IBM® WebSphere® Commerce

WebSphere Commerce アクセラレーター カスタマイズ・ガイド

バージョン 5.4

IBM® WebSphere® Commerce

WebSphere Commerce アクセラレーター カスタマイズ・ガイド

バージョン 5.4

ご注意!

本書および本書で紹介する製品をご使用になる前に、 97 ページの『特記事項』に記載されている情報をお読みください。

本書の内容は、新版で特に指定のない限り、IBM® WebSphere Commerce バージョン 5.4 以降のすべてのリリースおよびモディフィケーションに適用されます。製品のレベルにあった版を使用していることをご確認ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本マニュアルに関するご意見やご感想は、次の URL からお送りください。今後の参考にさせていただきます。

`http://www.ibm.com/jp/manuals/main/mail.html`

なお、日本 IBM 発行のマニュアルはインターネット経由でもご購入いただけます。詳しくは `http://www.ibm.com/jp/manuals/` の「ご注文について」をご覧ください。

(URL は、変更になる場合があります)

原 典 :	IBM® WebSphere™ Commerce WebSphere Commerce Accelerator Customization Guide Version 5.4
発 行 :	日本アイ・ビー・エム株式会社
担 当 :	ナショナル・ランゲージ・サポート

第1刷 2002.5

この文書では、平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、平成角ゴシック体™W5、および平成角ゴシック体™W7を使用しています。この(書体*)は、(財) 日本規格協会と使用契約を締結し使用しているものです。フォントとして無断複製することは禁止されています。

注* 平成明朝体™W3、平成明朝体™W9、平成角ゴシック体™W3、
平成角ゴシック体™W5、平成角ゴシック体™W7

© Copyright International Business Machines Corporation 2001, 2002. All rights reserved.

© Copyright IBM Japan 2002

目次

© Copyright IBM Corp. 2001, 2002

始めに

本書の表記規則

本書では、以下のような強調表示の規則を使用しています。

太文字は、コマンドまたはグラフィカル・ユーザー・インターフェース (GUI) のコントロール (フィールド、ボタン、またはメニュー選択項目の名前など) を表します。

モノスペース (Monospace) は、そこに示されているとおりに入力するテキストの例、およびディレクトリーのパスを表します。

イタリック は、強調のため、および実際に使用する値に置き換える変数を表すために使用されます。

このアイコンはヒントを表します。これは、タスクを完了するのに役立つ可能性のある追加情報です。

必要とされる知識

WebSphere Commerce アクセラレーターをカスタマイズするには、以下のことに関する知識が必要です。

- HTML および XML
- 構造化照会言語 (SQL)
- Java プログラミング

WebSphere Commerce のカスタマイズに関する詳細については、 WebSphere Commerce プログラマーズ・ガイドを参照してください。この資料は以下の Web サイトから入手できます。

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

本書の構成

以下の表に、本書の構成について示します。

表 1.

コンポーネント	説明	場所
ビジネス関係管理	このセクションでは、契約ツールをカスタマイズする際の考慮事項について、詳細に説明します。	3 ページの『第 1 章 ビジネス関係管理』
顧客サービス担当者	このセクションでは、顧客サービス担当者のツールをカスタマイズする際の考慮事項について、詳細に説明します。	9 ページの『第 2 章 顧客サービス担当者ツール』

© Copyright IBM Corp. 2001, 2002

表 1. (続き)

コンポーネント	説明	場所
顧客プロファイル	このセクションでは、顧客プロファイル・ツールをカスタマイズする際の考慮事項について、詳細に説明します。	21 ページの『第 4 章 顧客プロファイル』
キャンペーン	このセクションでは、キャンペーン・ツールをカスタマイズする際の考慮事項について、詳細に説明します。	29 ページの『第 5 章 キャンペーン』
クーポン	このセクションでは、クーポン・ツールをカスタマイズする際の考慮事項について、詳細に説明します。	53 ページの『第 7 章 クーポン』
割引	このセクションでは、割引ツールをカスタマイズする際の考慮事項について、詳細に説明します。	55 ページの『第 8 章 割引』
RFQ	このセクションでは、 RFQ ツールをカスタマイズする際の考慮事項について、詳細に説明します。	59 ページの『第 9 章 RFQ 応答』
在庫	このセクションでは、在庫ツールをカスタマイズする際の考慮事項について、詳細に説明します。	71 ページの『第 10 章 予測在庫』
ビジネス・インテリジェンス	このセクションでは、レポート・ツールをカスタマイズする際の考慮事項について、詳細に説明します。	73 ページの『第 11 章 ビジネス・インテリジェンス』
レポート・フレームワーク	このセクションでは、レポート・ツールをカスタマイズする際の考慮事項について、詳細に説明します。	75 ページの『第 12 章 レポート・フレームワークの概要』
検索	このセクションでは、カタログ検索ツールをカスタマイズする際の考慮事項について、詳細に説明します。	37 ページの『第 6 章 カタログ検索』
Blaze Rules Advisor	このセクションでは、 Brokat Advisor ルール・プロジェクトをカスタマイズする際の考慮事項について、詳細に説明します。	93 ページの『第 14 章 ルール・プロジェクト』
商品アドバイザー	このセクションでは、商品アドバイザー・ツールをカスタマイズする際の考慮事項について、詳細に説明します。	91 ページの『第 13 章 商品アドバイザー』

第 1 部 WebSphere Commerce アクセラレーターのカスタマイズ

本書は、 WebSphere Commerce アクセラレーターのカスタマイズの方法について説明します。本書では、 WebSphere Commerce アクセラレーターの設計上の決定に関する背景情報が提供されており、カスタマイズへのアプローチやカスタマイズに必要なステップについて詳細に説明します。

本書は WebSphere Commerce アクセラレーターをカスタマイズする際の全体の流れについて案内することを意図したものであって、可能なあらゆるカスタマイズ方法を網羅するものではありません。その代わり本書では、 WebSphere Commerce アクセラレーターの一般構成要素について紹介し、それらの部分をカスタマイズする方法について説明します。

WebSphere Commerce アクセラレーターは、サイトのビジネス・オペレーションを支援するために設計されたツールを集めたものです。ビジネスに携わる人が、そのサイトでのオペレーションを担当する IT スタッフと継続的に連絡を取らなくても、そのサイトでのオペレーションのあらゆる面を作成および変更できるようにするインターフェースとして意図されています。したがって、 WebSphere Commerce アクセラレーターを構成する非常に多くのコンポーネントは、ビジネスの重要な面のそれぞれを対象としたものになっています。これらのツールは汎用的なものになるよう努力が払われているものの、その一方でこのような同一の目標により、ユーザーによっては、特定の要件が満たされていない場合があるかもしれません。自分のサイトの作成時にデータベースを拡張しており、それらの拡張が本書に示されているツールのいずれかに関連している場合には、その関連するツールをカスタマイズするまで、 Accelerator はそのデータを認識しません。

本書では、 WebSphere Commerce アクセラレーター内の以下のコンポーネントをカスタマイズする方法について説明します。

© Copyright IBM Corp. 2001, 2002

第 1 章 ビジネス関係管理

この章で説明するエレメントは、 WebSphere Commerce アクセラレーター内のアカウント・コンポーネントおよび契約コンポーネントのユーザー・インターフェースを表します。これらは、アカウント、契約、およびセールス・マネージャーまたは会計担当者がビジネスに関連して実行する必要のある日常的なタスクに関係したその他のアクションの作成と保守を支援します。ビジネス関係管理コンポーネントには、以下のエレメントが含まれます。

- アカウント・ノートブック
- 契約ノートブック

カスタマイズの例

以下の例で、 WebSphere Commerce アクセラレーターのこの部分をカスタマイズする方法を概説します。

シナリオ 1: 契約ユーザー・インターフェースへの新しい条件の追加

インプリメンテーションの概説

WebSphere Commerce アクセラレーターを汎用的なものにし、できるだけ多くの人にとって有用なものにするため、いくつかの条件が省かれています。そのため、自分のビジネスに適用できるようにするために、自分のサイトに条件を追加しなければならない場合があります。このシナリオでは、自分のサイトにさらに条件を追加するのに必要なステップを紹介します。

カスタマイズ・ステップ

1. サーバー内の条件を定義します。ユーザー・インターフェースのカスタマイズを実行する前に、以下のことが完了していなければなりません。
   - `B2BTrading.dtd` ファイル内に新しい条件を定義します。詳細については、*IBM WebSphere Commerce プログラマーズ・ガイド* の第 7 章を参照してください。
   - 新しい条件のエンタープライズ bean またはアクセス bean を作成します。
2. ユーザー・インターフェースに条件を追加します。必要なエレメントをユーザー・インターフェースに追加するには、以下のようにします。
   a. 条件のユーザー・インターフェース・ページの JSP ファイルを作成します。このページには、ページの入力フィールドからデータを取り込む動的 JavaScript オブジェクトを組み込む必要があります。アクセス bean からのデータのロードをカプセル化するデータ bean を作成します。オプションとして、データベースからの条件データをロードするために、アクセス bean をページを直接に組み込むこともできます。データ・ロード・パフォーマンスを軽減させるために、 `contractId` パラメーターがすべてのページに渡されます。以下に、この JSP ファイルのサンプルを示します。

© Copyright IBM Corp. 2001, 2002

```
<!--=========================================================================
//*-------------------------------------------------------------------------------------
//* 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. 条件のためのユーザー・インターフェース・ページの JavaScript ファイルを作成します。妥当性検査する関数を作成し、 JavaScript データを送信します。データを送信する場合、新しい条件の XML 形式と一致する新しい JavaScript オブジェクトを作成する必要があります。以下に、この JavaScript ファイルのサンプルを示します。

```
//*-------------------------------------------------------------------------------------------
//* 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. ステップ 2b で作成した新しい submit および validate 関数を呼び出すために、 Contract.js ファイルを変更します。

d. 必要な新しいストリングを追加するために、リソース・バンドルを変更します。

e. 新しいパネルを追加するために、 ContractNotebook.xml を変更します。

シナリオ 2: 契約ユーザー・インターフェースからの条件の除去

インプリメンテーションの概要

いくつかの条件が足りないと感じる場合だけでなく、いくつかの条件が自分のビジネスでは必要ないと感じる場合があるかもしれません。このシナリオでは、関係のない条件を自分のサイトから除去するのに必要なステップを紹介します。

カスタマイズ・ステップ

ContractNotebook.xml ファイルを変更し、除去する条件のパネル・エレメントを含むセクションを除去します。このファイルは以下のディレクトリーにあります。

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 顧客サービス担当者ツール

この章で説明するエレメントは、 WebSphere Commerce アクセラレーター内の顧客サービス担当者 (CSR) に関係したツールのユーザー・インターフェースを表します。これらは、顧客、オーダー、および CSR がビジネスに関連する日常のタスクを実行しなければならない他のアクションの作成と保守を支援します。 CSR ツールのコンポーネントには、以下のエレメントが含まれます。

- 顧客ウィザード
- 顧客ノートブック
- オーダー・ウィザード
- オーダー・ノートブック

カスタマイズの例

以下の例で、 WebSphere Commerce アクセラレーター内の CSR ツールをカスタマイズする方法の概要を説明します。

シナリオ 1: 追加の顧客データの「Business Customer (ビジネスの顧客)」要約ダイアログへの追加

インプリメンテーションの概説

この例では、ビジネスの顧客の従業員番号を「Business Customer (ビジネスの顧客)」要約ダイアログへ追加する方法について示します。

カスタマイズ・ステップ

`ShopperSummaryB2BDialog.jsp` を拡張します。このファイルは以下のディレクトリーにあります。

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¥`

以下の例は、 JSP ファイルを変更して、要約ダイアログ内の従業員番号を表示する方法について示しています。従業員番号は `OptoolsRegisterDataBean` のプロパティーです。サンプルでは、ラベル `Employee Number` を保管するプロパティー・ファイルは使用されていません。以下に、顧客の要約ダイアログに従業員番号が表示されている出力例を示します。

```
<%--
//*************************************************************************
//*-------------------------------------------------------------------------
//* Licensed Materials - Property of IBM
//*
//* 5724-A18
```

© Copyright IBM Corp. 2001, 2002

```
//*
//* (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>
```

シナリオ 2: 顧客サービスでのショッピング・カートの使用可能化 - オーダー管理インターフェース

インプリメンテーションの概説

ストア設計者が CSR ツールを使用して、状態 P のショッパーのオーダーを管理する場合、 **CSROrderSearchB2B** および **CSROrderSearchB2C** ビューを変更しなければなりません。

カスタマイズ・ステップ

CSROrderSearchB2B.jsp および CSROrderSearchB2B.jsp JSP ファイルを拡張します。オーダーの状態の基準に「保留」オプションを追加することにより、 CSR で顧客の保留中のオーダーを検索し、管理できます。保留中のオーダーとは、ショッピング・カートに入っているオーダーのことです。これらのファイルは両方とも以下のディレクトリーにあります。

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¥

以下のコード・サンプルを参照してください。

```
<%--
//************************************************************************
//*------------------------------------------------------------------------
//* 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 コラボレーション・ワークスペース

この章で説明するエレメントでは、 QuickPlace のカスタマイズおよび WebSphere Commerce 付属の ToolTech テンプレートの作成に必要なステップを示すケース・スタディーを紹介します。ユーザーがコラボレーション・ワークスペースに参加するように招かれた時に送信される、電子メール (E メール) の内容をカスタマイズするのに必要なステップについても説明します。

カスタマイズの例

以下の例で、 WebSphere Commerce アクセラレーターのこの部分をカスタマイズする方法を概説します。

注: シナリオ 1 と 2 は、密接に関係しています。シナリオ 1 を実行する際、ほとんどの場合に、シナリオ 2 も続けて実行します。

シナリオ 1: ワークスペースのルック・アンド・フィールのカスタマイズ

インプリメンテーションの概説

コラボレーション・ワークスペース・フィーチャーで最初にカスタマイズするのは、作成されているコラボレーション・ワークスペースの「ルック・アンド・フィール」です。ルック・アンド・フィールをカスタマイズするには、 PlaceType というテンプレートを作成しなければなりません。カスタマイズの詳細については、レッド・ブック *Customizing QuickPlace* を参照してください。

カスタマイズ・ステップ

この例では、 WebSphere Commerce 5.4 に付属している ToolTech のサンプル・ストアを使用します。この例は、 QuickPlace を作成する方法について示しています。

1. 以下を行って、デフォルトの QuickPlace を作成します。
   a. QuickPlace サーバーが実行中かどうか確認します。ブラウザーで `http://qp_server/quickplace` を開き、 QuickPlace 管理者としてログオンします。
   b. **「Create A QuickPlace (QuickPlace の作成)」**をクリックします。
   c. **「Standard QuickPlace for Teams (チーム用の標準 QuickPlace)」**を選択します。適切なフィールドを入力して、**「次へ」**をクリックします。
   d. 作成者としてログオンします。
2. QuickPlace のウェルカム・ページを更新します。**「編集」**をクリックします。必要な変更を行い、**「PUBLISH (発行)」**をクリックします。これにより、ウェルカム・ページが更新されます。
3. 以下のようにして、 QuickPlace の「Logo (ロゴ)」を変更します。
   a. 目次で**「Customize (カスタマイズ)」**をクリックします。
   b. **「Basics (基本)」**をクリックします。

© Copyright IBM Corp. 2001, 2002

c. **「Change Basics (基本の変更)」**をクリックします。

d. **「Simple Text (シンプル・テキスト)」**、**「Logo Maker (ロゴ・マーカー)」**、**「Upload Logo Artwork (ロゴのアートワークの更新)」**のいずれかを使用して、ロゴをセットアップします。 ToolTech のサンプルでは、**「Upload Logo Artwork (ロゴのアートワークの更新)」**を使用しました。これで、ロゴが更新されました。

シナリオ 2: カスタム・テーマの作成

インプリメンテーションの概説

テーマは QuickPlace の「ルック・アンド・フィール」を制御します。この例では、Hypertext Markup Language (HTML) ページ、および QuickPlace のレイアウトおよびスタイル用のカスケード・スタイル・シート (CSS) を作成します。これらのファイルの作成方法に関する詳細については、レッド・ブック *Customizing QuickPlace* を参照してください。

ご注意!

メンバー管理は WebSphere Commerce によってなされるため、目次の中の**「メンバー」**のリンクは除去してください。リンクを除去するには、目次の挿入時に「Page Layout (ページ・レイアウト)」ファイルで以下のコードを使用します。

```
<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=}>
```

カスタマイズ・ステップ

1. 目次で**「Customize (カスタマイズ)」**をクリックし、**「Custom Themes (カスタム・テーマ)」**、**「New Theme (新規テーマ)」** の順にクリックします。タイトルを入力し、**「Style Sheet (スタイル・シート)」**および**「Page Layout (ページ・レイアウト)」**ファイルをアップロードします。

2. 「**次へ**」をクリックします。
3. 目次で「**Customize (カスタマイズ)**」をクリックしてから、「**Decorate (装飾)**」をクリックします。
4. 「**Choose a theme (テーマの選択)**」をクリックします。
5. 作成したばかりのテーマを選択し、「**次へ**」をクリックします。これにより、QuickPlace のルック・アンド・フィールが更新されるはずです。
6. 目次の「**メンバー**」リンクがなくなるので注意してください。
7. QuickPlace から PlaceType を作成します。これで、カスタマイズしたばかりの QuickPlace に基づいて、 PlaceType を作成できるようになりました。目次で「**Customize (カスタマイズ)**」をクリックしてから、「**PlaceType Options (PlaceType オプション)**」をクリックします。
8. 「**編集**」をクリックします。「**Allow PlaceTypes to be created from this QuickPlace? (この QuickPlace から PlaceType を作成できるようにしますか?)**」には「**はい**」を、「**Include current members of this QuickPlace in future QuickPlaces created from the PlaceType? (以後 PlaceType から作成する QuickPlace に、この QuickPlace の現在のメンバーを含めますか?)**」には「**いいえ**」を選択します。
9. 「**next (次へ)**」をクリックします。
10. ブラウザーで `http://qp_server/quickplace` を開き、 QuickPlace 管理者としてログオンします。
11. 「**PlaceTypes**」をクリックします。
12. 「**CREATE PLACETYPE... (PlaceType の作成...)**」をクリックします。PlaceType の名前を入力し、基本にする QuickPlace を選択します。
13. 「**next (次へ)**」をクリックします。新しい PlaceType が表示されるはずです。

シナリオ 3: E メール通知のカスタマイズ

インプリメンテーションの概説

すべてのストアで使用される、カスタム・ページへのデフォルトの E メール通知を変更するには、 `CollabEmailContent.jsp` ファイルを変更して、必要な内容を含めなければなりません。

カスタマイズ・ステップ

1. `CollabEmailContent.jsp` ファイルを変更します。このファイルは以下のディレクトリーにあります。

   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. 以下のようにして、特定のストア用に E メールをカスタマイズします。
   a. DB2 コマンド・ウィンドウをオープンします。
   b. WebSphere Commerce データベースに接続します。デフォルトのデータベースに接続する場合、以下のコマンドを使用します。

      ```
      db2 connect to mall
      ```

   c. 以下の DB2 コマンドを実行します。

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

   d. カスタマイズしたテンプレート・ファイル CollabEmailContent.jsp を以下のストア・ディレクトリーに入れます。

      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*
      この *store_name* は、ストアの発行時に選択したストア・ディレクトリー名です。

   e. WebSphere Administration Server を再始動します。

第 4 章 顧客プロファイル

この章で説明するエレメントは、 WebSphere Commerce アクセラレーター内の顧客プロファイル・コンポーネントのユーザー・インターフェースを表します。これらは、顧客プロファイル、およびセラーやマーチャントがビジネスに関連して実行する必要のある日常的なタスクに関係したすべてのアクションの作成と保守を支援します。顧客プロファイル・コンポーネントには、以下のエレメントが含まれます。

- 顧客プロファイル・ウィザード
- 顧客プロファイル・ノートブック

サンプル・コード

このセクションでは、以下の URL にある ZIP ファイルに含まれているコード・サンプルに言及しています。

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

この章のいくつかのセクションを完全に実行するには、 WebSphere Application Server 4 を実行している開発マシンでこのサンプル・コードをダウンロードし、インストールする必要があります。

ダウンロードしたら、この ZIP ファイルを以下のディレクトリーに解凍します。

AIX `/usr/WebSphere/CommerceServer`

400 `/QIBM/UserData/WebSphere/CommerceServer`

Linux `/opt/WebSphere/CommerceServer`

Solaris `/opt/WebSphere/CommerceServer`

Windows *drive*`:¥WebSphere¥CommerceServer`

カスタマイズの例

以下の例で、 WebSphere Commerce アクセラレーターのこの部分をカスタマイズする方法を概説します。

シナリオ 1: 顧客プロファイルへの属性の追加

インプリメンテーションの概説

この例では、 USERS テーブルの FIELD1 列に、飲み物に関するユーザーの好みが保管されています。以下の値が有効になります。

- ビール
- ワイン
- コーヒー
- 紅茶
- 水

© Copyright IBM Corp. 2001, 2002

例では、これらのステップのそれぞれについて示されています。この例は、サンプル・コードのインストール時に作成した profcust ディレクトリー内にあります。

カスタマイズ・ステップ

1. 新しい JSP ファイルを作成することにより、顧客プロファイル・ノートブックに新しいページを追加します。ここでの例では、このページには以下の 2 つのオプションが表示されます。
   - Ignore beverage preference (飲料の設定を無視する)
   - Target customers with one or more of the following beverage preferences (以下の飲料の設定の 1 つ以上を持つ顧客をターゲットにする):

   設定は 2 番目のオプションの下に、チェック・ボックスとして表示されます。詳細については、以下のディレクトリー内にある付属の JSP ページ BeveragePanel.jsp を調べてください。

   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. JSP ファイルを作成したら、それを VIEWREG テーブルに追加しなければなりません。詳細については、以下のディレクトリー内にある付属の SQL スクリプト beverage.sql を検査してください。

   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. 新しいページは、以下のディレクトリー内にある、顧客プロファイル・ノートブックによって認識されていなければなりません。

   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¥

   SegmentNotebook.xml のコピーを作成します。サンプルでは、この新しいファイルの名前は MySegmentNotebook.xml です。以下のエレメントが文書に追加されました。

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

新しいパネル名をセグメンテーション・プロパティー・ファイルに含めます。このファイルは、以下のディレクトリーにあります。

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¥

これが機能するためには、以下の行を Resources.properties に追加する必要があります。

```
segmentNotebookBeveragePanel=Preferred beverage
```

以下から、 MySegmentNotebook.xml をコピーすることもできます。

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

以下の場所にコピーします。

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. この時点で、新しい XML 文書が認識されるように設定する必要があります。そのためには、以下のディレクトリーにある Resources.xml ファイルを変更します。

   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¥

   以下のエレメントを変更します。

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

   このエレメントを以下に変更します

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

   MyResources.xml として保管します。

   以下から、 MyResources.xml をコピーすることもできます。

   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

   以下の場所にコピーします。

   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. アプリケーションが MyResources.xml を使用するように設定します。以下のディレクトリーにある demo.xml ファイルを変更します。

   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¥

   以下を探します。

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

   このエレメントを以下のように変更します

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

6. 顧客プロファイルを記述する、データ bean com.ibm.commerce.tools.segmentation.SegmentNotebookDataBean を拡張します。サンプル com.mycompany.tools.segmentation.MySegmentNotebookDataBean では、飲料の属性のプロパティーを提供するデータ bean を拡張します。このサンプル MySegmentNotebookDataBean.java は、以下のディレクトリーにあります。

   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. セグメント・ノートブックに新しいデータ bean を認識させるために、顧客プロファイル・ノートブックを記述する XML を変更する必要があります。MySegmentNotebook.xml に戻って、以下のように変更されていることを確認してください。

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

   これが以下に変更されました。

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

8. 顧客プロファイルを保管するコントローラー・コマンドを拡張します。そのコマンドは com.ibm.commerce.tools.segmentation.SegmentSaveControllerCmd です。以下のディレクトリーの MySegmentSaveControllerCmdImpl 内にあるサンプル・コマンドを調べてください。

   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¥

   この例では、飲料の条件の単純条件を構成します。

   アプリケーションにこの新しいコマンドを認識させるため、 CMDREG テーブルに追加する必要があります。詳細については、ステップ 1a での beverage.sql を参照してください。

9. 顧客プロファイルを評価するタスク・コマンド com.ibm.commerce.membergroup.commands.CheckUserInMemberGroupCmd を拡張します。新しい条件を評価する方法の詳細については、以下のディレクトリー内にあるサンプル・コマンド MyCheckUserInMemberGroupCmdImpl.java を検査してください。

   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¥

このコマンドを CMDREG テーブルで登録することも必要です。詳細については、beverage.sql を参照してください。

10. 顧客プロファイルの制約のリストを表示するタスク・コマンドを拡張します。そのコマンドは com.ibm.commerce.tools.segmentation.SegmentConstraintListCmd です。変更しなければならないものの詳細については、以下のディレクトリー内にあるサンプル・コマンド MySegmentConstraintListCmdImpl を調べてください。

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¥

このコマンドを CMDREG テーブルの中に登録することも必要です。詳細については、 bevarage.sql を参照してください。

11. 以下のクラスを追加します。
    - com.mycompany.membergroup.commands. MyCheckUserInMemberGroupCmdImpl
    - com.mycompany.tools.segmentation.MySegmentConstraintListCmdImpl
    - com.mycompany.tools.segmentation.MySegmentNotebookDataBean
    - com.mycompany.tools.segmentation.MySegmentSaveControllerCmdImpl

    これは以下のディレクトリーにあります。

    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

    追加先は、以下のディレクトリー内の wcsmcruntime.jar ファイルです。

    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. ステップ 1 からの BeveragePanel.jsp ファイルを、以下のディレクトリーにコピーします。

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. これらの変更によりアクセス・コントロール設定に変更を加えることが必要になりますが、それはこのマニュアルの範囲を超えています。以下の URL から入手できる、 *WebSphere Commerce アクセス・コントロール・ガイド* を参照してください。

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

第 5 章 キャンペーン

データベースへのルールの直接入力

ユーザー・インターフェースは、ルール・データをデータベースに入力する最も簡単な方法ですが、場合によっては、ルールを INITIATIVE テーブルの RULE 列に手動で入力しなければならないことがあります。ルールは XML を使用して定義され、その場合、以下に示す文書型定義 (DTD) に従っていなければなりません。

INITIATIVE テーブルの RULE 列で使用される DTD を以下に示します。

```
<!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 エレメント

イニシアチブのデフォルト・インプリメンテーションでは、以下の simpleCondition エレメントがサポートされています。これを orListCondition、andListCondition、および openCondition エレメントと任意の組み合わせで結合することによって、ルールの条件部分を作成できます。

表 2.

変数名	サポートされている演算子	サポートされている値	修飾子
segment	= !=	顧客プロファイルの名前。	なし
shoppingCartSku	= !=	商品の SKU。	なし
shoppingCartCategory	= !=	商品のカテゴリー名。	言語
purchaseHistorySku	= !=	商品の SKU。	なし

© Copyright IBM Corp. 2001, 2002

表 2. (続き)

変数名	サポートされている演算子	サポートされている値	修飾子
purchaseHistoryCategory	= !=	商品のカテゴリー名。	言語
shoppingCartTotal	= != > < >= <=	10 進数値。	通貨
dayOfWeek	= !=	SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY	なし

openCondition エレメント

デフォルト・インプリメンテーションでは、以下の openCondition エレメントがサポートされています。これを orListCondition、andListCondition、および simpleCondition エレメントと任意の組み合わせで結合することによって、ルールの条件部分を作成できます。

表 3.

オープン条件名	サポートされているパラメーター
shoppingCartTotal	comparisonType - ″>″、″<″、″=″ のいずれか。 value - 10 進数値の通貨。
shoppingCartCategory	comparisonType - ″=″ か ″!=″ のいずれか。 value - カテゴリー言語の名前。
shoppingCartSku	comparisonType - ″>″、″<″、″=″ のいずれか。 value - 10 進数値の通貨。
purchaseHistoryCategory	comparisonType - ″=″ か ″!=″ のいずれか。 value - カテゴリー通貨の名前。
purchaseHistorySku	comparisonType - ″>″、″<″、″=″ のいずれか。 value - 10 進数値の通貨。

アクション・エレメント

デフォルト・インプリメンテーションでは、以下のアクション・エレメントがサポートされています。これを任意の条件エレメントとともに使用することによってルールを構成できます。

表 4.

アクション名	サポートされているパラメーター	サポートされている値	サポートされているサブパラメーター
suggestiveSell	queryOperator	or and	なし
	queryType	skuQuery genericQuery	なし
	sku	商品の SKU	なし
	skuPrefix	商品の SKU の接頭部	なし
	category	商品のカテゴリー名	言語
	productDescription	商品のカテゴリー記述	言語
	inventoryQuantity	整数	comparisonOperator
	listPrice	10 進数値	comparisonOperator currency
	availabilityDate	日付 (yyyy-mm-dd-00.00.00 の形式)	comparisonOperator
collaborativeFiltering	なし		
awarenessAd	collateral	広告名	
discountAd	discountCollateral	広告名	割引 ID
couponAd	couponCollateral	広告名	
categoryRecommendation	category	カテゴリーの名前	

以下の例の先頭では、男性を対象とした春の広告、および女性を対象とした秋の広告が示されています。

```
<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>
```

インフラストラクチャー・エレメント

ルール・パッケージ

キャンペーン・コンポーネントでは、 com.ibm.commerce.rule パッケージを使用します。このパッケージでは、ルールのセットの XML 形式への変換、および XML 形式からのルールのセットの変換に役立つクラスを提供します。また、ルール・セット内でのルールの呼び出しのサポートも提供します。

表 5.

クラス / インターフェース名	説明
Action	Action クラスは、ルールのアクション部分によって使用される名前およびパラメーターのプロパティーを提供します。パラメーターには名前と値があり、サブパラメーターも指定できます。
ActionHandler	ActionHandler インターフェースは、ルールのアクション部分のインプリメンテーションを提供するためにインプリメントする必要があります。 ″performAction″ というメソッドが 1 つあり、これはルールの条件部分が true と評価される場合に呼び出されます。 Action クラスの適切なインスタンスが、 ″performAction″ メソッドのインプリメンテーションに渡されます。
Rule	Rule クラスは、 Rule のインスタンスの XML 形式への変換、および XML 形式からの Rule のインスタンスの変換に役立つメソッドを提供します。 Rule クラスには 3 つのプロパティーがあります。 1 つはルールの条件部分で使用されます。もう 1 つはルールのアクション部分で使用されます。さらにコメントで使用されるものもあります。条件プロパティーは、 com.ibm.commerce.condition パッケージ内にある、 Condition クラスのインスタンスでなければなりません。 Rule クラスにはメソッド ″invoke″ もあります。これは、 Evaluator インターフェースおよび ActionHandler インターフェースのインプリメンテーションを受け入れます。条件 true と評価される場合、 ActionHandler インプリメンテーションの ″performAction″ メソッドが呼び出されます。

表 5. (続き)

クラス / インターフェース名	説明
RuleConstants	RuleConstants インターフェースには、ルール・パッケージで使用されるいくつかの公開定数が含まれます。
RuleSet	RuleSet クラスは、 RuleSet のインスタンスの XML 形式への変換、および XML 形式からの RuleSet のインスタンスの変換に役立つメソッドを提供します。 Rule オブジェクトの配列である、プロパティーが含まれています。 Evaluator および ActionHandler インターフェースのインプリメンテーションを受け入れる呼び出しメソッドもあります。呼び出しメソッドが呼び出されると、 Rule オブジェクトの配列をループ処理によって次々に呼び出します。

Campaigns はルール・パッケージを使用して、キャンペーン・イニシアチブのルール部分を XML 形式へ保管したり、 XML 形式からキャンペーン・イニシアチブのルール部分をロードしたりします。また、ルールを呼び出すのにも役立ちます。 `CampaignInitiativeEvaluateCmd` タスク・コマンドは、キャンペーン・イニシアチブ・アクションを処理するのに使用される、 `ActionHandler` インターフェースのインプリメンテーションを提供します。

条件パッケージ

ルール・コンポーネントでは、 `com.ibm.commerce.condition` パッケージを使用します。このパッケージでは、ブール条件から XML 形式への変換、および XML 形式からのブール条件の変換に役立つクラスを提供します。条件の評価のサポートも提供します。

表 6.

クラス / インターフェース名	説明
AndListCondition	AndListCondition クラスは ConditionList クラスを拡張したもので、条件を提供します。これは条件のリストをブール演算子 AND で結合したものです。
Condition	Condition クラスは抽象クラスであり、 XML 形式への変換および XML 形式からの変換に役立つメソッドを提供します。これは、条件の評価に使用できる抽象メソッドも提供します。
ConditionConstants	ConditionConstants インターフェースは、条件パッケージによって使用される定数値を提供します。
ConditionList	ConditionList クラスは、 Condition クラスを拡張した抽象クラスです。これは条件のサポートを提供します。この条件は条件のリストで構成されます。

表 6. (続き)

クラス / インターフェース名	説明
ConditionUtil	ConditionUtil クラスは、条件の評価中に使用できる静的メソッドを多数提供します。
Evaluator	Evaluator インターフェースは、単純条件およびオープン条件を評価するためにインプリメントする必要があります。
OpenCondition	OpenCondition クラスは Condition クラスを拡張したもので、汎用条件を提供します。この条件は、名前と値の組のリストで構成されます。
OrListCondition	OrListCondition クラスは ConditionList クラスを拡張したもので、条件を提供します。これは条件のリストをブール演算子 OR で結合したものです。
SimpleCondition	SimpleCondition クラスは Condition クラスを拡張したもので、フォーム変数、演算子、値の条件を提供します。
TrueCondition	TrueCondition クラスは Condition クラスを拡張したもので、必ず true と評価する条件を提供します。

ルール・パッケージは条件パッケージを使用して、ルールの条件部分を XML 形式へ保管したり、 XML 形式からルールの条件部分をロードしたりします。また、条件を評価するのにも役立ちます。

抽象 Condition クラスの具象拡張が 5 つあります。これらのうちの 3 つ (AndListCondition、OrListCondition、および TrueCondition) は、それら自身を評価する方法を認識しています。残りの 2 つ (OpenCondition および SimpleCondition) は、その評価のために Evaluator インターフェースのインプリメンテーションが必要です。 CampaignInitiativeEvaluateCmd タスク・コマンドは、キャンペーン・イニシアチブの条件を評価するのに使用される Evaluator インターフェースのインプリメンテーションを提供します。

Blaze ルール・プロジェクト

キャンペーン・イニシアチブのデフォルトのインプリメンテーションでは、キャンペーン・イニシアチブ・ルールを評価するのに役立つ Blaze ルール・プロジェクトを使用します。デフォルトでは、すべてのオープン条件により、 Blaze ルール・プロジェクトが **CampaignInitiativeEvaluateCmd** タスク・コマンドによって呼び出されます。プロジェクト `CampaignInitiativeEvaluator` は、以下のディレクトリーに配置します。

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`

このプロジェクトのデフォルト・インプリメンテーションでは、キャンペーン・イ

ニシアチブのユーザー・インターフェースによって作成されたすべてのオープン条件を評価できます。追加のオープン条件のサポートをこのプロジェクトに追加できます。

カスタマイズ

顧客は、キャンペーン・イニシアチブのデフォルトのインプリメンテーションを拡張し、さらにサポートされている条件 (場合によっては追加のアクションと表示タイプも) を追加することが予期されます。

新しいイニシアチブの条件のランタイム・サポートの追加

キャンペーン・イニシアチブの条件を評価する blaze プロジェクトは、**CampaignInitiativeContext** のインスタンスを受け入れます。このコンテキストは、オープン条件の名前およびパラメーター以外に、コマンド・コンテキストへのアクセスを提供します。新しいオープン条件のサポートを追加するには、新しいルールを conditionEvaluator ルールに追加する必要があります。このルールでは、新しい条件の名前を検査し、条件を評価する必要があります。条件を評価する際に、コンテキスト・オブジェクトを介し、名前によりオープン条件パラメーターにアクセスできます。

新しいイニシアチブの条件の作成時サポートの追加

市場管理者が新しい条件を活用できるようにするために、キャンペーン・イニシアチブのユーザー・インターフェースを拡張して、新しい条件のサポートを提供する必要があります。これは、キャンペーン・イニシアチブのウィザードまたはノートブックの WHICH パネル (WhenAddPanel.jsp) を追加することによって実現できます。

CampaignInitiativeSaveControllerCmdImpl クラスを拡張して、新しい Condition オブジェクトを構成することも必要です。このファイルは以下のディレクトリーにあります。

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¥
```

新しいキャンペーン・イニシアチブのアクションのランタイム・サポートの追加

新しいキャンペーン・イニシアチブのアクションのサポートを提供するために、 **CampaignInitiativeEvaluateCmdImpl** クラスを拡張し、**performAction** メソッドをオーバーライドする必要があります。アクショ

ンの名前が新しいアクション名と一致する場合はそのアクションを実行し、そうでない場合はスーパー・メソッドを呼び出してデフォルトのインプリメンテーションを実行しなければなりません。

新しいキャンペーン・イニシアチブのアクションの作成時サポートの追加

市場管理者が新しいキャンペーン・イニシアチブのアクションを活用できるようにするために、キャンペーン・イニシアチブのウィザードまたはノートブックの WHAT パネル (InitiativeWhatPanel.jsp) を更新する必要があります。 **CampaignInitiativeSaveControllerCmdImpl** クラスを拡張して、新しい Action オブジェクトを構成することも必要です。このファイルは以下のディレクトリーにあります。

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¥
```

新しい e-マーケティング・スポット・タイプのサポートの追加

追加の出力タイプを提供する必要がある場合、 EMarketingSpot クラスを拡張して、ページ設計者によって使用される新しいプロパティーを提供できます。 **CampaignInitiativeEvaluateCmdImpl** クラスを拡張して、新しい出力タイプを戻すことも必要です。

第 6 章 カタログ検索

この章のエレメントでは、既存の検索機能を拡張するために必要なアクティビティーについて説明します。追加機能性として、検索コンポーネントによってアクセス可能な新規属性とテーブルの導入方法についても説明します。スキーマを最適化してランタイムに対するコストを削減すれば、同時に検索コンポーネントのパフォーマンスも改善できます。このコンポーネントには、カスタマイズ可能な以下のエレメントが含まれます。

- 検索 bean
- 検索エンジン
- 要約テーブル定義スクリプト

表 8 は、カタログ検索 bean で検索されるデータベース列のリストの要約です。サイトの要件はさまざまに異なるので、この表に掲載されていない列の検索が必要になる場合もあります。

カスタマイズ・シナリオ

検索エンジンですでに検索可能な検索属性を検索 bean に追加するには、検索 bean にその属性を追加することによってシナリオ 1 を実行します。検索エンジンでこの属性の検索がサポートされていなくても、検索エンジンがすでに検索したテーブルの 1 つの中でサポートされているなら、シナリオ 2 によりその属性をエンジンに追加してください。この属性が、検索エンジンの検索しているテーブルの 1 つの中にない場合は、シナリオ 3 を実行してテーブルを検索エンジンに追加します。検索 bean に rich 属性を追加する手順は、検索 bean への他の属性の追加 (シナリオ 1 の実行) と同じですが、インプリメンテーションが少し異なります。シナリオ 4 を参照してください。ランタイム・パフォーマンスを向上させるには、シナリオ 5 を実行し、各カテゴリーごとに 1 つのテーブルを作成して検索される行数を削減することによって、要約テーブル定義スクリプトを変更して最適化します。以下のシナリオは、このコンポーネントのカスタマイズ手順を表しています。

シナリオ 1: 検索 bean への属性の追加
検索エンジンですでに検索可能な新規属性を bean に導入します。

シナリオ 2: 検索エンジンへの属性の追加
検索エンジンがそのテーブルですでに検索を実行している場合、新規属性を検索エンジンに導入します。

シナリオ 3: 検索エンジンへのテーブルの追加
検索エンジンに新規テーブルを導入します。

シナリオ 4: 検索エンジンへの rich 属性の追加
bean に rich 属性を導入します。

シナリオ 5: 最適化要約テーブルによるパフォーマンスの改善
データ・セットの知識を使用して要約テーブル定義を最適化します。

注: 検索エンジン、検索インターフェース、および Unified Search Framework は同義であり、本書ではこれらを同じ意味で使用しています。

© Copyright IBM Corp. 2001, 2002

シナリオ 1: 検索 bean への属性の追加

インプリメンテーションの概要

カスタマイズできる最初の項目は、他の検索可能属性の追加です。このアクティビティーでは、検索 bean を変更する必要があります。また、この属性が検索エンジンでも検索できない場合は、オプションとして検索エンジンにも変更を加える必要があります (シナリオ 2 を参照)。検索 bean をカスタマイズするには、それをサブクラス化して既存メソッドを拡張する必要があります。 samples/search ディレクトリーには、これを行う方法を示すサンプル・コードがあります。

カスタマイズ・ステップ

以下の例は、com.ibm.search.catalog パッケージにそのメタデータ・クラスがあるカタログ検索 bean に、追加検索属性を追加する方法を示しています。カタログ検索データ bean はクラスです。これをカスタマイズするには、以下のようにします。

1. CatEntrySearchListDataBean クラスのサブクラスを作成します。
2. 追加する新しい検索可能属性の変数を宣言します。検索可能属性に関連した情報を保管するための 2 タイプの属性を宣言できます。第 1 のタイプは、属性の値の保管に使用できます。第 2 のタイプには、 LIKE 検索演算子、ブール検索、ケース・センシティブなどの検索で使用できる関連情報を含めることができます。これらの変数に保管された情報は、カタログ情報を照会する SQL 照会の生成で使用できます。

   注: SQL 照会の生成で使用されるデータベース列を、生成された照会の定数として使用するには、 Search Interface RuleQuery クラスまたはそのサブクラスの 1 つで事前定義する (『検索インターフェースのカスタマイズ』を参照) 必要があります。
3. 宣言された変数に対するアクセス機能 (GET および SET メソッド) を作成します。
4. populate() メソッドを作成します。これは以下のことを実行します。
   a. RuleQuery または RuleQuery のサブクラスのいずれかのインスタンスを生成します。
   b. setRuleQuery(ruleQueryInstance) メソッドを使用して、この RuleQuery のインスタンスを親 bean クラス内のインスタンスと照会します。
   c. super.setPredefinedAttributes() メソッドを呼び出します。
   d. 検索インターフェースを使用する新しい検索可能定数の <Predicate> を記述します。
   e. super.setIsAllNull(false) を使用して、新しい検索属性が追加されたことを親に通知します。
   f. super.execute() メソッドを呼び出します。

例

以下の例で、カタログ検索 bean に新しい検索属性 onSpecial を追加する方法を示します。この例では com.ibm.search.catalog パッケージにメタデータ・クラスが存在することを想定しています。

1. 新しいサブクラスを作成します。

2. 新しい検索可能属性の変数を作成します。

```
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 変数は、検索可能属性の値の保管に使用します。 onSpecialOperator 変数は、検索演算子 (like、=、<、>、<=、>= など) の保管に使用します。

3. すべての変数に対するアクセス機能を作成します。

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

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

4. populate() メソッドを作成します。

```
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
}
```

シナリオ 2: 検索エンジンへの属性の追加

インプリメンテーションの概要

検索エンジンは、照会を正しく公式化してカタログから結果を引き出すために属性とテーブルに関する情報を必要とします。この情報はメタデータとして保持されます。検索エンジンを拡張して新しい検索可能属性を追加するには、追加属性メタデータ定義、そしてオプションとしてテーブル・メタデータ定義が必要です。テーブル・メタデータは、検索エンジンに新規テーブルを導入する場合にのみ必要であり、これについてはシナリオ 3 で詳しく説明します。このシナリオを完了したら、『シナリオ 1: 検索 bean への属性の追加』を実行して、これを実際の JSP Page Designer で使用できるようにする必要があります。 samples/search ディレクトリーには、その方法を示すサンプル・コードがあります。

カスタマイズ・ステップ

検索エンジンは、Unified Search Framework の一部です。これは RuleQuery というクラスと、 AttributeInfo および TableInfo (それぞれ列名とテーブル名を示す) などの追加メタデータ・クラスとして表されます。以下の例で、com.ibm.search.catalog パッケージ内にないデータベース列に対するメタデータ・クラス (ただし、この列が属するテーブルのメタデータは存在) を作成することによって、検索エンジンに新規属性を追加する方法を示します。検索エンジンをカスタマイズするには、以下のようにします。

1. 検索可能にする各列およびテーブルごとに検索インターフェース・メタデータを定義します。これには、以下の要件があります。
   a. 各検索可能テーブルには、 TableInfo クラスのサブクラスであるクラスが対応していなければなりません。このサブクラスはテーブル名を指定していなければなりません。
   b. 各検索可能列には、AttributeInfo クラスのサブクラスであるクラスが対応していなければなりません。このサブクラスは列名、列が属する TableInfo サブクラス、および列の SQL データ・タイプを指定していなければなりません。
2. RuleQuery クラスのサブクラスを作成し、各新規データベース列ごとに静的整数参照を定義します。整数値 0 ～ 1000 はシステム用に予約済みであることにご注意ください。
3. findAttributeInfoName() メソッドを作成します。このメソッドで親 findAttributeInfoName() メソッドがオーバーライドされないようにするため、super.findAttributeInfoName() メソッドを呼び出すことをお勧めします。親 findAttributeInfoName() がオーバーライドされると、親クラスで定義されたデータベース列は使用できなくなります。
4. このメソッドに factory クラス作成論理を追加して、各新規データベース列に対する AttributeInfo メタデータ・クラスを作成します。
5. 新しい検索可能テーブルの事前定義テーブル結合関係を追加して、 search.xml ファイルを変更します。すべてのテーブルの組み合わせの結合関係が必要です。search.xml を変更するには、以下のようにします。
   a. エントリーが公式化するテーブル結合関係について説明する COMMENT セクションを追加します。
   b. エントリーを一意的に識別する KEY セクションを追加します。結合されたテーブルがユニーク・キーとして使用されます。
   c. キーと関連した値を指定する VALUE セクションを追加します。値セクションのエントリーは結合関係を定義します。値セクションの列名は、その列の AttributeInfo サブクラスを使用して定義します。

例

以下の例では、新しい検索可能データベース列 CATENTRY.ONSPECIAL が作成されます。

1.
   a. TableInfo クラスのサブクラスを作成して、テーブル名 CATENTRY を指定します (このテーブルのクラスが存在しない場合)。

```
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. AttributeInfo クラスのサブクラスを作成して、列名 ONSPECIAL を指定します。

```
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. 検索インターフェース RuleQuery のサブクラスを作成します。

```
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. findAttributeInfoName() メソッドを作成します。

```
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. 必要に応じて Search.xml を変更します (この例では search.xml ファイルを変更する必要はありません)。

注: この新規メタデータを検索 bean に追加するには、シナリオ 1 のステップに従ってください。ただし、RuleQuery クラスのインスタンスの代わりに、メタデータ情報 (上記の例では CustomQuery) を持つ RuleQuery の副照会のインスタンスを使用する必要があります。

シナリオ 3: 検索エンジンへのテーブルの追加

インプリメンテーション概要

検索エンジンで現在サポートされていない新規テーブルの属性を追加するには、テーブルに関する情報を定義して、このテーブルがエンジンの使用する他のテーブルとどのように関係するのかを説明する必要があります。この情報は各テーブルを他のテーブルと関連付ける結合述部を導き出すために使用される結合関係であり、メ

タデータとして保持されます。検索エンジンを拡張して新しい検索可能テーブルを追加するには、各テーブルと、そのテーブルと他のテーブルとの関係を定義する必要があります。

『シナリオ 2: 検索エンジンへの属性の追加』を実行して、このテーブルに関連した属性を検索 bean の作成者が使用できるようにする必要があります。
samples/search ディレクトリーには、その方法を示すサンプル・コードがあります。

カスタマイズ・ステップ

以下の例で、データベース・テーブルの結合関係メタデータ・クラスを作成する方法を示します。一般に、各々のすべての論理テーブル関係の組み合わせごとに 1 つの結合関係を定義する必要があります。

例

以下の例では、既存の search.xml を変更して、これに CATENTRY テーブルと OFFERPRICE テーブルの間の新規結合関係のエントリーを含めています。

1. COMMENT セクションを追加します。

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

2. KEY および VALUE セクションを追加します。

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

シナリオ 4: 検索 bean への rich 属性の追加

インプリメンテーション概要

bean に rich 属性の検索を追加するプロセスは、bean への他の検索可能属性の追加と同じです。データ・セットが認識されて、任意の rich 属性を識別するようになったら、シナリオ 1 に従って、検索 bean でこれらの属性が使用できるようにしてください。このシナリオとシナリオ 1 の間の唯一の相違点は、 rich 属性述部を構成するために使用されるメソッド・パラメーターにあります (詳細は、以下の例の中の太字のテキストを参照してください)。検索エンジンにはすでに、 ATTRIBUTE テーブルと ATTRVALUE テーブルで定義された rich 属性を検索する機能があります。

カスタマイズ・ステップ

カスタマイズ・シナリオ 1『検索 bean への属性の追加』と同じです。

例

以下の例で、新しい検索可能 rich 属性 color をカタログ検索 bean に追加する方法を示します。

1. 新しいサブクラスを作成します。
2. 新しい検索可能属性の変数を作成します。

```
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;
  }
```

   変数 colorValue は、検索可能属性の値を保管します。変数 colorValueOperator は、検索演算子 (like、=、<、>、<=、>= など) を保管します。

3. すべての変数に対するアクセス機能を作成します。

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

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

4. populate() メソッドを作成します。

```
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
}
```

シナリオ 5: 最適化要約テーブルによるパフォーマンスの改善

インプリメンテーション概要

データ・セットを定義したら、この知識を使用してより最適な要約テーブルを作成し、検索のパフォーマンスを大きく改善することができます。目標は、検出したい照会タイプに類似したテーブルを作成することです。ほとんどの場合、これらの要

約テーブルはずっと小さくなり、そのため、これらのテーブルで適用される照会は、より大規模な基礎の固有テーブルや他のあまり最適でない要約テーブルを使用する場合よりも高速になります。

カスタマイズ・ステップ

以下の例で、各カテゴリー・グループに対する rich 属性カテゴリー・グループ要約テーブルを作成する方法を示します。 rich 属性を持つアイテムを持つカテゴリー・グループ (CATGROUPS) が多数存在する場合は、各グループごとに要約テーブル定義を定義します。

この要約テーブルを作成するためのステップは以下のとおりです。

1. 要約テーブル定義として使用する Select 照会を使用してテーブル作成ステートメントを定義します。この Select ステートメントには、「group by」文節で定義された、結果セットにリストされているものと同じ列が含まれていなければなりません。詳細は、「自動要約テーブル」(AST) に関する DB2 資料か、あるいは「マテリアライズ表示」に関する Oracle 資料を参照してください。
2. Select ステートメントで、カテゴリーのカテゴリー・グループ述部を指定します。
3. 上記の照会で指定されているのと同じ結果セット列が同じ順序で含まれた索引を作成します。
4. 各カテゴリー・グループごとに上記のステップを繰り返します。

例

以下の例は、 10000、10001、および 10002 という 3 つの各カテゴリー・グループに対する、カテゴリー・グループの rich 属性の要約テーブル定義スクリプトを表しています。太字部分は、wcs.summary.richAttributeCatGroup.create.sql ファイルに表示されたこの例の基になる一般定義と比較して新しい部分です。

```
// 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;
```

カタログ検索データ bean の変数

URL (JSP.) を介してカタログ検索 bean に渡す必要のある変数のリスト

表 7.

名前	データ・タイプ	説明
beginIndex	ストリング	この変数は結果セットのページングに使用されます。値はページ内の最初の結果行の索引でなければなりません。
catGroupId	ストリング	この変数の値は、カテゴリー ID の検索で使用されます。
categoryTerm	ストリング	この変数の値は、カテゴリー名または説明、あるいはその両方の検索で使用されます。

表 7. (続き)

categoryTermCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
categoryTermOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (同等演算子) のいずれかです。
categoryTermScope	整数	ユーザーは検索範囲を、「名前」、「名前と簡略説明」、または「名前、簡略説明、および詳細説明」のいずれかに制限できます。この変数の値は、ユーザーの選択を保管するために使用されます。この値は、 1 (名前と簡略説明) または 2 (名前のみ)、または 3 (名前、簡略説明、および詳細説明) のいずれかです。
categoryType	ストリング	ユーザーは、 AND 検索、OR 検索、または完全一致検索の 3 つの検索基準のタイプを指定できます。この変数の値は、ユーザーの検索基準を保管するために使用されます。値は ALL (AND 検索)、ANY (OR 検索)、または EXACT (完全一致検索) のいずれかです。
currency	ストリング	この変数の値は、通過の検索で使用されます。
currencyCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
currencyOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。
filterTerm	ストリング	この変数の値は、指定された値の検索をフィルターに掛けるために使用されます。
filterTermCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
filterTermOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。

表 7. (続き)

filterType	ストリング	ユーザーは、All、Any、または Exact Phrase の 3 つの検索基準のタイプを指定できます。この変数の値は、ユーザーの検索基準を保管するために使用されます。値は ALL (AND 検索)、ANY (OR 検索)、または EXACT (完全一致検索) のいずれかです。
isListPriceOn	ストリング	カタログ検索 bean のユーザーは、表示価格か標準価格を選択して公開できます。表示価格を公開する場合はこの変数を yes に設定し、標準価格を公開する場合はこの変数を no に設定します。検索 bean はデフォルトで標準価格を公開します。
manufacturer	ストリング	この変数の値は、製造者の名前の検索で使用されます。
manufacturerCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
manufacturerOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。
manufacturerPartNum	ストリング	この変数の値は、製造者の部品番号の検索で使用されます。
manufacturerPartNumCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
manufacturerPartNumOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。
maxPrice/minPrice	ストリング	これらの変数の値は、価格範囲の検索で使用されます。
pageSize	ストリング	この変数の値は、1 ページに表示される検索結果行数を指定します。
price	ストリング	この変数の値は、価格の検索で使用されます。
priceOperator	ストリング	ユーザーは、検索演算子として =、<、>、!=、<=、>= のいずれかの演算子を選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は EQUAL、NOTEQUAL、GREATER、LESS、GREATER_EQUAL、LESS_EQUAL のいずれかです。
qtyAvailable	ストリング	この変数の値は、商品またはアイテムの在庫の検索で使用されます。

表 7. (続き)

qtyAvailableOperator	ストリング	ユーザーは、検索演算子として =、<、>、!=、<=、>= のいずれかの演算子を選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は EQUAL、NOTEQUAL、GREATER、LESS、GREATER_EQUAL、LESS_EQUAL のいずれかです。
qtyMeasure	ストリング	この変数の値は、数量単位の検索で使用されます。
qtyMeasureCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
qtyMeasureOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。
resultCount	ストリング	この変数には、検索で戻された結果の合計数が含まれます。
resultType	ストリング	マーチャントは、商品、またはアイテム、あるいは商品とアイテムの両方のいずれを検索結果に表示するか指定することができます。この変数の値は、この値を保管するために使用されます。値は 1 (商品のみ)、2 (アイテムのみ)、または 3 (商品とアイテムの両方) のいずれかです。
searchTerm	ストリング	この変数の値は、ワードの検索で使用されます。
searchTermCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
searchTermOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。
searchTermScope	整数	ユーザーは検索範囲を、「名前」、「名前と簡略説明」、「名前、簡略説明、および詳細説明」、または「キーワード」のいずれかに制限できます。この変数の値は、ユーザーの選択を保管するために使用されます。この値は、 1 (名前と簡略説明)、2 (名前のみ)、 3 (名前、簡略説明、および詳細説明)、または 4 (キーワード) のいずれかです。

表 7. (続き)

searchType	ストリング	ユーザーは、All、Any、または Exact Phrase の 3 つの検索基準のタイプを指定できます。この変数の値は、ユーザーの検索基準を保管するために使用されます。値は ALL (AND 検索)、ANY (OR 検索)、または EXACT (完全一致検索) のいずれかです。
sku	ストリング	この変数の値は、SKU の検索で使用されます。
skuCaseSensitive	ストリング	ユーザーは、検索で大文字小文字を区別するかどうかを選択できます。この変数の値は、検索で大文字小文字を区別するかどうかを識別するために使用されます。値は yes (大文字小文字を区別する) か no (大文字小文字を区別しない) のいずれかです。
skuOperator	ストリング	ユーザーは、検索演算子として like か equal のいずれかを選択できます。この変数の値は、ユーザーの選択を保管するために使用されます。値は LIKE (LIKE 演算子) か EQUAL (イコール演算子) のいずれかです。

カタログ検索データベースの列

カタログ検索 bean が検索できるデータベース列のリスト

表 8.

bean インターフェースで定義された検索可能属性	テーブル	列	ブール式のサポート
カテゴリー・グループ - すべての名前と記述	CatGrpDesc	NAME LONGDESCRIPTION SHORTDESCRIPTION	あり
カテゴリー・グループ - 表題のみ	CatGrpDesc	NAME	あり
カテゴリー・グループ - 表題と記述 (推奨デフォルト)	CatGrpDesc	NAME SHORTDESCRIPTION	あり
カテゴリー言語 ID	CatGrpDesc	LANGUAGE_ID	なし
カテゴリー ID	CatGpEnRel	CATGROUP_ID	なし
CatEntry 記述 - すべての名前と記述	CatEntDesc	NAME LONGDESCRIPTION SHORTDESCRIPTION	あり
CatEntry - 表題のみ	CatEntDesc	NAME	あり
CatEntry - 表題と記述 (推奨デフォルト)	CatEntDesc	NAME SHORTDESCRIPTION	あり
CatEntry 言語 ID	CatEntDesc	LANGUAGE_ID	なし
CatEntry 製造業者	CatEntry	MFNAME	なし
CatEntry 製造者部品番号	CatEntry	MFPARTNUMBER	なし
CatEntry SKU	CatEntDesc	PARTNUMBER	なし
Price	表示価格 / オファー価格	PRICE	なし

表 8. (続き)

bean インターフェースで定義された検索可能属性	**テーブル**	**列**	**ブール式のサポート**
価格範囲	表示価格 / オファー価格	PRICE	なし
通貨	表示価格 / オファー価格	CURRENCY	なし
購入可能数量	InvStVw/ StoreInv	QTYAVAILABLE	なし
数量単位	InvStVw/ StoreInv	QUANTITYMEASURE	なし

bean のユーザーは、 bean をカスタマイズすることによって、他にも rich 属性などの列を公開することができます。カスタマイズの詳細は、オンライン・ヘルプの ExtendedCatEntrySearchListDataBean クラス用の JavaDoc を参照してください。

第 7 章 クーポン

この章で説明されているエレメントは、 WebSphere Commerce アクセラレーターのクーポン・コンポーネントへのユーザー・インターフェースを表します。これは、クーポン、クーポン・オファー、およびセラーやマーチャントが日常で実行する他のタスクの作成と保守のためのものです。クーポン・コンポーネントには、以下のエレメントがあります。

- クーポン・ウィザード
- クーポン・ノートブック

カスタマイズ例

以下の例では、WebSphere Commerce アクセラレーターのこの部分をカスタマイズする方法を示します。

シナリオ 1: クーポン割引の作成時に新しい情報を追加する

インプリメンテーションの概要

このシナリオでは、クーポン・ツールをカスタマイズして、クーポンにクーポン割引を作成したユーザーなどの追加情報を記録するときに必要なステップを考慮します。

カスタマイズのステップ

1. 新しい **CustomCreateCouponDiscountCmdImpl** をインプリメントします。これは、タスク・コマンドのインプリメンテーション **CreateCouponDiscountCmdImpl** を拡張し、 **CreateCouponDiscountCmd** をインプリメントします。
2. CALCODE テーブルの FIELD1 列を使用して、このクーポン割引を作成したユーザーの userId を保管します。
3. 新しいタスク・コマンドを CMDREG テーブルに登録します。顧客ストア ID は 1 であると想定し、次の SQL ステートメントを実行します。

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

4. **CustomCreateCouponDiscountCmdImpl** を更新します。このコマンドに、performExecute() メソッドをインプリメントします。

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

© Copyright IBM Corp. 2001, 2002

第 8 章 割引

この章で説明されているエレメントは、WebSphere Commerce アクセラレーターの割引コンポーネントへのユーザー・インターフェースを表します。これは、割引、およびセラーやマーチャントが日常で実行する他のアクションの作成と保守のためのものです。割引コンポーネントには、以下のエレメントがあります。

- 割引ウィザード
- 割引ノートブック

カスタマイズ例

以下の例では、WebSphere Commerce アクセラレーターの割引コンポーネントをカスタマイズする方法を示します。

シナリオ 1: 割引の作成時に別の情報を追加する

インプリメンテーションの概要

このシナリオでは、割引の作成時に別の情報を追加する場合に必要なステップを考慮します。例では、割引を作成したユーザーの監査記録を作成することを想定しています。このシナリオでは、新しいパラメーターを送信する必要はありません。

カスタマイズのステップ

1. **新しい CustomCreateDiscountCmdImpl をインプリメントします。**これは、タスク・コマンド・インプリメンテーション **CreateDiscountCmdImpl** を拡張し、 **CreateDiscountCmd** をインプリメントするものです。
2. **CALCODE テーブルの field1 列を使用して、この割引を作成するユーザーの userId を保管します。**
3. **新しいタスク・コマンドを CMDREG テーブルに登録します。** 顧客ストア ID は 1 であると想定し、次の SQL ステートメントを実行します。

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

4. **CustomCreateDiscountCmdImpl に、** メソッド performExecute() をインプリメントします。

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

シナリオ 2: デフォルト動作を変更する

インプリメンテーションの概要

このシナリオでは、オーダーに割引を適用するときのシーケンスを指定できるように、割引ウィザードのデフォルト動作を変更します。さらにこのシナリオでは、

© Copyright IBM Corp. 2001, 2002

URL を使用して新しいパラメーターを渡す必要もあります。このときには、割引ウィザードの最初のパネルにフィールドを追加して、追加のデータを入手する必要があります。

カスタマイズのステップ

1. 以下を実行して、割引ウィザードに関係する JSP ファイルを更新します。
   a. 以下のディレクトリーに移動します。

      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. DiscountWizardWelcome.jsp を MyDiscountWizardWelcome.jsp にコピーします。
   c. MyDiscountWizardWelcome.jsp を更新します。フォーム・ブロックで、次のコードを追加します。

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

   d. フレームワークへのエントリーを保管します。 JSP では、JavaScript 関数 savePanelData があるので、その関数に次の行を追加します。

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

   e. VIEWREG テーブルへの新規エントリーを登録します。たとえば、顧客ストア ID が 1 であれば、次の SQL ステートメントを実行します。

      ```
      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. 新しい **DiscountSaveCmdCustomImpl** をインプリメントします。これは、コントローラー・コマンド・インプリメンテーション **DiscountSaveCmdImpl** を拡張し、 **DiscountSaveCmd** をインプリメントします。
3. この新しいインプリメンテーションを CMDREG テーブルに登録します。顧客ストア ID は 1 であると想定し、次の SQL ステートメントを実行します。

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

4. **DiscountSaveCmdCustomImpl** コマンドに、次のサンプル・コードを使用して 2 つのメソッドをインプリメントします。

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

```
/*
Your action here.
*/
}
```

第 9 章 RFQ 応答

カスタマイズ例

この章では、3 つのカスタマイズ例を取り上げます。それは以下のとおりです。

- 応答のコピー
- RFQ 応答のライフ・サイクルからの撤回済み状態の削除
- 応答の作成時の説明情報の入力

シナリオ 1: 応答のコピー

このカスタマイズ例では、選択した RFQ 応答を基にして複製 RFQ 応答を作成することを可能にする RFQ 応答ツールに機能を追加します。カスタマイズには、**「複製」**ボタンの追加が含まれます。このボタンをクリックすると、複製応答の固有名を入力するためのプロンプトが表示されます。作成された新規応答は `draft` 状態になります。

インプリメンテーション概要

この応答の詳細情報をすべて検索し、検索された情報を使用して新規応答を作成します。現行スキーマでは応答名を同じにすることができないので、ユーザーが他の名前を指定するためのページを作成する必要があります。このページでは、このページに含まれるすべての情報を準備して、それを、応答作成イベントを起こす **RFQResponseCopyCmd** コマンドに渡すことができます。そして、UBF は **RFQResponseCreateCmd** コマンドを呼び出して応答を作成します。

カスタマイズ・ステップ

1. 以下のようにして、インターフェース **RFQResponseCopyCmd** とそのインプリメンテーションである **RFQResponseCopyCmdImpl** を追加します。
   a. 新しいインターフェース **RFQResponseCopyCmd** を追加します。インターフェースを定義するコードは以下のとおりです。

```
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. `com.ibm.commerce.ubf.commands.ToolsBusinessFlowEventCmdImpl` を拡張し、**RFQResponseCopyCmd** をインプリメントすることによって、コマンド・インプリメンテーション **RFQResponseCopyCmdImpl** を追加します。このコマンドは、ダイアログを通して送信されたデータを使用して応答作成イベントを起動します。イベントが発生すると、 UBF は **RFQResponseCreateCmdImpl** を呼び出して新しい応答を作成します。コマンドを定義するコードは以下のとおりです。

```
public void performExecute() throws ECException
{
   TypedProperty parms = null;
   BusinessFlowEventData data = null;
```

© Copyright IBM Corp. 2001, 2002

```
  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. URLREG テーブルで **RFQResponseCopy** を登録します。このコマンドは、以下の SQL を実行して登録します。

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

2. ユーザーが応答の名前を入力し、応答に関連したすべての情報が検索される JSP を追加します。

a. この JSP ファイルには、ユーザーが新しい応答名を指定するためのテキスト・フィールドがあります。以下は、このテキスト・フィールドのソース・コードです。

```
<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>
```

ユーザーが **「OK」**をクリックすると、以下のコードにより、入力フィールド内の応答名が保管されます。

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

この JSP ファイルの初期化時には、以下のコードによって、応答に関連したすべての情報が検索されて保管されます。情報は、**RFQResponseCopyCmdImpl** に提供されます。

```
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. 以下の SQL ステートメントを実行して、新しい JSP ファイルをマップする VIEWREG テーブルに表示コマンドを登録します。

```
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. rfq_response_duplicate_dialog.xml という新しい XML ファイルを以下のディレクトリーに追加します。

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 は、登録された URL をマップする **RFQResponseCopy** です。このダイアログのパネルは **RFQRspDuplicateDialog** で、これは登録された表示コマンドをマップします。 XML ファイルのコードは以下のとおりです。

```
<?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. この XML を resource.xml に登録します。

1) resource.xml を以下のディレクトリーにバックアップします。

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) resource.xml ファイルに以下の 2 行を追加します。

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

e. rfq_response_duplicate_dialog.js という新しい JavaScript ファイルを以下のディレクトリーに追加します。

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

JavaScript ファイルには、ダイアログによって使用されるいくつかの JavaScript 機能が含まれています。以下は JavaScript ファイルのソース・コードです。

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

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

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

3. この機能を RFQ 応答リスト・ページへ統合します。
   a. 応答リスト・ページに**「複製」**ボタンを追加します。すなわち、XML ファイルを以下のように変更します。

      注: 以下のディレクトリーにある XML ファイル rfq_response_list.xml をバックアップします。

      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

      rfq_response_list.xml 内の <button>... </button> ノードの終了に、以下の行を追加します。

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

   b. rfq_response_list.jsp に、新しい JavaScript 機能である getDuplicateBCT() および duplicateRes() を追加します。

```
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. rfq_response_list.jsp に JavaScript 機能を追加し、それを呼び出して、対応する RFQ の状態が ACTIVE でない場合は**「複製」**ボタンが使用不可にな

るようにします。以下の関数を rfq_response_list.jsp に追加し、これを DisableNewButton() を呼び出した後に呼び出します。

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

4. 以下のディレクトリーにある RFQMessages_en_US.properties ファイルに新しいメッセージを追加します。

   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.

   以下の行をファイルに追加します。このメッセージは、コピーが失敗した場合に表示されます。

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

シナリオ 2: RFQ 応答のライフ・サイクルからの撤回済み状態の削除

初期状態では、ユーザーが応答を撤回すると、応答は Active 状態から Retracted 状態に変化します。あるいは、応答が撤回後すぐに Draft 状態に設定されるようにしたければ、応答のライフ・サイクルから Retracted 状態を削除することによって機能をカスタマイズする必要があります。

インプリメンテーション概要

このカスタマイズでは、まず応答状態マシンから Retracted 状態を削除する必要があります。 Retracted 状態がなくなったら、応答リスト・ページの表示リストからも Retracted 状態を除去する必要があります。

カスタマイズ・ステップ

1. UBFStatemachine.xml および UBFStatemachine_en_US.xml を変更して、再ロードします。 UBFStatemachine.xml および UBFStatemachine_en_US.xml を以下のディレクトリーにバックアップします。

   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

これらの 2 つのファイルに以下の変更を加えます。

- retractRFQResponse 遷移のターゲット状態を、 RETRACTED から DRAFT に変更します。
- RETRACTED 状態と、この状態からのすべての遷移に関する情報を削除します。

以下のコード例は、UBFStatemachine.xml ファイルに必要な変更を表しています。

変更前

```
<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>
```

変更後

```
<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>
```

UBFStatemachine.xml に対する変更は以下のとおりです。

変更前

```
<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>
```

変更後

```
<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. 応答のライフ・サイクルには Retracted 状態がなくなっているので、rfq_response_list.xml および rfq_response_state_list.xml で以下の行を削除します。この時点の状態には、 draft、cancelled、pendingapproval、rejected、 active、inevaluation、win、lost、wincomplete、および lostcomplete が含まれます。

   注: ここでは表示の目的で行を改行しています。

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

注:

1. この例は、状態の削除方法を示しています。状態を追加する場合は、同じステップを順序を変えて実行します。
   a. 状態マシンの XML ファイルに新しい状態遷移を追加して、再ロードします。
   b. 応答リスト・ページのビュー・リストに新しいビューを追加します。

   状態マシンの再ロード方法については、UBF オンライン・ヘルプを参照してください。

シナリオ 3: 応答の作成時の説明情報の入力

RFQ への応答の作成時にいくつかの説明テキストを入力する場合は、「RFQ 作成」ウィザードの最初のパネルにテキスト・フィールドを追加する必要があります。

インプリメンテーションの概要

「RFQ response creation (RFQ 応答の作成)」ウィザードの最初のパネルに新しいテキスト・フィールドを追加します。 **RFQResponseCreateCmdImpl** は追加入力データである、新しいインターフェースの **MyRFQResponseCreateCmd** とそのインプリメンテーションを処理しないので、 **RFQResponseCreateCmd** および **RFQResponseCreateCmdImpl** を拡張することによってコマンド **MyRFQResponseCmdImpl** を作成します。

カスタマイズ・ステップ

1. インターフェース **MyRFQResponseCreateCmd** とそのインプリメンテーションである **MyRFQResponseCreateCmdImpl** を追加します。
   a. 新しいインターフェース **MyRFQResponseCreateCmd** を追加します。インターフェースのコードは以下のとおりです。

```
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. **RFQResponseCreateCmdImpl** を拡張し、 **MyFQResponseCreateCmd** をインプリメントすることによって、コマンド・インプリメンテーション **MyRFQResponseCreateCmdImpl** を追加します。ユーザー・インターフェースから転送された応答説明を保管するための新しいフィールド `responseDescription` を追加します。また、このフィールドの getter メソッドと setter メソッドも追加します。

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

   `initParameters()` メソッドをオーバーライドします。このメソッドは **RFQResponseCreateCmdImp** の `checkParameters()` によって呼び出されて、パラメーターを初期化します。このメソッドと、それに対応する super クラス内のメソッドの間の違いは、 **setResponseDescription (MyRFQConstants.EC_RFQ_RESPONSE_DESCRIPTION, false)** の呼び出しで、ユーザー・インターフェースから転送された応答説明が取得されるという点です。メソッドのソース・コードは以下のとおりです。

```
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);
}
```

createResponse() メソッドをオーバーライドします。**RFQResponseAccessBean** および **TradingDescriptionAccessBean** 内の説明フィールドを設定しなければならないという点を除けば、このメソッドは同一です。このメソッドのソース・コードは以下のとおりです。

```
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. 以下の SQL ステートメントを実行して **MyRFQReponseCreate** コマンドを URLREG テーブルに登録します。

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

2. RFQConstants を拡張することによって、新しい定数クラス MyRFQConstants を作成します。新しい定数定義は次の 1 つしかありません。

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

3. rfq_w_response_general.jsp ファイルを変更して、新しいテキスト・フィールドと関連処理を追加します。このファイルは以下のディレクトリーにあります。

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

Form セクションを以下のように変更します。

```
<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>
```

savePanelData() 関数に、応答説明を保管するコードを追加します。変更した関数は以下のようになります。

```
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);
}
```

retrievePanelData() 関数に、応答説明を検索するコードを追加します。変更したコードは以下のようになります。

```
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. rfq_response_wizard.xml ファイル内のターゲット finishURL を、RFQResponseCreate から MyRFQResponseCreate に変更します。ファイルは以下のディレクトリーにあります。

   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

   以下のコード・セグメントのとおりに変更します。

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

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

5. UBFStatemachine.xml ファイルを変更して、再ロードします。ファイルは以下のディレクトリーにあります。

   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

   両方のファイルで、アクション・インターフェースを com.ibm.commerce.rfq.commands.RFQResponseCreateCmd から com.ibm.commerce.rfq.commands.RFQResponseCreateCmd に変更します。以下のサンプル・コードは、必要な変更点を示しています。

   変更前

```
<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>
```

   変更後

```
<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: 状態マシンの再ロード方法については、UBF オンライン・ヘルプを参照してください。

第 10 章 予測在庫

カスタマイズ例

以下の例では、WebSphere Commerce アクセラレーターのこの部分をカスタマイズする方法を示します。

シナリオ 1: 予測在庫レコードの作成時に新しい情報を追加する

このシナリオでは、予測在庫ツールをカスタマイズして、予測在庫に予測在庫レコードを作成したユーザーなどの追加情報を記録するときに必要なステップを考慮します。

1. 適切な userID の RA テーブルに列を追加します。新しい ExpectedInventoryRecords エンタープライズ Bean を作成し、bean にアクセスします。
2. 新しい **CustomExpectedInventoryRecordCreateCmdImpl** をインプリメントします。それにより、コントローラー・コマンド・インプリメンテーション **ExpectedInventoryRecordCreateCmdImp** が拡張され、**ExpectedInventoryRecordCreateCmd** がインプリメントされます。
3. 新しいコントローラー・コマンドを CMDREG に登録します。顧客ストア ID は 1 であると想定し、次の SQL ステートメントを実行します。

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

4. **CustomExpectedInventoryRecordCreateCmdImpl** を更新します。このコマンドに、performExecute() メソッドをインプリメントします。

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

© Copyright IBM Corp. 2001, 2002

第 11 章 ビジネス・インテリジェンス

動的コンテキスト

動的コンテキストは、ユーザーが実行できるグループ化されたアクションのリストです。このリストには、アクションの名前と簡単な説明が示されています。リストは、ユーザーの役割と使用可能なコンポーネントに基づいて動的に変化します。

カスタマイズ例

シナリオ: 既存のコンテキストに新しいアクションを追加する

既存のコンテキストに新しいアクションを追加するには、以下のようにします。

1. アクションをコンテキスト定義 XML ファイルに追加します。たとえば、xml/tools/bi ディレクトリーの biContext.xml ファイルに追加します。追加先のコンテキストを探し、そこにエントリーを追加します。次に示すのは、キャンペーン・コンテキストのキャンペーン・アクションです。

```
<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>
```

   キャンペーン・コンテキストを表示するには、WebSphere Commerce アクセラレーターの**「マーケティング」**メニューから**「キャンペーン」**を選択し、**「レポート」**をクリックします。

   上記のコード部分では、toolsComponent はオプションの属性です。これを指定する場合、アクションは指定したコンポーネントが構成マネージャーから使用可能になる場合にのみリストされます。

   ユーザーが役割エレメントで定義されている役割のいずれかである場合、アクションはそのユーザー向けにリストされます。

   コマンド・エレメントの名前属性は、アクションをそのリストから選択するときに実行されるコマンドの名前になります。パラメーターおよび値属性は、コマンド・パラメーターおよびパラメーターの値になります。

   エントリー・エレメントの appendQueryString 属性は、上記のコード部分には示されていません。この属性が存在して true にセットされている場合、現在の要求の要求プロパティーは、名前 / 値のペアの形式でアクション・コマンドに付

© Copyright IBM Corp. 2001, 2002

加されます。たとえば、下のコード・サンプルには、アカウント・レポート・コンテキストでの Orders by Account Report アクションが示されています。

```
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 クラスは、上記の定義に基づいて次のコマンドを生成します (コマンドは本来 1 行ですが、見やすくするためにここでは行を分割しています)。

```
/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
```

上記のコマンドでは、要求プロパティーから次の名前 / 値のペアが取り出され、コマンドに付加されています。

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. プロパティー・ファイルにプロパティー・キーを追加します。

   各国語で使用できるように、プロパティー・ファイルの中では、nameKey、descriptionKey、および breadCrumbTrailTextKey がキーとなります。上記の例のプロパティー・ファイルは、
   properties/com/ibm/commerce/tools/bi/properties/BINLS_en_US.properties
   です。 nameKey は、アクションの名前に割り当てられます。 descriptionKey は、アクションの説明に割り当てられます。 breadCrumbTrailTextKey は、そのアクションが選択される際に、 breadcrumb trail (パンくず式道しるべ) に付加されたテキストに割り当てられます。

第 12 章 レポート・フレームワークの概要

レポート・フレームワークには、サイトのほとんどの部分で使用できる、汎用かつカスタマイズ可能なレポート機能が含まれています。レポートは、Commerce アクセラレーターを使用するすべての役割からアクセス可能です。特定のレポートへのアクセスについて、レポート内で定義して制限することができます。 WebSphere Commerce アクセラレーター・ユーザーは、いつでもレポートを要求することができ、フレームワークは実動データベースに含まれるデータを使用してレポートを生成し、レポートをリアルタイムで表示します。

フレームワークは、汎用コントローラー・コマンド、データ bean、そして結果を表示する汎用ビューとで構成されます。有効な SQL 照会を追加し、生成されたレポートを要求して表示するために使用する JSP ファイルを定義することで、フレームワークをカスタマイズできます。

レポート・フレームワークのカスタマイズ

レポートは、Commerce アクセラレーターからアクセスできます。したがって、各レポートには、関連した資産がたくさん必要です。レポートそのものは、タブ区切り形式で示されるデータで構成されますが、基礎となる資産は、レポート ID、SQL 照会、アクセス・コントロール・エレメントなどで構成されます。レポート要求は、サーバー上でコントローラー・コマンドを立ち上げます。コントローラー・コマンドは、レポートが特定のビューを指定しない限り、汎用ビューを設定するタスクを呼び出します。さらにこのコマンドは、たくさんの必須変数も設定し、このデータを戻して、宛先 JSP ファイルに ReportDataBean を取り込みます。レポート用のアクセス・コントロールは、レポートを要求 (入力) して表示するビューで設定されます。データベースから戻される結果は、ハッシュ・テーブルのベクトルとしてデータ bean に保管されます。最後に、JSP ファイルはレポートを表示します。レポートが空であれば、JSP ファイルは代わりに汎用テキスト・ストリングを表示します。

レポート用のアクセス・コントロールは、レポートを要求 (入力) して出力するビューで設定されます。

新しいレポートを追加するには、以下のステップが必要です。

1. XML ファイルでレポートを定義します。
2. 必要であれば、レポートの要求元の JSP ファイルを作成します。
3. 汎用 JSP を使用するのでない場合、レポートを表示する JSP ファイルを作成します。

カスタマイズ例

XML ファイルでのレポートの定義

個々のレポートは、XML ファイルを使用して定義されます。各レポートには、対応する *reportName*.xml ファイルがあります。このファイルには、レポートを生成す

© Copyright IBM Corp. 2001, 2002

るのに必要なすべての情報が含まれています。 *reportname*.xml ファイルは、MyStoreOverviewReport の次の例のようになります。

```
<?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>
```

新しいレポートを作成するには、上記の例のような XML ファイルを作成する必要があります。各 XML エレメントの詳細な説明について、 『有効な XML エレメント』を参照してください。

有効な XML エレメント: 以下の XML エレメントを使用してレポートを定義します。

<Reporting></Reporting>
これはルート・エレメントです。

<Report></Report>

この必須エレメントは、特定のレポートを定義します。複数の <Report> エレメントを組み込むことにより、 1 つの XML ファイルで複数のレポートを定義することができます。このエレメントには、2 つの必須属性があります。

ReportName
レポートの固有名を定義するストリング。

online レポートがリアルタイムに使用できるかどうかを指定するブール値。現在は、true の値だけがサポートされています。

<comment></comment>

レポートを説明するときのオプションのエレメント。このストリングは翻訳不要です。このエレメントは、既存の <Report> エレメント内でのみ定義できます。

<SQLvalue></SQLvalue>

レポートを生成するときに使用する SQL 照会を定義する必須エレメント。必須ですが、このエレメントは空である場合があります。このエレメントは、既存の <Report> エレメント内でのみ定義できます。

<mergeOperation></mergeOperation>

複数の SQL 照会を 1 つのレポートにまとめられるようにするオプション・エレメント。これには、他の <Report> エレメントの reportName を指す、コンマ区切りの reportName のリストが入れられます。参照されるレポートにあるすべての SQL 照会では、同じ数の列を戻し、ハッシュ・テーブルのキーを識別する同じ列名を使用する必要があります。各 SQL 照会は、他の照会には依存しません。それぞれの SQL 照会は、独自のハッシュ・テーブルのベクトルを生成し、表示される前にこれらのベクトルが相互に付加されて 1 つのレポートを形成します。

上記のセクションの Store Overview レポートの例には、 <mergeOperation> エレメントの使用方法が示されています。最後のレポートの最初の行は、1 つの SQL 照会によって生成され、 2 番目の行はその後の照会によって生成されます。このエレメントは、既存の <Report> エレメント内でのみ定義できます。

<extended_object_class></extended_object_class>

拡張 Java クラスを使用して SQL ステートメントを作成するときに使用する Java クラス名を含むオプション・エレメント。このクラスはレポートを生成してから、データをレポートの Control Center に戻します。このエレメントは、再帰的レポートを作成するときに使用します。このエレメントは、既存の <Report> エレメント内でのみ定義できます。

<display></display>

結果の表示に使用されるパラメーターを定義するときに使用するオプション・エレメント。このエレメントは、既存の <Report> エレメント内でのみ定義できます。

<standardInfo></standardInfo>

ReportDataBean の中で getter によってアクセスできるエレメントをグループ化するときに使用する必須エレメント。これらのエレメントは、ほとんど

のレポートにとって基本的なエレメントです。このエレメントは、既存の <display> エレメント内でのみ定義できます。

<resourceBundle></resourceBundle>
レポートのために使用されるプロパティー・ファイルを指定するときに使用する必須エレメント。この値は、reports/resources.xml ファイルの中でも参照される必要があります。このエレメントは、既存の <standardInfo> エレメント内でのみ定義できます。

<title></title>
レポートのタイトルを指定するのに使用する必須エレメント。この値は、プロパティー・ファイルの中のキーでなければなりません。このエレメントは、既存の <standardInfo> エレメント内でのみ定義できます。

<message></message>
レポートに関連したメッセージを表示するのに使用する必須エレメント。たとえば、レポートの説明をするときにこれを使用できます。この値は、プロパティー・ファイルの中のキーでなければなりません。このエレメントは、既存の <standardInfo> エレメント内でのみ定義できます。

<columnTitles></columnTitles>
列タイトルを定義するときに使用する必須エレメント。これには、コンマ区切りの名前のリストが入れられます。この名前は、プロパティー・ファイルで定義されたキーでなければなりません。プロパティー・ファイルでキーが見つけられない場合、列タイトルとしてエレメントに備えられているキーが使用されます。このエレメントは、既存の <standardInfo> エレメント内でのみ定義できます。

<userDefinedParameters></userDefinedParameters>
レポート・フレームワークでカスタム・エレメントを定義するときに使用するオプション・エレメント。レポート・フレームワークは、次のようなエレメントになると思われます。

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

ReportDataBean には、getter メソッドが用意されています。それは、カスタマイズ表示 JSP で使用される上記のエレメントのハッシュ・テーブルを戻します。このエレメントは、既存の <display> エレメント内でのみ定義できます。

注: <display> エレメント内に含まれるエレメントは必要に応じてリストされますが、このことはオプションの表示エレメントが定義されている場合のみ当てはまります。

SQL 照会での変数: *reportName*.xml ファイルで変数を定義する場合、その変数は中括弧 ({*variableName*}) で囲む必要があります。これは、レポート・フレームワークに対して、その値がクライアント変数であること、そしてその値をクライアント・ハッシュ・テーブルから入手しなければならないことを指示します。上記のサンプル XML ファイルでは、 {revenue}、{beginDate}、{endDate}、{storeent_id}、および {orders} は、すべてクライアント変数です。

レポートの要求元の JSP ファイルの作成

レポートを生成するのに必要な情報の量に応じて、必要なデータを収集するのに、ダイアログを使用するかウィザードを使用するかを決定する必要があります。どのエレメントが適切であるとしても、 JSP には savePanelData JavaScript 関数を含める必要があります。

```
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;
}
```

reportInputData および reportResultPage への参照は、パラメーターをコントローラー・コマンドに渡すのに必要です。 SQLid および reportXML 変数も必須です。 variable1 ～ variableN、そして varProperties はオプションです。例では、variable1 ～ variableN は SQL 照会で使用される変数を表しています。たとえば、variable1 および variable2 は、 beginDate および endDate で置き換えられます。したがって、savePanelData() 関数の内部には以下のコードが存在します。

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

varProperties 変数は、プロパティー・ファイルから値を入手する変数をリストします。たとえば、次のようになります。

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

JSP でオブジェクト reportResultPage が参照されない場合、レポート・フレームワークに備えられている汎用ビューを使用してレポートを表示するため、コントローラー・コマンドがそのオブジェクトを設定します。 reportResultPage.cmd を設定することにより、使用するビューを指定できます。

下記のコード・サンプルには、レポートの入力データを集めるときに使用される JSP ファイルの例が示されています。

```
<!-- ============================================================================
  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>
```

レポート・フレームワーク・コマンド

レポート・フレームワークでは、WebSphere Commerce に付属する以下のコマンドを使用します。

ビュー・コマンド

表 9. レポート・フレームワークによって使用されるビュー・コマンド

ビュー名	JSP ファイル
ReportRedirectView	¥tools¥reporting¥ReportRedirect.jsp
ReportGenericView	¥tools¥reporting¥ReportGenericView.jsp

コントローラー・コマンド

表 10. レポート・フレームワークによって使用されるコントローラー・コマンド

URL	インターフェース
GenericReportController	com.ibm.commerce.tools.reporting.command.GenericReportControllerCmd

詳細は、以下のパッケージの WebSphere Commerce に付属する JavaDoc ヘルプを参照してください。

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

レポート・フレームワーク・オブジェクト・モデル

結果 JSP ファイルを作成するときには、ReportDataBean を使用する必要があります。 populate() メソッドには、リアルタイム・レポートをサポートするロジックが含まれています。データ bean では、以下のメソッドを使用できます。

表 11.

メソッド名	戻りタイプ	説明
populate()	void	レポートを生成するために SQL 照会を実行する
getErrorCode()	int	エラー・コードの getter メソッド
getNumberOfColumns()	int	レポートの列数のための getter メソッド
getNumberOfRows()	int	レポートの行のための getter メソッド
getColumnTitlesName(int)	ストリング	(i+1) 列名のための getter メソッド
getRow(i)	ハッシュ・テーブル	(i+1) 行のための getter メソッド
getValue(i,j)	ストリング	レポートの (i+1,j+1) 値の getter
getValue(i,keyname)	ストリング	キー名に関連付けられた列の (i+1) 行の getter メソッド
getUserDefinedParameters()	ハッシュ・テーブル	*reportName*.xml のユーザー定義パラメーターから作成したハッシュ・テーブルの getter メソッド
getEnv()	ハッシュ・テーブル	入力 JSP ファイルで定義した入力パラメーターを含むハッシュ・テーブルの getter メソッド

詳細は、WebSphere Commerce に付属する JavaDoc ヘルプを参照してください。

レポート JSP ファイルの再利用可能コンポーネント

レポート・フレームワークを調整するレポート JSP には、レポートの入力および出力ビューが用意されています。

各レポートには、入力ビューと出力ビューがあります。すべての JSP のタスクまたは目的は同じです。ルック・アンド・フィールは同じで、共用する入力ウィジェット・プールから異なる入力基準を要求します。一方で、出力ビューには共通した構造および形式があります。レポートの性質に基づき、レポート JSP の再利用可能部分はすべて、共用可能なコンポーネントとして作成されます。それらのコンポーネントは、以下のファイル構造および命名規則に基づいて作成されます。

ReportName は、新しいレポートの名前に置き換えられ、 *InputComponent* は、入力コンポーネント名に置き換えられます。すべての入力および出力レポート JSP ファイルでは、`Reports.properties` を使用することによって、マルチリンガルの問題を解決します。したがって、プロパティー・ファイルは、XML および JSP ファイルで使用されているすべてのタイトルとキーを割り当てます。

CSA 運用レポート用に最初に作成されているコンポーネントのリストを示します。

ReportDaysWaitedHelper.jsp
待機期間編集可能ボックス・コンポーネント

ReportFulfillmentHelper.jsp
フルフィルメント選択コンポーネント

ReportInventoryAdjustmentCodeHelper.jsp
在庫調整コード選択

ReportProductHelper.jsp
製品選択コンポーネント

ReportStartDateEndDateHelper.jsp
日付入力コンポーネントのペア

ReportVendorHelper.jsp
取引先選択コンポーネント

ReportFrameworkHelper.jsp
入力 JSP ファイルの共通機能

ReportOutputHelper.jsp
出力ページ・フォーマッター・コンポーネント

ReportProductFindDialogView.jsp
検索基準入力ページ

ProductSearch.jsp
検索結果選択ページ

`ReportProductFindView` および `ReportProductSearchView` は、`ReportProductHelper` によって使用される独立したページです。

Common.jsp は、レポートの入力ページと出力ページの両方で共用されます。 ReportFrameworkHelper は、すべてのレポート入力 JSP ファイルによって共用され、 ReportOutputHelper は、すべてのレポート出力ページに適用されます。

他の JSP ファイルはすべて入力コンポーネントであり、入力を必要とするレポート入力ページにドラッグできます。

各レポートには、3 つの XML ファイルが必要です。 *ReportName*ReportDefinition.xml は、レポートに使用される SQL ステートメント、そしてレポート内の各列の形式を定義します。 *ReportName*ReportDefinition.xml の作成方法については、 **Report Framework Design の資料**を参照してください。

*ReportName*ReportInputDialog.xml は、ダイアログ定義です。次のような構文になります。

```
<?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>
```

ReportName は、レポート名に置き換えます。ウィンドウ・タイトルが Reports.properties ファイルに記入されます。したがって、reporting.reportStrings は、xml/tools/reporting/resources.xml ファイルで定義され、properties/com/ibm/commerce/tools/reporting/properties/Reports.properties を指します。ヘルプ・キーは CMHelpMap.xml で記入されます。最後に、URL は、レポート入力ビュー JSP のビュー・コマンド名になります。

*ReportName*ReportOutputDialog.xml は、レポート出力プロパティーを指定するもう 1 つのダイアログ定義です。その形式は、次のとおりです。

```
<?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>
```

上記のサンプル出力ビューには、**「印刷」**、**「OK」**という 2 つのカスタマイズされたボタンがあります。それぞれの処理機能については、出力ビュー JSP で定義されます。また、ウィンドウのタイトルはプロパティー・ファイルに記入されますが、ヘルプ・キーは XML ファイルに記入されます。

設計は、再利用可能なコンポーネントの概念に基づきます。入力ビュー JSP ファイルは、レポートの仕様に合わせて、一群の基準項目で構成されています。基準項目は別のレポートの入力ページに示すことができるものなので、各基準項目は再利用可能なコンポーネントとして作成されます。この戦略を入力ビュー JSP ページの設計に適用すると、以下のことを実現できます。

1. 新しいレポート入力ビューを簡単に作成できる。
2. すべてのレポート入力ビューで一貫したルック・アンド・フィールを使用できる。
3. ツール・フレームワークで必要とされる機能は、それぞれコンポーネントで作成されるため、機能を簡単に検査、ロードおよび保管することができる。

出力ビュー JSP ファイルも、再利用可能な helper で作成されます。 Helper には、データ・タイプや言語の設定に応じて、必要なすべての書式設定および変換機能が備えられています。各列のデータ・タイプは、レポート定義 XML のユーザー定義のセクションで指定されます。 helper および XML 定義を調整することにより、出力ビューの作成が簡単になると同時に、洗練された出力形式がサポートされるようにもなります。

次のセクションでは、再利用可能なコンポーネントを使用して、レポートの入力ビューおよび出力ビュー JSP ファイルの作成方法を説明します。

レポートの入力ビュー JSP ファイルを作成するには、 JSP ファイルを必要とする必須コンポーネントをインポートしなければなりません。以下のコンポーネントが使用できます。

1. DaysWaited - 今日まで待機した日数を指定します。
2. FulfillmentCenter - 配送センターの選択を実行します。
3. InventoryAdjustment - 在庫調節コードの選択を実行します。
4. StartDateEndDate - 特定期間を指定します。
5. Vendor - 取引先の選択を実行します。

設計の戦略を満たしている限り、他のコンポーネントを作成することができます。helper の作成については、後で説明します。

これらのコンポーネントはすべて、 JSP ページ開発者に対して共通のインターフェースを提供します。

1. JSP 関数

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

   この関数は、入力ビューについてのコンポーネントを作成します。 *InputComponent* は、コンポーネントの名前です。各コンポーネントには containerName があります。これは、このコンポーネント用に JavaScript オブジェクトを識別する固有名です。 JavaScript 関数はすべて、containerName を参照します。すべてのコンポーネントで、レポート・リソース・バンドル `reportsRB`

が必要です。これは、現在の言語の設定を反映します。各コンポーネントには、ラベルから割り当てられた少なくとも 1 つのタイトルがあります。

2. JavaScript 関数

function onLoad*InputComponent***(containerName)**
この関数は、ページをロードするときに呼び出されます。ページがトランザクション内でロードされるのが初めての場合、 (データ bean から) *InputComponent* を初期設定します。このページがトランザクション内で再ロードされるものであれば、保管したデータを取り出します。

function validate*InputComponent***(containerName)**
この関数は、要求を実行依頼する前に呼び出す必要があります。これは、このコンポーネントの入力データを検証します。データが無効の場合、ダイアログ・ウィンドウが表示され、適切なメッセージが示されて無効であることを通知します。データが有効であれば true を戻し、データのいずれかの部分が無効であれば false を戻します。

function save*InputComponent***(containerName)**
この関数は、現在のページとは異なる他のページに移動する場合に、必ず呼び出す必要があります。これは、現在のページに戻るときに後で検索できるように、コンポーネントの現在の入力データを保管します。

function visibleList(state)
コールバック関数。これは、フレームワークが選択ボックスを表示するか、ページの選択ボックスを隠す場合に呼び出されます。この関数は、次のものを呼び出さなければなりません。

setSelect*Component***Visible(container, state)**
選択ボックスが使用される各入力コンポーネントで定義されます。入力コンポーネントにはすべて、名前が異なる同じインプリメンテーションがあります。

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

レポート入力 JSP ページは、*ReportName* ReportInputView.jsp として指定する必要があります。これは、次のディレクトリーにあります。

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

これは、ダイアログ・パネルです。ダイアログ・パネルをインプリメントするに

は、Tools Framework User's Guide を参照してください。ダイアログ・パネルには、2 つのセクションがあります。 HTML コンテンツ生成のセクションと JavaScript 関数のセクションです。 HTML セクションでは、入力画面に表示するコンポーネントごとに、 generate*InputComponent* を呼び出すことができます。 JavaScript セクションでは、initializeValue、savePanelData、および validatePanelData は、各入力コンポーネントで定義された対応する JavaScript 関数を呼び出さなければなりません。 initializedValue は、ページがロードされるときに呼び出されます。ツール・フレームワークは、savePanelData および validatePanelData を呼び出します。 SavePanelData は、レポート・フレームワークで必要な以下の JavaScript 関数を呼び出します。

1. setReportFrameworkOutputView("DialogView");
2. setReportFrameworkParameter("XMLFile","reporting.*OutputPanelName*")
3. setReportFrameworkReportXML("reporting.*ReportDefinitionXML*");
4. setReportFrameworkReportName("*SQLName*"); これは *ReportXML* で指定されます。

さらに、レポート出力 JSP ファイルで必要なパラメーターを設定するために、setReportFrameworkParameter("*name*", *value*) を呼び出すことも可能です。これは、名前と値のペアです。ジェネレーターに渡されるすべてのラベルと、setReportFrameworkParameter 関数呼び出しの値は、ロケールおよび言語の設定に基づく正しいストリングに割り当てられるキーです。この割り当ては、次のディレクトリーのプロパティー・ファイル Reports_en_US.properties で定義されます。

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

レポート入力および出力ページのための helper の使用

ダイアログ・パネルであるレポート入力ページでは、入力基準のための一群のコンポーネントが必要であり、以下の 4 つの Java スクリプト関数をインプリメントする必要があります。

initializeValues()

ページがロードされるたびに呼び出されます。

savePanelData()
このページからユーザーが出るたびに呼び出されます。

validatePanelData()
基準をレポート・フレームワークに送信する前に呼び出されます。

visibleList()
レポート・フレームワークで、このページでのコンポーネントの表示設定を変更する必要がある場合に呼び出されます。

レポート入力ページにコンポーネントを追加するには、コンポーネント JSP をインポートし、入力ページに 1 つの JSP 式を追加します。命名規則 generate*InputComponent* により、パラメーターとして、コンテナー名 (このページに固有な名前)、リソース・バンドル、および 1 つ以上のタイトルが指定されます。たとえば、入力ページで、次のようになっているとします。

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

この式は、ページに表示されるコンポーネントを生成するストリングを戻します。上記の 3 つの関数を調整するために、命名規則 onLoad*InputComponent*、save*InputComponent*、および validate*InputComponent* でコールバック関数が定義されます。これらはそれぞれ、initializeValue、savePanelData、および validatePanelDate 内で呼び出す必要があります。

SavePanelData 関数は、レポート・フレームワークで必要な情報も保管します。各入力コンポーネントには、そのコンポーネントに入力された ID、名前、または他のフィールドを戻す戻り関数が含まれています。これらの戻り関数の詳細については、いずれかの入力コンポーネント JSP を参照してください。

出力ページは、書式の設定を扱います。書式設定メソッドはすべて、ReportOutputHelper に含まれていて、レポート定義 XML ファイルを調整しています。レポート出力 JSP ファイルには、レポート名だけを指定する必要があります。任意のレポート出力 JSP ファイルをコピーし、reportPrefix 値をレポート名を反映するよう変更できます。

しかし、レポート定義 XML ファイルでは、列タイプに基づいて、すべての列およびその書式設定を指定します。列タイプのリストを以下に示します。

表 12.

列タイプ	デフォルト	カスタマイズ可能なプロパティー	デフォルト配置
ストリング	あり	maxEntryLength	右
整数	なし	setMinimumIntegerDigits setMaximumIntegerDigits	左
小数	なし	setMinimumIntegerDigits setMaximumIntegerDigits setMinimumFractionDigits setMaximumFractionDigits	左
通貨	なし	currencySymbolColumn	左

表 12. (続き)

列タイプ	デフォルト	カスタマイズ可能なプロパティー	デフォルト配置
列挙	なし		右
日付	なし		左
時刻	なし		左
月	なし		左

デフォルトの列タイプは、HTML 列オプション "align=left height=20 nowrap" の指定されたストリングです。すべての列タイプで、列に <columnOptions> タグを指定することにより、デフォルトの列オプションをオーバーライドすることができます。

オプションとして、すべての列で、それぞれの displayInReport 値を、 true か false のいずれかに設定できます。デフォルトは true ですが、これはレポートに列が表示されることを示します。値が false に設定されると、列は隠されます。この機能は、SQL 照会を変更せずにレポート出力ビューをカスタマイズするときに使用できます。また、通貨を書式設定する場合に、使用する通貨をフォーマッターに通知するために参照列を必要とする場合にも有効です。

整数、小数、日付、および時刻列は、コマンド・コンテキストで指定した言語と通貨の値に基づいて書式設定されます。整数および小数列では、整数の値と小数点以下の値の両方のために、最小および最大桁数を指定することもできます。

通貨列は、デフォルトでは、コマンド・コンテキストで指定した言語と通貨の値に基づいて書式設定されます。通貨列に currencySymbolColumn が指定される場合、3 文字の通貨記号がデータベースから取り出され、通貨を書式設定するために使用されます。レポート作成者が通貨記号のストリングを表示しないのであれば、参照される通貨記号列は表示しないように設定できます。

列挙列は、特別な列タイプで、データベースから取り出した値を、キーによって指定されたストリングに割り当てます。たとえば、テーブルから Y または N が取り出される場合があります。これらの値は、Yes か No などのより意味のあるストリングに割り当てることができます。別のレポートあるいは別の言語では、 Approved や Denied に割り当てることができます。これを実現するために、列を次のように定義することができます。

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

または

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

ここで、Yes、No、Approved、および Denied のストリングは、適切なプロパティー・ファイルで定義されていて、複数の言語に対応することができます。照会が

0、1、2 のような値 (特定の値の割り当て先にしようとする数字) を戻す場合、エレメントとして <X_n></X_n> を使用する必要があります。たとえば、

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

再利用可能な JSP ページ・コンポーネントを使用するレポートの作成

再利用可能な JSP ページ・コンポーネントを使用してレポートを作成するには、以下のものを作成する必要があります。

- JSP ファイル
  XXXReportInputView.jsp
  XXXReportOutputView.jsp
- XML ファイル
  XXXReportInputDialog.xml
  XXXReportDefinition.xml
  XXXReportOutputDialog.xml
- 更新済みプロパティー・ファイル
  Reports_en_US.properties (必要なすべての割り当てのためのセクションを追加する必要があります。)
- ビュー・コマンドをデータベースに追加する。
  XXXReportInputView および XXXReportOutputView コマンドを、データベースに追加する必要があります。
- 前のステップで追加した 2 つのビュー (XXXReportInputView および XXXReportOutputView) にアクセス・コントロールを設定する。

第 13 章 商品アドバイザー

カスタマイズ例

本書では、3 つのカスタマイズ・サンプルを取り上げます。それは以下のとおりです。

- 別のオペレーター・アイコンを使用するようにサンプル商品探査 JSP ファイルを変更する。
- 商品比較で使用するデフォルトのリンクを変更する。
- 提供されたウィジェットを使用せずに商品探査値をレンダリングする。

シナリオ 1: 別のオペレーター・アイコンを使用する

商品探査メタフォーでは、演算子アイコンに対する選択の X 座標 (1 つのイメージがすべての演算子を表す) を使用して、選択された演算子を判別します。イメージを置き換える場合、そのイメージをクリックしたときに適切な演算子が識別されるように、座標は同じままにしておきます。イメージ・ファイルは、CommerceDir¥web¥tools¥pa¥icons にあります。 equalone.gif は、=、<> 演算子を表し、equaltoo.gif は、数値属性値に使用される =、<>、<=、>= 演算子を表します。予測される X 座標は以下のとおりです。

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

値の選択によって X 座標のパラメーターが渡されると、適切な演算子がその選択に適用されます。

シナリオ 2: 商品比較メタフォーからリンクする

商品比較メタフォーは、個々のアイテムから別のページへのリンクをサポートします。現在は、(URL を値とする属性を作成しない限り) 1 つのリンク先ページだけがサポートされています。サンプル JSP では、ProductDisplay ページにリンクしています。このページを、ProductCompareDataBean の productLinkName プロパティーを使用して割り当てます。サンプルでは、商品比較メタフォーの使用状況に関する統計を収集する ClickInfo コマンドによってリダイレクトされます。他のページにリンクするときには、商品比較ページからパラメーターを移動することが必要な場合もあります。 ProductCompareDataBean の productLinkParameters プロパティーで、移動するパラメーターを確認します。該当パラメーターが存在する場合、ここで確認されるすべてのパラメーター名により、そのパラメーターはリンク・ページに移動します。さらに、サンプルでは ECConstants.EC_PRODUCT_ID として示されている productId パラメーターには、リンクが関連付けられているアイテムのカタログ・エントリー ID (catentry_id) の値が割り当てられます。この productId により、リンク・ページは、ユーザーが選択したアイテムを識別できます。現在は、商品比較テーブル内の個々のアイテムに固有な値が指定された他のパラメーターはサポートされていません。

© Copyright IBM Corp. 2001, 2002

シナリオ 3: 商品探査の戻しのカスタマイズ

ProductExploreDataBean は、DynamicForm ウィジェットによってレンダリングされます。レンダリングするには、DynamicForm クラス (メソッド・シグニチャー用の JavaDocs を参照) をサブクラス化してレンダリング・メソッドをオーバーライドするか、 ProductExploreDataBean から直接にデータを入手します。レンダリング・メソッドの中では、getDataBean() メソッドを使用して、ProductExploreDataBean オブジェクトを入手します。各属性列を表す ColumnDataBeans の集合を入手するには、ProductExploreDataBean.getFormElements() メソッドを使用します。それぞれの列の中では、その列の値は ColumnDataBean オブジェクトに入っています。個々の属性値を含む DsData オブジェクトの集合を入手するには、ColumnDataBean.getColumn() メソッドを使用します。メソッドの詳細は、ColumnDataBean および DsData についての JavaDocs を参照してください。属性値の書式設定された表記を取り出すには、DsData の getPresentationString() メソッドを使用し、生データには getUnformattedData() メソッドを使用します。使用する HTML フォームに適したパラメーターを作成する場合、正しいパラメーター名を取り出すには ColumnDataBean.getFormElementName() メソッドを、それぞれの値を入手するには DsData.getUnformattedData() メソッドを使用します。適切な演算子の選択については、演算子アイコンの情報を参照してください。

第 14 章 ルール・プロジェクト

ルール・プロジェクトのライフ・サイクルには、以下の段階が含まれます。

1. ルール・プロジェクトの作成
2. 新しいルール・プロジェクトに基づいたルール・サービスの構成
3. ルール・サービスの呼び出し
4. ルール・プロジェクトに基づいたルール・サービスの除去

前提事項: ここでの情報は、 2、3、および 4 のみを扱っています。ここでは、すでにルール・プロジェクトが作成されていることを前提としています。

ルール・プロジェクトの作成方法の詳細については、 Blaze プロフェッショナル・サービスにご相談ください。

カスタマイズしたルール・プロジェクトに基づいたルール・サービスの構成方法

ルール・システムは、 WebSphere Commerce に付属しているルール・プロジェクトとカスタマイズしたルール・プロジェクトを区別しません。管理コンソールを使用することにより、ルール・プロジェクトからのルール・サービスの作成、変更、または除去をいつでも実行できます。ただし、カスタマイズしたルール・プロジェクトでは、特定の要件を満たしていなければなりません。以下の要件があります。

1. ルール・プロジェクトに通知される外部イベントは、ルール・プロジェクトの実行を起動しなければなりません。
2. 外部イベントでは `com.ibm.commerce.rules.InvocationContext` インターフェースをインプリメントしなければなりません。このインターフェースは単純なタグ・インターフェースに過ぎません。

カスタマイズしたルール・プロジェクトに基づいたルール・サービスの呼び出し方法

ルール・サービス (前述の 2 つの基準を満たしているルール・プロジェクトから作成されたもの) は、 WebSphere Commerce に付属しているルール・プロジェクトから作成されたルール・サービスと同じようにして、呼び出すことができます。つまり、以下のタスク・コマンドを呼び出すことによって実行します。

`com.ibm.commerce.rules.commands.InvokePersonalizationRuleServiceCommand`

© Copyright IBM Corp. 2001, 2002

第 2 部 付録

© Copyright IBM Corp. 2001, 2002

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、米国以外の国においては本書で述べる製品、サービス、またはプログラムを提供しない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。ただし、IBM 以外の製品、プログラムまたはサービスの操作性の評価および検証は、お客様の責任で行っていただきます。

本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、またはサービスのみが使用可能であることを意味するものではありません。 IBM 製品、プログラムまたはサービスに代えて、 IBM の知的所有権を侵害することのない機能的に同等のプログラムまたは製品を使用することができます。ただし、IBM によって明示的に指定されたものを除き、他社の製品と組み合わせた場合の動作の評価と検証はお客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について実施権を許諾することを意味するものではありません。実施権の許諾については、下記の宛先に書面にてご照会ください。

〒106-0032 東京都港区六本木 3 丁目 2-31
IBM World Trade Asia Corporation
Intellectual Property Law & Licensing

以下の保証は、国または地域の法律に沿わない場合は、適用されません。

IBM およびその直接または間接の子会社は、本書を特定物として現存するままの状態で提供し、商品性の保証、特定目的適合性の保証および法律上の瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものとします。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

本書は定期的に見直され、必要な変更 (たとえば、技術的に不適切な表現や誤植など) は、本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものではありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

© Copyright IBM Corp. 2001, 2002

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、 (i) 独自に作成したプログラムとその他のプログラム (本プログラムを含む) との間での情報交換、および (ii) 交換された情報の相互利用を可能にすることを目的として、本プログラムに関する情報を必要とする方は、下記に連絡してください。

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

本プログラムに関する上記の情報は、適切な使用条件の下で使用することができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で決定されたものです。そのため、他の操作環境で得られた結果は、異なる可能性があります。一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。さらに、一部の測定値が、推定値である可能性があります。実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、もしくはその他の公に利用可能なソースから入手したものです。 IBM は、それらの製品のテストは行っておりません。したがって、他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

IBM の将来の方向または意向に関する記述については、予告なしに変更または撤回される場合があり、単に目標を示しているものです。

本書はプランニング目的としてのみ記述されています。記述内容は製品が使用可能になる前に変更になる場合があります。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。より具体性を与えるために、それらの例には、個人、企業、ブランド、あるいは製品などの名前が含まれている場合があります。これらの名称はすべて架空のものであり、名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

この製品で使用されているクレジット・カードのイメージ、商標、商号は、そのクレジット・カードを利用して支払うことを、それら商標等の所有者によって許可された人のみが、使用することができます。

商標

以下は、IBM Corporation の商標です。

Microsoft、Windows、Windows NT および Windows ロゴは、 Microsoft Corporation の米国およびその他の国における商標です。

Java およびすべての Java 関連の商標およびロゴは、 Sun Microsystems, Inc. の米国およびその他の国における商標または登録商標です。

他の会社名、製品名およびサービス名などはそれぞれ各社の商標または登録商標です。

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12