Management center customizations lab

What this exercise is about 1

Lab requirements 1

What you should be able to do 1

Introduction 2

Exercise instructions 2

Part 1: Change the text on a header label 3

Part 2: Move sections from the existing product pane into a new tab pane 6

Part 3: Add new advanced search option for searching catalog entries 10

What this exercise is about

The objective of this lab is to provide you with an understanding on how to customize the management center user interface. This lab provides instructions for some simple but common customization scenarios for the new business use tools.

This lab is provided **AS-IS**, with no formal IBM support.

Lab requirements

To perform the lab you must have the following:

- WebSphere Commerce Developer Version 6.0
- WebSphere Commerce Feature Pack 3.0.1
- Enabled management-center feature from Feature Pack 3.0.1.
- Enabled the ConsumerDirect store to be used by the management center by removing any extra languages and making sure there is only one supported language in the store.

NOTE: You should not copy the code directly from this instruction. All the lines of code required to be entered throughout this exercise are included in text file FEP301_ManagementCenterLabSolutions.txt. You can find this file after you extract LabFile.zip.

What you should be able to do

At the end of this lab you should be able to:

- Change any message string in the management center
- Know how to add/move/remove things around in a properties view

- Customize the advanced search dialog to add extra search options

Introduction

The WebSphere Commerce management center is the new line of business user tool that comes with Feature Pack 3. This lab exercise is designed to get you familiar with the programming techniques of Open Laszlo in order to perform some simple customizations in the management center user interface.

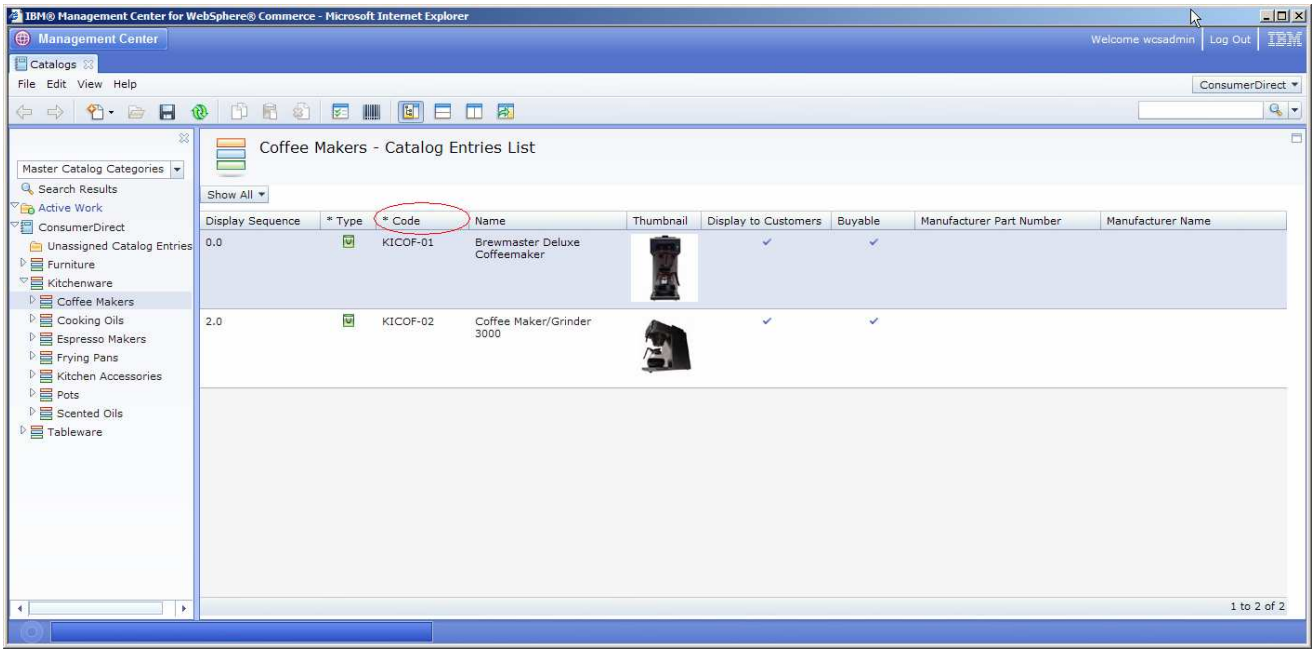
Exercise instructions

Before you start, ensure of the following:

- WebSphere Commerce test server is started. In WebSphere Commerce Developer Servers view, right click on WebSphere Commerce Test Server and select Start.
- Management center URL https://localhost:8000/lobtools/cmc/ManagementCenter works properly.
- Know the user and password of the site administrator for your system.

Part 1: Change the text on a header label

You are required to change one of the headers of the “Catalog Entries List” view. The requirement is to change the “Code” header text to “Part Number”. The following image shows the page that is being referred to:



To get to this page:

1. Launch the management center from a browser.
2. Logon as site administrator.
3. Select Catalogs from the application menu.
4. Select the ConsumerDirect store in the store selection area.
5. Expand the ConsumerDirect node in the explorer view.
6. Expand Kitchenware category.
7. Click on Coffee Makers category.

Exercise Steps

____ 1. Locate the key to the “Code” string that you want to change.

__ a. List view for this scenario comes from the default wcfNavigationListDefinition defined in the CatalogGroupPrimaryObjectDefinition.lzx. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog > objectDefinitions.

__ b. Open the CatalogGroupPrimaryObjectDefinition.lzx, find the wcfNavigationListDefinition which has the isDefault flag set to “true”. You should come to the conclusion that it should be:

```
<wcfNavigationListDefinition
      name="childCatEntriesNavList"
      listClass="catCatalogEntryChildList"
      listTitle="${catalogResources.catalogEntriesList.string}"
      displayName="${catalogResources.catalogEntriesListDisplayName.s
      tring"
      isDefault="true"
      toolbarIcon="catalogEntriesListToolbarIcon" />
```

__ c. The listClass attribute of the wcfNavigationListDefinition references the class that will eventually define the contents of the list view.

__ d. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog > listViewDefinitions.

__ e. Open the CatalogEntryGrid.lzx, search for the listClass attribute value catCatalogEntryChildList that you found in step b above.

__ f. You should see that it takes you to the catCatalogEntryChildList, which is a child list editor with another listClass attribute of catCatalogEntryBrowseGrid.

__ g. In the same CatalogEntryGrid.lzx, search for the class definition of catCatalogEntryBrowseGrid. In this class, you should be able to easily map the list view column entries to the code.

__ h. It should be clear to you that the key that you are looking for is: productPartnumber_ColumnHeader

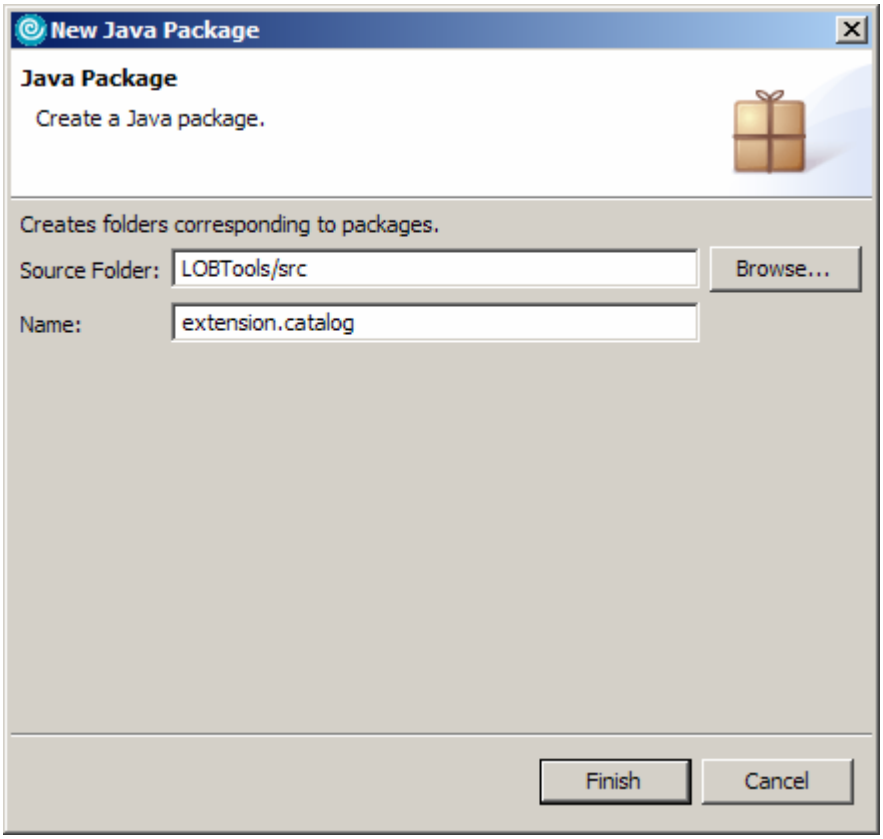
____ 2. Create an extension properties file to overwrite this message

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > Java Resources > src.

__ b. Right click on src, select New > Package.

__ c. Type extension.catalog in the Name field

__ d. Click Finish.



__ e. Right click on the new package created above.

__ f. Select New > Other > Simple > File.

__ g. Click on Next.

__ h. Under the File name enter CatalogLOB_en_US.properties.

__ i. Click on Finish.

__ j. In the new file, enter:

```
productPartnumber_ColumnHeader = Part Number
```

__ k. Save the file.

____ 3. Verify the changes.

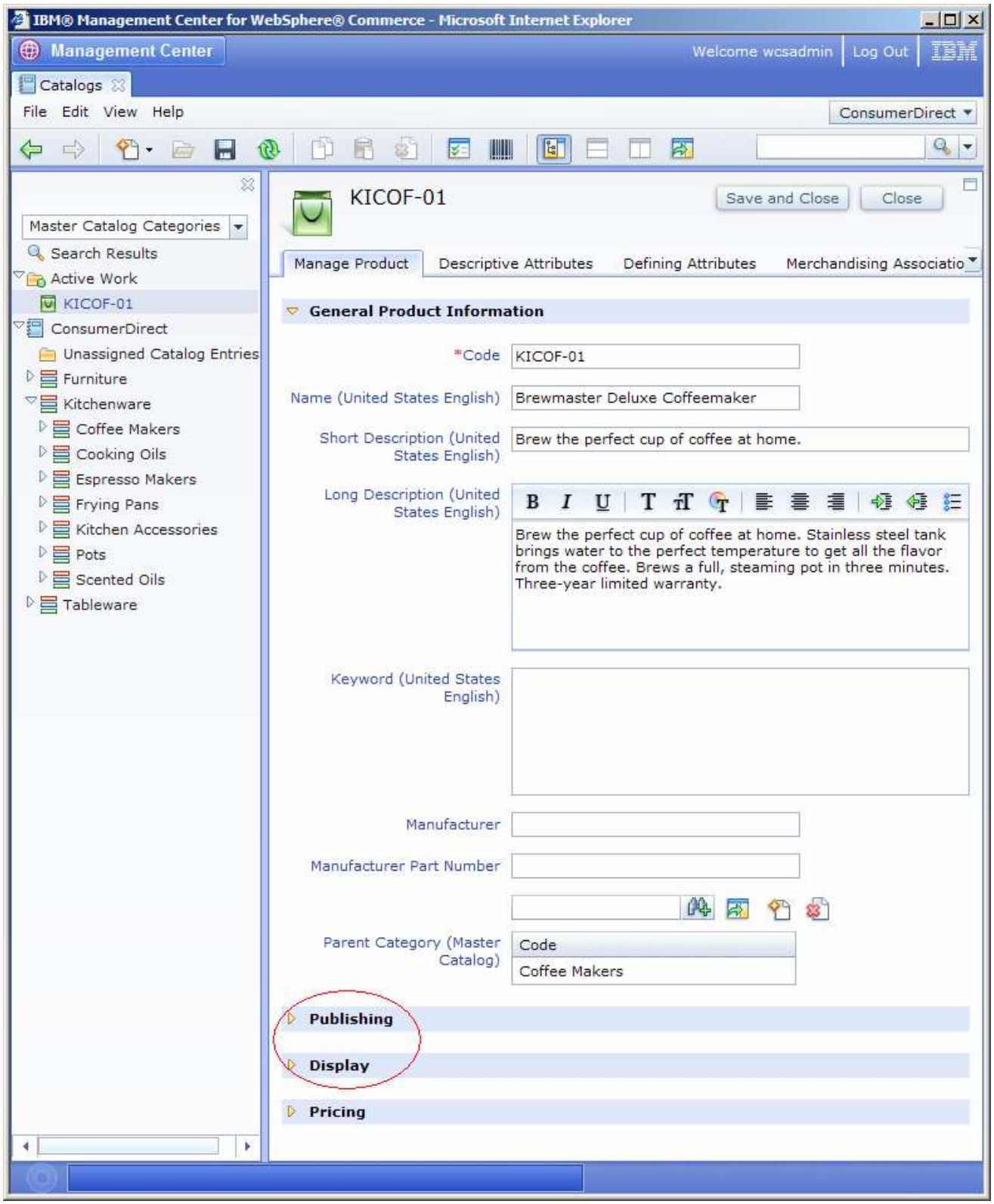
__ a. Open a browser and go to management center with this URL:
https://localhost:8000/lobtools/cmc/ManagementCenter?developmentMode=true

---

NOTE: Setting the developmentMode flag to true causes the management center to load all the properties files from the file system. This eliminates the need to restart the server to see the changes of a properties file. If you do a server restart then you can access the management center with the usual URL.

__ b. Repeat the steps of the Part 1 introduction section to see the changes.

Part 2: Move sections from the existing product pane into a new tab pane



You are required to change the Products properties view. The requirement is to move the Publishing and Display group disclosures into a new tab that should appear after Manage Product tab. The new tab is to be called Storefront Properties. The above image shows the page that you need to modify.

To get to this page:

1. Open a browser and go to the management center.
2. Logon as site administrator.
3. Select Catalogs from the application menu.
4. Select the ConsumerDirect store in the store selection area.
5. Expand the ConsumerDirect node in the explorer view.
6. Expand Kitchenware category.
7. Click on Coffee Makers category.
8. Select one of the products, right click and select Open.

Exercise Steps

____ 1. As part of this exercise you need to create a new key to hold the new string “Storefront Properties”. To do this you need to create a new Open Laszlo class to access the extension resource bundle.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx.

__ b. Right click on lzx, select New > New Folder.

__ c. Type mycompany in the Name field.

__ d. Click Finish.

__ e. Right click on mycompany, select New > Other > Simple > File.

__ f. Click on Next.

__ g. Type extCatalogManagementResourceBundle.lzx.

__ h. Click on Finish.

__ i. Enter this as the contents of the file:

```
<library>
      <class name=”extCatalogResourceBundle”
                  extends=”wcfResourceBundle”
                  baseName=”extension.catalog.CatalogLOB”>
            <wcfResourceBundleKey name=”storefrontTab”/>
      </class>
      <extCatalogResourceBundle id=”extCatalogResources”/>
</library>
```

____ 2. Define the new message in the resource bundle.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > Java Resources > src > extension.catalog.

__ b. Open the CatalogLOB_en_US.properties.

__ c. Add the following to the end of the file:

```
storefrontTab = Storefront Properties
```

____ 3. Define the new lzx file in the CatalogExtensionsLibrary.lzx so that it is picked up by the compiler.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog.

__ b. Open the CatalogExtensionsLibrary.lzx file.

__ c. Enter this:

```
<include
href="../../mycompany/extCatalogManagementResourceBundle.lzx" />
```

____ 4. Locate the file that describes the page you want to change.

__ a. Note that only primary objects can be created, updated, and deleted and so they have a properties view. Also, ProductPrimaryObjectDefinition represents the Product business object in the management center.

__ b. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog > objectDefinitions.

__ c. Open the ProductPrimaryObjectDefinition.lzx. Look at the value of propertiesClass, it should be catProductProperties. This value tells you the class that defines the properties view.

__ d. Note that all properties view classes are defined under the propertiesViews directory.

__ e. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog > propertiesViews.

__ f. Open the ProductPropertiesView.lzx, search for the catProductProperties. You should be able to easily map the page with the code.

____ 5. Create the new tab after the first tab.

__ a. Add this code after the manageProductTab wcfPropertyTabPane:

```
<wcfPropertyTabPane name="storefrontTab"
            text="${extCatalogResources.storefrontTab.string}">
      <myComStorefrontTab/>
</wcfPropertyTabPane>
```

__ b. Add this to the top of the file after the library tag:

```
<class name="myComStorefrontTab" extends="wcfPropertyPane">
      <catManagePublishingInformation/>
      <catManageDisplayInformation/>
</class>
```

__ c. Open the CatalogPropertyPane.lzx where the catManageProduct class is defined. Remove the two classes catManagePublishingInformation and catManageDisplayInformation instantiations as you have moved it to your new class.

__ d. Press Ctrl + Shift + S to save all files.

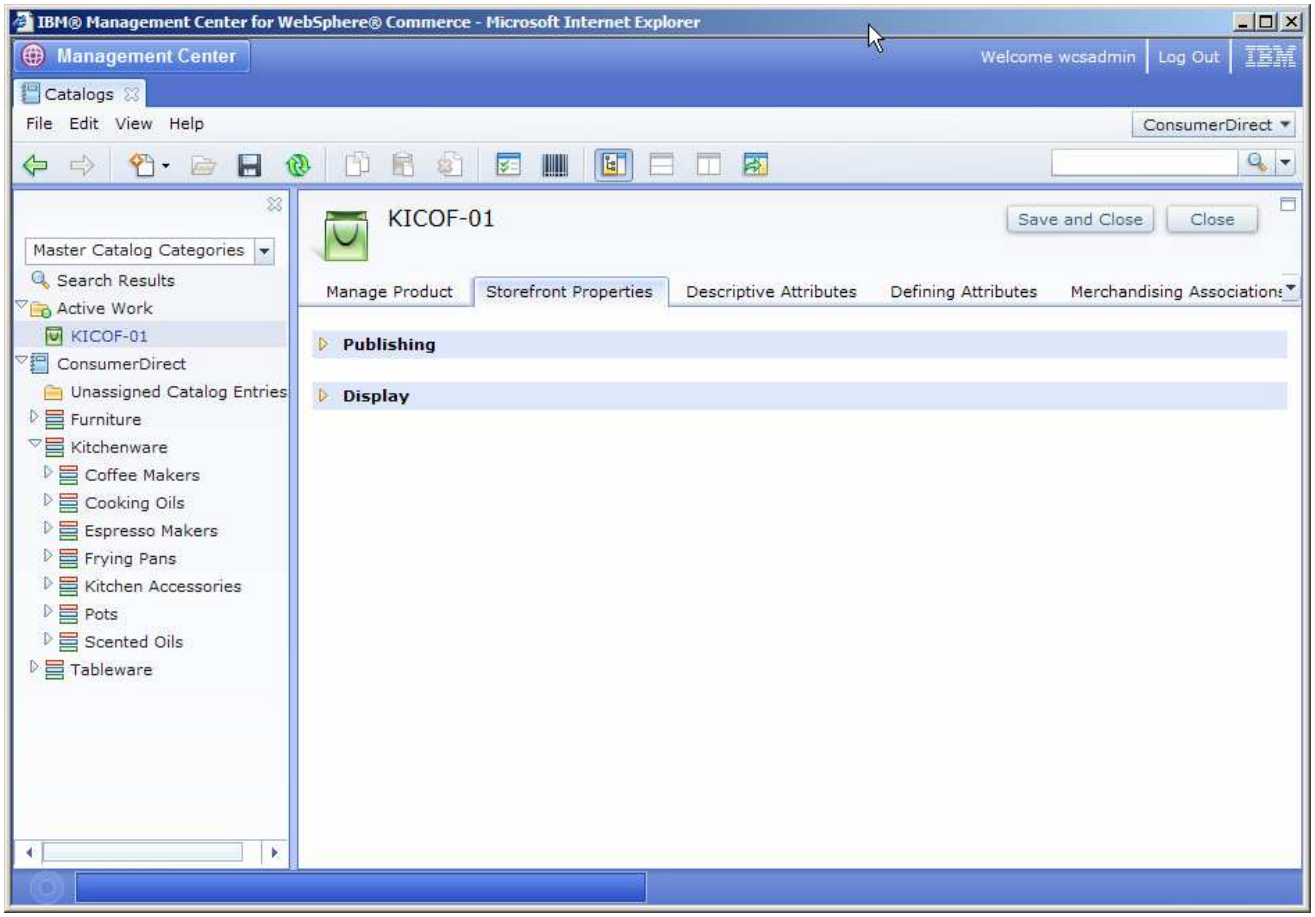
____ 6. Verify the changes.

__ a. Open a browser and go to management center with this URL:
https://localhost:8000/lobtools/cmc/ManagementCenter?developmentMode=true

---

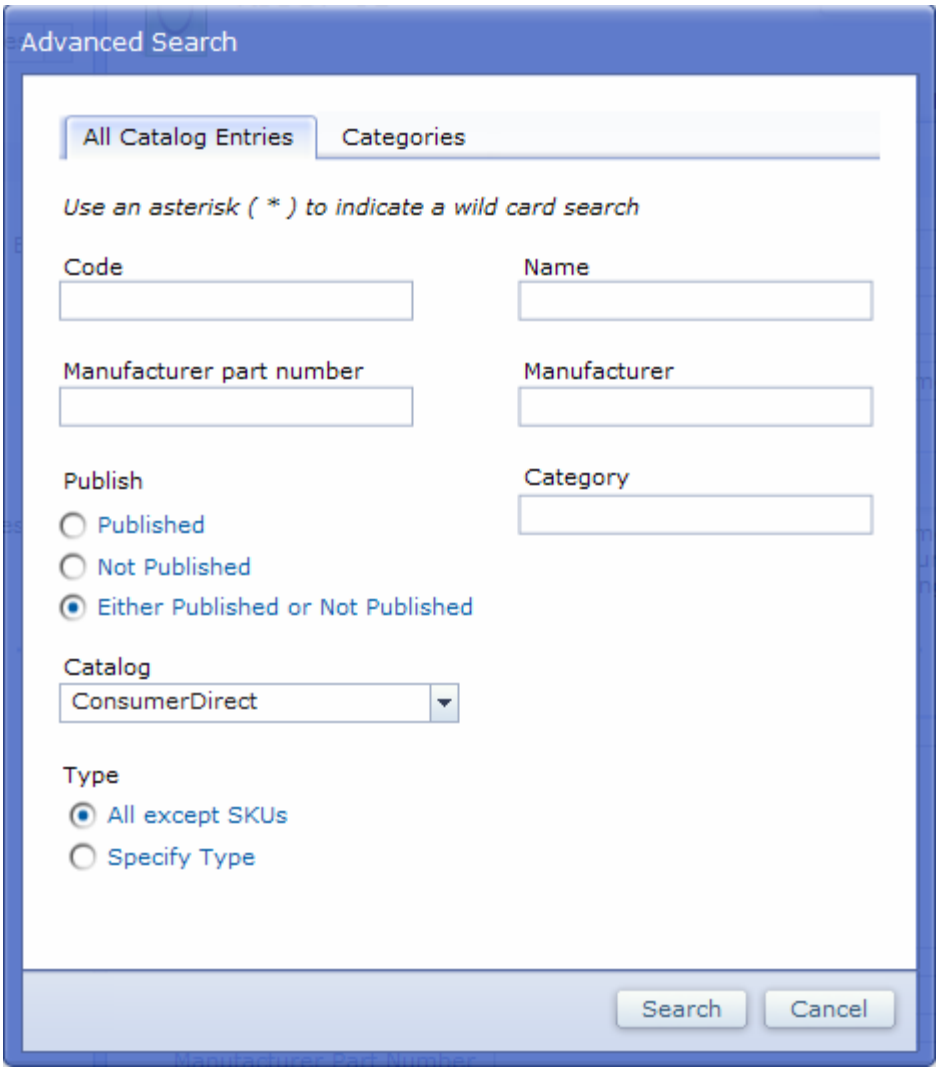
NOTE: Setting the developmentMode to true causes the management center to load all the properties files from the file system. This eliminates the need to restart the server to see the changes of a properties file. If you do a server restart then you can access the management center with the usual URL.

__ b. Repeat the steps of the Part 3 introduction section to see the changes.



Part 3: Add new advanced search option for searching catalog entries

You need to add a new search option to the All Catalog Entries advanced search dialog. The requirement is to be able to find the catalog entries that were recently updated, for example: last week, this week, last month, and this month. This image shows that page:



To get to this page:

1. Open a browser and go to the management center.
2. Logon as site administrator.
3. Select Catalogs from the application menu.
4. Click on the drop down icon and select Advanced Search in the Find Area.

Exercise Steps

Management center user interface changes

____ 1. Define the new messages in the resource bundle.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > Java Resources > src > extension.catalog.

__ b. Open the CatalogLOB_en_US.properties.

__ c. Add the following to the end of the file:

```
recentlyUpdatedLabel = Search for catalog entries updated:
none = None
thisWeek = This week
lastWeek = Last week
thisMonth = This month
lastMonth = Last month
```

____ 2. Create new keys for the new string that you have defined above.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > mycompany.

__ b. Open the extCatalogManagementResourceBundle.lzx.

__ c. Add these keys:

```
<wcfResourceBundleKey name=”recentlyUpdatedLabel”/>
<wcfResourceBundleKey name=”none”/>
<wcfResourceBundleKey name=”thisWeek”/>
<wcfResourceBundleKey name=”lastWeek”/>
<wcfResourceBundleKey name=”thisMonth”/>
<wcfResourceBundleKey name=”lastMonth”/>
```

____ 3. As part of this exercise you need to create a new Open Laszlo class that extends the stock advanced catalog class to add the new search option.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > mycompany.

__ b. Right click on mycompany, select New > Other > Simple > File.

__ c. Click on Next.

__ d. Type MyComAdvancedSearch.lzx.

__ e. Click on Finish.

__ f. Enter this as the contents of the file:

```
<library>
  <class name="myComAllCatalogEntriesAdvancedSearchContent"
              extends="catAllCatalogEntriesAdvancedSearchContent">
        <view name="section7" width="410">
        <simplelayout axis="y"/>
        <view name="filler7" height="10"/>
        <text
              text="${extCatalogResources.recentlyUpdatedLabel.string
                    }" width="100%"/>
        <radiogroup x="5" name="updatedSelection">
```

```
                    <radiobutton value="0"
                          text="${extCatalogResources.none.string}"
                          selected="true"/>
                    <radiobutton value="1"
                          text="${extCatalogResources.thisWeek.string}"/>
                    <radiobutton value="2"
                          text="${extCatalogResources.thisMonth.string}"/>
                    <radiobutton value="3"
                          text="${extCatalogResources.lastWeek.string}"/>
                    <radiobutton value="4"
                          text="${extCatalogResources.lastMonth.string}"/>
              </radiogroup>
              </view>

              <method name="setSearchOptions">
              <![CDATA[
                    super.setSearchOptions();
                    this.searchOptions.updatedSelection =
                                this.section7.updatedSelection.getValue();
                    ]]>
              </method>
        </class>
  </library>
```

____ 4. Define the new lzx file in the CatalogExtensionsLibrary.lzx so that it is picked up by the compiler.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog.

__ b. Open the CatalogExtensionsLibrary.lzx file.

__ c. Enter this:

```
<include href="../../mycompany/MyComAdvancedSearch.lzx" />
```

____ 5. Modify the stock search definition class and point it to the new class that was created in step 3.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > src > lzx > commerce > catalog > searchDefinitions.

__ b. Open the FindAllCatalogEntriesSearchDefinition.lzx file.

__ c. Replace the catAllCatalogEntriesAdvancedSearchContent occurrence with myComAllCatalogEntriesAdvancedSearchContent.

__ d. Press Ctrl + Shift + S in WebSphere Commerce Developer to save all files and start the Open Laszlo compiler.

Web application changes

____ 6. Create a custom get-data configuration file to build the new search expression. Get-data configuration files are used to build search expressions for WebSphere Commerce services that can be called by a JSP file. In this step you create a custom get-data configuration file, and define a new search expression called myComFindAllCatentriesAdvancedSearch which is similar to the original one but differs in accepting query on the catentry.lastupdate field. This expression uses the following WebSphere Commerce extended XPath notation search function to define the custom search: *search(lastupdate>'$startDate$' and lastupdate<'$endDate$').* Under the WebSphere Commerce SOA architecture, the parameters of this search function are dynamically converted into

a SQL where clause. In this example, the search function is converted to the following SQL where clause: *(CATENTRY.LASTUPDATE>'startDate' AND CATENTRY.LASTUPDATE<'endDate')*. To define the new search expression:

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF > config.

__ b. Right click on config, select New > Folder

__ c. Type com.ibm.commerce.catalog-ext in the Folder name field.

__ d. Click on Finish.

__ e. Right click on com.ibm.commerce.catalog-ext, select New > Other > Simple > File.

__ f. Click on Next.

__ g. Type get-data-config.xml.

__ h. Click on Finish.

__ i. Enter this on the new file:

```
<?xml version="1.0" encoding="UTF-8"?>
<_config:get-data-config
xmlns:_config="http://www.ibm.com/xmlns/prod/commerce/foundation/conf
ig" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.ibm.com/xmlns/prod/commerce/foundation
/config ../../xsd/get-data-config.xsd ">

      <data-type>
            <name>CatalogEntry</name>
            <type>com.ibm.commerce.catalog.facade.datatypes.CatalogEn
      tryType</type>
      </data-type>
      <client-facade>
            <data-type-name>CatalogEntry</data-type-name>
            <class>com.ibm.commerce.catalog.facade.client.CatalogFaca
      deClient</class>
            <method>getCatalogEntry</method>
      </client-facade>

      <expression-builder>
          <name>myComFindAllCatentriesAdvancedSearch</name>
          <data-type-name>CatalogEntry</data-type-name>

<class>com.ibm.commerce.catalog.internal.client.taglib.util.CatalogSe
archExpressionBuilder</class>
              <method>formatExpression</method>
              <param>
                        <name>template</name>
                        <value>/CatalogEntry[($catEntryTypes$) and
search(lastupdate&gt;='$startDate$' and lastupdate&lt;='$endDate$'
and
CatalogEntryIdentifier/ExternalIdentifier/PartNumber='$partNumber$'
and Description/Name='$name$' and
Description/ShortDescription='$shortDescription$' and
Description/Attributes/published='$published$' and
CatalogEntryAttribute/Attributes/mfPartNumber='$mfPartNumber$' and
CatalogEntryAttribute/Attributes/mfName='$mfName$') and
ParentCatalogGroupIdentifier[ExternalIdentifier[GroupIdentifier='$gro
upIdentifier$']]]</value>
          </param>
```

```
                <param>
                        <name>accessProfile</name>
                        <value>IBM_Details</value>
                </param>
                <param>
                        <name>searchType</name>
                        <value>catentry-advanced</value>
                </param>
            </expression-builder>

        </_config:get-data-config>
```

__ j. Save the file.

____ 7. Create a new JSP to call the new search expression builder and pass the new parameters. You are essentially taking the stock FindAllCatalogEntries.jsp and making small modifications.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > jsp.

__ b. Right click on jsp, select New > Folder

__ c. Type mycompany in the Folder name field.

__ d. Click on Finish.

__ e. Right click on mycompany, select New > Folder.

__ f. Type catalog in the Folder name field.

__ g. Click on Finish.

__ h. Right click on catalog, select New > Other > Simple > File.

__ i. Type FindAllCatalogEntries.jsp.

__ j. Click on Finish.

__ k. Enter this on the new file:

```
<?xml version="1.0" encoding="UTF-8"?>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"
%><%@ taglib uri="http://commerce.ibm.com/foundation" prefix="wcf"
%><%@ page import="java.util.Calendar"
%><%@ page import="java.util.GregorianCalendar"
%><%@ page import="java.text.SimpleDateFormat"
%><%
String lastUpdateSelection =
request.getParameter("updatedSelection");

Calendar timeNow = GregorianCalendar.getInstance(); // current date
Calendar startDate = GregorianCalendar.getInstance(); // the start
date
Calendar endDate = GregorianCalendar.getInstance(); // the end date
SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

if (lastUpdateSelection.equals("1")) {
      startDate.add( Calendar.DATE,
(startDate.get(Calendar.DAY_OF_WEEK)-1) * -1);
      startDate.set( Calendar.HOUR_OF_DAY, 0);
      startDate.set( Calendar.MINUTE, 0);
      startDate.set( Calendar.SECOND, 0);
      startDate.set( Calendar.MILLISECOND, 0);
```

```
} else if (lastUpdateSelection.equals("2")) {
      startDate.set( Calendar.DATE, 1);
      startDate.set( Calendar.HOUR_OF_DAY, 0);
      startDate.set( Calendar.MINUTE, 0);
      startDate.set( Calendar.SECOND, 0);
      startDate.set( Calendar.MILLISECOND, 0);
} else if (lastUpdateSelection.equals("3")) {
      startDate.add( Calendar.DATE,  (
(startDate.get(Calendar.DAY_OF_WEEK)-1) * -1) -7);
      startDate.set( Calendar.HOUR_OF_DAY, 0);
      startDate.set( Calendar.MINUTE, 0);
      startDate.set( Calendar.SECOND, 0);
      startDate.set( Calendar.MILLISECOND, 0);
      endDate.add( Calendar.DATE,
(endDate.get(Calendar.DAY_OF_WEEK)-1) * -1);
      endDate.set( Calendar.HOUR_OF_DAY, 0);
      endDate.set( Calendar.MINUTE, 0);
      endDate.set( Calendar.SECOND, 0);
      endDate.set( Calendar.MILLISECOND, 0);
} else if (lastUpdateSelection.equals("4")) {
      startDate.set( Calendar.DATE, 1);
      startDate.add( Calendar.MONTH, -1);
      startDate.set( Calendar.HOUR_OF_DAY, 0);
      startDate.set( Calendar.MINUTE, 0);
      startDate.set( Calendar.SECOND, 0);
      startDate.set( Calendar.MILLISECOND, 0);
      endDate.add( Calendar.DATE,
(endDate.get(Calendar.DAY_OF_MONTH)-1) * -1);
      endDate.set( Calendar.HOUR_OF_DAY, 0);
      endDate.set( Calendar.MINUTE, 0);
      endDate.set( Calendar.SECOND, 0);
      endDate.set( Calendar.MILLISECOND, 0);
}

String istrStartDate = sdf.format(startDate.getTime());
String istrEndDate = sdf.format(endDate.getTime());
request.setAttribute("startDate", istrStartDate);
request.setAttribute("endDate", istrEndDate);
%>

<c:choose>
      <c:when test="${(empty param.searchText) && (empty
param.mfPartNumber) && (empty param.manufacturer)
                              && (empty param.parentCategory) &&
(empty param.catalogEntryCode) && (empty param.catalogEntryName)
                              && ((empty param.published) ||
(param.published == '3'))}">
            <%-- No search criteria is specified --%>
            <objects
                  recordSetCompleteIndicator="true"
                  recordSetReferenceId=""
                  recordSetStartNumber=""
                  recordSetCount="0"
                  recordSetTotal="0">
            </objects>
      </c:when>

      <c:when test="${(empty param.searchText) && (catentryTypes ==
'2') && (param.typeProducts == 'false')
      && (param.typeSKUs == 'false') && (param.typeBundles ==
'false') &&  (param.typeKits == 'false')}">
            <objects
                  recordSetCompleteIndicator="true"
                  recordSetReferenceId=""
```

```
                  recordSetStartNumber=""
                  recordSetCount="0"
                  recordSetTotal="0">
            </objects>
      </c:when>

      <c:otherwise>
            <%-- Decide which expression builder to call based on the
input --%>
            <c:choose>
                  <c:when test="${! (empty param.searchText )}">
                        <c:set var="expressionBuilderName"
value="findAllCatentriesBasicSearch"/>
                        <c:set var="catentryCode"
value="${param.searchText}"/>
                        <c:set var="catentryName"
value="${param.searchText}"/>
                  </c:when>
                  <c:when test="${(empty param.searchText) && (empty
param.mfPartNumber) && (empty param.manufacturer)
                              && (empty param.catalogEntryCode) &&
(empty param.catalogEntryName) && (param.published == '3') &&
(param.updatedSelection == '0')}">
                        <c:set var="expressionBuilderName"
value="findAllCatentriesByParentCatgroupAdvancedSearch"/>
                  </c:when>
                  <c:when test="${param.updatedSelection != '0'}">
                        <c:set var="expressionBuilderName"
value="myComFindAllCatentriesAdvancedSearch"/>
                  </c:when>
                  <c:otherwise>
                        <c:set var="expressionBuilderName"
value="findAllCatentriesAdvancedSearch"/>
                  </c:otherwise>
            </c:choose>

            <c:set var="catalog" value="${param.masterCatalogId}"/>

            <c:if test="${ !(empty
param.catalogSelectionCatalogEntry) &&
(param.catalogSelectionCatalogEntry != 'undefined') }">
                  <c:set var="catalog"
value="${param.catalogSelectionCatalogEntry}"/>
            </c:if>

            <wcf:getData
type="com.ibm.commerce.catalog.facade.datatypes.CatalogEntryType[]"
                  var="catentries"
                  expressionBuilder="${expressionBuilderName}"
                  varShowVerb="showVerb"

      recordSetStartNumber="${param.recordSetStartNumber}"

      recordSetReferenceId="${param.recordSetReferenceId}"
                  maxItems="${param.maxItems}">
                  <wcf:contextData name="storeId"
data="${param.storeId}"/>
                  <wcf:contextData name="catalogId"
data="${catalog}"/>
                  <wcf:param name="shortDescription" value=""/>
                  <wcf:param name="mfPartNumber"
value="${param.mfPartNumber}"/>
                  <wcf:param name="mfName"
value="${param.manufacturer}"/>
```

```
                    <wcf:param name="groupIdentifier"
value="${param.parentCategory}"/>

                    <c:if test="${! (empty param.catalogEntryCode )}">
                          <c:set var="catentryCode"
value="${param.catalogEntryCode}"/>
                    </c:if>

                    <c:if test="${! (empty param.catalogEntryName)}">
                          <c:set var="catentryName"
value="${param.catalogEntryName}"/>
                    </c:if>

                    <wcf:param name="partNumber"
value="${catentryCode}"/>
                    <wcf:param name="name" value="${catentryName}"/>

                    <c:set var="productExp" value="ProductBean"/>
                    <c:set var="bundleExp" value="BundleBean"/>
                    <c:set var="kitExp" value="PackageBean"/>
                    <c:set var="dynamicKitExp" value="DynamicKitBean"/>
                    <c:set var="SKUExp" value="ItemBean"/>

                          <c:if test="${(empty param.published)}">
                                <wcf:param name="published" value=""/>
                          </c:if>

                          <c:if test="${param.published == '1'}" >
                                <wcf:param name="published" value="1"/>

                          </c:if>

                          <c:if test="${param.published == '2'}">
                                <wcf:param name="published" value="0"/>

                          </c:if>

                          <c:if test="${param.published == '3'}">
                                <wcf:param name="published" value=""/>

                          </c:if>

                          <c:if test="${(empty param.catentryTypes)}">
                                      <wcf:param name="catEntryTypes"
value="${productExp},${bundleExp},${kitExp},${dynamicKitExp},${SKUExp
}"/>
                          </c:if>
                          <c:if test="${param.catentryTypes == '1'}">
                                <wcf:param name="catEntryTypes"
value="${productExp},${bundleExp},${kitExp},${dynamicKitExp}"/>

                          </c:if>
                          <c:if test="${param.catentryTypes == '2'}">
                                <c:set var="typeParam" value=""/>

                                <c:if test="${param.typeProducts ==
'true'}">
                                      <c:if test="${typeParam != ''}" >
                                            <c:set var="typeParam"
value="${typeParam},"/>
                                      </c:if>
                                      <c:set var="typeParam"
value="${typeParam}${productExp}"/>
                                </c:if>
```

```
                                    <c:if test="${param.typeBundles ==
'true'}">
                                          <c:if test="${typeParam != ''}" >
                                                <c:set var="typeParam"
value="${typeParam},"/>
                                          </c:if>
                                          <c:set var="typeParam"
value="${typeParam}${bundleExp}"/>
                                    </c:if>

                                    <c:if test="${param.typeKits ==
'true'}">
                                          <c:if test="${typeParam != ''}" >
                                                <c:set var="typeParam"
value="${typeParam},"/>
                                          </c:if>
                                          <c:set var="typeParam"
value="${typeParam}${kitExp},${dynamicKitExp}"/>
                                    </c:if>

                                    <c:if test="${param.typeSKUs ==
'true'}">
                                          <c:if test="${typeParam != ''}" >
                                                <c:set var="typeParam"
value="${typeParam},"/>
                                          </c:if>
                                          <c:set var="typeParam"
value="${typeParam}${SKUExp}"/>
                                    </c:if>

                                    <wcf:param name="catEntryTypes"
value="${typeParam}"/>
                              </c:if>

                              <c:if test="${param.updatedSelection !=
'0'}">
                                    <wcf:param name="startDate"
value="${requestScope.startDate}"/>
                                    <wcf:param name="endDate"
value="${requestScope.endDate}"/>
                              </c:if>

                  </wcf:getData>


                  <jsp:directive.include
file="../../commerce/catalog/restricted/serialize/SerializeCatalogEnt
ries.jspf"/>
            </c:otherwise>
</c:choose>
```

____ 8. Re-map the search service URL to the new JSP file.

__ a. In WebSphere Commerce Developer, using the J2EE perspective, navigate to Dynamic Web Projects > LOBTools > WebContent > WEB-INF.

__ b. Open the struts-extension.xml file.

__ c. Add the following to the action-mapping section:

```
<action path="/FindCatalogEntries-All"
forward="/jsp/mycompany/catalog/FindAllCatalogEntries.jsp" />
```

Data Service Layer changes

____ 9. On the file system, use windows explorer and navigate to this directory: *WCDE_installdir*\xml\config\com.ibm.commerce.catalog

____ 10. Open the wc-component.xml with an editor (for example WorkPad), search for this:

```
<_config:basetable name="CATENTRY" useAllColumns="false">
```

and replace it with

```
<_config:basetable name="CATENTRY" useAllColumns="true">
```

____ 11. Restart the WebSphere Commerce server.

Verify changes

____ 12. Verify the changes.

__ a. Open a browser and go to management center URL.

__ b. Repeat the steps of the Part 3 introduction section to see the changes.