



IBM Software Group

WebSphere® Commerce V 6.0 Feature Pack 3

Management Center customizations



@business on demand.

© 2008 IBM Corporation
Updated May 2, 2008

Welcome to the WebSphere Commerce Feature Pack 3, Management Center customizations presentation.

Agenda

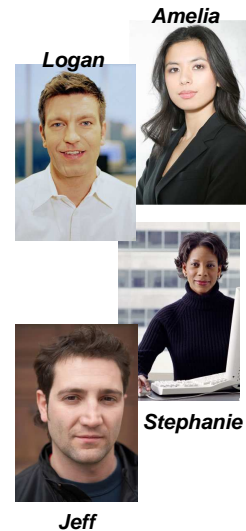
- Management Center overview
- Key concepts
- Development flow and customization approach



This presentation provides an overview of the Management Center and why it was created followed by its architecture and the technology chosen and highlights of some key concepts for understanding the programming model for the Management Center. Finally, it looks at some code samples and shows some customization details.

Objectives for the new business user tools

- Empowering business users
 - ▶ Consistent and productive UI tailored to the role
 - Category manager, product manager, marketing manager
 - ▶ Business user scenarios to drive the tools' capability and flow
 - Create products and SKUs, enrich and manage product content, create/manage a new promotion
- Minimizing cost of implementation
 - ▶ Reduce cost to IT role when extending or customizing the tools using declarative programming approach



The IBM Management Center for WebSphere Commerce, or Management Center, provides the next generation line of business tools for managing business tasks in online businesses. In Feature Pack 3, Management Center replaces the previous Accelerator tool for catalog, promotion and marketing management functions. Details on the functions provided can be found in other Feature Pack 3 presentation modules.

Another important aspect of the new tool is its ease of customization. Management Center introduces a new way of working with business objects that is consistent across the catalog, promotion and marketing tools.

Management Center at a glance

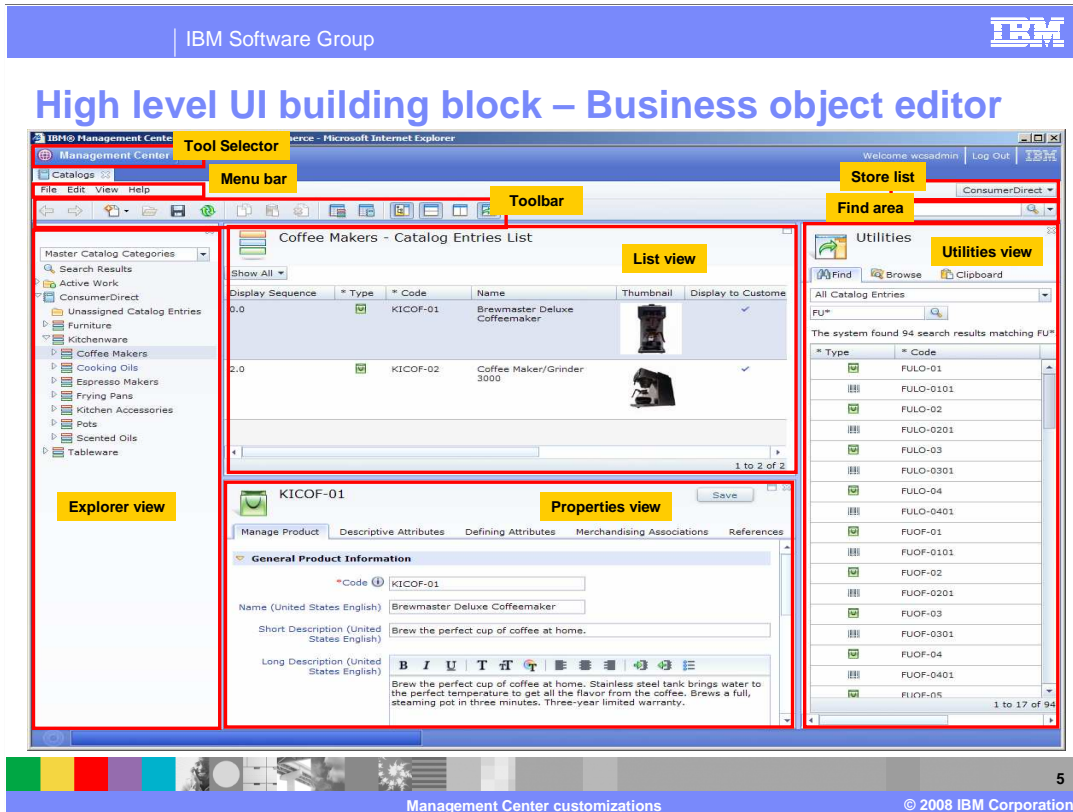
IBM® Management Center for WebSphere® Commerce

■ The Management Center UI consists of

- ▶ A shell that handles user logon and provides role-based business user tools
- ▶ An editor per business user tool

The Management Center has a common shell component that takes care of the functions that are common across all business user tools. This includes, logon, session management, access control and user preferences among others. The three business user tools, catalogs, marketing and promotions sit on top of the shell. Each tool has its own editor which allows all three tools to be open at the same time.

The shell contains a few customization points, but most customization points are in the individual business user tools.



Here is a complete view of all the features that the Management Center provides. Notice that the Management Center is very different than the layout of the Accelerator – with colorful graphics, buttons and different views.

Starting from the top, the tool selector allows you to select from the various tools available. The menu bar and toolbar link to functions that can be done as you work with the Management Center. On the right side, you see the store selection list and the find area to perform searches.

Moving down the diagram, there is an explorer view where you navigate and select the object that you need to work with. In the content area, made up of the list view and properties view, you can open up the business objects to work with them.

On the right side, you can open up the utilities view. Use the utilities view to locate objects you require to complete a task, without navigating away from the current view. From the Find, Browse, or Clipboard tabs, drag objects into the Management Center main work area.

Management Center takes on a new development approach where each tool is created by instantiating a high level widget called the business object editor. By declaring a business object editor, all the features of the framework are inherited.

Low level UI components

- Check box
- Combination box
- Date and time editor
- List view
- Calendar view
- Text editor
- Rich text editor
- Tab
- Group (disclosure control)
- Value selector

The screenshot displays several IBM Management Center UI components:

- Date editor:** A calendar for August 2007.
- Time editor:** A vertical time picker showing 02:00 PM to 11:00 PM.
- List view:** A table listing table lamps with columns for Type, Image, SKUs, Name, Short Description, and Price.

Type	Image	SKUs	Name	Short Description	Price
FULO-02		Mocha Linen Table Lamp	A mocha linen table lamp to illuminate any interior.	\$149.99	
FULO-0401		Brown Linen Table Lamp	A brown linen table lamp to illuminate any interior.	\$79.99	
FULO-04		Beige Linen Table Lamp	A beige linen table lamp to illuminate any interior.	\$149.99	
- Calendar view:** A Gantt-style chart showing activity bars across months from May 2007 to July 2007.
- Rich text editor:** A window titled "Long Description" with a rich text editor toolbar and the text "Beige Linen Table Lamp".
- Disclosure control:** A control with the text "Open Section Title" and "Disclosure control".
- Value Selector:** A numeric input field with the value "0" and up/down arrows.



A wide variety of low level widgets are available for displaying lists and properties of an object. These widgets are used in the various list and property views. They are available for you to reuse if you customize or add views.

By making use of the appropriate widget, you can provide a simple and intuitive way for users to edit information. For example, date and time pickers allow users to select a value from a visual representation. Also, rich text editors provide greater control for editing descriptive text while calendar views display objects in a chart that graphically represents some period of time.

Technology choice: OpenLaszlo + Flash

- OpenLaszlo v4.0 (www.openlaszlo.org)
 - ▶ An open source rich internet application platform for the web
 - ▶ Declarative UI layout and component/class definition
 - Using XML and JavaScript
 - ▶ Provide a rich library of UI components
 - ▶ Can generate to either Flash or DHTML
 - Management Center uses Flash
- Flash as the rendering platform
 - ▶ Consistent across platforms and browsers
 - ▶ Provides graphics rendering, scripting, HTTP connectivity



The technology chosen for developing the Management Center is called OpenLaszlo, which is an open source Flash development toolkit. OpenLaszlo is an object oriented language that facilitates the creation of rich internet applications. It is a combination of declarative XML and JavaScript.

The Management Center code is generated to Flash to ensure display consistency across different platforms and browsers. This means there is no need to worry about testing JavaScript code across multiple browsers and versions.

OpenLaszlo code sample

```

<class name="wcfWindow" extends="basewindow">
  <!-- The title of a window. It appears at the top of a window.-->
  <attribute name="title" type="string" value=""/>

  <!-- @keywords private -->
  <attribute name="defaultplacement" value="content" type="string"/>
  <!-- @keywords private -->
  <attribute name="useContentWidth" value="true" type="boolean"/>
  <!-- @keywords private -->
  <attribute name="useContentHeight" value="true" type="boolean"/>

  <!-- @keywords private -->
  <method name="construct" args="parent,args">
    <![CDATA[
      super.construct(parent, args);

      // Determine if the size of the window is based on its
      // content or that specified as window attributes.
      // Content size is used if and only if the window does not
      // have its own size(width or height) specified.
      setAttribute('useContentWidth', !hassetwidth);
      setAttribute('useContentHeight' !hassetheight);
    ]]>
  </method>
  <view name="content"/>
</class>

```

← Class Inheritance

← Embedded JavaScript

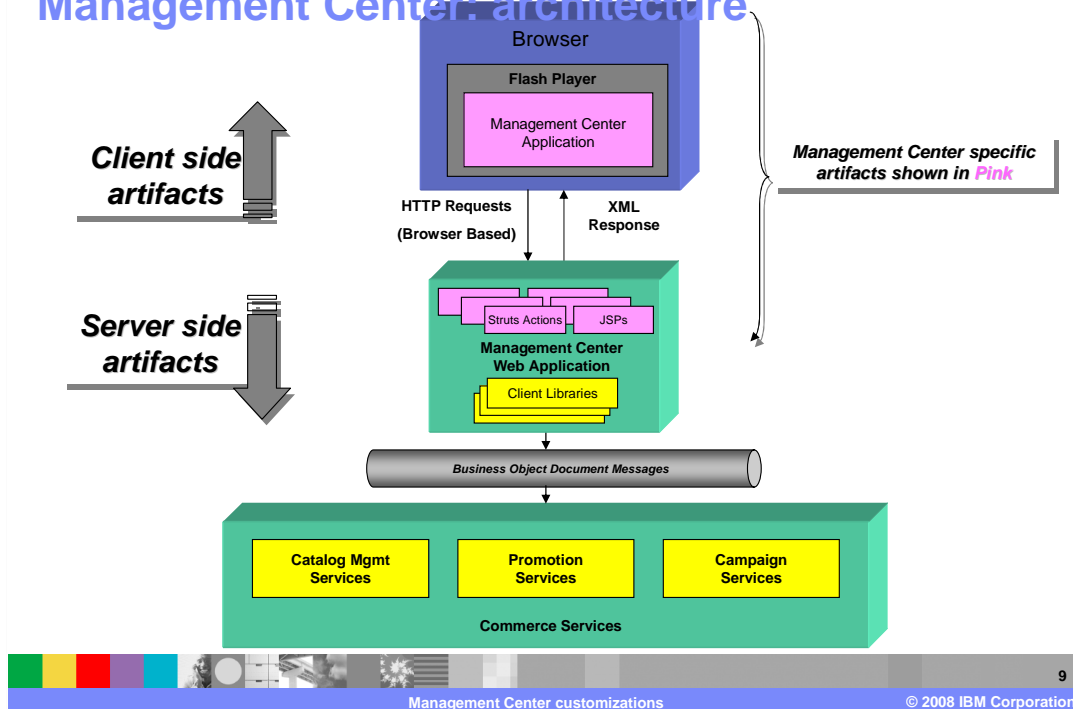
← Child view



This example demonstrates what OpenLaszlo source code looks like. At the top of the file a new class named `wcfWindow` is declared with `basewindow` as its superclass. The class declares some attributes that can be specified when an instance of the class is instantiated. There is a method that includes embedded JavaScript followed by a child view named `content` that is instantiated with the parent.

OpenLaszlo has a hierarchical design that is different from other object oriented languages. All nodes and views are part of this hierarchy. The root node is called the canvas. Within the hierarchy there are visible and non-visible nodes. The visible nodes are called views.

Management Center: architecture



This diagram shows a high level architecture of the Management Center.

The Management Center is a Flash application that runs on the browser's Flash player. In Feature Pack 3, Management Center supports both Internet Explorer V7 and Firefox V2 with Flash player V9 or above.

The Management Center client application communicates with the Management Center Web application using browser based HTTP requests. These requests are mapped to Struts actions in the Web application which then calls the various component services within WebSphere Commerce. The responses back to the client represent the client business objects in an XML format.

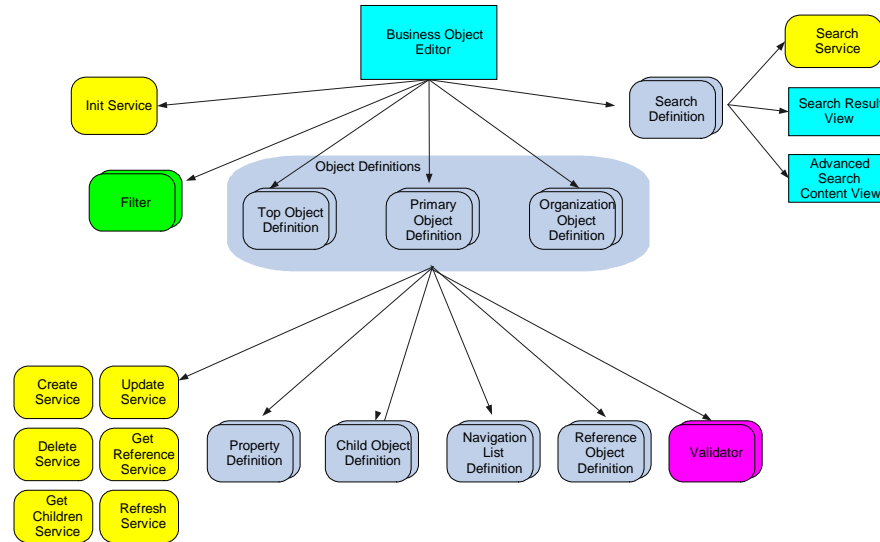
Section

Key concepts



This section covers some key concepts and reviews some code from the catalog component.

Management Center UI classes associations



This diagram shows the high level classes that make up the Management Center tool and how they relate to each other. These are the main classes you need to be familiar with when creating or updating a tool. All tools wanting to make use of the Management Center framework for managing business objects must extend the business object editor class. The business object editor class defines all the business objects that the tool manages.

From left to right, the init service is used to populate context values such as storeId and locale. The filters describe what objects are displayed in the explorer. At least one filter is required. In the middle are the object definitions. The top object definition is used on the client side only as the root node of the explorer tree. Primary objects represent stand-alone business objects such as product and catalog. Primary objects are searchable and can be referenced by other primary objects. Organization objects are used on the client side only for organizing the display of primary objects in the explorer. An example of an organization object is the Campaigns folder in the marketing tool. On the far right, the search definitions define the search services and results pages for basic and advanced searches.

On the left, under the primary object definition, are the services used by the object to perform basic create, update and delete operations and to access its children and other objects that reference the primary object. Merchandising associations are one example of a reference. The refresh service updates the local object from the server to pick up changes made by other users. Property definitions are used to define information that must be automatically validated by validators when objects are saved. Child object definitions represent objects that only exist in the context of their parent, such as a product description. The navigation list definition controls the display of the child object list in the main work area. Reference object definitions are another form of a child object definition which reference another primary object. Finally, validators can be used in association with

Key management center foundation OpenLaszlo classes

- **wcfBusinessObjectEditor**
 - ▶ Base class that all Management Center tools must extend
 - ▶ Includes support for menu, toolbar, search widget, navigation view and utilities view
 - ▶ Manages all interactions within the tool
 - ▶ Contains the definitions for all business objects managed by the business object editor

- **wcfObjectDefinition**
 - ▶ Base class for all object definitions
 - ▶ Describes a business object and its properties
 - ▶ Declares the services to be called for create, update, delete
 - ▶ Describes children objects and how to go about displaying them



The next three slides will describe some of the key OpenLaszlo classes in more detail.

The `wcfBusinessObjectEditor` class is the base class that all Management Center tools must extend. The business object editor includes support for the menu, toolbar, search widget, navigation view and utility view. It is responsible for managing all user interactions that allow you to edit the business objects declared with this business object editor.

`wcfObjectDefinition` is the base class of all object definition classes. An object definition is used to describe the characteristics of Management Center objects of a particular object type. An object definition also defines the services and actions available to the object type, such as the creation, update and deletion of objects, in addition to custom services and actions. Object definitions can be declared either as children of an instance of [wcfBusinessObjectEditor](#), or as children of another instance of `wcfObjectDefinition`.

Key management center foundation OpenLaszlo classes (continued)

- **wcfService**
 - ▶ Base class for all service types (init, create, update, delete, refresh, getChildren, getReference, search)
 - ▶ Responsible for invoking the corresponding component service request
- **wcfObjectGrid**
 - ▶ View that displays a list of business objects in table format with edit in place capabilities
- **wcfObjectProperties**
 - ▶ View that displays a business object and allows users to update all of its properties



The `wcfService` class is the base class for all services, it is not instantiated directly. Service classes define the URL and parameters for making a service request to the server. There are several types of services that can be defined: create, update, delete, refresh, get-children, search, init, and get reference.

`wcfObjectGrid` is the base class for all list views. Subclasses of this class are used for displaying a list of business objects in table format with quick edit in place capabilities. Each subclass defines the columns that appear in the list.

`wcfObjectProperties` is the base class for all properties views. Subclasses of this class are used to display a primary business object for creation and update. The class organizes the properties of an object using tabs and groups to make it easy for users to enter the information about the business objects.

Types of object definitions

- **wcfPrimaryObjectDefinition**
 - ▶ Base class for defining top level objects
 - ▶ Objects can be searched and referenced by other primary objects
- **wcfChildObjectDefinition**
 - ▶ Base class for defining child objects that are owned by primary objects
- **wcfReferenceObjectDefinition**
 - ▶ Base class for defining a relationship between two primary business objects
- **wcfParentReferenceObjectDefinition**
 - ▶ Special type of reference object that represents a parent/child relationship between two primary objects



These are several types of object definitions.

The `wcfPrimaryObjectDefinition` class contains the definition for a primary object. A primary object definition describes a top level business object such as a product. They are unique from other object definition types in that primary objects can be found using a search service and can be referenced by other primary objects. Primary object definitions are declared as direct child nodes of an instance of `wcfBusinessObjectEditor`. This allows the business object editor to manage the primary objects.

The `wcfChildObjectDefinition` class defines a child object. A child object definition describes a secondary business object that is owned by a primary object or another child object. Child object definitions are declared as direct child nodes of an instance of `wcfPrimaryObjectDefinition`, or as the child nodes of another `wcfChildObjectDefinition`. For example, a product description is defined as a child object, since the description is dependent on the product. When the product is deleted, the description is also be deleted. Child objects do not refer to the hierarchical relationship between the primary objects.

The `wcfReferenceObjectDefinition` class creates a reference object definition. It is used to describe a relationship between two primary objects such as a merchandising association.

Finally, the `wcfParentReferenceObjectDefinition` class defines a parent reference object definition. A parent reference object is a specialized type of `wcfReferenceObjectDefinition` reference object that is used to model a parent/child relationship between two primary objects. The child in this relationship must have a single parent. For example, a catalog `wcfCatalog` has a single master catalog category as its parent. Page 14 of 29

Sample catalog tool business object editor

```

<class name="catCatalogManagement" extends="wcfBusinessObjectEditor" helpLink="concepts/cpncatalogstool.htm">
...
  <!-- Init service to set the masterCatalogId-->
  <catCatalogInitService/>

  <!-- Filter definitions -->
  <catMasterCatalogFilter/>
  <catMasterCatalogGroupsFilter/>
  <catSalesCatalogGroupsFilter/>

  <!-- Object definitions -->
  <catCatalogTopObjectDefinition/>
  <catCatalogPrimaryObjectDefinition/>
  <catCatalogGroupPrimaryObjectDefinition/>
  <catProductPrimaryObjectDefinition/>
  <catBundlePrimaryObjectDefinition/>
  <catKitPrimaryObjectDefinition/>
  ...

  <!-- Search definitions -->
  <catFindKitsSearchDefinition/>
  <catFindBundlesSearchDefinition/>
  <catFindSKUsSearchDefinition/>
  ...
</class>

```

15

Management Center customizations

© 2008 IBM Corporation

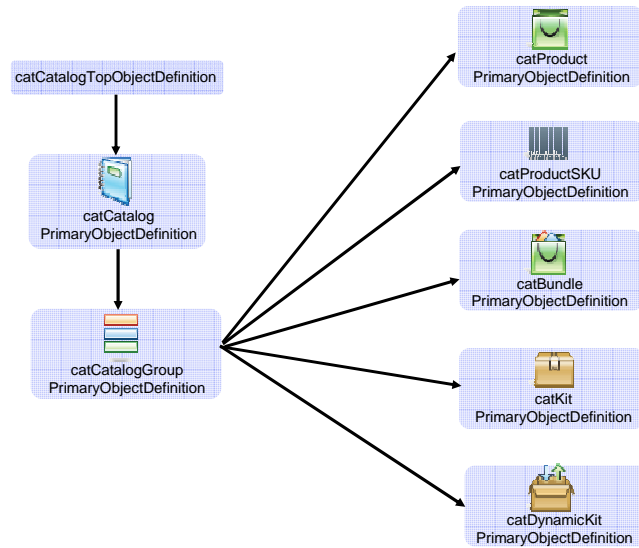
This slide shows sample OpenLaszlo code of how the catalog tool defines its `wcfBusinessObjectEditor` class. When the `catCatalogManagement` class is instantiated, the child nodes are instantiated with it. It includes an initialization service, `catCatalogInitService`, which is defined in another file and extends `wcfInitService`. The business object editor instance invokes the initialization service to load the master catalog ID.

There are three filters: `catMasterCatalogFilter`, `catMasterCatalogGroupsFilter`, and `catSalesCatalogGroupsFilter`. These filters are used by the business object editor to populate the filter selection drop-down in the explorer view. If there is only one filter, then the filter selector is not displayed. The filter selector is used to control which objects are visible in the explorer view.

The next section is the object definition. There are six object definitions shown here which are discussed on the next slide.

In the last section, there are three search definitions. The search definitions are used to populate the search drop-down menu in the top right corner of the business object editor.

Catalog tool business objects



16

Management Center customizations

© 2008 IBM Corporation

The objects shown here are all declared directly under the business object editor. The hierarchy is represented through reference objects which contain the primary object definitions.

The `catCatalogTopObjectDefinition` is declared as a convenience object to create a root explorer view node. It has no visual representation. When this object is instantiated, its `get-children` service is called and loads the catalog object, which is represented as an instance of the `catCatalogPrimaryObjectDefinition` class. The child objects of the catalog are the catalog group primary objects. The catalog group child objects are the product, sku, bundle, and kit primary objects.

Defining the explorer view content

This filter drop-down menu is derived from the filter definitions under the tool's business object editor.

This results from the objects returned after the `getChildren` services of the top object definition. This is a "Catalog" primary object defined as a `wcfPrimaryObjectDefinition`.

Note that there are no filters.

Each of these folders is an organization object. Organization objects are UI objects that do not have a server equivalent object. These objects can specify a `getChildrenService` to retrieve the related objects when you click on the folder.

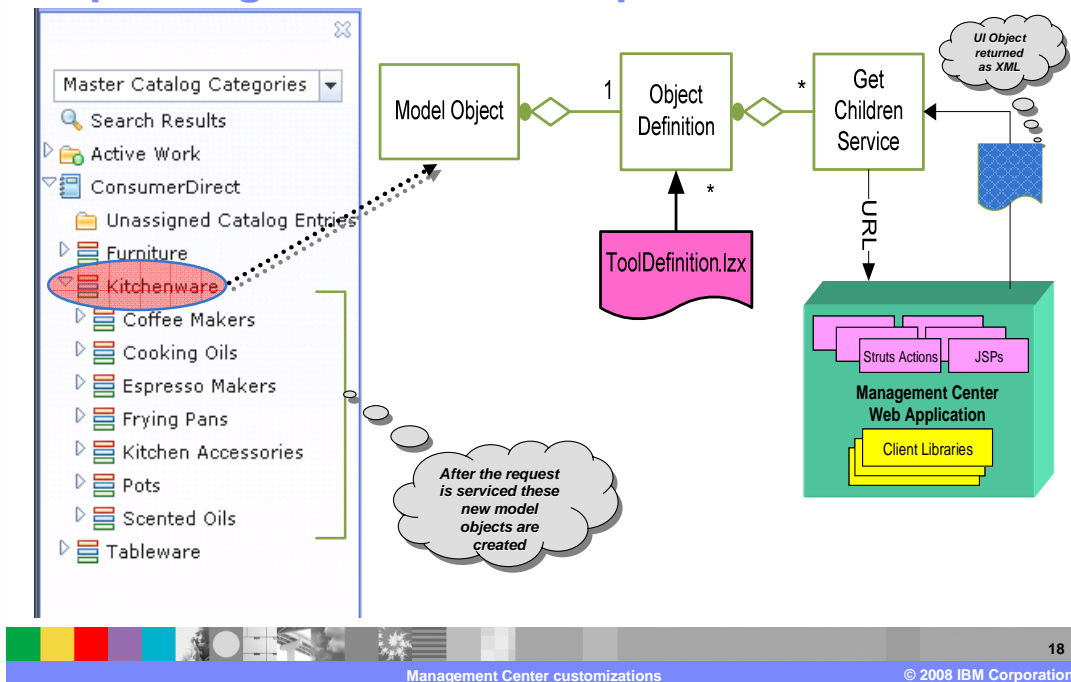
17
Management Center customizations © 2008 IBM Corporation

This slide shows examples of two explorer views, the catalog tool explorer view on the top and the marketing explorer view on the bottom. The type of objects managed in the two tools are different; therefore there are variations in the content of the explorer views as a result. The Management Center explorer tree has two predefined nodes: search results and active work. All other items are a result of the definitions in the business object editor class.

In the catalog explorer, the filter objects drop-down menu is derived from the filter definitions under the tool's business object editor. There are three predefined filters for the catalog tool: Master catalog, Master catalog categories and Sales catalog categories. The filter has no impact on the active work area. It only affects the browse area. For the catalog explorer, the browse area is populated by the top object definition calling the `getChildren` service to retrieve the catalog primary object as the root node.

The marketing explorer has two significant differences. First, there is no filter drop-down menu. The marketing tool does have a single filter defined. All tools must have at least one. In this case, the filter displays all objects. and since there are no other options to select from the filter drop-down menu, this menu is not displayed. The second difference is in the browse area. Rather than retrieve a primary object to be the root node, the top object definition defines organizational objects to be the highest level tree nodes. Organizational objects have no server side representation but they can have `getChildren` services in order to retrieve the primary objects from the server. Expanding one of these top level folders causes the organization object to call its `getChildren` service to retrieve its distinct primary objects such as campaigns, activities, e-mail templates, or e-marketing spots.

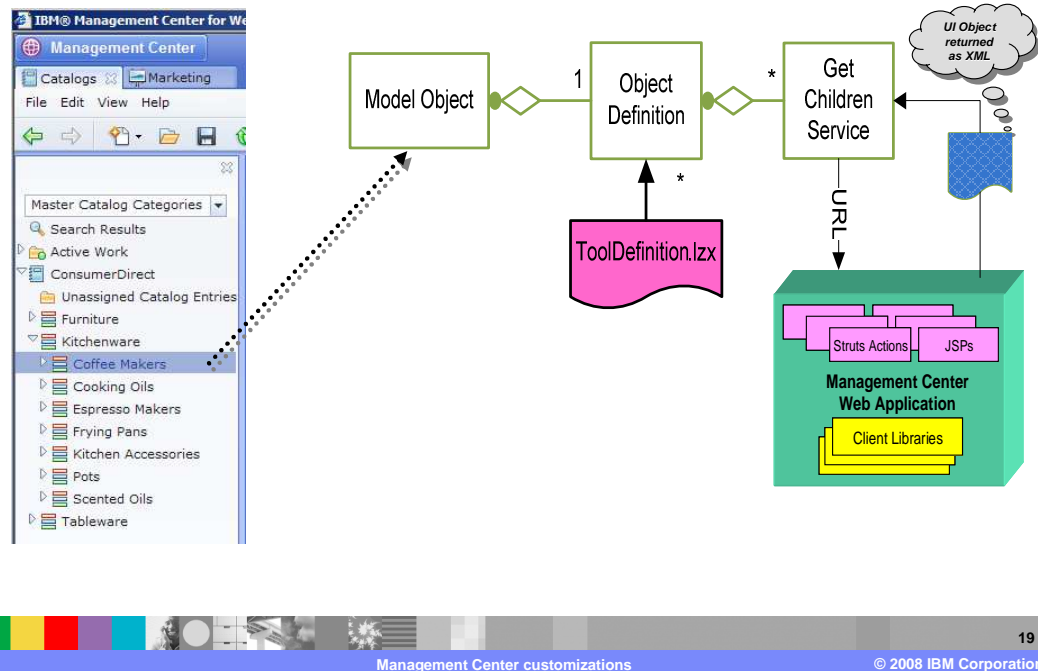
Expanding a node in the explorer view



Each business object is represented by a model object in the Management Center framework. When you click on a node to expand it, the framework uses the model object to locate the object definition. The framework then calls all the get-children services defined in the object definition.

The results of the get-children service requests are returned from the server in XML format. The framework then creates model objects for all the retrieved business objects and caches them. Once the model objects have been created, they are shown in the expanded explorer view tree.

Selecting a node in the explorer view



In Management Center, selecting a node in the explorer view has a different meaning than expanding a node. As seen in the previous slide, expanding a node causes Management Center to load its children and display them in the explorer view. Selecting a node, shown here, causes the children to be displayed as a list view in the work area.

When you click on a node, the Management Center framework still uses the model object to locate the object definition and invoke the get-children service. As before, the results of the get-children service requests are returned as XML. How the framework processes the response is shown on the next slide.

Selecting a node in the explorer view

The screenshot shows the IBM Management Center interface. The main window displays a table titled "Coffee Makers - Catalog Entries List". The table has columns for Display Sequence, *Type, *Code, Name, and Thumbnail. Two entries are visible:

Display Sequence	*Type	*Code	Name	Thumbnail
0.0		KICOF-01	Brewmaster Deluxe Coffeemaker	
2.0		KICOF-02	Coffee Maker/Grinder 3000	

A thought bubble points to the list view with the text: "Results of the get-children service request plus the 'default' navigation list definition are used to render the list view."

The UML class diagram below shows the following relationships:

- Model Object** (1) is associated with **Object Definition** (1).
- Object Definition** (*) is associated with **Navigation List Definition** (*).
- ToolDefinition.lzx** (pink box) has an association with **Object Definition** (*).

Once the server returns the objects to be displayed as an XML document, the framework creates model objects for all the retrieved objects and caches them.

In the object definition of the selected object in the tree, there is also a navigation list definition. This definition specifies which class should be used to create the default list view in the work area. The objects returned from the server are then displayed in the list view.

Defining search types for the find area

The available search types are derived from the search definitions under the tool's business object editor.

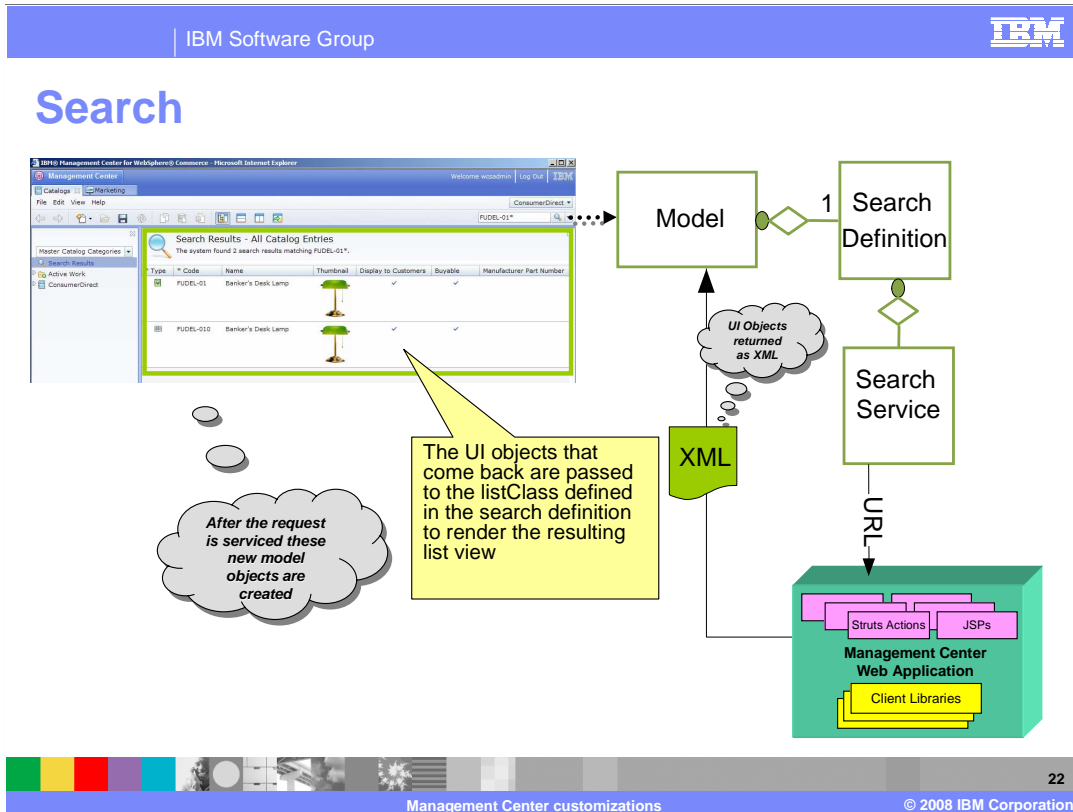
Advanced search appears if any of the search definitions are defined as such.

Having multiple advanced search definitions declared in the business object editor means that the advanced search dialog has multiple tabs.

21
Management Center customizations © 2008 IBM Corporation

Each tool can specify many different search types. You can add additional search types.

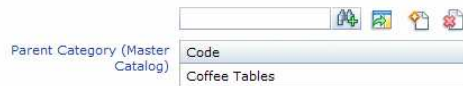
The number of search types available in the find area is determined by the number of `wcfSearchDefinition` classes defined in the business object editor extension class of the tool. Every search definition results in a search type option appearing in the list. Search definitions that also define the `advancedSearchContentClass` optional attribute display the Advanced Search option. If more than one search definition class has the `advancedSearchContentClass` attribute then each type of advanced search appears in a separate tab on the advanced search dialog.



The following diagram shows how the search function works in the Management Center.

When you type an entry in the search field and click on the search button, the Management Center framework finds the search definition for the selected search type. The framework then calls the search service URL defined in the definition with all its parameters together with the text entered in the search field. Once the search results come back from the server, the class defined as the listClass of the search definition is used to render the resulting business objects.

Reference editor widget



Parent Category (Master Catalog)

Code
Coffee Tables

- Appears only in the object properties view
- Used to create a reference such as a parent/child relationship
- Contains a search field for a quick object lookup (and add if the result is singular)
- Add and delete capability



A reference editor is used to provide support for editing the relationship between two primary objects. The reference editor is declared as part of the properties view of one primary object and allows you to select another primary object. A search field is provided for quick lookup.

A reference editor can be used to change the parent primary object. In general, it can be used to create or edit any one-to-one relationship with another primary object. If the relationship is one-to-many then a child list editor must be used.

Child list editor widget

Defining Attributes

Display Sequence (United State...	* Display Name (United States English)	* Data Type	Description (United States English)
3.00	Assortment	Text	The desired assortment of plates.
2.00	Serving Size	Whole number	The desired serving size.
1.00	Trim Color	Text	The color of the plate trim.

1 to 3 of 3

- Appears only in the object properties view
- Used to create multiple references between two primary objects
- Add and delete rows capability
- (Optional) Contains a search field for a quick object lookup and add



A child list editor is a widget that allows the child of a particular object to be edited. The child list editor provides the ability to create new child objects, delete child objects, or edit the existing child objects directly in the list. If there are no objects that can be created, then the button to create a new object does not appear on the editor.

The editor also contains a button that launches the utilities view and a search widget that adds results to the child list.

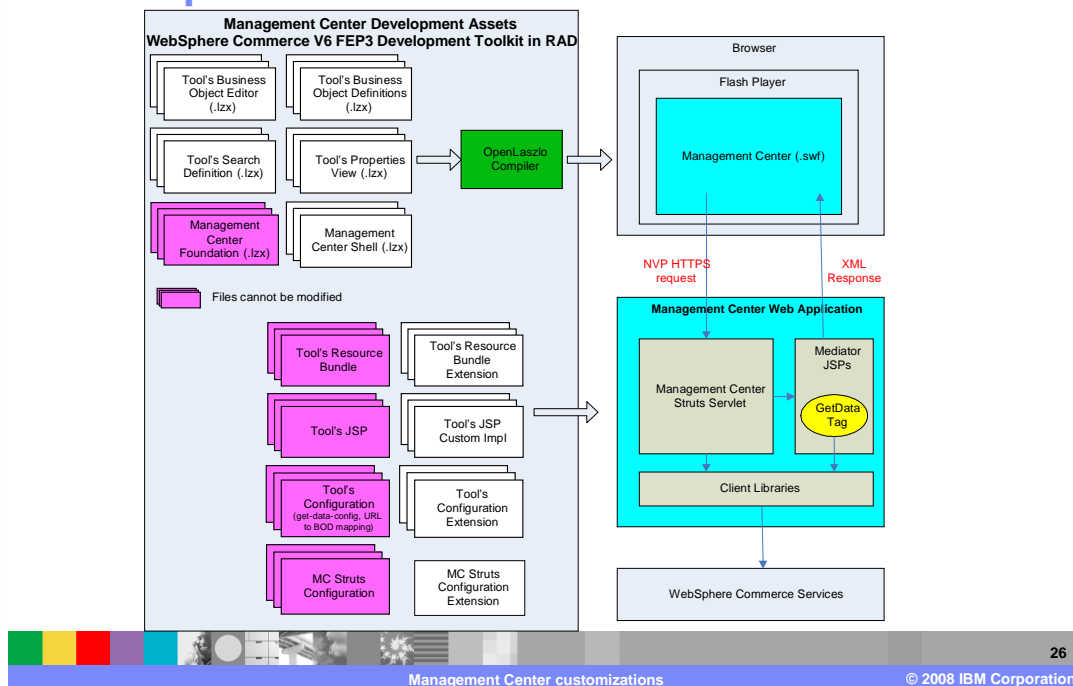
Section

Development flow and customization approach



The final section is on the recommended customization approach for the Management Center UI.

Development flow



26

Management Center customizations

© 2008 IBM Corporation

This diagram shows all the categories of development assets for the Management Center. The top group of assets relate to the Management Center client and have been discussed throughout this presentation. The bottom group of assets relate to the Management Center Web application and are covered in another presentation.

The Management Center client customization approach is fairly straightforward. The OpenLaszlo assets have been divided into two groups, customizable and restricted. Within the WebSphere Commerce Developer Toolkit, the OpenLaszlo files have also been organized by tool. You can add or change any assets shown in the white boxes. For example, in the catalog tool, you can modify part of the overall layout in the business object editor. You can also add or change any of the business object definitions, create a new search definition or update the properties view.

There are even a few customization points in the Management Center shell.

Once you've added or changed OpenLaszlo files, you need to recompile them to create a new Flash file. It's not necessary to restart the WebSphere Commerce server to see your changes.

On the Web application side, the customization approach is to make a copy of the file to be modified and then make changes as required. The Struts configuration files are used to point to your new version of the file. This topic is covered in more detail in the Web application presentation.

Development environment

The LOBTools project in WebSphere Commerce Developer, Feature Pack 3 has this file structure:

Development assets	Folders (within <LOBTools>/WebContent/)
Business object editor files for the tool	WEB-INF/src/lzx/commerce/<component>
Business object definitions	WEB-INF/src/lzx/commerce/<component>/objectDefinitions
Search definitions and related classes	WEB-INF/src/lzx/commerce/<component>/searchDefinitions
List views for the tool	WEB-INF/src/lzx/commerce/<component>/listViews
Properties view files of business objects	WEB-INF/src/lzx/commerce/<component>/propertiesViews
Open Laszlo Files that cannot be changed directly	WEB-INF/src/lzx/commerce/<component>/restricted
Messages bundles for the tool	src/com.ibm.commerce.<component>.client.lobtools.properties/ <component>LOB_en_US.properties
Customized messages bundles	src/extension/<component>/ <component>LOB_en_US.properties



This table shows how the files that make up the Management Center application are organized in the LOBTools project in the WebSphere Commerce Developer Toolkit. When you add new OpenLaszlo files, you should create a new directory under the lzx folder to hold your changes.

Feedback

Your feedback is valuable

You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject=Feedback_about_WCS6003_MCCustomizations.ppt

This module is also available in PDF format at: ..WCS6003_MCCustomizations.pdf



You can help improve the quality of IBM Education Assistant content by providing feedback.

Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM WebSphere

A current list of other IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

JavaScript, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

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

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

© Copyright International Business Machines Corporation 2008. All rights reserved.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.

