IBM® WebSphere™ Commerce

WebSphere Commerce 액셀러레이터 사용자 정의 안내서

버전 5.4

IBM® WebSphere™ Commerce

WebSphere Commerce 액셀러레이터 사용자 정의 안내서

버전 5.4

주!

이 책 및 이 책이 지원하는 제품을 사용하기 전에 103 페이지의 『주의사항』에 있는 일반 정보를 읽으십시오.

초판(2002년 6월)

이 개정판은 새 개정판에 별도로 명시하지 않는 한 IBM® WebSphere Commerce와 모든 후속 릴리스 및 수정에 적용됩니다. 제품 레벨에 맞는 올바른 버전을 사용하고 있는지 확인하십시오.

한국 IBM 담당자나 IBM 지사를 통해 서적을 주문하십시오. 다음 주소에서는 책을 구비하고 있지 않습니다.

IBM은 여러분의 의견을 환영합니다. 다음 중 한 가지 방법으로 사용자 의견을 보내실 수 있습니다.

1. 아래로 전자 우편을 보내십시오. 응답을 원하시는 경우 사용자의 전체 네트워크 주소를 기입해 주십시오.

   인터넷: ibmspoe@kr.ibm.com
2. 다음 팩스 번호를 사용하십시오.

   Fax:02-3781-5200
3. 우편으로 보내실 경우에는 다음 주소로 우송해 주십시오.

   135-270
   서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
   한국 아이.비.엠 주식회사
   고객만족센터
   전화번호: 080-023-8080

IBM에 정보를 보낼 경우, 그 정보가 타당하면 IBM은 적절한 방식으로 이를 사용하거나 배포할 수 있으며, 제공한 독자는 이에 대해 책임을 지거나 사용에 제한을 받지 않습니다.

목차

시작하기 전에

이 책에 사용된 규칙

이 책은 다음 강조표시 규칙을 사용합니다.

굵은체는 필드, 아이콘 또는 메뉴 선택사항의 이름과 같은 GUI(Guaphical User Interface) 제어 또는 명령을 표시합니다.

모노체는 디렉토리 경로 및 정확하게 입력해야 하는 텍스트의 예를 나타냅니다.

*기울임꼴*은 사용자가 값을 대체해야 하는 변수 및 강조에 사용됩니다.

이 아이콘은 태스크 완료에 도움을 주는 추가 정보를 표시합니다.

지식 요구사항

WebSphere Commerce 액셀러레이터를 사용자 정의하려면, 다음에 대한 지식이 필요합니다.

- HTML 및 XML
- SQL(Structured Query Language)
- Java 프로그래밍

WebSphere Commerce 사용자 정의에 대한 추가 정보는 WebSphere Commerce 프로그래머 안내서를 참조하십시오. 이 책은 다음 웹 사이트에서 구할 수 있습니다.

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

이 책의 구성 방법

다음 표는 책의 구성을 요약한 것입니다.

표 1.

구성요소	설명	위치
비즈니스 관계 관리	장기 구매 계약 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	3 페이지의 제 1 장 『비즈니스 관계 관리』
고객 서비스 영업대표	고객 서비스 영업대표 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	9 페이지의 제 2 장 『고객 서비스 영업대표 도구』

표 1. (계속)

구성요소	설명	위치
고객 프로파일	고객 프로파일 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	21 페이지의 제 4 장『고객 프로파일』
캠페인	캠페인 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	31 페이지의 제 5 장『캠페인』
쿠폰	쿠폰 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	57 페이지의 제 7 장『쿠폰』
할인	할인 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	59 페이지의 제 8 장『할인』
RFQ	RFQ 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	63 페이지의 제 9 장『RFQ 응답』
재고	재고 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	77 페이지의 제 10 장『예상 재고』
비즈니스 인텔리전스	보고 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	79 페이지의 제 11 장『비즈니스 정보』
보고 프레임워크	보고 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	81 페이지의 제 12 장『보고 프레임워크의 개요』
검색	카탈로그 검색 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	39 페이지의 제 6 장『카탈로그 검색』
Blaze 규칙 어드바이저	Brokat 어드바이저 규칙 프로젝트를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	99 페이지의 제 14 장『규칙 프로젝트』
상품 어드바이저	상품 어드바이저 도구를 사용자 정의할 때 고려해야 하는 부분들에 대해 자세히 설명합니다.	97 페이지의 제 13 장『상품 어드바이저』

제 1 부 WebSphere Commerce 액셀러레이터 사용자 정의

이 책에서는 WebSphere Commerce 액셀러레이터를 사용자 정의하는 방법에 대해 설명합니다. WebSphere Commerce 액셀러레이터의 설계 결정에 대한 배경 지식을 제공하여, 사용자 정의에 접근하는 방법을 알려주고 사용자 정의에 필요한 단계를 자세히 설명합니다.

이 책은 WebSphere Commerce 액셀러레이터의 사용자 정의 과정을 안내하기 위한 것이지만, 가능한 모든 사용자 정의 목록을 살펴보도록 설계된 것이 아닙니다. 대신, WebSphere Commerce 액셀러레이터를 구성하는 다른 일반 부분들을 소개하여 이 부분들을 사용자 정의하는 방법에 대해 설명합니다.

WebSphere Commerce 액셀러레이터는 사용자 사이트의 일일 비즈니스 조작을 쉽게 하기 위해 설계된 도구들의 콜렉션입니다. 즉, 비즈니스를 하는 사람들이 계속해서 사이트 운영에 대한 책임이 있는 IT 직원과 연락하지 않아도 사이트 운영의 여러 측면을 작성 및 수정할 수 있는 인터페이스로 설계되었습니다. 이와 같이, 각 비즈니의 주요 측면을 목표로 하는 여러 구성요소가 WebSphere Commerce 액셀러레이터를 구성합니다. 이러한 도구를 보편화하기 위해 모든 노력을 기울였지만 동일한 목표로 일부 사용자는 특정 요구사항이 충족되지 않았음을 알 수 있습니다. 사이트 작성 중 데이터베이스를 확장하였으며, 이 확장이 책에 나열된 도구 중 하나와 관련된 경우, 데이터는 관련 도구를 사용자 정의할 때까지 액셀러레이터에 표시되지 않습니다.

이 책은 WebSphere Commerce 액셀러레이터에서 다음 구성요소를 사용자 정의하는 방법에 대해 설명합니다.

제 1 장 비즈니스 관계 관리

이 장에 설명된 요소는 WebSphere Commerce 액셀러레이터의 계정 및 장기 구매 계약 구성요소에 대한 사용자 인터페이스를 나타냅니다. 이는 판매 관리자나 회계 담당이 비즈니스와 연관된 당일의 태스크를 수행하기 위해 필요로 하는 계정, 장기 구매 계약 및 기타 조치의 작성 및 유지보수를 용이하게 합니다. 비즈니스 관계 관리 구성요소에는 다음 요소들이 있습니다.

- 계정 노트북
- 장기 구매 계약 노트북

사용자 정의 예

다음 예에서는 WebSphere Commerce 액셀러레이터의 해당 부분을 사용자 정의하는 방에 대해 간략하게 보여줍니다.

시나리오 1: 장기 구매 계약 사용자 인터페이스에 새 규정 추가

구현 개요

가능하면 많은 사용자들에게 WebSphere Commerce 액셀러레이터를 일반화하고 유용하게 만들기 위해 일부 규정이 생략되었습니다. 그러므로 사용자의 비즈니스에 더 적합하도록 만들려면 사이트에 규정을 추가해야 합니다. 이 시나리오는 추가 규정을 사이트에 추가하는 데 필요한 단계를 설명합니다.

사용자 정의 단계

1. 서버에서 규정을 정의하십시오. 사용자 인터페이스 사용자 정의를 수행하기 전에, 먼저 다음을 완료해야 합니다.
   - `B2BTrading.dtd` 파일에서 새 규정을 정의하십시오. 자세한 정보는 *IBM WebSphere Commerce 프로그래머 안내서*의 제 7 장을 참조하십시오.
   - 새 규정에 대해 엔터프라이즈 Bean이나 액세스 Bean을 작성하십시오.
2. 사용자 인터페이스에 규정을 추가하십시오. 필요한 요소를 사용자 인터페이스에 추가하려면, 다음을 수행하십시오.
   a. 규정에 대한 사용자 인터페이스 페이지의 JSP 파일을 작성하십시오. 이 페이지에는 페이지상의 입력 필드에서 데이터를 캡처하기 위한 동적 JavaScript 오브젝트가 있어야 합니다. 액세스 Bean에서 데이터 로드를 캡슐화하는 데이터 bean을 작성하십시오. 선택적으로 페이지에 직접 액세스 bean을 포함하여 데이터베

이스로부터 규정을 로드하십시오. contractId 매개변수가 모든 페이지에 전달되어 데이터 로드 성능이 감소되도록 합니다. JSP 파일의 견본은 다음과 같습니다.

```
<!--========================================================================
//*------------------------------------------------------------------------
//* 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. 규정에 대한 사용자 인터페이스 페이지의 JavaScript 파일을 작성하십시오. JavaScript 데이터의 유효성을 확인하고 제출하기 위한 기능을 작성하십시오. 데이터를 제출할 때, 새 규정의 XML 형식과 일치하는 새 Javascript 오브젝트를 작성해야 합니다. JavaScript 파일의 견본은 다음과 같습니다.

```
//*-------------------------------------------------------------------------------------
//* 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. 2b단계에서 작성된 새 submit 및 validate 함수를 호출하도록 Contract.js 파일을 수정하십시오.

d. 자원 번들을 수정하여 필요로 하는 새 문자열을 추가하십시오.

e. ContractNotebook.xml을 수정하여 새 패널을 추가하십시오.

시나리오 2: 장기 구매 계약 사용자 인터페이스에서 규정 제거

구현 개요

일부 규정이 누락된 것을 알 수 있는 것처럼, 비즈니스의 일부 사항이 불필요하다는 것을 알 수 있습니다. 이 시나리오는 사용자 사이트에서 불필요한 규정을 제거하는 데 필요한 단계를 설명합니다.

사용자 정의 단계

ContractNotebook.xml 파일을 수정하고 제거할 규정에 대한 패널 요소가 있는 절을 제거하십시오. 다음 디렉토리에서 해당 파일을 찾으십시오.

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\

제 2 장 고객 서비스 영업대표 도구

이 장에 설명된 요소는 WebSphere Commerce 액셀러레이터의 고객 서비스 영업대표 (CSR) 도구에 대한 사용자 인터페이스를 나타냅니다. 이는 CSR이 비즈니스와 연관된 당일의 태스크를 수행하기 위해 필요로 하는 고객, 주문 및 기타 조치 작성 및 유지보수를 용이하게 합니다. CSR 도구 구성요소에는 다음 요소가 포함됩니다.

- 고객 마법사
- 고객 노트북
- 주문 마법사
- 주문 노트북

사용자 정의 예

다음 예에서는 WebSphere Commerce 액셀러레이터 내에서 CSR 도구를 사용자 정의하는 방법에 대해 간략하게 보여줍니다.

시나리오 1: 비즈니스 고객 정보 요약 대화 상자에 추가 고객 데이터 추가

구현 개요

이 예에서는 비즈니스 고객의 사원 번호를 비즈니스 고객 정보 요약 대화 상자에 추가하는 방법에 대해 보여줍니다.

사용자 정의 단계

`ShopperSummaryB2BDialog.jsp`를 확장하십시오. 다음 디렉토리에서 해당 파일을 찾으십시오.

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\`

JSP 파일을 변경하여 정보 요약 대화 상자에 사원 번호를 표시하는 방법에 대해 아래의 코드 견본을 참조하십시오. 사원 번호는 `OptoolsRegisterDataBean`의 특성입니다. 견본은 레이블 `Employee Number`를 저장하기 위해 특성 파일을 사용하지 않았습니다. 사원 번호가 있는 견본 출력이 고객 정보 요약 대화 상자에 표시되었습니다.

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

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

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

<script>
<%@ include file = "SummaryDisplay.jsp" %>
:
:
: Some code is omitted here
:
:
</script>

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

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

<h1><%= userNLS.get("customerSummaryTitle") %></h1>
<P><B><%=userNLS.get("generalHeader")%></B>
<BR>
   <%=userNLS.get("logonid")%>
   <I><%=UIUtil.toHTML(registerDataBean.getLogonId()) %></I>
<BR>
   <%=userNLS.get("custName")%>:
   <I><script>displayNameSummary()</script></I>
<BR>
   Employee Number:
```

```
   <I><%=UIUtil.toHTML(registerDataBean.getEmployeeId()) %></I>
<BR>
   <%=userNLS.get("orgAccount")%>:
<% if (accountDBean != null) { %>
   <I><%=UIUtil.toHTML(accountDBean.getAccountName()) %></I>
<% } %>
<BR>
   <%=userNLS.get("challengeQuestion")%>:
   <I><%=UIUtil.toHTML(registerDataBean.getChallengeQuestion())%></I>
<BR>
   <%=userNLS.get("challengeAnswer")%>:
   <I><%=UIUtil.toJavaScript(registerDataBean.getChallengeAnswer())%></I>
<BR>
   <%=userNLS.get("clientCertificate")%>:
   <I>
<% if (certStatus != "") { %>
   <% if (certStatus == "V") {%><%=userNLS.get("valid")%><%}%>
   <% if (certStatus == "E") {%><%=userNLS.get("expired")%><%}%>
   <% if (certStatus == "R") {%><%=userNLS.get("revoked")%><%}%>
<% } else { %>
   <%=userNLS.get("noCertificate")%>
<% } %>
   </I>
<BR>
   <%=userNLS.get("status")%>:
   <I>
<% if (userRegistry.getStatus().equals("1")) {%>
   <%=userNLS.get("accountStatusEnabled")%>
<% } else { %>
   <%=userNLS.get("accountStatusDisabled")%>
<% } %>
   </I>
<BR>
<P><B><%=userNLS.get("contactHeader")%></B>
   <TABLE border="0" CELLPADDING=0 CELLSPACING=0 >
      <TR valign="top">
          <TD><%=userNLS.get("address")%>:</TD>
          <TD><I><script>displayAddrSummary(0)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(1)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(2)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(3)</script></I></TD>
      </TR>
      <TR valign="top">
         <TD></TD>
         <TD><I><script>displayAddrSummary(4)</script></I></TD>
      </TR>
   </TABLE>
<%=userNLS.get("phone")%>:
   <I><%=UIUtil.toHTML(address.getPhone1())%></I>
<BR>
```

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

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

%>

</BODY>
</HTML>
```

시나리오 2: 고객 서비스에서 ShopCart 사용 - 주문 관리 인터페이스

구현 개요

상점 디자이너가 CSR 도구를 사용하여 상태 P에서 구매자 주문을 관리하려면 **CSROrderSearchB2B** 및 **CSROrderSearchB2C** 보기를 수정해야 합니다.

사용자 정의 단계

CSROrderSearchB2B.jsp 및 CSROrderSearchB2B.jsp JSP 파일을 확장하십시오. 주문 상태 기준으로 Pending 옵션을 추가하면, CSR이 고객의 보류 중 주문을 검색하여 관리할 수 있게 됩니다. 보류 중 주문은 장바구니의 주문입니다. 이 파일 둘 다 다음

디렉토리에 있습니다.

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\

다음 코드 견본을 참조하십시오.

```
<%--
//*************************************************************************
//*-------------------------------------------------------------------------
//* 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>
```

제 3 장 협업 작업 영역

이 장에 설명된 요소는 QuickPlace를 사용자 정의하는 데 필요한 단계를 자세히 보여주는 사례 학습을 제시하고, WebSphere Commerce와 함께 제공되는 ToolTech 템플리트를 작성합니다. 또한 사용자가 협업 작업 영역을 결합하기 위해 방문할 때 보낸 전자 우편의 컨텐츠를 사용자 정의하는 데 필요한 단계에 대해서도 간략하게 설명합니다.

사용자 정의 예

다음 예는 WebSphere Commerce 액셀러레이터의 부분을 사용자 정의하는 방법에 대해 간략하게 보여줍니다.

주: 시나리오 1 및 2는 거의 관련되어 있습니다. 시나리오 1을 수행할 경우에 종종 시나리오 2도 수행할 수 있습니다.

시나리오 1: 작업 영역 룩앤필 사용자 정의

구현 개요

협업 작업 영역 기능에 대해 사용자 정의할 첫 번째 사항은 작성된 협업 작업 영역의 "룩앤필"입니다. 룩앤필을 사용자 정의하려면 PlaceType 템플리트를 작성해야 합니다. 사용자 정의에 대한 자세한 내용은 *Customizing QuickPlace* 레드북을 참조하십시오.

사용자 정의 단계

이 예에서는 WebSphere Commerce 5.4와 함께 제공되는 ToolTech 견본을 사용합니다. 이 예는 QuickPlace 작성 방법을 보여줍니다.

1. 다음을 수행하여 기본 QuickPlace를 작성하십시오.
   a. QuickPlace 서버가 실행 중인지 확인하십시오. `http://qp_server/quickplace`에 대해 브라우저를 열고 QuickPlace 운영자로 로그온하십시오.
   b. **QuickPlace 작성**을 누르십시오.
   c. **팀을 위한 표준 QuickPlace**를 선택하십시오. 적절하게 필드를 채운 후 **다음**을 누르십시오.
   d. 작성자로 로그온하십시오.
2. QuickPlace에 대한 환영 페이지를 갱신하십시오. **편집**을 누르십시오. 원하는 변경사항을 수행한 후 **공개**를 누르십시오. 환영 페이지가 갱신됩니다.
3. 다음을 수행하여 QuickPlace에 대한 로고를 변경하십시오.
   a. 목차에서 **사용자 정의**를 누르십시오.
   b. **기본**을 누르십시오.

c. **기본 변경**을 누르십시오.

d. **단순 텍스트, 로고 메이커** 또는 **로고 아트워크 업로드**를 사용하여 로고를 설정하십시오. ToolTech 견본은 **로고 아트워크 업로드**를 사용하였습니다. 이제 로고가 갱신됩니다.

시나리오 2: 사용자 정의 주제 작성

구현 개요

주제는 QuickPlace의 "룩앤필"을 제어합니다. 이 예는 QuickPlace의 레이아웃과 스타일에 대한 HTML 페이지와 계단식 스타일시트(CSS)를 작성합니다. 해당되는 파일 작성 방법에 대한 추가 정보는 *Customizing QuickPlace* 레드북을 참조하십시오.

주:

구성원 관리는 WebSphere Commerce를 통해 관리되므로, 목차에서 **구성원** 링크를 제거하십시오. 링크를 제거하려면 목차를 삽입할 때 페이지 레이아웃 파일에서 다음 코드를 사용하십시오.

```
<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=}>
```

사용자 정의 단계

1. 목차에서 **사용자 정의**, **사용자 정의 주제**를, **새 테마**를 누르십시오. 제목을 채운 후 **스타일 시트**와 **페이지 레이아웃** 파일을 업로드하십시오.

2. **다음**을 누르십시오.
3. 목차에서 **사용자 정의**를 누르고 **장식**을 누르십시오.
4. **주제 선택**을 누르십시오.
5. 방금 작성한 주제를 선택하고 **다음**을 누르십시오. 그러면 QuickPlace의 룩앤필이 갱신되어야 합니다.
6. 목차에서 **구성원** 링크가 사라진 점에 주의하십시오.
7. QuickPlace에서 PlaceType 작성. 방금 사용자 정의한 QuickPlace를 기초로 PlaceType을 작성할 수 있습니다. 목차에서 **사용자 정의**를 누르고 **PlaceType 옵션**을 누르십시오.
8. **편집**을 누르십시오. **이 QuickPlace에서 PlaceTypes 작성을 허용하시겠습니까?**에 대해 **예**를 선택하고 **PlaceType에서 작성되는 차후 QuickPlace에 이 QuickPlace의 현재 구성원 포함?**에 대해 **아니오**를 선택하십시오.
9. **다음**을 누르십시오.
10. `http://qp_server/quickplace`에 대해 브라우저를 열고 QuickPlace 운영자로서 로그온하십시오.
11. **PlaceTypes**를 누르십시오.
12. **PLACETYPE 작성...**을 누르십시오. PlaceType의 이름을 입력하고 기초로 할 QuickPlace를 선택하십시오.
13. **다음**을 누르십시오. 새 PlaceType이 표시되어야 합니다.

시나리오 3: 전자 우편 알림 사용자 정의

구현 개요

기본 전자 우편 알림을 모든 상점에 대한 사용자 정의 페이지로 변경하려면, 원하는 컨텐츠를 포함하도록 `CollabEmailContent.jsp` 파일을 수정해야 합니다.

사용자 정의 단계

1. `CollabEmailContent.jsp` 파일을 수정하십시오. 다음 디렉토리에서 파일을 찾으십시오.

   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. 다음을 수행하여 특정 상점에 맞게 전자 우편을 사용자 정의하십시오.
   a. DB2 명령창을 여십시오.
   b. WebSphere Commerce 데이터베이스에 연결하십시오. 기본 데이터베이스에 연결하려면 다음 명령을 사용하십시오.

      ```
      db2 connect to mall
      ```

   c. 다음 DB2 명령을 실행하십시오.

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

   d. 사용자 정의된 템플리트 파일 CollabEmailContent.jsp를 상점 디렉토리에 놓으십시오.

      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*

      여기서 *store_name*은 상점을 공개할 때 선택한 상점 디렉토리 이름입니다.
   e. WebSphere 관리 서버를 다시 시작하십시오.

제 4 장 고객 프로파일

이 장에 설명된 요소는 WebSphere Commerce 액셀러레이터의 고객 프로파일 구성요소에 대한 사용자 인터페이스를 나타냅니다. 이는 판매자가 비즈니스와 연관되는 당일의 태스크를 수행하기 위해 필요로 하는 모든 조치와 고객 프로파일의 작성 및 유지보수를 용이하게 합니다. 고객 프로파일 구성요소에는 다음 요소들이 포함됩니다.

- 고객 프로파일 마법사
- 고객 프로파일 노트북

견본 코드

이 절에서는 다음 URL에 있는 zip 파일에 포함된 코드 견본을 참조합니다.

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

이 장에서 일부 절을 완료하려면, WebSphere Application Server 4를 실행 중인 개발 시스템에 견본 코드를 다운로드하여 설치해야 합니다.

다운로드되고 나면, 이 ZIP 파일을 다음 디렉토리로 압축 해제하십시오.

AIX `/usr/WebSphere/CommerceServer`

400 `/QIBM/UserData/WebSphere/CommerceServer`

Linux `/opt/WebSphere/CommerceServer`

Solaris `/opt/WebSphere/CommerceServer`

Windows *drive*`:\WebSphere\CommerceServer`

사용자 정의 예

다음 예는 WebSphere Commerce 액셀러레이터의 해당 부분을 사용자 정의하는 방법에 대해 간략하게 보여줍니다.

시나리오 1: 고객 프로파일에 속성 추가

구현 개요

이 예에서는 USERS 테이블의 FIELD1 열에 사용자의 음료 환경설정을 저장합니다. 다음 값이 유효합니다.

- 맥주
- 와인
- 커피

- 차
- 물

이 예에서는 각각의 단계를 보여줍니다. 견본 코드를 설치할 때 작성된 profcust 디렉토리에서 이 예를 찾으십시오.

사용자 정의 단계

1. 새 JSP 파일을 작성하여 고객 프로파일 노트북에 새 페이지를 추가하십시오. 이 예에서 이 페이지는 두 옵션을 표시합니다.
   - 음료 환경설정 무시
   - 다음 음료 환경설정 중 하나 이상을 가지고 있는 대상 고객

   환경설정은 두 번째 옵션에서 선택란으로 표시됩니다. 자세한 내용은 다음 디렉토리에 동반되는 JSP 페이지 BeveragePanel.jsp를 살펴보십시오.

   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 파일을 작성하고 나면 이를 VIEWREG 테이블에 추가해야 합니다. 자세한 내용은 다음 디렉토리에 동반되는 SQL 스크립트 beverage.sql을 살펴보십시오.

   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. 다음 디렉토리에 있는 고객 프로파일 노트북이 새 페이지를 인식합니다.

   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.xml의 사본을 만드십시오. 견본에서 이 새 파일의 이름은 MySegmentNotebook.xml입니다. 다음 요소가 문서에 추가되었습니다.

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

다음 디렉토리에 있는 세그먼트화 특성 파일에 새 패널 이름이 있어야 합니다.

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\

이 패널이 작동 하려면 다음 행을 Resources.properties에 추가해야 합니다.

```
segmentNotebookBeveragePanel=Preferred beverage
```

선택적으로 MySegmentNotebook.xml을 다음에서

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

다음으로 복사하십시오.

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. 이제는 새 XML 문서가 인식되는지 확인해야 합니다. 이를 수행하려면 다음 디렉토리에서 Resources.xml 파일을 수정하십시오.

   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\

   다음 요소를 수정하십시오.

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

   요소를 다음으로 변경하십시오.

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

   파일을 MyResources.xml로 저장하십시오.

   선택적으로 MyResources.xml을 다음에서

   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

다음으로 복사하십시오.

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. 응용프로그램이 MyResources.xml을 지시하도록 하십시오. 다음 디렉토리에서 demo.xml 파일을 수정하십시오.

   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\

   다음을 찾으십시오.

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

   요소를 다음으로 변경하십시오.

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

6. 고객 프로파일 com.ibm.commerce.tools.segmentation.SegmentNotebookDataBean을 설명하는 데이터 bean을 확장하십시오. 견본 com.mycompany.tools.segmentation.MySegmentNotebookDataBean에서 데이터 bean을 확장하여 음료 속성에 대한 특성을 제공합니다. 다음 디렉토리에서 견본 MySegmentNotebookDataBean.java를 찾으십시오.

   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. 세그먼트 노트북이 새 데이터 bean을 인식하도록 하려면, 고객 프로파일 노트북을 설명하는 XML을 수정해야 합니다. MySegmentNotebook.xml로 리턴하여 다음 변경사항이

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

다음으로 변경된 점에 주목하십시오.

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

8. 고객 프로파일을 저장하는 제어기 명령을 확장하십시오. 이 명령은 com.ibm.commerce.tools.segmentation.SegmentSaveControllerCmd입니다. 다음 디렉토리에 있는 MySegmentSaveControllerCmdImpl에서 견본 명령 구현을 살펴보십시오.

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\

이 예는 음료 조건에 대한 단순 조건을 구성합니다.

응용프로그램이 새 명령을 인식하도록 하려면, 이를 CMDREG 테이블에 추가해야 합니다. 자세한 정보는 위의 1a단계에서 beverage.sql을 참조하십시오.

9. com.ibm.commerce.membergroup.commands.CheckUserInMemberGroupCmd 새 조건 평가 방법에 대한 자세한 내용은 다음 디렉토리의 견본 명령을 평가하는 태스크 명령을 확장하십시오. 다음 디렉토리에서, 새 조건을 평가할 방법에 대한 세부사항에 대해 견본 명령 MyCheckUserInMemberGroupCmdImpl.java를 살펴보십시오.

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\

또한 CMDREG 테이블에 명령을 등록해야 합니다. 자세한 정보는 beverage.sql을 참조하십시오.

10. 고객 프로파일에 대한 제한자 목록을 표시하는 태스크 명령을 확장하십시오. 이 명령은 com.ibm.commerce.tools.segmentation.SegmentConstraintListCmd입니다. 반드시 변경해야 하는 사항에 대한 자세한 내용은 다음 디렉토리의 견본 명령 MySegmentConstraintListCmdImpl을 살펴보십시오.

    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\

    또한 CMDREG 테이블에 명령을 등록해야 합니다. 자세한 정보는 beverage.sql을 참조하십시오.

11. 다음 클래스를 추가하십시오.

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

    위의 클래스는 다음 디렉토리에 있습니다.

    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

위의 클래스를 다음 디렉토리에 있는 wcsmcruntime.jar 파일에 추가하십시오.

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. 1단계의 BeveragePanel.jsp 파일을 다음 디렉토리로 복사하십시오.

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. 이러한 변경사항을 수행하려면 이 책의 범위를 벗어나는 액세스 제어 설정도 변경해야 합니다. 다음 URL에서 볼 수 있는 *WebSphere Commerce 액세스 제어 안내서*를 참조하십시오.

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

제 5 장 캠페인

데이터베이스에서 직접 규칙 입력

사용자 인터페이스가 규칙 데이터를 데이터베이스에 입력할 수 있는 가장 쉬운 방법이지만, 일부 상황에서는 INITIATIVE 테이블의 RULE 열에서 수동으로 규칙을 입력해야 할 수도 있습니다. 규칙은 XML을 사용하여 정의됩니다. 이러한 경우 아래에 표시된 DTD(Document Type Definition)에 따라야 합니다.

INITIATIVE 테이블의 RULE 열에 대한 DTD는 다음과 같습니다.

```
<!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 요소

행사의 기본 구현은 다음 simpleCondition 요소를 지원합니다. orListCondition, andListCondition 및 openCondition 요소에 이를 결합하여 규칙의 조건 부분을 작성할 수 있습니다.

표 2.

변수 이름	지원되는 연산자	지원되는 값	규정자
segment	= !=	고객 프로파일의 이름	없음
shoppingCartSku	= !=	상품 SKU	없음
shoppingCartCategory	= !=	상품 카테고리 이름	언어
purchaseHistorySku	= !=	상품 SKU	없음
purchaseHistoryCategory	= !=	상품 카테고리 이름	언어
shoppingCartTotal	= != > < >= <=	10진수 값	통화
dayOfWeek	= !=	SUNDAY MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY	없음

openCondition 요소

기본 구현은 다음 openCondition 요소를 지원합니다. orListCondition, andListCondition 및 simpleCondition 요소에 이를 결합하여 규칙의 조건 부분을 작성할 수 있습니다.

표 3.

열린 조건 이름	지원되는 매개변수
shoppingCartTotal	comparisonType - ″>″, ″<″ 또는 ″=″ 중 하나. 값 - 10진수 값 통화
shoppingCartCategory	comparisonType - ″=″ 또는 ″!=″ 중 하나. 값 - 카테고리 언어의 이름
shoppingCartSku	comparisonType - ″>″, ″<″ 또는 ″=″ 중 하나. 값 - 10진수 값 통화
purchaseHistoryCategory	comparisonType - ″=″ 또는 ″!=″ 중 하나. 값 - 카테고리 통화의 이름
purchaseHistorySku	comparisonType - ″>″, ″<″ 또는 ″=″ 중 하나. 값 - 10진수 값 통화

조치 요소

기본 구현은 다음 조치 요소를 지원합니다. 이 요소들은 규칙을 구성하기 위해 조건 요소와 같이 있을 수 있습니다.

표 4.

조치 이름	지원되는 매개변수	지원되는 값	지원되는 부속 매개변수
suggestiveSell	queryOperator	or and	없음
	queryType	skuQuery genericQuery	none
	sku	상품 SKU	none
	skuPrefix	상품 SKU 접두부	none
	category	상품 카테고리 이름	language
	productDescription	상품 카테고리 설명	language
	inventoryQuantity	정수	comparisonOperator
	listPrice	10진수 값	comparisonOperator currency
	availabilityDate	yyyy-mm-dd-00.00.00 양식의 날짜	comparisonOperator
collaborativeFiltering	없음		
awarenessAd	collateral	광고 이름	
discountAd	discountCollateral	광고 이름	Discount identifier
couponAd	couponCollateral	광고 이름	
categoryRecommendation	category	카테고리의 이름	

다음 행사는 남성에게는 봄 광고를, 여성에게는 가을 광고를 보여줍니다.

```
<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>
```

Infrastructure 요소

규칙 패키지

캠페인 구성요소는 com.ibm.commerce.rule 패키지를 사용할 수 있게 만듭니다. 이 패키지는 규칙 세트를 XML 형식으로 또는 XML 형식으로부터 변환하는 것을 도와주는 클래스를 제공합니다. 또한 규칙 세트에서 규칙을 호출할 수 있는 지원을 제공합니다.

표 5.

클래스/인터페이스 이름	설명
Action	Action 클래스는 규칙의 조치 부분에서 사용되는 이름과 매개변수에 대한 특성을 제공합니다. 매개변수는 이름과 값이 선택적으로 부속 매개변수도 있을 수 있습니다.
ActionHandler	규칙의 조치 부분 구현을 제공하기 위해 ActionHandler 인터페이스를 구현해야 합니다. 규칙의 조건 부분이 true로 평가될 때 호출되는 하나의 메소드 ″performAction″이 있습니다. Action 클래스의 적절한 인스턴스가 ″performAction″ 메소드 구현에 전달됩니다.
Rule	Rule 클래스는 규칙 인스턴스를 XML 형식으로 또는 XML 형식으로부터 변환하는 것을 도와주는 메소드를 제공합니다. Rule 클래스에는 세 가지의 특성이 있습니다. 규칙의 조건 부분에 대한 특성, 규칙의 조건 부분에 대한 특성 그리고 설명에 대한 특성입니다. 조건 특성은 com.ibm.commerce.condition 패키지에서 볼 수 있는 condition 클래스의 인스턴스여야 합니다. Rule 클래스에는 또한 Evaluator 인스턴스 및 ActionHandler 인터페이스의 구현을 승인하는 메소드 ″invoke″가 있습니다. 조건이 true로 평가되면 ActionHandler 구현의 ″performAction″ 메소드가 호출됩니다.
RuleConstants	RuleConstants 인터페이스에는 규칙 패키지에서 사용되는 일부 공용 상수가 있습니다.

표 5. (계속)

클래스/인터페이스 이름	설명
RuleSet	RuleSet 클래스는 RuleSet 인스턴스를 XML 형식으로 또는 XML 형식으로부터 변환하는 것을 도와주는 메소드를 제공합니다. RuleSet 클래스는 규칙 오브젝트의 배열 특성이 있습니다. 또한 Evaluator 및 ActionHandler 인터페이스의 구현을 승인하는 invoke 메소드가 있습니다. invoke 메소드가 호출될 때, 이 메소드는 Rule 오브젝트 배열을 통해 루핑하면서 하나의 Rule 오브젝트 다음에 다른 규칙 오브젝트를 호출합니다.

캠페인은 규칙 패키지를 사용하여 XML 형식으로 또는 XML 형식으로부터 캠페인 행사의 규칙 부분을 저장하고 로드합니다. 또한 규칙을 호출하는 것도 돕습니다. `CampaignInitiativeEvaluateCmd` 태스크 명령은 캠페인 행사 조치를 처리하기 위해 사용되는 `ActionHandler` 인터페이스 구현을 제공합니다.

조건 패키지

규칙 패키지는 `com.ibm.commerce.condition` 패키지를 사용할 수 있게 합니다. 이 패키지는 부울 조건을 XML 형식으로 또는 XML 형식으로부터 변환하는 것을 도와주는 클래스를 제공합니다. 또한 조건을 평가하기 위한 지원을 제공합니다.

표 6.

클래스/인터페이스 이름	설명
AndListCondition	AndListCondition 클래스는 ConditionList 클래스를 확장하여 AND 부울 연산자가 결합한 조건 목록으로 구성하는 조건을 제공합니다.
Condition	Condition 클래스는 XML 형식으로 또는 XML 형식으로부터 변환하는 것을 도와주는 메소드를 제공하는 추상 클래스입니다. 이 클래스는 조건을 평가하기 위해 사용할 수 있는 추상 메소드도 제공합니다.
ConditionConstants	ConditionConstants 인터페이스는 조건 패키지에서 사용되는 상수 값을 제공합니다.
ConditionList	ConditionList 클래스는 Condition 클래스를 확장하는 추상 클래스입니다. 이는 조건 목록을 구성하는 조건에 대한 지원을 제공합니다.
ConditionUtil	ConditionUtil 클래스는 조건 평가 동안 사용할 수 있는 정적 방법 수를 제공합니다.
Evaluator	단순 조건이나 열린 조건을 평가하려면 Evaluator 인터페이스를 구현해야 합니다.
OpenCondition	OpenCondition 클래스는 Condition 클래스를 확장하여 일반 조건을 제공합니다. 이 조건은 이름 값 쌍 목록으로 구성됩니다.
OrListCondition	OrListCondition 클래스는 ConditionList 클래스를 확장하여 OR 부울 연산자가 결합한 조건 목록으로 구성하는 조건을 제공합니다.

표 6. (계속)

클래스/인터페이스 이름	설명
SimpleCondition	SimpleCondition 클래스는 Condition 클래스를 확장하여 변수, 연산자, 값 양식의 조건을 제공합니다.
TrueCondition	TrueCondition 클래스는 Condition 클래스를 확장하여 항상 true로 평가되는 조건을 제공합니다.

규칙 패키지는 조건 패키지를 사용하여 XML 형식으로 또는 XML 형식으로부터 규칙의 조건 부분을 저장하고 로드합니다. 또한 조건 평가를 돕습니다.

추상 Condition 클래스에 대해 5가지의 실제 확장이 있습니다. 이 중에서 세 가지(AndListCondition, OrListCondition, TrueCondition)는 자체 평가 방법을 알고 있습니다. 나머지 두 가지(OpenCondition 및 SimpleCondition)는 평가받기 위해 Evaluator 인터페이스의 구현이 필요합니다. CampaignInitiativeEvaluateCmd 태스크 명령은 캠페인 행사 조건을 평가하기 위해 사용되는 Evaluator 인터페이스의 구현을 제공합니다.

Blaze 규칙 프로젝트

캠페인 행사의 기본 구현은 캠페인 행사 규칙을 평가하도록 도와주는 Blaze 규칙 프로젝트를 사용합니다. 기본적으로 모든 열린 조건은 **CampaignInitiativeEvaluateCmd** 태스크 명령이 Blaze 규칙 프로젝트를 호출하게 합니다. 다음 디렉토리에서 프로젝트 `CampaignInitiativeEvaluator`를 찾으십시오.

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`

이 프로젝트의 기본 구현은 캠페인 행사 사용자 인터페이스가 작성한 모든 열린 조건을 평가할 수 있습니다. 이 프로젝트에 추가 열린 조건에 대한 지원을 추가할 수 있습니다.

사용자 정의

고객은 캠페인 행사의 기본 구현을 확장하여 지원되는 추가 조건과 가능할 경우 추가 조치 및 표시 유형을 추가하려고 할 것입니다.

새 행사 조건에 대한 런타임 지원 추가

캠페인 행사 조건을 평가하는 blaze 프로젝트는 **CampaignInitiativeContext** 인스턴스를 승인합니다. 이 컨텍스트는 명령 컨텍스트에 대한 액세스와 열린 조건의 이름 및 매개변수를 제공합니다. 새 열린 조건에 대한 지원을 추가하려면, 새 규칙을 `conditionEvaluator` 규칙에 추가해야 합니다. 이 규칙은 새 조건의 이름을 확인하고 조건을 평가해야 합니다. 조건 평가 중, 컨텍스트 오브젝트를 통해 이름별로 열린 조건 매개변수에 액세스할 수 있습니다.

새 행사 조건에 대한 빌드 시간 지원 추가

마케팅 관리자가 새 조건을 이용하려면 캠페인 행사 사용자 인터페이스 새 조건에 대한 지원을 제공하도록 확장되어야 합니다. 이는 캠페인 행사 마법사나 노트북(WhenAddPanel.jsp)의 패널에 새 조건을 추가하여 수행할 수 있습니다. 또한 **CampaignInitiativeSaveControllerCmdImpl** 클래스를 확장하여 새 조건 오브젝트를 구성해야 합니다. 다음 디렉토리에서 파일을 찾으십시오.

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\

새 캠페인 행사 조치에 대한 런타임 지원 추가

새 캠페인 행사 조치에 대한 지원을 제공하려면, **CampaignInitiativeEvaluateCmdImpl** 클래스를 확장하여 **performAction** 메소드를 대체해야 합니다. 조치의 이름이 새 조치 이름과 일치할 경우 조치를 수행해야 하고, 그렇지 않으면 기본 구현을 수행하기 위해 super 메소드를 호출해야 합니다.

새 캠페인 행사 조치에 대한 빌드 시간 지원 추가

마케팅 관리자가 새 캠페인 행사 조치를 이용하도록 하려면, 캠페인 행사 마법사나 노트북(InitiativeWhatPanel.jsp)의 항목 패널을 갱신해야 합니다. 또한 **CampaignInitiativeSaveControllerCmdImpl** 클래스를 확장하여 새 조치 오브젝트를 구성해야 합니다. 다음 디렉토리에서 파일을 찾으십시오.

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\

새 e-Marketing 지점 유형에 대한 지원 추가

추가 출력 유형을 제공해야 할 경우, EMarketingSpot 클래스를 확장하여 페이지 디자이너가 사용할 수 있는 새 특성을 제공합니다. 또한 **CampaignInitiativeEvaluateCmdImpl** 클래스를 확장하여 새 출력 유형을 리턴해야 합니다.

제 6 장 카탈로그 검색

이 장에 설명된 요소는 기존 검색 기능을 확장하기 위해 필요한 활동에 대해 설명합니다. 추가 기능에는 검색 구성요소에 의해 액세스 가능한 새 속성 및 테이블을 도입하기 위한 지시사항이 포함됩니다. 스키마를 최적화하여 런타임에 대한 비용을 줄일 수 있도록 하면 검색 구성요소의 성능도 개선됩니다. 이 구성요소에는 다음과 같은 사용자 정의 가능한 요소가 포함됩니다.

- 검색 bean
- 검색 엔진
- 정보 요약 테이블 정의 스크립트

표 8은 카탈로그 검색 bean에 의해 검색되는 데이터베이스 열 목록을 요약한 것입니다. 다양한 사이트 요구사항으로 인해, 이 표에 설명되지 않은 검색을 열에 제공해야 한다는 것을 알게 될 것입니다.

사용자 정의 시나리오

검색 엔진에서 이러한 속성이 이미 사용 가능한 검색 bean으로 검색 속성을 추가하려면, 속성을 검색 bean에 추가하여 시나리오 1을 구현하십시오. 검색 엔진이 속성에 대해 검색을 지원하지 않지만 이미 검색에서 사용하는 테이블 중 하나에서 사용 가능한 경우, 시나리오 2를 구현하여 속성을 엔진에 추가하십시오. 검색 엔진이 검색하는 테이블 중 하나에 이 속성이 없을 경우, 시나리오 3을 구현하여 테이블을 검색 엔진에 추가하십시오. 검색 bean에 충분한 속성을 추가하기 위한 프로시저는 다른 속성을 검색 bean에 추가하는 것(시나리오 1 구현)과 같지만 구현에서 약간 다릅니다. 시나리오 4를 참조하십시오. 런타임 성능을 개선하려면, 시나리오 5를 구현하여 각 카테고리마다 하나의 테이블을 작성함으로써 검색되는 행 수를 줄여서 정보 요약 테이블 정의 스크립트를 수정 및 최적화하십시오. 다음 시나리오는 구성요소를 사용자 정의할 시기를 요약한 것입니다.

시나리오 1: 검색 bean에 속성 추가

검색 엔진에서 이미 사용 가능한 bean에 새 속성을 도입합니다.

시나리오 2: 검색 엔진에 속성 추가

테이블을 사용하여 이미 검색한 검색 엔진에 새 속성을 도입합니다.

시나리오 3: 검색 엔진에 테이블 추가

검색 엔진에 새 테이블을 도입합니다.

시나리오 4: 검색 bean에 충분한 속성 추가

Bean에 충분한 속성을 도입합니다.

시나리오 5: 최적화된 정보 요약 테이블로 성능 개선

데이터 세트의 지식을 사용하여 정보 요약 테이블 정의를 최적화합니다.

주: 이 책에서 검색 엔진, 검색 인터페이스 및 단일화된 검색 프레임워크에 대한 설명은 동일한 것이므로 서로 바꿔서 사용됩니다.

시나리오 1: 검색 bean에 속성 추가

구현 개요

사용자 정의할 수 있는 첫 번째 사항은 다른 검색 가능한 속성을 추가하는 것입니다. 이 활동에서는 어느 곳에도 해당 속성을 사용할 수 없는 경우에(시나리오 2에 설명) 검색 bean을 변경하고 선택적으로 검색 엔진을 변경해야 합니다. 검색 bean의 사용자 정의는 검색 bean을 서브클래스화하고 기존 메소드를 확장해야 합니다. 이를 수행하는 방법에 대한 견본 코드는 `samples/search` 디렉토리에 제공됩니다.

사용자 정의 단계

이 예는 추가 검색 속성을 `com.ibm.search.catalog` 패키지에 있는 메타데이터 클래스에 카탈로그 검색 bean을 추가하는 방법에 대해 보여줍니다. 카탈로그 검색 데이터 bean은 클래스입니다. 이를 사용자 정의하려면, 다음을 수행하십시오.

1. `CatEntrySearchListDataBean` 클래스의 서브클래스를 작성하십시오.
2. 추가할 새 검색 가능한 속성에 대해 변수를 선언하십시오. 두 가지 유형의 변수를 선언하여 검색 가능한 속성과 연관되는 정보를 저장할 수 있습니다. 첫 번째 유형은 속성에 대한 값을 저장하는 데 사용할 수 있습니다. 두 번째 유형은 검색 연산자, 부울 검색, 대소문자 구분과 같은 검색에서 사용할 수 있는 관련 정보가 있을 수 있습니다. 이 변수에 저장되는 정보는 카탈로그 정보를 조회하는 SQL 조회 생성 시 사용될 수 있습니다.

   주: SQL 조회 생성에서 사용되는 데이터베이스 열은 생성된 조회에서 제한자로 사용할 수 있기 전에 검색 인터페이스 RuleQuery 클래스나 서브클래스 중 하나에서 이전에 정의되어야 합니다.
3. 선언된 변수에 대해 액세서(get 및 set 메소드)를 작성하십시오.
4. 다음을 수행하는 `populate()` 메소드를 작성하십시오.
   a. `RuleQuery`나 `RuleQuery` 서브클래스의 인스턴스를 생성하십시오.
   b. `setRuleQuery(ruleQueryInstance)` 메소드를 사용하여 상위 bean 클래스의 인스턴스에 대해 `RuleQuery`의 인스턴스를 참조하십시오.
   c. `super.setPredefinedAttributes()` 메소드를 호출하십시오.
   d. 검색 인터페이스를 사용하는 검색 가능한 새 제한자에 대해 `<Predicate>`를 공식화하십시오.
   e. `super.setIsAllNull(false)`을 사용하여 새 검색 속성이 추가되었음을 상위 요소에 알리십시오.

f. super.execute() 메소드를 호출하십시오.

예

이 예는 com.ibm.search.catalog 패키지에 메타데이터 클래스가 있다고 가정하고 검색 가능한 새 속성 onSpecial을 카탈로그 검색 bean에 추가하는 방법에 대해 보여줍니다.

1. 새 서브클래스를 작성하십시오.
2. 검색 가능한 속성에 대해 새 변수를 작성하십시오.

```
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 변수는 검색 가능한 속성값을 저장하는 데 사용됩니다.

onSpecialOperator 변수는 검색 연산자(예: like, =, <, >, <=, >=)를 저장하는 데 사용됩니다.

3. 모든 변수에 대해 액세서를 작성하십시오.

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

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

4. populate() 메소드를 작성하십시오.

```
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
}
```

시나리오 2: 검색 엔진에 속성 추가

구현 개요

검색 엔진은 카탈로그에서 결과를 검색하기 위해 조회를 적절하게 공식화하기 위한 속성 및 테이블에 대한 정보가 필요합니다. 이 정보는 메타데이터로 보관됩니다. 검색 엔

진을 확장하여 검색 가능한 새 속성을 추가하려면, 추가 속성 메타데이터와 선택적으로 메타데이터 정의가 필요합니다. 테이블 메타데이터는 새 테이블을 검색 엔진에 도입할 경우에만 필요합니다. 추가 세부사항은 시나리오 3에 제공되어 있습니다. 이 시나리오가 완료되면, 시나리오 1(검색 Bean에 속성 추가)을 구현하여 JSP 페이지 디자이너에 대해 이를 사용 가능하게 만들어야 합니다. 이를 수행하는 방법에 대한 견본 코드는 samples/search 디렉토리에 있습니다.

사용자 정의 단계

검색 엔진은 단일화된 검색 프레임워크의 일부입니다. 이는 RuleQuery 클래스와 각각 열 및 테이블 이름을 설명하는 AttributeInfo 및 TableInfo와 같은 추가 메타데이터 클래스로 표시됩니다. 이 예는 com.ibm.search.catalog 패키지에 없는(그러나 테이블에 있는 열의 메타데이터는 존재함) 데이터베이스 열에 대한 메타데이터 클래스를 작성하여 검색 엔진에 새 속성을 추가하는 방법에 대해 보여줍니다. 검색 엔진을 사용자 정의하려면, 다음을 수행하십시오.

1. 검색 가능하게 각각의 열 및 테이블에 대해 검색 인터페이스 메타데이터를 정의하십시오. 이 때 다음 사항이 요구됩니다.
   a. 각 검색 가능한 테이블에는 TableInfo 클래스의 서브클래스인 해당되는 클래스가 있어야 합니다. 이 서브클래스는 테이블 이름을 지정해야 합니다.
   b. 각 검색 가능한 테이블에는 AttributeInfo 클래스의 서브클래스인 해당되는 클래스가 있어야 합니다. 이 서브클래스는 열 이름, TableInfo 서브클래스의 열 및 열의 SQL 데이터 유형을 지정해야 합니다.
2. RuleQuery 클래스의 서브클래스를 작성하고, 각각의 새 데이터베이스 열에 대한 정적 정수 참조를 정의하십시오. 정수 값 0 - 1000은 시스템용으로 예약되어 있습니다.
3. findAttributeInfoName() 메소드를 작성하십시오. 이 메소드가 상위 findAttributeInfoName() 메소드를 대체하지 않으려면 super.findAttributeInfoName() 메소드를 호출하십시오. 상위 findAttributeInfoName()이 대체될 경우, 상위 클래스에 정의된 데이터베이스 열을 사용할 수 없습니다.
4. 이 메소드에 팩토리 클래스 작성 로직을 추가하여 새로운 각 데이터베이스 열에서 AttributeInfo 메타데이터 클래스를 작성하십시오.
5. 검색 가능한 새 테이블에 대해 사전 정의된 테이블 결합 관계를 추가하여 search.xml 파일을 수정하십시오. 모든 테이블 조합에 대해 결합 관계가 필요합니다. search.xml을 수정하려면 다음을 수행하십시오.
   a. COMMENT 섹션을 추가하여 항목에 의해 공식화된 테이블 결합 관계를 설명하십시오.
   b. KEY 섹션을 추가하여 항목을 고유하게 식별하십시오. 결합된 테이블은 고유한 키로 사용됩니다.

c. VALUE 섹션을 추가하여 키와 연관되는 값을 지정하십시오. 값 섹션의 항목은 결합 관계를 정의합니다. 값 섹션의 열 이름은 해당되는 열의 AttributeInfo 서브클래스를 사용하여 정의됩니다.

예

다음 예는 검색 가능한 새 데이터베이스 열 CATENTRY.ONSPECIAL을 작성합니다.

1.

a. TableInfo 클래스의 서브클래스를 작성하여 CATENTRY 테이블을 지정하십시오 (테이블에 대한 클래스가 있지 않을 경우).

```
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 클래스의 서브클래스를 작성하여 열 이름 ONSPECIAL을 지정하십시오.

```
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. 검색 인터페이스 RuleQuery의 서브클래스를 작성하십시오.

```
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() 메소드를 작성하십시오.

```
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. 필요할 경우 Search.xml을 수정하십시오(이 예에서는 search.xml 파일을 수정하지 않아도 됩니다).

주: 새 메타데이터를 검색 bean에 추가하려면 시나리오 1에 있는 단계를 따르십시오. 그러나 RuleQuery 클래스의 인스턴스 대신 메타데이터 정보가 있는 RuleQuery 하위 조회의 인스턴스를 사용해야 합니다(위의 예에서 CustomQuery 참조).

시나리오 3: 검색 엔진에 테이블 추가

구현 개요

현재 검색 엔진에서 지원되지 않는 새 테이블에 속성을 추가하려면, 테이블에 대한 정보를 정의하고 이 테이블이 어떻게 엔진에서 사용되는 다른 테이블과 연관되는지 설명해야 합니다. 이 정보는 각 테이블을 다른 테이블에 관련짓는 결합 술어를 유도하기 위해 사용되는 결합 관계이며, 메타데이터로 보관됩니다. 검색 엔진을 확장하여 검색 가능한 새 테이블을 추가하려면, 각 테이블과 다른 테이블과의 관계를 정의해야 합니다.

이 테이블과 관련되는 속성을 검색 bean 작성자가 사용할 수 있도록 하려면 시나리오 2 "검색 엔진에 속성 추가"를 구현해야 합니다. 이를 수행하는 방법에 대한 견본 코드는 samples/search 디렉토리에 있습니다.

사용자 정의 단계

이 예는 데이터베이스 테이블에 대한 결합 관계 메타데이터 클래스를 작성하는 방법에 대해 보여줍니다. 일반적으로 모든 로직 테이블 관계 조합에 대한 하나의 결합 관계를 정의해야 합니다.

예

이 예는 기존 search.xml은 CATENTRY 테이블과 OFFERPRICE 테이블 사이의 새 결합 관계 항목을 포함하도록 변경합니다.

1. COMMENT 섹션을 추가하십시오.

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

2. KEY 및 VALUE 섹션을 추가하십시오.

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

시나리오 4: 검색 bean에 충분한 속성 추가

구현 개요

충분한 속성에 대한 검색을 Bean에 추가하기 위한 프로세스는 다른 검색 가능한 속성을 bean에 추가하는 것과 같습니다. 데이터 세트가 알려지고 충분한 속성이 식별된 경우, 시나리오 1에 따라 이 속성들이 검색 bean에서 사용할 수 있도록 만드십시오. 이 시나리오와 시나리오 1 사이의 유일한 차이점은 충분한 속성 술어를 구성하기 위해 사용되는 메소드 매개변수들에 있습니다(자세한 내용은 아래 예에서 굵게 표시된 텍스트를 참조하십시오). 검색 엔진은 ATTRIBUTE 및 ATTRVALUE 테이블에 정의된 충분한 속성에 대해 검색하기 위한 기능을 이미 가지고 있습니다.

사용자 정의 단계

사용자 정의 시나리오 1(검색 bean에 속성 추가)과 같습니다.

예

이 예는 검색 가능한 충분한 새 속성인 color를 카탈로그 검색 bean에 추가하는 방법에 대해 보여줍니다.

1. 새 서브클래스를 작성하십시오.
2. 검색 가능한 속성에 대한 새 변수를 작성하십시오.

```
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;
  }
```

변수 colorValue는 검색 가능한 속성값을 저장합니다. 변수 colorValueOperator는 검색 연산자(예: like, =, <, >, <=, >=)를 저장하는 데 사용됩니다.

3. 모든 변수에 대해 액세서를 작성하십시오.

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

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

4. populate() 메소드를 작성하십시오.

```
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
}
```

시나리오 5: 최적화된 정보 요약 테이블로 성능 개선

구현 개요

데이터 세트가 정의될 때, 이 지식은 검색 성능을 탁월하게 개선하기 위해 더 최적의 정보 요약 테이블을 작성하는 데 사용할 수 있습니다. 발견될 것으로 예상되는 조회 유형과 비슷한 테이블을 작성하는 것이 목적입니다. 대부분의 상황에서 정보 요약 테이블은 훨씬 크기가 작으므로, 테이블에 적용되는 조회가 기반이 되는 더 큰 원시 테이블이나 다른 덜 최적화된 정보 요약 테이블을 사용하는 것보다 더 빠릅니다.

사용자 정의 단계

이 예는 각 카테고리 그룹에 대해 충분한 속성 카테고리 그룹 정보 요약 테이블을 작성하는 방법을 보여줍니다. 풍부한 속성의 항목을 가지고 있는 많은 카테고리 그룹(CATGROUPS)이 있을 경우, 각 그룹에 대해 정보 요약 테이블 정의를 정의하십시오.

정보 요약 테이블을 작성하기 위한 단계는 다음과 같습니다.

1. 정보 요약 테이블 정의로 사용될 select 조회를 사용하여 테이블 작성 명령문을 정의하십시오. select문에는 결과 세트에 나열된 대로 ″group by″ 절에 지정된 것과 같은 열이 있어야 합니다. 자세한 정보는 ″자동 요약 테이블″(AST)에 대한 DB2 문서나 ″실제화된 보기″에 대한 Oracle 문서를 참조하십시오.
2. select문에서 카테고리에 대한 카테고리 그룹 술어를 지정하십시오.
3. 같은 결과 세트 열을 사용하여 위의 조회에 지정된 것과 같은 순서로 색인을 작성하십시오.
4. 각 카테고리 그룹에 대해 위의 단계를 반복하십시오.

예

이 예는 세 가지의 카테고리 그룹 10000, 10001, 10002 각각에 대해 카테고리 그룹에 있는 충분한 속성의 정보 요약 테이블 작성 스크립트를 보여줍니다. 굵은체 부분은 이 예가 기초로 하고 있는 wcs.summary.richAttributeCatGroup.create.sql 파일에 표시된 일반 정의에 새로 비교됩니다.

```
// 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;
```

카탈로그 검색 데이터 bean 변수

URL(JSP)을 통해 카탈로그 검색 bean으로 전달해야 하는 변수 목록

표 7.

이름	데이터 유형	설명
beginIndex	string	이 변수는 결과 세트를 페이징하는 데 사용됩니다. 값은 페이지에서 첫 번째 결과 행의 색인이어야 합니다.
catGroupId	string	이 변수의 값은 카테고리 ID에 대한 검색에 사용됩니다.
categoryTerm	string	이 변수의 값은 카테고리 이름이나 설명, 또는 둘 다에 대한 검색에서 사용됩니다.
categoryTermCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나, 대소문자를 구분하지 않는 검색에 대한 no 중 하나여야 합니다.
categoryTermOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
categoryTermScope	Integer	사용자는 검색 범위를 이름, 이름 및 간단한 설명 또는 이름, 간단한 설명 및 자세한 설명 중 하나로 제한할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 이름 및 간단한 설명에 대한 1, 이름에 대한 2 또는 이름, 간단한 설명 및 자세한 설명에 대한 3 중 하나여야 합니다.
categoryType	string	사용자는 세 가지 유형의 검색 기준(All,Any 또는 Exact Phrase)을 지정할 수 있습니다. 이 변수의 값은 사용자의 검색 기준을 저장하는 데 사용됩니다. 값은 모두 검색 기준에 대한 ALL, 임의 검색 기준에 대한 ANY 또는 정확히 일치 검색 기준에 대한 EXACT여야 합니다.

표 7. (계속)

currency	String	이 변수의 값은 통화에 대한 검색에서 사용됩니다.
currencyCaseSensitive	String	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.
currencyOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
filterTerm	string	이 변수의 값은 지정된 값에 대한 검색을 필터링하는 데 사용됩니다.
filterTermCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.
filterTermOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
filterType	string	사용자는 세 가지 유형의 검색 기준(All, Any 또는 Exact Phrase)을 지정할 수 있습니다. 이 변수의 값은 사용자의 검색 기준을 저장하는 데 사용됩니다. 값은 모두 검색 기준에 대한 ALL, 임의 검색 기준에 대한 ANY 또는 정확히 일치 검색 기준에 대한 EXACT 중 하나여야 여야 합니다.

표 7. (계속)

isListPriceOn	string	카탈로그 검색 bean 사용자는 원가 또는 표준가를 노출할 것을 선택할 수 있습니다. 원가를 노출하려면 이 변수를 yes로 설정하고, 표준가를 노출하려면 이 변수를 no로 설정하십시오. 기본적으로, 검색 bean은 표준가를 노출합니다.
manufacturer	string	이 변수의 값은 제조업체의 이름에 대한 검색에 사용됩니다.
manufacturerCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.
manufacturerOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
manufacturerPartNum	string	이 변수의 값은 제조업체의 부품 번호에 대한 검색에 사용됩니다.
manufacturerPartNumCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.
manufacturerPartNumOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
maxPrice/minPrice	string	이 변수의 값은 가격 범위에 대한 검색에 사용됩니다.
pageSize	string	이 변수의 값은 페이지마다 표시될 검색 결과 행 수를 지정합니다.
price	string	이 변수의 값은 가격에 대한 검색에 사용됩니다.

표 7. (계속)

priceOperator	string	사용자는 검색 연산자로 =, <, >, !=, <=, >= 연산자 중 하나를 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 EQUAL, NOTEQUAL, GREATER, LESS, GREATER_EQUAL, LESS_EQUAL 중 하나여야 합니다.
qtyAvailable	string	이 변수의 값은 상품 또는 항목의 재고에 대한 검색에 사용됩니다.
qtyAvailableOperator	string	사용자는 검색 연산자로 =, <, >, !=, <=, >= 연산자 중 하나를 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 EQUAL, NOTEQUAL, GREATER, LESS, GREATER_EQUAL, LESS_EQUAL 중 하나여야 합니다.
qtyMeasure	string	이 변수의 값은 수량 측정에 대한 검색에 사용됩니다.
qtyMeasureCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.
qtyMeasureOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
resultCount	string	이 변수는 검색에 대해 리턴된 총 결과 수를 포함합니다.
resultType	string	판매자는 검색 결과에 상품이나 항목, 또는 상품과 항목 둘 다를 표시할 것을 지정할 수 있습니다. 이 변수의 값은 해당되는 값을 저장하는 데 사용됩니다. 값은 상품만 표시하는 1, 항목만 표시하는 2 또는 상품과 항목 둘 다를 표시하는 3 중 하나여야 합니다.
searchTerm	string	이 변수의 값은 단어 검색에 사용됩니다.

표 7. (계속)

searchTermCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.
searchTermOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.
searchTermScope	Integer	사용자는 검색 범위를 이름, 이름 및 간단한 설명, 이름, 간단한 설명 및 자세한 설명 또는 키워드 중 하나로 제한할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 이름 및 간단한 설명에 대한 1, 이름에 대한 2 또는 이름, 간단한 설명 및 자세한 설명에 대한 3, 키워드에 대한 4 중 하나여야 합니다.
searchType	string	사용자는 세 가지 유형의 검색 기준(모두, 임의 또는 정확히 일치)을 지정할 수 있습니다. 이 변수의 값은 사용자의 검색 기준을 저장하는 데 사용됩니다. 값은 모두 검색 기준에 대한 ALL, 임의 검색 기준에 대한 ANY 또는 정확히 일치 검색 기준에 대한 EXACT 중 하나여야 여야 합니다.
sku	string	이 변수의 값은 SKU 대한 검색에서 사용됩니다.
skuCaseSensitive	string	사용자는 대소문자 구분 또는 대소문자 구분 안함을 선택할 수 있습니다. 이 변수의 값은 검색의 대소문자 구분 여부를 식별하는 데 사용됩니다. 값은 대소문자를 구분하는 검색에 대한 yes나 대소문자 구분하지 않은 검색에 대한 no 중 하나여야 합니다.

표 7. (계속)

skuOperator	string	사용자는 검색 연산자로 like 또는 equal을 선택할 수 있습니다. 이 변수의 값은 사용자의 선택사항을 저장하는 데 사용됩니다. 값은 like 연산자에 대한 LIKE나 equal 연산자에 대한 EQUAL 중 하나여야 합니다.

카탈로그 검색 데이터베이스 열

카탈로그 검색 bean이 탐색할 수 있는 데이터베이스 열 목록:

표 8.

Bean 인터페이스에 정의된 검색 가능한 속성	**테이블**	**열**	**부울 표현식 지원 여부**
Category Group - 모든 이름 및 설명	CatGrpDesc	N A M E L O N G D E S C R I P T I O N SHORTDESCRIPTION	YES
Category Group - 직위만	CatGrpDesc	NAME	YES
Category Group - 직위 및 설명(제안되는 기본값)	CatGrpDesc	N A M E SHORTDESCRIPTION	YES
Category Language ID	CatGrpDesc	LANGUAGE_ID	NO
Category Identifier	CatGpEnRel	CATGROUP_ID	No
CatEntry 설명 - 모든 이름 및 설명	CatEntDesc	N A M E L O N G D E S C R I P T I O N SHORTDESCRIPTION	YES
CatEntry - 직위만	CatEntDesc	NAME	YES
CatEntry - 직위 및 설명(제안되는 기본값)	CatEntDesc	N A M E SHORTDESCRIPTION	YES
CatEntry 언어 ID	CatEntDesc	LANGUAGE_ID	NO
CatEntry 제조업체 이름	CatEntry	MFNAME	NO
CatEntry 제조업체 부품 번호	CatEntry	MFPARTNUMBER	NO
CatEntry SKU	CatEntDesc	PARTNUMBER	NO
Price	Listprice/ Offerprice	PRICE	NO
Price Range	Listprice/ Offerprice	PRICE	No
Currency	Listprice/ Offerprice	CURRENCY	NO
Quantity Available	InvStVw/ StoreInv	QTYAVAILABLE	NO
Quantity Measure	InvStVw/ StoreInv	QUANTITYMEASURE	NO

Bean의 사용자는 bean을 사용자 정의하여 충분한 속성과 같이 열을 더 노출할 수 있습니다. 사용자 정의에 대한 추가 정보는 온라인 도움말에서 ExtendedCatEntrySearchListDataBean 클래스에 대한 JavaDoc을 참조하십시오.

제 7 장 쿠폰

이 장에 설명된 요소는 WebSphere Commerce 액셀러레이터의 쿠폰 구성요소에 대한 사용자 인터페이스를 나타냅니다. 이 요소는 쿠폰, 쿠폰 판매 설정 및 판매자가 매일 수행하는 기타 태스크 작성 및 유지보수를 용이하게 합니다. 쿠폰 구성요소에는 다음 요소가 포함됩니다.

- 쿠폰 마법사
- 쿠폰 노트북

사용자 정의 예

다음 예는 WebSphere Commerce 액셀러레이터의 해당 부분을 사용자 정의하는 방법에 대해 간략하게 보여줍니다.

시나리오 1: 쿠폰 할인을 작성할 때 새 정보 추가

구현 개요

이 시나리오는 쿠폰 할인을 작성한 사람과 같이 쿠폰에 추가 정보를 기록하기 위해 쿠폰 도구를 사용자 정의하는 데 필요한 단계를 안내합니다.

사용자 정의 단계

1. 태스크 명령 구현 **CreateCouponDiscountCmdImpl**을 확장하고 CreateCouponDiscountCmd를 구현하는 새 **CustomCreateCouponDiscountCmdImpl**을 구현하십시오.
2. CALCODE 테이블의 FIELD1 열을 사용하여 쿠폰 할인을 작성한 사용자의 userId를 저장하십시오.
3. CMDREG 테이블에 새 태스크 명령을 등록하십시오. 고객 상점 ID는 1이고 다음의 SQL 문을 실행한다고 가정하십시오.

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

4. **CustomCreateCouponDiscountCmdImpl**을 갱신하십시오. 명령에서 performExecute() 메소드를 구현하십시오.

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

제 8 장 할인

이 장에 설명된 요소는 WebSphere Commerce 액셀러레이터의 할인 구성요소에 대한 사용자 인터페이스를 나타냅니다. 이 요소는 할인과 판매자가 매일 수행하는 기타 조치의 작성 및 유지보수를 용이하게 합니다. 할인 구성요소에는 다음 요소들이 포함됩니다.

- 할인 마법사
- 할인 노트북

사용자 정의 예

다음 예는 WebSphere Commerce 액셀러레이터의 할인 구성요소를 사용자 정의하는 방법에 대해 간략하게 보여줍니다.

시나리오 1: 할인을 작성할 때 추가 정보 추가

구현 개요

이 시나리오는 할인을 작성할 때 추가 정보를 추가하고자 할 경우에 필요한 단계를 설명합니다. 이 예는 할인 작성자의 감사 추적을 작성할 것으로 가정합니다. 이 시나리오에서는 새 매개변수를 제출하지 않아도 됩니다.

사용자 정의 단계

1. 태스크 명령 **CreateDiscountCmdImpl**을 확장하고 **CreateDiscountCmd**를 구현하는 새 **CustomCreateDiscountCmdImpl**을 구현하십시오.
2. **CALCODE 테이블의 field1 열을 사용하여 할인을 작성한 사용자의 userId를 저장하십시오.**
3. **CMDREG 테이블에서 새 태스크 명령을 등록하십시오.** 고객 상점 ID는 1이고, 다음 SQL 문을 실행한다고 가정합시다.

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

4. **CustomCreateDiscountCmdImpl에서** 메소드 performExecute()를 구현하십시오.

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

시나리오 2: 기본 작동 변경

구현 개요

이 시나리오는 할인이 주문에 적용되는 순서를 지정할 수 있도록 할인 마법사의 기본 작동을 변경합니다. 이 시나리오는 또한 사용자가 URL을 통해 새 매개변수를 전달하도록 요구합니다. 이 때 할인 마법사의 첫 번째 패널에 필드를 추가하여 여분의 데이터를 확보해야 합니다.

사용자 정의 단계

1. 다음을 수행하여 할인 마법사에 대한 책임이 있는 JSP 파일을 갱신하십시오.
   a. 다음 디렉토리로 변경하십시오.

      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.jsp를 MyDiscountWizardWelcome.jsp로 복사하십시오.

   c. MyDiscountWizardWelcome.jsp를 갱신하십시오. 양식 블록에서 다음 코드를 추가하십시오.

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

   d. 프레임워크에 항목을 저장하십시오. Javascript 함수 savePanelData가 있는 사용자 JSP에서 다음 행을 함수에 추가하십시오.

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

   e. VIEWREG 테이블에 새 항목을 등록하십시오. 예를 들어, 고객 상점 ID가 1일 경우 다음 SQL 문을 실행하십시오.

      ```
      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. 제어기 명령 구현 **DiscountSaveCmdImpl**을 확장하고 **DiscountSaveCmd**를 구현하는 새 **DiscountSaveCmdCustomImpl**을 구현하십시오.

3. CMDREG 테이블에 이러한 새 구현을 등록하십시오. 고객 상점 ID는 1이고 다음의 SQL 문을 실행한다고 가정합시다.

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

4. **DiscountSaveCmdCustomImpl** 명령에서 다음 견본 코드를 사용하여 두 개의 메소드를 구현하십시오.

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

제 9 장 RFQ 응답

사용자 정의 예

이 책에서는 세 가지의 사용자 정의 견본을 제공합니다. 다음과 같습니다.

- 응답 복사
- RFQ 응답의 사용 주기에서 철회된 상태 삭제
- 작성 중 응답에 대한 설명 정보 입력

시나리오 1: 응답 복사

이 견본 사용자 정의는 사용자가 선택된 RFQ 응답을 기초로 중복되는 RFQ 응답을 작성할 수 있는 기능을 RFQ 응답 도구에 추가합니다. 사용자 정의에는 누를 경우 중복 응답에 대한 고유 이름을 입력하도록 프롬프트되는 **복제 작성** 버튼 추가 작업이 포함됩니다. 새 응답이 작성되면 draft 상태에 놓입니다.

구현 개요

응답에 대한 자세한 모든 정보를 검색하고 검색된 정보를 사용하여 새 응답을 작성하십시오. 응답 이름은 현재 스키마에서 같을 수 없으므로, 사용자가 다른 이름을 제공할 수 있는 페이지를 제공해야 합니다. 이는 이 페이지의 모든 정보를 준비하고 이를 **RFQResponseCopyCmd** 명령으로 전달합니다. 이 명령을 응답 작성 이벤트를 생성합니다. 그리고 나서 UBF는 **RFQResponseCreateCmd** 명령을 호출하여 응답을 작성합니다.

사용자 정의 단계

1. 다음을 수행하여 인터페이스 **RFQResponseCopyCmd**와 해당되는 구현 **RFQResponseCopyCmdImpl**을 추가하십시오.
   a. 새 인터페이스 **RFQResponseCopyCmd**를 추가하십시오. 인터페이스를 정의하는 코드는 다음과 같습니다.

```
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.ToolsBusinessFlowEventCmdImpl을 확장하고 **RFQResponseCopyCmd**를 구현하여 명령 구현 **RFQResponseCopyCmdImpl**을 추가하십시오. 이 명령은 대화 상자를 통해 제출된 데이터를 사용하여 응답 작성 이벤트를 트리거합니다. 이벤트는 UBF가

RFQResponseCreateCmdImpl을 호출하여 새 응답을 작성하도록 합니다. 명령을 정의하는 코드는 다음과 같습니다.

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

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

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

 for (Enumeration pns = parms.keys(); pns.hasMoreElements();) {
    String paramName = (String) pns.nextElement();
 if (paramName.equals(ECConstants.EC_VIEWTASKNAME))
    responseProperties.put(ECConstants.EC_VIEWTASKNAME, "DialogNavigation");
 else if (paramName.equals(UIProperties.SUBMIT_FINISH_MESSAGE))
    responseProperties.put(UIProperties.SUBMIT_FINISH_MESSAGE,
              "The response was copied successfully!");
 else {
            Object newVal = parms.get(paramName);
            responseProperties.put(paramName, newVal);
 }
 }
 } catch (ECException e) {
    parms = e.getErrorProperties();
 responseProperties = new TypedProperty();
 responseProperties.put(ECConstants.EC_VIEWTASKNAME, "DialogNavigation");
 responseProperties.put(UIProperties.SUBMIT_ERROR_STATUS, "ERROR");
 responseProperties.put(UIProperties.SUBMIT_ERROR_MESSAGE,
         "Copying response failed, please input another name.");
 throw new ECApplicationException(new ECMessage(ECMessageSeverity.INFO,
         ECMessageType.USER, "_ERR_RESPONSE_COPY",
         "com.ibm.commerce.tools.rfq.properties.RFQMessages"),
         this.getClass().getName(), "",
         "DialogNavigation",responseProperties);
   }
}
```

c. URLREG 테이블에서 **RFQResponseCopy**를 등록하십시오. 다음 SQL을 실행하여 명령을 등록하십시오.

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

2. 사용자가 응답에 대한 이름을 입력할 수 있도록 JSP를 추가하십시오. JSP는 응답과 연관되는 모든 정보를 검색합니다.

   a. JSP 파일은 사용자가 새 응답 이름을 제공할 수 있도록 텍스트 필드를 제공합니다. 다음은 텍스트 필드에 대한 소스 코드입니다.

```
<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>
```

사용자가 **확인**을 누를 때, 다음 코드는 응답 이름을 입력 필드에 저장합니다.

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

JSP 파일을 초기화할 때, 다음 코드는 응답과 연관되는 모든 정보를 검색하여 이를 저장합니다. 정보는 **RFQResponseCopyCmdImpl**에 제공됩니다.

```
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. 다음 SQL 문을 실행하여 새 JSP 파일을 맵핑하는 VIEWREG 테이블에 보기 명령을 등록하십시오.

```
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.xml이라고 하는 새 XML 파일을 다음 디렉토리에 추가하십시오.

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

finishURL은 등록된 URL을 맵핑하는 **RFQResponseCopy**입니다. 이 대화 상자의 패널은 등록된 보기 명령을 맵핑하는 **RFQRspDuplicateDialog**입니다. XML 파일의 코드는 다음과 같습니다.

```
<?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. 이 XML을 resource.xml에 등록하십시오.

1) 다음 디렉토리에서 resource.xml을 백업하십시오.

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) 다음 두 행을 resource.xml 파일에 추가하십시오.

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

e. rfq_response_duplicate_dialog.js라고 하는 새 Javascript 파일을 다음 디렉토리에 추가하십시오.

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 파일에는 대화 상자에서 사용되는 일부 JavaScript 함수가 있습니다. 다음은 Javascript 파일의 소스 코드입니다.

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

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

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

3. 이 함수를 RFQ 응답 목록 페이지에 통합하십시오.

a. **복제 작성** 버튼을 응답 목록 페이지에 추가하십시오. 즉, 다음과 같이 XML 파일을 변경하십시오.

   주: 다음 디렉토리에 있는 XML 파일 rfq_response_list.xml을 백업하십시오.

   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.xml에서 <button>... </button> 노드의 끝에 다음 행을 추가하십시오.

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

b. 새 JavaScript 함수 getDuplicateBCT()와 duplicateRes()를 rfq_response_list.jsp에 추가하십시오.

```
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. JavaScript 함수를 rfq_response_list.jsp에 추가하고 해당되는 RFQ의 상태가 활성화가 아닐 경우 이를 호출하여 **복제 작성** 버튼을 사용하지 않도록 설정하십시오. 다음 함수를 rfq_response_list.jsp에 추가하고, DisableNewButton()를 호출한 다음 이를 호출하십시오.

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

4. 다음 디렉토리에 있는 RFQMessages_en_US.properties 파일에 새 메시지를 추가하십시오.

   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.

다음 행을 파일에 추가하십시오. 이 메시지는 복사가 실패할 때 표시됩니다.

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

시나리오 2: RFQ 응답의 사용 주기에서 철회된 상태 삭제

영역 밖에서, 사용자가 응답을 철회할 때 응답은 Active 상태를 Retracted 상태로 변경됩니다. 그 대신 사용자가 철회 후 바로 응답을 draft 상태로 설정하려면 응답의 사용 주기에서 retracted 상태를 삭제하여 기능을 사용자 정의해야 합니다.

구현 개요

이 사용자 정의에서, 먼저 응답 상태 시스템으로부터 Retracted 상태를 삭제해야 합니다. 더이상 Retracted 상태가 없을 경우, 응답 목록 페이지의 보기 목록에서 Retracted 상태도 제거해야 합니다.

사용자 정의 단계

1. UBFStatemachine.xml 및 UBFStatemachine_en_US.xml을 변경하고 이를 다시 로드하십시오. 다음 디렉토리에서 UBFStatemachine.xml 및 UBFStatemachine_en_US.xml을 백업하십시오.

   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

   다음 변경사항을 두 파일에 수행하십시오.

- retractRFQResponse 이전의 대상 상태를 RETRACTED에서 DRAFT로 변경하십시오.
- RETRACTED 상태에 대한 정보와 모든 변환을 이 상태에서 삭제하십시오.

아래의 코드 샘플은 UBFStatemachine.xml 파일에 필요한 변경사항을 보여줍니다.

변경 이전

```
<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>
```

변경 이후

```
<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.xml에 대한 변경사항은 다음과 같이 표시됩니다.

변경 이전

```
<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>
```

변경 이후

```
<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. Retracted 응답의 사용 주기에 Retracted 상태가 더이상 없으므로, rfq_response_list.xml 및 rfq_response_state_list.xml에서 다음 행을 삭제하십시오. 여기에 있는 상태는 draft, draft, cancelled, pendingapproval, rejecter, active, inevaluation, win, lost, wincomplete, lostcomplete가 있습니다.

   주: 여기에서는 표시를 위해 행이 구분되어 있습니다.

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

주:

1. 이 견본은 상태를 삭제하는 방법에 대해 보여줍니다. 상태를 추가하려면, 순서가 반대인 것을 제외하고는 유사한 단계를 수행하십시오.
   a. 상태 시스템 XML 파일에서 새 상태 이전을 추가하고 이를 다시 로드하십시오.
   b. 응답 목록 페이지의 보기 목록에 새 보기를 추가하십시오.

   상태 시스템을 다시 로드하는 방법에 대해서는 UBF 온라인 도움말을 참조하십시오.

시나리오 3: 작성 중 응답에 대한 설명 정보 입력

사용자가 RFQ에 대한 응답을 작성할 때 일부 설명 텍스트를 입력하려면, RFQ 작성 마법사의 첫 번째 패널에 텍스트 필드를 추가해야 합니다.

구현 개요

RFQ 응답 작성 마법사의 첫 번째 패널에 새 텍스트 필드를 추가하십시오. **RFQResponseCreateCmdImpl**은 여분의 입력, 새 인터페이스 **MyRFQResponseCreateCmd** 및 이 인터페이스의 구현을 처리하지 않으므로, 명령 **MyRFQResponseCmdImpl**은 **RFQResponseCreateCmd** 및 **RFQResponseCreateCmdImpl**을 확장하여 작성됩니다.

사용자 정의 단계

1. 인터페이스 **MyRFQResponseCreateCmd**와 이 인터페이스의 구현 **MyRFQResponseCreateCmdImpl**을 추가하십시오.
   a. 새 인터페이스 **MyRFQResponseCreateCmd**를 추가하십시오. 인터페이스의 코드는 다음과 같습니다.

   ```
   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**을 확장하고 **MyFQResponseCreateCmd**를 구현하여 명령 구현 **MyRFQResponseCreateCmdImpl**을 추가하십시오. 사용자 인터페이스에서 전송된 응답 설명을 저장하기 위해 새 필드 `responseDescription`을 추가하십시오. 또한 이 필드에 대한 getter 메소드와 setter 메소드를 추가하십시오.

```
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() 메소드를 대체하십시오. 이 메소드는 **RFQResponseCreateCmdImp**에서 checkParameters()에 의해 호출되어 매개변수를 초기화합니다. 이 메소드와 해당되는 최상의 클래스의 메소드 사이의 차이점은 **setResponseDescription(MyRFQConstants. EC_RFQ_RESPONSE_DESCRIPTION, false)**의 호출로 사용자 인터페이스에서 전송되는 응답 설명을 가져온다는 것입니다. 메소드의 소스 코드는 다음과 같습니다.

```
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() 메소드를 대체하십시오. 이 메소드는 **RFQResponseAccessBean** 및 **TradingDescriptionAccessBean**에 설명 필드를 설정해야 하는 것을 제외하고 같습니다. 메소드의 소스 코드는 다음과 같습니다.

```
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. 다음 SQL 문을 실행하여 **MyRFQReponseCreate** 명령을 URLREG 테이블에 등록하십시오.

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

2. RFQConstants를 확장하여 새 상수 클래스 MyRFQConstants를 작성하십시오. 하나의 새 상수 정의만 있습니다.

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

3. rfq_w_response_general.jsp 파일을 변경하여 새 텍스트 필드 및 연관되는 처리를 추가하십시오. 다음 디렉토리에서 이 파일을 찾으십시오.

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 섹션을 다음과 같이 변경하십시오.

```
<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() 함수에서 응답 설명을 저장할 코드를 추가하십시오. 변경된 함수는 다음과 같습니다.

```
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() 함수에서 응답 설명을 검색할 코드를 추가하십시오. 변경된 코드는 다음과 같습니다.

```
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 파일의 대상 finishURL을 RFQResponseCreate에서 MyRFQResponseCreate로 변경하십시오. 다음 디렉토리에서 파일을 위치 지정하십시오.

   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

   다음 코드 세그먼트에 따라 변경하십시오.

```
<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 파일을 변경하고 이를 다시 로드하십시오. 다음 디렉토리에서 파일을 위치 지정하십시오.

   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

   두 파일 모두에서 조치 인터페이스를 com.ibm.commerce.rfq.commands.RFQResponseCreateCmd에서 com.ibm.commerce.rfq.commands.RFQResponseCreateCmd로 변경하십시오. 다음 견본 코드는 필수 변경사항을 보여줍니다.

   변경 이전

   ```
   <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>
   ```

   변경 이후

   ```
   <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>
   ```

   주: 상태 시스템을 다시 로드하는 방법에 대해서는 UBF 온라인 도움말을 참조하십시오.

제 10 장 예상 재고

사용자 정의 예

다음 예에서는 WebSphere Commerce 액셀러레이터의 해당 부분을 사용자 정의하는 방법에 대해 간략하게 보여줍니다.

시나리오 1: 예상 재고 레코드를 작성할 때 새 정보 추가

이 시나리오는 예상 재고 레코드를 작성자와 같이, 예상 재고에 추가 정보를 기록하기 위한 예상 재고 도구를 사용자 정의하는 데 필요한 단계를 안내합니다.

1. 적절한 userID에 대해 열을 RA 테이블에 추가하십시오. 새 ExpectedInventoryRecords 엔터프라이즈 bean과 액세스 bean을 작성하십시오.
2. 제어기 명령 구현 **ExpectedInventoryRecordCreateCmdImp**를 확장하고 **ExpectedInventoryRecordCreateCmd**를 구현하는 새 **CustomExpectedInventoryRecordCreateCmdImpl**을 구현합니다.
3. CMDREG에 새 제어기 명령을 등록하십시오. 고객 상점 ID는 1이고, 다음의 SQL 문을 실행한다고 가정합시다.

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

4. **CustomExpectedInventoryRecordCreateCmdImpl**을 갱신하십시오. 명령에서 performExecute() 메소드를 구현하십시오.

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

제 11 장 비즈니스 정보

동적 컨텍스트

동적 컨텍스트는 사용자가 수행하기 위해 선택할 수 있는 그룹화된 조치 목록입니다. 목록에는 조치의 이름과 간략한 설명이 있습니다. 목록은 사용자 역할 및 사용하는 구성 요소를 기초로 동적으로 변경합니다.

사용자 정의 예

시나리오: 기존 컨텐스트에 새 조치 추가

새 조치 기존 컨텍스트에 추가하려면 다음을 수행하십시오.

1. 컨텍스트 정의 XML 파일에 조치를 추가하십시오. 예를 들어, xml/tools/bi 디렉토리에 biContext.xml 파일을 추가하십시오. 추가할 컨텍스트를 찾아서 항목을 컨텍스트에 추가하십시오. 다음은 캠페인 컨텍스트의 캠페인 조치입니다.

```
<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>
```

캠페인 컨텍스트를 보려면, WebSphere Commerce 액셀러레이터의 **마케팅** 메뉴에서 **캠페인**을 선택한 후 **보고서**를 누르십시오.

위의 코드 단편에서 toolsComponent는 선택 속성입니다. 이것은 지정한 경우, 조치는 지정된 구성요소가 구성 관리자에서 사용될 경우에만 나열됩니다.

사용자가 역할 요소에 정의된 역할 중 하나일 경우, 조치는 사용자에 대해 나열됩니다.

명령 요소의 이름 속성은 목록에서 조치가 선택될 때 수행된 명령의 이름입니다. 매개변수와 값 속성은 명령 매개변수와 매개변수의 값입니다.

항목 요소의 appendQueryString 속성은 위의 코드 단편에 표시되지 않습니다. 속성이 표시되고 true로 설정될 경우, 현재 요청의 요청된 특성은 이름 값 쌍 형식으

로 조치 명령이 추가됩니다. 예를 들어, 아래의 코드 샘플은 계정 보고서 컨텍스트의 계정별 주문 보고서 조치를 보여줍니다.

```
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 클래스는 위의 정의를 기초로 다음 명령을 생성합니다(명령은 한 행이며, 여기에서는 표시하기 위해 분할되어 있습니다).

```
/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
```

위의 명령에서, 다음 이름 값 쌍은 요청 특성에서 추출되어 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 명령에 추가됩니다.

2. 특성 파일에 특성 키를 추가하십시오.

   자국어를 사용할 수 있는 경우, nameKey, descriptionKey 및 breadCrumbTrailTextKey는 특성 파일의 키입니다. 위 예의 특성 파일은 properties/com/ibm/commerce/tools/bi/properties/BINLS_en_US.properties입니다. nameKey는 조치의 이름에 맵핑됩니다. descriptionKey는 조치 설명에 맵핑됩니다. breadCrumbTrailTextKey는 조치가 선택될 때 breadcrumb trail에 추가된 텍스트에 맵핑됩니다.

제 12 장 보고 프레임워크의 개요

보고 프레임워크는 사이트의 대부분의 측면에 대해 일반적인 사용자 정의 가능한 보고 기능을 제공합니다. 보고서는 Commerce 액셀러레이터를 사용하는 역할로 액세스할 수 있습니다. 특정 보고서에 대한 액세스는 보고서 내에서 정의하여 제한할 수 있습니다. WebSphere Commerce 액셀러레이터 사용자는 언제든지 보고서를 요청할 수 있으며 프레임워크는 프로덕션 데이터베이스에 있는 데이터를 사용하여 보고서를 생성하고 실시간에 보고서를 표시합니다.

프레임워크는 일반 제어기 명령, 데이터 bean 및 결과를 표시하는 일반 보기로 구성됩니다. 유효한 SQL 조회를 추가하고 생성된 보고서를 요청 및 표시하기 위해 사용되는 JSP 파일을 정의하여 프레임워크를 사용자 정의할 수 있습니다.

보고 프레임워크 사용자 정의

보고서는 Commerce 액셀러레이터에서 사용 가능합니다. 결국, 각 보고서는 몇 개의 연관된 자원을 요구합니다. 보고서 자체가 표 형식으로 표시되는 데이터로 구성되는 반면, 기반이 되는 자원은 보고서 식별자, SQL 조회, 액세스 제어 요소 등으로 구성됩니다. 보고서 요청은 서버에서 제어기 명령을 실행합니다. 제어기 명령은 보고서가 특정 보기를 지정하지 않는 한 태스크를 호출하여 일반 보기를 설정합니다. 명령은 또한 여러 필수 변수를 설정하고 이 데이터를 리턴하여 대상 JSP 파일의 ReportDataBean을 대량 자료 반입합니다. 보고서에 대한 액세스 제어는 보고서를 요청(입력)하고 표시하는 보기에서 설정됩니다. 데이터베이스에서 리턴된 결과는 해시 테이블의 벡터로 데이터 bean에 저장됩니다. 마지막으로 JSP 파일은 보고서를 표시합니다. 보고서가 비어 있을 경우, JSP 파일은 일반 텍스트 문자열을 대신 표시합니다.

보고서에 대한 액세스 제어는 보고서를 요청(입력)하고 출력하는 보기에서 설정됩니다.

새 보고서를 추가하려면 다음 단계가 필요합니다.

1. XML 파일에서 보고서를 정의하십시오.
2. 필요할 경우, 보고서가 요청되는 JSP 파일을 작성하십시오.
3. 일반 JSP가 사용되지 않으면, JSP 파일을 작성하여 보고서를 표시하십시오.

사용자 정의 예

XML 파일에서 보고서 정의

개인 보고서는 XML 파일을 사용하여 정의됩니다. 각 보고서에는 해당되는 *reportName*.xml 파일이 있습니다. 이 파일에는 보고서를 생성하는 데 필요한 모든 정보가 있습니다. *reportname*.xml 파일은 MyStoreOverviewReport에 대한 다음 예와 유사합니다.

```
<?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>
```

새 보고서를 작성하려면, 위의 예와 유사한 XML 파일을 작성해야 합니다. 각 XML 요소에 대한 자세한 설명은『올바른 XML 요소』아래에 있는 절을 참조하십시오.

올바른 XML 요소: 다음 XML 요소를 사용하여 보고서를 정의하십시오.

<Reporting></Reporting>

루트 요소입니다.

<Report></Report>

이 필수 요소는 특정 보고서를 정의합니다. 하나 이상의 <Report> 요소가 있는 단일 XML 파일에 여러 보고서를 정의할 수 있습니다. 이 요소에는 두 개의 필수 요소가 있습니다.

ReportName

보고서에 대한 고유 이름을 정의한 문자열.

online 실시간에서 보고서를 사용할 수 있는지 지정하는 부울 값. 현재 true 가 유일하게 지원되는 값입니다.

<comment></comment>

보고서를 설명할 수 있는 선택 요소. 이 문자열은 변환을 요구하지 않습니다. 이 요소는 기존 <Report> 요소 내에서만 정의할 수 있습니다.

<SQLvalue></SQLvalue>

보고서를 생성하기 위해 사용되는 SQL 조회를 정의하는 필수 요소. 이 요소는 반드시 필요하지만 비어 있을 수도 있습니다. 이 요소는 기존 <Report> 요소 내에서만 정의할 수 있습니다.

<mergeOperation></mergeOperation>

여러 SQL 조회를 하나의 보고서에 결합할 수 있도록 하는 선택 요소. 여기에는 다른 <Report> 요소의 reportName 속성을 지시하는 쉼표로 구분되는 reportName 목록이 있습니다. 참조된 보고서의 모든 SQL 조회는 같은 개수의 열을 리턴해야 하며 해시 테이블의 키를 식별하는 같은 열 이름을 사용해야 합니다. 각 SQL 조회는 다른 조회와 독립적입니다. 각 SQL 조회는 해시 테이블의 고유 벡터를 생성하며, 표시 이전에 이 벡터는 단일 보고서를 형성하기 위해 서로 간에 추가됩니다.

위의 절에 있는 상점 개요 보고서 예는 <mergeOperation> 요소의 사용을 보여줍니다. 최종 보고서의 첫 행은 하나의 SQL 조회에서 제공되고, 두 번째 행은 후속 조회에서 제공됩니다. 이 요소는 기존 <Report> 요소 내에서만 정의할 수 있습니다.

<extended_object_class></extended_object_class>

확장된 Java 클래스를 사용하여 SQL 문을 작성하기 위해 사용되는 Java 클래스 이름이 있는 선택 요소. 클래스는 보고서를 생성한 후 데이터를 다시 보고 제어 센터에 보냅니다. 반복되는 보고서를 작성하려면 이 요소를 사용하십시오. 이 요소는 기존 <Report> 요소 내에서만 정의할 수 있습니다.

<display></display>

결과 표시에 사용되는 매개변수를 정의하기 위해 사용되는 선택 요소. 이 요소는 기존 <Report> 요소 내에서만 정의할 수 있습니다.

<standardInfo></standardInfo>

ReportDataBean의 getter를 통해 액세스 가능한 요소들을 함께 그룹화하기 위해 사용되는 필수 요소. 이 요소는 대부분의 보고서에 대해 기본 요소입니다. 이 요소는 기존 <display> 요소 내에서만 정의할 수 있습니다.

<resourceBundle></resourceBundle>

보고서에 대해 사용될 특성 파일을 지정하기 위해 사용되는 필수 요소. 값은 reports/resources.xml 파일에서도 언급해야 합니다. 이 요소는 기존 <standardInfo> 요소 내에서만 정의할 수 있습니다.

<title></title>

보고서 제목을 지정하기 위해 사용되는 필수 요소. 값은 특성 파일의 키여야 합니다. 이 요소는 기존 <standardInfo> 요소 내에서만 정의할 수 있습니다.

<message></message>

보고서에 관련된 메시지를 표시하기 위해 사용되는 필수 요소. 예를 들어, 이것은 보고서에 대한 설명을 제공하기 위해 사용할 수 있습니다. 값은 특성 파일의 키여야 합니다. 이 요소는 기존 <standardInfo> 요소 내에서만 정의할 수 있습니다.

<columnTitles></columnTitles>

열 제목을 정의하기 위해 사용되는 필수 요소. 여기에는 쉼표로 구분되는 이름 목록이 포함됩니다. 이름은 특성 파일에 정의된 키여야 합니다. 특성 파일에서 키를 찾을 수 없으면, 요소에서 제공된 키는 열 제목으로 사용됩니다. 이 요소는 기존 <standardInfo> 요소 내에서만 정의할 수 있습니다.

<userDefinedParameters></userDefinedParameters>

보고 프레임워크에서 사용자 정의 요소를 정의하기 위해 사용되는 선택 요소. 보고 프레임워크는 다음 양식의 요소를 볼 수 있을 것으로 예상합니다.

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

ReportDataBean은 사용자 정의된 표시 JSP에 사용되도록 위 요소의 해시 테이블을 리턴하는 getter 메소드를 제공합니다. 이 요소는 기존 <display> 요소 내에서만 정의할 수 있습니다.

주: <display> 요소 내에 포함된 요소들이 필수 요소인 것으로 나열된 반면, 이는 선택적 표시 요소가 정의된 경우에만 적용됩니다.

SQL 조회의 변수: *reportName*.xml 파일에서 변수를 정의할 때 변수는 대괄호({*variableName*}) 내에 포함되어야 합니다. 이는 값이 클라이언트 변수이고 클라이언트 해시 테이블로부터 확보해야 하는 보고 프레임워크에 대해 표시됩니다. 위에 표시된 견본 XML 파일에서 {revenue}, {beginDate}, {endDate}, {storeent_id}, {orders}는 모두 클라이언트 변수입니다.

보고서가 요청되는 JSP 파일 작성

보고서를 생성하기 위해 필요한 정보의 양에 따라, 필요한 데이터를 수집하기 위해 대화 상자나 마법사 중에서 어느 것을 사용할 것인지 결정해야 합니다. 어느 요소가 적절하든지, JSP에는 savePanelData JavaScript 함수가 있어야 합니다.

```
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 및 reportResultPage에 대한 참조는 매개변수를 제어기 명령에 전달하기 위해 필요합니다. SQLid 및 reportXML 변수도 필수입니다. variableN 및 varProperties에서 variable1은 선택적입니다. 예에서 variable1 - variableN은 SQL 조회에서 사용되는 변수를 나타냅니다. 예를 들어, variable1 및 variable2는 beginDate 및 endDate에 의해 바꿀 수 있습니다. 그러므로 다음 코드는 savePanelData() 함수 내에 표시됩니다.

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

varProperties 변수는 특성 파일에서 값을 확보하는 변수들을 나열합니다. 예를 들어, 이는 다음과 유사하게 보일 수 있습니다.

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

오브젝트 reportResultPage가 JSP에 참조되지 않을 경우, 제어기 명령은 보고서를 표시하기 위해 보고 프레임워크에서 제공되는 일반 보기를 사용하도록 이를 설정합니다. reportResultPage.cmd를 설정하면 사용할 보기를 지정할 수 있게 됩니다.

아래의 코드 견본은 보고서에 대한 입력 데이터를 수집하기 위해 사용되는 JSP 파일의 예를 보여줍니다.

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

보고 프레임워크 명령

보고 프레임워크는 WebSphere Commerce와 함께 제공되는 다음 명령을 사용합니다.

보기 명령

표 9. 보고 프레임워크에서 사용되는 보기 명령

보기 이름	JSP 파일
ReportRedirectView	\tools\reporting\ReportRedirect.jsp
ReportGenericView	\tools\reporting\ReportGenericView.jsp

제어기 명령

표 10. 보고 프레임워크에서 사용되는 제어기 명령

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

추가 정보는 다음 패키지에 대해 WebSphere Commerce와 함께 제공된 JavaDoc 도움말을 참조하십시오.

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

- com.ibm.commerce.tools.reporting.util

보고 프레임워크 오브젝트 모델

결과 JSP 파일을 작성할 때 ReportDataBean을 사용해야 합니다. populate() 메소드에는 실시간 보고서를 지원하기 위한 로직이 있습니다. 다음 메소드는 데이터 bean에서 사용 가능합니다.

표 11.

메소드 이름	리턴 유형	설명
populate()	void	SQL 조회를 실행하여 보고서 생성
getErrorCode()	int	오류 코드에 대한 getter 메소드
getNumberOfColumns()	int	보고서의 여러 행에 대한 getter 메소드
getNumberOfRows()	int	보고서의 여러 열에 대한 getter 메소드
getColumnTitlesName(int)	string	(i+1) 열 이름에 대한 getter 메소드
getRow(i)	hashtable	(i+1) 행에 대한 getter 메소드
getValue(i.j)	string	보고서에서 (i+1,j+1) 값에 대한 getter
getValue(i,keyname)	string	키 이름과 연관되는 열의 (i+1) 행에 대한 getter 메소드
getUserDefinedParameters()	hashtable	*reportName*.xml에서 메소드를 정의한 사용자로부터 작성된 해시 테이블의 getter 메소드
getEnv()	hashtable	입력 JSP 파일에 정의된 입력 매개변수를 포함하는 해시 테이블에 대한 getter 메소드

추가 정보는 WebSphere Commerce와 함께 제공된 JavaDoc 도움말을 참조하십시오.

보고 JSP 파일에 대해 재사용 가능한 구성요소

보고 프레임워크를 사용하여 조정하는 보고서 JSP는 보고서에 대한 입력 및 출력 보기를 제공합니다.

각 보고서에는 입력 보기와 출력 보기가 있습니다. 모든 입력 JSP의 목적과 태스크는 같습니다. 입력 JSP는 같은 룩앤필을 공유하며, 공유된 입력 위젯 풀에서 다른 입력 기준을 요청합니다. 출력 보기는 공통 구조 및 포맷을 갖습니다. 보고서의 본질에 따라, 보고서 JSP의 모든 재사용 가능한 부분은 공유 가능한 구성요소로 빌드됩니다. 이러한 구성요소는 다음의 파일 구조 및 이름 지정 규칙에 따라 빌드됩니다.

여기서 *ReportName*은 새 보고서의 이름으로 바꾸고 *InputComponent*는 입력 구성요소의 이름으로 바꾸어야 합니다. 모든 입력 및 출력 보고서 JSP 파일은 Reports.properties를 이용하여 다국어 지원 문제점을 해결합니다. 그러므로 특성 파일은 XML 및 JSP 파일에 사용되는 모든 제목 및 키를 맵핑합니다.

다음은 CSA 운영 보고서에 대해 초기에 빌드된 구성요소 목록입니다.

ReportDaysWaitedHelper.jsp
편집 가능한 상자 구성요소를 기다린 일 수

ReportFulfillmentHelper.jsp
서비스 센터 선택사항 구성요소

ReportInventoryAdjustmentCodeHelper.jsp
재고 조정 코드 선택사항

ReportProductHelper.jsp
상품 구성요소 선택

ReportStartDateEndDateHelper.jsp
한 쌍의 날짜 입력 구성요소

ReportVendorHelper.jsp
공급업체 선택사항 구성요소

ReportFrameworkHelper.jsp
입력 JSP 파일에 대한 공통 기능

ReportOutputHelper.jsp
출력 페이지 포맷터 구성요소

ReportProductFindDialogView.jsp
검색 기준 입력 페이지

ProductSearch.jsp
검색 결과 선택사항 페이지

ReportProductFindView 및 ReportProductSearchView는 ReportProductHelper에서 사용되는 독립형 페이지입니다.

Common.jsp는 보고서 입력 및 출력 페이지 둘 다에 의해 공유됩니다. ReportFrameworkHelper는 모든 보고서 입력 JSP 파일에 의해 공유되고, ReportOutputHelper는 모든 보고서 출력 페이지에 적용됩니다.

다른 모든 JSP 파일은 입력 구성요소이며, 입력을 요구하는 보고서 입력 페이지에 끌어다 놓을 수 있습니다.

각 보고서에는 세 개의 XML 파일이 필요합니다. *ReportName*ReportDefinition.xml은 보고서에 사용되는 SQL 문과 보고서에서 각 열의 포맷을 정의합니다. *ReportName*ReportDefinition.xml 작성 방법에 대해서는 **보고서 프레임워크 설계 문서**를 참조하십시오.

*ReportName*ReportInputDialog.xml은 대화 상자 정의입니다. 구문은 다음과 같습니다.

```
<?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>
```

여기서 *ReportName*은 보고서 이름으로 바꿔야 하고, 창 제목은 Reports.properties 파일에서 맵핑되어야 합니다. 그러므로 reporting.reportStrings는 xml/tools/reporting/resources.xml 파일에서 정의되며 properties/com/ibm/commerce/tools/reporting/properties/Reports.properties를 지시합니다. 도움말 키는 CMHelpMap.xml에서 맵핑됩니다. 마지막으로 URL은 보고서 입력 보기 JSP의 보기 명령 이름입니다.

*ReportName*ReportOutputDialog.xml은 보고서 출력 특성을 지정하는 또다른 대화상자 정의입니다. 포맷은 다음과 같습니다.

```
<?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>
```

위의 견본 출력 보기에는 두 가지의 사용자 정의된 버튼(**인쇄** 및 **확인**)이 있습니다. 처리 기능은 출력 보기 JSP에 정의됩니다. 창의 제목은 특성 파일에서 맵핑되는 반면, 도움말 키는 XML 파일에서 맵핑됩니다.

설계는 재사용 가능한 구성요소 개념에 기초합니다. 입력 보기 JSP 파일은 보고서 스펙에 따라 기준 항목 세트로 구성됩니다. 기준 항목은 다른 보고서의 입력 페이지에 표시될 수 있으므로, 각 기준 항목은 재사용 가능한 구성요소로 빌드됩니다. 이 전략을 입력 보기 JSP 페이지 설계에 적용하면 다음이 가능하게 됩니다.

1. 쉽게 새 보고서 입력 보기 작성
2. 모든 보고서 입력 보기의 일관성 있는 룩앤필
3. 도구 프레임워크가 요구하는 단일화된 유효성 확인, 로드 및 저장 기능(이러한 기능은 각 구성요소에 내장되어 있음)

출력 보기 JSP 파일도 재사용 가능한 헬퍼에 내장됩니다. 헬퍼는 데이터 유형 및 언어 환경설정을 기초로 필요한 모든 포맷팅 및 변환을 제공합니다. 각 열에 대한 데이터 유형은 보고서 정의 XML의 사용자 정의된 섹션에 지정됩니다. 헬퍼 및 XML 정의를 조정하면 복잡한 출력 포맷이 지원될 때 출력 보기 작성이 간단하게 됩니다.

다음 절에서는 재사용 가능한 구성요소를 사용하여 보고서 입력 보기와 출력 보기 JSP 파일을 작성하는 방법에 대해 설명합니다.

보고서 입력 보기 JSP 파일을 작성하려면 JSP 파일을 필요로 하는 필수 구성요소를 반입해야 합니다. 다음 구성요소를 사용할 수 있습니다.

1. DaysWaited - 오늘까지 기다린 일 수를 지정합니다.
2. FulfillmentCenter - 서비스 센터 선택을 용이하게 합니다.
3. InventoryAdjustment - 재고 조정 코드 선택을 용이하게 합니다.
4. StartDateEndDate - 기간을 지정합니다.
5. Vendor - 공급업체 선택을 용이하게 합니다.

설계 전략과 일치하는 한 다른 구성요소를 작성할 수 있습니다. 나중에 헬퍼 작성에 대해 논의할 것입니다.

이러한 모든 구성요소는 JSP 페이지 개발자에 대한 공통 인터페이스를 제공했습니다.

1. JSP 함수

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

   이 함수는 입력 보기에서 구성요소를 작성합니다. *InputComponent*는 구성요소의 이름입니다. 각 구성요소는 이 구성요소에 대한 JavaScript 오브젝트를 식별하는 고유한 이름의 containerName을 가지고 있습니다. 모든 JavaScript 함수는 containerName을 참조합니다. 모든 구성요소는 현재 언어 환경설정을 반영하는 보고서 자원 번들 reportsRB를 요구합니다. 각 구성요소에는 최소한 하나의 제목(레이블에서 맵핑되는)이 있습니다.

2. JavaScript 함수:

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

   이 함수는 페이지 로드시 호출됩니다. 이것이 처음으로 페이지가 트랜잭션 내에서 로드되는 경우라면, 데이터 bean에서 *InputComponent*를 초기화하십시오. 이 페이지가 트랜잭션 내에서 다시 로드될 경우, 저장된 데이터를 검색하십시오.

function validate*InputComponent*(containerName)

이 함수는 요청을 제출하기 전에 호출해야 합니다. 해당 구성요소의 입력 데이터에 대한 유효성을 확인합니다. 데이터가 올바르지 않을 경우, 사용자에게 상기시키기 위한 해당 메시지가 있는 대화 상자 창이 뜹니다. 데이터가 올바르면 true를 리턴하고, 데이터의 일부가 올바르지 않으면 false를 리턴합니다.

function save*InputComponent*(containerName)

이 함수는 현재 페이지에서 다른 페이지로 탐색할 때마다 호출해야 합니다. 현재 페이지로 다시 돌아가서 탐색할 때 나중 검색을 위해 구성요소의 현재 입력 데이터를 저장합니다.

function visibleList(state)

콜백 함수. 이는 프레임워크가 페이지에서 선택 상자를 표시하거나 숨기려고 할 때 호출됩니다. 이 함수는 다음을 호출해야 합니다.

setSelect*Component*Visible(container, state)

선택 상자가 사용되는 각 입력 구성요소에서 정의됩니다. 모든 입력 구성요소는 다른 이름의 같은 구현을 가지고 있습니다.

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

보고서 입력 JSP 페이지의 이름은 *ReportName*ReportInputView.jsp로 지정해야 하고, 다음 디렉토리에 위치되어야 합니다.

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.

이는 대화 상자 패널입니다. 대화 상자 패널을 구현하려면, Tools Framework User's Guide를 참조하십시오. 대화 상자 패널에는 HTML 컨텐츠 생성 섹션과 JavaScript 함수 섹션, 두 가지 섹션이 있습니다. HTML 섹션에서 입력 화면에 표시할 각 구성요소에 대해 generate*InputComponent*를 호출할 수 있습니다. JavaScript 섹션에서 initializeValue, savePanelData, validatePanelData는 각 입력 구성요소에 정의된 해당되는 JavaScript 함수를 호출해야 합니다. initializedValue는 페이지가 로

드될 때 호출됩니다. 도구 프레임워크는 savePanelData 및 validatePanelData를 호출합니다. SavePanelData는 다음 보고서 프레임워크를 JavaScript 함수에 다음 보고서 프레임워크를 호출해야 합니다.

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

또한 setReportFrameworkParameter("*name*", *value*)를 호출하여, 보고서 출력 JSP 파일에서 필요로 하는 매개변수를 설정할 수도 있습니다. 이는 이름 및 값 쌍입니다. 작성기에 전달된 모든 레이블과 setReportFrameworkParameter 함수 호출의 값은 로케일 및 언어 환경설정을 기초로 해당되는 문자열에 맵핑될 키입니다. 맵핑은 다음 디렉토리에서 특성 파일 Reports_en_US.properties에 정의됩니다.

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

보고서 입력 및 출력 페이지에 대한 헬퍼 사용

대화 상자 패널인 보고서 입력 페이지는 입력 기준에 맞는 구성요소 세트를 가지고 있어야 하며 다음 네 가지의 java 스크립트 함수를 구현해야 합니다.

initializeValues()

페이지가 로드될 때마다 호출됩니다.

savePanelData()

이 페이지에서 사용자가 종료할 때마다 호출됩니다.

validatePanelData()

보고 프레임워크에 기준을 보내기 전에 호출됩니다.

visibleList()

보고 프레임워크가 이 페이지의 구성요소 가시성 설정에서 변경을 요구할 때 호출됩니다.

구성요소를 보고서 입력 페이지에 추가하려면, 구성요소 JSP를 반입하고 하나의 JSP 표현식을 입력 페이지에 반입하십시오. 이는 이름 지정 규칙에 의해 컨테이너 이름(이 페이지에 고유함), 자원 번들 그리고 매개변수로 하나 또는 두 개의 제목을 사용하여 generate*InputComponent*로 이름이 지정됩니다. 예를 들어 입력 페이지에서 다음과 유사한 것을 수반하게 됩니다.

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

표현식은 페이지에서 볼 수 있는 구성요소를 생성할 문자열을 리턴합니다. 위의 세 함수로 조정하기 위해, 이름 지정 규칙을 사용하여 콜백 함수가 정의됩니다 (onLoad*InputComponent*, save*InputComponent* 및 validate*InputComponent*). 이 함수들은 각각 initializeValue, savePanelData, validatePanelDate 내에서 호출해야 합니다.

SavePanelData 함수는 또한 보고 프레임워크에서 필요로 하는 정보를 저장합니다. 각 입력 구성요소는 해당 구성요소에 입력된 ID, 이름 또는 기타 필드를 되돌리는 리턴 함수를 제공합니다. 이러한 리턴 함수에 대한 자세한 내용은 입력 구성요소 JSP를 참조하십시오.

출력 페이지는 포맷팅 처리에 대한 책임이 있습니다. 모든 포맷팅 메소드는 보고서 정의 XML 파일로 조정되어 ReportOutputHelper에 포함됩니다. 보고서 출력 JSP 파일에서는 보고서 이름만 지정해야 합니다. 보고서 출력 JSP 파일을 복사하고 reportPrefix 값을 수정하여 보고서 이름을 반영할 수 있습니다.

그러나 보고서 정의 XML 파일은 열 유형을 기초로 모든 열과 해당되는 포맷팅을 지정합니다. 다음은 열 유형 목록입니다.

표 12.

열 유형	기본값	사용자 정의 가능 특성	기본 맞추기
문자열	Yes	maxEntryLength	오른쪽
정수	No	setMinimumIntegerDigits setMaximumIntegerDigits	왼쪽
10진수	No	setMinimumIntegerDigits setMaximumIntegerDigits setMinimumFractionDigits setMaximumFractionDigits	왼쪽
통화	No	currencySymbolColumn	왼쪽
열거	No		오른쪽
날짜	No		왼쪽

표 12. (계속)

열 유형	기본값	사용자 정의 가능 특성	기본 맞추기
시간	No		왼쪽
달	No		왼쪽

기본 열 유형은 HTML 열 옵션 "align=left height=20 nowrap"가 잇는 문자열입니다. 모든 열 유형은 열에서 <columnOptions> 태그를 지정하여 기본 열 옵션을 대체할 수 있습니다.

선택적으로, 모든 열은 true 또는 false로 설정된 displayInReport 값이 있을 수 있습니다. 기본값은 true로, 설정되면 이는 열이 보고서에 표시됨을 의미합니다. 값이 false로 설정되면, 열은 숨겨집니다. 이 기능은 SQL 조회를 변경하지 않고 보고서 출력 보기를 사용자 정의하기 위해 사용될 수 있습니다. 이는 또한 통화 포맷팅에서 통화를 포맷터에 알려주기 위한 참조 열이 필요할 경우에 유용합니다.

정수, 10진수, 날짜 및 시간 열은 명령 컨텍스트에 지정된 언어 및 통화 값을 기초로 포맷팅됩니다. 정수 및 10진수 열은 정수와 분수 값 둘 다에 대해 최소 및 최대 숫자를 지정할 수도 있습니다.

기본적으로 통화 열은 명령 컨텍스트에 지정된 언어 및 통화 값을 기초로 포맷팅됩니다. 통화 열에 currencySymbolColumn을 지정할 경우, 데이터베이스에서 세 자로 된 통화 기호가 검색되어 통화 포맷팅에 사용됩니다. 참조된 통화 기호 열은 보고서 작성자가 통화 기호 문자열을 표시하지 않으려고 할 경우에 보이지 않도록 설정할 수 있습니다.

열거는 데이터베이스에서 검색된 값이 키에 지정된 문자열에 맵핑되는 특수 열 유형입니다. 예를 들어, 테이블에서 Y 또는 N이 검색될 수 있습니다. 이 값은 다른 보고서나 다른 언어에서 Yes 또는 No나, Approved 또는 Denied와 같이 더 의미 있는 문자열로 맵핑될 수 있습니다. 이렇게 하려면 열을 다음과 같이 정의하면 됩니다.

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

또는

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

여기서 Yes, No, Approved, Denied에 대한 문자열은 여러 언어로도 사용 가능하도록 해당 특성 파일에 정의됩니다. 조회에서 특정 값을 맵핑할 0,1,2 등과 같은 숫자 값을 리턴할 경우, 요소로 <X_n></X_n>을 사용해야 합니다. 예를 들면 다음과 같습니다.

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

재사용 가능한 JSP 페이지 구성요소를 이용하여 보고서 작성

재사용 가능한 JSP 페이지 구성요소를 이용하여 보고서를 작성하려면, 다음을 작성해야 합니다.

- JSP 파일:
  XXXReportInputView.jsp
  XXXReportOutputView.jsp
- XML 파일:
  XXXReportInputDialog.xml
  XXXReportDefinition.xml
  XXXReportOutputDialog.xml
- 갱신된 특성 파일:
  Reports_en_US.properties(필요한 모든 맵핑에 대해 섹션을 추가해야 함)
- 데이터베이스에 보기 명령을 추가하십시오.
  XXXReportInputView 및 XXXReportOutputView 명령을 데이터베이스에 추가해야 합니다.
- 추가 된 두 보기(XXXReportInputView 및 XXXReportOutputView)의 액세스 제어를 이전 단계에서 설정하십시오.

제 13 장 상품 어드바이저

사용자 정의 예

이 책에서는 세 가지의 사용자 정의 견본을 제공합니다. 다음과 같습니다.

- 다른 연산자 아이콘을 사용하여 견본 상품 탐색 JSP 파일 변경
- 상품 비교에 사용된 기본 링크 변경
- 제공된 위젯을 사용하지 않고 상품 탐색 렌더링

시나리오 1: 다른 연산자 아이콘 사용

상품 탐색 시뮬레이션에서 연산자 아이콘(하나의 이미지가 모든 연산자를 표시함)에서 선택의 X 좌표는 선택된 연산자를 판별하기 위해 사용됩니다. 이미지를 바꿀 때, 이미지를 눌러서 적절한 연산자를 식별할 수 있도록 좌표가 동일하게 있는지 확인하십시오. 이미지 파일은 CommerceDir\web\tools\pa\icons에 있습니다. equalone.gif는 =, <> 연산자를 나타내고, equaltoo.gif는 숫자 속성 값에 대해 사용되는 =,<>, <=, >= 연산자를 나타냅니다. 예상되는 X 좌표는 다음과 같습니다.

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

값 선택에서 X 좌표 매개변수를 전달할 때, 적절한 연산자가 선택에 적용됩니다.

시나리오 2: 상품 비교 시뮬레이션으로부터의 링크

상품 비교 시뮬레이션은 개별 항목에서 다른 페이지로의 링크를 지원합니다. 현재 단 하나의 다른 페이지의 링크만 지원됩니다(값에 대해 URL을 사용하여 속성을 작성하지 않을 경우). 견본 JSP는 ProductDisplay 페이지에 링크합니다. ProductCompareDataBean에서 productLinkName 특성을 통해 이 페이지를 지정하십시오. 견본에서는 상품 비교 시뮬레이션 사용에 대한 통계를 수집하는 ClickInfo 명령을 통해 경로가 재지정됩니다. 다른 페이지에 링크할 때, 상품 비교 페이지로부터 매개변수를 통해 전달해야 할 수도 있습니다. ProductCompareDataBean의 productLinkParameters 특성을 통해 전달할 매개변수를 식별하십시오. 여기에서 식별되는 매개변수 이름은 매개변수가 존재할 경우 링크 페이지를 통해 전달되는 매개변수를 수반하게 됩니다. 또한 견본에 ECConstants.EC_PRODUCT_ID로 표시되는 productId 매개변수에는 링크와 연관되는 항목에 대한 카탈로그 항목 ID(catentry_id)의 값이 지정됩니다. productId는 사용자가 선택한 항목을 링크 페이지가 식별할 수 있도록 합니다. 상품 비교 테이블의 개별 항목에 대해 고유한 값을 사용하는 다른 매개변수에 대한 지원은 현재 없습니다.

시나리오 3: 상품 탐색 렌더링의 사용자 정의

ProductExploreDataBean은 DynamicForm 위젯에 의해 렌더링됩니다. 사용자 스스로 렌더링하려면, DynamicForm 클래스의 서브클래스를 생성하고(메소드 서명에 대해 JavaDocs 참조) render 메소드를 대체하거나, ProductExploreDataBean에서 직접 데이터를 가져오십시오. render 메소드 내에서 getDataBean() 메소드를 사용하여 ProductExploreDataBean 오브젝트를 가져오십시오. ProductExploreDataBean.getFormElements() 메소드를 사용하여 각 속성 열을 나타내는 ColumnDataBeans의 콜렉션을 가져오십시오. 각각의 열 내에서, ColumnDataBean 오브젝트에는 해당되는 열의 값이 있습니다. ColumnDataBean.getColumn() 메소드를 사용하여 개별 속성 값을 포함하는 DsData 오브젝트의 콜렉션을 가져오십시오. 메소드 정보에 대해서는 ColumnDataBean 및 DsData에 대한 JavaDoc을 참조하십시오. DsData의 getPresentationString() 메소드를 사용하여 속성값의 포맷된 표시와 원래 데이터에 대한 getUnformattedData() 메소드를 검색하십시오. 사용자의 HTML 양식에 맞는 올바른 매개변수를 작성하려면, ColumnDataBean.getFormElementName() 메소드를 사용하여 적절한 매개변수 이름을 검색하고 DsData.getUnformattedData() 메소드를 사용하여 각각의 값을 가져오십시오. 적절한 연산자 선택에 대해서는 연산자 아이콘 정보를 참조하십시오.

제 14 장 규칙 프로젝트

규칙 프로젝트의 사용 주기는 다음 단계에 포함됩니다.

1. 규칙 프로젝트 작성
2. 새 규칙 프로젝트를 기초로 규칙 서비스 구성
3. 규칙 서비스 호출
4. 규칙 프로젝트를 기초로 규칙 서비스 제거

가정: 여기에 수록된 정보에서는 2, 3, 4만 다룹니다. 규칙 프로젝트는 이미 작성되어 있는 것으로 가정합니다.

규칙 프로젝트를 작성하는 방법에 대해서는 Blaze 전문 서비스를 문의하십시오.

사용자 정의된 규칙 프로젝트를 기초로 규칙 서비스를 구성하는 방법

규칙 시스템은 WebSphere Commerce 또는 사용자 정의된 규칙 프로젝트와 함께 제공되는 규칙 프로젝트 사이를 구별하지 않습니다. 항상 관리 콘솔을 사용하여 규칙 프로젝트로부터 규칙 서비스를 작성, 수정 또는 제거할 수 있습니다. 그러나 사용자 정의된 규칙 프로젝트는 특정 요구사항을 만족해야 합니다. 요구사항은 다음과 같습니다.

1. 규칙 프로젝트에 이송된 외부 이벤트는 규칙 프로젝트의 실행을 트리거해야 합니다.
2. 외부 이벤트는 `com.ibm.commerce.rules.InvocationContext` 인터페이스를 구현해야 합니다. 이 인터페이스는 유일한 단순 태그 인터페이스입니다.

사용자 정의된 규칙 프로젝트를 기초로 규칙 서비스를 호출하는 방법

WebSphere Commerce와 함께 제공된 규칙 프로젝트에서 작성된 규칙 서비스와 같은 형태로 규칙 서비스(위의 두 기준에 알맞는 규칙 프로젝트에서 작성된)를 호출할 수 있습니다. 즉, 태스크 명령 `com.ibm.commerce.rules.commands.InvokePersonalizationRuleServiceCommand`를 호출합니다.

제 2 부 부록

주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급하는 것이 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운용에 대한 평가 및 검증은 사용자의 책임입니다.

또한 이 책에서 IBM의 사용 허가된 프로그램을 언급하는 것이 해당 IBM 프로그램만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 타사의 기능상 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수 있습니다. IBM이 명시적으로 지정한 경우를 제외하고, 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 문서를 제공한다고 해서 특허에 대한 사용권까지 제공하는 것은 아닙니다. 사용권에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전호번호: 080-023-8080

2바이트(DBCS) 정보에 관한 사용권 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

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

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다.

IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을

"현상태대로" 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에 기술된 제품 및(또는) 프로그램을 사전 통고없이 언제든지 개선 및(또는) 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램 및 기타 프로그램(이 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 사용권자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건(예를 들어, 사용료 지불 등)에 따라 사용할 수 있습니다.

이 정보에 기술된 사용권 프로그램 및 사용 가능한 모든 사용권 자료는 IBM이 IBM 기본 계약, IBM 프로그램 사용권 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 레벨 상태의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 더우기, 일부 측정치는 추정을 통해 추측되었을 수도 있습니다. 실제 결과는 다를 수 있습니다. 이 책의 사용자는 본인의 특정 환경에 대해 해당 데이터를 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 비IBM 제품을 테스트하지 않았으므로, 이들 제품과 관련된 성능의 정확성, 호환성 또는 배상 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 어떠한 언급도 특별한 통지없이 변경될 수 있습니다.

이 정보는 계획 수립 목적으로만 사용됩니다. 이 정보는 기술된 제품이 GA(General Availibility)되기 전에 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이 예제에는 가능한 완벽하게 개념을 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 포함될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

이 제품에 나오는 신용 카드 이미지, 상표 및 상호는 해당 신용 카드에 의한 지불을 승인하는 신용 카드 상표 소유주에 의해 인가된 판매자만이 사용할 수 있습니다.

상표

다음 용어는 미국 또는 기타 국가에서 사용되는 International Business Machines Corporation의 상표 또는 등록상표입니다.

Blaze Advisor는 미국 또는 기타 국가에서 사용되는 HNC Software Inc.의 상표입니다.

Microsoft, Windows, Windows NT 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표 또는 등록상표입니다.

Netscape는 미국 또는 기타 국가에서 사용되는 Netscape Communications Corporation의 등록상표입니다.

Oracle은 미국 또는 기타 국가에서 사용되는 Oracle Corporation의 상표 또는 등록상표입니다.

Java, JavaBeans, 및 모든 Java 기반 상표 및 로고는 Sun Microsystems, Inc.의 상표 또는 등록상표입니다.

기타 회사, 제품 및 서비스 이름은 타사의 상표 또는 서비스표입니다.

IBM 한글 지원에 관한 설문

FAX : (02) 3787-0123

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목: IBM® WebSphere™ Commerce
WebSphere Commerce 액셀레이터
사용자 정의 안내서
버전 5.4

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☞ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

_____

_____

_____

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.