemplo, pulse ″Registrar″ y, a continuación, cree el usuario ″shopper″.
3. Como usuario nuevo registrado, examine la tienda, añada un artículo al carro de la compra y, a continuación, realice la compra. Podrá añadir un mensaje de regalo al pedido, como se muestra en la siguiente instantánea de pantalla (versión en inglés):

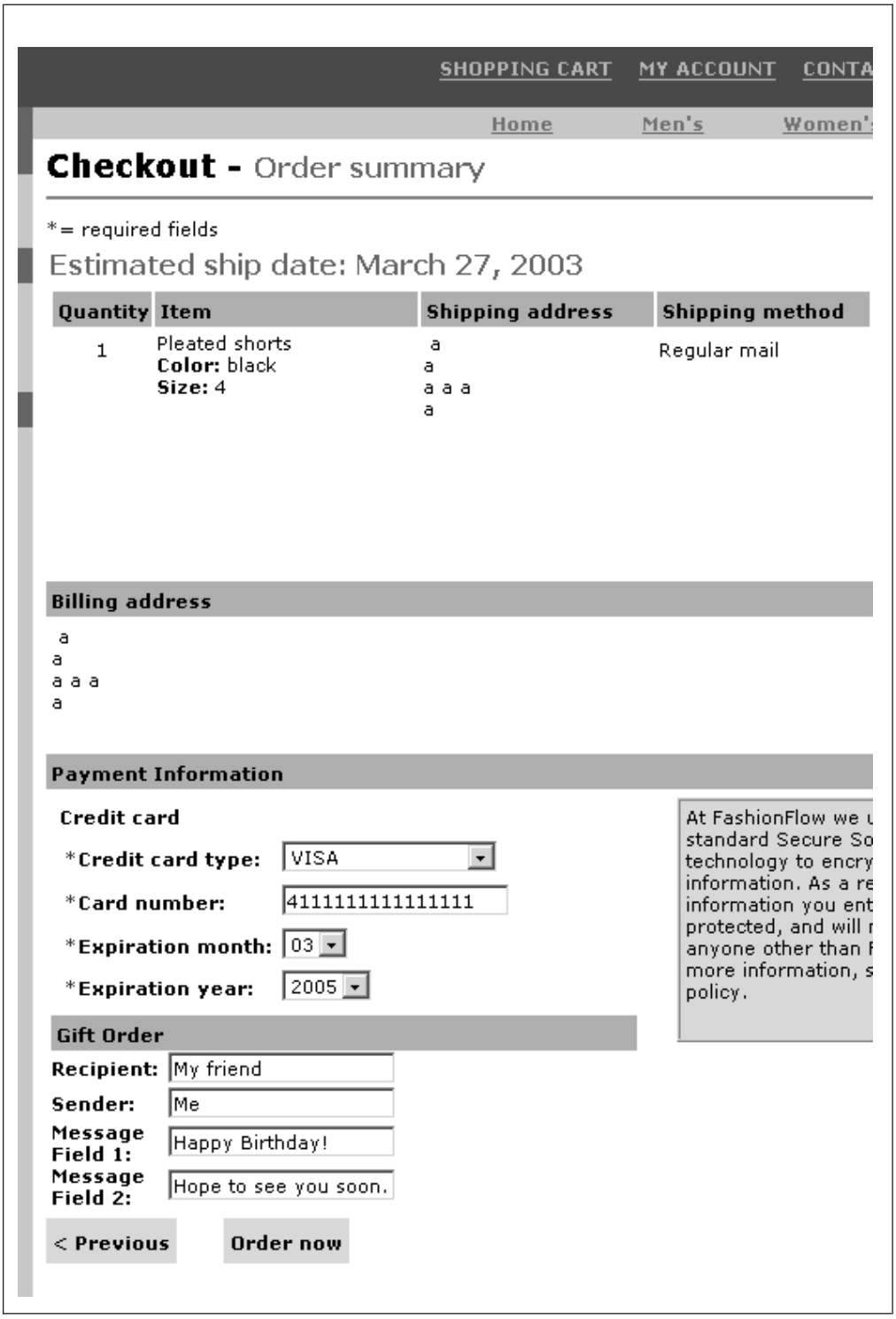


Figura 51.

Anote el número de pedido asociado con este pedido.

4. Pulse **Mi cuenta**.
5. Pulse **Ver pedidos**.
6. Seleccione el pedido que ha creado en el paso 3. Verá una pantalla similar a la siguiente:

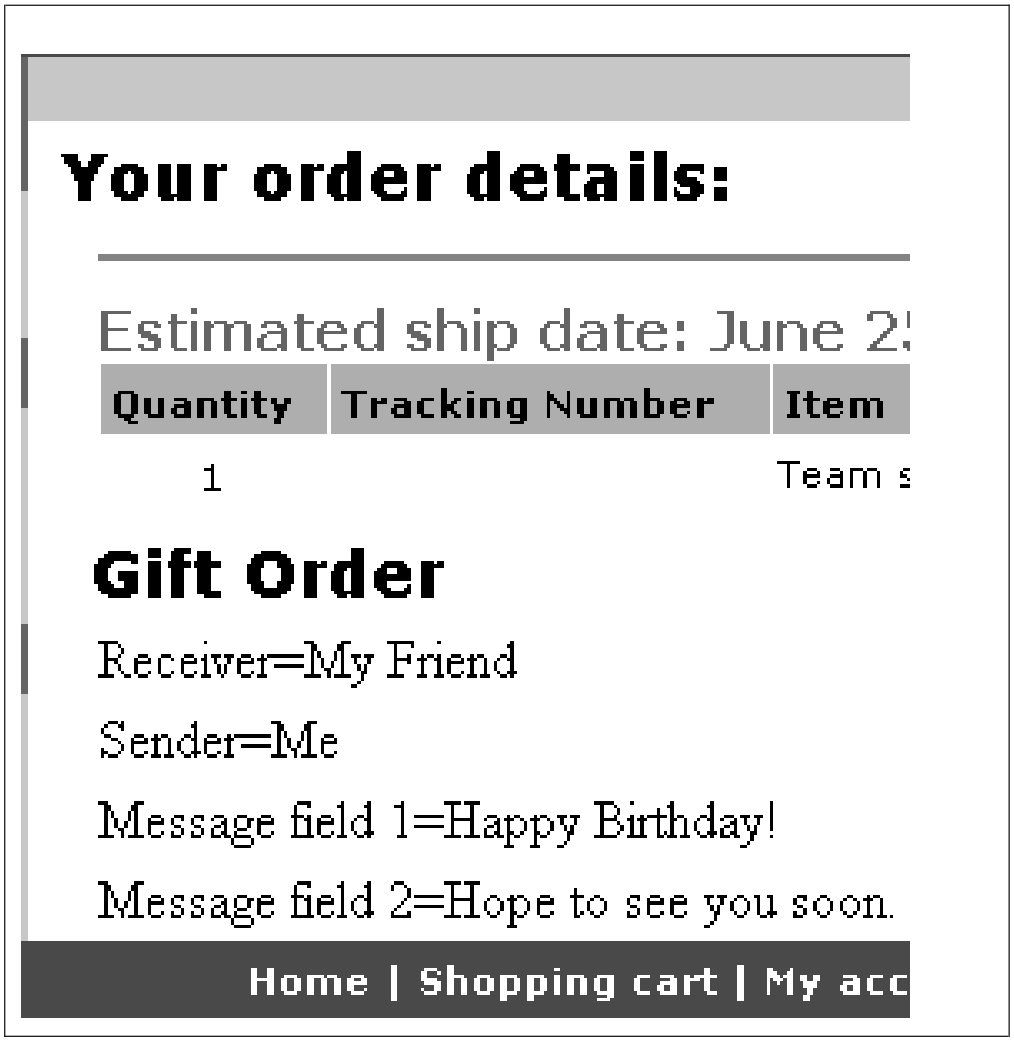

Your order details:
Estimated ship date: June 2
Quantity
Tracking Number
Item
1
Team s
Gift Order
Receiver=My Friend
Sender=Me
Message field 1=Happy Birthday!
Message field 2=Hope to see you soon.
Home | Shopping cart | My acc

Figura 52.

Capítulo 13. Guía de aprendizaje: Ampliación de un bean de entidad de WebSphere Commerce existente

En esta guía de aprendizaje se describe el proceso utilizado para modificar un bean de entidad de WebSphere Commerce existente. Se utiliza un escenario en el que se añade una encuesta de información sobre la vivienda adicional al proceso de registro de usuario. En este caso, se modifica el bean de entidad User para incluir campos adicionales que almacenan el valor de tipo de vivienda de un usuario y un valor de ubicación. Se crea una nueva tabla de base de datos (denominada XHOUSING) y se modifica la información de correlación existente para el bean User a fin de incluir esta nueva tabla.

Además de los cambios de tabla nueva y de bean de entidad, se crea una nueva clase de implementación MyPostUserRegistrationAddCmdImpl. Ésta contiene la lógica que procesa la información de encuesta sobre la vivienda y actualiza la base de datos con la nueva información.

Para poder reunir la información sobre la vivienda y, posteriormente, visualizar los resultados, se modifican los archivos UserRegistrationAddForm.jsp y UserRegistrationUpdateForm.jsp.

Requisitos previos

Para esta guía de aprendizaje no se requiere haber completado otras guías de aprendizaje. Si ha completado otras guías de aprendizaje, no hay ningún peligro en dejar el código en el espacio de trabajo, puesto que dicho código no está en conflicto con esta guía de aprendizaje.

Antes de empezar esta guía de aprendizaje, tendrá que haber publicado una tienda basada en la tienda de ejemplo FashionFlow. En esta tienda, debe poder realizar una compra (por ejemplo, examinar el catálogo, añadir artículos al carro de la compra, pasar por caja y ver la confirmación del pedido).

Creación de la tabla XHOUSING e inserción de datos en ella

Como preparación para la creación del bean de entidad, primero deberá crear una nueva tabla de base de datos y llenarla de datos. La tabla que se debe crear se denomina XHOUSING.

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para crear la tabla:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**) y pulse la pestaña **Scripts**.
2. En la ventana Script, entre lo siguiente:

```
connect to BDdesarrollo user usuariobd using contraseñabd;
create table XHOUSING (MEMBERID bigint not null,
   HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
   references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

donde

- *BDdesarrollo* es el nombre de la base de datos de desarrollo
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña para el usuario de base de datos

Pulse el icono Ejecutar.
Verá un mensaje que indica que la sentencia SQL se ha completado satisfactoriamente.

Oracle Si está utilizando una base de datos Oracle, realice 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 SQL:

```
create table XHOUSING (MEMBERID number not null,
   HOUSINGTYPE integer, LOCATION integer,
   constraint p_xhousing primary key (MEMBERID),
   constraint f_xhousing foreign key (MEMBERID)
   references USERS (USERS_ID) on delete cascade);
insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
```

   y pulse Intro para ejecutar la sentencia SQL. La tabla XHOUSING ya se ha creado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

Adición de campos nuevos en el bean de entidad User

En esta sección, deberá añadir en el bean de entidad User dos campos nuevos que se utilizan para capturar información sobre la vivienda. Los nuevos campos se denominan housingType y location. Estos campos se correlacionan finalmente con las columnas HOUSINGTYPE y LOCATION de la tabla XHOUSING.

Para añadir estos campos nuevos, realice lo siguiente:

1. Inicie entorno de desarrollo de WebSphere Commerce de este modo:
   - Business Professional **Inicio > Programas > IBM WebSphere Commerce Studio > entorno de desarrollo de WebSphere Commerce**
   - Express Express **Inicio > Programas > IBM WebSphere - Express Developer Edition > entorno de desarrollo de WebSphere Commerce**
2. Conmute a la perspectiva de Servidor y asegúrese de que el servidor de prueba WebSphereCommerceServer esté detenido.
3. Conmute a la perspectiva de J2EE y seleccione la vista de Jerarquía J2EE.
4. Expanda **Módulos EJB** y, a continuación, efectúe una doble pulsación en el proyecto EJB **Member-MemberManagementData**. Esto hará que se abra el editor del Descriptor de despliegue de EJB.
5. Pulse la pestaña **Beans** y seleccione el bean **User** en la lista de beans visualizados.

6. Pulse **Añadir** junto al recuadro de texto de campos de CMP.
   Se abrirá la ventana Crear atributo CMP.
7. Cree un campo CMP con las propiedades siguientes:
   a. En el campo **Nombre**, entre `housingType`.
   b. En el campo Tipo, entre `java.lang.Integer`.
   c. Habilite **Acceso con métodos get y set**.
   d. Elimine la marca del recuadro de selección **Promocionar método get y set a interfaz remota**. (Esto elimina automáticamente la opción para hacer que get sea una opción de sólo lectura).
   e. Pulse **Aceptar**.
8. Pulse otra vez **Añadir** para crear otro campo CMP con las propiedades siguientes:
   a. En el campo **Nombre**, entre `location`.
   b. En el campo Tipo, entre `java.lang.Integer`.
   c. Habilite **Acceso con métodos get y set**.
   d. Elimine la marca del recuadro de selección **Promocionar método get y set a interfaz remota**. (Esto elimina automáticamente la opción para hacer que get sea una opción de sólo lectura).
   e. Pulse **Aceptar**.

   Los nuevos campos se visualizan en la lista de campos CMP.
9. Guarde los cambios y, a continuación, cierre el editor del Descriptor de despliegue de EJB.

Actualización de la información de esquema y de correlación de tablas

En las secciones siguientes, actualizará el esquema User con la nueva tabla XHOUSING, 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 XHOUSING. Al seguir esta propuesta para ampliar el modelo de objeto, en el código parece como si se hubieran añadido columnas nuevas directamente en la tabla USERS.

Creación de la definición de tabla para la tabla XHOUSING

Para crear la definición de tabla XHOUSING y crear la relación de clave externa entre las tablas USERS y XHOUSING, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Bases de datos**.
2. Expanda la base de datos **Member-MemberManagementData** y, a continuación, expanda el esquema **NULLID**.
3. Pulse con el botón derecho del ratón en **Tablas** y seleccione **Nueva > Definición de tabla nueva**.
   Se abrirá la ventana Definición de tabla.
4. En el campo **Nombre de tabla**, entre `XHOUSING` y pulse **Siguiente**.
5. Añada la columna de clave en la definición de tabla, del modo siguiente:

   Business Professional

   a. Pulse **Añadir otra**.
   b. En el campo **Nombre de columna**, entre `ORDERSID`.
   c. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:

      DB2 `BIGINT`

      Oracle `NUMBER`

d. Seleccione **Columna de clave**.

e. Oracle En el campo **Precisión numérica**, entre 38.

f. Oracle Deje el valor para **Escala numérica** en 0.

Express

a. Pulse **Añadir otra**.

b. En el campo **Nombre de columna**, entre ORDERSID.

c. Seleccione **Columna de clave**.

d. En la lista desplegable **Tipo de columna**, seleccione lo siguiente:

DB2 BIGINT

Oracle NUMBER

e. Oracle En el campo **Precisión numérica**, entre 38.

f. Oracle Deje el valor para **Escala numérica** en 0.

6. Añada columnas adicionales en la definición de tabla, del modo siguiente:

a. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 26.

Propiedad	Valor
Nombre de columna	HOUSINGTYPE
Tipo de columna	INTEGER
Nullable	Select

b. Pulse **Añadir otra** y cree una columna con las propiedades siguientes:

Tabla 27.

Propiedad	Valor
Nombre de columna	LOCATION
Tipo de columna	INTEGER
Nullable	Select

c. Pulse **Siguiente**.

7. En el campo **Nombre de clave primaria**, entre p_xhousing y pulse **Siguiente**.

8. Pulse **Añadir otra** para añadir la clave externa. Especifique los valores siguientes:

a. En el campo **Nombre de clave externa**, entre f_xhousing.

b. En la lista desplegable **Al suprimir**, seleccione **CASCADE**

c. En la lista desplegable **Tabla de destino**, seleccione **NULLID.USERS**

d. En el panel Columnas origen, pulse **MEMBERID** y, a continuación, pulse > para añadir la clave externa.

9. Pulse **Finalizar**.

10. Oracle Deberá editar la definición de tabla utilizando un editor de texto, como se indica a continuación:

a. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.

b. Expanda el proyecto **Member-MemberManagementData**.

c. Expanda lo siguiente: **ejbModule > META-INF >Schema**.

d. Pulse con el botón derecho del ratón en el archivo **Member-MemberManagementData_NULL_XHOUSING.xmi** y seleccione **Abrir con > Editor de texto**.

e. Sustituya todas las apariciones de `SQLNumeric_6` por `SQLNumeric_3`.

f. Guarde los cambios y cierre el editor de texto.

La nueva definición de tabla Member-MemberManagementData_NULL_XHOUSING se puede ver expandiendo la carpeta Tablas bajo Member-MemberManagementData/NULLID.

Creación de la correlación de la tabla XHOUSING

En esta sección, se crea la correlación entre las dos columnas (HOUSINGTYPE y LOCATION) de la tabla XHOUSING y los dos campos (housingType y location) del bean de entidad User.

Para crear esta correlación, realice lo siguiente:

1. Cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda las carpetas siguientes: **Member-MemberManagementData > ejbModule >META-INF**.
3. Efectúe una doble pulsación en el archivo **Map.mapxmi**.
4. Express En el panel Beans enterprise, expanda el grupo **Miembro** y, a continuación, pulse con el botón derecho del ratón en el bean de entidad **User**.
5. En el panel Tablas, resalte las tablas siguientes:
   - **MEMBER**
   - **USERS**
   - **XHOUSING**

   (Mantenga pulsada la tecla Control para seleccionar varias tablas a la vez).
6. Business Professional En el panel Beans enterprise (en la parte superior del editor), expanda el grupo **Miembro** y, a continuación, pulse con el botón derecho del ratón en el bean de entidad **User** y seleccione **Crear correlación**.

   Express En el panel Beans enterprise (en la parte superior del editor), pulse con el botón derecho del ratón el bean de entidad **User** y seleccione **Crear correlación**.
7. Expanda el bean de entidad **User**.
8. En el panel Tablas, expanda la tabla **XHOUSING**, para que se puedan ver las columnas.
9. Resalte el atributo de bean **housingType** y arrástrelo hasta la columna **HOUSINGTYPE** para crear la correlación.
10. Resalte el atributo de bean **location** y arrástrelo hasta la columna **LOCATION** para crear la correlación.
11. Guarde los cambios y, a continuación, cierre el archivo Map.mapxmi.

Actualización del archivo de correlación

1. Expanda las carpetas siguientes: **Member-MemberManagementData > ejbModule >META-INF**.
2. Pulse con el botón derecho del ratón en el archivo **Map.mapxmi** y seleccione **Abrir con > Editor de texto**.
3. Localice la serie siguiente:

```
User_EJB
```

Dos líneas más abajo de esto, es posible que encuentre:

```
<discriminatorValues>User</discriminatorValues>
```

Si encuentra la línea anterior, deberá cambiarla por

```
<discriminatorValues>'U'</discriminatorValues>
```

4. Guarde el cambio.

Generación de los beans de acceso y del código desplegado

Puesto que ha modificado el bean de entidad User, deberá volver a generar el bean de acceso y el código desplegado.

Para volver a generar el bean de acceso, realice lo siguiente:

1. En la vista de Jerarquía J2EE, expanda **Módulos EJB**.
2. Pulse con el botón derecho del ratón en el módulo EJB **Member-MemberManagementData** y seleccione **Beans de acceso> Editar beans de acceso**.
3. Seleccione **CopyHelper** y pulse **Siguiente**.
4. En la ventana Seleccionar proyecto EJB, pulse **Siguiente**.
5. En la ventana Bean de acceso con ayudante de copia, realice lo siguiente:
   a. En la lista desplegable Beans enterprise, seleccione **User**.
   b. En la lista desplegable **Método constructor**, seleccione **findByPrimaryKey(com.ibm.commerce.user.objects.MemberKey)**.
   c. En la lista Ayudantes de atributo, asegúrese de que los atributos **housingType** y **location** estén seleccionados junto con todos los demás atributos.
6. Pulse **Finalizar**.
7. Pulse con el botón derecho del ratón en el módulo EJB **Member-MemberManagementData** y seleccione **Beans de acceso> Regenerar beans de acceso**.
8. Pulse **Seleccionar todo** y, a continuación, pulse **Finalizar**.

Para volver a generar el código desplegado, realice lo siguiente:

1. Pulse con el botón derecho del ratón en el módulo EJB **Member-MemberManagementData** y seleccione **Generar > Código de despliegue y RMIC**.
2. Pulse **Seleccionar todo** y, a continuación, pulse **Finalizar**.

Creación de la implementación MyPostUserRegistrationAddCmdImpl

En este paso, se amplía el mandato de controlador UserRegistrationAddCmd para incluir lógica nueva que se utiliza para analizar la nueva información sobre la vivienda que se reúne en el formulario de registro. La ampliación se realiza creando una nueva clase de implementación MyPostUserRegistrationAddCmdImpl que implementa las interfaces PostUserRegistrationCmd. Después de crear esta nueva clase de implementación, deberá actualizar el registro de mandatos para reflejar este cambio. Igual que con otras guías de aprendizaje, se proporciona el código para este nuevo mandato.

Para crear la nueva clase de implementación MyPostUserRegistrationAddCmdImpl, realice lo siguiente:

1. Asegúrese de que ha completado los pasos del apartado "Localización del código de ejemplo" en la página 192.
2. En WebSphere Studio Application Developer, abra la perspectiva Java (**Ventana > Abrir perspectiva > Java**).
3. Expanda el proyecto **WebSphereCommerceServerExtensionsLogic**.
4. Pulse con el botón derecho del ratón en la carpeta **src** y seleccione **Importar**. Se abrirá el asistente para Importar.
5. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
6. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
   suDirectorio\WC_SAMPLE_55.zip
   donde *suDirectorio* es el directorio en el que ha bajado el paquete.
7. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione importar el archivo siguiente:
   - com\ibm\commerce\sample\commands\ MyPostUserRegistrationAddCmdImpl.java
8. En el campo **Carpeta**, la carpeta `WebSphereCommerceServerExtensionsLogic/src` ya está especificada. Conserve este valor.
9. Pulse **Finalizar**.
10. Expanda el paquete **com.ibm.commerce.sample.commands**.
11. Efectúe una doble pulsación en la nueva clase **MyPostUserRegistrationAddCmdImpl**.
12. Elimine la marca de comentario de la Sección 1. Este código configura variables y crea métodos get y set para estas variables:

```
/// Section 1 ////////////////
 Integer housingType = null;
 Integer location = null;

 public Integer getHousingType() {
  return housingType;
 }

 public Integer getLocation() {
  return location;
 }

 public void setHousingType(Integer newHousingType) {
  housingType = newHousingType;
 }

 public void setLocation(Integer newLocation) {
  location = newLocation;
 }
/// End of Section 1 ////////////////
```

13. Elimine la marca de comentario de la Sección 2. Esto inserta el código siguiente en la clase:

```
/// Section 2 ////////////////
public void performExecute() throws ECException {
  super.performExecute();

  // Set the needed fields before processing the survey
  setHousingType(requestProperties.getInteger("housingType", 0));
  setLocation(requestProperties.getInteger("location", 0));
```

```
 processSurvey();
}

/// End of Section 2 ////////////////
```

El código anterior llama al método performExecute de la superclase para que se ejecute la lógica normal de PostUserRegistrationAddCmdImpl. Una vez que esto se ha completado, se llama al nuevo método processSurvey.

14. Elimine la marca de comentario de la Sección 3 para insertar el código siguiente en la clase:

```
/// Section 3 ////////////////

 private void processSurvey() throws ECException {

  try {
   // load up the user data
   UserAccessBean abUser = new UserAccessBean();
   abUser.setInitKey_MemberId(commandContext.getUserId().toString());
   abUser.refreshCopyHelper();

   // store the new attributes
   abUser.setHousingType(getHousingType());
   abUser.setLocation(getLocation());

   abUser.commitCopyHelper();
  } catch (javax.ejb.FinderException e) {
   throw new ECSystemException(
    ECMessage._ERR_FINDER_EXCEPTION,
    this.getClass().getName(),
    "processSurvey");
  } catch (javax.naming.NamingException e) {
   throw new ECSystemException(
    ECMessage._ERR_NAMING_EXCEPTION,
    this.getClass().getName(),
    "processSurvey");
  } catch (java.rmi.RemoteException e) {
   throw new ECSystemException(
    ECMessage._ERR_REMOTE_EXCEPTION,
    this.getClass().getName(),
    "processSurvey");
  } catch (javax.ejb.CreateException e) {
   throw new ECSystemException(
    ECMessage._ERR_CREATE_EXCEPTION,
    this.getClass().getName(),
    "processSurvey");
  }

 }
/// End of Section 3 ////////////////
```

15. Guarde los cambios.
16. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Crear proyecto**.

Modificación del registro de mandatos

Debe modificar el registro de mandatos para que se utilice la nueva clase de implementación en el flujo de compra.

Para modificar el registro de mandatos, realice lo siguiente:

1. DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyPostUserRegistrationAddCmdImpl:

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. Marque el recuadro de selección **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 BDdesarrollo user usuariobd using contraseñabd;
insert into CMDREG values (id_enttienda_FashionFlow,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Command for modified user bean tutorial',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

donde

- *BDdesarrollo* es el nombre de la base de datos de desarrollo
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña para el usuario de base de datos
- *Id_enttienda_FashionFlow* es el identificador exclusivo de entidad de tienda para la tienda.

Pulse el icono **Ejecutar**.

2. Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyPostUserRegistrationAddCmdImpl:
   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 la siguiente sentencia SQL:

```
insert into CMDREG values (id_enttienda_FashionFlow,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Command for modified user bean tutorial',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

   Pulse Intro para ejecutar la sentencia SQL.
   f. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

   y pulse Intro para ejecutar la sentencia SQL.

Modificación de las plantillas JSP para reunir y visualizar información sobre la vivienda

En este paso, se modifican las plantillas UserRegistrationAddForm y UserRegistrationUpdateForm para que el cliente puede entrar información sobre la vivienda al iniciar la sesión, así como ver la información sobre la vivienda en una página de resumen. La estrategia para modificar estas páginas es incluir plantillas JSP adicionales que especifiquen la nueva información para la página. Estas páginas nuevas (UserRegistrationAddFormInclude.jsp y UserRegistrationUpdateFormInclude.jsp) utilizan JSTL para visualizar la nueva información.

Para modificar estas páginas, realice lo siguiente:

1. Conmute a la perspectiva de Web.
2. Si no ha completado el Capítulo 10, "Guía de aprendizaje: Creación de lógica de negocio nueva", en la página 191, deberá modificar las propiedades del proyecto web Stores, del modo siguiente:
   a. Pulse con el botón derecho del ratón en el proyecto Web **Stores** y seleccione **Propiedades**.
   b. Business Professional Seleccione **Web** en el panel izquierdo y, a continuación, en la lista Características de proyecto Web disponibles, seleccione **Incluir la biblioteca de códigos estándar JSP**.

      Express Seleccione **Características de proyecto Web** en el panel izquierdo y, a continuación, en la lista Características de proyecto Web disponibles, seleccione **Biblioteca de códigos estándar JSP**.

      Pulse **Aplicar**. Cuando la actualización se haya completado, pulse **Aceptar** para cerrar el editor de propiedades.
3. Expanda el directorio siguiente:
   Stores\Web Content*nombre_FashionFlow*.
4. Cree una copia de seguridad del archivo UserRegistrationAddForm.jsp, realizando lo siguiente:
   a. Expanda los directorios **UserArea>AccountSection>RegistrationSubsection**.
   b. Pulse con el botón derecho del ratón en el archivo **UserRegistrationAddForm.jsp** y seleccione **Redenominar**.
   c. En la ventana Redenominar, entre `UserRegistrationAddForm_bak.jsp` y pulse **Aceptar**.
   d. Cuando se le solicite si desea actualizar los enlaces a este archivo, pulse **No**.
5. Cree una copia de seguridad del archivo UserRegistrationUpdateForm.jsp, realizando lo siguiente:
   a. Expanda los directorios **UserArea>AccountSection>RegistrationSubsection**.
   b. Pulse con el botón derecho del ratón en el archivo **UserRegistrationUpdateForm.jsp** y seleccione **Redenominar**.
   c. En la ventana Redenominar, entre `UserRegistrationUpdateForm_bak.jsp` y pulse **Aceptar**.
   d. Cuando se le solicite si desea actualizar los enlaces a este archivo, pulse **No**.
6. Pulse con el botón derecho del ratón en el directorio *nombre_FashionFlow* y seleccione **Importar**.
   Se abrirá el asistente para Importar.
7. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
8. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
   suDirectorio\WC_SAMPLE_55.zip
   donde *suDirectorio* es el directorio en el que ha bajado el paquete.
9. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione los archivos siguientes para importarlos.
   - UserArea\AccountSection\RegistrationSubsection\ UserRegistrationAddForm.jsp

- UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddFormInclude.jsp
- UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateForm.jsp
- UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateFormInclude.jsp

10. En el campo **Carpeta**, la carpeta `Stores/Web Content/`*`nombre_FashionFlow`* ya está especificada. Conserve este valor.
11. Pulse **Finalizar**.

Si examina el archivo UserRegistrationAddForm.jsp, encontrará que se ha añadido la sección siguiente para esta guía de aprendizaje:

```
<%-- Add for tutorial --%>
   <tr>
      <td colspan="3">
         <jsp:include page="UserRegistrationAddFormInclude.jsp" flush="true" />
      </td>
   </tr>
<%-- End of tutorial --%>
```

También puede examinar el archivo UserRegistrationAddFormInclude.jsp para ver cómo se reúne la nueva información, utilizando JSTL. De forma similar, examine los archivos UserRegistrationAddForm.jsp y UserRegistrationAddFormInclude.jsp.

También deberá importar el archivo de propiedades que contiene los valores de serie utilizados en las plantillas JSP modificadas. Este archivo se denomina Housing.properties. Para importar este archivo, realice lo siguiente:

1. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, expanda los siguientes directorios:
   Stores > Web Content> WEB-INF >classes > *nombre_FashionFlow*.
2. Pulse con el botón derecho del ratón en el directorio *nombre_FashionFlow* y seleccione **Importar**.
   Se abrirá el asistente para Importar.
3. En la lista **Seleccionar un origen de importación**, seleccione **Archivo zip** y pulse **Siguiente**.
4. Pulse **Examinar** (junto al campo **Archivo zip**) y navegue hasta el código de ejemplo. Este archivo está en la ubicación siguiente:
   suDirectorio\WC_SAMPLE_55.zip
   donde *suDirectorio* es el directorio en el que ha bajado el paquete.
5. Pulse **Deseleccionar todo** y, a continuación, expanda los directorios y seleccione el archivo siguiente para importarlo.
   - Housing.properties
6. En el campo **Carpeta**, la carpeta `Stores/Web Content/WEB-INF/classes/`*`nombre_FashionFlow`* ya está especificada. Conserve este valor.
7. Pulse **Finalizar**.

Comprobación del código modificado

A continuación, deberá comprobar la información de registro modificada, realizando lo siguiente:

1. Conmute a la perspectiva de Servidores.
2. Pulse con el botón derecho del ratón en el servidor de prueba **WebSphereCommerceServer** y seleccione **Iniciar**.

3. Pulse con el botón derecho del ratón en **index.jsp** bajo el directorio Stores\Web Content*nombre_FashionFlow* y seleccione **Ejecutar en servidor**.
4. Cuando se abra la página de presentación de la tienda, pulse **Registrar**.
5. Pulse **Registrar** otra vez para crear un usuario nuevo.
6. Aparecerá una página de registro modificada, que ahora contendrá la encuesta sobre la vivienda, tal como se muestra en la siguiente captura de pantalla:

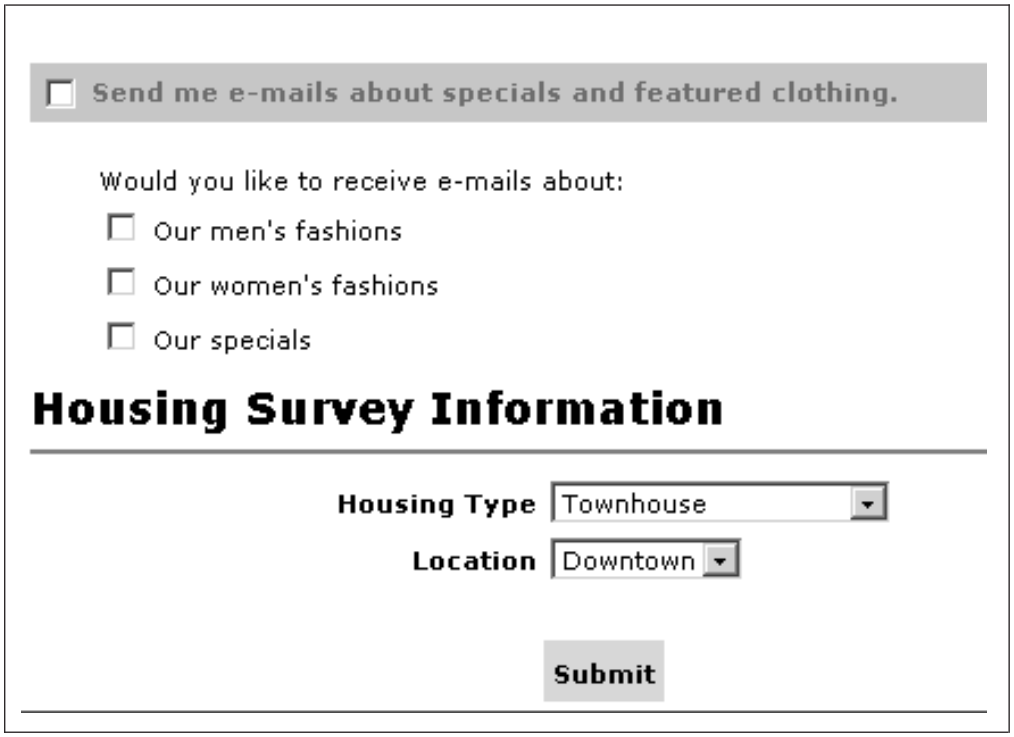


Figura 53.

7. Entre la información que sea apropiada para el nuevo usuario y pulse **Someter**.
8. Cuando la información se haya sometido, pulse **Modificar información personal** para verificar que la información sobre la vivienda se ha capturado. Aparecerá una pantalla que muestra un resumen de las respuestas a la encuesta, como se muestra a continuación:

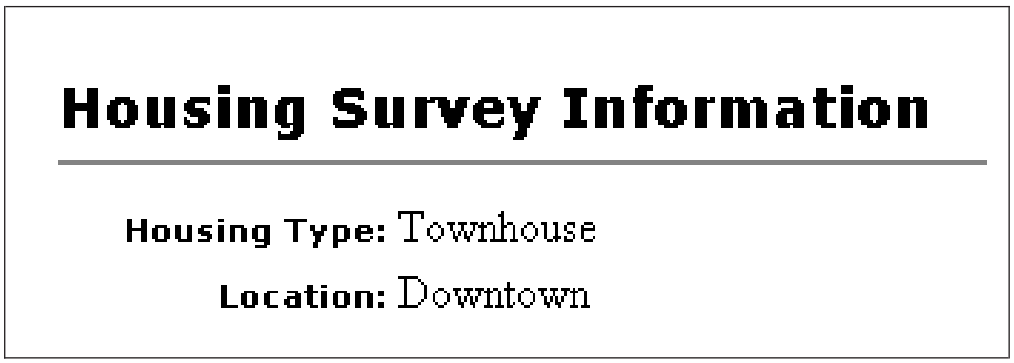


Figura 54.

Despliegue de la lógica de encuesta sobre la vivienda

Esta sección describe cómo desplegar la nueva lógica de negocio en una tienda que se ejecuta en un WebSphere Commerce Server remoto. Antes de empezar estos pasos de despliegue, tiene que haber creado una tienda (basada en la tienda de ejemplo FashionFlow) 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.

Existen varios tipos diferentes de elementos que debe desplegar en el WebSphere Commerce Server de destino. Éstos incluyen:

- Lógica de mandatos
- Lógica de beans enterprise modificada
- Plantillas JSP
- Un archivo de propiedades
- Actualizaciones de base de datos que incluyen actualizaciones de esquema (nueva tabla) así como actualizaciones de registro de mandatos

Esta sección describe cómo desplegar todos estos elementos, *de forma incremental* en el WebSphere Commerce Server de destino.

Creación del archivo JAR de mandatos

Esta sección describe cómo crear el archivo JAR que contiene la nueva lógica de MyPostUserRegistrationAddCmdImpl.

Para crear este archivo JAR, realice los pasos siguientes en la máquina de desarrollo:

1. En el sistema de archivos local, cree un directorio denominado \ExportTemp4.
2. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
3. Pulse con el botón derecho del ratón en el proyecto **WebSphereCommerceServerExtensionsLogic** y seleccione **Exportar**. Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR** y pulse **Siguiente**.
   b. El panel izquierdo bajo **Seleccionar los recursos para exportar** está previamente lleno con el nombre del proyecto. Deje este valor como está.
   c. En el panel derecho, asegúrese de que sólo estén seleccionados los recursos siguientes:
      - .classpath
      - .project
      - .serverPreference
   d. Asegúrese de que **Exportar archivos de clases y recursos generados** esté seleccionado.
   e. *No* seleccione **Exportar archivos fuente y recursos Java**.
   f. En el campo **Seleccionar el destino de exportación**, entre el nombre de archivo JAR totalmente calificado que se debe utilizar. En este caso, entre *unidad*:\ExportTemp4\WebSphereCommerceServerExtensionsLogic.jar. Tenga

en cuenta que el nombre de archivo JAR debe ser WebSphereCommerceServerExtensionsLogic.jar.

g. Pulse **Finalizar**.

Creación del archivo JAR EJB

Para crear el archivo JAR EJB, realice lo siguiente:

1. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda el proyecto **Member-MemberManagementData**.
3. Efectúe una doble pulsación en **Descriptor de despliegue EJB**.
4. Con la pestaña Visión general seleccionada, desplácese a la parte inferior del panel para localizar la sección **Enlaces de WebSphere**.
5. En el campo **Nombre JNDI de origen de datos**, entre el nombre JNDI de origen de datos del WebSphere Commerce Server de destino. A continuación se muestra un valor de ejemplo:

   DB2 `jdbc/WebSphere Commerce DB2 DataSource demo`
   donde el WebSphere Commerce Server de destino está utilizando una base de datos DB2 y el nombre de instancia de WebSphere Commerce es “demo”

   Oracle `jdbc/WebSphere Commerce Oracle DataSource demo`
   donde el WebSphere Commerce Server de destino está utilizando una base de datos Oracle y el nombre de instancia de WebSphere Commerce es “demo”.
6. Guarde los cambios del descriptor de despliegue (Control+S).
7. En la vista de Business Professional Navegador J2EE Express Navegador de proyectos, pulse con el botón derecho del ratón en el proyecto **Member-MemberManagementData** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
8. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Archivo JAR EJB** y pulse **Siguiente**.
   b. Business Professional El valor para **¿Qué recursos desea exportar?** está previamente lleno con el nombre del proyecto EJB.
      Express El nombre del proyecto EJB ya está rellenado.
      Deje el valor que aparece.
   c. Business Professional En el campo **¿Dónde desea exportar los recursos?** entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      Express Para el destino, entre el nombre de archivo JAR totalmente calificado que se debe utilizar.
      En este caso, entre *unidad*`:\ExportTemp4\Member-MemberManagementData.jar`.
   d. Pulse **Finalizar**.
9. Después de que se haya creado el archivo JAR, deshaga los cambios efectuados en el descriptor de despliegue local en el paso 5, para restaurar el valor que es necesario para el servidor de prueba local.

Exportación de elementos de tienda

Para exportar las plantillas JSP modificadas y el nuevo archivo de propiedades, realice lo siguiente:

1. Abra WebSphere Studio Application Developer y cambie a la vista de Business Professional Navegador J2EE Express Navegador de proyectos.
2. Expanda la carpeta **Stores**.

3. Pulse con el botón derecho del ratón en la carpeta **Web Content** y seleccione **Exportar**.
   Se abrirá el asistente para Exportar.
4. En el asistente para Exportar, realice lo siguiente:
   a. Seleccione **Sistema de archivos** y pulse **Siguiente**.
   b. Seleccione los recursos siguientes para desplegarlos:
      - Web Content*nombre_FashionFlow*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddForm.jsp
      - Web Content*nombre_FashionFlow*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationAddFormInclude.jsp
      - Web Content*nombre_FashionFlow*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateForm.jsp
      - Web Content*nombre_FashionFlow*\UserArea\AccountSection\RegistrationSubsection\UserRegistrationUpdateFormInclude.jsp
      - Web Content\WEB-INF\lib\jstl.jar
      - Web Content\WEB-INF\lib\standard.jar
      - Web Content\WEB-INF\classes*nombre_FashionFlow*\Housing.properties
   c. Seleccione **Crear estructura de directorios para archivos seleccionados**.
   d. En el campo Directorio, entre un directorio temporal en el que se colocarán estos recursos. Por ejemplo, entre `C:\ExportTemp4`
   e. Pulse **Finalizar**.

Transferencia de elementos al WebSphere Commerce Server de destino

En este paso, creará un directorio temporal en el WebSphere Commerce Server de destino y, a continuación, copiará los elementos de encuesta sobre la vivienda en este directorio. En los pasos subsiguientes, colocará los diferentes tipos de código en el lugar apropiado de la aplicación WebSphere Commerce.

Para copiar los archivos de la máquina de desarrollo en el WebSphere Commerce Server de destino, realice lo siguiente:

1. En el WebSphere Commerce Server de destino, cree un directorio temporal denominado \ImportTemp4.
2. Determine cómo copiará los archivos de un sistema a otro. Puede realizar esta tarea correlacionando una unidad del WebSphere Commerce Server de destino con la máquina de desarrollo o utilizando una aplicación FTP, si la tiene configurada.
3. En la máquina de desarrollo, copie el contenido de \ExportTemp4 en \ImportTemp4 del WebSphere Commerce Server de destino.

Detención del WebSphere Commerce Server de destino

Antes de empezar los pasos de despliegue, deberá detener el WebSphere Commerce Server de destino emitiendo el mandato stopServer en la línea de mandatos. Para obtener detalles sobre este mandato, consulte la publicación Business Professional *WebSphere Commerce Studio, Guía de instalación* o Express *WebSphere Commerce - Express Developer Edition, Guía de instalación*.

Actualización de la base de datos en el WebSphere Commerce Server de destino

Creación de la tabla XHOUSING

DB2 Si está utilizando una base de datos DB2, realice lo siguiente para crear la tabla:

1. Abra el Centro de mandatos de DB2 (**Inicio > Programas > IBM DB2 > Herramientas de la línea de mandatos > Centro de mandatos**) y pulse la pestaña **Scripts**.
2. En la ventana Script, entre lo siguiente:

   ```
   connect to BDdesarrollo user usuariobd using contraseñabd;
   create table XHOUSING (MEMBERID bigint not null,
      HOUSINGTYPE integer, LOCATION integer,
      constraint p_xhousing primary key (MEMBERID),
      constraint f_xhousing foreign key (MEMBERID)
      references USERS (USERS_ID) on delete cascade);
   insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
   ```

   donde

   - *BDdesarrollo* es el nombre de la base de datos de desarrollo
   - *usuariobd* es el usuario de base de datos
   - *contraseñabd* es la contraseña para el usuario de base de datos

   Pulse el icono Ejecutar.
   Verá un mensaje que indica que la sentencia SQL se ha completado satisfactoriamente.

Oracle Si está utilizando una base de datos Oracle, realice 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 SQL:

   ```
   create table XHOUSING (MEMBERID number not null,
      HOUSINGTYPE integer, LOCATION integer,
      constraint p_xhousing primary key (MEMBERID),
      constraint f_xhousing foreign key (MEMBERID)
      references USERS (USERS_ID) on delete cascade);
   insert into XHOUSING (MEMBERID) (select USERS_ID from USERS);
   ```

   y pulse Intro para ejecutar la sentencia SQL. La tabla XHOUSING ya se ha creado.
6. Entre lo siguiente para comprometer los cambios en la base de datos:

   ```
   commit;
   ```

   y pulse Intro para ejecutar la sentencia SQL.

Registro del mandato de tarea

Para modificar el registro de mandatos, realice lo siguiente:

1. DB2 Si está utilizando una base de datos DB2, realice lo siguiente para registrar MyPostUserRegistrationAddCmdImpl:

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. Marque el recuadro de selección **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 BDdesarrollo user usuariobd using contraseñabd;
insert into CMDREG values (id_enttienda_FashionFlow,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Description',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

donde

- *BDdesarrollo* es el nombre de la base de datos de desarrollo
- *usuariobd* es el usuario de base de datos
- *contraseñabd* es la contraseña para el usuario de base de datos
- *Id_enttienda_FashionFlow* es el identificador exclusivo de entidad de tienda para la tienda.

Pulse el icono **Ejecutar**.

2. Oracle Si está utilizando una base de datos Oracle, realice lo siguiente para registrar MyPostUserRegistrationAddCmdImpl:
   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 la siguiente sentencia SQL:

```
insert into CMDREG values (id_enttienda_FashionFlow,
'com.ibm.commerce.usermanagement.commands.PostUserRegistrationAddCmd'
'Description',
'com.ibm.commerce.sample.commands.MyPostUserRegistrationAddCmdImpl',
null, null, 'Local');
```

Pulse Intro para ejecutar la sentencia SQL.
   f. Entre lo siguiente para comprometer los cambios en la base de datos:

```
commit;
```

y pulse Intro para ejecutar la sentencia SQL.

Actualización de elementos de tienda en el WebSphere Commerce Server de destino

En este paso, actualice la tienda con los elementos de tienda modificados, como se indica a continuación:

1. Haga una copia de seguridad del directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear\Stores.war (donde *nombreCélula* es normalmente el nombre de sistema principal de la máquina).
2. Vaya al directorio: \ImportTemp4\Stores\Web Content.

3. Copie las carpetas *nombre_FashionFlow* y WEB-INF en el directorio siguiente: *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInst*.ear\Stores.war

donde *nombreInst* es el nombre de la instancia de WebSphere Commerce.

Actualización del archivo JAR de mandatos en el WebSphere Commerce Server de destino

En este paso actualizará el WebSphere Commerce Server de destino para utilizar el nuevo archivo JAR de mandatos, como se indica a continuación:

1. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.
   b. Haga una copia del archivo WebSphereCommerceServerExtensionsLogic.jar y guárdela en una ubicación de copia de seguridad.
2. Copie el nuevo archivo WebSphereCommerceServerExtensionsLogic.jar del directorio \ImportTemp4 en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear

donde *nombreInstancia* es el nombre de la instancia de WebSphere Commerce.

Actualización del archivo JAR EJB en el WebSphere Commerce Server de destino

En este paso actualizará el WebSphere Commerce Server de destino para utilizar el nuevo archivo JAR EJB, como se indica a continuación:

1. Deberá hacer una copia de seguridad del archivo JAR existente, del modo siguiente:
   a. Vaya al directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear.
   b. Haga una copia del archivo Member-MemberManagementData.jar y guárdela en una ubicación de copia de seguridad.
2. Copie el nuevo archivo Member-MemberManagementData.jar del directorio \ImportTemp4 en el directorio *dir_instal_WAS*\installedApps*nombreCélula*\WC_*nombreInstancia*.ear
3. A continuación, deberá modificar la información de descriptor de despliegue EJB, del modo siguiente:
   a. Localice el depósito de despliegue (directorio META-INF) para esta célula de WebSphere Application Server. Éste toma normalmente el formato siguiente:
   dir_instal_WAS\config\cells*nombreCélula*
   \applications\WC_*nombre_instancia*.ear\deployments\
   WC_*nombre_instancia**nombreMóduloEJB*.jar\META-INF.
   A continuación se muestra un ejemplo específico de esto:
   D:\WebSphere\AppServer\config\cells\myCell\applications\
   WC_demo.ear\deployments\WC_demo\ Member-
   MemberManagementData.jar\META-INF
   donde
   - myCell es el nombre de la célula de WebSphere Application Server
   - demo es el nombre de la instancia de WebSphere Commerce
   - Member-MemberManagementData es el nombre del módulo EJB personalizado

   b. El directorio contiene los archivos siguientes:

- ejb-jar.xml
- ibm-ejb-access-bean.xmi
- ibm-ejb-jar-bnd.xmi
- ibm-ejb-jar-ext.xmi
- MANIFEST.MF

Haga una copia de seguridad de todos estos archivos.

c. Utilice una herramienta para abrir el nuevo archivo Member-MemberManagementData.jar y ver su contenido.

d. Extraiga el contenido del directorio meta-inf de este archivo Member-MemberManagementData.jar en el directorio del paso 3a.

4. Utilizando el mandato startServer de WebSphere Application Server en la línea de mandatos, reinicie la instancia de WebSphere Commerce.

Verificación de la lógica de la encuesta sobre la vivienda en el WebSphere Commerce Server de destino

En este paso, verificará que la lógica de la encuesta sobre la vivienda se haya desplegado satisfactoriamente en el WebSphere Commerce Server de destino, realizando lo siguiente:

1. Abra un navegador Web y entre el URL para iniciar la tienda que está basada en la tienda de ejemplo FashionFlow.
2. Cuando se abra la página de presentación de la tienda, pulse **Registrar**.
3. Pulse **Registrar** otra vez para crear un usuario nuevo.
4. Aparecerá una página de registro modificada, que ahora contendrá la encuesta sobre la vivienda, tal como se muestra en la siguiente captura de pantalla:

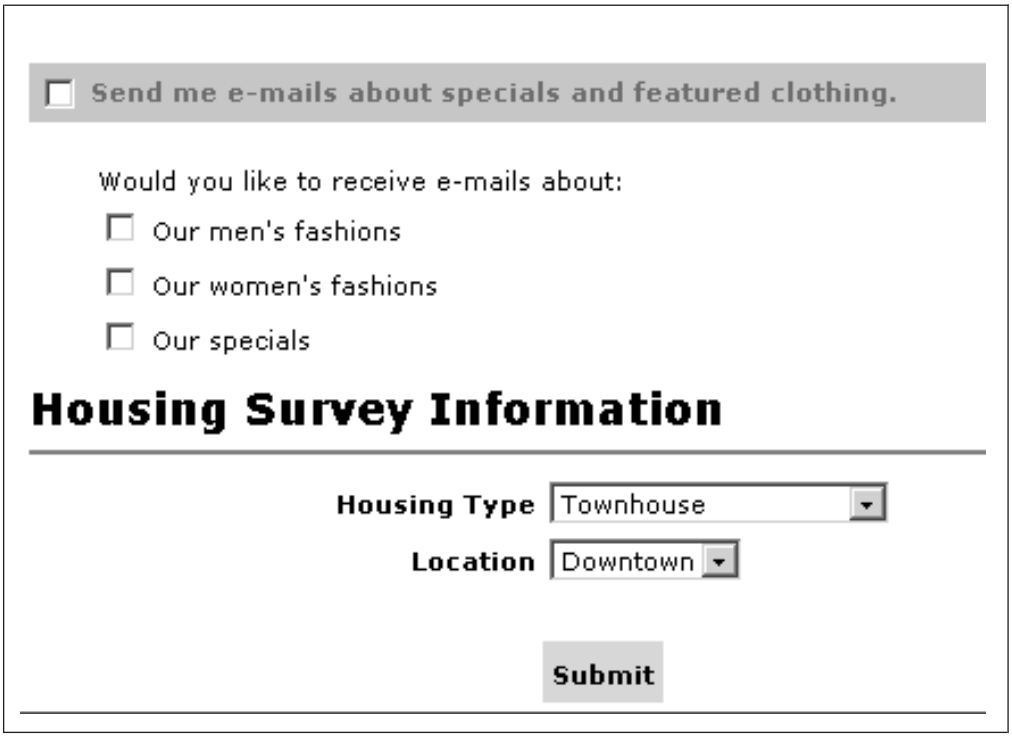


Figura 55.

5. Entre la información que sea apropiada para el nuevo usuario y pulse **Someter**.

6. Cuando la información se haya sometido, pulse **Modificar información personal** para verificar que la información sobre la vivienda se ha capturado. Aparecerá una pantalla que muestra un resumen de las respuestas a la encuesta, como se muestra a continuación:

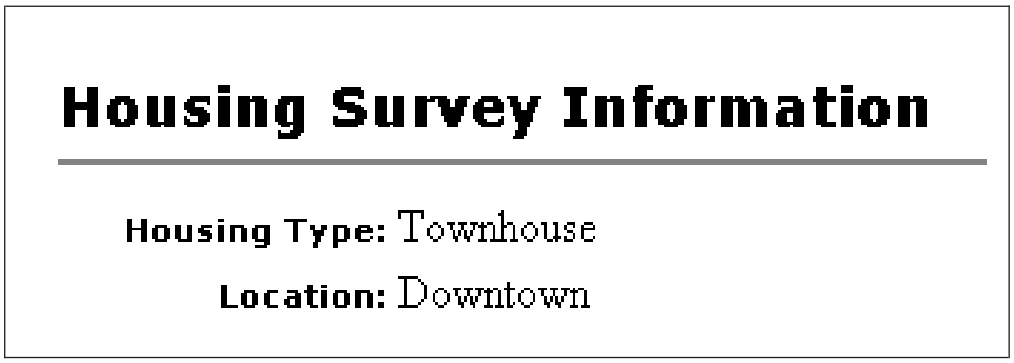


Figura 56.

Parte 5. Apéndices

Apéndice A. Configuración del rastreo de componentes de WebSphere Commerce en el Entorno de desarrollo de WebSphere Commerce

Este apéndice describe cómo habilitar el rastreo para los diversos componentes de WebSphere Commerce, cuando se ejecuta en el entorno de desarrollo de WebSphere Commerce. Para habilitar el rastreo de componentes, realice lo siguiente:

1. Si es necesario, abra el entorno de desarrollo de WebSphere Commerce (**Inicio > Programas > IBM el Entorno de desarrollo de WebSphere Commerce > Entorno de desarrollo de WebSphere Commerce**).
2. Conmute a la perspectiva de servidor.
3. En la vista Servidores, pulse con el botón derecho del ratón en **WebSphereCommerceServer** y seleccione **Detener** (si el servidor está en ejecución actualmente).
4. En la vista Configuración de servidor, expanda la carpeta **Configuraciones de servidor**.
5. Efectúe una doble pulsación en **WebSphereCommerceServer**.
   Se abre el editor WebSphereCommerceServer.
6. Seleccione la pestaña Rastreo.
7. Seleccione **Habilitar rastreo**.
8. En el recuadro de texto **Serie de rastreo**, especifique los componentes para los que se debe habilitar el rastreo. Utilice el valor de identificador de anotador de rastreo de extensiones JRas de WebSphere, seguido de `"=all=enabled"`. Para obtener una lista completa de estos valores, consulte el tema "Configuración" de la publicación *WebSphere Commerce, Guía de administración*. Se deberá utilizar el signo de dos puntos (:) para separar varios componentes. Por ejemplo, si desea habilitar el rastreo para los componentes SERVER y RAS, especifique la serie de rastreo del modo siguiente:

   ```
   com.ibm.websphere.commerce.WC_SERVER=all=enabled:
      com.ibm.websphere.commerce.WC_RAS=all=enabled
   ```

   Tenga en cuenta que la línea se muestra partida sólo a efectos de presentación.
9. Guarde los cambios (Control+S).

Archivo de salida

El archivo de anotaciones de salida se denomina activity.log por omisión. Este archivo está ubicado en el directorio siguiente:
dir_espaciotrabajo\.metadata\.plugin\com.ibm.etools.server.core\tmp0\logs

Debido al hecho de que el archivo activity.log es un archivo binario, se utiliza el Analizador de anotaciones para leer este archivo. Una vez que haya habilitado el rastreo de componentes, JRas de WebSphere también grabará en el archivo de salida de rastreo las entradas de anotación en formato de texto corriente junto con las entradas de rastreo.

Para obtener información sobre la configuración de la herramienta Analizador de anotaciones en el Entorno de desarrollo de WebSphere Commerce, consulte la

publicación Business Professional *WebSphere Commerce Studio, Guía de instalación* o Express *WebSphere Commerce - Express Developer Edition, Guía de instalación*.

Adicionalmente, se visualizan mensajes en la vista de Consola de WebSphere Studio Application Developer.

Apéndice B. Dónde encontrar más información

Se puede obtener más información sobre el sistema el Entorno de desarrollo de WebSphere Commerce y sus componentes en diversas fuentes de formatos diferentes. Las secciones siguientes indican qué información está disponible y cómo se accede a la misma.

Información de el Entorno de desarrollo de WebSphere Commerce

Las fuentes de información de el Entorno de desarrollo de WebSphere Commerce son las siguientes:

- "Ayuda en línea de el Entorno de desarrollo de WebSphere Commerce"
- "Sitio Web de WebSphere Commerce"
- "Dominio de desarrollador de WebSphere" en la página 334
- "Manuales técnicos de IBM" en la página 334

Ayuda en línea de el Entorno de desarrollo de WebSphere Commerce

La información en línea de el Entorno de desarrollo de WebSphere Commerce es la fuente principal de información para crear y publicar tiendas en el Entorno de desarrollo de WebSphere Commerce.

Para ver la ayuda en línea de el Entorno de desarrollo de WebSphere Commerce, realice lo siguiente:

1. Inicie el Entorno de desarrollo de WebSphere Commerce seleccionando **Inicio → Programas → IBM WebSphere Commerce Studio → Entorno de desarrollo de WebSphere Commerce**.
2. En el menú **Ayuda**, seleccione **Contenido de la ayuda**.

Nota: Si consulta las instrucciones para varias plataformas de la "Ayuda en línea de el Entorno de desarrollo de WebSphere Commerce", asegúrese de seguir las instrucciones para el Entorno de desarrollo de WebSphere Commerce. Cuando una página de ayuda contiene información para varias plataformas, la información específica de el Entorno de desarrollo de WebSphere Commerce se marca con el icono siguiente:

Si no hay disponible información específica de el Entorno de desarrollo de WebSphere Commerce, asegúrese de seguir las instrucciones específicas de Windows. Cuando una página de ayuda contiene instrucciones para varias plataformas, las instrucciones para Windows se marcan con el icono siguiente:

Sitio Web de WebSphere Commerce

En el sitio Web de WebSphere Commerce, hay disponible información sobre el producto el Entorno de desarrollo de WebSphere Commerce. Consulte los URL siguientes para obtener más información sobre el producto.

http://www.ibm.com/software/webservers/commerce/library/

Dominio de desarrollador de WebSphere

También hay disponible información adicional sobre el Entorno de desarrollo de WebSphere Commerce y WebSphere Commerce en la Zona de WebSphere Commerce en el dominio de desarrollador de WebSphere:

http://www.ibm.com/websphere/developer/zones/commerce/

Manuales técnicos de IBM

Hay información de el Entorno de desarrollo de WebSphere Commerce y WebSphere Commerce disponible en el sitio Web de IBM Redbooks:

http://www.ibm.com/redbooks

Información de WebSphere Studio Application Developer

Las fuentes de información para WebSphere Studio Application Developer son las siguientes:

- "Ayuda en línea de WebSphere Studio Application Developer"
- "Sitio Web de WebSphere Studio Application Developer"
- "Dominio de desarrollador de WebSphere"
- "Manuales técnicos de IBM"

Ayuda en línea de WebSphere Studio Application Developer

La ayuda en línea de WebSphere Studio Application Developer es la fuente principal de información sobre cómo realizar tareas en WebSphere Studio Application Developer.

Para ver la ayuda en línea de WebSphere Studio Application Developer, realice lo siguiente:

1. Inicie el Entorno de desarrollo de WebSphere Commerce seleccionando **Inicio → Programas → IBM WebSphere Studio → Application Developer 5.0**.
2. En el menú **Ayuda**, seleccione **Contenido de la ayuda**.

Sitio Web de WebSphere Studio Application Developer

En el sitio Web de WebSphere Studio Application Developer, hay disponible información sobre el producto WebSphere Studio Application Developer:

http://www.ibm.com/software/ad/studioappdev/library/

Dominio de desarrollador de WebSphere

Hay disponible información adicional sobre WebSphere Studio Application Developer en la página de WebSphere Studio Application Developer de la zona de Studio de WebSphere del dominio de desarrollador de WebSphere:

http://www.ibm.com/websphere/developer/zones/studio/appdev/

Manuales técnicos de IBM

Hay información de WebSphere Studio Application Developer disponible en el sitio Web de IBM Redbooks:

http://www.ibm.com/redbooks

Avisos

Esta información se ha desarrollado para productos y servicios ofrecidos en EE.UU. 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
EE.UU.

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
Tokio 106, Japón

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 propietarios de licencia 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

El logotipo de IBM y los siguientes términos son marcas registradas de International Business Machines Corporation en EE UU y/o en otros países:

400
AIX
AS/400
DB2
DB2 Universal Database
@server
IBM
iSeries
WebSphere

Windows es una marca registrada de Microsoft Corporation en EE UU y/o en otros países.

Java y todas las marcas registradas y los logotipos basados en Java son marcas registradas de Sun Microsystems, Inc. en EE UU y/o en otros países.

Otros nombres de empresas, productos y servicios, pueden ser marcas registradas o marcas de servicio de otras empresas.

Índice