IBM® WebSphere™ Commerce

WebSphere Commerce Accelerator, Guía de personalización

Versión 5.4

IBM® WebSphere™ Commerce

WebSphere Commerce Accelerator, Guía de personalización

Versión 5.4

Nota:
Antes de utilizar esta información y el producto al que da soporte, lea la información general del apartado "Avisos" en la página 93.

Primera edición (enero de 2002).

Esta edición se aplica a la versión 5.4 de IBM WebSphere Commerce, así como a todos los releases y las modificaciones subsiguientes hasta que se indique lo contrario en nuevas ediciones. Asegúrese de que está utilizando la edición correcta para el nivel del producto.

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

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

1. Por correo electrónico, a la dirección que se indica más abajo. Si desea obtener una respuesta, asegúrese de incluir su dirección de correo electrónico completa.

   `Internet: hojacom@vnet.ibm.com`

2. Por FAX, al número siguiente:

   `34 93 321 6134`

3. Por correo, a la dirección siguiente:

   IBM S.A.
   National Language Solutions Center
   Av. Diagonal 571,
   ″Ed. L'Illa″
   08029 Barcelona
   España

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

Contenido

Antes de empezar

Convenios utilizados en este manual

En este manual se utilizan los convenios de resaltado siguientes:

La **negrita** indica mandatos o controles de interfaz gráfica de usuario (GUI), como nombres de campos, botones u opciones de menú.

El `monoespaciado` indica ejemplos de texto que se escriben exactamente como se muestra, así como vías de acceso de directorios.

La *cursiva* se utiliza para dar énfasis y para las variables que se deben sustituir por valores propios.

Este icono indica un Consejo — información adicional que puede ayudarle a realizar una tarea.

Conocimientos necesarios

Para personalizar WebSphere Commerce Accelerator, necesita tener conocimientos sobre los temas siguientes:

- HTML y XML
- Lenguaje de Consulta Estructurada (SQL)
- Programación Java

Consulte la publicación WebSphere Commerce, Guía del programador si desea obtener más información sobre cómo personalizar WebSphere Commerce. Este manual está disponible en el sitio Web siguiente:

`www.ibm.com/software/webservers/commerce/wcs_pro/lit-tech-general.html`

Cómo está organizado este manual

La tabla siguiente describe la organización del manual:

Tabla 1.

Componente	Descripción	Ubicación
Gestión de relaciones comerciales	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de Contratos	Capítulo 1, "Gestión de relaciones comerciales" en la página 3
Representante de servicio al cliente	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas del Representante de servicio al cliente	Capítulo 2, "Herramientas del Representante de servicio al cliente" en la página 9
Perfiles de cliente	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de Perfil de cliente	Capítulo 4, "Perfiles de cliente" en la página 21

Tabla 1. (continuación)

Componente	Descripción	Ubicación
Campañas	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de Campañas	Capítulo 5, "Campañas" en la página 29
Cupones	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de Cupones	Capítulo 7, "Cupones" en la página 53
Descuentos	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de Descuentos	Capítulo 8, "Descuentos" en la página 55
RFQ	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de RFQ	Capítulo 9, "Respuesta a RFQ" en la página 57
Inventario	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de Inventario	Capítulo 10, "Inventario esperado" en la página 69
Business Intelligence	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de informes	Capítulo 11, "Business intelligence" en la página 71
Infraestructura de informes	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de informes	Capítulo 12, "Visión general de la infraestructura de informes" en la página 73
Búsqueda	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas de búsqueda de catálogo	Capítulo 6, "Búsqueda de catálogo" en la página 37
Blaze Rules Advisor	Esta sección detalla los elementos que debe tener en cuenta al personalizar los proyectos de normas de Brokat Advisor	Capítulo 14, "Proyectos de normas" en la página 89
Asesor de productos	Esta sección detalla los elementos que debe tener en cuenta al personalizar las herramientas del Asesor de productos	Capítulo 13, "Asesor de productos" en la página 87

Parte 1. Personalización de WebSphere Commerce Accelerator

Este manual describe cómo personalizar WebSphere Commerce Accelerator. Muestra el modo en que debe plantearse la personalización y detalla los pasos necesarios para la misma, proporcionando los conocimientos básicos acerca de las decisiones de diseño para WebSphere Commerce Accelerator.

Aunque este manual está destinado a ayudarle a seguir el proceso de personalización de WebSphere Commerce Accelerator, no está destinado a guiarle a lo largo de una lista exhaustiva de todas las personalizaciones que son posibles. En lugar de ello, la estructura del manual presenta los diferentes elementos genéricos que forman WebSphere Commerce Accelerator y explica cómo personalizar dichos elementos.

WebSphere Commerce Accelerator es un conjunto de herramientas diseñadas para facilitar las operaciones de negocio diarias del sitio. Es decir, está pensado como una interfaz para que la gente de negocio cree y modifique diversos aspectos de las operaciones del sitio, sin tener que ponerse en contacto continuamente con el personal de IT responsable del funcionamiento del sitio. Como tal, WebSphere Commerce Accelerator está formado por numerosos componentes, cada uno de los cuales se encarga de un aspecto clave del negocio. A pesar de que se han hecho todos los esfuerzos para que estas herramientas fueran universales, puede que, por esta razón, algunos usuarios encuentren que se no satisfacen determinados requisitos. Si ha ampliado la base de datos durante la creación del sitio y estas ampliaciones están relacionadas con cualquiera de las herramientas listadas en este documento, los datos no serán visibles en Accelerator hasta que personalice la herramienta pertinente.

Este manual describe cómo personalizar los componentes siguientes de WebSphere Commerce Accelerator.

Capítulo 1. Gestión de relaciones comerciales

Los elementos descritos en este capítulo representan la interfaz de usuario para los componentes Cuentas y Contratos de WebSphere Commerce Accelerator. Dichos elementos facilitan la creación y el mantenimiento de cuentas, contratos y otras acciones que un Director de ventas o un Representante de cuentas necesita para realizar las tareas diarias asociadas con el negocio. El componente de gestión de relaciones comerciales incluye los elementos siguientes:

- Cuaderno Cuentas
- Cuaderno Contratos

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar esta parte de WebSphere Commerce Accelerator.

Escenario 1: Añadir un término y condición nuevo a la interfaz de usuario de Contrato

Visión general de la implementación

En un intento de hacer que WebSphere Commerce Accelerator fuera genérico y proporcionara la máxima utilidad al mayor número de personas posible, se han omitido algunos términos y condiciones. Por consiguiente, puede que encuentre que es necesario añadir términos y condiciones al sitio para que éste sea más aplicable al negocio. Este escenario le guía durante los pasos necesarios para añadir términos y condiciones adicionales al sitio.

Pasos para la personalización

1. Defina el término y condición en el servidor. Para poder realizar la personalización de la interfaz de usuario, ya tiene que haber completado las acciones siguientes:
   - Definir un término y condición nuevo en el archivo `B2BTrading.dtd`. Para obtener detalles completos, consulte el Capítulo 7 de la publicación *IBM WebSphere Commerce, Guía del programador*.
   - Crear un Bean Enterprise o un Bean de acceso para el nuevo término y condición.
2. Añada el término y condición a la interfaz de usuario. Para añadir los elementos necesarios a la interfaz de usuario, realice lo siguiente:
   a. Cree un archivo JSP para la página de interfaz de usuario para el término y condición. Esta página debe incluir un objeto JavaScript dinámico para capturar datos de los campos de entrada en la página. Cree un bean de datos que encapsule la carga de los datos desde el Bean de acceso. Opcionalmente, incluya el Bean de acceso directamente en la página para cargar los datos de términos y condiciones de la base de datos. El parámetro `contractId` se pasa a todas las páginas para ayudar a reducir la carga de datos. A continuación se muestra un ejemplo de este archivo JSP:

```
<!--============================================================================
//*----------------------------------------------------------------------------
//* El ejemplo aquí contenido se proporciona "TAL CUAL".
//*
//* IBM lo proporciona como simple ejemplo y no se ha probado de forma minuciosa
//* bajo todas las condiciones. Por consiguiente, IBM no puede garantizar su fiabilidad,
//* servicio o funcionalidad.
//*
```

```
//* Este ejemplo puede incluir los nombres de personas, empresas, marcas y
//* productos para ilustrar los conceptos de la forma más completa posible. Todos
//* estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados
//* por personas o empresas reales es pura coincidencia.
//*-------------------------------------------------------------------------
//*
  ========================================================================-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<%@page language="JAVA"
import="com.ibm.commerce.tools.util.UIUtil,
com.ibm.commerce.beans.DataBeanManager,
com.ibm.commerce.tools.contract.beans.MemberDataBean,
com.ibm.commerce.tools.contract.beans.MyTCDataBean,
com.ibm.commerce.tools.contract.beans.PolicyDataBean,
com.ibm.commerce.tools.contract.beans.PolicyListDataBean"
%>

<%@include file="../common/common.jsp" %>
<%@include file="ContractCommon.jsp" %>

<HTML>

<HEAD>
 <%= fHeader %>
 <LINK rel="stylesheet" href="<%= UIUtil.getCSSFile(fLocale) %>" type="text/css">

 <TITLE><%= contractsRB.get("MyTCHeading") %></TITLE>
 <SCRIPT LANGUAGE="JavaScript" SRC="/wcs/javascript/tools/common/Util.js"></SCRIPT>
 <SCRIPT LANGUAGE="JavaScript" SRC="/wcs/javascript/tools/contract/ContractUtil.js"></SCRIPT>
 <SCRIPT LANGUAGE="JavaScript" SRC="/wcs/javascript/tools/contract/Extensions.js"></SCRIPT>


<SCRIPT LANGUAGE="JavaScript">

var MyTCModel;

////////////////////////////////////////////
// SCRIPTS DE CARGA-SALVAGUARDA-VALIDACIÓN
////////////////////////////////////////////
function onLoad() {

   if (parent.setContentFrameLoaded) {
       parent.setContentFrameLoaded(true);
   }

   // Comprobar si el modelo ya se ha cargado
   var isModelLoaded = parent.get("ContractMyTCModelLoaded", null);

   if (isModelLoaded) {
       // Al volver a esta página, volver a cargar desde el modelo
       // Obtener el modelo
       MyTCModel = parent.get("ContractMyTCModel", null);
   }
   else {
       // Primera visita a esta página, crear el modelo

       // Crear el modelo para almacenar los datos de MyTC
       MyTCModel = new ContractMyTCModel();

       // Conservar el modelo
       parent.put("ContractMyTCModel", MyTCModel);
       parent.put("ContractMyTCModelLoaded", true);

var myTCPolicyList = new Array();
<%
try {
// Cargar todas las políticas de la base de datos
PolicyListDataBean policyList = new PolicyListDataBean();
PolicyDataBean policy[] = null;

policyList.setPolicyType(policyList.TYPE_PRODUCT_SET);
DataBeanManager.activate(policyList, request);
policy = policyList.getPolicyList();

for (int i = 0; i < policy.length; i++) {
MemberDataBean mdb = new MemberDataBean();
mdb.setId(policy[i].getStoreMemberId());
DataBeanManager.activate(mdb, request);
%>
myTCPolicyList[myTCPolicyList.length] =
new PolicyObject('<%=UIUtil.toJavaScript(policy[i].getShortDescription())%>',
'<%= policy[i].getPolicyName() %>',
'<%= policy[i].getId() %>',
'<%= policy[i].getStoreIdentity() %>',
new Member('<%= mdb.getMemberType() %>',
'<%= mdb.getMemberDN() %>',
'<%= mdb.getMemberGroupName() %>',
'<%= mdb.getMemberGroupOwnerMemberType() %>',
'<%= mdb.getMemberGroupOwnerMemberDN() %>')
);
<%
}
} catch (Exception e) {}
%>
MyTCModel.policyList = myTCPolicyList;

// Comprobar si se trata de una actualización del contrato
if (<%= foundContractId %> == true) {
// Cargar los datos del bean de datos
<%
if (foundContractId) {
MyTCDataBean tc = new MyTCDataBean(new Long(contractId));
DataBeanManager.activate(tc, request);
if (tc.getHasMyTC()) {
%>
```

```
MyTCModel.attr1 = '<%= UIUtil.toJavaScript((String)tc.getAttr1()) %>';
MyTCModel.attr2 = '<%= UIUtil.toJavaScript((String)tc.getAttr2()) %>';
MyTCModel.tcReferenceNumber = '<%= tc.getReferenceNumber() %>';
MyTCModel.policyReferenceNumber = '<%= tc.getPolicyReferenceNumber() %>';
for (var i = 0; i < myTCPolicyList.length; i++) {
if (myTCPolicyList[i].policyId == '<%= tc.getPolicyReferenceNumber() %>') {
MyTCModel.selectedPolicyIndex = i;
}
}
<%
}
}
%>
}

   }

   loadPanelData();

    // volver a manejar mensajes de error de la página de validación
    if (parent.get("attr1Empty", false))
     {
      parent.remove("attr1Empty");
      alertDialog("<%= UIUtil.toJavaScript((String)contractsRB.get("attr1Empty"))%>");
     }
    else if (parent.get("attr2Empty", false))
     {
      parent.remove("attr2Empty");
      alertDialog("<%= UIUtil.toJavaScript((String)contractsRB.get("attr2Empty"))%>");
     }
    else if (parent.get("attr1TooLong", false))
     {
      parent.remove("attr1TooLong");
      alertDialog("<%= UIUtil.toJavaScript((String)contractsRB.get("attr1TooLong"))%>");
     }
    else if (parent.get("attr2TooLong", false))
     {
      parent.remove("attr2TooLong");
      alertDialog("<%= UIUtil.toJavaScript((String)contractsRB.get("attr2TooLong"))%>");
     }

   return;
}

function loadPanelData() {

// Establecer los campos de entrada
document.MyTCForm.Attr1.value = MyTCModel.attr1;
document.MyTCForm.Attr2.value = MyTCModel.attr2;

// Cargar las políticas
for (var i = 0; i < MyTCModel.policyList.length; i++) {
if (MyTCModel.selectedPolicyIndex == i) {
document.MyTCForm.PolicyList.options[i] = new Option(MyTCModel.policyList[i].displayText,
i, true, true);
} else {
   document.MyTCForm.PolicyList.options[i] = new Option(MyTCModel.policyList[i].displayText,
i, false, false);
}
}
}

function savePanelData() {
    MyTCModel.attr1 = document.MyTCForm.Attr1.value;
    MyTCModel.attr2 = document.MyTCForm.Attr2.value;
    MyTCModel.selectedPolicyIndex = document.MyTCForm.PolicyList.selectedIndex;
}
</SCRIPT>

</HEAD>

<!--
////////////////////////////////////////
// SECCIÓN HTML
////////////////////////////////////////
-->

<BODY onLoad="onLoad()" class="content">

   <H1>
   <%= contractsRB.get("MyTCHeading") %>
   </H1>

<FORM NAME="MyTCForm">

   <%= contractsRB.get("MyTCAttr1Label") %>
   <BR>
   <INPUT type=text name=Attr1 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCAttr2Label") %>
   <BR>
   <INPUT type=text name=Attr2 value="" size=10 maxlength=10>
   <BR>

   <%= contractsRB.get("MyTCPolicyLabel") %>
   <BR>
   <SELECT NAME="PolicyList" SIZE="1">
   </SELECT>

</FORM>
```

```
</BODY>
</HTML>
```

b. Cree un archivo JavaScript para la página de interfaz de usuario para el término y condición. Cree funciones para validar y someter los datos JavaScript. Cuando someta los datos, necesitará crear un objeto JavaScript nuevo que coincida con el formato XML para el nuevo término y condición. A continuación se muestra un ejemplo del archivo JavaScript:

```
//*-------------------------------------------------------------------------------------
//* El ejemplo aquí contenido se proporciona "TAL CUAL".
//*
//* IBM lo proporciona como simple ejemplo y no se ha probado de forma minuciosa
//* bajo todas las condiciones. Por consiguiente, IBM no puede garantizar su fiabilidad,
//* servicio o funcionalidad.
//*
//* Este ejemplo puede incluir los nombres de personas, empresas, marcas y productos
//* para ilustrar los conceptos de la forma más completa posible. Todos estos nombres
//* son ficticios y cualquier similitud con nombres y direcciones utilizados por personas
//* o empresas reales es pura coincidencia.
//*-------------------------------------------------------------------------------------
//*

function ContractMyTCModel() {

  this.tcReferenceNumber = "";
  this.policyReferenceNumber = "";

  this.attr1 = "";
  this.attr2 = "";

  this.policyList = new Array();
  this.selectedPolicyIndex = "0";
}

function validateMyTCPanel() {

  var tcModel = get("ContractMyTCModel");
  if (tcModel != null) {
  // Comprobar si attr1 está vacío o es demasiado largo
  if (!tcModel.attr1)
   {
    put("attr1Empty", true);
    gotoPanel("MyTCHeading");
    return false;
   }

  if (!isValidUTF8length(tcModel.attr1, 10))
   {
    put("attr1TooLong", true);
    gotoPanel("MyTCHeading");
    return false;
   }
  // Comprobar si attr2 está vacío o es demasiado largo
  if (!tcModel.attr2)
   {
    put("attr2Empty", true);
    gotoPanel("MyTCHeading");
    return false;
   }

  if (!isValidUTF8length(tcModel.attr2, 10))
   {
    put("attr2TooLong", true);
    gotoPanel("MyTCHeading");
    return false;
   }
  }

}

function submitMyTC(termsAndConditions) {

  var tcModel = get("ContractMyTCModel");

  if (tcModel != null) {

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

myTC.MyTC.PolicyReference = new Object();
myTC.MyTC.PolicyReference.policyName = tcModel.policyList[tcModel.selectedPolicyIndex].policyName;
myTC.MyTC.PolicyReference.policyType = "ProductSet";
myTC.MyTC.PolicyReference.storeIdentity = tcModel.policyList[tcModel.selectedPolicyIndex].storeIdentity;
myTC.MyTC.PolicyReference.Member = tcModel.policyList[tcModel.selectedPolicyIndex].member;

if (tcModel.tcReferenceNumber != "") {
// Cambiar el término y condición
myTC.action = "update";
myTC.referenceNumber = tcModel.tcReferenceNumber;
}
else {
// Crear un término y condición nuevo
myTC.action = "new";
}
```

```
termsAndConditions[termsAndConditions.length] = myTC;
  }

  return true;
}

function validateContractExtensions() {
   if (this.validateMyTCPanel) {
     if (validateMyTCPanel() == false) {
return false;
     }
   }
   return true;
}

function submitContractExtensions(termsAndConditions) {
   if (this.submitMyTC) {
     if (submitMyTC(termsAndConditions) == false) {
return false;
     }
   }
}
```

c. Modifique el archivo `Contract.js` para llamar a las funciones `submit` y `validate` nuevas creadas en el paso 2b.

d. Modifique el paquete de recursos para añadir las series nuevas que necesite.

e. Modifique `ContractNotebook.xml` para añadir el panel nuevo.

Escenario 2: Eliminar un término y condición de la interfaz de usuario de Contrato

Visión general de la implementación

Del mismo modo que puede encontrar que faltan algunos términos y condiciones, puede que también encuentre que algunos no son necesarios para su negocio. Este escenario le guía durante los pasos necesarios para eliminar del sitio términos y condiciones ajenos.

Pasos para la personalización

Modifique el archivo `ContractNotebook.xml` y elimine la sección que contiene el elemento de panel para el término y condición que desea eliminar. Localice este archivo en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/xml/tools/contract/`

400 `/QIBM/UserData/WebSphere/CommerceServer/CommerceServer/xml/tools/contract/`

Linux `/opt/WebSphere/CommerceServer/xml/tools/contract/`

Solaris `/opt/WebSphere/CommerceServer/xml/tools/contract/`

Windows *unidad*`:\WebSphere\CommerceServer\xml\tools\contract\`

Capítulo 2. Herramientas del Representante de servicio al cliente

Los elementos descritos en este capítulo representan la interfaz de usuario para las herramientas del Representante de servicio al cliente (CSR) de WebSphere Commerce Accelerator. Facilitan la creación y el mantenimiento de clientes, pedidos y otras acciones que un CSR necesita para realizar las tareas diarias asociadas con el negocio. Las herramientas del componente CSR incluyen los elementos siguientes:

- Asistente de cliente
- Cuaderno Cliente
- Asistente de pedido
- Cuaderno Pedido

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar las herramientas del CSR en WebSphere Commerce Accelerator.

Escenario 1: Añadir datos de cliente adicionales al diálogo Resumen del cliente de negocio

Visión general de la implementación

Este ejemplo muestra cómo añadir el número de empleado de un Cliente de negocio al diálogo Resumen del cliente de negocio.

Paso para la personalización

Amplíe el archivo `ShopperSummaryB2BDialog.jsp`. Localice este archivo en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/web/tools/csr/`

400 `/QIBM/UserData/WebSphere/CommerceServer/CommerceServer/web/tools/csr/`

Linux `/opt/WebSphere/CommerceServer/web/tools/csr/`

Solaris `/opt/WebSphere/CommerceServer/web/tools/csr/`

Windows *unidad*`:\WebSphere\CommerceServer\web\tools\csr\`

Consulte el código de ejemplo que se proporciona a continuación para saber cómo cambiar el archivo JSP a fin de visualizar el número de empleado en el diálogo de resumen. El número de empleado es propiedad de `OptoolsRegisterDataBean`. El ejemplo no utiliza archivos de propiedades para almacenar la etiqueta `Número de empleado`. La salida de ejemplo que tiene el número de empleado se visualiza en el diálogo Resumen del cliente:

```
<%--
//*******************************************************************
//*-------------------------------------------------------------------
//* Materiales bajo licencia - Propiedad de IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001
//*
//* Derechos restringido de los usuarios del Gobierno de EE.UU. - Utilización,
```

```
//* duplicación o divulgación restringida por el GSA ADP Schedule Contract
//* con IBM Corp.
//*
//*-------------------------------------------------------------------
//*
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
:
:
: Aquí se ha omitido parte del código
:
:
<HTML>

<HEAD>
<LINK rel=stylesheet
        href="<%= UIUtil.getCSSFile(cmdContext.getLocale()) %>"
        type="text/css">
<SCRIPT SRC="/wcs/javascript/tools/common/Util.js"></SCRIPT>

<script>
<%@ include file = "SummaryDisplay.jsp" %>
:
:
: Aquí se ha omitido parte del código
:
:
</script>

</HEAD>
<BODY CLASS=content onLoad = "initializeState();">
<p>
</p>
<FORM NAME="profile" action="" Method="POST">

   <INPUT type="hidden" name="logonId" value="">
   <INPUT type="hidden" name="XML" value="">
   <INPUT type="hidden" name="URL" value="">

<h1><%= userNLS.get("customerSummaryTitle") %></h1>
<P><B><%=userNLS.get("generalHeader")%></B>
<BR>
   <%=userNLS.get("logonid")%>
   <I><%=UIUtil.toHTML(registerDataBean.getLogonId()) %></I>
<BR>
   <%=userNLS.get("custName")%>:
   <I><script>displayNameSummary()</script></I>
<BR>
   Número de empleado:
   <I><%=UIUtil.toHTML(registerDataBean.getEmployeeId()) %></I>
<BR>
   <%=userNLS.get("orgAccount")%>:
<% if (accountDBean != null) { %>
   <I><%=UIUtil.toHTML(accountDBean.getAccountName()) %></I>
<% } %>
<BR>
   <%=userNLS.get("challengeQuestion")%>:
   <I><%=UIUtil.toHTML(registerDataBean.getChallengeQuestion())%></I>
<BR>
   <%=userNLS.get("challengeAnswer")%>:
   <I><%=UIUtil.toJavaScript(registerDataBean.getChallengeAnswer())%></I>
<BR>
   <%=userNLS.get("clientCertificate")%>:
   <I>
<% if (certStatus != "") { %>
   <% if (certStatus == "V") {%><%=userNLS.get("valid")%><%}%>
   <% if (certStatus == "E") {%><%=userNLS.get("expired")%><%}%>
   <% if (certStatus == "R") {%><%=userNLS.get("revoked")%><%}%>
<% } else { %>
   <%=userNLS.get("noCertificate")%>
<% } %>
   </I>
<BR>
   <%=userNLS.get("status")%>:
   <I>
<% if (userRegistry.getStatus().equals("1")) {%>
   <%=userNLS.get("accountStatusEnabled")%>
<% } else { %>
   <%=userNLS.get("accountStatusDisabled")%>
```

```
<% } %>
   </I>
<BR>
<P><B><%=userNLS.get("contactHeader")%></B>
   <TABLE border="0" CELLPADDING=0 CELLSPACING=0 >
      <TR valign="top">
          <TD><%=userNLS.get("address")%>:</TD>
          <TD><I><script>displayAddrSummary(0)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(1)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(2)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(3)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(4)</script></I></TD>
      </TR>
   </TABLE>
<%=userNLS.get("phone")%>:
   <I><%=UIUtil.toHTML(address.getPhone1())%></I>
<BR>
   <%=userNLS.get("fax")%>:
   <I><%=UIUtil.toHTML(address.getFax1())%></I>
<BR>
   <%=userNLS.get("email")%>:
   <I><%=UIUtil.toHTML(address.getEmail1())%></I>
<BR>
<P><B><%=userNLS.get("orgHeader")%></B>
<BR>
   <TABLE border="0" CELLPADDING=0 CELLSPACING=0 >
      <TR valign="top">
         <TD><%=orgEntityNLS.get("OrgEntityDeliveryDescription")%>:</TD>
         <TD>
<% if (orgEntity != null) { %>
   <I><%=UIUtil.toHTML(orgEntity.getDescription())%></I>
<% } %>
         </TD>
      </TR>
   </TABLE>
   <TABLE border="0" CELLPADDING=0 CELLSPACING=0 >
      <TR valign="top">
         <TD><%=orgEntityNLS.get("OrgEntityGeneralBusCat")%>:</TD>
         <TD>
<% if (orgEntity != null) { %>
      <I><%=UIUtil.toHTML(orgEntity.getBusinessCategory())%></I>
<% } %>
         </TD>
      </TR>
   </TABLE>

</FORM>
<%
}
catch (Exception e)
{
   e.printStackTrace();
}

%>

</BODY>
</HTML>
```

Escenario 2: Habilitar el Carro de la compra en la interfaz de Servicio al cliente - Gestión de pedidos

Visión general de la implementación

Si el diseñador de la tienda desea gestionar los pedidos del comprador que están en estado `P` utilizando las herramientas del CSR, deberá modificar las vistas **CSROrderSearchB2B** y **CSROrderSearchB2C**.

Paso para la personalización

Amplíe los archivos JSP `CSROrderSearchB2B.jsp` y `CSROrderSearchB2B.jsp`. Si se añade la opción `Pendiente` en los criterios de estado de pedido, el CSR podrá buscar los pedidos pendientes del cliente y gestionarlos. Los pedidos pendientes son los pedidos que están en el carro de la compra. Estos dos archivos están en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/web/tools/order/`

400 `/QIBM/UserData/WebSphere/CommerceServer/CommerceServer/web/tools/order/`

Linux `/opt/WebSphere/CommerceServer/web/tools/order/`

Solaris `/opt/WebSphere/CommerceServer/web/tools/order/`

Windows *unidad*`:\WebSphere\CommerceServer\web\tools\order\`

Consulte el código de ejemplo siguiente:

```
<%--
//*************************************************************************
//*-------------------------------------------------------------------------
//* Materiales bajo licencia - Propiedad de IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001
//*
//* Derechos restringido de los usuarios del Gobierno de EE.UU. - Utilización,
//* duplicación o divulgación restringida por el GSA ADP Schedule Contract
//* con IBM Corp.
//*
//*-------------------------------------------------------------------------
//* --%>
<%@ page language="java" %>
<%@ page import="java.util.*" %>
<%@ page import="com.ibm.commerce.tools.util.*" %>
<%@ page import="com.ibm.commerce.command.CommandContext" %>
<%@ page import="com.ibm.commerce.server.*" %>
<%@ page import="com.ibm.commerce.tools.optools.user.beans.*" %>
<%@ page import="com.ibm.commerce.tools.optools.order.commands.*" %>
<%@ page import="com.ibm.commerce.tools.contract.beans.*" %>
<%@ page import="com.ibm.commerce.beans.*" %>
<%@ page import="com.ibm.commerce.tools.util.UIUtil" %>
<%@include file="../common/common.jsp" %>

<%! public String getUserLogon(String customerId, HttpServletRequest request) {
   try {
      if (customerId != null && !customerId.equals("")) {
         OptoolsRegisterDataBean userBean = new OptoolsRegisterDataBean();
         userBean.setUserId(customerId);
         DataBeanManager.activate(userBean, request);

         if (userBean.getLogonId() != null)
            return userBean.getLogonId();
      }
   } catch (Exception ex) {
      return "";
   }
   return "";
}
%>

<HTML>
```

```
<HEAD>
<%
   // obtener el paquete de recursos para visualizarlo
   CommandContext cmdContextLocale =
     (CommandContext)request.getAttribute(com.ibm.commerce.server.ECConstants.EC_COMMANDCONTEXT);
   Locale jLocale = cmdContextLocale.getLocale();
   Hashtable orderLabels =
     (Hashtable)ResourceDirectory.lookup("order.orderLabels", jLocale);

      // recuperar parámetros de petición
      JSPHelper jspHelp = new JSPHelper(request);
      String customerId =
        jspHelp.getParameter(ECOptoolsConstants.EC_OPTOOL_CUSTOMER_ID);

      if (customerId == null) {
         customerId = "";
      }

      //componer lista entera de nombres de cuenta
      AccountListDataBean acctListDB = new AccountListDataBean();
      DataBeanManager.activate(acctListDB, request);
      AccountDataBean[] acctList = acctListDB.getAccountList();

%>
<link rel=stylesheet
   href="<%= UIUtil.getCSSFile(jLocale) %>" type="text/css">
<TITLE></TITLE>
<SCRIPT SRC="/wcs/javascript/tools/common/Util.js"></SCRIPT>
<SCRIPT LANGUAGE="JavaScript">
<!---- ocultar script de navegadores antiguos

function initializeState()
{
   parent.setContentFrameLoaded(true);
}

function savePanelData()
{
}

function onLoad() {
   initializeState()
}

function isEmpty(id) {
   return !id.match(/[^\s]/);
}
function isNumber(word)
{
   var numbers="0123456789";
   for (var i=0; i < word.length; i++)
   {
      if (numbers.indexOf(word.charAt(i)) == -1)
      return false;
   }
   return true;
}
function formValid() {
   var invalidChars = /[,;&#%|"'\\/]/
   if (document.orderFindForm.orderId.value.match(invalidChars) ||
      document.orderFindForm.userLogon.value.match(invalidChars))
   {
      // alert(",;no es válido");
      alertDialog("<%=UIUtil.toJavaScript(orderLabels.get
               ("invalidChars").toString()) %>");
      return false;
   }
   return true;
}

function validateEntries()
{
   if (isEmpty(document.orderFindForm.orderId.value)
    && isEmpty(document.orderFindForm.userLogon.value)
    && (document.orderFindForm.orderState.value == "all")
    && isEmpty(document.orderFindForm.accountId.value)) {
      alertDialog('<%=UIUtil.toJavaScript((String)orderLabels.get("findDialogNoCriteria"))%>');
      return false;
   } else
```

```
   if (!isEmpty(document.orderFindForm.orderId.value)) {
      if (!isNumber(document.orderFindForm.orderId.value)) {
         alertDialog ('<%=UIUtil.toJavaScript((String)orderLabels.get
                  ("findDialogInvalidNumber"))%>');
         return false;
      }
   } else if (!formValid()) {
      return false;
   }

   return true;
}
function findAction() {
   if (validateEntries() == true) {
      url = '/webapp/wcs/tools/servlet/NewDynamicListView';
      var urlPara = new Object();
      urlPara.listsize='22';
      urlPara.startindex='0';
      if ("<%=customerId%>" != "") {
         urlPara.ActionXMLFile='order.csadminOrderListB2B';
      } else {
         urlPara.ActionXMLFile='order.csOrderListB2B';
      }
      urlPara.cmd='OrderListViewB2B';
      urlPara.orderId=document.orderFindForm.orderId.value;
      urlPara.userLogon=document.orderFindForm.userLogon.value;
      urlPara.accountId=document.orderFindForm.accountId.value;
      urlPara.orderType=document.orderFindForm.orderState.value;
      urlPara.orderby='orderid';

      top.setContent("<%= UIUtil.toJavaScript((String)orderLabels.get
                   ("findResultBCT")) %>",url,true, urlPara);
      return true;
   }
   return false;
}
function cancelAction() {
   top.goBack();
}
// -->
</SCRIPT>
</HEAD>
<BODY CLASS=content ONLOAD="initializeState();">
<H1><%=orderLabels.get("findDialog")%></H1>
<P><%=orderLabels.get("findCSOrderInst")%>
<FORM NAME="orderFindForm">
<TABLE>
   <TBODY>
   <TR>
      <TD><%=orderLabels.get("orderNumber")%></TD>
   </TR>
   <TR>
      <TD><INPUT size="9" type="text" maxlength="9" name="orderId"></TD>
   </TR>
   <TR>
      <TD></TD>
   </TR>
   <TR>
      <TD><%=orderLabels.get("customerName")%></TD>
   </TR>
   <TR>
      <TD><INPUT size="31" type="text" maxlength="31" name="userLogon"
                 value="<%=getUserLogon(customerId, request)%>"></TD>
   </TR>
   <TR>
       <TD></TD>
   </TR>
   <TR>
      <TD><%= orderLabels.get("orderStatus") %></TD>
   </TR>
   <TR>
      <TD>
         <SELECT name="orderState">
            <OPTION value="all"></OPTION>
            <OPTION value="P"><%= orderLabels.get("P") %></OPTION>
            <OPTION value="I"><%= orderLabels.get("I") %></OPTION>
            <OPTION value="W"><%= orderLabels.get("W") %></OPTION>
            <OPTION value="N"><%= orderLabels.get("N") %></OPTION>
            <OPTION value="M"><%= orderLabels.get("M") %></OPTION>
            <OPTION value="B"><%= orderLabels.get("B") %></OPTION>
```

```
            <OPTION value="C"><%= orderLabels.get("C") %></OPTION>
            <OPTION value="E"><%= orderLabels.get("E") %></OPTION>
            <OPTION value="R"><%= orderLabels.get("R") %></OPTION>
            <OPTION value="S"><%= orderLabels.get("S") %></OPTION>
            <OPTION value="D"><%= orderLabels.get("D") %></OPTION>
            <OPTION value="L"><%= orderLabels.get("L") %></OPTION>
            <OPTION value="T"><%= orderLabels.get("T") %></OPTION>
            <OPTION value="A"><%= orderLabels.get("A") %></OPTION>
            <OPTION value="F"><%= orderLabels.get("F") %></OPTION>
            <OPTION value="G"><%= orderLabels.get("G") %></OPTION>
            <OPTION value="X"><%= orderLabels.get("X") %></OPTION>
         </SELECT>
      </TD>
   </TR>
   <TR>
      <TD></TD>
   </TR>
   <TR>
      <TD><%= orderLabels.get("accountName") %></TD>
   </TR>
   <TR>
      <TD>
         <SELECT name="accountId">
            <OPTION value=""></OPTION>
            <%   if (acctList != null) {
                    for (int i=0; i<acctList.length; i++) { %>
                    <OPTION value="<%=acctList[i].getAccountId()%>">
                          <%=acctList[i].getAccountName()%></OPTION>
            <%      }
                 } %>
         </SELECT>
      </TD>
   </TR>
   <TR>
      <TD></TD>
   </TR>
   </TBODY>
</TABLE>
</FORM>
<SCRIPT LANGUAGE="JavaScript">
<!--
//Para IE si (document.all) {
   onLoad();
}
//-->
</SCRIPT>

</BODY>
</HTML>
```

Capítulo 3. Espacios de trabajo colaborativos

Los elementos descritos en este capítulo presentan un estudio de caso que detalla los pasos necesarios para personalizar un QuickPlace y crear la plantilla de ToolTech que se envía con WebSphere Commerce. Dicho estudio también describe los pasos necesarios para personalizar el contenido del correo electrónico enviado cuando se ha invitado a un usuario a unirse a un espacio de trabajo colaborativo.

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar esta parte de WebSphere Commerce Accelerator.

Nota: Los escenarios 1 y 2 están estrechamente relacionados. Si ejecuta el escenario 1, lo más probable es que también desee ejecutar el escenario 2.

Escenario 1: Personalizar el aspecto del espacio de trabajo

Visión general de la implementación

Puede que lo primero que desee personalizar para la característica de Espacios de trabajo colaborativos es el ″aspecto″ de los espacios de trabajo colaborativos creados. Para personalizar el aspecto, deberá crear una plantilla denominada PlaceType. Para obtener más detalles sobre la personalización, consulte el manual técnico *Customizing QuickPlace*.

Pasos para la personalización

Este ejemplo utiliza la tienda de ejemplo ToolTech que se envía en WebSphere Commerce 5.4. Este ejemplo ilustra cómo crear el QuickPlace:

1. Cree un QuickPlace por omisión realizando lo siguiente:
   a. Asegúrese de que el servidor QuickPlace esté en ejecución. Abra el navegador en `http://servidor_qp/quickplace` y conéctese como administrador de QuickPlace.
   b. Pulse **Crear un QuickPlace**.
   c. Elija **QuickPlace estándar para equipos**. Rellene los campos del modo que sea aplicable y pulse **SIGUIENTE**.
   d. Conéctese como creador.
2. Actualice la página de Bienvenida para QuickPlace. Pulse **Editar**. Realice los cambios deseados y pulse **PUBLICAR**. Esto actualiza la página de Bienvenida.
3. Cambie el logotipo para el QuickPlace realizando lo siguiente:
   a. Pulse **Personalizar** en la tabla de contenido.
   b. Pulse **Información básica**.
   c. Pulse **Cambiar información básica**.
   d. Configure el logotipo utilizando **Texto simple**, **Creador de logotipos** o **Subir imagen de logotipo**. El ejemplo de ToolTech utiliza **Subir imagen de logotipo**. El logotipo ya está actualizado.

Escenario 2: Crear un tema personalizado

Visión general de la implementación

Un tema controla el ″aspecto″ de un QuickPlace. Este ejemplo crea una página HTML (Hypertext Markup Language - Lenguaje de marcación de hipertexto) y una hoja de estilo en cascada (CSS) para el diseño y estilo del QuickPlace. Para obtener más información sobre cómo escribir esos archivos, consulte el manual técnico *Customizing QuickPlace*.

Nota:

Dado que la administración de miembros se gestiona mediante WebSphere Commerce, elimine el enlace **Miembros** de la tabla de contenido. Para eliminar el enlace, utilice el código siguiente en el archivo de Diseño de página al insertar la tabla de contenido:

```
<QuickPlaceSkinComponent
   name=TOC
  Format={
  <tr>
  <td class=h-toc-text><br></td>
  <td class=h-toc-text><Item class=h-toc-text></td>
  <td class=h-toc-text><br></td>
  </tr>
  <tr>
  <td colspan=3><img src="../../../$resources.nsf/h_ResourcesByName/
                transparent.gif/$FILE/transparent.gif?OpenElement"
                width=1 height=3 alt="" border="0"></td>
  }
  SelectedFormat={
  <tr>
  <td class=h-tocSelected-text><br></td>
  <td class=h-tocSelected-text><Item class=h-tocSelected-text></td>
  <td class=h-tocSelected-text><br></td>
  </tr>
  <tr><td colspan=3><img src="../../../$resources.nsf/h_ResourcesByName/
                transparent.gif/$FILE/transparent.gif?OpenElement"
                width=1 height=3 alt="" border="0">
  </td>
  }
  EmptyFormat={}
  ReplaceString={Members=}>
```

Pasos para la personalización

1. Pulse **Personalizar** en la tabla de contenido, **Personalizar temas** y, a continuación, **Tema nuevo**. Rellene el título y suba los archivos de **Hoja de estilo** y **Diseño de página**.
2. Pulse **Siguiente**.
3. Pulse **Personalizar** en la tabla de contenido y, a continuación, **Decorar**
4. Pulse **Elegir un tema**.
5. Seleccione el tema que acaba de crear y pulse **Siguiente**. Esto actualizará el aspecto del QuickPlace.
6. Observe que el enlace **Miembros** de la tabla de contenido ha desaparecido.
7. Crear un PlaceType a partir de un QuickPlace. Ahora puede crear un PlaceType basándose en el QuickPlace que acaba de personalizar. Pulse **Personalizar** en la tabla de contenido y, a continuación, **Opciones de PlaceType**.

8. Pulse **Editar**. Elija **Sí** para **¿Permitir que se creen PlaceTypes a partir de este QuickPlace?** y **No** para **¿Incluir miembros actuales de este QuickPlace en QuickPlaces futuros creados a partir del PlaceType?**
9. Pulse **Siguiente**.
10. Abra el navegador en `http://servidor_qp/quickplace` y conéctese como administrador de QuickPlace.
11. Pulse **PlaceTypes**.
12. Pulse **CREAR PLACETYPE...**. Entre el nombre del PlaceType y elija el QuickPlace en el que desea que se base.
13. Pulse **Siguiente**. Verá el PlaceType nuevo.

Escenario 3: Personalizar la notificación de correo electrónico

Visión general de la implementación

Para cambiar la notificación de correo electrónico por omisión por una página personalizada para todas las tiendas, necesita modificar el archivo `CollabEmailContent.jsp` para que incluya el contenido deseado.

Pasos para la personalización

1. Modifique el archivo `CollabEmailContent.jsp`. Localice el archivo en el directorio siguiente:

   AIX

   `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcstores.war`

   400

   `/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/wcstores.war`

   Linux

   `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcstores.war`

   Solaris

   `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcstores.war`

   Windows

   unidad`:\WebSphere\AppServerinstalledApps\WC_Enterprise_App_demo.ear\wcstores.war`

2. Personalice el correo electrónico para una tienda específica realizando lo siguiente:
   a. Abra una ventana de mandatos de DB2.
   b. Conéctese a la base de datos de WebSphere Commerce. Para conectarse a la base de datos por omisión, utilice el mandato siguiente:

      ```
      db2 connect to mall
      ```

   c. Ejecute el mandato de DB2 siguiente:

      ```
      db2 update viewreg set properties = 'docname=CollabEmailContent.jsp'
          where viewname = 'CollabEmailContentView'
      ```

   d. Ponga el archivo de plantilla personalizado `CollabEmailContent.jsp` en el directorio de tienda:

      AIX

      `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcstores.war/`*nombre_tienda*

      400

/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/wcstores.war/*nombre_tienda*

Linux

/opt/WebSphere/AppServerinstalledApps/WC_Enterprise_App_demo.ear/wcstores.war/*nombre_tienda*

Solaris

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcstores.war/*nombre_tienda*

Windows

unidad:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\wcstores.war*nombre_tienda*

donde *nombre_tienda* es el nombre de directorio de tienda que ha elegido al publicar la tienda.

e. Reinicie el Servidor de administración de WebSphere.

Capítulo 4. Perfiles de cliente

Los elementos descritos en este capítulo representan la interfaz de usuario para los componentes Perfil de cliente de WebSphere Commerce Accelerator. Facilitan la creación y el mantenimiento de los perfiles de cliente y todas las acciones que un Vendedor o Comerciante necesita para realizar las tareas diarias asociadas con el negocio. El componente Perfiles de cliente incluye los elementos siguientes:

- Asistente de perfil de cliente
- Cuaderno Perfil de cliente

Código de ejemplo

Esta sección hace referencia a ejemplos de código contenidos en el archivo zip ubicado en el URL siguiente:

`ftp://ftp.software.ibm.com/software/websphere/commerce/54/profcust.zip`

Para completar algunos apartados de este capítulo, deberá bajar e instalar este código de ejemplo en una máquina de desarrollo que ejecute WebSphere Application Server 4.

Una vez bajado, desempaquete este archivo ZIP en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer`

400 `/QIBM/UserData/WebSphere/CommerceServer`

Linux `/opt/WebSphere/CommerceServer`

Solaris `/opt/WebSphere/CommerceServer`

Windows *unidad*`:\WebSphere\CommerceServer`

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar esta parte de WebSphere Commerce Accelerator.

Escenario 1: Añadir un atributo al perfil de cliente

Visión general de la implementación

En este ejemplo, la columna `FIELD1` de la tabla `USERS` almacena las preferencias de bebida del usuario. Serán válidos los valores siguientes:

- cerveza
- vino
- café
- té
- agua

Un ejemplo muestra cada uno de estos pasos. Localice este ejemplo en el directorio `profcust` creado al instalar el código de ejemplo.

Pasos para la personalización

1. Añada una página nueva en el cuaderno de perfil de cliente creando un archivo JSP nuevo. En el ejemplo, esta página visualizará dos opciones:

- Ignorar preferencia de bebida
- Clientes de destino con una o varias de las preferencias de bebida siguientes:

Las preferencias aparecerán como recuadros de selección bajo la segunda opción. Examine los detalles adicionales en la página JSP que lo acompaña, `BeveragePanel.jsp`, en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/profcust/web/tools/segmentation/`

400 `/QIBM/UserData/WebSphere/CommerceServer/profcust/web/tools/segmentation/`

Linux `/opt/WebSphere/CommerceServer/profcust/web/tools/segmentation/`

Solaris `/opt/WebSphere/CommerceServer/profcust/web/tools/segmentation/`

Windows *unidad*`:\WebSphere\CommerceServer\profcust\web\tools\segmentation\`

2. Después de crear el archivo JSP, deberá añadirlo a la tabla `VIEWREG`. Examine los detalles adicionales en el script SQL que lo acompaña, `beverage.sql`, en el directorio siguiente:

   AIX `/usr/WebSphere/CommerceServer/profcust/schema/`

   400 `/QIBM/UserData/WebSphere/CommerceServer/profcust/schema/`

   Linux `/opt/WebSphere/CommerceServer/profcust/schema/`

   Solaris `/opt/WebSphere/CommerceServer/profcust/schema/`

   Windows *unidad*`:\WebSphere\CommerceServer\profcust\schema\`

3. El cuaderno Perfil de cliente debe reconocer la página nueva que se encuentra en el directorio siguiente:

   AIX `/usr/WebSphere/CommerceServer/xml/tools/segmentation/`

   400 `/QIBM/UserData/WebSphere/CommerceServer/xml/tools/segmentation/`

   Linux `/opt/WebSphere/CommerceServer/xml/tools/segmentation/`

   Solaris `/opt/WebSphere/CommerceServer/xml/tools/segmentation/`

   Windows *unidad*`:\WebSphere\CommerceServer\xml\tools\segmentation\` Haga una copia de `SegmentNotebook.xml`. En el ejemplo, el nombre de este nuevo archivo es `MySegmentNotebook.xml`. Se ha añadido al documento el elemento siguiente:

   ```
   <panel name="segmentNotebookBeveragePanel"
          url="SegmentNotebookBeveragePanelView"
          group="segmentNotebookMiscPanelGroup" />
   ```

   Debe incluir el nuevo nombre de panel en el archivo de propiedades de segmentación, ubicado en el directorio siguiente:

   AIX

   `/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/properties/com/ibm/commerce/tools/segmentation`

   400

   `/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/properties/com/ibm/commerce/tools/segmentation`

   Linux

   `/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/`

properties/com/ibm/commerce/tools/segmentation/

Solaris

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/properties/com/ibm/commerce/tools/segmentation/

Windows

unidad:\WebSphere\AppServer\installedApps/WC_Enterprise_App_demo.ear\properties\com\ibm\commerce\tools\segmentation\

Para que esto funcione, necesitará añadir la línea siguiente en Resources.properties:

```
segmentNotebookBeveragePanel=Bebida preferida
```

Opcionalmente, copie MySegmentNotebook.xml de:

AIX

/usr/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MySegmentNotebook.xml

400 /QIBM/UserData/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MySegmentNotebook.xml

Linux /opt/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MySegmentNotebook.xml

Solaris /opt/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MySegmentNotebook.xml

Windows

unidad:\WebSphere\CommerceServer\profcust\xml\tools\segmentation\MySegmentNotebook.xml

en:

AIX /usr/WebSphere/CommerceServer/xml/tools/segmentation/MySegmentNotebook.xml

400

/QIBM/UserData/WebSphere/CommerceServer/xml/tools/segmentation/MySegmentNotebook.xml

Linux /opt/WebSphere/CommerceServer/xml/tools/segmentation/MySegmentNotebook.xml

Solaris /opt/WebSphere/CommerceServer/xml/tools/segmentation/MySegmentNotebook.xml

Windows *unidad*:\WebSphere\CommerceServer\xml\tools\segmentation\MySegmentNotebook.xml

4. Ahora necesita asegurarse de que se reconozca el nuevo documento XML. Para ello, modifique el archivo Resources.xml del directorio siguiente:

   AIX /usr/WebSphere/CommerceServer/xml/tools/segmentation/

   400

   /QIBM/UserData/WebSphere/CommerceServer/xml/tools/segmentation/

   Linux /opt/WebSphere/CommerceServer/xml/tools/segmentation/

   Solaris /opt/WebSphere/CommerceServer/xml/tools/segmentation/

   Windows *unidad*:\WebSphere\CommerceServer\xml\tools\segmentation\

   Modifique el elemento siguiente:

   ```
   <XML name="SegmentNotebook"
        file="segmentation/SegmentNotebook.xml" />
   ```

   Cambie el elemento por:

```
<XML name="SegmentNotebook"
     file="segmentation/MySegmentNotebook.xml" />
```

Guarde el archivo como MyResources.xml

Opcionalmente, copie MyResources.xml de:

AIX

/usr/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MyResources.xml

400 /QIBM/UserData/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MyResources.xml

Linux /opt/WebSphere/CommerceServer/profcust/xml/tools/segmentation/ MyResources.xml

Solaris /opt/WebSphere/CommerceServer/profcust/xml/tools/segmentation/MyResources.xml

Windows

unidad:\WebSphere\CommerceServer\profcust\xml\tools\segmentation\MyResources.xml

en:

AIX /usr/WebSphere/CommerceServer/xml/tools/segmentation

400 /QIBM/UserData/WebSphere/CommerceServer/xml/tools/segmentation

Linux /opt/WebSphere/CommerceServer/xml/tools/segmentation

Solaris /opt/WebSphere/CommerceServer/xml/tools/segmentation

Windows *unidad*:\WebSphere\CommerceServer\xml\tools\segmentation

5. Apunte la aplicación a MyResources.xml. Modifique el archivo demo.xml del directorio siguiente:

   AIX /usr/WebSphere/CommerceServer/instances/demo/xml/

   400 /QIBM/UserData/WebSphere/CommerceServer/instances/demo/xml/

   Linux /opt/WebSphere/CommerceServer/instances/demo/xml/

   Solaris /opt/WebSphere/CommerceServer/instances/demo/xml/

   Windows *unidad*:\WebSphere\CommerceServer\instances\demo\xml\

   Busque:

   ```
   <resourceConfig file="segmentation/Resources.xml" />
   ```

   Cambie el elemento por:

   ```
   <resourceConfig file="segmentation/MyResources.xml" />
   ```

6. Amplíe el bean de datos que describe perfiles de cliente, com.ibm.commerce.tools.segmentation.SegmentNotebookDataBean. En el ejemplo, com.mycompany.tools.segmentation.MySegmentNotebookDataBean amplía el bean de datos para proporcionar propiedades para el atributo de bebida. Localice este ejemplo, MySegmentNotebookDataBean.java, en el directorio siguiente:

   AIX

   /usr/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

   400

   /QIBM/UserData/WebSphere/CommerceServer/profcust/lib/com/mycompany/

tools/segmentation/

Linux

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Solaris

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Windows

unidad:\WebSphere\CommerceServer\profcust\lib\com\mycompany\tools\segmentation\

7. Para que el cuaderno de segmento reconozca el nuevo bean de datos, necesitará modificar el XML que describe el cuaderno Perfil de cliente. Vuelva a MySegmentNotebook.xml y observe el cambio siguiente:

```
<databean name="segmentDetails"
    class="com.mycompany.tools.segmentation.SegmentNotebookDataBean" />
```

se ha cambiado por:

```
<databean name="segmentDetails"
    class="com.mycompany.tools.segmentation.MySegmentNotebookDataBean"
    stoplevel="2" />
```

8. Amplíe el mandato de controlador que guarda el perfil de cliente. Este mandato es com.ibm.commerce.tools.segmentation.SegmentSaveControllerCmd. Examine la implementación de mandato de ejemplo en MySegmentSaveControllerCmdImpl en el directorio siguiente:

AIX

/usr/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

400

/QIBM/UserData/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Linux

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Solaris

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Windows

unidad:\WebSphere\CommerceServer\profcust\lib\com\mycompany\tools\segmentation\

Este ejemplo creará una condición simple para la condición de bebida.

Para que la aplicación reconozca este nuevo mandato, será necesario añadirlo a la tabla CMDREG. Consulte beverage.sql en el paso 1a anterior para obtener más detalles.

9. Amplíe el mandato de tarea que evalúa el perfil de cliente, que es com.ibm.commerce.membergroup.commands.CheckUserInMemberGroupCmd. Examine los detalles sobre cómo evaluar una nueva condición en el mandato de ejemplo, MyCheckUserInMemberGroupCmdImpl.java, en el directorio siguiente:

AIX

/usr/WebSphere/CommerceServer/profcust/lib/com/mycompany/membergroup/commands/

400

/QIBM/UserData/WebSphere/CommerceServer/profcust/lib/com/mycompany/membergroup/commands/

Linux

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/membergroup/commands/

Solaris

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/membergroup/commands/

Windows

unidad:\WebSphere\CommerceServer\profcust\lib\com\mycompany\membergroup\commands\

También será necesario que registre el mandato en la tabla CMDREG. Consulte beverage.sql para obtener más detalles.

10. Amplíe el mandato de tarea que visualiza la lista de restricciones para un perfil de cliente. Este mandato es com.ibm.commerce.tools.segmentation.SegmentConstraintListCmd. Examine en el mandato de ejemplo, MySegmentConstraintListCmdImpl, del directorio siguiente los detalles sobre lo que debe cambiar:

AIX

/usr/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

400

/QIBM/UserData/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Linux

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Solaris

/opt/WebSphere/CommerceServer/profcust/lib/com/mycompany/tools/segmentation/

Windows

unidad:\WebSphere\CommerceServer\profcust\lib\com\mycompany\tools\segmentation\

También será necesario que registre el mandato en la tabla CMDREG. Consulte bevarage.sql para obtener más detalles.

11. Añada las clases siguientes

- com.mycompany.membergroup.commands. MyCheckUserInMemberGroupCmdImpl
- com.mycompany.tools.segmentation.MySegmentConstraintListCmdImpl
- com.mycompany.tools.segmentation.MySegmentNotebookDataBean
- com.mycompany.tools.segmentation.MySegmentSaveControllerCmdImpl

ubicadas en el directorio siguiente:

AIX /usr/WebSphere/CommerceServer/profcust/lib

400 /QIBM/UserData/WebSphere/CommerceServer/profcust/lib

Linux /opt/WebSphere/CommerceServer/profcust/lib

Solaris /opt/WebSphere/CommerceServer/profcust/lib

Windows *unidad*:\WebSphere\CommerceServer\profcust\lib

al archivo wcsmcruntime.jar ubicado en el directorio siguiente:

AIX

/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation

400

/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation

Linux

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation/

Solaris

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation/

Windows

unidad:\WebSphere\AppServer\installedApps/WC_Enterprise_App_demo.ear\
properties\com\ibm\commerce\tools\segmentation\

12. Copie el archivo BeveragePanel.jsp del paso 1 en el directorio siguiente:

AIX

/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation

400

/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation

Linux

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation/

Solaris

/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
properties/com/ibm/commerce/tools/segmentation/

Windows

unidad:\WebSphere\AppServer\installedApps/WC_Enterprise_App_demo.ear\
properties\com\ibm\commerce\tools\segmentation\

13. Estos cambios requieren que se realicen cambios en los valores de control de acceso, lo cual está fuera del ámbito de esta guía. Consulte la publicación *WebSphere Commerce, Guía de control de acceso* disponible en el URL siguiente:
www.ibm.com/software/webservers/commerce/wc_be/lit-tech-general.html

Capítulo 5. Campañas

Entrada directa de normas en la base de datos

Aunque la interfaz de usuario es el método más fácil mediante el cual se entran datos de normas en la base de datos, puede que algunas situaciones requieran que se entren manualmente las normas en la columna RULE de la tabla INITIATIVE. Las normas se definen utilizando XML y, en este caso, se deberá seguir la definición de tipo de documento (DTD) que se presenta a continuación:

La DTD para la columna RULE de la tabla INITIATIVE es la siguiente:

```
<!DOCTYPE rule [
   <!ELEMENT rule (comment?, (orListCondition |andListCondition |simpleCondition |trueCondition |openCondition), action)>

   <!ELEMENT comment EMPTY>
   <!ATTLIST comment text CDATA #REQUIRED>

   <!ELEMENT action (parameter*)>
   <!ATTLIST action name CDATA #REQUIRED>

   <!ELEMENT orListCondition (not?, (orListCondition |andListCondition | simpleCondition | trueCondition |openCondition)+)>

   <!ELEMENT andListCondition (not?, (orListCondition |andListCondition | simpleCondition | trueCondition |openCondition)+)>

   <!ELEMENT simpleCondition (not?, variable, operator, value, qualifier*)>

   <!ELEMENT openCondition (not?, parameter*)>
   <!ATTLIST openCondition name CDATA #REQUIRED>

   <!ELEMENT trueCondition (not?)>

   <!ELEMENT not EMPTY>

   <!ELEMENT variable EMPTY>
   <!ATTLIST variable name CDATA #REQUIRED>

   <!ELEMENT operator EMPTY>
   <!ATTLIST operator name CDATA #REQUIRED>

   <!ELEMENT value EMPTY>
   <!ATTLIST value data CDATA #REQUIRED>

   <!ELEMENT qualifier EMPTY>
   <!ATTLIST qualifier name CDATA #REQUIRED>
   <!ATTLIST qualifier data CDATA #REQUIRED>

   <!ELEMENT parameter (parameter*)>
   <!ATTLIST parameter name CDATA #REQUIRED>
   <!ATTLIST parameter value CDATA #REQUIRED>
]>
```

Elementos simpleCondition

La implementación por omisión de iniciativas soporta los elementos simpleCondition siguientes. Puede utilizar estos elementos en cualquier combinación con los elementos orListCondition, andListCondition y openCondition para crear la parte de condición de una norma.

Tabla 2.

Nombre de variable	Operadores soportados	Valores soportados	Calificadores
segment	= !=	El nombre de un perfil de cliente.	ninguno
shoppingCartSku	= !=	Un código de artículo de producto.	ninguno
shoppingCartCategory	= !=	Un nombre de categoría de producto.	language
purchaseHistorySku	= !=	Un código de artículo de producto.	ninguno

Tabla 2. (continuación)

Nombre de variable	Operadores soportados	Valores soportados	Calificadores
purchaseHistoryCategory	= !=	Un nombre de categoría de producto.	language
shoppingCartTotal	= != > < >= <=	Un valor decimal.	currency
dayOfWeek	= !=	DOMINGO LUNES MARTES MIÉRCOLES JUEVES VIERNES SÁBADO	ninguno

Elementos openCondition

La implementación por omisión soporta los elementos openCondition siguientes. Puede utilizar estos elementos en cualquier combinación con los elementos orListCondition, andListCondition y simpleCondition para crear la parte de condición de una norma.

Tabla 3.

Nombre de condición abierta	Parámetros soportados
shoppingCartTotal	comparisonType - puede ser uno de ″>″, ″<″ o ″=″. valor - una moneda en valor decimal
shoppingCartCategory	comparisonType - puede ser uno de ″=″ o ″!=″ valor - nombre de la categoría language
shoppingCartSku	comparisonType - puede ser uno de ″>″, ″<″ o ″=″ valor - una moneda en valor decimal
purchaseHistoryCategory	comparisonType - puede ser uno de ″=″ o ″!=″ valor - nombre de una categoría currency
purchaseHistorySku	comparisonType - puede ser uno de ″>″, ″<″ o ″=″ valor - una moneda en valor decimal

Elementos de acción

La implementación por omisión soporta los elementos de acción siguientes. Estos pueden estar con cualquier elemento de condición para crear una norma.

Tabla 4.

Nombre de acción	Parámetros soportados	Valores soportados	Subparámetros soportados
suggestiveSell	queryOperator	or and	ninguno
	queryType	skuQuery genericQuery	ninguno
	sku	Un código de artículo de producto	ninguno
	skuPrefix	un prefijo de código de artículo de producto	ninguno
	category	Un nombre de categoría de producto	language
	productDescription	Una descripción de categoría de producto	language
	inventoryQuantity	Un entero	comparisonOperator
	listPrice	Un calor decimal	comparisonOperator currency
	availabilityDate	Una fecha con el formato aaaa-mm-dd-00.00.00	comparisonOperator
collaborativeFiltering	ninguno		
awarenessAd	collateral	Un nombre de anuncio	
discountAd	discountCollateral	Un nombre de anuncio	Discount identifier
couponAd	couponCollateral	Un nombre de anuncio	
category Recommendation	category	Nombre de una categoría	

La iniciativa de ejemplo siguiente muestra un anuncio de primavera para caballeros y un anuncio de otoño para señoras:

```
<ruleSet>
   <rule>
      <comment text="mostrar anuncio de primavera a caballeros"/>
      <simpleCondition>
         <variable name="segment"/>
         <operator name="="/>
         <value data="caballeros"/>
      </simpleCondition>
      <action name="awarenessAd">
         <parameter name="collateral" value="Primavera">
         </parameter>
      </action>
   </rule>
   <rule>
      <comment text="mostrar anuncio de otoño a señoras"/>
      <simpleCondition>
         <variable name="segment"/>
```

```
        <operator name="="/>
        <value data="señoras"/>
      </simpleCondition>
      <action name="awarenessAd">
        <parameter name="collateral" value="Otoño">
        </parameter>
      </action>
   </rule>
</ruleSet>
```

Elementos de infraestructura

Paquete de normas

El componente Campañas utiliza el paquete `com.ibm.commerce.rule`. Este paquete proporciona clases que le ayudan a realizar la conversión de un paquete de normas al formato XML y de dicho formato. También proporciona soporte para llamar a las normas del conjunto de normas.

Tabla 5.

Nombre de clase/interfaz	Descripción
Action	La clase Action proporciona propiedades para el nombre y los parámetros utilizados por la parte de acción de una norma. Los parámetros tienen nombres y valores y opcionalmente también pueden tener subparámetros.
ActionHandler	La interfaz ActionHandler debe implementarse a fin de proporcionar la implementación de la parte de acción de una norma. Existe un método ″performAction″ que se llama cuando la parte de condición de la norma se evalúa como verdadera. La instancia apropiada de la clase Action se pasará a la implementación del método ″performAction″.
Rule	La clase Rule proporciona métodos que le ayudan a realizar la conversión de una instancia de Rule al formato XML y de dicho formato. La clase Rule tiene tres propiedades. Una para la parte de condición de la norma. Otra para la parte de acción de la norma. Y la tercera para el comentario. La propiedad de condición debe ser una instancia de la clase Condition que se pueden encontrar en el paquete com.ibm.commerce.condition. La clase Rule también tiene un método ″invoke″ que acepta una implementación de la interfaz Evaluator y la interfaz ActionHandler. Si la condición se evalúa como verdadera, entonces se invoca el método ″performAction″ de la implementación de ActionHandler.
RuleConstants	La interfaz RuleConstants contiene algunas constantes públicas que el paquete de normas utiliza.

Tabla 5. (continuación)

Nombre de clase/interfaz	Descripción
RuleSet	La clase RuleSet proporciona métodos que le ayudan a realizar la conversión de una instancia de RuleSet al formato XML y de dicho formato. Tiene una propiedad que es una matriz de los objetos Norma. También tiene una método de invocación que acepta una implementación de las interfaces Evaluator y ActionHandler. Cuando se llama al método de invocación, éste realiza un bucle en la matriz de los objetos Norma y los invoca uno tras otro.

Las campañas utilizan el paquete de normas para guardar y cargar la parte de normas de la iniciativa de campaña en formato XML y a partir de dicho formato. También lo utilizan para invocar las normas. El mandato de tarea `CampaignInitiativeEvaluateCmd` proporciona la implementación de la interfaz `ActionHandler` que se utiliza para manejar las acciones de iniciativa de campaña.

Paquete de condiciones

El paquete de normas utiliza el paquete `com.ibm.commerce.condition`. Este paquete proporciona clases que le ayudan a realizar la conversión de una condición booleana al formato XML y de dicho formato. También proporciona soporte para evaluar una condición.

Tabla 6.

Nombre de clases/interfaz	Descripción
AndListCondition	La clase AndListCondition amplía la clase ConditionList para proporcionar una condición que consta de una lista de condiciones unidas por el operador booleano AND.
Condition	La clase Condition es una clase abstracta que proporciona métodos que pueden ayudarle a realizar la conversión al formato XML y de dicho formato. También proporciona un método abstracto que puede utilizarse para evaluar la condición.
ConditionConstants	La interfaz ConditionConstants proporciona valores de constantes que el paquete de condiciones utiliza.
ConditionList	La clase ConditionList es una clase abstracta que amplía la clase Condition. Proporciona soporte para una condición que conste de una lista de condiciones.
ConditionUtil	La clase ConditionUtil proporciona diversos métodos estáticos que se pueden utilizar durante la evaluación de condición.
Evaluator	La interfaz Evaluator debe implementarse a fin de evaluar una condición simple o una condición abierta.

Tabla 6. (continuación)

Nombre de clases/interfaz	Descripción
OpenCondition	La clase OpenCondition amplía la clase Condition para proporcionar una condición genérica. Esta condición consta de una lista de parejas de nombre y valor.
OrListCondition	La clase OrListCondition amplía la clase ConditionList para proporcionar una condición que consta de una lista de condiciones unidas por el operador booleano OR.
SimpleCondition	La clase SimpleCondition amplía la clase Condition para proporcionar una condición en formato de Variable, operador, valor.
TrueCondition	La clase TrueCondition amplía la clase Condition para proporcionar una condición que se evalúa siempre como verdadera.

El paquete de normas utiliza el paquete de condiciones para guardar y cargar la parte de condición de las normas en formato XML y a partir de dicho formato. También ayuda a evaluar las condiciones.

Existen cinco extensiones concretas de la clase abstracta Condition. Tres de ellas (AndListCondition, OrListCondition y TrueCondition) saben cómo evaluarse a sí mismas. Las dos restantes (OpenCondition y SimpleCondition) necesitan una implementación de la interfaz Evaluator para poderse evaluar. El mandato de tarea CampaignInitiativeEvaluateCmd proporciona las implementaciones de la interfaz Evaluator que se utiliza para evaluar condiciones de iniciativa de campaña.

Proyecto de normas Blaze

La implementación por omisión de las iniciativas de campaña utiliza un proyecto de normas Blaze que ayuda a evaluar las normas de iniciativa de campaña. Por omisión, todas las condiciones abiertas hacen que el mandato de tarea **CampaignInitiativeEvaluateCmd** invoque el proyecto de normas Blaze. Localice el proyecto, `CampaignInitiativeEvaluator`, en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/rules/campaigns`

400 `/QIBM/UserData/WebSphere/CommerceServer/rules/campaigns`

Linux `/opt/WebSphere/CommerceServer/rules/campaigns`

Solaris `/opt/WebSphere/CommerceServer/rules/campaigns`

Windows *unidad*`:\WebSphere\CommerceServer\rules\campaigns`

La implementación por omisión de este proyecto es capaz de evaluar todas las condiciones abiertas creadas por la interfaz de usuario de iniciativa de campaña. Puede añadir soporte para condiciones abiertas adicionales en este proyecto.

Personalización

Es de esperar que nuestros clientes desearán ampliar la implementación por omisión de las iniciativas de campaña para añadir más condiciones soportadas y posiblemente acciones y tipos de visualización adicionales.

Adición de soporte de ejecución para condiciones de iniciativa nuevas

El proyecto blaze que evalúa las condiciones de iniciativa de campaña acepta una instancia de **CampaignInitiativeContext**. Este contexto proporciona acceso al contexto de mandato así como al nombre y a los

parámetros de la condición abierta. Si desea añadir soporte para una condición abierta nueva, necesitará añadir una norma nueva en la norma `conditionEvaluator`. Esta norma necesitará comprobar el nombre de la condición nueva y evaluar la condición. Durante la evaluación de la condición, puede acceder a los parámetros de condición abierta por el nombre, a través del objeto de contexto.

Adición de soporte de creación para condiciones de iniciativa nuevas

Para permitir que el director de marketing aproveche las condiciones nuevas, la interfaz de usuario de iniciativa de campaña necesitará ampliarse para proporcionar soporte para la nueva condición. Esto puede realizarse añadiendo una condición nueva al panel Cuál del asistente o cuaderno de iniciativa de campaña (`WhenAddPanel.jsp`). También necesitará ampliar la clase **CampaignInitiativeSaveControllerCmdImpl** para crear el nuevo objeto Condición. Localice el archivo en el directorio siguiente:

AIX

```
/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

400

```
/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

Linux

```
/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

Solaris

```
/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

Windows

```
unidad:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\
wctools.war\tools\campaigns\
```

Adición de soporte de ejecución para acciones de iniciativa de campaña nuevas

Si desea proporcionar soporte para acciones de iniciativa de campaña nuevas, necesitará ampliar la clase **CampaignInitiativeEvaluateCmdImpl** y alterar temporalmente el método **performAction**. Si el nombre de la acción coincide con el nombre de acción nuevo, deberá realizar la acción o, de lo contrario, deberá llamar al supermétodo para realizar la implementación por omisión.

Adición de soporte de creación para acciones de iniciativa de campaña nuevas

Para permitir que el directorio de marketing aproveche las acciones de iniciativa de campaña nuevas, necesitará actualizar el panel Cuál del asistente o cuaderno de iniciativa de campaña (`InitiativeWhatPanel.jsp`). También necesitará ampliar la clase **CampaignInitiativeSaveControllerCmdImpl** para crear el nuevo objeto Acción. Localice el archivo en el directorio siguiente:

AIX

```
/usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

400

```
/QIBM/ProdData/WebAsAdv4/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

Linux

```
/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

Solaris

```
/opt/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/
wctools.war/tools/campaigns/
```

Windows

```
unidad:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\
wctools.war\tools\campaigns\
```

Adición de soporte para tipos nuevos de zona de e-Marketing

Si cree que necesita proporcionar tipos de salida adicionales, puede ampliar la clase `EMarketingSpot` para proporcionar una propiedad nueva que el diseñador de páginas puede utilizar. También necesitará ampliar la clase **CampaignInitiativeEvaluateCmdImpl** para volver al nuevo tipo de salida.

Capítulo 6. Búsqueda de catálogo

Los elementos descritos en este capítulo describen las actividades que son necesarias para ampliar la funcionalidad existente de la búsqueda. La funcionalidad adicional incluye asesoramiento para entrar nuevos atributos y tablas a los que se pueda acceder mediante el componente de búsqueda. La optimización del esquema para ayudar a reducir el coste en la ejecución también puede mejorar el rendimiento de los componentes de búsqueda. Este componente incluye los siguientes elementos personalizables:

- Bean de búsqueda
- Motor de búsqueda
- Scripts de definición de tabla de resumen

La Tabla 8 resume la lista de columnas de base de datos en las que busca el bean de búsqueda de catálogo. Debido a la diversidad de los requisitos de sitio, probablemente encontrará necesario proporcionar búsqueda en una columna que no se describe en esta tabla.

Escenarios de personalización

Para añadir un atributo de búsqueda al bean de búsqueda en un lugar donde este atributo ya está disponible desde el motor de búsqueda, implemente el escenario 1 añadiendo el atributo al bean de búsqueda. Si el motor de búsqueda no soporta la búsqueda en este atributo, pero está disponible en una de las tablas en las que ya realiza las búsquedas, implemente el escenario 2 para añadir el atributo al motor. Si este atributo no está en una de las tablas en las que el motor de búsqueda realiza las búsquedas, implemente el escenario 3 para añadir la tabla al motor de búsqueda. El procedimiento para añadir atributos significativos al bean de búsqueda es el mismo que para añadir cualquier otro atributo al bean de búsqueda (implemente el escenario 1), aunque difiere ligeramente en la implementación. Consulte el escenario 4. Si desea ayudar a mejorar el rendimiento de la ejecución, implemente el escenario 5 para modificar y optimizar los scripts de definición de tabla de resumen creando una tabla para cada categoría a fin de reducir el número de filas en las que se realiza la búsqueda. Los escenarios siguientes describen cuándo se debe personalizar este componente:

Escenario 1: Añadir un atributo al bean de búsqueda
Entrar el atributo nuevo para el bean en el lugar donde ya está disponible en el motor de búsqueda.

Escenario 2: Añadir un atributo al motor de búsqueda
Entrar el atributo nuevo para el motor de búsqueda en el lugar donde ya realiza búsquedas en dicha tabla.

Escenario 3: Añadir una tabla al motor de búsqueda
Entrar la tabla nueva en el motor de búsqueda.

Escenario 4: Añadir atributos significativos al bean de búsqueda
Entrar atributos significativos en el bean.

Escenario 5: Mejorar el rendimiento con tablas de resumen optimizadas
Utilizar la información del conjunto de datos para optimizar la definición de tabla de resumen.

Nota: Las referencias al Motor de búsqueda, a la Interfaz de búsqueda y a la Infraestructura de búsqueda unificada son sinónimos y se utilizan de forma intercambiable en la documentación.

Escenario 1: Añadir un atributo al bean de búsqueda

Visión general de la implementación

Es posible que lo primero que desee personalizar sea la adición de otro atributo de búsqueda. Esta actividad requiere cambios en el bean de búsqueda y, opcionalmente, en el motor de búsqueda, si este atributo tampoco está disponible allí (se describe en el Escenario 2). La personalización del bean de búsqueda necesita que se establezca una subclase para el mismo y que se amplíen los métodos existentes. En el directorio `samples/search` se proporciona código de ejemplo que muestra cómo realizar dichas tareas.

Pasos para la personalización

Este ejemplo muestra cómo añadir atributos de búsqueda adicionales en el bean de búsqueda de catálogo, para el que existen las clases de metadatos en el paquete `com.ibm.search.catalog`. El bean de datos de búsqueda de catálogo es una clase. Para personalizarlo, realice lo siguiente:

1. Cree una subclase de la clase `CatEntrySearchListDataBean`.
2. Declare variables para los nuevos atributos de búsqueda que desea añadir. Puede declarar dos tipos de variables para almacenar la información asociada con un atributo de búsqueda. El primer tipo puede utilizarse para almacenar el valor para el atributo. El segundo tipo puede contener información relacionada que puede utilizarse en la búsqueda, por ejemplo operador de búsqueda, búsqueda booleana, sensible a las mayúsculas y minúsculas. La información almacenada en estas variables puede utilizarse en la generación de consultas de SQL que consultan información de catálogo.

   Nota: Las columnas de base de datos que se utilizan en la generación de consultas de SQL deben definirse anteriormente en la clase RuleQuery de la Interfaz de búsqueda o en una de sus subclases (consulte 'Personalización de la Interfaz de búsqueda') para que estén disponibles como restricciones en una consulta generada.
3. Cree métodos de acceso (métodos get y set) para las variables declaradas.
4. Cree un método `populate()` que realice lo siguiente:
   a. Cree una instancia de `RuleQuery` o de una subclase de `RuleQuery`.
   b. Haga referencia a esta instancia de `RuleQuery` en la instancia de la clase de bean padre utilizando el método `setRuleQuery(ruleQueryInstance)`.
   c. Llame al método `super.setPredefinedAttributes()`.
   d. Formule un `<Predicate>` para las nuevas restricciones de búsqueda que utilizan la Interfaz de búsqueda.
   e. Utilice `super.setIsAllNull(false)` para notificar al padre que se ha añadir un nuevo atributo de búsqueda.
   f. Llame al método `super.execute()`.

Ejemplo

Este ejemplo muestra cómo añadir atributos de búsqueda nuevos onSpecial en el bean de búsqueda de catálogo suponiendo que existen las clases de metadatos en el paquete `com.ibm.search.catalog`:

1. Cree una subclase nueva.
2. Cree variables nuevas para los atributos de búsqueda.

```
public class CustomCatEntrySearchListDataBean
        extends CatEntrySearchListDataBean {
  private static final String COPYRIGHT =
        com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;
  protected java.lang.String onSpecial;
  protected java.lang.String onSpecialOperator;
  }
```

La variable `onSpecial` se utiliza para almacenar el valor de los atributos de búsqueda. La variable `onSpecialOperator` se utiliza para almacenar el operador de búsqueda (por ejemplo like, =, <, >, <=, >=).

3. Cree métodos de acceso para todas las variables

```
public java.lang.String getOnSpecial(){
      return onSpecial;
}

public void setOnSpecial(java.lang.String newOnSpecial) {
  onSpecial = newOnSpecial;
}
```

4. Cree un método `populate()`

```
public void populate() throws Exception {
   String langId = commandContext.getLanguageId().toString();
   RuleQuery ruleQueryInstance = new RuleQuery();
   setRuleQuery(ruleQueryInstance);
   super.setPredefinedAttributes();
      if(onSpecial != null){
      ruleQueryInstance.addSelectAttribute(ruleQueryInstance.CATENTRY_ONSPECIAL_Attr,
  getNumeric(onSpecialOperator), onSpecial);
      ruleQueryInstance.addSelectOperator(ruleQueryInstance.AND_Operator);
      }

   super.setIsAllNull(false);
   }
   q.addSelectOperator(q.AND_Operator);
   }
   super.execute(); //Step 4f
}
```

Escenario 2: Añadir un atributo al motor de búsqueda

Visión general de la implementación

El motor de búsqueda necesita información sobre atributos y tablas a fin de formular correctamente las consultas para recuperar resultados del catálogo. Esta información se conserva como metadatos. Para ampliar el motor de búsqueda a fin de añadir atributos de búsqueda nuevos, se necesitan metadatos de atributo adicionales y, opcionalmente, definiciones de metadatos de tabla. Los metadatos de tabla sólo son necesarios para entrar una tabla nueva en el motor de búsqueda y se proporcionan detalles adicionales sobre los mismos en el Escenario 3. Una vez completado este escenario, deberá implementar el Escenario 1: Añadir un atributo al bean de búsqueda, a fin de dejarlo disponible para el diseñador de páginas JSP. En el directorio `samples/search` se proporciona código de ejemplo para realizar dicha acción.

Pasos para la personalización

El motor de búsqueda forma parte de la infraestructura de búsqueda unificada. Se representa como una clase, denominada `RuleQuery`, y clases de metadatos adicionales tales como `AttributeInfo` y `TableInfo` que describen los nombres de columna y tabla respectivamente. Este ejemplo muestra cómo añadir atributos nuevos al motor de búsqueda creando clases de metadatos para columnas de base de datos que no están en el paquete `com.ibm.search.catalog` (pero existen metadatos para la tabla a la que pertenece la columna). Para personalizar el motor de búsqueda, realice lo siguiente:

1. Defina metadatos de Interfaz de búsqueda para cada columna y tabla en la que desea que se puedan realizar búsquedas. Para ello es necesario lo siguiente:

a. Cada tabla de búsqueda debe tener una clase correspondiente que sea una subclase de la clase `TableInfo`. Esta subclase debe especificar el nombre de tabla.
b. Cada columna de búsqueda debe tener una clase correspondiente que sea una subclase de la clase `AttributeInfo`. Esta subclase debe especificar el nombre de columna, la subclase `TableInfo` a la que pertenece la columna y el tipo de datos SQL de la columna.

2. Cree una subclase de la clase `RuleQuery` y defina referencias de entero estáticas para cada columna nueva de la base de datos. Tenga en cuenta que los valores enteros 0 a 1000 están reservados para el sistema.
3. Cree un método `findAttributeInfoName()`. Para evitar que este método altere temporalmente el método `findAttributeInfoName()` padre, se recomienda llamar al método `super.findAttributeInfoName()`. Si se altera temporalmente el `findAttributeInfoName()` padre, las columnas de base de datos definidas en la clase padre no estarán disponibles.
4. Añada lógica de creación de clase de fábrica a este método para crear una clase de metadatos `AttributeInfo` para cada nueva columna de base de datos.
5. Modifique el archivo `search.xml` añadiendo relaciones de unión de tabla predefinidas para las tablas de búsqueda nuevas. Las relaciones de unión son necesarias para todas las combinaciones de tabla. Para modificar `search.xml`, realice lo siguiente:
   a. Añada una sección `COMMENT` para describir la relación de unión de tabla formulada por la entrada.
   b. Añada una sección `KEY` para identificar de forma exclusiva la entrada. La tabla que se une se utiliza como clave exclusiva.
   c. Añada una sección `VALUE` para especificar valores que están asociados con una clave. La entrada de la sección de valor (value) define una relación de unión. Los nombres de columna de la sección value se definen utilizando la subclase `AttributeInfo` de dicha columna.

Ejemplo

El ejemplo siguiente creará una nueva columna de base de datos de búsqueda `CATENTRY.ONSPECIAL`:

1.
   a. Cree una subclase de la clase `TableInfo` para especificar el nombre de tabla `CATENTRY` (si no existe la clase para esta tabla):

```
public final class CatEntryTableInfo extends TableInfo
{
   private static final String COPYRIGHT =
      com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;

   // singleton idiom in Java
   private static CatEntryTableInfo info = new CatEntryTableInfo ();

   /**
    * El constructor establece el nombre de tabla.
    */
         public CatEntryTableInfo() {
            super("CATENTRY");
         }
   /**
    * El método devuelve la expresión sigleton.
    * @return com.ibm.commerce.search.catalog.CatEntryTableInfo
    */
         public static CatEntryTableInfo getSingleton()
         {
            return info;
         }
}
```

   b. Cree una subclase de la clase `AttributeInfo` para especificar el nombre de columna ONSPECIAL:

```
public final class CatEntryOnSpecialAttributeInfo extends AttributeInfo
{
   private static final String COPYRIGHT =
      com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;

   // singleton idiom in Java
   private static CatEntryOnSpecialAttributeInfo info = new
         CatEntryOnSpecialAttributeInfo(CatEntryTableInfo.getSingleton());

   /**
    * El constructor establece el nombre de columna, el nombre de tabla de la columna y el tipo de datos.
    */
   public CatEntryOnSpecialAttributeInfo(TableInfo info) {
      super(info);
      columnName = "ONSPECIAL";
      sqltype = SQLTYPE_NUM;
   }

   /**
    * El método devuelve la expresión sigleton.
    * @return com.ibm.commerce.search.catalog.CatEntryOnSpecialAttributeInfo
    */
   public static CatEntryOnSpecialAttributeInfo getSingleton()
   {
      return info;
   }
}
```

2. Cree una subclase de la interfaz de búsqueda `RuleQuery`:

```
public class CustomQuery extends RuleQuery {
   private static final String COPYRIGHT =
      com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;

   public final static int CATENTRY_ONSPECIAL_Attr = 1001; //Referencia estática
}
```

3. Cree el método `findAttributeInfoName()`:

```
protected String findAttributeInfoName(int attrId) {
   String className;
      className = super.findAttributeInfoName(attrId);
      switch (attrId) {
         case CATENTRY_ONSPECIAL_Attr:
         className = new String("CatEntryOnSpecialAttributeInfo");
         break;
      }
      return className;
}
```

4. Modifique `Search.xml` si es necesario (en este ejemplo no es necesario modificar el archivo `search.xml`)

Nota: Para añadir estos metadatos nuevos al bean de búsqueda, siga los pasos del escenario 1. Pero deberá utilizar una instancia de la subconsulta de `RuleQuery` que tenga información de metadatos (consulte `CustomQuery` en el ejemplo anterior) en lugar de una instancia de la clase `RuleQuery`.

Escenario 3: Añadir una tabla al motor de búsqueda

Visión general de la implementación

Para añadir atributos de una tabla nueva no soportada actualmente por el motor de búsqueda, deberá definir información sobre la tabla y describir cómo se relaciona esta tabla con otras tablas utilizadas por el motor. Esta información son las relaciones de unión utilizadas para obtener predicados de unión que relacionan cada tabla con otras tablas y se conserva como metadatos. Para ampliar el motor de búsqueda para añadir tablas de búsqueda nuevas, deberá definir cada tabla y sus relaciones con otras tablas.

Deberá implementar el escenario 2: ″Añadir un atributo al motor de búsqueda″ para que los atributos relacionados con esta tabla estén disponibles para el creador del bean de búsqueda. En el directorio `samples/search` se proporciona código de ejemplo para realizar dicha acción.

Pasos para la personalización
Estos ejemplos muestran cómo crear clases de metadatos de relaciones de unión para tablas de la base de datos. Normalmente, deberá definir una relación de unión para cada combinación lógica de relaciones de tabla.

Ejemplo
Este ejemplo cambia un `search.xml` existente para incluir una entrada de relación de unión nueva entre la tabla CATENTRY y la tabla OFFERPRICE:

1. Añada la sección `COMMENT`:

```
<--  ***** WWWW *****
   setW.add("OFFER.CATENTRY_ID = CATENTRY.CATENTRY_ID");
   setW.add("OFFER.OFFER_ID = OFFERPRICE.OFFER_ID");
-->
```

2. Añada las secciones `KEY` y `VALUE`:

```
<entry>
   <key1>OFFERPRICE</key1>
   <key2>CATENTRY</key2>
   <value>
      <attrinfo1>OfferOfferIdAttributeInfo</attrinfo1>
      <attrinfo2>OfferPriceOfferIdAttributeInfo</attrinfo2>
   </value>
   <value>
      <attrinfo1>CatEntryIdentifierAttributeInfo</attrinfo1>
      <attrinfo2>OfferCatentryIdAttributeInfo</attrinfo2>
   </value>
</entry>
```

Escenario 4: Añadir atributos significativos al bean de búsqueda

Visión general de la implementación
El proceso para añadir búsqueda en los atributos significativos al bean es el mismo que para añadir cualquier otro atributo de búsqueda al bean. Una vez que se conoce el archivo y que se han identificado los atributos significativos, siga el escenario 1 para dejar estos atributos disponibles en el bean de búsqueda. La única diferencia entre este escenario y el escenario 1 se encuentra en los parámetros de método utilizados para crear el predicado de atributo significativo (consulte el texto en negrita del ejemplo siguiente para obtener detalles). El motor de búsqueda ya tiene la posibilidad de buscar los atributos significativos que están definidos en las tablas ATTRIBUTE y ATTRVALUE.

Pasos para la personalización
Los mismos que para el escenario de personalización 1, Añadir un atributo al bean de búsqueda.

Ejemplo
Este ejemplo muestra cómo añadir un nuevo atributo significativo de búsqueda, `color`, al bean de búsqueda de catálogo.

1. Cree una subclase nueva.
2. Cree variables nuevas para los atributos de búsqueda.

```
public class ExtendedCatEntrySearchListDataBean
        extends CatEntrySearchListDataBean implements
        ExtendedCatEntrySearchListInputDataBean,
        ExtendedCatEntrySearchListSmartDataBean {
  private static final String COPYRIGHT =
        com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;
```

```
protected java.lang.String colorValue;
protected java.lang.String colorValueCaseSensitive;
protected java.lang.String colorValueOperator;
}
```

La variable `colorValue` almacena el valor de los atributos de búsqueda. La variable `colorValueOperator` almacena el operador de búsqueda (por ejemplo like, =, <, >, <=, >=).

3. Cree métodos de acceso para todas las variables

```
public java.lang.String getColorValue(){
      return colorValue;
}

public void setColorValue(java.lang.String newColorValue) {
  colorValue = newColorValue;
}
```

4. Cree un método `populate()`

```
public void populate() throws Exception {
   String langId = commandContext.getLanguageId().toString();
   RuleQuery ruleQueryInstance = new RuleQuery();
   setRuleQuery(ruleQueryInstance);
   super.setPredefinedAttributes();
      if(isEmpty(colorValue)){
      if(colorValueCaseSensitive ==null || !colorValueCaseSensititve.equals(CASE_SENSITIVE)){

         ruleQueryInstance.addSelectAttribute("Color",
     getStringOperator(colorValueOperator),
           colorValue.toUpperCase(),
           ruleQueryInstance.ATTRVALUE_STRINGVALUE_Attr,langId, null, ruleQueryInstance.UPPER_Function);
        super.setIsAllNull=false;
        }else{
        ruleQueryInstance.addSelectAttribute("Color",
     getStringOperator(colorValueOperator),
           colorValue,
           ruleQueryInstance.ATTRVALUE_STRINGVALUE_Attr,langId, null);
        super.setIsAllNull=false;
        }
        ruleQueryInstance.addSelectOperator(q.AND_Operator);
   }
super.setIsAllNull(false);

q.addSelectOperator(q.AND_Operator);
}
super.execute(); //Paso 4f
}
```

Escenario 5: Mejorar el rendimiento con tablas de resumen optimizadas

Visión general de la implementación

Cuando el archivo está definido, se puede utilizar esta información para crear tablas de resumen mejores a fin de mejorar de forma significativa el rendimiento de la búsqueda. El objetivo consiste en crear tablas que parezcan los tipos de consulta anticipados que se van a encontrar. En la mayoría de las situaciones, estas tablas de resumen serán mucho más pequeñas y, de este modo, las consultas aplicadas a través de estas tablas serán mucho más rápidas que si se utilizan las tablas nativas subyacentes de mayor tamaño u otras tablas de resumen menos óptimas.

Pasos para la personalización

Este ejemplo muestra cómo crear una tabla de resumen de grupo de categorías de atributos significativos para cada grupo de categorías. Si existen muchos grupos de categorías (`CATGROUPS`) que tienen artículos con atributos significativos, cree una definición de tabla de resumen para cada grupo.

Pasos para crear esta tabla de resumen:

1. Defina una sentencia de creación de tabla con una consulta de selección que se deberá utilizar como definición de tabla de resumen. Esta sentencia de selección (select) debe tener especificadas las mismas columnas en una cláusula ″group by″ tal como se lista en el conjunto de resultados. Para obtener más detalles, consulte la documentación de DB2 para las ″Tablas de resumen automáticas″ (AST) o la documentación de Oracle para las ″Vistas materializadas.″
2. En la sentencia de selección (select), especifique un predicado de grupo de categorías para la categoría.
3. Cree el índice con las mismas columnas de conjunto de resultados que en la consulta anterior y en el mismo orden especificado en dicha consulta.
4. Repita los pasos anteriores para cada grupo de categorías.

Ejemplo

Este ejemplo muestra un script de creación de tabla de resumen para atributo significativo con grupos de categorías para cada uno de tres grupos de categorías 10000, 10001 y 10002. Las partes que están en negrita son nuevas respecto a la definición general mostrada en el archivo `wcs.summary.richAttributeCatGroup.create.sql` en el que se basa este ejemplo.

```
// Definición de tabla de resumen para catgroup 10000
CREATE TABLE RICHATTRCG0 AS
(
  SELECT ATTRIBUTE.NAME,
         ATTRIBUTE.LANGUAGE_ID,
         ATTRVALUE.CATENTRY_ID,
         ATTRVALUE.FLOATVALUE,
         ATTRVALUE.INTEGERVALUE,
         ATTRVALUE.STRINGVALUE,
         CATGPENREL.CATGROUP_ID,
         COUNT(*) AS C5
  FROM ATTRIBUTE, ATTRVALUE, CATGPENREL
  WHERE ATTRIBUTE.ATTRIBUTE_ID=ATTRVALUE.ATTRIBUTE_ID
         AND ATTRIBUTE.CATENTRY_ID=CATGPENREL.CATENTRY_ID
         AND CATGPENREL.CATGROUP_ID = 10000
  GROUP BY ATTRIBUTE.NAME,
           ATTRIBUTE.LANGUAGE_ID,
           ATTRVALUE.CATENTRY_ID,
           ATTRVALUE.FLOATVALUE,
           ATTRVALUE.INTEGERVALUE,
           ATTRVALUE.STRINGVALUE,
           CATGPENREL.CATGROUP_ID
)DATA INITIALLY DEFERRED REFRESH IMMEDIATE NOT LOGGED INITIALLY;
ALTER TABLE RICHATTRCG0 ACTIVATE NOT LOGGED INITIALLY;
REFRESH TABLE RICHATTRCG0;
commit;

SET CURRENT REFRESH AGE = ANY;

CREATE INDEX I_RICHATTRCG0 ON RICHATTRCG0
(
   NAME           ASC,
   LANGUAGE_ID    ASC,
   CATENTRY_ID    ASC,
   FLOATVALUE     ASC,
   INTEGERVALUE   ASC,
   STRINGVALUE     ASC,
   CATGROUP_ID     ASC
);
```

```
commit;

// Definición de tabla de resumen para catgroup 10001
CREATE TABLE RICHATTRCG1 AS
(
  SELECT ATTRIBUTE.NAME,
         ATTRIBUTE.LANGUAGE_ID,
         ATTRVALUE.CATENTRY_ID,
         ATTRVALUE.FLOATVALUE,
         ATTRVALUE.INTEGERVALUE,
         ATTRVALUE.STRINGVALUE,
         CATGPENREL.CATGROUP_ID,
         COUNT(*) AS C5
  FROM ATTRIBUTE, ATTRVALUE, CATGPENREL
  WHERE ATTRIBUTE.ATTRIBUTE_ID=ATTRVALUE.ATTRIBUTE_ID
        AND ATTRIBUTE.CATENTRY_ID=CATGPENREL.CATENTRY_ID
        AND CATGPENREL.CATGROUP_ID = 10001
  GROUP BY ATTRIBUTE.NAME,
           ATTRIBUTE.LANGUAGE_ID,
           ATTRVALUE.CATENTRY_ID,
           ATTRVALUE.FLOATVALUE,
           ATTRVALUE.INTEGERVALUE,
           ATTRVALUE.STRINGVALUE,
           CATGPENREL.CATGROUP_ID
)DATA INITIALLY DEFERRED REFRESH IMMEDIATE NOT LOGGED INITIALLY;
ALTER TABLE RICHATTRCG1 ACTIVATE NOT LOGGED INITIALLY;
REFRESH TABLE RICHATTRCG1;

commit;

SET CURRENT REFRESH AGE = ANY;

CREATE INDEX I_RICHATTRCG1 ON RICHATTRCG1
(
   NAME           ASC,
   LANGUAGE_ID    ASC,
   CATENTRY_ID    ASC,
   FLOATVALUE     ASC,
   INTEGERVALUE   ASC,
   STRINGVALUE    ASC,
   CATGROUP_ID    ASC
);

commit;

// Definición de tabla de resumen para catgroup 10002
CREATE TABLE RICHATTRCG2 AS
(
  SELECT ATTRIBUTE.NAME,
         ATTRIBUTE.LANGUAGE_ID,
         ATTRVALUE.CATENTRY_ID,
         ATTRVALUE.FLOATVALUE,
         ATTRVALUE.INTEGERVALUE,
         ATTRVALUE.STRINGVALUE,
         CATGPENREL.CATGROUP_ID,
         COUNT(*) AS C5
  FROM ATTRIBUTE, ATTRVALUE, CATGPENREL
  WHERE ATTRIBUTE.ATTRIBUTE_ID=ATTRVALUE.ATTRIBUTE_ID
        AND ATTRIBUTE.CATENTRY_ID=CATGPENREL.CATENTRY_ID
        AND CATGPENREL.CATGROUP_ID = 10002  GROUP BY ATTRIBUTE.NAME,
```

```
            ATTRIBUTE.LANGUAGE_ID,
            ATTRVALUE.CATENTRY_ID,
            ATTRVALUE.FLOATVALUE,
            ATTRVALUE.INTEGERVALUE,
            ATTRVALUE.STRINGVALUE,
            CATGPENREL.CATGROUP_ID
)DATA INITIALLY DEFERRED REFRESH IMMEDIATE NOT LOGGED INITIALLY;
ALTER TABLE RICHATTRCG2 ACTIVATE NOT LOGGED INITIALLY;
REFRESH TABLE RICHATTRCG2;

commit;

SET CURRENT REFRESH AGE = ANY;

CREATE INDEX I_RICHATTRCG2 ON RICHATTRCG2
(
   NAME              ASC,
   LANGUAGE_ID       ASC,
   CATENTRY_ID       ASC,
   FLOATVALUE        ASC,
   INTEGERVALUE      ASC,
   STRINGVALUE       ASC,
   CATGROUP_ID       ASC
);

commit;
```

Variables de bean de datos de búsqueda de catálogo

Lista de variables que se deben pasar al bean de búsqueda de catálogo a través del URL (JSP.)

Tabla 7.

Nombre	Tipo de datos	Descripción
beginIndex	serie	Esta variable se utilizar para paginar el conjunto de resultados. El valor debe ser el índice de la primera fila de resultados de una página.
catGroupId	serie	El valor de esta variable se utiliza en la búsqueda de un ID de categoría.
categoryTerm	serie	El valor de esta variable es utiliza en las búsquedas de nombre y/o descripciones de categoría.
categoryTermCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para las búsquedas sensibles a las mayúsculas y minúsculas o `no` para las búsquedas no sensibles a las mayúsculas y minúsculas.
categoryTermOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE`, para el operador like, o `EQUAL`, para el operador equal.

Tabla 7. (continuación)

categoryTermScope	Entero	Un usuario puede restringir el ámbito de una búsqueda a uno entre nombre, nombre y descripción corta o nombre, descripción corta y descripción larga. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser uno entre 1 para nombre y descripción corta, 2 para nombre solamente o 3 para nombre, descripción corta y descripción larga.
categoryType	serie	Un usuario puede especificar tres tipos de criterios de búsqueda: All, Any o Exact Phrase. El valor de esta variable se utiliza para almacenar los criterios de búsqueda de un usuario. El valor debe ser `ALL` para todos los criterios de búsqueda, `ANY` para cualquier criterio de búsqueda o `EXACT` para criterios de búsqueda de frase exacta.
currency	serie	El valor de esta variable se utiliza en las búsqueda de una moneda.
currencyCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.
currencyOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para un operador like o `EQUAL` para un operador equal.
filterTerm	serie	El valor de esta variable se utiliza para filtrar una búsqueda de un valor especificado.
filterTermCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.
filterTermOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para el operador like o `EQUAL` para el operador equal.

Tabla 7. (continuación)

filterType	serie	Un usuario puede especificar tres tipos de criterios de búsqueda `All`, `Any` o `Exact Phrase`. El valor de esta variable se utiliza para almacenar los criterios de búsqueda de un usuario. El valor debe ser uno entre `ALL` para todos los criterios de búsqueda, `ANY` para cualquier criterio de búsqueda o `EXACT` para criterios de frase exacta.
isListPriceOn	serie	El usuario del bean de búsqueda de catálogo puede elegir exponer el precio de catálogo o el precio estándar. Para exponer el precio de catálogo, establezca esta variable en `yes` y para exponer el precio estándar, establezca esta variable en `no`. Por omisión, el bean de búsqueda expone el precio estándar.
manufacturer	serie	El valor de esta variable se utiliza para buscar el nombre de un fabricante.
manufacturerCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.
manufacturerOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para el operador like o `EQUAL` para el operador equal.
manufacturerPartNum	serie	El valor de esta variable se utiliza en la búsqueda del número de pieza de un fabricante.
manufacturerPartNumCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.
manufacturerPartNumOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para un operador like o `EQUAL` para un operador equal.
maxPrice/minPrice	serie	Los valores de estas variables se utilizan en la búsqueda de rango de precios.
pageSize	serie	El valor de esta variable especifica el número de filas de resultado de búsqueda que se deben visualizar por página.
price	serie	El valor de esta variable se utiliza en la búsqueda de precio.

Tabla 7. (continuación)

priceOperator	serie	Un usuario puede elegir uno de los operadores siguientes como operador de búsqueda: =, <, >, !=, <=, >=. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser cualquiera de los siguientes: `EQUAL`, `NOTEQUAL`, `GREATER`, `LESS`, `GREATER_EQUAL`, `LESS_EQUAL`.
qtyAvailable	serie	El valor de esta variable se utiliza en la búsqueda de inventario de un producto o artículo.
qtyAvailableOperator	serie	Un usuario puede elegir uno de los operadores siguientes como operador de búsqueda: =, <, >,!=, <=, >=. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser cualquiera de los siguientes:`EQUAL`, `NOTEQUAL`, `GREATER`, `LESS`, `GREATER_EQUAL`, `LESS_EQUAL`
qtyMeasure	serie	El valor de esta variable se utiliza en la búsqueda de medida de cantidad.
qtyMeasureCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.
qtyMeasureOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para un operador like o `EQUAL` para un operador equal.
resultCount	serie	Esta variable contendrá el número total de resultados devueltos para una búsqueda.
resultType	serie	El comerciante puede especificar si desea mostrar Productos o Artículos o bien Productos y Artículos en el resultado de la búsqueda. El valor de esta variable se utiliza para almacenar este valor. El valor debe ser uno entre `1` para sólo productos, `2` para sólo artículos o `3` para productos y artículos.
searchTerm	serie	El valor de esta variable se utiliza en la búsqueda de una palabra.
searchTermCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.

Tabla 7. (continuación)

searchTermOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para un operador like o `EQUAL` para un operador equal.
searchTermScope	entero	Un usuario puede restringir el ámbito de una búsqueda a uno de nombre, nombre y descripción corta, nombre, descripción corta y descripción larga o palabra clave. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser uno entre `1` para nombre y descripción corta, `2` para nombre solamente, `3` para nombre, descripción corta y descripción larga o `4` para palabra clave.
searchType	serie	Un usuario puede especificar tres tipos de criterios de búsqueda `All`, `Any` o `Exact Phrase`. El valor de esta variable se utiliza para almacenar los criterios de búsqueda de un usuario. El valor debe ser `ALL` para todos los criterios de búsqueda, `ANY` para cualquier criterio de búsqueda o `EXACT` para criterios de frase exacta.
sku	serie	El valor de esta variable se utiliza en las búsquedas de un código de artículo.
skuCaseSensitive	serie	Un usuario puede elegir la búsqueda sensible a las mayúsculas y minúsculas o no sensible a las mayúsculas y minúsculas. El valor de esta variable se utiliza para identificar si una búsqueda es sensible a las mayúsculas y minúsculas o no lo es. El valor debe ser `yes` para una búsqueda sensible a las mayúsculas y minúsculas o `no` para una búsqueda no sensible a las mayúsculas y minúsculas.
skuOperator	serie	Un usuario puede elegir `like` o `equal` como operadores de búsqueda. El valor de esta variable se utiliza para almacenar la elección de un usuario. El valor debe ser `LIKE` para un operador like o `EQUAL` para un operador equal.

Columnas de base de datos de búsqueda de catálogo

Lista de columnas de base de datos en las que el bean de búsqueda de catálogo puede buscar:

Tabla 8.

Atributo de búsqueda definido en interfaz de bean	**Tabla**	**Columnas**	**Soporta expresiones booleanas**
Category Group - Todos los nombres y las descripciones	CatGrpDesc	NAME LONGDESCRIPTION SHORTDESCRIPTION	SÍ
Category Group - Sólo el título	CatGrpDesc	NAME	SÍ

Tabla 8. (continuación)

Atributo de búsqueda definido en interfaz de bean	**Tabla**	**Columnas**	**Soporta expresiones booleanas**
Category Group - Título y descripción (valor por omisión sugerido)	CatGrpDesc	NAME SHORTDESCRIPTION	SÍ
Category Language ID	CatGrpDesc	LANGUAGE_ID	NO
Category Identifier	CatGpEnRel	CATGROUP_ID	No
CatEntry Description - Todos los nombres y las descripciones	CatEntDesc	NAME LONGDESCRIPTION SHORTDESCRIPTION	SÍ
CatEntry - Sólo el título	CatEntDesc	NAME	SÍ
CatEntry - Título y descripción (valor sugerido)	CatEntDesc	NAME SHORTDESCRIPTION	SÍ
CatEntry Language ID	CatEntDesc	LANGUAGE_ID	NO
CatEntry Manufacturer Name	CatEntry	MFNAME	NO
CatEntry Manufacturer PartNumber	CatEntry	MFPARTNUMBER	NO
CatEntry SKU	CatEntDesc	PARTNUMBER	NO
Price	Listprice/ Offerprice	PRICE	NO
Price Range	Listprice/ Offerprice	PRICE	No
Currency	Listprice/ Offerprice	CURRENCY	NO
Quantity Available	InvStVw/ StoreInv	QTYAVAILABLE	NO
Quantity Measure	InvStVw/ StoreInv	QUANTITYMEASURE	NO

El usuario del bean puede exponer más columnas como atributos significativos personalizando el bean. Para obtener más información sobre la personalización, consulte el JavaDoc para la clase ExtendedCatEntrySearchListDataBean en la ayuda en línea.

Capítulo 7. Cupones

Los elementos descritos en este capítulo representan la interfaz de usuario para los componentes Cupones de WebSphere Commerce Accelerator. Facilitan la creación y el mantenimiento de cupones, ofertas de cupones y otras tareas que un Vendedor o Comerciante realiza diariamente. El componente Cupones incluye los elementos siguientes:

- Asistente de cupones
- Cuaderno Cupones

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar esta parte de WebSphere Commerce Accelerator.

Escenario 1: Añadir información nueva al crear descuentos de cupones

Visión general de la implementación

Este escenario le guía durante la realización de los pasos necesarios para personalizar las herramientas de Cupón a fin de registrar información adicional en el cupón, por ejemplo quién ha creado el descuento de cupón.

Pasos para la personalización

1. Implemente un **CustomCreateCouponDiscountCmdImpl** nuevo que amplíe la implementación del mandato de tarea **CreateCouponDiscountCmdImpl** e implemente **CreateCouponDiscountCmd**.
2. Utilice la columna `FIELD1` de la tabla `CALCODE` para almacenar el id del usuario (`userId`) que ha creado este descuento de cupón.
3. Registre el nuevo mandato de tarea en la tabla `CMDREG`. Suponga que el ID de tienda de cliente es 1 y, a continuación, ejecute la sentencia de SQL siguiente:

```
insert into cmdreg( storeent_id,interfacename, classname)
  values(1,'com.ibm.commerce.tools.ecoupon.CreateCouponDiscountCmd',
  'com.ibm.commerce.tools.ecoupon.CustomCreateCouponDiscountCmdImpl')
```

4. Actualice **CustomCreateCouponDiscountCmdImpl**. En el mandato, implemente el método `performExecute()`:

```
public void performExecute() throws ECSystemException, ECException {
   super.performExecute();
   /** Obtener el Id de usuario del contexto del mandato
   Almacenar el Id de usuario en la tabla calcode
   **/
   }
```

Capítulo 8. Descuentos

Los elementos descritos en este capítulo representan la interfaz de usuario para los componentes Descuentos de WebSphere Commerce Accelerator. Facilitan la creación y el mantenimiento de descuentos y otras acciones que un Vendedor o Comerciante realiza diariamente. El componente Descuentos incluye los elementos siguientes:

- Asistente de descuento
- Cuaderno Descuento

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar el componente Descuentos de WebSphere Commerce Accelerator.

Escenario 1: Añadir información adicional al crear descuentos

Visión general de la implementación

Este escenario le guía durante los pasos necesarios que debe realizar si desea añadir información adicional al crear descuentos. El ejemplo supone que desea crear un seguimiento de comprobación de quién ha creado el descuento. Este escenario no necesita que se sometan parámetros nuevos.

Pasos para la personalización

1. Implemente un **CustomCreateDiscountCmdImpl** nuevo, que amplíe la implementación del mandato de tarea **CreateDiscountCmdImpl** e implemente **CreateDiscountCmd**.
2. Utilice la columna field1 de la tabla **CALCODE** para almacenar el Id del usuario que crea este descuento.
3. Registre el nuevo mandato de tarea en la tabla **CMDREG**. Suponga que el ID de tienda de cliente es 1 y, a continuación, ejecute la sentencia de SQL siguiente:

```
insert into cmdreg( storeent_id,interfacename, classname)
  values(1,'com.ibm.commerce.tools.promotions.CreateDiscountCmd',
  'com.ibm.commerce.tools.promotions.CustomCreateDiscountCmdImpl')
```

4. En **CustomCreateDiscountCmdImpl**, implemente el método performExecute():

```
public void performExecute() throws ECSystemException, ECException {
  super.performExecute();
   /** Obtener el Id de usuario del contexto del mandato
   Almacenar el Id de usuario en la tabla calcode
  **/
  }
```

Escenario 2: Cambiar el comportamiento por omisión

Visión general de la implementación

Este escenario cambia el comportamiento por omisión del asistente de descuento para que pueda especificar la secuencia en la que se aplica un descuento en un pedido. Este escenario también necesita que se pasen parámetros nuevos mediante el URL. Para ello es necesario añadir un campo en el primer panel del asistente de Descuento a fin de obtener los datos adicionales.

Pasos para la personalización

1. Actualice el archivo JSP responsable del asistente de Descuento realizando lo siguiente:
   a. Vaya al directorio siguiente:

      AIX `/usr/WebSphere/CommerceServer/wcstool.war/tools/promotions/`

      400
      `/QIBM/UserData/WebSphere/CommerceServer/wcstool.war/tools/promotions/`

      Linux `/opt/WebSphere/CommerceServer/wcstool.war/tools/promotions/`

      Solaris `/opt/WebSphere/CommerceServer/wcstool.war/tools/promotions/`

      Windows
      unidad`:\WebSphere\CommerceServer\wcstool.war\tools\promotions\`

   b. Copie `DiscountWizardWelcome.jsp` en `MyDiscountWizardWelcome.jsp`
   c. Actualice `MyDiscountWizardWelcome.jsp`. En el bloque de formato, añada el código siguiente:

      ```
      <input name="sequence" type="TEXT" size=3 MAXLENGTH=3>  Sequence
      ```

   d. Guarde la entrada en la infraestructura. En el JSP, hay una función JavaScript `savePanelData`. Añada la línea siguiente en la función:

      ```
      parent.put("sequence", document.welcomeForm.sequence.value);
      ```

   e. Registre una entrada nueva en la tabla VIEWREG. Por ejemplo, si el ID de tienda de cliente es 1, ejecute la sentencia de SQL siguiente:

      ```
      insert into viewreg (viewname, devicefmt_id, storeent_id, interfacename,
            classname, properties, https, internal)
        values('CusDiscountWizWelcomeView', -1, 0,
            'com.ibm.commerce.tools.command.ToolsForwardViewCommand',
            'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
            'docname=tools/promotions/extend/CusDiscountWizardWelcome.jsp', 0, 1)
      ```

2. Implemente un **DiscountSaveCmdCustomImpl** nuevo, que amplíe la implementación del mandato de controlador **DiscountSaveCmdImpl** e implemente **DiscountSaveCmd**.
3. Registre esta nueva implementación en la tabla `CMDREG`. Suponga que el ID de tienda de cliente es 1 y, a continuación, ejecute la sentencia de SQL siguiente:

   ```
   insert into cmdreg(storeent_id,interfacename,classname,target)
      values (1,'com.ibm.commerce.tools.promotions.DiscountSaveCmd',
      'com.ibm.commerce.tools.promotions.DiscountSaveCmdCustomImpl')
   ```

4. En el mandato **DiscountSaveCmdCustomImpl**, implemente dos métodos utilizando el código de ejemplo siguiente:

   ```
   Public  void validateParameters() {
     super.validateParameters();
     /*
     Obtener los parámetros nuevos de requestProperty
     */
     }
   Public void customMethod() {
     super.customMethod();
     /*
     Aquí la acción.
     */
     }
   ```

Capítulo 9. Respuesta a RFQ

Ejemplos de personalización

En este documento, proporcionamos tres ejemplos de personalización. Estos ejemplos son:

- Copiar una respuesta
- Suprimir el estado de Retractada del ciclo de vida de una respuesta a RFQ
- Entrar información descriptiva para una respuesta durante la creación

Escenario 1: Copiar una respuesta

Esta personalización de ejemplo añade funcionalidad a las herramientas de respuesta a RFQ lo que permite a un usuario crear un respuesta a RFQ duplicada basándose en la respuesta a RFQ seleccionada. La personalización incluye la adición del botón **Duplicar** que, cuando se pulsa, solicita al usuario que entre un nombre exclusivo para la respuesta duplicada. Al crearse, la nueva respuesta está en estado `borrador`.

Visión general de la implementación

Recupere toda la información de detalle de esta respuesta y cree la respuesta nueva utilizando la información recuperada. Dado que el nombre de respuesta no puede ser el mismo que en el esquema actual, necesitamos proporcionar una página para permitir que los usuarios proporcionen otro nombre. Esto ayuda a preparar toda la información en esta página y la pasa al mandato **RFQResponseCopyCmd**, que presenta un suceso de crear respuesta. Entonces UBF llama al mandato **RFQResponseCreateCmd** para crear una respuesta.

Pasos para la personalización

1. Añada una interfaz **RFQResponseCopyCmd** y su implementación **RFQResponseCopyCmdImpl** realizando lo siguiente:
   a. Añada la nueva interfaz **RFQResponseCopyCmd**. A continuación se muestra el código que define la interfaz:

```
public interface RFQResponseCopyCmd extends ControllerCommand{
   public final static String NAME =
"com.ibm.commerce.rfq.commands.RFQResponseCopyCmd";
   public final static String defaultCommandClassName =
      "com.ibm.commerce.rfq.commands.RFQResponseCopyCmdImpl";
}
```

   b. Añada la implementación de mandato **RFQResponseCopyCmdImpl** ampliando `com.ibm.commerce.ubf.commands.ToolsBusinessFlowEventCmdImpl` e implementando **RFQResponseCopyCmd**. Este mandato utiliza los datos sometidos mediante el diálogo para desencadenar un suceso de crear respuesta. El suceso hace que UBF llame a **RFQResponseCreateCmdImpl** para crear una respuesta nueva. A continuación se muestra el código que define el mandato:

```
public void performExecute() throws ECException
{
   TypedProperty parms = null;
   BusinessFlowEventData data = null;
   try {
    parms = getRequestProperties();
 parms.put(BusinessFlowConstants.EC_FLOWID,RFQConstants.EC_FLOW_RESPONSE_ID);
 parms.put(BusinessFlowConstants.EC_BUSINESS_FLOW_EVENT_IDENTIFIER,
```

```
          "createRFQResponse");

data = new BusinessFlowEventData(getCommandContext(), parms);
BusinessFlowEvent event = new BusinessFlowEvent(data,true);

parms = data.getResponseProperties();
responseProperties = new TypedProperty();

for (Enumeration pns = parms.keys(); pns.hasMoreElements();) {
   String paramName = (String) pns.nextElement();
if (paramName.equals(ECConstants.EC_VIEWTASKNAME))
   responseProperties.put(ECConstants.EC_VIEWTASKNAME, "DialogNavigation");
else if (paramName.equals(UIProperties.SUBMIT_FINISH_MESSAGE))
   responseProperties.put(UIProperties.SUBMIT_FINISH_MESSAGE,
             "¡La respuesta se ha copiado satisfactoriamente!");
else {
           Object newVal = parms.get(paramName);
           responseProperties.put(paramName, newVal);
}
}
} catch (ECException e) {
   parms = e.getErrorProperties();
responseProperties = new TypedProperty();
responseProperties.put(ECConstants.EC_VIEWTASKNAME, "DialogNavigation");
responseProperties.put(UIProperties.SUBMIT_ERROR_STATUS, "ERROR");
responseProperties.put(UIProperties.SUBMIT_ERROR_MESSAGE,
        "La copia de la respuesta ha fallado, entre otro nombre.");
throw new ECApplicationException(new ECMessage(ECMessageSeverity.INFO,
        ECMessageType.USER, "_ERR_RESPONSE_COPY",
        "com.ibm.commerce.tools.rfq.properties.RFQMessages"),
        this.getClass().getName(), "",
        "DialogNavigation",responseProperties);
  }
}
```

c. Registre **RFQResponseCopy** en la tabla URLREG. Ejecute el SQL siguiente para registrar el mandato:

```
insert into urlreg values('RFQResponseCopy',0,'com.ibm.commerce.rfq.commands.RFQResponseCopyCmd',0,null,null,1);
```

2. Añada un archivo JSP para permitir a los usuarios entrar un nombre para la respuesta y que recupere toda la información asociada con la respuesta:

   a. Este archivo JSP proporciona un campo de texto para que los usuarios faciliten un nombre de respuesta nuevo. A continuación se muestra el código fuente para el campo de texto:

```
<FORM name="responseCopyForm>
<table>
    <tr><td>
       <label><%= rfqNLS.get("name") %> <%= rfqNLS.get("required") %></label>
  </td></tr>
  <tr><td>
       <input type="Text" name="response_name" size="30" maxlength="200">
  </td></tr>
</table>
</FORM>
```

   Cuando los usuarios pulsan **Aceptar**, el código siguiente guarda el nombre de respuesta en el campo de entrada:

```
function savePanelData() {
var form = document.responseCopyForm;
parent.put("<%= RFQConstants.EC_RFQ_RESPONSE_NAME %>",form.response_name.value);
return true;
}
```

   Al inicializar este archivo JSP, el código siguiente recupera toda la información asociada con la respuesta y la guarda. La información se proporcionará a **RFQResponseCopyCmdImpl**.

```
function initializeState(){
parent.setContentFrameLoaded(false);
parent.put("<%= RFQConstants.EC_RFQ_REQUEST_ID %>"  ,<%= RequestId %>);
parent.put("<%= RFQConstants.EC_RFQ_RESPONSE_REMARK%>",
      "<%= UIUtil.toJavaScript((String)RFQres.getRemarks())%>");
```

```
var rfqCommentsArrary = new Array() ;
<%
RFQResCommentsPair[] commentsPair =
        RFQResProdHelper.getRFQLevelCommentsPair(RequestId,ResponseId,null);
for (int index=0; commentsPair  != null && index <commentsPair.length; index++){
  %>

rfqCommentsArrary[<%=index%>] = new Object();
   rfqCommentsArrary[<%=index%>].<%=RFQConstants.EC_REQUEST_TC_ID%>="<%=
                commentsPair[index].getRFQ_TC_ID() %>";
           rfqCommentsArrary[<%=index%>].<%=RFQConstants.EC_TC_RFQ_LEVEL_COMMENTS%>="
              <%= UIUtil.toJavaScript((String)commentsPair[index].getRFQ_value())%>";
           rfqCommentsArrary[<%=index%>].<%=RFQConstants.EC_ATTR_MANDATORY%>= "
               <%= commentsPair[index].getMandatory() %>";
           rfqCommentsArrary[<%=index%>].<%=RFQConstants.EC_ATTR_CHANGEABLE%>= "
              <%= commentsPair[index].getChangeable()%>";
        rfqCommentsArrary[<%=index%>].<%=RFQConstants.EC_ATTR_RES_CMMENTS_VALUE%>="
              <%= UIUtil.toJavaScript((String)commentsPair[index].getRes_value())%>";
<%}%>
parent.put("<%= RFQConstants.EC_TC_RFQ_LEVEL_COMMENTS %>",rfqCommentsArrary);

var ProductsArray = new Array();
<%
RFQResNewProd[] ResPros = RFQResProdHelper.getResAllProds(RequestId,ResponseId,langId);
int i=0;
for(;ResPros != null && i < ResPros.length;i++){
%>
   ProductsArray[<%=i%>] = new Object();
    ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_CATENTRYID%>="
          <%=ResPros[i].getCatentry_id() %>";
      ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRICE%>= "
          <%=ResPros[i].getPrice() %>";
    ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_QUANTITY%>="
          <%=ResPros[i].getQuantity() %>";
 ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_CURRENCY%>= "
          <%=ResPros[i].getCurrency() %>";
 ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_UNIT%>="
          <%=ResPros[i].getUnit() %>";
 ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>=new Array();
    ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>=new Array();
<%
RFQResProdAttributes[] resAttrs=RFQResProdHelper.getResAllAttributes(RequestId,
           ResponseId, ResPros[i].getCatentry_id(), langId,
           rfqNLS.get("valuedelim").toString());
int m=0,n=0;
for (int j=0;resAttrs != null && j<resAttrs.length>

ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m++%>] = new Object;
ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m-1%>].
              <%=RFQConstants.EC_ATTR_PATTRID%> = "<%= resAttrs[j].getPAttribute_id()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m-1%>].
              <%=RFQConstants.EC_ATTR_NAME%> = "<%= UIUtil.toJavaScript((String)resAttrs[j].getName())%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m-1%>].
              <%=RFQConstants.EC_ATTR_VALUE%> = "<%= UIUtil.toJavaScript((String)resAttrs[j].getRes_value())%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m-1%>].
              <%=RFQConstants.EC_ATTR_MANDATORY%> = "<%= resAttrs[j].getMandatory()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m-1%>].
              <%=RFQConstants.EC_ATTR_CHANGEABLE%> = "<%= resAttrs[j].getChangeable()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_ATTR_PRODUCT_COMMENTS%>[<%=m-1%>].
              <%=RFQConstants.EC_REQUEST_TC_ID%> = "<%= resAttrs[j].getReq_tc_id()%>";
<%}else{%>
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n++%>] = new Object;
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_PATTRID%> = "<%= resAttrs[j].getPAttribute_id()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_NAME%> = "<%= UIUtil.toJavaScript((String)resAttrs[j].getName())%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_VALUE%> = "<%= UIUtil.toJavaScript((String)resAttrs[j].getRes_value())%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_VALUEDELIM%> = "<%=rfqNLS.get("valuedelim")%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_MANDATORY%> = "<%= resAttrs[j].getMandatory()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_CHANGEABLE%> = "<%= resAttrs[j].getChangeable()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_REQUEST_TC_ID%> = "<%= resAttrs[j].getReq_tc_id()%>";
   ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_OPERATOR%> = "<%= resAttrs[j].getOperator_id()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_UNIT%> = "<%= resAttrs[j].getUnit()%>";
ProductsArray[<%=i%>].<%=RFQConstants.EC_OFFERING_PRODATTRLIST%>[<%=n-1%>].
              <%=RFQConstants.EC_ATTR_TYPE%> = "<%= resAttrs[j].getType()%>";

<% }

}%>
<%
}
%>
parent.put("<%= RFQConstants.EC_OFFERING_PRODITEM %>",ProductsArray);

parent.setContentFrameLoaded(true);
}
.
.
.
<BODY class="content" onLoad="initializeState()">
```

b. Ejecute la sentencia de SQL siguiente para registrar un mandato de vista en la tabla VIEWREG que correlacione el nuevo archivo JSP.

```
insert into viewreg values('RFQRspDuplicateDialog', -1, 0,
   'com.ibm.commerce.tools.command.ToolsForwardViewCommand',
   'com.ibm.commerce.tools.command.ToolsForwardViewCommandImpl',
   'docname=tools/rfq/rfq_response_duplicate_dialog.jsp', null, 0, null, 0);
```

c. Añada un archivo XML nuevo denominado `rfq_response_duplicate_dialog.xml` en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/xml/tools/rfq`

400 `/QIBM/UserData/WebSphere/CommerceServer/xml/tools/rfq`

Linux `/opt/WebSphere/CommerceServer/xml/tools/rfq`

Solaris `/opt/WebSphere/CommerceServer/xml/tools/rfq`

Windows *unidad*`:\WebSphere\CommerceServer\xml\tools\rfq`

El `finishURL` es **RFQResponseCopy**, que correlaciona el URL registrado. El panel de este diálogo es **RFQRspDuplicateDialog**, que correlaciona el mandato de vista registrado. El código del archivo XML es el siguiente:

```
<?xml version="1.0"?>

<dialog resourceBundle="utf.utfNLS"
   windowTitle="dupliateDialog_Title"
   finishURL="RFQResponseCopy">

   <panel name="general"
       url="/webapp/wcs/tools/servlet/RFQRspDuplicateDialog"
       parameters="responseId"
       hasFinish="YES"/>
   <jsFile src="/wcs/javascript/tools/rfq/rfq_response_duplicate_dialog.js"/>
</dialog>
```

d. Registre este XML en `resource.xml`.

1) Haga una copia de seguridad de `resource.xml` en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/xml/tools/rfq`

400 `/QIBM/UserData/WebSphere/CommerceServer/xml/tools/rfq`

Linux `/opt/WebSphere/CommerceServer/xml/tools/rfq`

Solaris `/opt/WebSphere/CommerceServer/xml/tools/rfq`

Windows *unidad*`:\WebSphere\CommerceServer\xml\tools\rfq`

2) Añada las dos líneas siguientes al archivo `resource.xml`:

```
<XML name="rfqRspDuplicateDialog"
 file="rfq/rfq_response_duplicate_dialog.xml"/>
```

e. Añada un archivo JavaScript nuevo denominado `rfq_response_duplicate_dialog.js` en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/web/javascript/tools/rfq`

400 `/QIBM/UserData/WebSphere/CommerceServer/web/javascript/tools/rfq`

Linux `/opt/WebSphere/CommerceServer/web/javascript/tools/rfq`

Solaris `/opt/WebSphere/CommerceServer/web/javascript/tools/rfq`

Windows *unidad*`:\WebSphere\CommerceServer\web\javascript\tools\rfq`

El archivo JavaScript contiene algunas funciones JavaScript utilizadas por el diálogo. A continuación se muestra el código fuente del archivo JavaScript:

```
function submitErrorHandler (errMessage){
   self.CONTENTS.alertDialog(errMessage);
}

function submitFinishHandler (finishMessage){
   alertDialog(finishMessage);
```

```
   top.goBack();
}

function submitCancelHandler(){
   top.goBack();
}
```

3. Integre esta función en la página de lista de respuestas a RFQ.
   a. Añada un botón **Duplicar** en la página de lista de respuestas. Es decir, cambie el archivo XML como se indica a continuación:

      Nota: Haga una copia de seguridad del archivo XML `rfq_response_list.xml`, que se encuentra en el directorio siguiente:

      AIX `/usr/WebSphere/CommerceServer/xml/tools/rfq`

      400 `/QIBM/UserData/WebSphere/CommerceServer/xml/tools/rfq`

      Linux `/opt/WebSphere/CommerceServer/xml/tools/rfq`

      Solaris `/opt/WebSphere/CommerceServer/xml/tools/rfq`

      Windows *unidad*`:\WebSphere\CommerceServer\xml\tools\rfq`

      Al final del nodo `<button>... </button>` de `rfq_response_list.xml`, añada la línea siguiente.

```
<menu name="duplicate"
   action="basefrm.duplicateRes()"
   selection="single">
</menu>
```

   b. Añada las funciones JavaScript nuevas `getDuplicateBCT()` y `duplicateRes()` en `rfq_response_list.jsp`:

```
function getDuplicateBCT() {
   return "<%= rfqNLS.get("duplicate") %>";
}

function duplicateRes(){
   if(isButtonDisabled(parent.buttons.buttonForm.duplicateButton))
   return;
   var checkedEntries = parent.getChecked().toString();
   var parms = checkedEntries.split(';');
   var resId = parms[0];
   top.setContent(getDuplicateBCT(),'/webapp/wcs/tools/servlet/DialogView?
          XMLFile=rfq.rfqRspDuplicateDialog&responseId='+resId,true)
}
```

   c. Añada una función JavaScript en `rfq_response_list.jsp` y llámela para inhabilitar el botón **Duplicar** cuando el estado de la RFQ correspondiente no sea ACTIVE. Añada la función siguiente en el `rfq_response_list.jsp` y llámela después de llamar a `DisableNewButton()`:

```
function DisableDuplicateButton()
{
   var reqState;
   var active=<%= com.ibm.commerce.utf.helper.UTFOtherConstants.EC_STATE_ACTIVE %>;
   reqState="<%=rfqState%>";
   if (reqState !=active){
      parent.hideButton('duplicate');
   }
}
```

4. Añada un mensaje nuevo en el archivo `RFQMessages_es_ES.properties` que se encuentra en el directorio siguiente:

   AIX

   `/usr/WebSphere/CommerceServer/properties/com/ibm/commerce/tools/rfq/properties`

   400

   `/QIBM/UserData/WebSphere/CommerceServer/properties/com/ibm/commerce/tools/rfq/properties`

   Linux

   `/opt/WebSphere/CommerceServer/properties/com/ibm/commerce/tools/rfq/properties`

Solaris

/opt/WebSphere/CommerceServer/properties/com/ibm/commerce/tools/rfq/properties

Windows

unidad:\WebSphere\CommerceServer\properties\com\ibm\commerce\tools\rfq\properties.

Añada la línea siguiente al archivo. Cuando la copia falla, se visualiza este mensaje:

```
_ERR_RESPONSE_COPY                = Ha fallado la copia de la respuesta.
```

Escenario 2: Suprimir el estado de Retractada del ciclo de vida de una respuesta a RFQ

Originalmente, cuando un usuario retracta una respuesta, la respuesta pasa del estado Activa al estado Retractada. Si, en lugar de ello, el usuario desea que la respuesta se establezca directamente al estado Borrador después de retractarla, deberá personalizar la función suprimiendo el estado Retractada del ciclo de vida de una respuesta.

Visión general de la implementación

En esta personalización, primero necesitamos suprimir el estado Retractada de la máquina de estado de respuesta. Una vez que ya no existe ningún estado Retractada, se deberá eliminar también el estado Retractada de la lista de vistas de la página de lista de respuestas.

Pasos para la personalización

1. Cambie los archivos UBFStatemachine.xml y UBFStatemachine_es_ES.xml y vuelva a cargarlos. Haga una copia de seguridad de UBFStatemachine.xml y UBFStatemachine_es_ES.xml en el directorio siguiente:

   AIX

   /usr/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   400

   /QIBM/UserData/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   Linux

   /opt/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   Solaris

   /opt/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   Windows

   unidad:\WebSphere\CommerceServer\xmlloadutility\businessfollows\xml

   Realice los cambios siguientes en esos dos archivos:

   - Cambie el estado de destino de la transición retractRFQResponse de RETRACTED (retractada) a DRAFT (borrador).
   - Suprima la información acerca del estado RETRACTED y todas las transiciones de este estado

   El código de ejemplo siguiente ilustra los cambios necesarios en el archivo UBFStatemachine.xml:

   Antes de los cambios

```
<Flow identifier="RFQ response process totally" priority="2">
   <State identifier="ACTIVE">
      <Transition eventidentifier="retractRFQResponse" approval="0" priority="2">
      <Action
          interface="com.ibm.commerce.rfq.commands.RFQResponseChangeStateAdvCmd">
      <AccessControlGuard actiongroup="RFQResponseManage">
      <TargetState identifier="RETRACTED"/>
```

```
      </Transition>
   </State>
   <State identifier="RETRACTED">
      <Transition eventidentifier="changeRFQResponseToDraft" approval="0" priority="1">
      <Action
          interface="com.ibm.commerce.rfq.commands.RFQResponseChangeStateAdvCmd">
      <AccessControlGuard actiongroup="RFQResponseManage">
      <TargetState identifier="DRAFT"/>
      </Transition>
      <Transition eventidentifier="cancelRFQResponse" approval="0" priority="2">
      <Action
          interface="com.ibm.commerce.rfq.commands.RFQResponseChangeStateAdvCmd">
      <AccessControlGuard actiongroup="RFQResponseManage">
      <TargetState identifier="CANCELLED"/>
      </Transition>
      <Transition eventidentifier="closeRFQRequest" approval="0" priority="3">
      <Action
          interface="com.ibm.commerce.rfq.commands.RFQResponseChangeStateBaseCmd">
      <AccessControlGuard actiongroup="RFQManage">
      <TargetState identifier="CANCELLED"/>
      </Transition>
      <Transition eventidentifier="cancelRFQRequest" approval="0" priority="4">
      <Action
          interface="com.ibm.commerce.rfq.commands.RFQResponseChangeStateBaseCmd">
      <AccessControlGuard actiongroup="RFQManage"/>
      <TargetState identifier="CANCELLED"/>
      </Transition>
   </State>
</Flow>
```

Después de los cambios

```
<Flow identifier="RFQ response process totally" priority="2">
   <State identifier="ACTIVE">
      <Transition eventidentifier="retractRFQResponse" approval="0" priority="2">
      <Action
          interface="com.ibm.commerce.rfq.commands.RFQResponseChangeStateAdvCmd">
      <AccessControlGuard actiongroup="RFQResponseManage"/>
      <TargetState identifier="DRAFT"/>
      </Transition>
   </State>
</Flow>
```

Los cambios en `UBFStatemachine.xml` se muestran del modo siguiente:

Antes de los cambios

```
<State identifier="ACTIVE">
   <TransitionDesc eventidentifier="closeRFQRequest"
      transitiondescription=
      "Change the state of response to In-evaluation when closing request"
      eventdescription="closeRFQRequest">
   <TargetState identifier="IN-EVALUATION"/>
   </TransitionDesc>
   <TransitionDesc eventidentifier="retractRFQResponse"
      transitiondescription="Retract the response"
      eventdescription="retractRFQResponse">
   <TargetState identifier="RETRACTED"/>
   </TransitionDesc>
   <TransitionDesc eventidentifier="cancelRFQRequest"
      transitiondescription="Cancel the response when cancelling the rfq"
      eventdescription="cancelRFQRequest">
   <TargetState identifier="CANCELLED"/>
   </TransitionDesc>
</State>
<State identifier="RETRACTED">
   <TransitionDesc eventidentifier="changeRFQResponseToDraft"
      transitiondescription="Change the retracted response to draft"
      eventdescription="changeRFQResponseToDraft">
   <TargetState identifier="DRAFT"/>
   </TransitionDesc>
   <TransitionDesc eventidentifier="cancelRFQResponse"
      transitiondescription="Cancel the response"
      eventdescription="cancelRFQResponse">
   <TargetState identifier="CANCELLED"/>
   </TransitionDesc>
   <TransitionDesc eventidentifier="closeRFQRequest"
      transitiondescription="Cancel the response when closing the rfq"
      eventdescription="closeRFQRequest">
   <TargetState identifier="CANCELLED"/>
   </TransitionDesc>
   <TransitionDesc eventidentifier="cancelRFQRequest"
      transitiondescription="Cancel the response when cancelling the rfq"
      eventdescription="cancelRFQRequest">
   <TargetState identifier="CANCELLED"/>
   </TransitionDesc>
</State>
```

Después de los cambios

```
<State identifier="ACTIVE">
   <TransitionDesc eventidentifier="closeRFQRequest"
      transitiondescription=
      "Change the state of response to In-evaluation when closing request"
      eventdescription="closeRFQRequest">
   <TargetState identifier="IN-EVALUATION"/>
   </TransitionDesc>
   <TransitionDesc eventidentifier="retractRFQResponse"
      transitiondescription="Retract the response"
```

```
            eventdescription="retractRFQResponse">
          <TargetState identifier="DRAFT"/>
          </TransitionDesc>
          <TransitionDesc eventidentifier="cancelRFQRequest"
            transitiondescription="Cancel the response when cancelling the rfq"
            eventdescription="cancelRFQRequest">
          <TargetState identifier="CANCELLED"/>
          </TransitionDesc>
        </State>
```

2. Puesto que ya no hay ningún estado `Retractada` en el ciclo de vida de una respuesta, suprima las líneas siguientes en `rfq_response_list.xml` y `rfq_response_state_list.xml`. Aquí los estados incluyen `draft`, `cancelled`, `pendingapproval`, `rejected`, `active`, `inevaluation`, `win`, `lost`, `wincomplete` y `lostcomplete`.

   Nota: Aquí la línea se ha partido para poderla mostrar.

```
<view name="resretracted"
   action="top.setContent(basefrm.getPageTitle(),
   '/webapp/wcs/tools/servlet/NewDynamicListView?
   ActionXMLFile=rfq.rfqresponseretractedlist&
   cmd=RFQResponseList&rfqId='+basefrm.getRfqId(),false)"/>
```

Notas:

1. Este ejemplo ilustra cómo suprimir un estado. Si, en lugar de ello, desea añadir un estado, realice pasos similares en orden inverso:
   a. Añada las transiciones de estado nuevas en el archivo XML de máquina de estado y vuelva a cargarlo.
   b. Añada la vista nueva en la lista de vistas de la página de lista de respuestas.

   Para obtener información sobre cómo volver a cargar la máquina de estado, consulte la ayuda en línea de UBF.

Escenario 3: Entrar información descriptiva para una respuesta durante la creación

Si el usuario desea entrar texto descriptivo al crear una respuesta a una RFQ, deberá añadir un campo de texto en el primer panel del asistente de creación de RFQ.

Visión general de la implementación

Añada un campo de texto nuevo en el primer panel del asistente de creación de respuesta a RFQ. Dado que **RFQResponseCreateCmdImpl** no maneja la entrada adicional, se creará una interfaz nueva **MyRFQResponseCreateCmd** y la implementación de la misma, **MyRFQResponseCmdImpl**, mediante la ampliación de **RFQResponseCreateCmd** y **RFQResponseCreateCmdImpl**.

Pasos para la personalización

1. Añada una interfaz **MyRFQResponseCreateCmd** y su implementación **MyRFQResponseCreateCmdImpl**.
   a. Añada una interfaz **MyRFQResponseCreateCmd** nueva. El código de la interfaz es el siguiente:

```
public interface MyRFQResponseCreateCmd extends RFQResponseCreateCmd {
   public final static String NAME =
       "com.ibm.commerce.rfq.commands.MyRFQResponseCreateCmd";
   public static final String COPYRIGHT =
       com.ibm.commerce.copyright.IBMCopyright.SHORT_COPYRIGHT;
   public static String defaultCommandClassName =
       "com.ibm.commerce.rfq.commands.MyRFQResponseCreateCmdImpl";
   }
```

   b. Añada la implementación de mandato **MyRFQResponseCreateCmdImpl** ampliando **RFQResponseCreateCmdImpl** e implementando **MyFQResponseCreateCmd**. Añada un campo nuevo `responseDescription`

para almacenar la descripción de respuesta transferida desde la interfaz de usuario. Asimismo, añada un método getter y un método setter para este campo:

```
protected String responseDescription ="";
public java.lang.String getResponseDescription() {
   return responseDescription;
}
public void setResponseDescription(String name, boolean isReq) throws ECApplicationException {
   try {
      responseDescription = (String)getToolXMLObject().get(name);
   } catch (Exception e) { }

   if(isReq) {
      if (getResponseDescription()==null || getResponseDescription().length()==0) {
         setErrorFlag(true);

      getErrorContent().put(ECRFQMessageKey._ERR_RFQ_MISSING_RESPONSEREMARKS,"");
      }
   }
}
```

Altere temporalmente el método `initParameters()`. Este método lo invocará `checkParameters()` en **RFQResponseCreateCmdImp** para inicializar los parámetros. La diferencia entre este método y el método de la superclase correspondiente consiste en que la invocación de **setResponseDescription (MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION, false)** hace que la descripción de respuesta se transfiera desde la interfaz de usuario. El código fuente del método es el siguiente:

```
protected void initParameters()
throws com.ibm.commerce.exception.ECApplicationException
{
   setRequestId(RFQConstants.EC_RFQ_REQUEST_ID, true);
   setResponseName(RFQConstants.EC_RFQ_RESPONSE_NAME, true);
   setResponseRemarks(RFQConstants.EC_RFQ_RESPONSE_REMARK, false);
   setCommentsRFQLevelList(RFQConstants.EC_TC_RFQ_LEVEL_COMMENTS,false);
   setResProductsList(RFQConstants.EC_OFFERING_PRODITEM, false);

   //obtener descripción de respuesta de la interfaz de usuario
   setResponseDescription(MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION, false);
}
```

Altere temporalmente el método `createResponse()`. Este método es idéntico, excepto en que debe establecer el campo de descripción en **RFQResponseAccessBean** y **TradingDescriptionAccessBean**. A continuación se proporciona este código fuente del método:

```
protected RFQResponseAccessBean createResponse ()
throws ECApplicationException,ECException {
   RFQResponseDataBean responseDB = null;
   try{
      CreateResponseBasicInfoCmd createResCmd = null;
      createResCmd = (CreateResponseBasicInfoCmd)
        CommandFactory.createCommand(CreateResponseBasicInfoCmd.NAME, getStoreId());
      createResCmd.setRequestId(getRequestId());
      createResCmd.setOwnerId(getOwnerId());
      createResCmd.setResponseName(getResponseName());
      createResCmd.setResponseRemarks(getResponseRemarks());
      createResCmd.setStateIdentifier(getStateIdentifier());
      createResCmd.setCommandContext(getCommandContext());

      createResCmd.execute();

      responseDB=createResCmd.getResponseDataBean();
      //Este id lo tomará BusinessFlowEventListener
      //para añadir un registro en la tabla FLINSTANCE

      this.setEntityObject(responseDB);

      //establecer este número de referencia de respuesta
      setResponseId(responseDB.getRfqResponseIdInEJBType());

      responseDB.setDescription(responseDescription);
      responseDB.commitCopyHelper();
      TradingDescriptionAccessBean trdDesc = new TradingDescriptionAccessBean();
      trdDesc.setInitKey_languageId(getCommandContext().getLanguageId().toString());
      trdDesc.setInitKey_tradingId(responseDB.getRfqResponseId());
      trdDesc.refreshCopyHelper();
      if (getResponseDescription()==null || getResponseDescription().length()==0) {
         trdDesc.setShortDescription(responseDB.getName());
      }
      trdDesc.setShortDescription(getResponseDescription());
      trdDesc.setTimeCreated(TimestampHelper.systemCurrentTimestamp());
      trdDesc.commitCopyHelper();
   }catch (javax.ejb.CreateException e) {
```

```
            throw new ECApplicationException(ECRFQMessage._ERR_RESPONSE_BASICINFO_SAVE,
                this.getClass().getName(), "performExecute");
        }catch (javax.naming.NamingException e) {
            throw new ECSystemException(ECMessage._ERR_NAMING_EXCEPTION,
                this.getClass().getName(), "createResponse");
        }catch (java.rmi.RemoteException e) {
            throw new ECSystemException(ECMessage._ERR_REMOTE_EXCEPTION,
                this.getClass().getName(), "createResponse");
        }catch (javax.ejb.FinderException e) {
            throw new ECSystemException(ECMessage._ERR_FINDER_EXCEPTION,
                this.getClass().getName(), "createResponse");
        }
        return responseDB;
    }
```

c. Ejecute la sentencia de SQL siguiente para registrar el mandato **MyRFQReponseCreate** en la tabla URLREG:

```
insert into urlreg values('MyRFQResponseCreate', 0,
   'com.ibm.commerce.ubf.commands.ToolsBusinessFlowEventCmd', 0,
   null, null, 1);
```

2. Cree una clase de constantes nueva `MyRFQConstants` ampliando `RFQConstants`. Sólo hay una nueva definición de constante:

```
public final static String EC_RFQ_RESPONSE_DESCRIPTION = "response_description";
```

3. Cambie el archivo `rfq_w_response_general.jsp` para añadir un campo de texto nuevo y el proceso asociado. Localice el archivo en el directorio siguiente:

AIX `/usr/WebSphere/CommerceServer/web/tools/rfq`

400 `/QIBM/UserData/WebSphere/CommerceServer/web/tools/rfq`

Linux `/opt/WebSphere/CommerceServer/web/tools/rfq`

Solaris `/opt/WebSphere/CommerceServer/web/tools/rfq`

Windows *unidad*`:\WebSphere\CommerceServer/web/tools/rfq`

Cambie la sección `Form` del modo siguiente:

```
<FORM name="rfqCreateForm">
<table COLS=3 WIDTH="60%">
<tr>
<td><%= rfqNLS.get("name") %></td>
</tr>
<tr>
<td><INPUT name="response_name" maxlength=100></td>
</tr>
<tr>
<td><%= rfqNLS.get("remark") %></td>
</tr>
<tr>
<td><TEXTAREA rows="4" cols="40" name="response_remark"></TEXTAREA></TD>
</tr>
<tr>
<td><%= rfqNLS.get("desc") %></td>
</tr>
<tr>
<td><TEXTAREA rows="4" cols="40"  name="response_description"></TEXTAREA></TD>
</TR>
</table>
</FORM>
```

En la función `savePanelData()`, añada código para guardar la descripción de respuesta. La función cambiada es la siguiente:

```
function savePanelData(){
   VPDResult = validatePanelData0();
   if(!VPDResult)
      return;
   if (isFirstTimeLogonWizard== "1")
   parent.put("<%=RFQConstants.EC_RFQ_REQUEST_ID%>", getRequestId());
   parent.put("<%=RFQConstants.EC_RFQ_RESPONSE_NAME%>",
       document.rfqCreateForm.response_name.value);
   parent.put("<%=RFQConstants.EC_RFQ_RESPONSE_REMARK%>",
       document.rfqCreateForm.response_remark.value);
   parent.put("<%=BusinessFlowConstants.EC_FLOWID%>",
       "<%=RFQConstants.EC_FLOW_RESPONSE_ID%>");
   parent.put("<%=BusinessFlowConstants.EC_BUSINESS_FLOW_EVENT_IDENTIFIER%>",
       "createRFQResponse");
   parent.put("<%=MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION%>",
       document.rfqCreateForm.response_description.value);
}
```

En la función `retrievePanelData()`, añada código para recuperar la descripción de respuesta. El código cambiado es el siguiente:

```
function retrievePanelData(){
   var form = document.rfqCreateForm;
   form.response_name.value = parent.get("<%=RFQConstants.EC_RFQ_RESPONSE_NAME%>","");
   form.response_remark.value = parent.get("<%=RFQConstants.EC_RFQ_RESPONSE_REMARK%>","");
   form.response_description.value = parent.get("<%=MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION%>","");\
}
```

4. Cambie el finishURL destino en el archivo rfq_response_wizard.xml de RFQResponseCreate a MyRFQResponseCreate. Localice el archivo en el directorio siguiente:

   AIX /usr/WebSphere/CommerceServer/xml/tools/rfq

   400 /QIBM/UserData/WebSphere/CommerceServer/xml/tools/rfq

   Linux /opt/WebSphere/CommerceServer/xml/tools/rfq

   Solaris /opt/WebSphere/CommerceServer/xml/tools/rfq

   Windows *unidad*:\WebSphere\CommerceServer/xml/tools/rfq

   Cámbielo de acuerdo con el segmento de código siguiente.

```
<wizard resourceBundle="utf.utfNLS"
   windowTitle="matchlisttitle"
   finishConfirmation="ex_finish"
   cancelConfirmation="ex_cancel"
   finishURL="MyRFQResponseCreate"
   tocBackgroundImage="/wcs/images/tools/toc/W_merchand.jpg">
```

5. Cambie el archivo UBFStatemachine.xml y vuelva a cargarlo. Localice el archivo en el directorio siguiente:

   AIX

   /usr/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   400 /QIBM/UserData/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   Linux

   /opt/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   Solaris

   /opt/WebSphere/CommerceServer/xmlloadutility/businessfollows/xml

   Windows

   unidad:\WebSphere\CommerceServer\xmlloadutility\businessfollows\xml

   Cambie la interfaz de acción com.ibm.commerce.rfq.commands.RFQResponseCreateCmd por com.ibm.commerce.rfq.commands.RFQResponseCreateCmd en ambos archivos. El código de ejemplo siguiente ilustra los cambios necesarios:

   Antes de los cambios

```
<Flow identifier="RFQ response process totally" priority="2">
   <StartState identifier="START">
      <Transition eventidentifier="createRFQResponse" approval="0" priority="1">
         <Action interface="com.ibm.commerce.rfq.commands.RFQResponseCreateCmd"/>
         <AccessControlGuard actiongroup="RFQResponseCreate"/>
         <TargetState identifier="DRAFT"/>
      </Transition>
   </StartState>
```

   Después de los cambios

```
<Flow identifier="RFQ response process totally" priority="2">
   <StartState identifier="START">
      <Transition eventidentifier="createRFQResponse" approval="0" priority="1">
      <Action interface="com.ibm.commerce.rfq.commands.RFQResponseCreateCmd"/>
      <AccessControlGuard actiongroup="RFQResponseCreate"/>
      <TargetState identifier="DRAFT">
      </Transition>
   </StartState>
```

   Nota: Para obtener información sobre cómo volver a cargar la máquina de estado, consulte la ayuda en línea de UBF.

Capítulo 10. Inventario esperado

Ejemplos de personalización

Los ejemplos siguientes describen cómo personalizar esta parte de WebSphere Commerce Accelerator.

Escenario 1: Añadir información nueva al crear Registros de inventario esperado

Este escenario le guía durante los pasos necesarios para personalizar las herramientas de Inventario esperado a fin de registrar información adicional en el Inventario esperado, por ejemplo quién ha creado el Registro de inventario esperado.

1. Añada una columna en la tabla `RA` para el `userID` (id de usuario) apropiado. Cree un bean enterprise `ExpectedInventoryRecords` nuevo y un bean de acceso.
2. Implemente un **CustomExpectedInventoryRecordCreateCmdImpl** nuevo que amplíe la implementación de mandato de controlador **ExpectedInventoryRecordCreateCmdImp** e implemente **ExpectedInventoryRecordCreateCmd**.
3. Registre el nuevo mandato de controlador en `CMDREG`. Suponga que el ID de tienda de cliente es 1 y, a continuación, ejecute la sentencia de SQL siguiente:

```
insert into cmdreg(storeent_id,interfacename,classname)
   values(1,'com.ibm.commerce.tools.inventory.ExpectedInventoryRecordCreateCmd ',
   'com.ibm.commerce.tools.inventory.CustomExpectedInventoryRecordCreateCmdImpl ')
```

4. Actualice **CustomExpectedInventoryRecordCreateCmdImpl**. En el mandato, implemente el método `performExecute()`:

```
public void performExecute()throws ECSystemException,ECException {
   super.performExecute();
   /**Obtener el ID de usuario del contexto de mandato
   Almacenar el ID de usuario en la tabla RA
   **/
}
```

Capítulo 11. Business intelligence

Contexto dinámico

El contexto dinámico es una lista de acciones agrupadas que los usuarios pueden elegir llevar a cabo. La lista contiene los nombres y descripciones breves de las acciones. La lista cambia de forma dinámica basándose en los roles del usuario y en los componentes que se habilitan.

Ejemplos de personalización

Escenario: Añadir una acción nueva en un contexto existente

Para añadir una acción nueva en un contexto existente, realice lo siguiente:

1. Añada una acción al archivo XML de definición de contexto. Por ejemplo, el archivo `biContext.xml` del directorio `xml/tools/bi`. Necesita buscar el contexto en el que desea realizar la adición y, a continuación, añadir una entrada al mismo. A continuación se muestra la acción Campaña del contexto Campaña:

```
<entry nameKey="campaign" descriptionKey="campaignDescription"
      breadCrumbTrailTextKey="Report" toolsComponent="CommerceAnalyzer">
   <roles>
      <role>siteOwner</role>
      <role>siteAdmin</role>
      <role>seller</role>
      <role>merchant</role>
      <role>makMgr</role>
      <role>podMgr</role>
   </roles>
   <command name = "BIShowReport">
      <parameter name="reportId"  value="BICampaignsIndex.html" />
   </command>
</entry>
```

   Para ver el contexto Campaña, seleccione **Campañas** en el menú **Marketing** de WebSphere Commerce Accelerator y, a continuación, pulse **Informes**.

   En el fragmento de código anterior, `toolsComponent` es un atributo opcional. Si se especifica, la acción sólo se listará cuando el componente especificado se habilite desde el Gestor de configuración.

   Si el usuario es uno de los roles que se definen en el elemento roles, se listará la acción para el usuario.

   El atributo de nombre (name) del elemento de mandato (command) es el nombre del mandato realizado cuando se selecciona la acción en la lista. Los atributos de parámetro (parameter) y valor (value) son los parámetros de mandato y los valores de los parámetros.

   El atributo `appendQueryString` del elemento de entrada (entry) no se muestra en el fragmento de código anterior. Si el atributo está presente y está establecido en verdadero (true), las propiedades solicitadas de la petición actual se añadirán al mandato de acción en forma de parejas de nombre y valor. Por ejemplo, el código siguiente ilustra la acción Informe de pedidos por cuenta en el contexto de informe de cuenta:

```
entry nameKey="ordersByAccount" descriptionKey="ordersByAccountDescription"
     breadCrumbTrailTextKey="reportCriteria" appendQueryString="true">
   <roles>
      <role>siteOwner</role>
      <role>siteAdmin</role>
      <role>seller</role>
      <role>actRep</role>
      <role>salesMgr</role>
   </roles>
   <command name = "OrdersByAccountDialogView">
      <parameter name="XMLFile" value="reporting.OrdersByAccountReportDialog"/>
   </command>
</entry>
```

La clase `ContextEntry` genera el mandato siguiente basándose en la definición anterior (el mandato es una sola línea, que aquí se ha partido para su presentación):

```
/webapp/wcs/tools/servlet/OrdersByAccountDialogView
   ?contextConfigXML=contract.brmReportContext
   &startindex=0&ActionXMLFile=contract.rptAccountContextList
   &accountId=10001&docname=tools/bi/ContextList.jsp&resultsize=0
   &storeId=135&context=account&langId=-1
   &XMLFile=reporting.OrdersByAccountReportDialog&listsize=15
```

En el mandato anterior, las parejas de nombre y valor siguientes se extraen de las propiedades de petición y se añaden al mandato: contextConfigXML=contract.brmReportContext &startindex=0 &ActionXMLFile=contract.rptAccountContextList &accountId=10001&docname=tools/bi/ContextList.jsp &resultsize=0&storeId=135 &context=account&langId=-1&listsize=15.

2. Añada las claves de propiedades al archivo de propiedades.

   Para la habilitación del idioma nacional, `nameKey`, `descriptionKey` y `breadCrumbTrailTextKey` son las claves del archivo de propiedades. El archivo de propiedades del ejemplo anterior es `properties/com/ibm/commerce/tools/bi/properties/BINLS_es_ES.properties`. `nameKey` se correlaciona con el nombre de la acción. `descriptionKey` se correlaciona con la descripción de las acciones. `breadCrumbTrailTextKey` se correlaciona con el texto añadido en el seguimiento detallado cuando se selecciona la acción.

Capítulo 12. Visión general de la infraestructura de informes

La infraestructura de informes proporciona una funcionalidad de informes genérica personalizable para casi cualquier aspecto del sitio. Cualquiera de los roles que utilizan Commerce Accelerator puede acceder a los informes. El acceso a un informe determinado puede definirse y limitarse dentro del informe. Los usuarios de WebSphere Commerce Accelerator pueden solicitar los informes en cualquier momento y la infraestructura genera el informe utilizando los datos contenidos en la base de datos de producción y visualiza el informe en tiempo real.

La infraestructura consta de un mandato de controlador genérico, un bean de datos y una vista genérica que visualiza el resultado. Puede personalizar la infraestructura añadiendo consultas de SQL válidas y definiendo archivos JSP utilizados para solicitar y visualizar los informes generados.

Personalización de la infraestructura de informes

Desde Commerce Accelerator se puede acceder a los informes. En consecuencia, cada informe necesita diversos elementos asociados. Mientras que el informe propiamente dicho consta de datos que se representan en formato tabular, los elementos subyacentes constan del identificador de informe, de una consulta de SQL, de elementos de control de acceso, etc. La petición de informe inicia un mandato de controlador en el servidor. El mandato de controlador llama a las tareas para establecer la vista genérica, a no ser que el informe especifique una vista determinada. El mandato también establece diversas variables y devuelve estos datos para llenar el bean de datos de informe en el archivo JSP de destino. El control de acceso para los informes se establece en las vistas que solicitan (entran) y visualizan el informe. Los resultados devueltos de la base de datos se almacenan en el bean de datos como un vector de tabla de totales de control. Finalmente, el archivo JSP visualiza el informe. Si el informe está vacío, en lugar de ello el archivo JSP visualiza texto genérico.

El control de acceso para los informes se establece en las vistas para solicitar (entrar) y producir el informe.

Para añadir un informe nuevo es necesario realizar los pasos siguientes:

1. Definir el informe en un archivo XML.
2. Crear el archivo JSP a partir del cual se solicita el informe, si es necesario.
3. Crear el archivo JSP para visualizar el informe, a no ser que se utilice el JSP genérico.

Ejemplos de personalización

Definición del informe en un archivo XML

Los informes individuales se definen utilizando archivos XML. Cada informe tiene un archivo *nombreInforme*.xml correspondiente. Este archivo contiene toda la información necesaria para generar un informe. El archivo *nombreinforme*.xml tiene un aspecto similar al ejemplo siguiente para MyStoreOverviewReport:

```
<?xml version="1.0" standalone="yes" ?>
<Reporting>
<!-- owner="nombrePropietario" location="vía_acceso_a_este_archivo_XML " -->
<!-- Una colección consta de SQL para informes de WCS  -->

<Report reportName="MyStoreOverviewReport" online="true">
```

```
<comment>store_overview, yesterday, all measurements</comment>
<SQLvalue>
</SQLvalue>
<mergeOperation>1000000,1000001</mergeOperation>
<display>
<standardInfo>
<resource>reporting.ReportingString</resource>
<title></title>
<message>messageMyReport</message>
<columnTitles>CRITERIA,KEY,VALUE,CURRENCY,DATESTMP</columnTitles>
</standardInfo>
<userDefinedParameters>
</userDefinedParameters>
</display>
</Report>
<Report reportName="1000000" online="true">
<comment>store_overview, yesterday, revenue</comment>
<SQLvalue>
      select {revenue} as criteria, storeent_id as key,
             sum(totalproduct+totalshipping+totaltax+totaltaxshipping) as value,
             currency as currency, 0 as datestmp
       from orders
       where $DB_DATE_GREATER_EQUAL_FUNC(lastupdate,{beginDate})$  and
             $DB_DATE_LESS_EQUAL_FUNC(lastupdate,{endDate})$
             and status in ('C','M','S') and storeent_id={storeent_id}
       group by storeent_id, currency
</SQLvalue>
<display>
<standardInfo>
<resource>reporting.ReportingString</resource>
<title></title>
<message>message1000000</message>
<columnTitles>CRITERIA,KEY,VALUE,CURRENCY,DATESTMP</columnTitles>
</standardInfo>
<userDefinedParameters>
</userDefinedParameters>
</display>
</Report>
<Report reportName="1000001" online="true">
   <comment>store_overview, yesterday, number of orders</comment>
   <SQLvalue>
       select {orders} as criteria, storeent_id as key, count(*) as value,
              '-' as currency,
              0 as datestmp
       from orders
       where $DB_DATE_GREATER_EQUAL_FUNC(lastupdate,{beginDate})$  and
             $DB_DATE_LESS_EQUAL_FUNC(lastupdate,{endDate})$
             and status in ('C','M','S') and storeent_id={storeent_id}
       group by storeent_id
</SQLvalue>
<display>
     <standardInfo>
<resource>reporting.ReportingString</resource>
<title></title>
<message>message1000000</message>
<columnTitles>CRITERIA,KEY,VALUE,CURRENCY,DATESTMP</columnTitles>
</standardInfo>
<userDefinedParameters>
</userDefinedParameters>
</display>
</Report>
</Reporting>
```

Para crear un informe nuevo, deberá crear un archivo XML similar al ejemplo anterior. Para obtener una explicación detallada de cada elemento XML, consulte el apartado siguiente titulado “Elementos XML válidos”.

Elementos XML válidos: Defina informes utilizando los elementos XML siguientes:

<Reporting></Reporting>
Es el elemento raíz.

<Report></Report>
Este elemento necesario define un informe determinado. Puede definir varios informes en un solo archivo XML incluyendo más de un elemento <Report>. Este elemento tiene dos atributos necesarios:

ReportName
Serie que define el nombre exclusivo del informe.

online Valor booleano que especifica si el informe está disponible en tiempo real. Actualmente, el único valor soportado es `true`.

<comment></comment>
Elemento opcional en el que se puede describir el informe. Esta serie no necesita traducirse. Este elemento sólo se puede definir en un elemento `<Report>` existente.

<SQLvalue></SQLvalue>
Elemento necesario que define la consulta de SQL utilizada para generar el informe. Aunque es necesario, este elemento puede estar vacío. Este elemento sólo se puede definir en un elemento `<Report>` existente.

<mergeOperation></mergeOperation>
Elemento opcional que le permite combinar varias consultas de SQL en un informe. Contiene una lista de nombres de informe delimitados por coma que apuntan a los atributos reportName de otros elementos `<Report>`. Todas las consultas de SQL de los informes a los que se hace referencia deben devolver el mismo número de columnas y utilizar los mismos nombres de columna que identifican las claves de la tabla de totales de control. Cada consulta de SQL es independiente de las demás consultas. Cada consulta de SQL produce su propio vector de tabla de totales de control y, antes de la presentación, estos vectores de añaden unos a otros para formar un solo informe.

El ejemplo de informe de visión general de la tienda del apartado anterior muestra el uso del elemento `<mergeOperation>`. La primera fila del informe final procede de una consulta de SQL y la segunda procede de una consulta subsiguiente. Este elemento sólo se puede definir en un elemento `<Report>` existente.

<extended_object_class></extended_object_class>
Elemento opcional que contiene un nombre de clase Java utilizado para crear una sentencia de SQL usando una clase Java ampliada. La clase genera un informe y, a continuación, vuelve a enviar los datos al centro de control de informes. Utilice este elemento para crear un informe repetitivo. Este elemento sólo se puede definir en un elemento `<Report>` existente

<display></display>
Elemento opcional utilizado para definir los parámetros usados en la visualización del resultado. Este elemento sólo se puede definir en un elemento `<Report>` existente.

<standardInfo></standardInfo>
Elemento necesario utilizado para agrupar elementos a los que se puede acceder mediante métodos getter en el bean de datos de informe. Estos elementos son elementos básicos para la mayoría de los informes. Este elemento sólo se puede definir en un elemento `<display>` existente.

<resourceBundle></resourceBundle>
Elemento necesario utilizado para especificar el archivo de propiedades que se debe utilizar para el informe. También se debe hacer referencia al valor en el archivo `reports/resources.xml`. Este elemento sólo se puede definir en un elemento `<standardInfo>` existente.

<title></title>
Elemento necesario utilizado para especificar el título del informe. El valor debe ser una clave del archivo de propiedades. Este elemento sólo se puede definir en un elemento `<standardInfo>` existente.

<message></message>
Elemento necesario utilizado para visualizar un mensaje relacionado con el informe. Por ejemplo, se puede utilizar para proporcionar una descripción para el informe. El valor debe ser una clave del archivo de propiedades. Este elemento sólo se puede definir en un elemento `<standardInfo>` existente.

<columnTitles></columnTitles>
Elemento necesario utilizado para definir los títulos de columna. Contiene una lista de nombres delimitados por coma. Los nombres deben ser claves definidas en el archivo de propiedades. Si no se puede encontrar la clave en el archivo de propiedades, se utiliza la clave proporcionada en el elemento como título de columna. Este elemento sólo se puede definir en un elemento `<standardInfo>` existente.

<userDefinedParameters></userDefinedParameters>
Elemento opcional utilizado para definir elementos personalizados en la infraestructura de informes. La infraestructura de informes espera ver elementos con el formato:

```
<elemento1>valor1</elemento1>
<elemento2>valor2</elemento2>
```

El bean de datos de informe proporciona un método getter que devuelve una tabla de totales de control de los elementos anteriores que se deberán utilizar en el JSP de visualización personalizado. Este elemento sólo se puede definir en un elemento `<display>` existente.

Nota: Mientras que los elementos contenidos en el elemento `<display>` se listan según sean necesarios, esto sólo sucede si está definido el elemento de visualización opcional.

Variables de las consultas de SQL: Cuando se definen variables en el archivo *reportName*`.xml`, las variables deben escribirse entre llaves ({*variableName*}). Esto indica a la infraestructura de informes que el valor es una variable de cliente y que debe obtenerse de la tabla de totales de control de cliente. En el archivo XML mostrado anteriormente, `{revenue}`, `{beginDate}`, `{endDate}`, `{storeent_id}`y `{orders}` son todas ellas variables de cliente.

Crear el archivo JSP a partir del cual se solicita el informe

En función de la cantidad de información necesaria para generar el informe, deberá decidir si va a utilizar un diálogo o un Asistente para reunir los datos necesarios. Sea cual sea el elemento apropiado, el JSP deberá incluir la función JavaScript `savePanelData`:

```
function savePanelData()
{

var reportInputData = new Object();

   reportInputData.SQLid = "el nombre del informe solicitado" ;
reportInputData.reportXML = "algún archivo";
   reportInputData.variable1 = "algún valor 1";
   reportInputData.variable2 = "algún valor 2" ;
.
   .
   .
reportInputData.variableN = "algún valor N";
   reportInputData.varProperties = "una lista de variables separadas por coma";
   parent.put("reportInputData", reportInputData);

// La sección siguiente puede utilizarse para indicar una Vista diferente a utilizar
// var reportResultPage = new Object();
// reportResultPage.cmd = "ASpecificDisplayReportView";
```

```
// parent.put("reportResultPage",reportResultPage);

   return true;
}
```

Las referencias a reportInputData y reportResultPage son necesarias para pasar los parámetros al mandato de controlador. Las variables SQLid y reportXML también son necesarias. variable1 a variableN y varProperties son opcionales. En el ejemplo variable1 a variableN representan las variables utilizadas en la consulta de SQL. Por ejemplo variable1 y variable2 pueden sustituirse por beginDate y endDate. De este modo, en la función savePanelData() habrá el código siguiente:

```
reportInputData.begindate = " algún valor";
reportInputData.enddate   = " algún valor";
```

La variable varProperties lista variables que obtienen sus valores de un archivo de propiedades. Por ejemplo, puede tener un aspecto similar a lo siguiente:

```
reportInputData.varProperties = "revenue,orders,pages,customers,visits";
```

Si en el archivo JSP no se hace referencia al objeto reportResultPage, el mandato de controlador lo establece de forma que utilice la vista genérica proporcionada por la infraestructura de informes para visualizar el informe. Si establece reportResultPage.cmd, tendrá la posibilidad de especificar qué vista se debe utilizar.

El código siguiente muestra un archivo JSP de ejemplo utilizado para reunir datos de entrada para un informe:

```
<!-- ========================================================================
  Materiales bajo licencia - Propiedad de IBM

  5724-A18

  (c) Copyright IBM Corp. 2001

  Derechos restringidos de los usuarios del Gobierno de EE.UU. - Utilización, duplicación
  o divulgación restringida por el GSA ADP Schedule Contract con IBM Corp.
 ------------------------------------------------------------------------------
 OrderSummaryReportInputView.jsp
 ========================================================================-->
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
<%@page import="java.util.*" %>
<%@page import="com.ibm.commerce.tools.util.*" %>
<%@page import="com.ibm.commerce.tools.xml.*" %>

<%@include file="common.jsp" %>
<%@include file="ReportStartDateEndDateHelper.jsp" %>
<%@include file="ReportFrameworkHelper.jsp" %>

<HTML>
<HEAD>
   <%=fHeader%>

   <TITLE><%=reportsRB.get("OrderSummaryReportInputViewTitle")%></TITLE>

   <SCRIPT SRC="/wcs/javaScript/tools/common/Util.js"></SCRIPT>
   <SCRIPT SRC="/wcs/javaScript/tools/common/DateUtil.js"></SCRIPT>
   <SCRIPT SRC="/wcs/javaScript/tools/common/SwapList.js"></SCRIPT>
   <SCRIPT SRC="/wcs/javascript/tools/reporting/ReportHelpers.js"></SCRIPT>

   <SCRIPT>

      //////////////////////////////////////////////////////////////////
      // Llamar a las rutinas de inicialización para los diversos elementos de la página
      //////////////////////////////////////////////////////////////////
      function initializeValues()
      {
         onLoadStartDateEndDate("enquiryPeriod");
         if (parent.setContentFrameLoaded) parent.setContentFrameLoaded(true);
      }

      //////////////////////////////////////////////////////////////////
      // Llamar a las rutinas para guardar para los diversos elementos de la página
      //////////////////////////////////////////////////////////////////
      function savePanelData()
      {
         saveStartDateEndDate("enquiryPeriod");
```

```
        ////////////////////////////////////////////////////////////////
        // Especificar los detalles de la infraestructura de informe
        ////////////////////////////////////////////////////////////////
        setReportFrameworkOutputView("DialogView");
        setReportFrameworkParameter("XMLFile","reporting.OrderSummaryReportOutputDialog");
        setReportFrameworkReportXML("reporting.OrderSummaryReport");
        setReportFrameworkReportName("OrderSummaryReport");

        ////////////////////////////////////////////////////////////////
        // Especificar los parámetros específicos de informe y guardarlos
        ////////////////////////////////////////////////////////////////
        setReportFrameworkParameter("StartDate", returnStartDateAsJavaTimestamp("enquiryPeriod"));
        setReportFrameworkParameter("EndDate", returnEndDateAsJavaTimestamp("enquiryPeriod"));
        saveReportFramework();
        return true;
     }

     ////////////////////////////////////////////////////////////////
     // Llamar a las rutinas de validación para los diversos elementos de la página
     ////////////////////////////////////////////////////////////////
     function validatePanelData()
     {
        if (validateStartDateEndDate("enquiryPeriod") == false) return false;
        return true;
     }

</SCRIPT>
</HEAD>

<BODY ONLOAD="initializeValues()" CLASS=content>

   <H1><%=reportsRB.get("OrderSummaryReportInputViewTitle") %></H1>
   <i><%=reportsRB.get("OrderSummaryReportDescription")%></i>
   <p>

   <DIV ID=pageBody STYLE="display: block; margin-left: 20">
      <%=generateStartDateEndDate("enquiryPeriod", reportsRB, null)%>
   </DIV>

</BODY>
</HTML>
```

Mandatos de infraestructura de informes

La infraestructura de informes utiliza los mandatos siguientes enviados con WebSphere Commerce:

Mandatos de vista

Tabla 9. Mandatos de vista utilizados por la infraestructura de informes

Nombre de vista	Archivo JSP
ReportRedirectView	\tools\reporting\ReportRedirect.jsp
ReportGenericView	\tools\reporting\ReportGenericView.jsp

Mandatos de controlador

Tabla 10. Mandatos de controlador utilizados por la infraestructura de informes

URL	Interfaz
GenericReportController	com.ibm.commerce.tools.reporting.command.GenericReportControllerCmd

Para obtener más información, consulte en la ayuda de JavaDoc que se envía con WebSphere Commerce los paquetes siguientes:

- com.ibm.commerce.tools.reporting.commands
- com.ibm.commerce.tools.reporting.framework
- com.ibm.commerce.tools.reporting.reports
- com.ibm.commerce.tools.reporting.util

Modelo de objeto de infraestructura de informes

Deberá utilizar el bean de datos de informe al crear el archivo JSP de resultados. El método `populate()` contiene la lógica para soportar informes en tiempo real. En el bean de datos están disponibles los métodos siguientes.

Tabla 11.

Nombre de método	Tipo de retorno	Descripción
populate()	void	ejecuta la consulta de SQL para generar el informe
getErrorCode()	int	método getter para el código de error
getNumberOfColumns()	int	método getter para el número de columnas del informe
getNumberOfRows()	int	método getter para el número de filas del informe
getColumnTitlesName(int)	string	método getter para el nombre de columna (i+1)
getRow(i)	hashtable	método getter para la fila (i+1)
getValue(i.j)	string	getter para el valor (i+1,j+1) del informe
getValue(i,keyname)	string	método getter para la fila (i+1) de la columna asociada con el nombre de clave
getUserDefinedParameters()	hashtable	método getter para la tabla de totales de control creada a partir de parámetros definidos por el usuario de *reportName*.xml
getEnv()	hashtable	método getter para la tabla de totales de control que contiene los parámetros de entrada definidos en el archivo JSP de entrada

Para obtener más información, consulte la ayuda de JavaDoc que se envía con WebSphere Commerce.

Componentes reutilizables para archivos JSP de informes

Los archivos JSP de informe, que se coordinan con la infraestructura de informes, proporcionan las vistas de entrada y salida para los informes.

Cada informe tiene una vista de entrada y una vista de salida. La tarea o finalidad de todos los JSP de entrada es la misma. Comparten el mismo aspecto, solicitan criterios de entrada diferentes aunque desde una agrupación de formatos de entrada compartida. Por otra parte, las vistas de salida tienen una estructura y formatos comunes. Basándose en la naturaleza de los informes, todas las partes reutilizables de los archivos JSP de informe se crean como componentes que se puede compartir. Estos componentes se crean basándose en la estructura de archivo y el convenio de denominación siguientes.

NombreInforme debe sustituirse por el nombre del nuevo informe y *ComponenteEntrada* debe sustituirse por el nombre del componente de entrada. Todos los archivos JSP de informe de entrada y salida utilizan `Reports.properties` para resolver temas multilingües. De este modo, los archivos de propiedades correlacionan todos los títulos y las claves que se utilizan en los archivos XML y JSP.

A continuación se proporciona una lista de componentes creados inicialmente para informes de operación de CSA:

ReportDaysWaitedHelper.jsp
Componente de recuadro editable de días esperados

ReportFulfillmentHelper.jsp
Componente de selección de despacho de pedidos

ReportInventoryAdjustmentCodeHelper.jsp
Selección de código de ajuste de inventario

ReportProductHelper.jsp
Componente de selección de producto

ReportStartDateEndDateHelper.jsp
Componente de entrada compuesto por una pareja de fechas

ReportVendorHelper.jsp
Componente de selección de proveedor

ReportFrameworkHelper.jsp
Funciones comunes para archivos JSP de entrada

ReportOutputHelper.jsp
Componente de formateador de páginas de salida

ReportProductFindDialogView.jsp
Página de entrada de criterios de búsqueda

ProductSearch.jsp
Página de selección de resultados de búsqueda

`ReportProductFindView` y `ReportProductSearchView` son páginas autónomas utilizadas por `ReportProductHelper`.

`Common.jsp` lo comparten las páginas de entrada y salida de informe. `ReportFrameworkHelper` está compartido por todos los archivos JSP de entrada de informe y `ReportOutputHelper` se aplica a todas las páginas de salida de informe.

Todos los demás archivos JSP son componentes de entrada y pueden arrastrarse a una página de entrada de informe que necesite la entrada.

Cada informe necesita tres archivos XML. *NombreInforme*`ReportDefinition.xml` define sentencias de SQL que se utilizan para el informe y el formato de cada columna del informe. Para saber cómo se escribe *NombreInforme*`ReportDefinition.xml`, consulte la documentación del diseño de la infraestructura de informes.

NombreInforme`ReportInputDialog.xml` es una definición de diálogo. Tiene la sintaxis siguiente:

```
<?xml version="1.0"?>

<dialog resourceBundle="reporting.reportStrings"
    windowTitle="NombreInformeReportWindowTitle"
    finishURL="GenericReportController" >

  <panel name="report"
    url="NombreInformeReportInputView"
```

```
    hasFinish="YES"
    helpKey="CM.reports.NombreInformeReportInputView.Help" />

</dialog>
```

Donde *NombreInforme* debe sustituirse por el nombre de informe. El título de ventana debe correlacionarse en el archivo `Reports.properties`. De este modo, `reporting.reportStrings` se define en el archivo `xml/tools/reporting/resources.xml` y apunta a `properties/com/ibm/commerce/tools/reporting/properties/Reports.properties`. La tecla de ayuda se correlaciona en `CMHelpMap.xml`. Finalmente, el URL es el nombre de vista de mandato del JSP de vista de entrada de informe.

NombreInforme`ReportOutputDialog.xml` es otra definición de diálogo que especifica las propiedades de salida de informe. Tiene el formato siguiente:

```
<?xml version="1.0"?>

  <dialog resourceBundle="reporting.reportStrings"
          windowTitle="NombreInformeReportOutputViewTitle"
          finishURL="" >

    <panel name="report"
           url="NombreInformeReportOutputView"
           passAllParameters="true"
           hasFinish="NO"
           hasCancel="NO"
           helpKey="CM.reporting.NombreInformeReportOutputView.Help" />

    <button name="ReportOutputViewPrintTitle"
           action="CONTENTS.printButton()" />

    <button name="ReportOutputViewOkTitle"
           action="CONTENTS.okButton()" />

  </dialog>
```

La vista de salida de ejemplo anterior tiene dos botones personalizados, **Print** y **OK**. Las funciones de proceso se definen en el JSP de vista de salida. De nuevo, el título de la ventana se correlaciona en el archivo de propiedades mientras que la tecla de ayuda se correlaciona en el archivo XML.

El diseño se basa en el concepto de componente reutilizable. Un archivo JSP de vista de entrada se compone de un conjunto de elementos de criterios, en función de la especificación de informe. Dado que los elementos de criterios pueden aparecer en páginas de entrada de informes diferentes, cada elemento de criterio se crea como un componente reutilizable. Si se aplica esta estrategia a la página JSP de vista de entrada, se logra lo siguiente:

1. Es fácil crear una nueva vista de entrada de informe.
2. Todas las vistas de entrada de informe tienen un aspecto coherente.
3. Simplificación de las funciones de validar, cargar y guardar necesarias para la Infraestructura de herramientas, porque estas funciones están incorporadas en cada componente.

El archivo JSP de vista de salida también está incorporado en ayudantes reutilizables. Los ayudantes proporcionan todos los formatos y las conversiones necesarios basándose en el tipo de datos y la preferencia de idioma. Los tipos de datos para cada columna se especifican en una sección definida por el usuario de

un XML de definición de informe. La coordinación de los ayudantes y la definición de XML hace que la creación de vista de salida sea simple al mismo tiempo que soporta formatos de salida sofisticados.

La sección siguiente explica cómo utilizar componentes reutilizables para crear archivos JSP de vista de salida y vista de entrada de informe.

Para crear un archivo JSP de vista de entrada de informe, deberá importar los componentes necesarios para el archivo JSP. Están disponibles los componentes siguientes:

1. DaysWaited – especifica el número de días esperados hasta hoy.
2. FulfillmentCenter – facilita la selección del centro de despacho de pedidos.
3. InventoryAdjustment – facilita la selección de código de ajuste de inventario
4. StartDateEndDate – especifica un periodo de tiempo.
5. Vendor – facilita la selección de proveedor.

Puede crear otros componentes a condición de que satisfagan la estrategia de diseño. Más adelante describiremos la creación de ayudantes.

Todos estos componentes proporcionan la interfaz común para los desarrolladores de páginas JSP:

1. Función JSP:

   ```
   String generateComponenteEntrada(String containerName,
       Hashtable reportsRB, String label1 [, String label2])
   ```

   Esta función crea un componente en la vista de entrada. *ComponenteEntrada* es el nombre de un componente. Cada componente tiene un nombre de contenedor (containerName), que es un nombre exclusivo que identifica el objeto JavaScript para este componente. Todas las funciones JavaScript harán referencia a containerName. Todos los componentes necesitan un paquete de recursos de informes, `reportsRB`, que refleja las preferencias de idioma actuales. Cada componente tiene como mínimo un título correlacionado a partir de las etiquetas.
2. Funciones JavaScript:

   function onLoad*ComponenteEntrada***(containerName)**
   Esta función se llamará cuando se cargue la página. Si es la primera vez que se está cargando la página en la transacción, inicialice el *ComponenteEntrada* (del bean de datos). Si se está volviendo a cargar esta página dentro de la transacción, recupere los datos guardados.

   function validate*ComponentEntrada***(containerName)**
   Esta función debe llamarse antes de someter la petición. Valida los datos de entrada de este componente. Si los datos no son válidos, haga emerger una ventana de diálogo con el mensaje apropiado para advertir al usuario. Devuelve verdadero si los datos son válidos y falso si alguna parte de los datos no es válida.

   function save*ComponenteEntrada***(containerName)**
   Esta función debe llamarse siempre que se navega a otra página diferente de la página actual. Guardará los datos de entrada actuales del componente para posteriormente recuperarlos cuando se vuelva a navegar a la página actual.

function visibleList(state)
Volver a llamar a la función. Se llama cuando la infraestructura desea visualizar recuadros de selección u ocultar recuadros de selección en la página. Esta función debe llamar a:

setSelect*Componente***Visible(container, state)**
Se define en cada componente de entrada en el que se utiliza el recuadro de selección. Todos los componentes de entrada tienen la misma implementación con nombres diferentes:

```
function setSelect<Component>Visible(container, state) {
    document.forms[container].ProductHelperSelectBox.style.visibility = state;
}
```

Una página JSP de entrada de informe debe denominarse *NombreInforme*ReportInputView.jsp y debe estar ubicada en el directorio siguiente:

AIX

/usr/WebSphere/AppServer/installedApps/*directorio_ear*/wctools.war/tools/reporting

400

/QIBM/ProdData/WebAsAdv4/installedApps/*directorio_ear*/wctools.war/tools/reporting

Linux

/opt/WebSphere/AppServer/installedApps/*directorio_ear*/wctools.war/tools/reporting

Solaris /opt/WebSphere/AppServer/tools/ reporting/

Windows

unidad:\WebSphere\AppServer\installedApps*directorio_ear*\wctools.war\tools\reporting.

Es un panel de diálogo. Para implementar un panel de diálogo, consulte la publicación Tools Framework User's Guide. Un panel de diálogo contiene dos secciones; una sección de generación de contenido HTML y una sección de función JavaScript. En la sección HTML, puede llamar a generate*ComponenteEntrada* para cada componente que desee mostrar en la pantalla de entrada. En la sección JavaScript, initializeValue, savePanelData y validatePanelData deben llamar a las funciones JavaScript correspondientes definidas en cada componente de entrada. initializedValue se llama cuando se está cargando la página. La Infraestructura de herramientas llamará a savePanelData y validatePanelData. SavePanelData deberá llamar a las siguientes funciones JavaScript necesarias de Infraestructura de informe:

1. setReportFrameworkOutputView("DialogView");
2. setReportFrameworkParameter("XMLFile","reporting.*NombrePanelSalida*")
3. setReportFrameworkReportXML("reporting.*XMLDefiniciónInforme*");
4. setReportFrameworkReportName("*NombreSQL*");, que se especifica en *XMLInforme*

También puede llamar a setReportFrameworkParameter("*nombre*", *valor*) para establecer los parámetros que son necesarios para el archivo JSP de salida de informe. Es una pareja de nombre y valor. Todas las etiquetas pasadas en el generador y el valor en las llamadas de función setReportFrameworkParameter son claves que se correlacionarán en la serie correcta basándose en el entorno nacional y la preferencia de idioma. La correlación se define en el archivo de propiedades Reports_es_ES.properties del directorio siguiente:

AIX

/usr/WebSphere/AppServer/installedApps/*directorio_ear*/properties/com/ibm/commerce/tools/reporting/properties

400

/QIBM/ProdData/WebAsAdv4/installedApps/*directorio_ear*/properties/com/ibm/commerce/tools/reporting/properties

Linux

/opt/WebSphere/AppServer/installedApps/*directorio_ear*/properties/com/ibm/commerce/tools/reporting/properties

Solaris

/opt/WebSphere/AppServer/installedApps/*directorio_ear*/properties/com/ibm/commerce/tools/reporting/properties

Windows

unidad:\WebSphere\AppServer\installedApps*directorio_ear*\properties\com\ibm\commerce\tools\reporting\properties

Utilización de ayudantes para las páginas de entrada y salida de informe

Una página de entrada de informe, que es un panel de diálogo, debe tener un conjunto de componentes para los criterios de entrada e implementar las cuatro funciones de script java siguientes:

initializeValues()
Se llama cada vez que se carga la página.

savePanelData()
Se llama cada vez que un usuario sale de esta página.

validatePanelData()
Se llama antes de enviar los criterios a la infraestructura de informes.

visibleList()
Se llama cuando la infraestructura de informes necesita un cambio en los valores de visibilidad de los componentes de esta página.

Para añadir un componente en una página de entrada de informe, importe la JSP de componente y añada una expresión JSP en la página de entrada. Por convenio de denominación, se denomina `generate`*`ComponenteEntrada`* con un nombre de contenedor (exclusivo de esta página), un paquete de recursos y uno o dos títulos como parámetros. Por ejemplo, en la página de entrada, tendrá algo similar a lo siguiente:

```
<%page "ReportStartDateEndDateHelper.jsp" %>
...
<body>
...
<%=generateStartDateEndDate("RequestPeriod", reportRB, "RequestPeriodTitleKey") %>
...
</body>
```

La expresión devuelve una serie que generará el componente visible en la página. Para coordinar las tres funciones anteriores, las funciones de devolución de llamada se definen con el convenio de denominación: `onLoad`*`ComponenteEntrada`*, `save`*`ComponentEntrada`* y `validate`*`ComponenteEntrada`*. Deben llamarse en `initializeValue`, `savePanelData` y `validatePanelDate` respectivamente.

La función `SavePanelData` también guarda información necesaria para la infraestructura de informes. Cada componente de entrada proporciona funciones de retorno que devuelven en dicho componente los ID, los nombres u otros campos entrados. Para obtener detalles sobre estas funciones de retorno, consulte cualquier JSP de componente de entrada.

Las páginas de entrada son responsables de manejar el formato. Todos los métodos de formato están contenidos en `ReportOutputHelper`, que se coordina con el archivo XML de definición de informe. Sólo el nombre de informe necesita especificarse en un archivo JSP de salida de informe. Puede copiar cualquier archivo JSP de salida de informe y modificar el valor `reportPrefix` para reflejar el nombre de informe.

Sin embargo, el archivo XML de definición de informe especifica todas las columnas y el formato de las mismas basándose en el tipo de columna. A continuación se proporciona una lista de tipos de columna:

Tabla 12.

Tipo de columna	Es el valor por omisión	Propiedades personalizables	Alineación por omisión
string	Sí	maxEntryLength	Derecha
integer	No	setMinimumIntegerDigits setMaximumIntegerDigits	Izquierda
decimal	No	setMinimumIntegerDigits setMaximumIntegerDigits setMinimumFractionDigits setMaximumFractionDigits	Izquierda
currency	No	currencySymbolColumn	Izquierda
enumeration	No		Derecha
date	No		Izquierda
time	No		Izquierda
month	No		Izquierda

El tipo de columna por omisión es string con las opciones de columna HTML `"align=left height=20 nowrap"`. Todos los tipos de columna pueden alterar temporalmente las opciones de columna por omisión especificando un código `<columnOptions>` en la columna.

Opcionalmente, todas las columnas también puede tener el valor `displayInReport` establecido en `true` o `false`. El valor por omisión es `true`, que significa que la columna se visualiza en el informe. Si el valor se establece en `false`, se oculta la columna. Esta característica puede utilizarse para personalizar la vista de salida de informe sin cambiar la consulta de SQL. También es útil cuando el formato de moneda necesita una columna de referencia para indicar al formateador de qué moneda se trata.

Las columnas integer, decimal, date y time se formatean basándose en los valores de idioma y moneda especificados en el contexto del mandato. Las columnas integer y decimal también pueden especificar un número mínimo y máximo de dígitos para los valores enteros y fraccionarios.

Por omisión, las columnas currency se formatean basándose en los valores de idioma y moneda especificados en el contexto del mandato. Si se especifica una columna de símbolo de moneda en una columna de moneda (currency), el símbolo de moneda de tres caracteres se recupera de la base de datos y se utiliza para formatear la moneda. La columna de símbolo de moneda a la que se hace referencia puede establecerse como invisible si el creador de informe no desea mostrar la serie de símbolo de moneda.

Enumeration es un tipo de columna especial que correlaciona un valor recuperado de la base de datos con una serie especificada por una clave. Por ejemplo, se puede recuperar `Y` o `N` de una tabla. Estos valores pueden correlacionarse con series más significativas, por ejemplo `Sí` o `No`, o `Aprobado` o `Rechazado` en informes diferentes o en idiomas diferentes. Para que esto sea posible, la columna puede definirse del modo siguiente:

```
<columns>
    <columnKey>C2</columnKey>
    <columnName>yyyColumnTitle</columnName>
    <columnType>enumeration</columnType>
        <Y>Sí</Y>
        <N>No</N>
</columns>
```

o

```
<columns>
    <columnKey>C2</columnKey>
    <columnName>yyyColumnTitle</columnName>
    <columnType>enumeration</columnType>
        <Y>Aprobado</Y>
        <N>Rechazado</N>
</columns>
```

donde las series para `Sí`, `No`, `Aprobado` y `Rechazado` se definen en el archivo de propiedades adecuado para permitir varios idiomas. Si la consulta devuelve valores tales como 0,1,2, etc. (dígitos) con los que desea correlacionar valores específicos, necesitará utilizar `<X_n></X_n>` como elemento. Por ejemplo:

```
<columnType>enumeration</columnType>
   <X_0>ValorPara0</X_0>
   <X_1>ValorPara1</X_1>
```

Escribir un informe usando componentes de página JSP reutilizables

Para escribir un informe usando componentes de página JSP reutilizables, deberá crear lo siguiente:

- Archivos JSP:
  `XXXReportInputView.jsp`
  `XXXReportOutputView.jsp`
- Archivos XML:
  `XXXReportInputDialog.xml`
  `XXXReportDefinition.xml`
  `XXXReportOutputDialog.xml`
- Archivos de propiedades actualizados:
  `Reports_es_ES.properties`, donde deberá añadir una sección para todas las correlaciones necesarias.
- Añadir mandatos de vista en la base de datos
  Se deben añadir los mandatos `XXXReportInputView` y `XXXReportOutputView` en la base de datos.
- Establecer control de acceso en las dos vistas (`XXXReportInputView` y `XXXReportOutputView`) añadidas en el paso anterior.

Capítulo 13. Asesor de productos

Ejemplos de personalización

En este documento, proporcionamos tres ejemplos de personalización. Estos ejemplos son:

- Cambiar el archivo JSP de Explorador de productos de ejemplo para utilizar iconos de operador diferentes.
- Cambiar los enlaces por omisión utilizados en Comparación de productos.
- Presentar valores de Explorador de productos sin utilizar los formatos proporcionados.

Escenario 1: Utilización de iconos de operador diferentes

En la metáfora del Explorador de productos, se utiliza la coordenada X de la selección en el icono de operador (una imagen representa todos los operadores) para determinar qué operador se ha seleccionado. Al sustituir la imagen, asegúrese de que las coordenadas siguen siendo las mismas para que, al pulsar en la imagen, se identifique el operador correcto. Los archivos de imágenes están en `CommerceDir\web\tools\pa\icons`. `equalone.gif` representa los operadores =, <> y `equaltoo.gif` representa los operadores =,<>, <=, >= utilizados para valores numéricos de atributo. Las coordenadas X esperadas son:

- = 0-22
- <> 23-44
- <= 45-66
- >= 67-88

Cuando la selección de valor pasa un parámetro de coordenada X, se aplica el operador apropiado a dicha selección.

Escenario 2: Enlaces desde la metáfora de Comparación de productos

La metáfora de Comparación de productos soporta enlaces de productos individuales a otra página. Actualmente sólo existe soporte para otra página (a no ser que se cree un atributo con los URL para los valores). La JSP de ejemplo enlaza a la página de Visualización de producto. Asigne esta página mediante la propiedad `productLinkName` en el `ProductCompareDataBean`. En el ejemplo, esto se redirige mediante el mandato `ClickInfo` que reúne estadísticas sobre la utilización de la metáfora de Comparación de productos. Cuando se enlaza a otra página, puede que sea necesario pasar los parámetros desde la página de Comparación de productos. Identifique los parámetros que se deben pasar en la propiedad `productLinkParameters` del `ProductCompareDataBean`. Para los nombres de parámetros identificados aquí se pasará el parámetro a la página de enlace si el parámetro existe. Además, al parámetro `productId`, que en el ejemplo se ve como `ECConstants.EC_PRODUCT_ID`, se le asignará el valor del ID de entrada de catálogo (`catentry_id`) para el artículo con el que está asociado el enlace. `productId` permite a la página de enlace identificar qué artículo ha seleccionado el usuario. Actualmente no existe soporte para otros parámetros con valores exclusivos de artículos individuales en la tabla de Comparación de productos.

Escenario 3: Personalización de la presentación de Explorador de producto

El formato DynamicForm presenta el `ProductExploreDataBean`. Para realizar la presentación usted mismo, establezca una subclase para la clase DynamicForm (consulte los JavaDocs para conocer las signaturas de método) y altere temporalmente el método de presentación o bien obtenga los datos directamente de `ProductExploreDataBean`. En el método de presentación, utilice el método `getDataBean()` para obtener el objeto `ProductExploreDataBean`. Utilice el método `ProductExploreDataBean.getFormElements()` para obtener una colección de los `ColumnDataBeans` que representan cada columna de atributo. Dentro de cada columna, el objeto `ColumnDataBean` contiene los valores para dicha columna. Utilice el método `ColumnDataBean.getColumn()` para obtener la colección de objetos `DsData` que contienen los valores de atributo individuales. Consulte los JavaDocs en `ColumnDataBean` y `DsData` para obtener información de métodos. Utilice el método `getPresentationString()` del `DsData` para recuperar una representación formateada del valor de atributo y el método `getUnformattedData()` para los datos no procesados. Para crear los parámetros correctos para el formulario HTML, utilice el método `ColumnDataBean.getFormElementName()` para recuperar el nombre de parámetro correcto y el método `DsData.getUnformattedData()` a fin de obtener cada valor. Para realizar la selección correcta de operador, consulte la información de iconos de Operador.

Capítulo 14. Proyectos de normas

El ciclo de vida de un proyecto de normas incluye las etapas siguientes:

1. Crear el proyecto de normas
2. Configurar un servicio de normas basado en el nuevo proyecto de normas
3. Invocar el servicio de normas
4. Eliminar el servicio de normas basado en el proyecto de normas

Suposición: La información aquí proporcionada sólo incluye los puntos 2, 3 y 4. Se supone que el proyecto de normas ya se ha creado.

Para obtener información sobre cómo crear proyectos de normas, consulte los servicios profesionales de Blaze.

Cómo configurar un servicio de normas basado en el proyecto de normas personalizado

El sistema de normas no distingue entre un proyecto de normas proporcionado con WebSphere Commerce o un proyecto de normas personalizado. Siempre puede utilizar la Consola de administración para crear, modificar o eliminar un servicio de normas de un proyecto de normas. Sin embargo, el proyecto de normas personalizado debe satisfacer determinados requisitos. Los requisitos son los siguientes:

1. Un suceso externo pasado al proyecto de normas debe desencadenar la ejecución del proyecto de normas
2. El suceso externo tiene que implementar la interfaz `com.ibm.commerce.rules.InvocationContext`. Esta interfaz es sólo una simple interfaz de identificadores.

Cómo invocar un servicio de normas basado en un proyecto de normas personalizado

Puede invocar servicios de normas (creados a partir de proyectos de normas que satisfacen los dos criterios anteriores) de la misma manera que los servicios de normas creados a partir de proyectos de normas enviados con WebSphere Commerce. Es decir, llamando al mandato de tarea siguiente: `com.ibm.commerce.rules.commands.InvokePersonalizationRuleServiceCommand`.

Parte 2. Apéndices

Avisos

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

Es posible que IBM no ofrezca en otros países los productos, servicios o características descritos en este documento. Solicite información al representante local de IBM acerca de los productos y servicios disponibles actualmente en su región. 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.

Cualquier referencia hecha en esta publicación a un programa bajo licencia de IBM no pretende afirmar ni implica que sólo pueda utilizarse ese programa bajo licencia 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. Es responsabilidad del usuario la evaluación y verificación del funcionamiento junto con otros productos, excepto aquellos designados expresamente por IBM.

IBM puede tener patentes o solicitudes de patente pendientes que cubran temas descritos en este documento. La adquisición de este documento no confiere ninguna licencia sobre dichas patentes. Puede enviar consultas sobre licencias, por escrito, a:

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

Para realizar consultas sobre licencias relacionadas con la 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 licencias 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
Av. Diagonal 571, Edif. L'illa
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 este documento, 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.

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 utilizados en operaciones comerciales cotidianas. Para ilustrar los ejemplos de la forma más completa posible, éstos incluyen nombres de personas, empresas, tipos de tarjetas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es puramente casual.

Las imágenes, marcas registradas y marcas comerciales de tarjetas de crédito que se proporcionan en este producto únicamente deben utilizarlas los comerciantes a quienes el propietario de la marca de la tarjeta de crédito haya autorizado a aceptar pagos mediante esa tarjeta de crédito.

Marcas registradas

Los términos siguientes son marcas comerciales o marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países:

Blaze Advisor es una marca registrada de HNC Software Inc. en los Estados Unidos y/o en otros países.

Microsoft, Windows, Windows NT y el logotipo de Windows son marcas registradas de Microsoft Corporation en los Estados Unidos y/o en otros países.

Netscape es una marca registrada de Netscape Communications Corporation en los Estados Unidos y/o en otros países.

Oracle es una marca registrada de Oracle Corporation en los Estados Unidos y/o en otros países.

Java, JavaBeans y todas las marcas registradas y logotipos basados en Java son marcas registradas de Sun Microsystems, Inc.

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