IBM® WebSphere™ Commerce

WebSphere 贸易加速器 定制指南

版本 5.4

IBM® WebSphere™ Commerce

WebSphere 贸易加速器 定制指南

版本 5.4

注意:

在使用本资料及其支持的产品之前，请务必阅读第 89 页的『声明』中的一般信息。

第一版（2002 年 1 月）。

本版本适用于 IBM® WebSphere Commerce 版本 5.4，以及所有后续发行版和修订版，直至在新版本中另有声明。确认您正在使用本产品级别的正确版本。

通过您当地的 IBM 代表或 IBM 分部可订购出版物。以下地址不备有出版物。

IBM 欢迎您提出宝贵意见。您可以将意见通过以下任何一种方式发送给我们:

1. 请将电子邮件发送到下面列出的网络标识之一。如果需要答复，请确保在电子邮件中提供您完整的网络地址。

   因特网: torrcf@ca.ibm.com
   IBMLink: toribm(torrcf)

2. 通过传真方式，请使用以下号码:

   美国和加拿大: 416-448-6161
   其它国家或地区: (+1)-416-448-6161

3. 邮寄到以下地址:

   IBM Canada Ltd. Laboratory
   B3/KB7/8200/MKM
   8200 Warden Avenue
   Markham, Ontario,
   L6G 1C7
   Canada

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 可以它认为合适的任何方式使用或分发此信息，而无须对您承担任何责任。

目录

在开始之前

本书中使用的约定

本书使用以下突出显示的约定:

粗体字表示命令或图形用户界面（GUI）控件（例如，字段名、按钮或菜单选项）。

等宽字表示完全按显示原样输入的文本示例和目录路径。

*斜体*字用于表示强调和可用您自己的值替换的变量。

此图标标记一个“技巧”— 可以帮助您完成任务的附加信息。

知识需求

要定制 WebSphere 贸易加速器，您需要以下知识:

- HTML 和 XML
- 结构化查询语言（SQL）
- Java 编程

有关定制 WebSphere Commerce 的更多信息，请参阅《WebSphere Commerce 程序员指南》。此书可以从以下 Web 站点中得到:

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

本书的组织方式

下表概括了本书的组织方式:

表 1.

组成部分	描述	位置
业务关系管理	此部分详细描述了在定制“合同”工具时必须考虑的问题	第 3 页的第 1 章，『业务关系管理』
客户服务代表	此部分详细描述了在定制“客户服务代表”工具时必须考虑的问题	第 9 页的第 2 章，『客户服务代表工具』
顾客简要表	此部分详细描述了在定制“顾客简要表”工具时必须考虑的问题	第 21 页的第 4 章，『顾客简要表』
竞销	此部分详细描述了在定制“竞销”工具时必须考虑的问题	第 29 页的第 5 章，『竞销』
赠券	此部分详细描述了在定制“赠券”工具时必须考虑的问题	第 49 页的第 7 章，『赠券』
折扣	此部分详细描述了在定制“折扣”工具时必须考虑的问题	第 51 页的第 8 章，『折扣』

表 1. (续)

组成部分	描述	位置
RFQ	此部分详细描述了在定制 RFQ 工具时必须考虑的问题	第 53 页的第 9 章, 『RFQ 响应』
库存	此部分详细描述了在定制“库存”工具时必须考虑的问题	第 65 页的第 10 章, 『预期库存』
商务智能	此部分详细描述了在定制报表工具时必须考虑的问题	第 67 页的第 11 章, 『商务智能』
报表框架	此部分详细描述了在定制报表工具时必须考虑的问题	第 69 页的第 12 章, 『报表框架的概述』
搜索	此部分详细描述了在定制产品目录搜索工具时必须考虑的问题	第 37 页的第 6 章, 『产品目录搜索』
Blaze Rules Advisor	此部分详细描述了在定制 Brokat Advisor 规则项目时必须考虑的问题	第 85 页的第 14 章, 『规则项目』
产品顾问	此部分详细描述了在定制“产品顾问”工具时必须考虑的问题	第 83 页的第 13 章, 『产品顾问』

第 1 部分 定制 WebSphere 贸易加速器

本书描述如何定制 WebSphere 贸易加速器。通过提供有关 WebSphere 贸易加速器设计决策的背景知识，本书将教会您如何进行定制，并详细描述了定制所必需的步骤。

虽然本书的目的在于指导您定制 WebSphere 贸易加速器，但是它并不是设计来让您浏览一系列以穷举方式列出的所有可能定制。相反，本书的结构引入了构成 WebSphere 贸易加速器的不同类属部分并解释如何定制这些部分。

WebSphere 贸易加速器是一个工具的集合，设计它的目的在于使您的站点的日常业务运作变得更为方便。即，它意在作为商业人员用于创建和修改站点操作的任何方面的一个界面，这样商务人员无需不断的联系负责站点操作的 IT 工作人员。因此，WebSphere 贸易加速器包含很多个组件，每个组件针对商务的一个关键方面。虽然为了使这些工具更通用我们做出了很大的努力，但是这个目标同样意味着某些用户可能会发现某些特定的需要不能得到满足。如果您在站点的创建期间扩展了数据库，且这些扩展与本文档中所列出的任何工具有关，则数据对于加速器是不可见的，直至您定制了相关的工具。

本书描述了如何定制 WebSphere 贸易加速器中的以下组件。

第 1 章 业务关系管理

本章中讨论的元素代表 WebSphere 贸易加速器中“帐户”和“合同”组件的用户界面。这些元素使帐户、合同和以下其它操作的创建和维护更为方便：销售经理或客户代表需要这些操作以执行与业务关联的日常任务。业务关系管理组件包含以下元素:

- “帐户”笔记本
- “合同”笔记本

定制示例

以下示例概括了如何定制此部分的 WebSphere 贸易加速器。

方案 1: 向合同用户界面添加新的条款和条件

实现概述

在尝试使 WebSphere 贸易加速器通用并对尽可能多的人员有用，有一些条款和条件被省略了。所以，您可能会发现有必要向您的站点添加条款和条件以使其更加适合于您的业务。此方案将在向站点添加附加条款和条件所必需的所有步骤中向您提供指导。

定制步骤

1. 在服务器中定义条款和条件。在您可以执行用户界面的定制之前，您必须已完成了以下操作:
   - 在 `B2BTrading.dtd` 文件中定义新的条款和条件。有关完整的详细信息，请参阅《*IBM WebSphere Commerce 程序员指南*》中的第 7 章。
   - 为新的条款和条件创建企业 bean 或访问 bean。
2. 向用户界面添加条款和条件。要向用户界面添加必需的元素，请执行以下操作:
   a. 为条款和条件的用户界面页面创建 JSP 文件。此页面必须包含动态 JavaScript 对象以从页面上的输入字段捕获数据。创建数据 bean，它封装了从访问 bean 装入数据的操作。还可以选择直接在页面上包含访问 bean 以从数据库中装入条款和条件数据。`contractId` 参数被传递给所有的页面，这有助于减少数据装入的执行。此 JSP 文件的样本如下:

```
<!--==========================================================================
//*--------------------------------------------------------------------------
//* The sample contained herein is provided to you "AS IS".
//*
//* It is furnished by IBM as a simple example and has not been thoroughly tested
//* under all conditions.  IBM, therefore, cannot guarantee its reliability,
//* serviceability or functionality.
//*
//* This sample may include the names of individuals, companies, brands and
//* products in order to illustrate concepts as completely as possible.  All of
//* these names are fictitious and any similarity to the names and addresses used
//* by actual persons or business enterprises is entirely coincidental.
//*--------------------------------------------------------------------------
//*
  ==========================================================================-->
<!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;

////////////////////////////////////////
// LOAD-SAVE-VALIDATE SCRIPTS
////////////////////////////////////////
function onLoad() {

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

   // Check to see if the model has already been loaded
   var isModelLoaded = parent.get("ContractMyTCModelLoaded", null);

   if (isModelLoaded) {
       // Returning to this page, reload from the model
       // Get the model
       MyTCModel = parent.get("ContractMyTCModel", null);
   }
   else {
       // First visit to this page, create the model

       // Create the model to store the MyTC data
       MyTCModel = new ContractMyTCModel();

       // Persist the model
       parent.put("ContractMyTCModel", MyTCModel);
       parent.put("ContractMyTCModelLoaded", true);

var myTCPolicyList = new Array();
<%
try {
// Load all the policies from the database
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;

// Check if this is an update of the contract
if (<%= foundContractId %> == true) {
// Load the data from the databean
<%
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();
```

```
    // handle error messages back from the validate page
    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() {

// Set the input fields
document.MyTCForm.Attr1.value = MyTCModel.attr1;
document.MyTCForm.Attr2.value = MyTCModel.attr2;

// Load the policies
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>

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

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

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

<FORM NAME="MyTCForm">

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

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

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

</FORM>

</BODY>
</HTML>
```

b. 为条款和条件的用户界面页面创建 JavaScript 文件。创建用于验证和提交 JavaScript 数据的函数。当您提交数据时，您需要创建与新条款和条件的 XML 格式匹配的新 JavaScript 对象。此 JavaScript 文件的样本如下:

```
//*-------------------------------------------------------------------------------------
//* The sample contained herein is provided to you "AS IS".
//*
//* It is furnished by IBM as a simple example and has not been thoroughly tested
//* under all conditions.  IBM, therefore, cannot guarantee its reliability,
//* serviceability or functionality.
//*
//* This sample may include the names of individuals, companies, brands and products
//* in order to illustrate concepts as completely as possible.  All of these names
```

```
//* are fictitious and any similarity to the names and addresses used by actual persons
//* or business enterprises is entirely coincidental.
//*-------------------------------------------------------------------------------------
//*

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) {
  // Check if attr1 is empty or too long
  if (!tcModel.attr1)
   {
    put("attr1Empty", true);
    gotoPanel("MyTCHeading");
    return false;
   }

  if (!isValidUTF8length(tcModel.attr1, 10))
   {
    put("attr1TooLong", true);
    gotoPanel("MyTCHeading");
    return false;
   }
  // Check if attr2 is empty or too long
  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 != "") {
// Change the term and condition
myTC.action = "update";
myTC.referenceNumber = tcModel.tcReferenceNumber;
}
else {
// Create a new term and condition
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. 修改 Contract.js 文件以调用在步骤 2b 中创建的新 submit 和 validate 函数。

d. 修改资源束以添加您所需的新字符串。

e. 修改 ContractNotebook.xml 以添加新的面板。

方案 2: 从合同用户界面中除去条款和条件

实现概述

正如您可能会发现缺少一些条款和条件，您还可能会发现有些条款和条件对您的业务而言是不必要的。此方案将在从站点中除去多余条款和条件所必需的所有步骤中向您提供指导。

定制步骤

修改 ContractNotebook.xml 文件并除去包含有您希望除去的条款和条件的面板元素的部分。在以下目录中找到这个文件:

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 *drive*:\WebSphere\CommerceServer\xml\tools\contract\

第 2 章 客户服务代表工具

本章中讨论的元素代表 WebSphere 贸易加速器中“客户服务代表（CSR）”工具的用户界面。这些元素使顾客、订单和以下其它操作的创建和维护更为方便: CSR 需要这些操作以执行与业务关联的日常任务。CSR 工具组件包含以下元素:

- “顾客”向导
- “顾客”笔记本
- “订单”向导
- “订单”笔记本

定制示例

以下示例概括了如何定制 WebSphere 贸易加速器中的 CSR 工具。

方案 1: 向商业顾客摘要对话框添加附加的顾客数据

实现概述

此示例说明了如何向“商业顾客”摘要对话框添加“商业顾客”的雇员号。

定制步骤

扩展 ShopperSummaryB2BDialog.jsp。在以下目录中找到这个文件:

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 *drive*:\WebSphere\CommerceServer\web\tools\csr\

关于如何更改 JSP 文件以在摘要对话框中显示雇员号，请参阅下面的代码样本。雇员号是 OptoolsRegisterDataBean 的属性。此样本没有使用属性文件来存储标号 EmployeeNumber。样本输出将雇员号显示在“顾客摘要”对话框中:

```
<%--
//*************************************************************************
//*-------------------------------------------------------------------------
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
//*
//*-------------------------------------------------------------------------
//*
--%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0//EN">
:
:
: Some code is omitted here
:
```

```
:
<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" %>
:
:
: Some code is omitted here
:
:
</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>
   Employee Number:
   <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>
```

方案 2: 在“顾客服务 — 订单”管理界面中启用购物车

实现概述

如果商店设计者希望使用 CSR 工具管理状态为 P 的购物者订单，则他们必须修改 **CSROrderSearchB2B** 和 **CSROrderSearchB2C** 视图。

定制步骤

扩展 CSROrderSearchB2B.jsp 和 CSROrderSearchB2B.jsp JSP 文件。通过在订单状态条件中添加 Pending 选项，CSR 将可以搜索顾客的未决订单并管理它们。未决订单是那些在购物车中的订单。这两个文件都位于以下目录中:

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 *drive*:\WebSphere\CommerceServer\web\tools\order\

请参阅以下代码样本:

```
<%--
//*************************************************************************
//*-------------------------------------------------------------------------
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
//*
//* (c) Copyright IBM Corp. 2001
//*
//* US Government Users Restricted Rights - Use, duplication or
//* disclosure restricted by GSA ADP Schedule Contract with 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>
<%
   // obtain the resource bundle for display
   CommandContext cmdContextLocale =
     (CommandContext)request.getAttribute(com.ibm.commerce.server.ECConstants.EC_COMMANDCONTEXT);
   Locale jLocale = cmdContextLocale.getLocale();
   Hashtable orderLabels =
     (Hashtable)ResourceDirectory.lookup("order.orderLabels", jLocale);

      // retrieve request parameters
      JSPHelper jspHelp = new JSPHelper(request);
      String customerId =
        jspHelp.getParameter(ECOptoolsConstants.EC_OPTOOL_CUSTOMER_ID);

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

      //compose entire list of account names
      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">
<!---- hide script from old browsers

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(",;is not valid");
      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">
<!--
//For IE if (document.all) {
   onLoad();
}
//-->
</SCRIPT>

</BODY>
</HTML>
```

第 3 章 协作工作空间

本章中讨论的元素代表一个案例学习，它详细描述了定制 QuickPlace 和创建随 WebSphere Commerce 附带的“多乐五金店”模板所必需的步骤。它还概括了定制邀请用户加入协作工作空间时发送的电子邮件（e-mail）的内容所必需的步骤。

定制示例

以下示例概括了如何定制此部分的 WebSphere 贸易加速器。

注: 方案 1 和 2 密切相关。如果您执行了方案 1，则您通常也希望遵循方案 2。

方案 1: 定制工作空间的外观

实现概述

您希望首先定制的“协作工作空间”功能可能是所创建的协作工作空间的“外观”。要定制外观，您必须创建称为 PlaceType 的模板。有关定制的更多详细信息，请参阅《*定制 QuickPlace*》红皮书。

定制步骤

此示例使用 WebSphere Commerce 5.4 中附带的“多乐五金店”样本商店。此示例说明了如何创建 QuickPlace:

1. 执行以下操作，创建缺省 QuickPlace:
   a. 请确保 QuickPlace 服务器正在运行。打开浏览器，浏览到 `http://qp_server/quickplace` 并以 QuickPlace 管理员身份登录。
   b. 单击**创建 QuickPlace**。
   c. 选择**标准的组 QuickPlace**。填写适用的字段，并单击**下一步**。
   d. 以创建者身份登录。
2. 更新 QuickPlace 的“欢迎”页面。单击**编辑**。做出期望的更改并单击**发布**。这将更新“欢迎”页面。
3. 执行以下操作，更改 QuickPlace 的徽标:
   a. 单击目录中的**定制**。
   b. 单击**基础**。
   c. 单击**更改基础**。
   d. 使用**简单文本**、**徽标制作程序**或**上载徽标图形**来设置徽标。“多乐五金店”样本曾使用**上载徽标图形**。现在徽标已更新

方案 2: 创建定制主题

实现概述

主题控制 QuickPlace 的“外观”。此示例为 QuickPlace 的布局和样式创建一个“超文本标记语言（HTML）”页面和一个级联样式表（CSS）。有关如何编写这些文件的更多信息，请参阅 *Customizing QuickPlace* 红皮书。

注意:

因为成员管理是通过 WebSphere Commerce 管理的，所以请除去目录中的**成员**链接。要除去链接，请在插入目录时在“页面布局”文件中使用以下代码:

```
<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=}>
```

定制步骤

1. 单击目录中的**定制**，再单击**定制主题**，然后单击**新建主题**。填写标题，上载**样式表**和**页面布局**文件。
2. 单击**下一步**。
3. 单击目录中的**定制**，然后单击**装饰**
4. 单击**选择主题**。
5. 选择您刚创建的主题并单击**下一步**。这应该会更新 QuickPlace 的外观。
6. 注意，目录中的**成员**链接消失了。
7. 从 QuickPlace 中创建 PlaceType。您现在可以根据您刚定制的 QuickPlace 创建 PlaceType 了。单击目录上的**定制**，然后单击 **PlaceType 选项**。
8. 单击**编辑**。对**允许从该 QuickPlace 中创建 PlaceType 吗**? 选择**是**，对**在未来从 PlaceType 创建的 QuickPlace 中包含该 QuickPlace 的当前成员**? 选择**否**
9. 单击**下一步**。
10. 打开浏览器，浏览到 `http://qp_server/quickplace` 并以 QuickPlace 管理员身份登录。
11. 单击 **PlaceType**。
12. 单击**创建 PLACETYPE...**。输入 PlaceType 的名称并选择您希望作为基础的 QuickPlace。
13. 单击**下一步**。您应该可以看见新的 PlaceType。

方案 3: 定制电子邮件通知

实现概述

要将所有商店的缺省电子邮件通知更改为一个定制页面，您需要修改 CollabEmailContent.jsp 文件以使其包含您期望的内容。

定制步骤

1. 修改 CollabEmailContent.jsp 文件。在以下目录中找到这个文件:

   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 *drive*:\WebSphere\AppServerinstalledApps\WC_Enterprise_App_demo.ear\wcstores.war

2. 执行以下操作，为特定商店定制电子邮件:

   a. 打开 DB2 命令窗口。

   b. 连接到 WebSphere Commerce 数据库。要连接到缺省数据库，请使用以下命令:

   ```
   db2 connect to mall
   ```

   c. 执行以下 DB2 命令:

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

   d. 将您定制的模板文件 CollabEmailContent.jsp 放入商店目录:

   AIX

   /usr/WebSphere/AppServer/installedApps/WC_Enterprise_App_demo.ear/wcstores.war/*store_name*

   400

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

   Linux

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

   Solaris

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

   Windows

   drive:\WebSphere\AppServer\installedApps\WC_Enterprise_App_demo.ear\wcstores.war*store_name*

   其中 *store_name* 是在发布商店时您选择的商店目录名。

   e. 重新启动您的 WebSphere 管理服务器。

第 4 章 顾客简要表

本章中讨论的元素代表 WebSphere 贸易加速器中“顾客简要表”组件的用户界面。这些元素使顾客简要表和所有以下操作的创建和维护更为方便：卖方或商家需要这些操作以执行与业务关联的日常任务。顾客简要表组件包含以下元素:

- “顾客简要表”向导
- “顾客简要表”笔记本

样本代码

本部分引用的代码样本包含在位于以下 URL 的 zip 文件中:

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

要完成本章中的某些部分，您必须在运行 WebSphere Application Server 4 的开发机器上下载并安装此样本代码。

一旦下载完毕，将此 zip 文件解压缩到以下目录中:

AIX `/usr/WebSphere/CommerceServer`

400 `/QIBM/UserData/WebSphere/CommerceServer`

Linux `/opt/WebSphere/CommerceServer`

Solaris `/opt/WebSphere/CommerceServer`

Windows *drive*`:\WebSphere\CommerceServer`

定制示例

以下示例概括了如何定制此部分的 WebSphere 贸易加速器。

方案 1: 向顾客简要表添加属性

实现概述

在此示例中，`USERS` 表的 `FIELD1` 列中存储用户的饮料首选项。以下值是有效的:

- 啤酒
- 葡萄酒
- 咖啡
- 茶
- 水

示例演示了这些步骤中的每个步骤。在您安装样本代码时创建的 `profcust` 目录中找到此示例。

定制步骤

1. 通过创建新的 JSP 文件来向顾客简要表笔记本添加新的页面。在我们的示例中，此页面将显示两个选项:

- 忽略饮料首选项
- 目标顾客具有以下一个或多个饮料首选项:

首选项将作为复选框出现在第二个选项下。有关更多详细信息，请查阅以下目录中附带的 JSP 页面 BeveragePanel.jsp:

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 *drive*:\WebSphere\CommerceServer\profcust\web\tools\ segmentation\

2. 在您创建了 JSP 文件之后，您必须将其添加到 VIEWREG 表。有关更多详细信息，请查阅以下目录中附带的 SQL 脚本 beverage.sql:

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 *drive*:\WebSphere\CommerceServer\profcust\schema\

3. 新的页面必须由顾客简要表笔记本识别，顾客简要表笔记本位于以下目录中:

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 *drive*:\WebSphere\CommerceServer\xml\tools\segmentation\ 复制 SegmentNotebook.xml。在此样本中，这个新文件的名称是 MySegmentNotebook.xml。以下元素将被添加到文档:

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

您必须在分段属性文件中包含新的面板名称，该属性文件位于以下目录:

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

drive:\WebSphere\AppServer\installedApps/WC_Enterprise_App_demo.ear\

properties\com\ibm\commerce\tools\segmentation\
为了使其工作，您需要将向 Resources.properties 添加以下行:

```
segmentNotebookBeveragePanel=Preferred beverage
```

还可以选择从以下目录中将 MySegmentNotebook.xml:

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 *drive*:\WebSphere\CommerceServer\profcust\xml\tools\segmentation\MySegmentNotebook.xml

复制到:

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 *drive*:\WebSphere\CommerceServer\xml\tools\segmentation\MySegmentNotebook.xml

4. 现在您需要确保已识别新的 XML 文档。要实现此操作，请修改以下目录中的 Resources.xml 文件:

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 *drive*:\WebSphere\CommerceServer\xml\tools\segmentation\

修改以下元素:

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

将元素更改为:

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

将文件保存为 MyResources.xml

还可以选择从以下目录中将 MyResources.xml:

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 *drive*:\WebSphere\CommerceServer\profcust\xml\tools\segmentation\MyResources.xml

复制到:

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 *drive*:\WebSphere\CommerceServer\xml\tools\segmentation

5. 使应用程序指向 MyResources.xml。修改以下目录中的 demo.xml 文件:

   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 *drive*:\WebSphere\CommerceServer\instances\demo\xml\

   查找:

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

   将元素更改为:

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

6. 扩展描述顾客简要表的数据 bean com.ibm.commerce.tools.segmentation.SegmentNotebookDataBean。在此示例中，com.mycompany.tools.segmentation.MySegmentNotebookDataBean 扩展数据 bean 来为饮料属性提供特性。在以下目录中找到此样本 MySegmentNotebookDataBean.java:

   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 *drive*:\WebSphere\CommerceServer\profcust\lib\com\mycompany\tools\segmentation\

7. 为了使段笔记本识别新的数据 bean，您需要修改用于描述顾客简要表笔记本的 XML。返回到 MySegmentNotebook.xml 并注意以下更改:

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

更改为:

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

8. 扩展用于保存顾客简要表的控制器命令。此命令为 com.ibm.commerce.tools.segmentation.SegmentSaveControllerCmd。请查阅以下目录中的 MySegmentSaveControllerCmdImpl 中的样本命令实现:

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 *drive*:\WebSphere\CommerceServer\profcust\lib\com\mycompany\tools\segmentation\

此示例将为饮料条件构造一个简单条件。

为了使应用程序识别这个新命令，您需要将其添加到 CMDREG 表。有关更多详细信息，请参阅上面步骤 1a 中的 beverage.sql。

9. 扩展用于评估顾客简要表的任务命令，该命令是 com.ibm.commerce.membergroup.commands.CheckUserInMemberGroupCmd。有关如何评估新条件的详细信息，请查阅以下目录中的样本命令 MyCheckUserInMemberGroupCmdImpl.java:

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

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

您还需要在 CMDREG 表中注册此命令。有关更多详细信息，请参阅 beverage.sql。

10. 扩展用于显示顾客简要表的约束列表的任务命令。此命令为 com.ibm.commerce.tools.segmentation.SegmentConstraintListCmd。有关您必须更改的内容的详细信息，请查阅以下目录中的样本命令 MySegmentConstraintListCmdImpl:

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 *drive*:\WebSphere\CommerceServer\profcust\lib\com\mycompany\tools\segmentation\

您还需要在 CMDREG 表中注册此命令。有关更多详细信息，请参阅 bevarage.sql。

11. 将以下类

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

（它们位于以下目录:

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 *drive*:\WebSphere\CommerceServer\profcust\lib）

添加到 wcsmcruntime.jar 文件中，该文件位于以下目录:

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

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

12. 将步骤 1 中的 BeveragePanel.jsp 文件复制到以下目录中:

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

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

13. 这些更改需要对您的访问控制设置做出更改，而这已超出了本指南的讨论范围。请参阅《*WebSphere Commerce 访问控制指南*》，您可以从以下 URL 处得到它:

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

第 5 章 竞销

在数据库中直接输入规则

虽然用户界面是向数据库输入规则数据使用的最简便方法，但是某些场合可能需要您在 INITIATIVE 表的 RULE 列中手工输入规则。规则是使用 XML 定义的，在此情况下，您必须遵循文档类型定义（DTD），如下所示:

INITIATIVE 表的 RULE 列的 DTD 如下:

```
<!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>
]>
```

simpleCondition 元素

活动的缺省实现支持以下 simpleCondition 元素。您可以将这些元素以任意组合方式与 orListCondition、andListCondition 和 openCondition 元素组合在一起以创建规则的条件部分。

表 2.

变量名	支持的运算符	支持的值	限定符
segment	= !=	顾客简要表的名称。	无
shoppingCartSku	= !=	产品库存标识。	无
shoppingCartCategory	= !=	产品类别名。	语言
purchaseHistorySku	= !=	产品库存标识。	无
purchaseHistoryCategory	= !=	产品类别名。	语言
shoppingCartTotal	= != > < >= <=	十进制值。	货币

表 2. (续)

变量名	支持的运算符	支持的值	限定符
dayOfWeek	= !=	SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY	无

openCondition 元素

缺省实现支持以下 openCondition 元素。您可以将这些元素以任意组合方式与 orListCondition、andListCondition 和 simpleCondition 元素组合在一起以创建规则的条件部分。

表 3.

开放条件名称	支持的参数
shoppingCartTotal	comparisonType — 可以是 “>”、“<” 或 “=” 之一。值 — 十进制值货币
shoppingCartCategory	comparisonType — 可以是 “=” 或 “!=” 之一。值 — 类别语言的名称
shoppingCartSku	comparisonType — 可以是 “>”、“<” 或 “=” 之一。值 — 十进制值货币
purchaseHistoryCategory	comparisonType — 可以是 “=” 或 “!=” 之一。值 — 类别货币的名称
purchaseHistorySku	comparisonType — 可以是 “>”、“<” 或 “=” 之一。值 — 十进制值货币

操作元素

缺省实现支持以下操作元素。这些元素可以与任何条件元素一起构造一个规则。

表 4.

操作名称	支持的参数	支持的值	支持的子参数
suggestiveSell	queryOperator	or and	无
	queryType	skuQuery genericQuery	无
	sku	产品库存标识	无
	skuPrefix	产品库存标识前缀	无
	category	产品类别名	语言
	productDescription	产品类别描述	语言
	inventoryQuantity	整数	comparisonOperator
	listPrice	十进制值	comparisonOperator 货币
	availabilityDate	日期，格式为 yyyy-mm-dd-00.00.00	comparisonOperator
collaborativeFiltering	无		
awarenessAd	collateral	广告名称	

表 4. (续)

操作名称	支持的参数	支持的值	支持的子参数
discountAd	discountCollateral	广告名称	折扣标识
couponAd	couponCollateral	广告名称	
categoryRecommendation	category	类别的名称	

以下示例活动显示了一个针对男性的春季广告以及一个针对女性的秋季广告:

```
<ruleSet>
   <rule>
      <comment text="show spring ad to men"/>
      <simpleCondition>
         <variable name="segment"/>
         <operator name="="/>
         <value data="men"/>
      </simpleCondition>
      <action name="awarenessAd">
         <parameter name="collateral" value="Spring">
         </parameter>
      </action>
   </rule>
   <rule>
      <comment text="show fall ad to women"/>
      <simpleCondition>
         <variable name="segment"/>
         <operator name="="/>
         <value data="women"/>
      </simpleCondition>
      <action name="awarenessAd">
         <parameter name="collateral" value="Fall">
         </parameter>
      </action>
   </rule>
</ruleSet>
```

基础结构元素

规则包

竞销组件使用 com.ibm.commerce.rule 包。此包提供的类可以帮助您将一组规则转换为 XML 格式或从 XML 格式进行转换。它还提供对调用规则集中的规则的支持。

表 5.

类 / 接口名称	描述
Action	Action 类提供了由规则的操作部分所使用的名称和参数的属性。参数具有名称和值，并且它们还可以具有子参数（这是可选的）。
ActionHandler	必须实现 ActionHandler 接口以便提供规则的操作部分的实现。有一个方法“performAction”，当规则的条件部分评估为真时将调用它。Action 类相应的实例将被传递给“performAction”方法的实现。

表 5. (续)

类 / 接口名称	描述
Rule	Rule 类提供的方法可以帮助您将 Rule 的实例转换成 XML 格式或从 XML 格式进行转换。Rule 类有三个属性。其中一个针对规则的条件部分。另一个针对规则的操作部分。最后一个用于注释。condition 属性必须是 com.ibm.commerce.condition 包中可以找到的 Condition 类的实例。Rule 类还具有方法“invoke”，它接受 Evaluator 接口和 ActionHandler 接口的实现。如果条件评估为真，则调用 ActionHandler 实现的“performAction”方法。
RuleConstants	RuleConstants 接口包含规则包使用的一些公共常量。
RuleSet	RuleSet 类提供的方法可以帮助您将 RuleSet 的实例转换成 XML 格式或从 XML 格式进行转换。它有一个属性，这个属性是 Rule 对象的数组。它还有一个 invoke 方法，这个方法接受 Evaluator 和 ActionHandler 接口的实现。当调用 invoke 方法时，它在 Rule 对象的数组内循环并逐个调用。

竞销使用规则包将竞销活动的规则部分保存和装入为 XML 格式，也可从 XML 格式进行保存和装入。它还帮助调用规则。`CampaignInitiativeEvaluateCmd` 任务命令提供 `ActionHandler` 接口的实现，该接口用于处理竞销活动操作。

条件包

规则包使用 `com.ibm.commerce.condition` 包。此包提供的类可以帮助您将一个布尔条件转换为 XML 格式或从 XML 格式进行转换。它还提供对评估条件的支持。

表 6.

类 / 接口名称	描述
AndListCondition	AndListCondition 类扩展 ConditionList 类以提供一个条件，此条件由 AND 布尔运算符所联接的一系列条件构成。
Condition	Condition 类是一个抽象类，它提供的方法可以帮助您转换到 XML 格式以及从 XML 格式进行转换。它还提供一个抽象方法，此方法可以用于评估条件。
ConditionConstants	ConditionConstants 接口提供由条件包使用的常量值。
ConditionList	ConditionList 类是一个抽象类，它扩展 Condition 类。它提供对由一系列条件所构成的条件的支持。
ConditionUtil	ConditionUtil 类提供很多静态方法，这些方法可以在条件评估期间使用。
Evaluator	必须实现 Evaluator 接口以便评估简单条件或开放条件。

表 6. (续)

类 / 接口名称	描述
OpenCondition	OpenCondition 类扩展 Condition 类以提供一个一般条件。此条件由一系列“名称值”对构成。
OrListCondition	OrListCondition 类扩展 ConditionList 类以提供一个条件，此条件由 OR 布尔运算符所联接的一系列条件构成。
SimpleCondition	SimpleCondition 类扩展 Condition 类以提供一个“变量、运算符、值”形式的条件。
TrueCondition	TrueCondition 类扩展 Condition 类以提供一个始终评估为真的条件。

规则包使用条件包将规则的条件部分保存和装入为 XML 格式，也可从 XML 格式进行保存和装入。它还帮助评估条件。

抽象 Condition 类有五个具体扩展。其中三个（AndListCondition、OrListCondition 和 TrueCondition）知道如何评估它们自己。其余两个（OpenCondition 和 SimpleCondition）需要 Evaluator 接口的实现来评估它们。CampaignInitiativeEvaluateCmd 任务命令提供 Evaluator 接口的实现，该接口用于评估竞销活动条件。

Blaze 规则项目

竞销活动的缺省实现使用 Blaze 规则项目，该规则项目帮助评估竞销活动规则。缺省情况下，所有的开放条件都将导致 **CampaignInitiativeEvaluateCmd** 任务命令调用 Blaze 规则项目。在以下目录中找到项目 `CampaignInitiativeEvaluator`:

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 *drive*`:\WebSphere\CommerceServer\rules\campaigns`

此项目的缺省实现能够评估所有由竞销活动用户界面创建的开放条件。您可以向此项目添加对附加开放条件的支持。

定制

预计我们的顾客将希望扩展竞销活动的缺省实现以添加更多的受支持的条件，并可能要添加附加操作和显示类型。

添加对新活动条件的运行时支持

评估竞销活动条件的 Blaze 项目接受 **CampaignInitiativeContext** 的一个实例。该上下文提供对命令上下文的访问以及对开放条件的名称和参数的访问。要添加对新开放条件的支持，您需要向 `conditionEvaluator` 规则添加新规则。此规则需要检查新条件的名称并评估该条件。在条件评估期间，您可以通过上下文对象按名称访问开放条件参数。

添加对新活动条件的构建时支持

为了使市场部经理能够使用新条件，需要扩展竞销活动用户界面以提供对新条件的支持。这可以通过向竞销活动向导或笔记本的“条件”面板

（WhenAddPanel.jsp）添加新的条件来实现。您还需要扩展 **CampaignInitiativeSaveControllerCmdImpl** 类以构造新的 Condition 对象。在以下目录中找到这个文件:

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

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

添加对新竞销活动操作的运行时支持

为了提供对新竞销活动操作的支持，您需要扩展 **CampaignInitiativeEvaluateCmdImpl** 类并覆盖 **performAction** 方法。如果操作的名称与新操作的名称匹配，则您应该执行此操作，否则您应该调用 super 方法执行缺省实现。

添加对新竞销活动操作的构建时支持

为了使市场部经理能够使用新竞销活动操作，您需要更新竞销活动向导或笔记本的“产品”面板（InitiativeWhatPanel.jsp）。您还需要扩展 **CampaignInitiativeSaveControllerCmdImpl** 类以构造新的 Action 对象。在以下目录中找到这个文件:

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

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

添加对新电子广告位类型的支持

如果您发现您需要提供附加输出类型，您可以扩展 `EMarketingSpot` 类以提供可以由 Page Designer 使用的新属性。您还需要扩展 **CampaignInitiativeEvaluateCmdImpl** 类以返回新的输出类型。

第 6 章 产品目录搜索

本章中讨论的元素描述了扩展现有搜索功能时所需的操作。附加功能包含对引入可由搜索组件访问的新属性和表的指导。对模式进行有助于减小运行时花费的优化，则还可以提高搜索组件的性能。该组件包含以下可定制的元素:

- 搜索 bean
- 搜索引擎
- 摘要表定义脚本

表 8 总结了由产品目录搜索 bean 搜索的数据库列的列表。由于各种不同的站点需求，您可能会发现有必要提供对本表中未描述的列的搜索。

定制方案

要向搜索 bean（在该搜索 bean 中这些属性已可以从搜索引擎中得到）添加搜索属性，请通过向搜索 bean 添加属性来实现方案 1。如果搜索引擎不支持对此属性的搜索，但是该属性在引擎已搜索的一个表中可用，请实现方案 2 来向引擎添加属性。如果此属性不在搜索引擎所搜索的某个表中，请实现方案 3 以向搜索引擎添加表。向搜索 bean 添加丰富属性的过程与向搜索 bean 添加其它任何属性的过程（实现方案 1）相同，但是在实现中有微小的不同。请参阅方案 4。要帮助改进运行时的性能，请为每个类别创建一个表以减小搜索的行数，以此实现方案 5 来修改和优化摘要表定义脚本。以下方案概括了何时定制该组件:

方案 1: 向搜索 bean 添加属性
向 bean 引入新属性，在该 bean 中此属性已在搜索引擎中可用。

方案 2: 向搜索引擎添加属性
向搜索引擎引入新属性，该搜索引擎已对该表进行了搜索。

方案 3: 向搜索引擎添加表
向搜索引擎引入新表。

方案 4: 向搜索 bean 添加丰富属性
向 bean 引入丰富属性。

方案 5: 利用已优化的摘要表改进性能
使用数据集的知识来优化摘要表定义。

注: 在文档中，对“搜索引擎”、“搜索接口”和“统一搜索框架”的称谓是同义的并且是可互换使用的。

方案 1: 向搜索 bean 添加属性

实现概述

您希望定制的第一件事情可能是要添加其它可搜索的属性。如果此属性也不可用（如方案 2 中所述），则该操作需要对搜索 bean 和搜索引擎做出更改（对搜索引擎的更改是可选的）。搜索 bean 的定制需要对其划分子类并扩展现有的方法。`samples/search` 目录中提供了有关如何实现此操作的样本代码。

定制步骤

此示例说明了如何向产品目录搜索 bean（该搜索 bean 的元数据类存在于 com.ibm.search.catalog 包中）添加附加的搜索属性。产品目录搜索数据 bean 是一个类。要定制它，请执行以下操作:

1. 创建 CatEntrySearchListDataBean 类的子类。
2. 为您希望添加的新的可搜索属性声明变量。您可以声明两种类型的变量来存储与可搜索属性相关联的信息。第一个类型可以用于存储属性的值。第二个类型可以包含可用在搜索中的有关信息（例如搜索运算符、布尔搜索、区分大小写）。这些变量中存储的信息可以用于生成 SQL 查询（它查询产品目录信息）。

   注: 用于生成 SQL 查询的数据库列必须在可用作生成查询中的约束之前，在搜索接口 RuleQuery 类或其子类之一中预先定义（参阅 “搜索接口的定制” ）。
3. 为已声明的变量创建访问者（get 和 set 方法）。
4. 创建 populate() 方法，它执行以下操作:
   a. 实例化 RuleQuery 或 RuleQuery 的子类。
   b. 使用 setRuleQuery(ruleQueryInstance) 方法将 RuleQuery 的此实例引用为父 bean 类中的实例。
   c. 调用 super.setPredefinedAttributes() 方法。
   d. 为使用搜索接口的新的可搜索约束制定 <Predicate>。
   e. 使用 super.setIsAllNull(false) 通知父类已添加了新的搜索属性。
   f. 调用 super.execute() 方法。

示例

此示例说明了如何向产品目录搜索 bean 添加新的可搜索属性 onSpecial（假定元数据类存在于 com.ibm.search.catalog 包中）:

1. 创建新的子类。
2. 为可搜索属性创建新的变量。

```
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;
  }
```

   onSpecial 变量用于存储可搜索属性的值。onSpecialOperator 变量用于存储搜索运算符（例如: like、=、<、>、<=、>=）。
3. 为所有的变量创建访问者

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

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

4. 创建 populate() 方法

```
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
}
```

方案 2: 向搜索引擎添加属性

实现概述

搜索引擎需要有关属性和表的信息以便正确地制定查询以从产品目录中检索结果。此信息作为元数据保存。要扩展搜索引擎以添加新的可搜索属性，需要附加属性元数据和表的元数据定义（表的元数据定义是可选的）。仅当将新表引入搜索引擎时才需要表的元数据，方案 3 中提供了进一步的详细信息。一旦完成此方案，您必须实现“方案 1: 向搜索 Bean 添加属性”以使其可用于 JSP Page Designer。samples/search 目录中提供了有关如何实现此操作的样本代码。

定制步骤

搜索引擎是统一搜索框架的一部分。它表示为一个称为 RuleQuery 的类和一些附加元数据类（例如分别描述列名和表名的 AttributeInfo 和 TableInfo）。此示例说明了如何通过为 com.ibm.search.catalog 包中没有的数据库列创建元数据类（但是列所属的表的元数据存在），来向搜索引擎添加新属性。要定制搜索引擎，请执行以下操作:

1. 为您希望使其可搜索的每个列和表定义“搜索接口”元数据。这需要以下操作:
   a. 每个可搜索的表必须具有一个对应的类（它是 TableInfo 类的子类）。该子类必须指定表名。
   b. 每个可搜索列必须具有一个对应的类（它是 AttributeInfo 类的子类）。该子类必须指定列名、列所属的 TableInfo 子类以及列的 SQL 数据类型。
2. 创建 RuleQuery 类的子类，并为每个新的数据库列定义静态整数引用。注意，整数值 0 到 1000 保留供系统使用。
3. 创建 findAttributeInfoName() 方法。要防止此方法覆盖父 findAttributeInfoName() 方法，建议您调用 super.findAttributeInfoName() 方法。如果覆盖了父 findAttributeInfoName()，则父类中定义的数据库列将不可用。
4. 向此方法添加工厂类创建逻辑以为每个新的数据库列创建 AttributeInfo 元数据类。
5. 修改 search.xml 文件，为任何新的可搜索表添加预定义的表联接关系。所有的表组合都需要联接关系。要修改 search.xml，请执行以下操作:
   a. 添加 COMMENT 部分以描述由此条目制定的表联接关系。
   b. 添加 KEY 部分以唯一地标识此条目。被联接的表用作唯一键。
   c. 添加 VALUE 部分以指定与键相关联的值。value 部分中的条目定义了联接关系。value 部分中的列名是使用该列的 AttributeInfo 子类定义的。

示例

以下示例将创建新的可搜索数据库列 CATENTRY.ONSPECIAL:

1.
   a. 创建 TableInfo 类的子类以指定表名 CATENTRY（如果此表的类不存在）:

```
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 ();

    /**
     * The constructor set's the table name.
     */
          public CatEntryTableInfo() {
             super("CATENTRY");
          }
    /**
     * The method returns the sigleton idiom.
     * @return com.ibm.commerce.search.catalog.CatEntryTableInfo
     */
          public static CatEntryTableInfo getSingleton()
          {
             return info;
          }
}
```

b. 创建 AttributeInfo 类的子类以指定列名 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());

   /**
    * The constructor set's the column name, column's table name, and datatype.
    */
   public CatEntryOnSpecialAttributeInfo(TableInfo info) {
      super(info);
      columnName = "ONSPECIAL";
      sqltype = SQLTYPE_NUM;
   }

   /**
    * The method returns the sigleton idiom.
    * @return com.ibm.commerce.search.catalog.CatEntryOnSpecialAttributeInfo
    */
   public static CatEntryOnSpecialAttributeInfo getSingleton()
   {
      return info;
   }
}
```

2. 创建搜索接口 RuleQuery 的子类:

```
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; //Static reference
}
```

3. 创建 findAttributeInfoName() 方法:

```
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. 如有必要，修改 Search.xml（在此示例中，不需要修改 search.xml 文件）

注: 要向搜索 bean 添加这个新的元数据，请遵循方案 1 中的步骤。但是您必须使用具有元数据信息的 RuleQuery 的子查询的实例（请参阅上面示例中的 CustomQuery）而不是 RuleQuery 类的实例。

方案 3: 向搜索引擎添加表

实现概述

要从搜索引擎当前不支持的新表中添加属性，您必须定义有关此表的信息，并描述此表与引擎使用的其它表如何相关。此信息是用于派生联接谓词的联接关系（联接谓词将每个表与其它表相关联）并作为元数据存储。要扩展搜索引擎以添加新的可搜索表，您必须定义每个表及其与其它表的关系。

您必须实现方案 2: “向搜索引擎添加属性”以使与此表有关的属性可用于搜索 bean 创建程序。samples/search 目录中提供了有关如何实现此操作的样本代码。

定制步骤

这些示例说明了如何为数据库表创建联接关系元数据类。通常情况下，您应该为每个逻辑表关系组合定义一个联接关系。

示例

此示例更改现有的 search.xml 以包含 CATENTRY 表和 OFFERPRICE 表之间的新联接关系条目:

1. 添加 COMMENT 部分:

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

2. 添加 KEY 和 VALUE 部分:

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

方案 4: 向搜索 bean 添加丰富属性

实现概述

向 bean 添加对丰富属性的搜索的过程与向 bean 添加任何其它可搜索属性的过程相同。一旦数据集是已知的并识别了任何丰富属性，请遵循方案 1 以使这些属性在搜索 bean 中可用。该方案与方案 1 之间唯一的不同在于用于构造丰富属性谓词的方法参数（有关详细信息，请参阅以下示例中以粗体字显示的文本）。搜索引擎已能够搜索在 ATTRIBUTE 和 ATTRVALUE 表中定义的丰富属性。

定制步骤

与定制方案 1 “向搜索 bean 添加属性”相同。

示例

此示例说明了如何向产品目录搜索 bean 添加新的可搜索丰富属性 color。

1. 创建新的子类。

2. 为可搜索属性创建新的变量。

```
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;
  }
```

变量 colorValue 存储可搜索属性的值。变量 colorValueOperator 存储搜索运算符（例如，like、=、<、>、<=、>=）。

3. 为所有的变量创建访问者

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

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

4. 创建 populate() 方法

```
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(); //Step 4f
}
```

方案 5: 利用已优化的摘要表改进性能

实现概述

当定义了数据集后，此知识可以用于创建更多优化的摘要表以显著改进搜索性能。目标是创建与要遇到的预期查询类型相类似的表。在大多数情况下，这些摘要表会小得多，所以应用于这些表上的查询要快于使用更大的底层原始表或其它次优的摘要表。

定制步骤

此示例说明了如何为每个类别组创建一个丰富属性类别组摘要表。如果有很多类别组（CATGROUPS）具有带丰富属性的商品，请为每个组都定义一个摘要表定义。

创建该摘要表的步骤:

1. 使用 select 查询定义表创建语句，它将用作摘要表定义。此 select 语句必须在“group by”子句中指定与结果集中列出的列相同的列。有关更多详细信息，请参阅 DB2 文档“自动摘要表（AST）”，或 Oracle 文档“物化视图”。
2. 在 select 语句中，为类别指定一个类别组谓词。
3. 用以上查询中指定的相同结果集列以及相同顺序创建索引。
4. 为每个类别组重复以上步骤。

示例

此示例对于三个类别组 10000、10001 和 10002 中的每一个，显示丰富属性类别组的摘要表创建脚本。粗体部分是新的（比起此示例基于的 wcs.summary.richAttributeCatGroup.create.sql 文件中显示的一般定义）。

```
// Summary table definition for 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;

// Summary table definition for 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;

// Summary table definition for 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;
```

产品目录搜索数据 bean 变量

必须通过 URL（JSP）传递给产品目录搜索 bean 的变量列表。

表 7.

名称	数据类型	描述
beginIndex	string	此变量用于标明结果集的页索引。该值必须是页面中首个结果行的索引。
catGroupId	string	此变量的值用于类别标识的搜索。
categoryTerm	string	此变量的值用于类别名和／或类别描述的搜索。
categoryTermCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
categoryTermOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
categoryTermScope	Integer	用户可以将搜索作用域限制为以下三种之一：名称、名称和简短描述，或名称、简短描述和详细描述。此变量中的值用于存储用户的选项。此值必须是以下三者之一：1（名称和简短描述），或 2（只有名称），或 3（名称、简短描述和详细描述）。
categoryType	string	用户可以指定三种类型的搜索条件：All，Any 或 Exact Phrase。此变量中的值用于存储用户的搜索条件。此值必须是 ALL（all 搜索条件）、ANY（any 搜索条件）或 EXACT（exact phrase 条件）。
currency	String	此变量的值用于货币的搜索。

表 7. (续)

currencyCaseSensitive	String	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
currencyOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
filterTerm	string	此变量中的值用于过滤指定值的搜索。
filterTermCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
filterTermOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
filterType	string	用户可以指定三种类型的搜索条件: All、Any 或 Exact Phrase。此变量中的值用于存储用户的搜索条件。此值必须是 ALL（all 搜索条件）、ANY（any 搜索条件）或 EXACT（exact phrase 条件）之一。
isListPriceOn	string	产品目录搜索 bean 用户可以选择列出标价或标准价格。要列出标价，请将此变量设置为 yes，要列出标准价格，请将此变量设置为 no。缺省情况下，搜索 bean 列出标准价格。
manufacturer	string	此变量的值用于生产商名称的搜索。
manufacturerCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
manufacturerOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
manufacturerPartNum	string	此变量的值用于生产商部件号的搜索。
manufacturerPartNumCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
manufacturerPartNumOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
maxPrice/minPrice	string	这些变量的值用于价格范围的搜索。
pageSize	string	此变量的值指定每页中要显示的搜索结果行的数目。
price	string	此变量的值用于价格的搜索。

表 7. (续)

priceOperator	string	用户可以选择以下一个运算符作为搜索运算符: =、<、>、!=、<=、>=。此变量中的值用于存储用户的选项。此值必须是以下任何一个: EQUAL、NOTEQUAL、GREATER、LESS、GREATER_EQUAL、LESS_EQUAL。
qtyAvailable	string	此变量的值用于产品或商品库存的搜索。
qtyAvailableOperator	string	用户可以选择以下一个运算符作为搜索运算符: =、<、>、!=、<=、>=。此变量中的值用于存储用户的选项。此值必须是以下任何一个: EQUAL、NOTEQUAL、GREATER、LESS、GREATER_EQUAL、LESS_EQUAL
qtyMeasure	string	此变量的值用于数量单位的搜索。
qtyMeasureCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
qtyMeasureOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
resultCount	string	此变量将包含搜索返回的结果总数。
resultType	string	商家可以指定他们是否希望在搜索结果中显示“产品”和/或“商品”。此变量中的值用于存储此值。此值必须是 1（只有产品）、2（只有商品）或 3（产品和商品）之一。
searchTerm	string	此变量的值用于词的搜索。
searchTermCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。
searchTermOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。
searchTermScope	Integer	用户可以将搜索作用域限制为以下四种之一: 名称、名称和简短描述、名称和简短描述以及详细描述、关键字。此变量中的值用于存储用户的选项。此值必须是 1（名称和简短描述）、2（只有名称）、3（名称、简短描述和详细描述）或 4（关键字）之一。
searchType	string	用户可以指定三种类型的搜索条件: All、Any 或 Exact Phrase。此变量中的值用于存储用户的搜索条件。此值必须是 ALL（all 搜索条件）、ANY（any 搜索条件）或 EXACT（exact phrase 条件）。
sku	string	此变量的值用于库存标识的搜索。
skuCaseSensitive	string	用户可以选择区分大小写或不区分大小写的搜索。此变量中的值用于标识搜索是否是区分大小写的。此值必须是 yes（区分大小写的搜索）或 no（不区分大小写的搜索）。

表 7. (续)

skuOperator	string	用户可以选择 like 或 equal 作为搜索运算符。此变量中的值用于存储用户的选项。此值必须是 LIKE（表示 like 运算符），或 EQUAL（表示 equal 运算符）。

产品目录搜索数据库列

产品目录搜索 bean 可以搜索的数据库列列表:

表 8.

bean 接口中定义的可搜索属性	表	列	支持布尔表达式
类别组 — 所有的名称和描述	CatGrpDesc	NAME LONGDESCRIPTION SHORTDESCRIPTION	是
类别组 — 只有标题	CatGrpDesc	NAME	是
类别组 — 标题和描述（推荐的缺省值）	CatGrpDesc	NAME SHORTDESCRIPTION	是
类别语言标识	CatGrpDesc	LANGUAGE_ID	否
类别标识	CatGpEnRel	CATGROUP_ID	否
产品目录条目描述 — 所有的名称和描述	CatEntDesc	NAME LONGDESCRIPTION SHORTDESCRIPTION	是
产品目录条目 — 只有标题	CatEntDesc	NAME	是
产品目录条目 — 标题和描述（推荐的缺省值）	CatEntDesc	NAME SHORTDESCRIPTION	是
产品目录条目语言标识	CatEntDesc	LANGUAGE_ID	否
产品目录条目生产商名称	CatEntry	MFNAME	否
产品目录条目生产商部件号	CatEntry	MFPARTNUMBER	否
产品目录条目库存标识	CatEntDesc	PARTNUMBER	否
价格	Listprice/ Offerprice	PRICE	否
价格范围	Listprice/ Offerprice	PRICE	否
货币	Listprice/ Offerprice	CURRENCY	否
可用的数量	InvStVw/ StoreInv	QTYAVAILABLE	否
数量单位	InvStVw/ StoreInv	QUANTITYMEASURE	否

通过定制 bean, bean 的用户可以列出更多列（例如丰富属性）。有关定制的更多信息, 请参阅联机帮助中 ExtendedCatEntrySearchListDataBean 类的 JavaDoc。

第 7 章 赠券

本章中讨论的元素代表 WebSphere 贸易加速器中“赠券”组件的用户界面。这些元素使创建和维护赠券、赠券提供以及卖方或商家每日执行的一些其它操作更为方便。赠券组件包含以下元素:

- “赠券”向导
- “赠券”笔记本

定制示例

以下示例概括了如何定制此部分的 WebSphere 贸易加速器。

方案 1: 在创建赠券折扣时添加新信息

实现概述

此方案将在为了记录赠券的附加信息（例如谁创建了赠券折扣）而定制“赠券”工具时所需的所有步骤中向您提供指导。

定制步骤

1. 实现新的 **CustomCreateCouponDiscountCmdImpl**，它扩展任务命令实现 **CreateCouponDiscountCmdImpl** 并实现 **CreateCouponDiscountCmd**。
2. 使用 CALCODE 表的 FIELD1 列存储创建此赠券折扣的用户的用户标识。
3. 在 CMDREG 表中注册新的任务命令。假定顾客商店标识是 1，则运行以下 SQL 语句。

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

4. 更新 **CustomCreateCouponDiscountCmdImpl**。在命令中，实现 performExecute() 方法:

```
public void performExecute() throws ECSystemException, ECException {
   super.performExecute();
   /** Get the userId from the command context
   Store the userId to table calcode
   **/
   }
```

第 8 章 折扣

本章中讨论的元素代表 WebSphere 贸易加速器中“折扣”组件的用户界面。这些元素使创建和维护折扣以及卖方或商家每日执行的一些其它操作更为方便。折扣组件包含以下元素:

- “折扣”向导
- “折扣”笔记本

定制示例

以下示例概括了如何定制 WebSphere 贸易加速器的折扣组件。

方案 1: 在创建折扣时添加附加信息

实现概述

如果您希望在创建折扣时添加附加信息，则此方案将在必需的所有步骤中指导您。此示例假定您希望创建对“谁创建了折扣”的审计跟踪。此方案不需要您提交新的参数。

定制步骤

1. 实现新的 **CustomCreateDiscountCmdImpl**，它扩展任务命令实现 **CreateDiscountCmdImpl** 并实现 **CreateDiscountCmd**。
2. 使用 **CALCODE** 表的 **field1** 列存储创建该折扣的用户的用户标识。
3. 在 **CMDREG** 表中注册新的任务命令。 假定顾客商店标识是 1，运行以下 SQL 语句:

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

4. 在 **CustomCreateDiscountCmdImpl** 中，实现方法 performExecute():

```
public void performExecute() throws ECSystemException, ECException {
  super.performExecute();
  /**  Get the userId from the command context
  Store the userId to table calcode
  **/
  }
```

方案 2: 更改缺省行为

实现概述

此方案更改折扣向导的缺省行为以便您可以指定向订单应用折扣时所用的顺序。此方案还需要您通过 URL 传递新参数。这需要向“折扣”向导的首个面板添加一个字段以获得额外的数据。

定制步骤

1. 通过执行以下操作，更新对应于“折扣”向导的 JSP 文件:

a. 切换至以下目录:

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 *drive*:\WebSphere\CommerceServer\wcstool.war\tools\promotions\

b. 将 DiscountWizardWelcome.jsp 复制为 MyDiscountWizardWelcome.jsp

c. 更新 MyDiscountWizardWelcome.jsp。在表单块中，添加以下代码:

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

d. 将条目保存到框架。在您的 JSP 中，有一个 JavaScript 函数 savePanelData，请在此函数中添加以下行:

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

e. 将新条目注册到 VIEWREG 表。例如，如果顾客商店标识是 1，运行以下 SQL 语句:

```
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. 实现新的 **DiscountSaveCmdCustomImpl**，它扩展控制器命令实现 **DiscountSaveCmdImpl** 并实现 **DiscountSaveCmd**。

3. 在 CMDREG 表中注册这个新的实现。假定顾客商店标识是 1，则运行以下 SQL 语句。

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

4. 在 **DiscountSaveCmdCustomImpl** 命令中，使用以下样本代码实现两个方法:

```
Public  void validateParameters() {
  super.validateParameters();
  /*
  Get your new parameters from requestProperty
  */
  }
Public void customMethod() {
  super.customMethod();
  /*
  Your action here.
  */
  }
```

第 9 章 RFQ 响应

定制示例

在此文档中，我们提供了三个定制样本。它们是:

- 复制响应
- 从 RFQ 响应的生命周期中删除“已撤销”状态
- 在创建期间输入响应的描述性信息

方案 1: 复制响应

此样本定制向 RFQ 响应工具添加功能，该工具使用户可以根据选定的 RFQ 响应创建复制的 RFQ 响应。定制包含添加**复制**按钮，当单击此按钮时，将提示用户输入复制响应的唯一名称。当创建完成之后，新的响应处于草稿状态。

实现概述

检索该响应的所有详细信息，并通过使用检索到的信息创建新的响应。因为在当前模式中响应名称不能相同，所以我们需要提供一个页面让用户提供另外的名称。这有助于准备此页面中的所有信息并将其传递给 **RFQResponseCopyCmd** 命令，此命令引起创建响应事件。然后，UBF 调用 **RFQResponseCreateCmd** 命令创建响应。

定制步骤

1. 通过执行以下操作，添加一个接口 **RFQResponseCopyCmd** 及其实现 **RFQResponseCopyCmdImpl**:

   a. 添加新的接口 **RFQResponseCopyCmd**。定义接口的代码如下:

```
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. 通过扩展 com.ibm.commerce.ubf.commands.ToolsBusinessFlowEventCmdImpl 并实现 **RFQResponseCopyCmd**，添加命令实现 **RFQResponseCopyCmdImpl**。此命令使用通过对话框提交的数据来触发创建响应事件。此事件导致 UBF 调用 **RFQResponseCreateCmdImpl** 来创建新的响应。定义命令的代码如下:

```
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,
              "The response was copied successfully!");
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,
        "Copying response failed, please input another name.");
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. 在 URLREG 表中注册 **RFQResponseCopy**。运行以下 SQL 注册此命令:

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

2. 添加一个 JSP 以使用户可以输入响应的名称，该 JSP 检索所有与此响应相关联的信息:

   a. 该 JSP 文件提供一个文本字段让用户提供新响应名称。以下是文本字段的源代码:

```
<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>
```

   当用户单击**确定**时，以下代码在输入字段中保存响应名称:

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

   当初始化此 JSP 文件时，以下代码检索所有与响应相关联的信息并保存它。信息将被提供给 **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. 运行以下 SQL 语句将一个视图命令注册到 VIEWREG 表，此表映射新的 JSP 文件。

```
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. 将名为 rfq_response_duplicate_dialog.xml 的新 XML 文件添加到以下目录:

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 *drive*:\WebSphere\CommerceServer\xml\tools\rfq

finishURL 是 **RFQResponseCopy**，后者映射已注册的 URL。此对话框的面板是 **RFQRspDuplicateDialog**，后者映射已注册的视图命令。XML 文件的代码如下:

```
<?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. 在 resource.xml 中注册此 XML。

1) 在以下目录中备份 resource.xml:

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 *drive*:\WebSphere\CommerceServer\xml\tools\rfq

2) 将以下两行添加到 resource.xml 文件:

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

e. 将名为 rfq_response_duplicate_dialog.js 的新 JavaScript 文件添加到以下目录:

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 *drive*:\WebSphere\CommerceServer\web\javascript\tools\rfq

此 JavaScript 文件包含对话框使用的一些 JavaScript 函数。以下是 JavaScript 文件的源代码:

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

function submitFinishHandler (finishMessage){
   alertDialog(finishMessage);
   top.goBack();
}

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

3. 将此函数集成到 RFQ 响应列表页面中。

a. 向响应列表页面添加一个**复制**按钮。即，如下更改 XML 文件:

注: 备份 XML 文件 rfq_response_list.xml，您可以在以下目录中找到它:

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 *drive*:\WebSphere\CommerceServer\xml\tools\rfq

在 rfq_response_list.xml 中的 <button>... </button> 节点的结尾处，添加以下行。

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

b. 在 rfq_response_list.jsp 中添加新的 JavaScript 函数 getDuplicateBCT() 和 duplicateRes():

```
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. 向 rfq_response_list.jsp 添加一个 JavaScript 函数并在相应的 RFQ 的状态不是 ACTIVE 时调用该函数以禁用**复制**按钮。向 rfq_response_list.jsp 添加以下函数并在调用 DisableNewButton() 之后调用它:

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

4. 在 RFQMessages_en_US.properties 文件中添加新的消息，您可以在以下目录中找到此文件:

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

drive:\WebSphere\CommerceServer\properties\com\ibm\commerce\tools\rfq\properties。

向文件添加以下行。当复制失败时将显示此消息。

```
_ERR_RESPONSE_COPY                  = Copying Response failed.
```

方案 2: 从 RFQ 响应的生命周期中删除“已撤销”状态

在原始提供的产品中，当用户撤消响应时，响应从 Active 状态更改为 Retracted 状态。如果用户希望在撤消响应之后直接将响应设置为 Draft 状态，则您必须通过从响应的生命周期中删除 Retracted 状态来定制此功能。

实现概述

在此定制中，我们首先需要从响应状态机中删除 Retracted 状态。一旦 Retracted 状态不再存在，则 Retracted 状态也应该从响应列表页面的视图列表中除去。

定制步骤

1. 更改 UBFStatemachine.xml 和 UBFStatemachine_en_US.xml 并重新装入它们。在以下目录中备份 UBFStatemachine.xml 和 UBFStatemachine_en_US.xml:

   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 *drive*:\WebSphere\CommerceServer\xmlloadutility\businessfollows\xml

   对这两个文件作以下更改:

   - 将 retractRFQResponse 转换的目标状态从 RETRACTED 更改为 DRAFT。
   - 删除有关 RETRACTED 状态的信息和从此状态开始的所有转换

   下面的代码样本说明了必需对 UBFStatemachine.xml 文件作出的更改:

   更改之前

```
<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>
```

更改之后

```
<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>
```

对 UBFStatemachine.xml 的更改如下所示:

更改之前

```
<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>
```

更改之后

```
<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. 因为在响应的生命周期中不再存在 Retracted 状态，所以在 rfq_response_list.xml 和 rfq_response_state_list.xml 中删除以下行。此处的状态包含 draft、cancelled、pendingapproval、rejected、active、inevaluation、win、lost、wincomplete 和 lostcomplete。

   注: 此处为了显示的目的将行分开了。

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

注:

1. 此样本说明了如何删除状态。如果希望添加状态，请执行相似的步骤，但是要反向执行:
   a. 在状态机 XML 文件中添加新的状态转换并重新装入它。
   b. 将新的视图添加到响应列表页面的视图列表中

   有关如何重新装入状态机的信息，请参阅 UBF 联机帮助。

方案 3: 在创建期间输入响应的描述性信息

如果用户希望在对 RFQ 创建一个响应时输入一些描述性的文本，您必须在 RFQ 创建向导的首个面板中添加一个文本字段。

实现概述

在 RFQ 响应创建向导的首个面板中添加新的文本字段。因为 **RFQResponseCreateCmdImpl** 不处理额外的输入，所以要通过扩展 **RFQResponseCreateCmd** 和 **RFQResponseCreateCmdImpl** 来创建新的接口 **MyRFQResponseCreateCmd** 及其实现（命令 **MyRFQResponseCmdImpl**）。

定制步骤

1. 添加接口 **MyRFQResponseCreateCmd** 及其实现 **MyRFQResponseCreateCmdImpl**。
   a. 添加新的接口 **MyRFQResponseCreateCmd**。接口的代码如下:

```
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. 通过扩展 **RFQResponseCreateCmdImpl** 并实现 **MyFQResponseCreateCmd** 来添加命令实现 **MyRFQResponseCreateCmdImpl**。添加新的字段 responseDescription 来存储从用户界面中传送来的响应描述。而且还要为此字段添加获取程序方法和设置程序方法:

```
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,"");
      }
   }
}
```

   覆盖 initParameters() 方法。此方法将由 **RFQResponseCreateCmdImp** 中的 checkParameters() 调用来初始化参数。此方法和对应的超类中的方法之间的不同在于 **setResponseDescription**

(MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION, false) 的调用获取从用户界面中传送来的响应描述。此方法的源代码如下:

```
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);

   //get response description from the user interface
   setResponseDescription(MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION, false);
}
```

覆盖 `createResponse()` 方法。此方法是完全相同的，除了您必须在 **RFQResponseAccessBean** 和 **TradingDescriptionAccessBean** 中设置描述字段。此方法的源代码如下:

```
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();
      //This id will be picked by BusinessFlowEventListener
      //to add a record in FLINSTANCE table

      this.setEntityObject(responseDB);

      //set this response reference number
      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. 运行以下 SQL 语句将 **MyRFQReponseCreate** 命令注册到 URLREG 表:

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

2. 通过扩展 `RFQConstants`，创建新的常量类 `MyRFQConstants`。仅有一个新的常量定义:

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

3. 更改 `rfq_w_response_general.jsp` 文件以添加新的文本字段和关联的处理。在以下目录中找到此文件:

   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 *drive*:\WebSphere\CommerceServer/web/tools/rfq

如下更改 Form 部分:

```
<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>
```

在 savePanelData() 函数中，添加代码以保存响应描述。更改后的函数如下:

```
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);
}
```

在 retrievePanelData() 函数中，添加代码以检索响应描述。更改后的代码如下:

```
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. 将 rfq_response_wizard.xml 文件中的目标 finishURL 从 RFQResponseCreate 更改为 MyRFQResponseCreate。在以下目录中找到这个文件:

   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 *drive*:\WebSphere\CommerceServer/xml/tools/rfq

   根据以下代码段进行更改。

```
<wizard resourceBundle="utf.utfNLS"
   windowTitle="matchlisttitle"
   finishConfirmation="ex_finish"
```

```
cancelConfirmation="ex_cancel"
finishURL="MyRFQResponseCreate"
tocBackgroundImage="/wcs/images/tools/toc/W_merchand.jpg">
```

5. 更改 UBFStatemachine.xml 文件并重新装入它。在以下目录中找到这个文件:

   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 *drive*:\WebSphere\CommerceServer\xmlloadutility\businessfollows\xml

   在这两个文件中将操作接口从 com.ibm.commerce.rfq.commands.RFQResponseCreateCmd 更改为 com.ibm.commerce.rfq.commands.RFQResponseCreateCmd。以下样本代码说明了必需的更改:

   更改之前

```
<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>
```

   更改之后

```
<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>
```

   注: 有关如何重新装入状态机的信息，请参阅 UBF 联机帮助。

第 10 章 预期库存

定制示例

以下示例概括了如何定制此部分的 WebSphere 贸易加速器。

方案 1: 在创建预期库存记录时添加新信息

此方案将在为了在预期库存中记录附加信息（例如谁创建了预期库存记录）而定制“预期库存”工具时所需的所有步骤中向您提供指导。

1. 为适当的 userID 向 RA 表添加一列。创建新的 ExpectedInventoryRecords 企业 bean 和访问 bean。
2. 实现新的 **CustomExpectedInventoryRecordCreateCmdImpl**，它扩展控制器命令实现 **ExpectedInventoryRecordCreateCmdImp** 并实现 **ExpectedInventoryRecordCreateCmd**。
3. 在 CMDREG 中注册新的控制器命令。假定顾客商店标识是 1，运行以下 SQL 语句:

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

4. 更新 **CustomExpectedInventoryRecordCreateCmdImpl**。在命令中，实现 performExecute() 方法:

```
public void performExecute()throws ECSystemException,ECException {
   super.performExecute();
   /**Get the userID from the command context
   Store the userID to the RA table
   **/
}
```

第 11 章 商务智能

动态上下文

动态上下文是用户可以选择执行的已分组的操作的列表。此列表包含操作的名称和简短描述。此列表会根据用户的角色以及启用的组件而动态更改。

定制示例

方案: 向现有的上下文添加新操作

要向现有上下文添加新的操作，请执行以下操作:

1. 向上下文定义 XML 文件添加一个操作。例如，xml/tools/bi 目录中的 biContext.xml 文件。您需要查找您希望对其作添加的上下文，然后向其添加一个条目。以下是“竞销”上下文的“竞销”操作:

```
<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>
```

要查看“竞销”上下文，您从 WebSphere 贸易加速器的**市场营销**菜单中选择**竞销**，然后单击**报表**。

在以上代码段中，toolsComponent 是一个可选的属性。如果指定，则仅当从配置管理器中启用了指定的组件时才列出操作。

如果用户是角色元素中定义的角色之一，则将为该用户列出此操作。

当从列表中选择操作时，命令元素的名称属性是执行的命令的名称。参数和值属性分别是命令参数和参数的值。

条目元素的 appendQueryString 属性没有在以上代码段中显示。如果属性显示并设置为真，则当前请求所请求的属性将以“名称值”对的格式附加到操作命令中。例如，下面的代码样本说明了帐户报表上下文中的“报表: 按帐户列出订单”操作:

```
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>
```

ContextEntry 类根据以上定义生成以下命令（命令在一行上，此处分开仅是为了显示目的）：

```
/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
```

在以上命令中，从请求属性中抽取以下“名称值”对并将其附加到命令：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. 向属性文件添加属性键。

   nameKey、descriptionKey 和 breadCrumbTrailTextKey 是属性文件中用于实现本地语言的键。以上示例的属性文件是 properties/com/ibm/commerce/tools/bi/properties/BINLS_en_US.properties。nameKey 映射为操作的名称。descriptionKey 映射为操作的描述。breadCrumbTrailTextKey 映射为选定了操作时附加到 breadcrumb trail 的文本。

第 12 章 报表框架的概述

报表框架几乎为站点的任何方面都提供一个一般化且可定制的报表功能。可以通过任何使用贸易加速器的角色来访问报表。可以定义对特定报表的访问，并将其限制在该报表内。WebSphere 贸易加速器用户可以在任何时候请求报表，框架使用在生产数据库中包含的数据生成报表并实时显示报表。

此框架由一个一般控制器命令、一个数据 bean 和一个用于显示结果的一般视图构成。您可以通过添加有效 SQL 查询并定义用于请求和显示已生成报表的 JSP 文件来定制框架。

定制报表框架

报表可以从贸易加速器中访问。因此，每个报表需要很多关联的有用资源。虽然报表自身由以表格格式表示的数据构成，但是底层的有用资源由报表标识、SQL 查询、访问控制元素等构成。报表请求在服务器上启动一个控制器命令。此控制器命令调用任务以设置一般视图，除非报表指定了特定的视图。此命令还设置很多必需的变量，并返回此数据以填充目标 JSP 文件中的 ReportDataBean。报表的访问控制在视图上设置，视图请求（输入）并显示报表。从数据库中返回的结果作为散列表向量存储在数据 bean 中。最终，JSP 文件显示报表。如果报表是空的，JSP 文件将转而显示一般文本字符串。

报表的访问控制设置在用于请求（输入）并输出报表的视图上。

添加新报表需要以下步骤:

1. 在 XML 文件中定义报表。
2. 创建从中请求报表的 JSP 文件（如果需要）。
3. 创建用于显示报表的 JSP 文件，除非使用一般 JSP。

定制示例

在 XML 文件中定义报表

单独的报表是使用 XML 文件定义的。每个报表具有一个对应的 *reportName*.xml 文件。此文件包含生成报表所必需的所有信息。*reportname*.xml 文件看上去类似于以下 MyStoreOverviewReport 的示例:

```
<?xml version="1.0" standalone="yes" ?>
<Reporting>
<!-- owner="ownerName" location="path_to_this_XML_file " -->
<!-- A Collection consists of SQLs for WCS reporting  -->

<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>
```

要创建新的报表，您必须创建一个类似于以上示例的 XML 文件。有关每个 XML 元素的详细解释，请参阅下面标题为『有效 XML 元素』的部分。

有效 XML 元素: 使用以下 XML 元素定义报表:

<Reporting></Reporting>

这是根元素。

<Report></Report>

该必需元素定义特定的报表。您可以通过包含多个 <Report> 元素在单个 XML 文件中定义多个报表。此元素有两个必需属性:

ReportName

一个字符串，它定义报表的唯一名称。

online 一个布尔值，它指定报表是否实时可用。当前，true 是唯一支持的值。

<comment></comment>

一个可选的元素，您可以在其中描述报表。此字符串不需要翻译。此元素仅可在现有的 <Report> 元素中定义。

<SQLvalue></SQLvalue>

一个必需的元素，它定义用于生成报表的 SQL 查询。尽管它是必需的，但是此元素可以为空。此元素仅可在现有的 <Report> 元素中定义。

<mergeOperation></mergeOperation>

一个可选的元素，它允许您将多个 SQL 查询组合为一个报表。它包含一个用逗号定界的 reportName 列表，reportName 指向其它 <Report> 元素的 reportName 属性。引用的报表中所有的 SQL 查询必须返回相同数量的列，并使用相同的列名（它标识散列表的键）。每个 SQL 查询都独立于其它查询。每个 SQL 查询产生它自己的散列表向量，并且在显示出来之前，这些向量彼此附加在一起以形成单个报表。

以上部分中的“商店概述”报表示例说明了 <mergeOperation> 元素的使用。最终报表的第一行来自一个 SQL 查询，第二行来自随后的查询。此元素仅可在现有的 <Report> 元素中定义。

<extended_object_class></extended_object_class>

一个可选的元素，它包含一个 Java 类名，该类名用于使用扩展的 Java 类来创建 SQL 语句。此类生成一个报表，然后将数据发送回报表控制中心。使用此元素创建递归报表。此元素仅可在现有的 <Report> 元素中定义。

<display></display>

一个可选的元素，它用于定义显示结果时使用的参数。此元素仅可在现有的 <Report> 元素中定义。

<standardInfo></standardInfo>

必需的元素，它用于将可以通过 ReportDataBean 中的获取程序访问的元素分组在一起。这些元素对于大部分报表是基本元素。此元素仅可在现有的 <display> 元素中定义。

<resourceBundle></resourceBundle>

必需的元素，它用于指定用于报表的属性文件。还必须在 reports/resources.xml 文件中引用此值。此元素仅可在现有的 <standardInfo> 元素中定义。

<title></title>

必需的元素，它用于指定报表的标题。此值必须是属性文件中的键。此元素仅可在现有的 <standardInfo> 元素中定义。

<message></message>

必需的元素，它用于显示与报表有关的消息。例如，它可以用于提供报表的描述。此值必须是属性文件中的键。此元素仅可在现有的 <standardInfo> 元素中定义。

<columnTitles></columnTitles>

必需的元素，它用于定义列标题。它包含一个逗号定界的名称列表。名称必须是属性文件中定义的键。如果在属性文件中未能找到此键，则元素中提供的此键将用作列标题。此元素仅可在现有的 <standardInfo> 元素中定义。

<userDefinedParameters></userDefinedParameters>

可选的元素，它用于定义报表框架中的定制元素。报表框架期望看见的元素格式为:

```
<element1>value1</element1>
<element2>value2</element2>
```

ReportDataBean 提供一个获取程序方法，它返回要在定制的显示 JSP 中使用的以上元素的散列表。此元素仅可在现有的 <display> 元素中定义。

注: 尽管 <display> 元素中包含的元素被列为必需，但是仅当定义了可选的显示元素时才确实如此。

SQL 查询中的变量: 当在 *reportName*.xml 文件中定义变量时，变量必须包含在花括号中（{*variableName*}）。这向报表框架指示此值是客户机变量并必须从客户机散列表中获得。在以上显示的样本 XML 文件中，{revenue}、{beginDate}、{endDate}、{storeent_id} 和 {orders} 都是客户机变量。

创建从中请求报表的 JSP 文件

根据生成报表所需的信息量，您必须确定是使用对话框还是使用向导来收集必需的数据。无论什么元素是适当的，JSP 必须包含 savePanelData JavaScript 函数:

```
function savePanelData()
{

var reportInputData = new Object();

   reportInputData.SQLid = "the requested report  name" ;
reportInputData.reportXML = "some file";
   reportInputData.variable1 = "some value 1";
   reportInputData.variable2 = "some value 2" ;
.
   .
   .
reportInputData.variableN = "some value N";
   reportInputData.varProperties = "a list of variable separated by a comma";
   parent.put("reportInputData", reportInputData);

// The section below can be used to indicate a different View to be used
// var reportResultPage = new Object();
// reportResultPage.cmd = "ASpecificDisplayReportView";
// parent.put("reportResultPage",reportResultPage);

   return true;
}
```

向控制器命令传递参数需要对 reportInputData 和 reportResultPage 的引用。SQLid 和 reportXML 变量也是必需的。variable1 到 variableN 以及 varProperties 是可选的。在示例中，variable1 到 variableN 代表 SQL 查询中使用的变量。例如，variable1 和 variable2 可以被替换为 beginDate 和 endDate。因此，以下代码将出现在 savePanelData() 函数中:

```
reportInputData.begindate = " some value";
reportInputData.enddate   = " some value";
```

varProperties 变量列出的变量从属性文件中获取它们的值。例如，它可能看起来类似于以下代码:

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

如果没有在 JSP 中引用对象 reportResultPage，则控制器命令将设置它使用报表框架提供的一般视图来显示报表。通过设置 reportResultPage.cmd，您可以指定要使用哪个视图。

下面的代码样本显示了用于收集报表输入数据的 JSP 文件的示例:

```
<!-- ============================================================================
 Licensed Materials - Property of IBM

  5724-A18

  (c) Copyright IBM Corp. 2001

  US Government Users Restricted Rights - Use, duplication or
  disclosure restricted by GSA ADP Schedule Contract with 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>

      ////////////////////////////////////////////////////////////////////
      // Call the initialize routines for the various elements of the page
      ////////////////////////////////////////////////////////////////////
      function initializeValues()
      {
         onLoadStartDateEndDate("enquiryPeriod");
         if (parent.setContentFrameLoaded) parent.setContentFrameLoaded(true);
      }

      ////////////////////////////////////////////////////////////////////
      // Call the save routines for the various elements of the page
      ////////////////////////////////////////////////////////////////////
      function savePanelData()
      {
         saveStartDateEndDate("enquiryPeriod");

         ////////////////////////////////////////////////////////////////////
         // Specify the report framework particulars
         ////////////////////////////////////////////////////////////////////
         setReportFrameworkOutputView("DialogView");
         setReportFrameworkParameter("XMLFile","reporting.OrderSummaryReportOutputDialog");
         setReportFrameworkReportXML("reporting.OrderSummaryReport");
         setReportFrameworkReportName("OrderSummaryReport");

         ////////////////////////////////////////////////////////////////////
         // Specify the report specific parameters and save
         ////////////////////////////////////////////////////////////////////
         setReportFrameworkParameter("StartDate", returnStartDateAsJavaTimestamp("enquiryPeriod"));
         setReportFrameworkParameter("EndDate", returnEndDateAsJavaTimestamp("enquiryPeriod"));
         saveReportFramework();
         return true;
      }

      ////////////////////////////////////////////////////////////////////
      // Call the validate routines for the various elements of the page
      ////////////////////////////////////////////////////////////////////
      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>
```

报表框架命令

报表框架使用与 WebSphere Commerce 一起提供的以下命令:

视图命令

表 9. 由报表框架使用的视图命令

视图名称	JSP 文件
ReportRedirectView	\tools\reporting\ReportRedirect.jsp
ReportGenericView	\tools\reporting\ReportGenericView.jsp

控制器命令

表 10. 由报表框架使用的控制器命令

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

有关更多信息，请参阅与 WebSphere Commerce 一起提供的有关以下包的 JavaDoc 帮助:

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

报表框架对象模型

在创建您的结果 JSP 文件时必须使用 ReportDataBean。populate() 方法包含支持实时报表的逻辑。以下方法在数据 bean 中可用。

表 11.

方法名称	返回类型	描述
populate()	void	运行 SQL 查询以生成报表
getErrorCode()	int	出错码的获取程序方法
getNumberOfColumns()	int	报表中列数的获取程序方法
getNumberOfRows()	int	报表中行数的获取程序方法
getColumnTitlesName(int)	string	(i+1) 列名的获取程序方法
getRow(i)	hashtable	(i+1) 行的获取程序方法
getValue(i.j)	string	报表中 (i+1,j+1) 值的获取程序
getValue(i,keyname)	string	与键名关联的列中 (i+1) 行的获取程序方法
getUserDefinedParameters()	hashtable	从 *reportName*.xml 中用户定义的参数中创建的散列表的获取程序方法
getEnv()	hashtable	包含了输入 JSP 文件中定义的输入参数的散列表的获取程序方法

有关更多信息，请参阅与 WebSphere Commerce 一起提供的 JavaDoc 帮助。

报表 JSP 文件的可重用组件

报表 JSP（它与报表框架协同工作）提供报表的输入和输出视图。

每个报表具有一个输入视图和一个输出视图。所有的输入 JSP 的任务或目的是相同的。它们具有相同的外观；然而它们从共享的输入小配件池中请求不同的输入条件。另一方面，输出视图具有公共的结构和格式。根据报表的性质，报表 JSP 所有的可重用部分都构建为可共享的组件。这些组件是根据以下文件结构和命名约定来构建的。

其中 *ReportName* 应该替换为新报表的名称，*InputComponent* 应该替换为输入组件的名称。所有的输入和输出报表 JSP 文件都利用 Reports.properties 来解析多语言问题。因此，属性文件映射 XML 和 JSP 文件中使用的所有标题和键。

这里是为 CSA 操作报表初始构建的组件列表:

ReportDaysWaitedHelper.jsp
"已等待天数" 可编辑框组件

ReportFulfillmentHelper.jsp
"实现选择" 组件

ReportInventoryAdjustmentCodeHelper.jsp
库存调整代码选择

ReportProductHelper.jsp
"选择产品" 组件

ReportStartDateEndDateHelper.jsp
"日期输入对" 组件

ReportVendorHelper.jsp
"供应商选择" 组件

ReportFrameworkHelper.jsp
输入 JSP 文件的公共函数

ReportOutputHelper.jsp
"输出页面格式化程序" 组件

ReportProductFindDialogView.jsp
"搜索条件输入" 页面

ProductSearch.jsp
"搜索结果选择" 页面

ReportProductFindView 和 ReportProductSearchView 是 ReportProductHelper 使用的独立页面。

Common.jsp 由报表输入和输出页面共享。ReportFrameworkHelper 由所有的报表输入 JSP 文件共享，ReportOutputHelper 适用于所有的报表输出页面。

其它所有的 JSP 文件都是输入组件并可以被拖动到需要输入的报表输入页面上。

每个报表需要三个 XML 文件。*ReportName*ReportDefinition.xml 定义用于报表的 SQL 语句和报表中每列的格式。有关如何编写 *ReportName*ReportDefinition.xml，请参阅 **Report Framework Design documentation**。

*ReportName*ReportInputDialog.xml 是一个对话框定义。它具有以下语法:

```
<】?xml version="1.0"?>

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

  <panel name="report"
    url="ReportNameReportInputView"
    hasFinish="YES"
    helpKey="CM.reports.ReportNameReportInputView.Help" />

</dialog>
```

其中 *ReportName* 应该替换为报表名。应该在 Reports.properties 文件中映射窗口标题。这样，reporting.reportStrings 在 xml/tools/reporting/resources.xml 文件中定义，并且它指向 properties/com/ibm/commerce/tools/reporting/properties/Reports.properties。帮助键在 CMHelpMap.xml 中映射。最终，URL 是报表输入视图 JSP 的视图命令名。

*ReportName*ReportOutputDialog.xml 是另一个对话框定义，它指定报表输出属性。它具有以下格式:

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

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

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

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

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

  </dialog>
```

以上的样本输出视图具有两个定制的按钮 — **打印**和**确定**。它们的处理函数是在输出视图 JSP 中定义的。同样，窗口标题在属性文件中映射，而帮助键在 XML 文件中映射。

这种设计是基于可重用组件概念的。输入视图 JSP 文件是从一组条件项中编写的，这取决于报表的规格。因为条件项可能出现在不同报表的输入页面中，所以每个条件项都被构建为可重用组件。将此策略应用于输入视图 JSP 页面设计，可达到以下目的:

1. 易于创建新的报表输入视图。
2. 所有的报表输入视图具有一致的外观。
3. 简化了验证、装入和保存“工具框架”所需的函数，因为这些函数内置于每个组件中。

输出视图 JSP 文件也构建在可重用帮助程序上。帮助程序根据数据类型和语言首选项提供所有必要的格式化和转换。每列的数据类型是在报表定义 XML 的用户定义部分中指定的。在支持复杂输出格式的同时，帮助程序和 XML 定义的协同工作使输出视图的创建更为简单。

下一个部分将解释如何使用可重用组件来创建报表输入视图和输出视图 JSP 文件。

要创建报表输入视图 JSP 文件，您必须将必需的组件导入到您的 JSP 文件中。以下组件是可用的:

1. DaysWaited — 指定直至今天已等待的天数。
2. FulfillmentCenter — 便于实现中心的选择。
3. InventoryAdjustment — 便于库存调整代码的选择。
4. StartDateEndDate — 指定时间段。
5. Vendor — 便于供应商的选择。

您可以创建其它组件，只要符合设计策略。我们将在稍后讨论帮助程序的创建。

所有这些组件为 JSP 页面开发者提供公共的接口:

1. JSP 函数:

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

   此函数在输入视图上创建组件。*InputComponent* 是组件的名称。每个组件具有一个 containerName，它是标识此组件的 JavaScript 对象的唯一名称。所有的 JavaScript 函数将引用 containerName。所有的组件需要报表资源束 reportsRB，它反映了当前语言首选项。每个组件至少具有一个从标号中映射的标题。

2. JavaScript 函数:

   function onLoad*InputComponent***(containerName)**

   当页面装入时将调用此函数。如果这是首次在事务中装入此页面，则将（从数据 bean 中）初始化 *InputComponent*。如果正在事务中重新装入此页面，则将检索保存的数据。

   function validate*InputComponent***(containerName)**

   在您提交请求之前应当调用此函数。它验证此组件的输入数据。如果数据无效，则弹出一个带有相应消息的对话框窗口提醒用户。如果数据有效，它返回真；如果数据的任何部分无效，则为假。

   function save*InputComponent***(containerName)**

   无论何时您浏览到不同于当前页面的另一页面，应调用此函数。它将保存组件的当前输入数据以供稍后在浏览回当前页面时向后检索。

   function visibleList(state)

   回调函数。当框架希望在页面上显示选择框或隐藏选择框时，将调用它。此函数应调用:

   setSelect*Component***Visible(container, state)**

   它在将在其中使用选择框的每个输入组件中定义。所有的输入组件具有相同的实现，但是名称不同:

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

报表输入 JSP 页面应该命名为 *ReportName*ReportInputView.jsp，它们应该位于以下目录中:

AIX

/usr/WebSphere/AppServer/installedApps/*ear_directory*/wctools.war/tools/

reporting

400

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

Linux

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

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

Windows

drive:\WebSphere\AppServer\installedApps*ear_directory*\wctools.war\tools\reporting。

它是一个对话框面板。要实现对话框面板，请参阅 Tools Framework User's Guide。对话框面板包含两个部分：HTML 内容生成部分，JavaScript 函数部分。在 HTML 部分中，您可以为您希望在输入屏幕上显示的每个组件调用 generate*InputComponent*。在 JavaScript 部分中，initializeValue、savePanelData 和 validatePanelData 应该调用每个输入组件中定义的相应 JavaScript 函数。当装入页面时，调用 initializedValue。"工具框架"将调用 savePanelData 和 validatePanelData。SavePanelData 应调用以下报表框架所需的 JavaScript 函数:

1. setReportFrameworkOutputView("DialogView");
2. setReportFrameworkParameter("XMLFile","reporting.*OutputPanelName*")
3. setReportFrameworkReportXML("reporting.*ReportDefinitionXML*");
4. setReportFrameworkReportName("*SQLName*");（在 *ReportXML* 中指定）

您还可以调用 setReportFrameworkParameter("*name*", *value*) 来设置报表输出 JSP 文件必需的参数。它是"名称值"对。传递给生成程序的所有标号以及 setReportFrameworkParameter 函数调用中的值都是键，它们将根据语言环境和语言首选项映射为正确的字符串。在位于以下目录的 Reports_en_US.properties 属性文件中定义了该映射:

AIX

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

400

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

Linux

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

Solaris

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

Windows

drive:\WebSphere\AppServer\installedApps*ear_directory*\properties\com\ibm\commerce\tools\reporting\properties

对报表输入和输出页面使用帮助程序

报表输入页面（它是一个对话框面板）必须具有一套组件，用于输入条件及实现下列四个 java 脚本函数:

initializeValues()

每次装入页面时调用。

savePanelData()

用户每次从此页面中退出时调用。

validatePanelData()

在向报表框架发送条件之前调用。

visibleList()

当报表框架需要更改此页面上组件的可视设置时调用。

要向报表输入页面添加组件，请导入组件 JSP 并在输入页面中添加一个 JSP 表达式。它按照命名约定命名为 generate*InputComponent*，并带有容器名称（对此页面唯一）、资源束以及一个或两个标题作为参数。作为示例，在您的输入页面中，您将有一些内容与以下类似:

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

表达式返回一个字符串，此字符串将在页面上生成可视的组件。为了使以上三个函数协同工作，用命名约定定义了回调函数: onLoad*InputComponent*、save*InputComponent* 和 validate*InputComponent*。它们应该分别在 initializeValue、savePanelData 和 validatePanelDate 中被调用。

SavePanelData 函数还保存报表框架的必需信息。每个输入组件提供返回函数，这些函数返回该组件中已输入的标识、名称或其它字段。有关这些返回函数的详细信息，请参阅任何输入组件 JSP。

输出页面负责处理格式化。所有的格式化方法都包含在 ReportOutputHelper 中，并与报表定义 XML 文件协同工作。在报表输出 JSP 文件中只需要指定报表名称。您可以复制任何报表输出 JSP 文件并修改 reportPrefix 值以反映您的报表名称。

然而，报表定义 XML 文件根据列类型指定所有的列及其格式化。以下是列类型的列表:

表 12.

列类型	是缺省值吗	可定制属性	缺省对齐方式
string	是	maxEntryLength	右对齐
integer	否	setMinimumIntegerDigits setMaximumIntegerDigits	左对齐
decimal	否	setMinimumIntegerDigits setMaximumIntegerDigits setMinimumFractionDigits setMaximumFractionDigits	左对齐
currency	否	currencySymbolColumn	左对齐
enumeration	否		右对齐

表 12. (续)

列类型	是缺省值吗	可定制属性	缺省对齐方式
date	否		左对齐
time	否		左对齐
month	否		左对齐

缺省列类型是 string，带有 HTML 列选项 "align=left height=20 nowrap"。通过指定列中 <columnOptions> 标记，所有的列类型可以覆盖缺省列选项。

可选的，所有的列还可以将它们的 displayInReport 值设置为 true 或 false。缺省值是 true，这意味着在报表中显示此列。如果此值设置为 false，则此列将被隐藏。此特征可以用于定制报表输出视图，而无需更改 SQL 查询。在格式化货币需要一个引用列来告诉格式化程序它是何种货币时，这也是有用的。

integer、decimal、date 和 time 列是根据命令上下文中指定的语言和货币值来格式化的。integer 和 decimal 列还可以为整数和小数值指定最小位数和最大位数。

缺省情况下，currency 列是根据命令上下文中指定的语言和货币值来格式化的。如果在currecy 列中指定了 currencySymbolColumn，则将从数据库中检索三字符的货币符号并用于格式化货币。如果报表创建者不希望显示货币符号字符串，则引用的货币符号列可以设置为不可见。

enumeration 是一种特殊的列类型，它将从数据库中检索到的值映射为由键指定的字符串。例如，Y 或 N 可以从表中检索到。这些值可以在不同的报表中或以不同的语言映射为更具含义的字符串，例如 Yes 或 No，以及 Approved 或 Denied。要使这成为可能，可以如下定义列:

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

or

```
<columns>
    <columnKey>C2</columnKey>
    <columnName>yyyColumnTitle</columnName>
    <columnType>enumeration</columnType>
        <Y>Approved</Y>
        <N>Denied</N>
</columns>
```

其中 Yes、No、Approved 和 Denied 的字符串在相应的属性文件中定义以用于多种语言。如果您希望对查询返回的诸如 0、1、2 等（数字）的值映射特定值，则您需要将 <X_n></X_n> 用作一个元素。例如:

```
<columnType>enumeration</columnType>
   <X_0>ValueFor0</X_0>
   <X_1>ValueFor1</X_1>
```

利用可重用 JSP 页面组件编写报表

要通过利用可重用 JSP 页面组件编写报表，您必须创建以下文件:

- JSP 文件:
  XXXReportInputView.jsp
  XXXReportOutputView.jsp
- XML 文件:
  XXXReportInputDialog.xml
  XXXReportDefinition.xml
  XXXReportOutputDialog.xml
- 更新的属性文件:
  Reports_en_US.properties，在其中您必须为所有必要的映射添加一个部分。
- 向数据库添加视图命令
  XXXReportInputView 和 XXXReportOutputView 命令必须被添加到数据库。
- 在先前步骤中添加的两个视图（XXXReportInputView 和 XXXReportOutputView）上设置访问控制。

第 13 章 产品顾问

定制示例

在此文档中，我们提供了三个定制样本。它们是:

- 更改样本“产品探测”JSP 文件以使用不同的运算符图标。
- 更改“产品比较”中使用的缺省链接。
- 描绘“产品探测”值而不使用提供的小配件。

方案 1: 使用不同的运算符图标

在“产品探测”暗喻符中，在运算符图标上（一个图像代表所有运算符）选择的 X 坐标用于确定选择了哪个运算符。当替换图像时，请确保坐标保持不变，这样单击图像可以标识正确的运算符。图像文件在 CommerceDir\web\tools\pa\icons 中。equalone.gif 代表 =、<> 运算符，equaltoo.gif 代表用于数字属性值的 =、<>、<=、>= 运算符。预期的 X 坐标是:

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

当值的选择传递 X 坐标参数时，相应的运算符将应用于该选择。

方案 2: “产品比较”暗喻符中的链接

“产品比较”暗喻符支持从单独的商品链接到其它页面。当前仅支持一个其它的页面（除非您对值用 URL 创建一个属性）。样本 JSP 链接到 ProductDisplay 页面。通过 ProductCompareDataBean 上的 productLinkName 属性指定此页面。在样本中，这是通过 ClickInfo 命令重定向的，此命令收集有关“产品比较”暗喻符使用情况的统计信息。当链接到其它页面时，可能需要从“产品比较”页面传递参数。在 ProductCompareDataBean 的 productLinkParameters 属性中标识要传递的参数。此处标识的任何参数名称将让参数传递到链接页面（如果参数存在）。另外，将此链接与之关联的商品的产品目录条目标识（catentry_id）的值指定给 productId参数（样本中称为 ECConstants.EC_PRODUCT_ID）。productId 使链接页面能够识别用户选择的商品。当前不支持其值对“产品比较”表中的单独商品唯一的其它参数。

方案 3: 产品探测描绘的定制

ProductExploreDataBean 由 DynamicForm 小配件描绘。要自己进行描绘，请生成 DynamicForm 类的子类（有关方法签名，请参阅 JavaDoc）并覆盖描绘方法，或直接从 ProductExploreDataBean 中获取数据。在描绘方法中，使用 getDataBean() 方法获取 ProductExploreDataBean 对象。使用 ProductExploreDataBean.getFormElements() 方法获取代表每个属性列的 ColumnDataBeans 的集合。在每个列中，ColumnDataBean 对象包含该列的值。使用 ColumnDataBean.getColumn() 方法获取包含单独属性值的 DsData 对象的集合。有关方法信息，请参阅有关 ColumnDataBean 和 DsData 的 JavaDoc。使用 DsData 的 getPresentationString() 方法检索属性值的格式化表示，使用

getUnformattedData() 方法获取原始数据。要为 HTML 表单创建正确的参数，请使用 ColumnDataBean.getFormElementName() 方法检索正确的参数名，使用 DsData.getUnformattedData() 方法获取每个值。有关正确的运算符选择，请参阅运算符图标信息。

第 14 章 规则项目

规则项目的生命周期包含以下阶段:

1. 创建规则项目
2. 根据新的规则项目配置规则服务
3. 调用规则服务
4. 根据规则项目除去规则服务

假定: 此处的信息仅涉及 2、3 和 4。我们假定规则项目已创建。

有关如何创建规则项目的信息，请咨询 Blaze 专业服务。

如何根据定制的规则项目配置规则服务

规则系统在 WebSphere Commerce 附带的规则项目和定制的规则项目之间不作区分。您可以始终使用管理控制台创建、修改规则服务或从规则项目中除去规则服务。然而，定制的规则项目必须满足某些需求。需求是:

1. 传递到规则项目中的外部事件必须触发规则项目的执行
2. 外部事件必须实现 `com.ibm.commerce.rules.InvocationContext` 接口。此接口仅是一个简单标记接口。

如何根据定制的规则项目调用规则服务

您可使用从随 WebSphere Commerce 附带的规则项目中创建的规则服务的同一样式，调用从满足上述两个条件的规则项目中创建的规则服务。即，通过调用以下任务命令: `com.ibm.commerce.rules.commands.InvokePersonalizationRuleServiceCommand`。

第 2 部分 附属资料

声明

本信息是为在美国提供的产品和服务编写的。

IBM 可能在其它国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代理咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

本出版物中任何对 IBM 许可程序的引用并非意在明示或暗示只能使用 IBM 的许可程序。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以用来代替 IBM 的产品、程序或服务。在与其它产品结合使用时，除了那些由 IBM 明确指定的产品之外，其评估和验证均由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可证。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504–1785
U.S.A.

有关双字节（DBCS）信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

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

本条款不适用联合王国或任何这样的条款与当地法律不一致的国家或地区:

国际商业机器公司以“按现状”的基础提供本出版物，不附有任何形式的（无论是明示的，还是默示的）保证，包括（但不限于）对非侵权性、适销性和适用于某特定用途的默示保证。某些国家或地区在某些交易中不允许免除明示或默示的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本出版物的新版本中。IBM 可以随时对本出版物中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。该 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其它程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Ltd.
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
Canada

只要遵守适当的条件和条款，包括某些情形下的一定数量的付费，都可获得这方面的信息。

本资料中描述的许可程序及其所有可用的许可资料均有 IBM 依据 IBM 客户协议、IBM 国际程序许可证协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其它操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其它可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其它关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息仅用于规划的目的。此处的信息在所描述的产品变得可用之前将会面临更改。

此信息包含日常业务运作中使用的数据和报表的示例。为尽可能表述完整，这些示例包含人名及公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际企业使用的名称和地址有任何雷同，实属巧合。

本产品中提供的信用卡图像、商标和贸易名称，应当仅供已由信用卡标记的所有者授予通过该信用卡接受支付的权限的商家使用。

商标

以下术语是国际商业机器公司在美国和／或其它国家或地区的商标或注册商标:

Blaze Advisor 是 HNC Software Inc. 在美国和／或其它国家或地区的商标。

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和／或其它国家或地区的商标或注册商标。

Netscape 是 Netscape Communications Corporation 在美国和 / 或其它国家或地区的注册商标。

Oracle 是 Oracle Corporation 在美国和 / 或其它国家或地区的商标或注册商标。

Java、JavaBeans 和所有基于 Java 的商标和徽标是 Sun Microsystems, Inc 的商标或注册商标。

其它公司、产品或服务名称可能是其它公司的商标或服务标记。

中国印刷