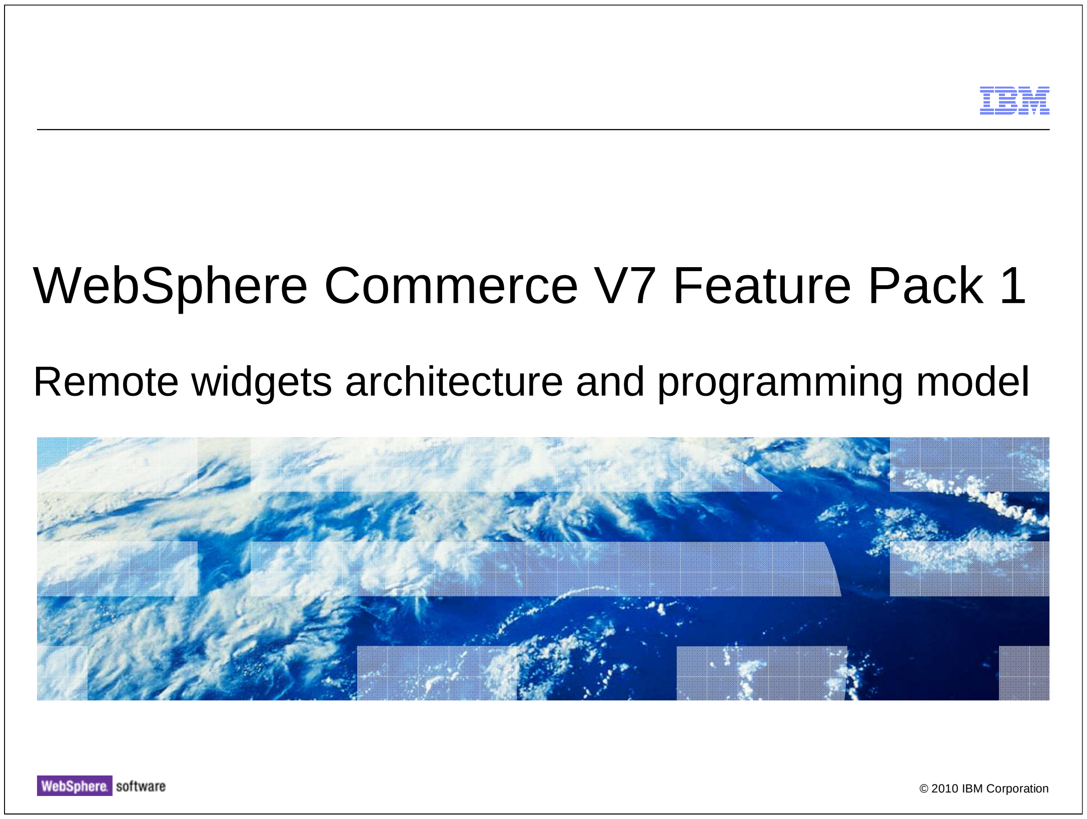
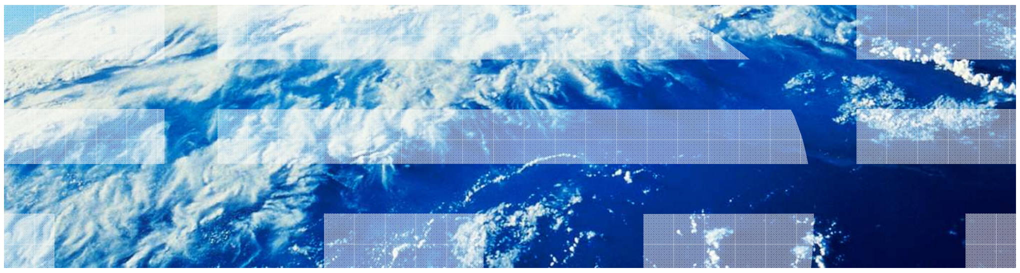


WebSphere Commerce V7 Feature Pack 1

Remote widgets architecture and programming model



WebSphere software

This presentation provides an introduction to the architecture and programming model of the remote widgets solution in WebSphere® Commerce V7.0 feature pack 1.

Goals

- To understand the architecture and service changes for the remote widgets solution
- To understand the customization options

At the end of this presentation, you should understand the architecture of the remote widgets solution including the new gift registry and wish list services. You should also understand the customization options available.

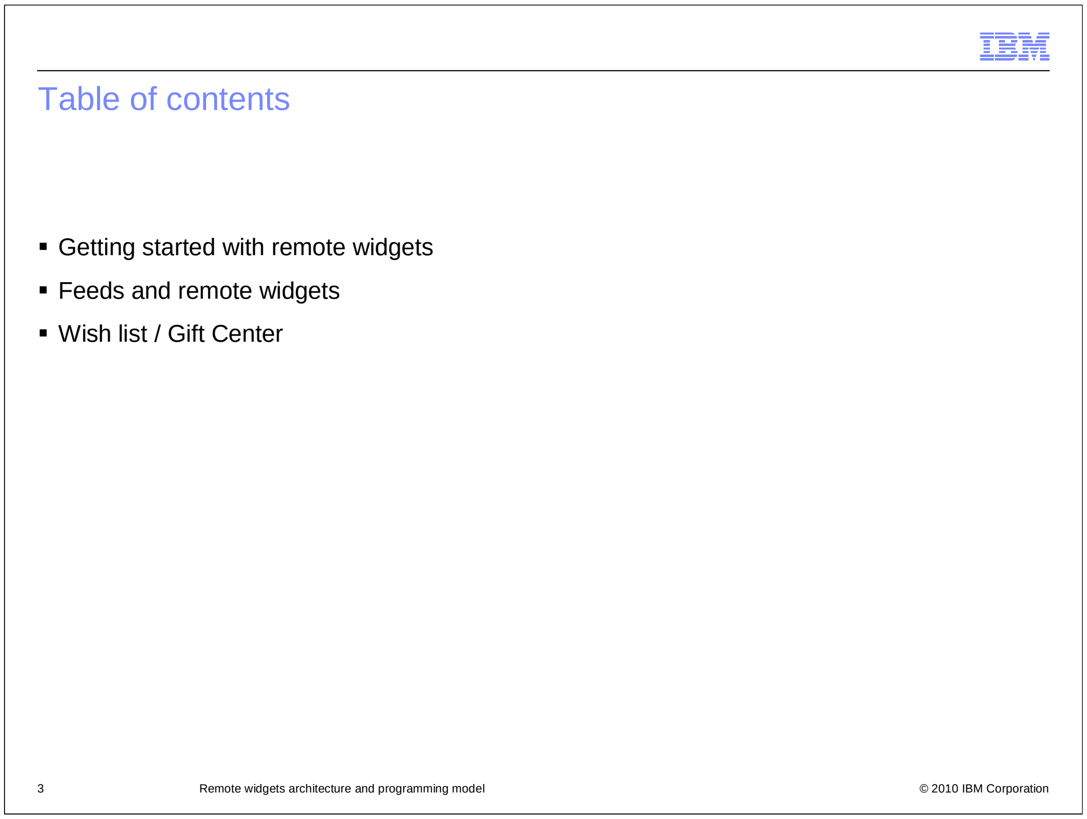
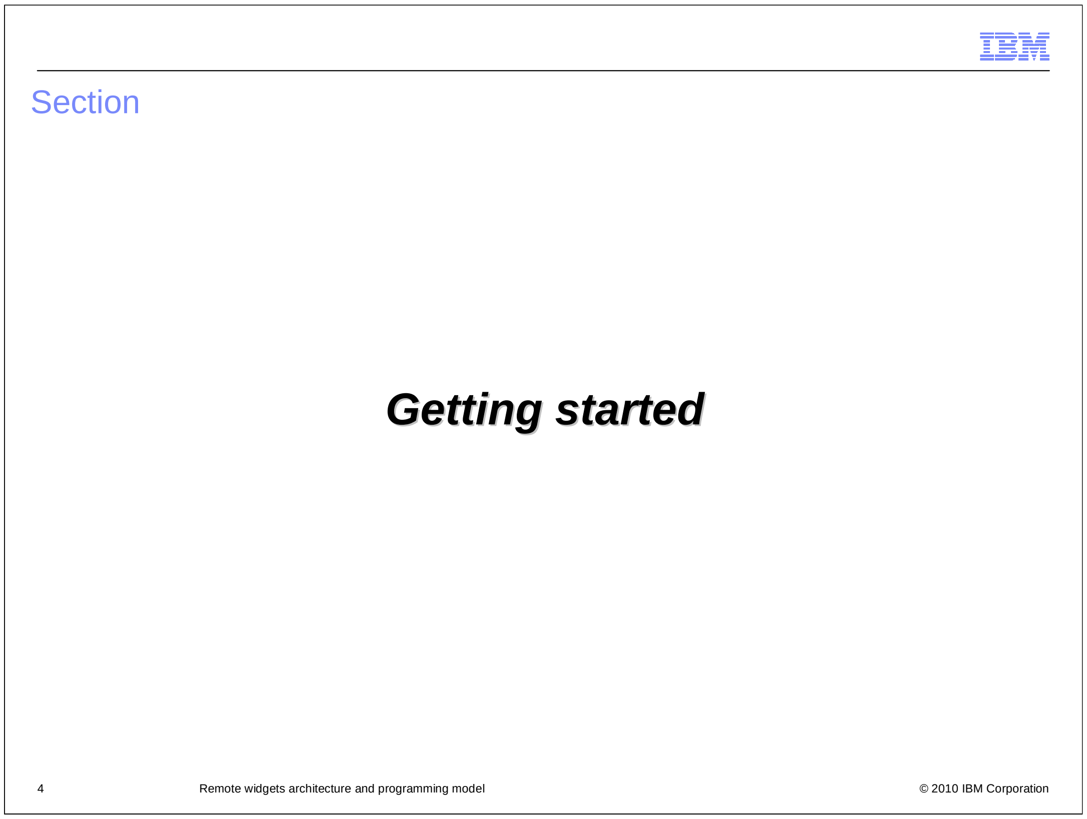


Table of contents

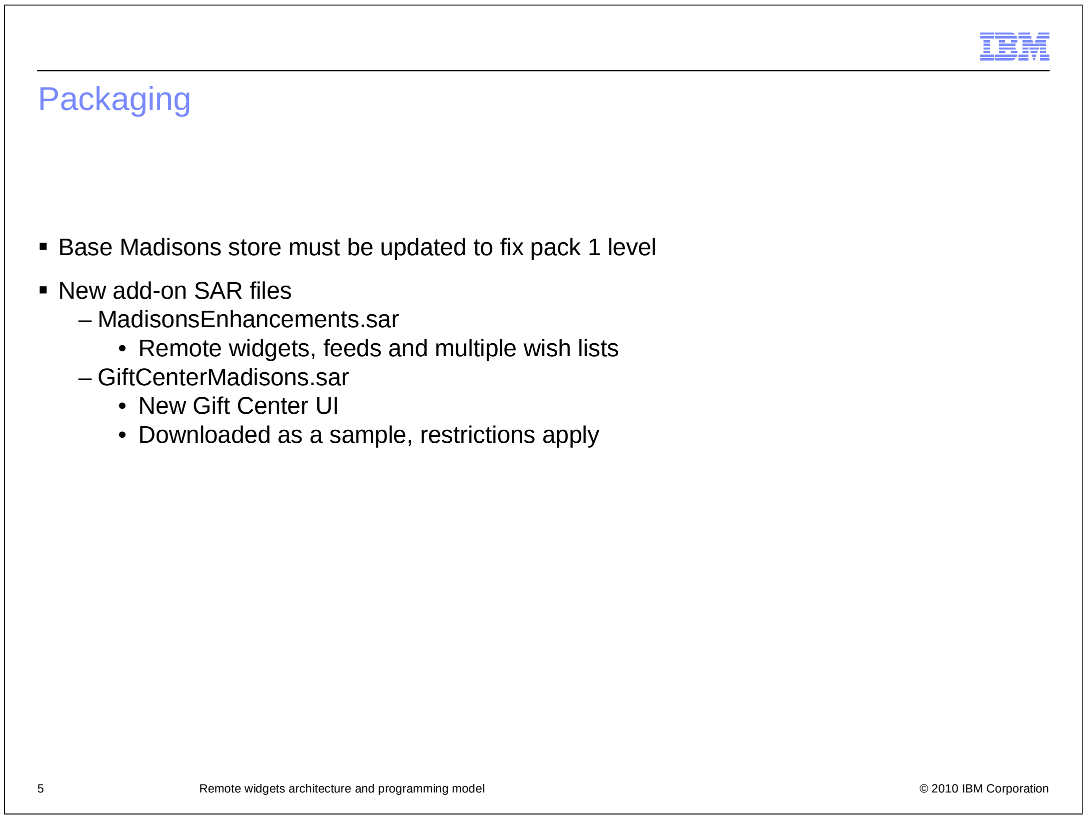
This presentation begins with an overview of the steps required to enable the remote widgets features. The second section focuses on the details of feeds and remote widgets. The presentation concludes with a look at the new wish list and gift center service implementation.



Section

Getting started

This section covers the steps for getting started with remote widgets and gift center.



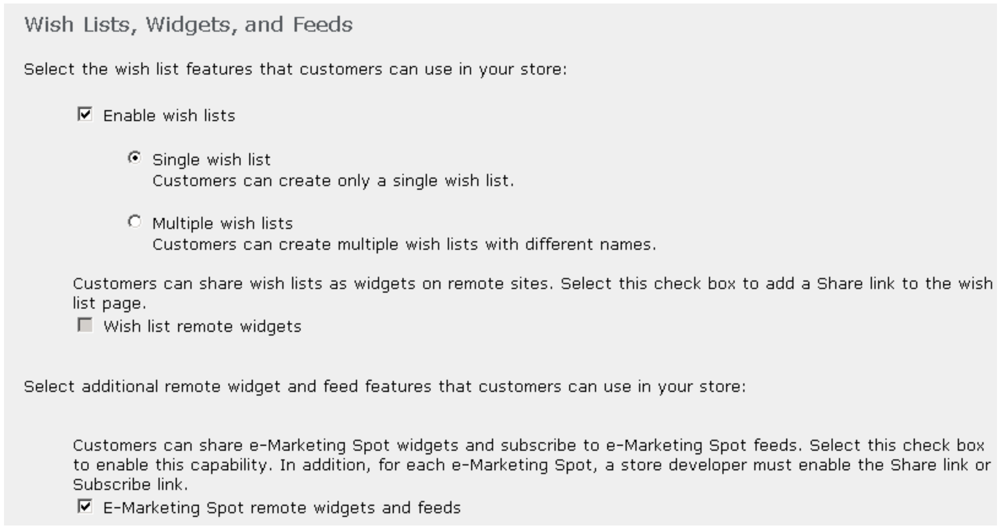
Packaging

- Base Madisons store must be updated to fix pack 1 level
- New add-on SAR files
  - MadisonsEnhancements.sar
    - Remote widgets, feeds and multiple wish lists
  - GiftCenterMadisons.sar
    - New Gift Center UI
    - Downloaded as a sample, restrictions apply

5 Remote widgets architecture and programming model © 2010 IBM Corporation

The store assets for the remote widgets feature are packaged in an add-on sar file called MadisonsEnhancements.sar. Once you have installed feature pack 1 and enabled the store-enhancements feature, you need to publish this sar file. A full list of required and optional steps is available in the Information Center.

New change flow panel - 1


Wish Lists, Widgets, and Feeds

Select the wish list features that customers can use in your store:

[✓] Enable wish lists

(●) Single wish list
Customers can create only a single wish list.

( ) Multiple wish lists
Customers can create multiple wish lists with different names.

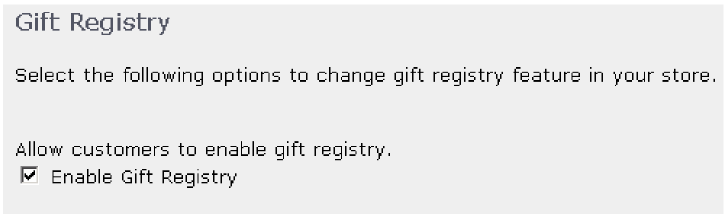
Customers can share wish lists as widgets on remote sites. Select this check box to add a Share link to the wish list page.
[ ] Wish list remote widgets

Select additional remote widget and feed features that customers can use in your store:

Customers can share e-Marketing Spot widgets and subscribe to e-Marketing Spot feeds. Select this check box to enable this capability. In addition, for each e-Marketing Spot, a store developer must enable the Share link or Subscribe link.
[✓] E-Marketing Spot remote widgets and feeds

Once you have published the StoreEnhancements sar file a new panel is available in the Accelerator Change Flow notebook. The panel is called Wish Lists, Widgets, and Feeds. From here, you can configure which of the new features you want turned on in the store. This screen capture shows the default values. E-Marketing Spot remote widgets and feeds are turned on but multiple wish lists are turned off.

New change flow panel - 2

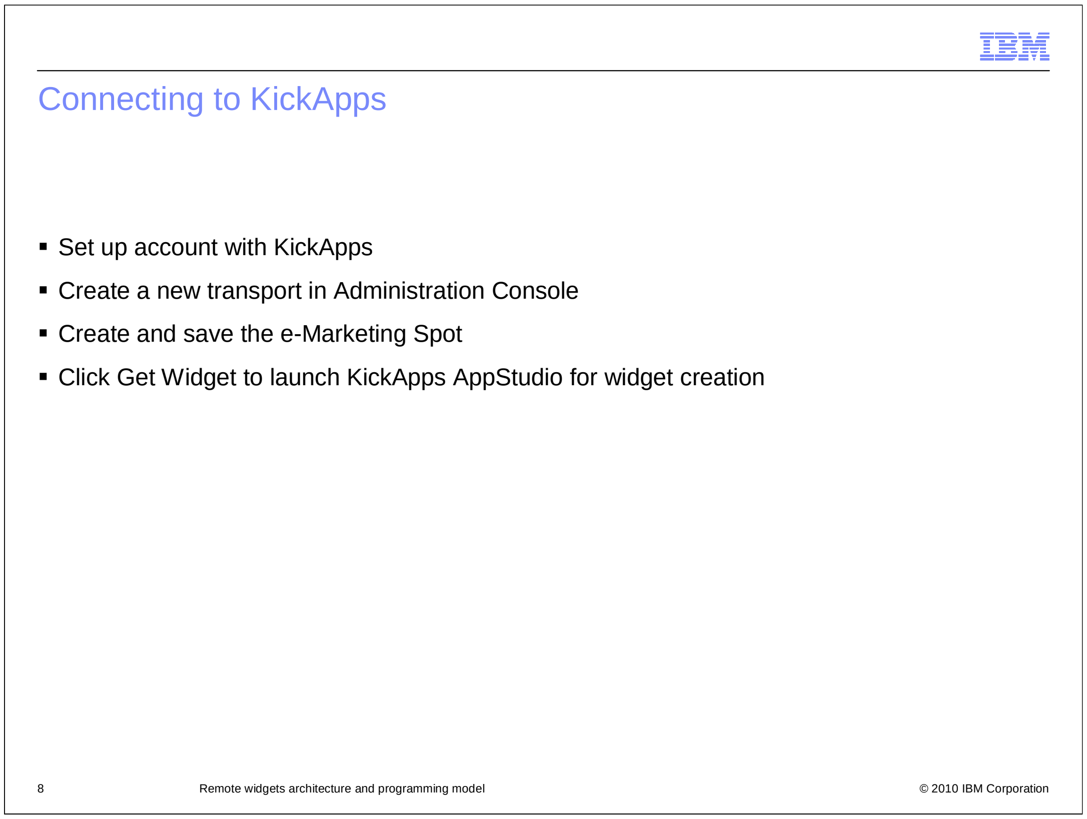

Gift Registry

Select the following options to change gift registry feature in your store.

Allow customers to enable gift registry.

☑ Enable Gift Registry

When you publish the GiftCenterMadisons sar file the new Gift Registry change flow panel is available in the Accelerator Change Flow notebook. The Gift Registry feature is enabled by default and you can use this panel to disable it.



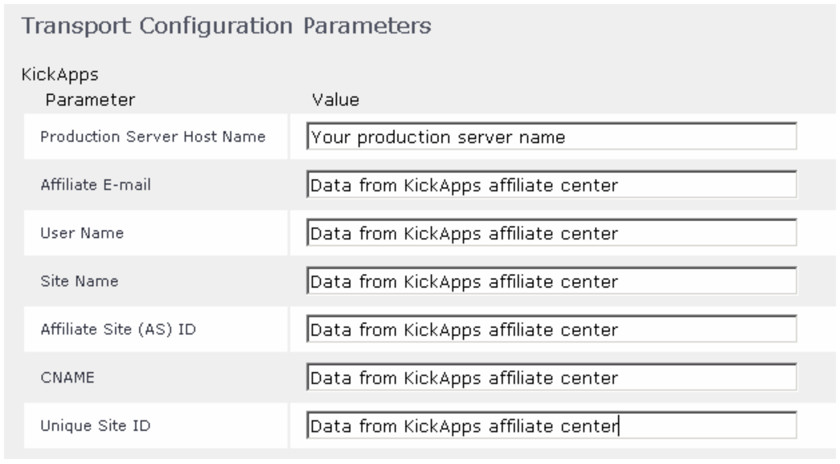
Connecting to KickApps

- Set up account with KickApps
- Create a new transport in Administration Console
- Create and save the e-Marketing Spot
- Click Get Widget to launch KickApps AppStudio for widget creation

8 Remote widgets architecture and programming model © 2010 IBM Corporation

If you choose to use the KickApps integration to build and deploy remote widgets, there are some steps required to set it up. The first thing is to complete your service agreement with KickApps and get all your account details. Once you have that information, you use it to configure a new transport in Administration Console. The transport is used to enable single sign on so you can launch KickApps directly from Management Center using the Get Widget button on any e-Marketing Spot properties page.

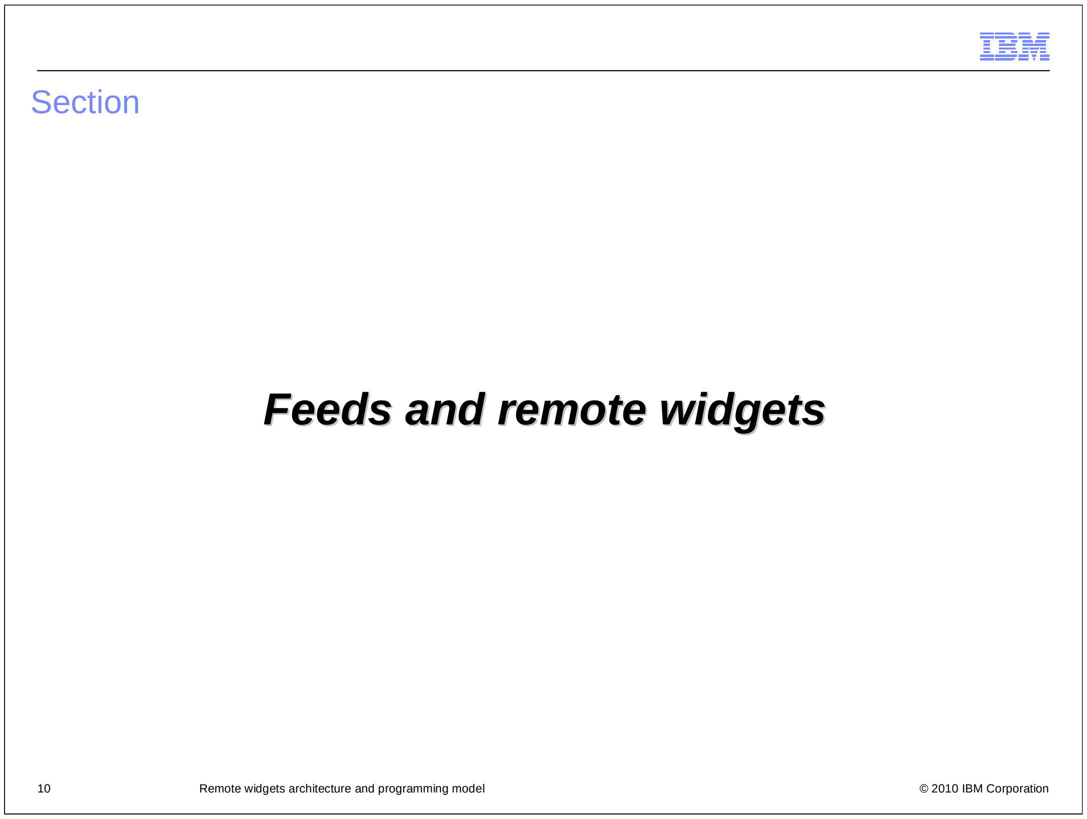
KickApps transport


Transport Configuration Parameters

KickApps

Parameter	Value
Production Server Host Name	Your production server name
Affiliate E-mail	Data from KickApps affiliate center
User Name	Data from KickApps affiliate center
Site Name	Data from KickApps affiliate center
Affiliate Site (AS) ID	Data from KickApps affiliate center
CNAME	Data from KickApps affiliate center
Unique Site ID	Data from KickApps affiliate center

This screen capture shows the data you need from KickApps to configure the transport. This transport must be configured to create remote widgets using Management Center and display a Share link next to a wish list or e-Marketing Spot in the store front.



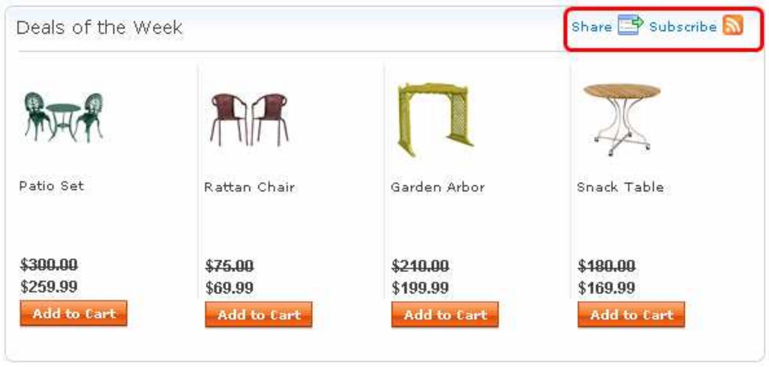
Section

Feeds and remote widgets

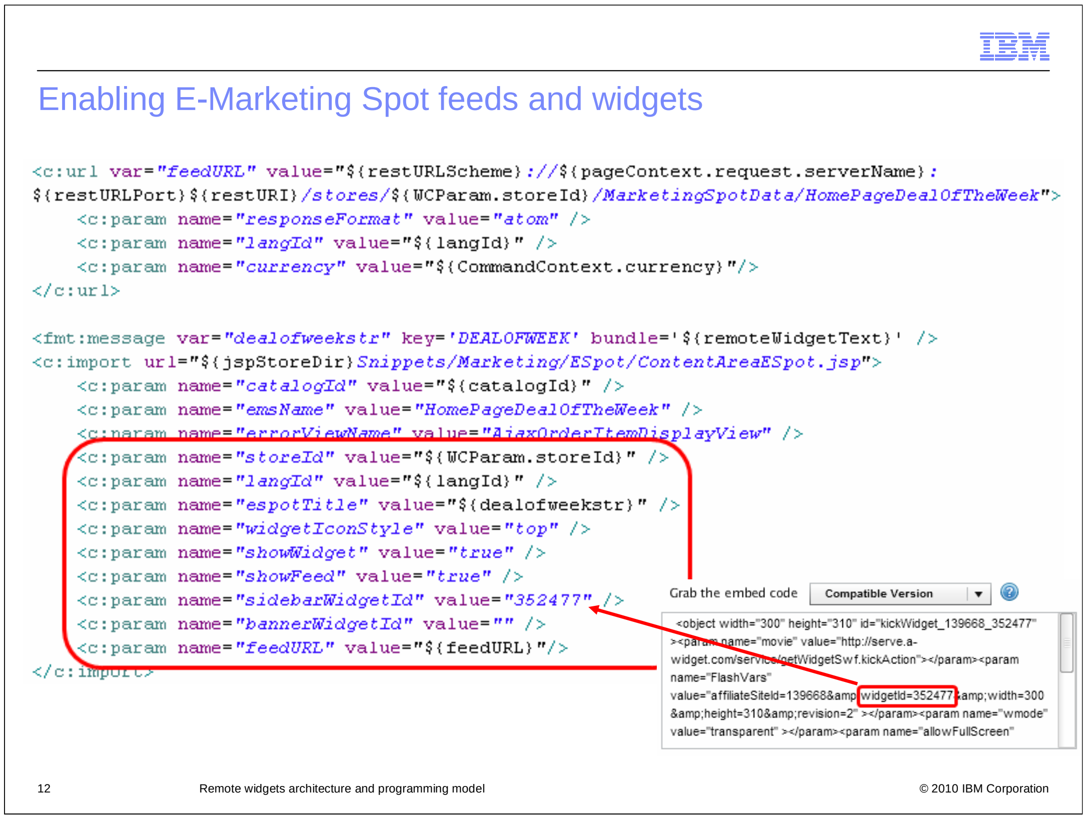
This section covers the details of feeds and remote widgets.

Adding Share and Subscribe links

- Add parameters to the <c:import> tag for the e-Marketing Spot
- Update RemoteWidgetButtons.jsp to change formatting



The Share and Subscribe links can be added to any e-Marketing Spot in the store by specifying some additional parameters in the import tag for the e-Marketing Spot. The file RemoteWidgetButtons.jsp is responsible for displaying the links. You can modify this file if you want to change how the links are displayed.



Enabling E-Marketing Spot feeds and widgets

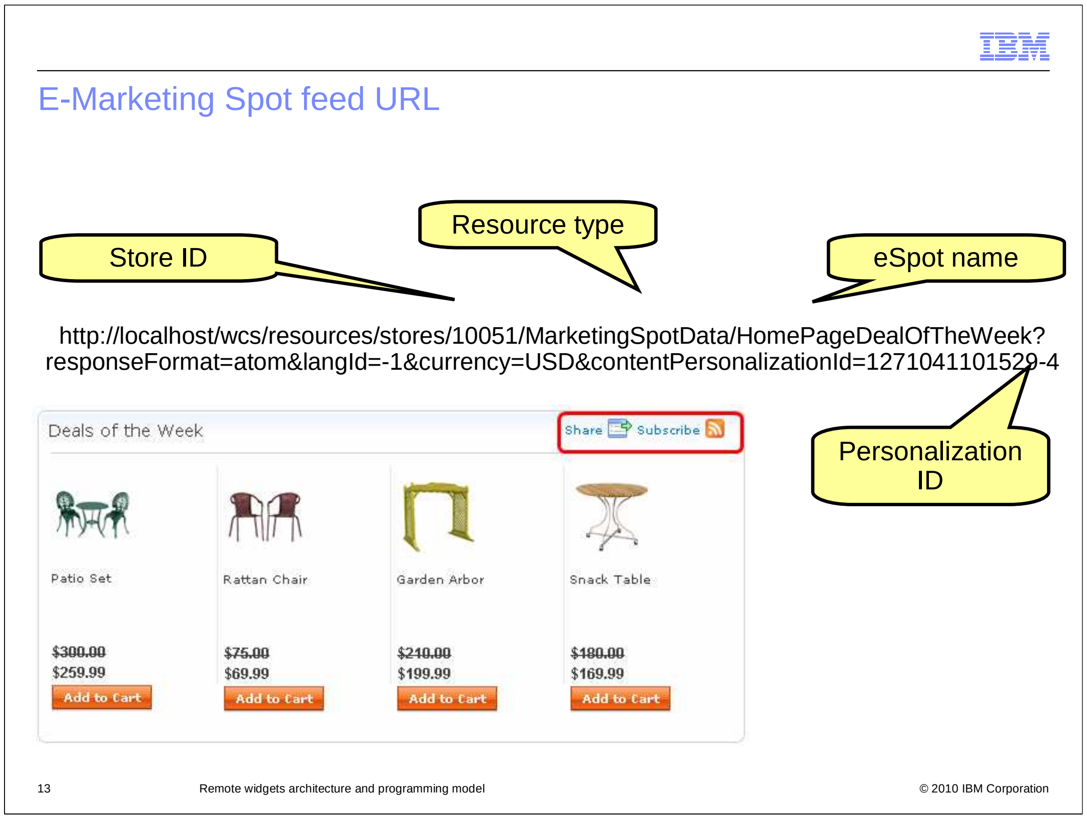
```
<c:url var="feedURL" value="${restURLScheme}://${pageContext.request.serverName}:
${restURLPort}${restURI}/stores/${WCParam.storeId}/MarketingSpotData/HomePageDealOfTheWeek">
    <c:param name="responseFormat" value="atom" />
    <c:param name="langId" value="${langId}" />
    <c:param name="currency" value="${CommandContext.currency}"/>
</c:url>

<fmt:message var="dealofweekstr" key='DEALOFWEEK' bundle='${remoteWidgetText}' />
<c:import url="${jspStoreDir}Snippets/Marketing/ESpot/ContentAreaESpot.jsp">
    <c:param name="catalogId" value="${catalogId}" />
    <c:param name="emsName" value="HomePageDealOfTheWeek" />
    <c:param name="errorViewName" value="AjaxOrderItemDisplayView" />
    <c:param name="storeId" value="${WCParam.storeId}" />
    <c:param name="langId" value="${langId}" />
    <c:param name="espotTitle" value="${dealofweekstr}" />
    <c:param name="widgetIconStyle" value="top" />
    <c:param name="showWidget" value="true" />
    <c:param name="showFeed" value="true" />
    <c:param name="sidebarWidgetId" value="352477" />
    <c:param name="bannerWidgetId" value="" />
    <c:param name="feedURL" value="${feedURL}"/>
</c:import>
```

Grab the embed code
Compatible Version

```
<object width="300" height="310" id="kickWidget_139668_352477"
><param name="movie" value="http://serve.a-
widget.com/service/getWidgetSwf.kickAction"></param><param
name="FlashVars"
value="affiliateSiteId=139668&amp;widgetId=352477&amp;width=300
&amp;height=310&amp;revision=2" ></param><param name="wmode"
value="transparent" ></param><param name="allowFullScreen"
```

12 Remote widgets architecture and programming model © 2010 IBM Corporation

This slide shows the <import> tag for an e-Marketing Spot with additional parameters for the Share and Subscribe links. The parameters showWidget and showFeed are the two that trigger the links to be displayed next to the e-Marketing Spot. In order to show the Share link, you also need to provide at least one widget ID and the KickApps transport must be configured. The feedURL parameter is discussed in more detail on the next slide.

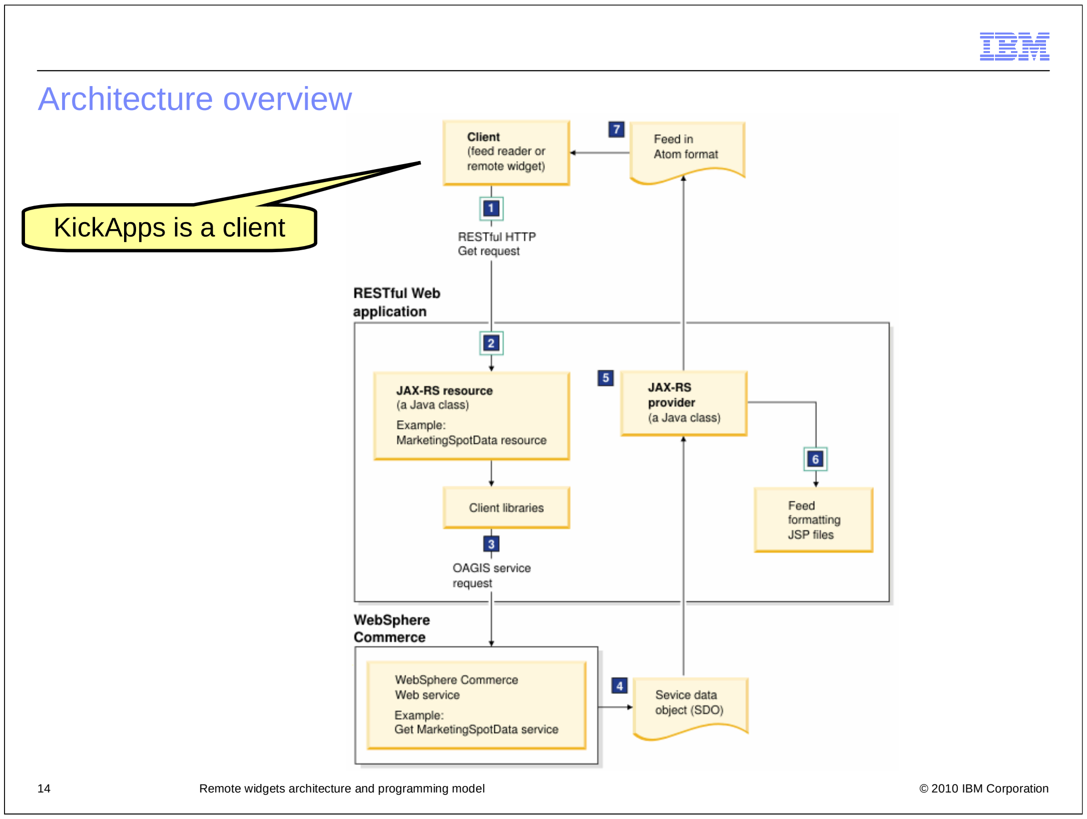


E-Marketing Spot feed URL

http://localhost/wcs/resources/stores/10051/MarketingSpotData/HomePageDealOfTheWeek?responseFormat=atom&langId=-1&currency=USD&contentPersonalizationId=1271041101529-4

13 Remote widgets architecture and programming model © 2010 IBM Corporation

This slide shows the URL for an e-Marketing spot feed. The same URL format is also used by remote widgets to access the content they display. The unique part of the URL starts with the storeId. Following that is the type of resource being requested, in this case MarketingSpotData, and an identifier specific to the resource type, in this case the e-Marketing Spot name. The response format used for all requests is atom. Following language and currency is a personalization id. This ID is used to identify whose data should be returned when a request is made from a remote widget or feed reader outside of WebSphere Commerce.



Architecture overview

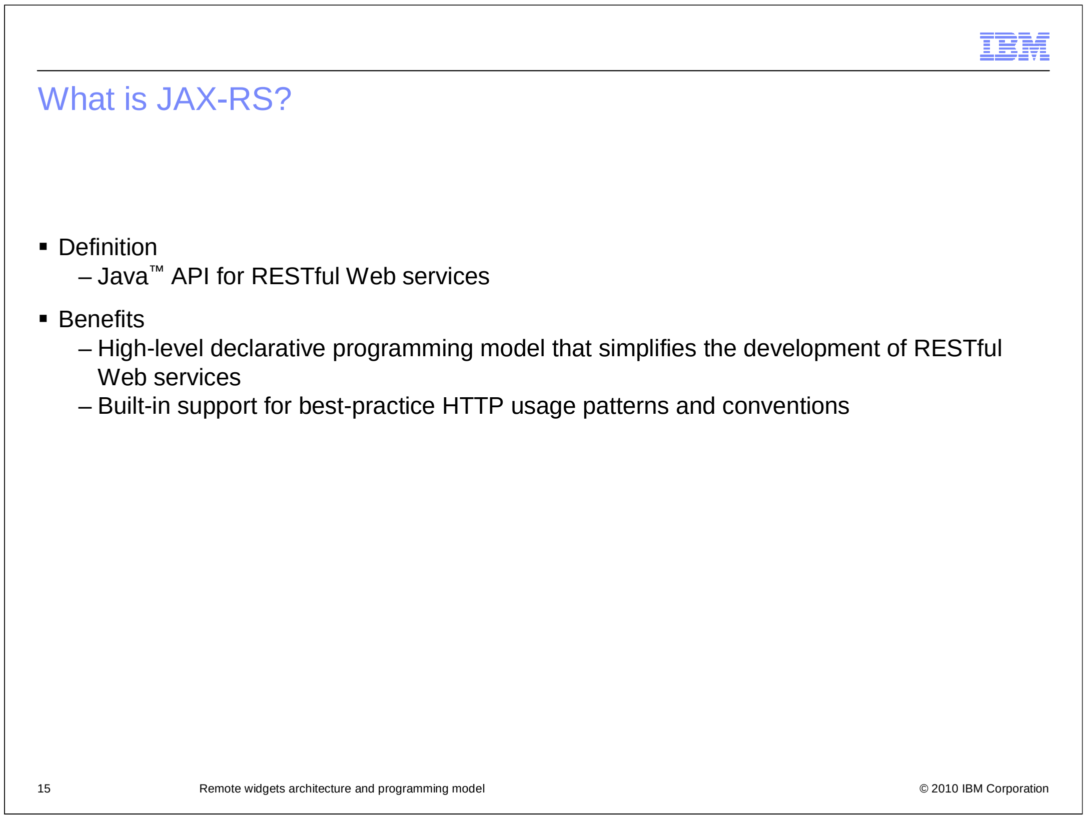
KickApps is a client
Client (feed reader or remote widget)
7
Feed in Atom format
1
RESTful HTTP Get request
RESTful Web application
2
JAX-RS resource (a Java class) Example: MarketingSpotData resource
5
JAX-RS provider (a Java class)
6
Feed formatting JSP files
Client libraries
3
OAGIS service request
WebSphere Commerce
WebSphere Commerce Web service Example: Get MarketingSpotData service
4
Sevice data object (SDO)

14 Remote widgets architecture and programming model © 2010 IBM Corporation

This diagram shows the flow of a remote widget or feed reader request through WebSphere Commerce. The request is initially received by a new RESTful Web application. The resource type specified in the URL is used to identify which JAX-RS resource should handle the request. The JAX-RS resource uses the remaining parameters and the WebSphere Commerce client libraries to initiate a standard OAGIS service request. The service request is processed by WebSphere Commerce the same as any other service request. This includes making use of cached responses. When the response is received, the JAX-RS runtime directs it to the appropriate JAX-RS provider based on the specified response format. The JAX-RS provider makes use of a set of feed formatting JSP files to convert the SDO to the specified response format, typically atom. The response is then returned to the client that initiated the request.

The blue boxes around steps one, two and six in this diagram represent customization points in the flow. They are the request URL, the JAX-RS resource and the response formatting JSP files. Customization will be discussed later in the presentation.

You can find this diagram and a detailed explanation of the flow in the Information Center.

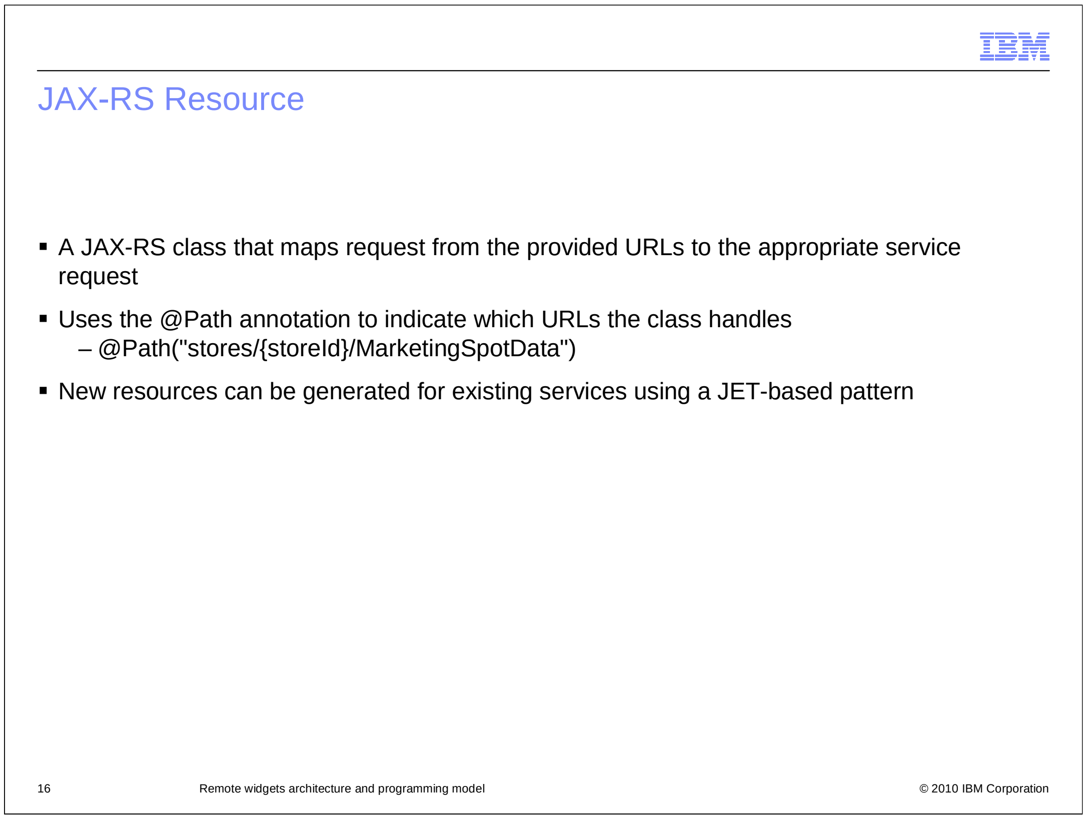


What is JAX-RS?

- Definition
  - Java™ API for RESTful Web services
- Benefits
  - High-level declarative programming model that simplifies the development of RESTful Web services
  - Built-in support for best-practice HTTP usage patterns and conventions

15 Remote widgets architecture and programming model © 2010 IBM Corporation

JAX-RS, the Java API for RESTful Web services, is used to simplify the development of new RESTful Web services. As described in the previous slide, the JAX-RS runtime handles the processing of incoming client requests and responses from the WebSphere Commerce server. The API also provides support for best practices.



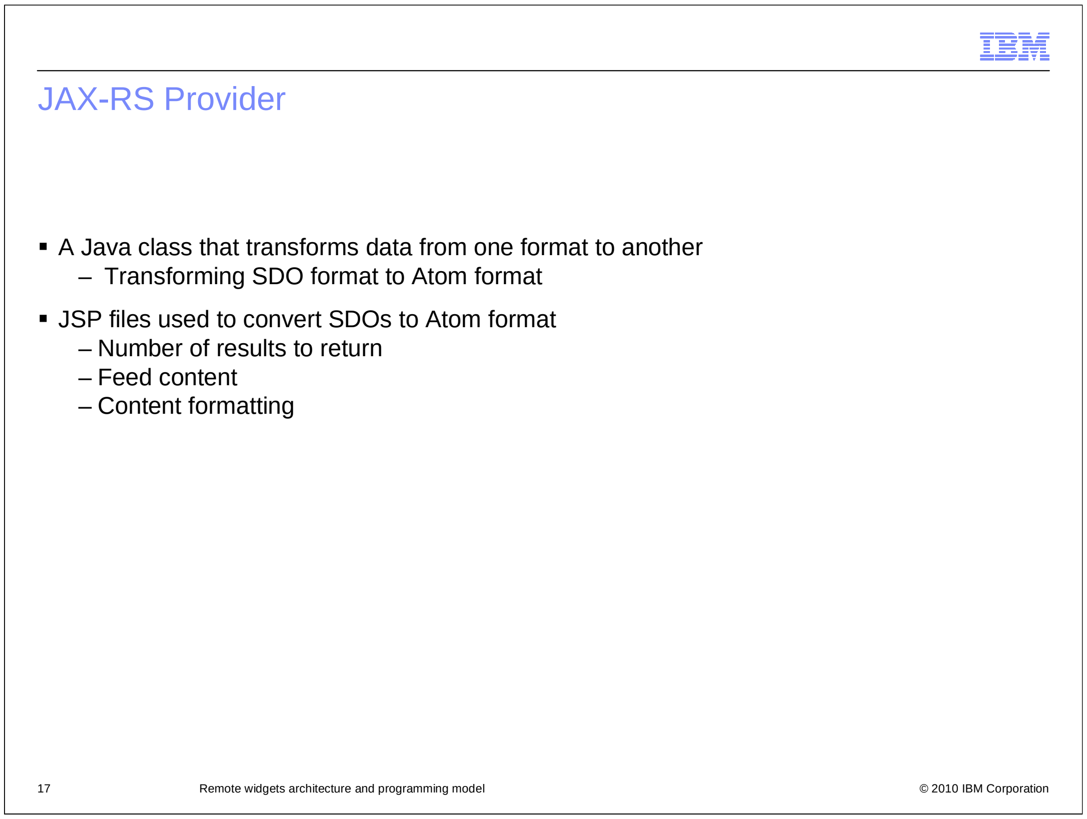
JAX-RS Resource

- A JAX-RS class that maps request from the provided URLs to the appropriate service request
- Uses the @Path annotation to indicate which URLs the class handles
  - – @Path("stores/{storeId}/MarketingSpotData")
- New resources can be generated for existing services using a JET-based pattern

16 Remote widgets architecture and programming model © 2010 IBM Corporation

A JAX-RS resource is a Java class that converts the RESTful Web service request sent by the client into an OAGIS service request recognized by the WebSphere Commerce server. The class uses the @Path annotation to identify which incoming service requests it can handle. Notice that the example shown here for the MarketSpotData resource and it matches the format of the e-Marketing Spot feed URL shown earlier.

New resources can be easily generated using a JET-based pattern. This process is described on an upcoming slide.

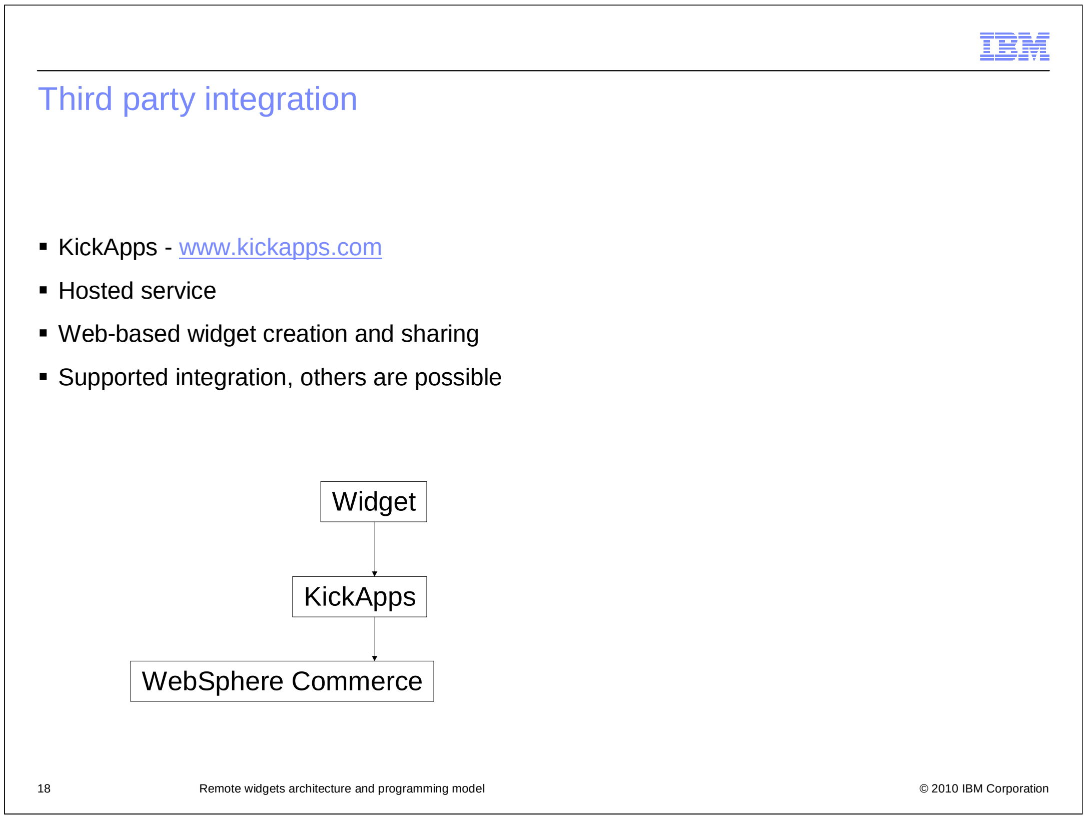


JAX-RS Provider

- A Java class that transforms data from one format to another
  - Transforming SDO format to Atom format
- JSP files used to convert SDOs to Atom format
  - Number of results to return
  - Feed content
  - Content formatting

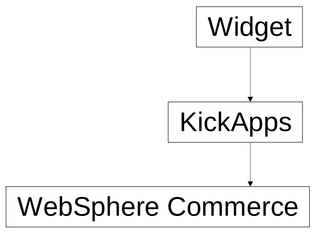
17 Remote widgets architecture and programming model © 2010 IBM Corporation

A JAX-RS Provider is a Java class that transforms data from one format to another. In the application flow shown previously, the source data format is the SDO returned by the OAGIS service request. The target data format is Atom by default but JSON and XML formats are also supported. The JAX-RS Provider uses JSP files to read the SDOs and create the target format. You can replace the existing JSP files to change the output format for e-Marketing Spot and wish lists feeds or you can create new JSP files to support a new resource type.



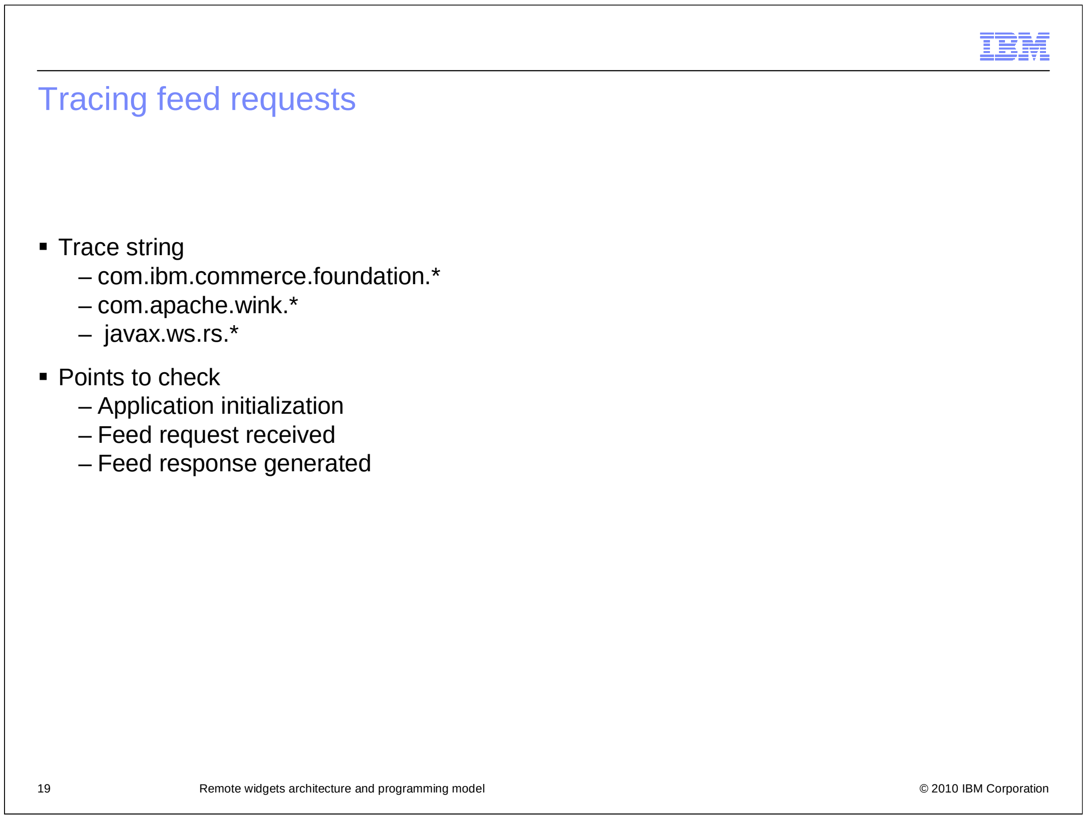
Third party integration

- KickApps - www.kickapps.com
- Hosted service
- Web-based widget creation and sharing
- Supported integration, others are possible



18 Remote widgets architecture and programming model © 2010 IBM Corporation

KickApps is one possible client of the flow just described. As you saw in the Remote Widgets overview presentation, it is a hosted service that can integrate with Management Center to support the creation and sharing of remote widgets. A shared widget does not request data directly from WebSphere Commerce. It requests data from it's parent, KickApps, and KickApps uses the WebSphere Commerce request format to obtain the actual data. KickApps is the supported integration for remote widgets but you can choose to use another vendor or create a custom solution that can make a RESTful request and consume an atom response.

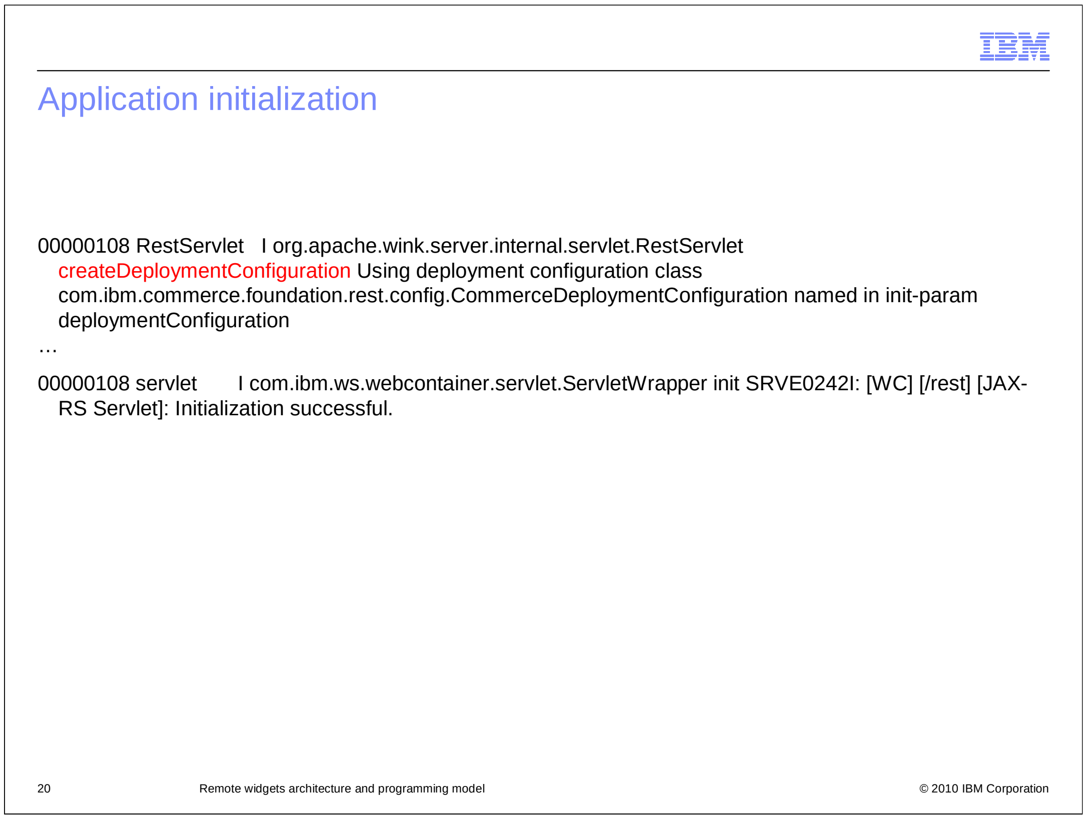


Tracing feed requests

- Trace string
  - com.ibm.commerce.foundation.*
  - com.apache.wink.*
  - javax.ws.rs.*
- Points to check
  - Application initialization
  - Feed request received
  - Feed response generated

19 Remote widgets architecture and programming model © 2010 IBM Corporation

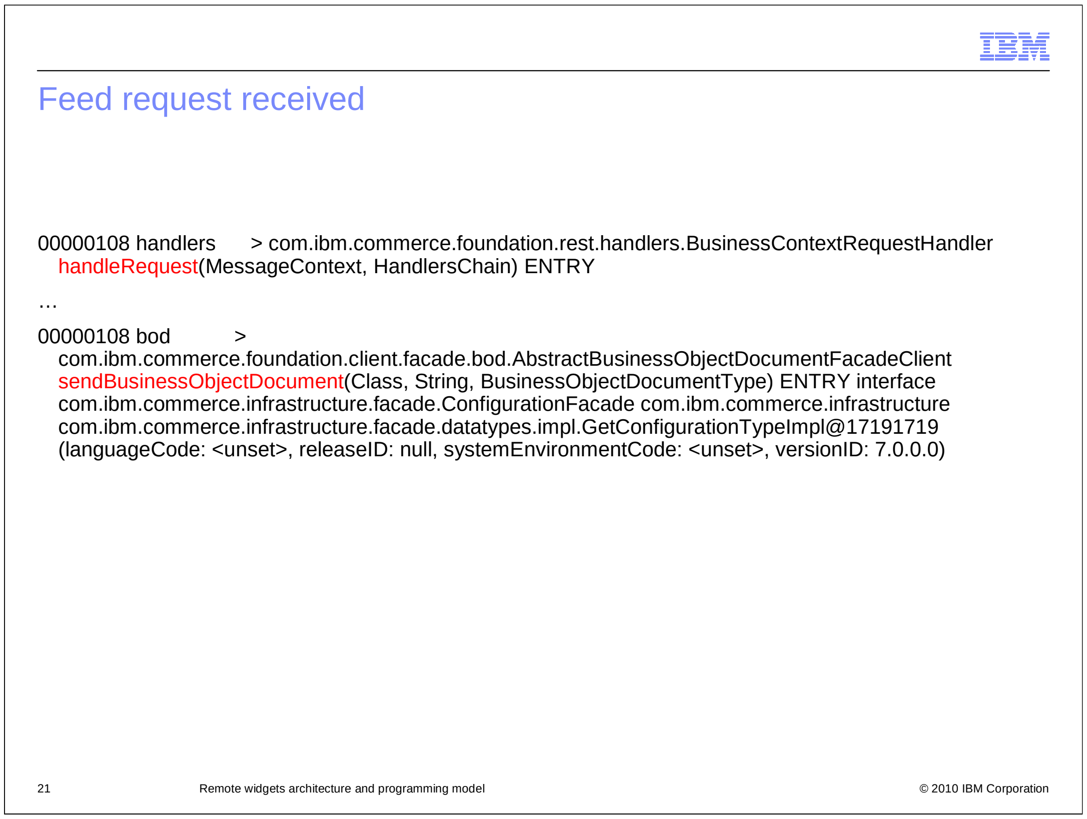
You can trace feed and remote widget requests coming in and atom responses going out by turning on the com.ibm.commerce.foundation trace. This trace allows you to see when the RESTful Web application is initialized and when requests are received and responses generated. To view more details of a request being processed by WebSphere Commerce, turn on the trace for the resource type you are interested in. For example com.ibm.commerce.marketing for e-Marketing Spot feed requests. To view more details of the internal RESTful Web application processing, turn on the com.apache.wink and javax.ws.rs traces.



Application initialization

```
00000108 RestServlet   I org.apache.wink.server.internal.servlet.RestServlet
   createDeploymentConfiguration Using deployment configuration class
   com.ibm.commerce.foundation.rest.config.CommerceDeploymentConfiguration named in init-param
   deploymentConfiguration
…
00000108 servlet       I com.ibm.ws.webcontainer.servlet.ServletWrapper init SRVE0242I: [WC] [/rest] [JAX-
   RS Servlet]: Initialization successful.
```

This slide shows an example trace for a successful application initialization.

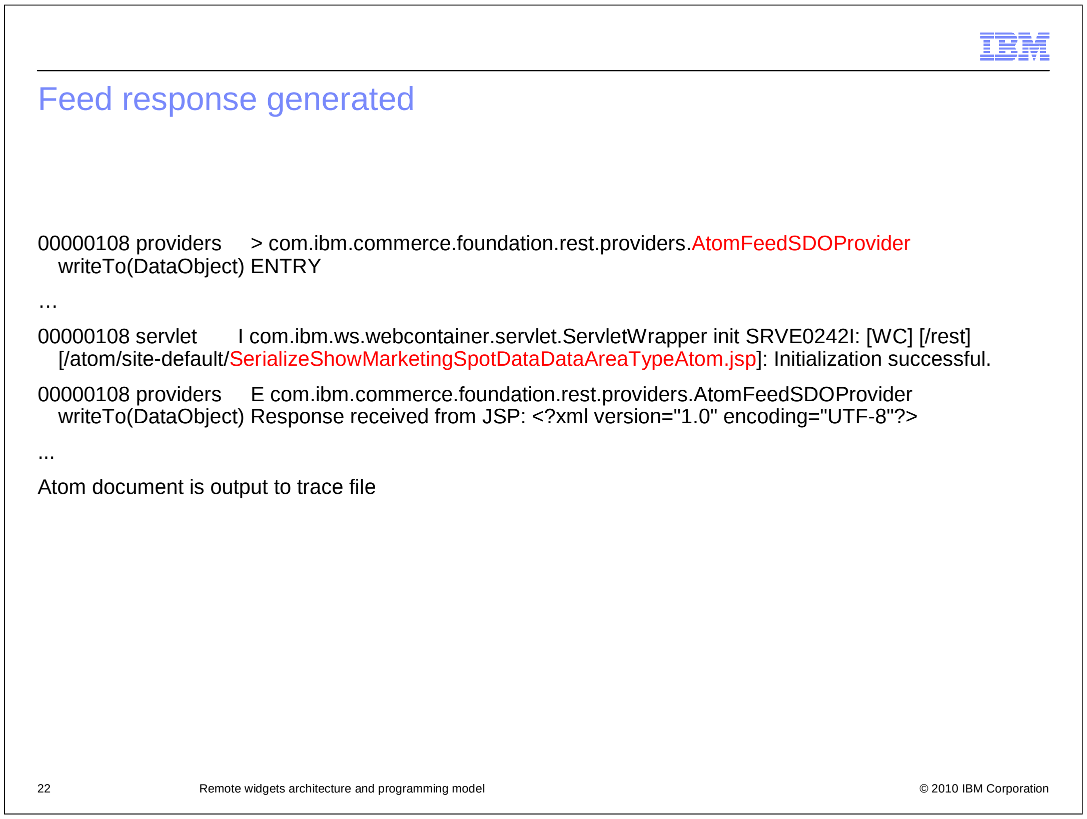


Feed request received

```
00000108 handlers     > com.ibm.commerce.foundation.rest.handlers.BusinessContextRequestHandler
   handleRequest(MessageContext, HandlersChain) ENTRY
…
00000108 bod          >
   com.ibm.commerce.foundation.client.facade.bod.AbstractBusinessObjectDocumentFacadeClient
   sendBusinessObjectDocument(Class, String, BusinessObjectDocumentType) ENTRY interface
   com.ibm.commerce.infrastructure.facade.ConfigurationFacade com.ibm.commerce.infrastructure
   com.ibm.commerce.infrastructure.facade.datatypes.impl.GetConfigurationTypeImpl@17191719
   (languageCode: <unset>, releaseID: null, systemEnvironmentCode: <unset>, versionID: 7.0.0.0)
```

21 Remote widgets architecture and programming model © 2010 IBM Corporation

This slide shows a section of trace when a feed request is received. The handleRequest method is called and it triggers the sendBusinessObjectDocument method call. This is the OAGIS service request to WebSphere Commerce. If you need to trace beyond this method call, you should look at the trace for the specific Get service being invoked.



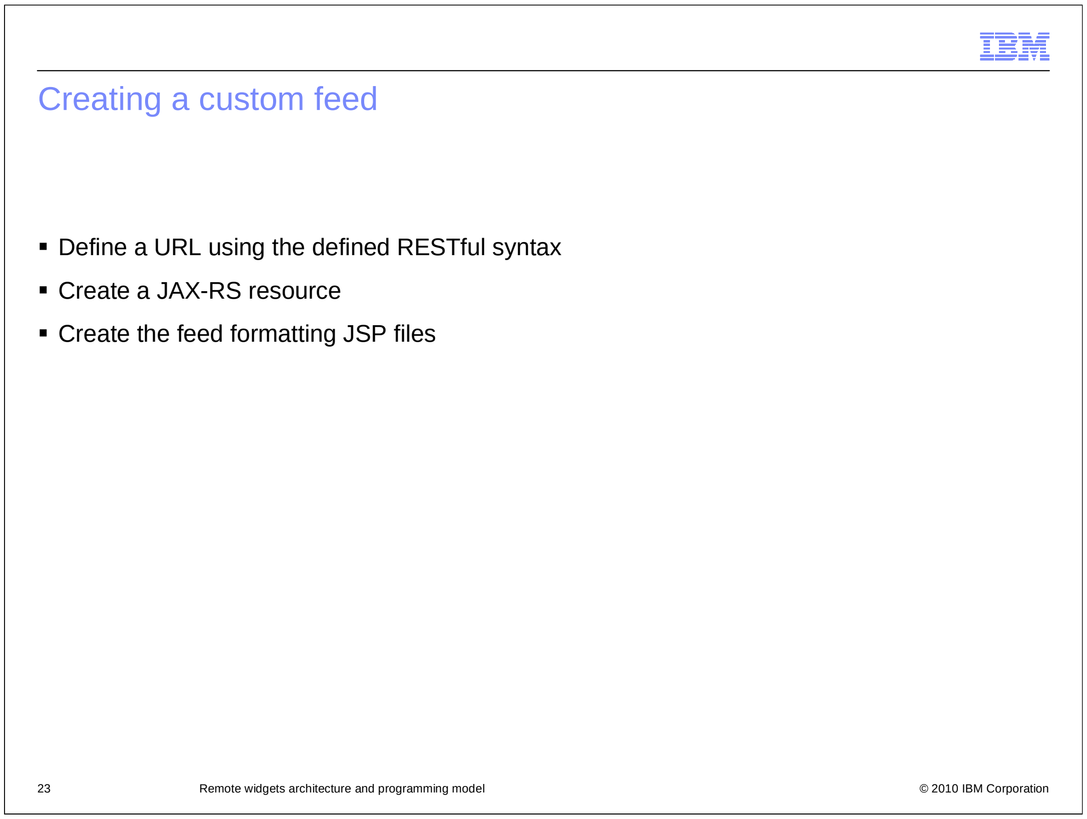
Feed response generated

```
00000108 providers     > com.ibm.commerce.foundation.rest.providers.AtomFeedSDOProvider
   writeTo(DataObject) ENTRY
…
00000108 servlet       I com.ibm.ws.webcontainer.servlet.ServletWrapper init SRVE0242I: [WC] [/rest]
   [/atom/site-default/SerializeShowMarketingSpotDataDataAreaTypeAtom.jsp]: Initialization successful.
00000108 providers     E com.ibm.commerce.foundation.rest.providers.AtomFeedSDOProvider
   writeTo(DataObject) Response received from JSP: <?xml version="1.0" encoding="UTF-8"?>
...
```

Atom document is output to trace file

22 Remote widgets architecture and programming model © 2010 IBM Corporation

This trace example shows some of the steps in processing a feed response. The Get service response is directed to the AtomFeedSDOProvider when the responseFormat of the original feed request is Atom. This provider initializes the JSP specified in the Struts configuration file for the current resource type. The resulting Atom document is output to the trace file.

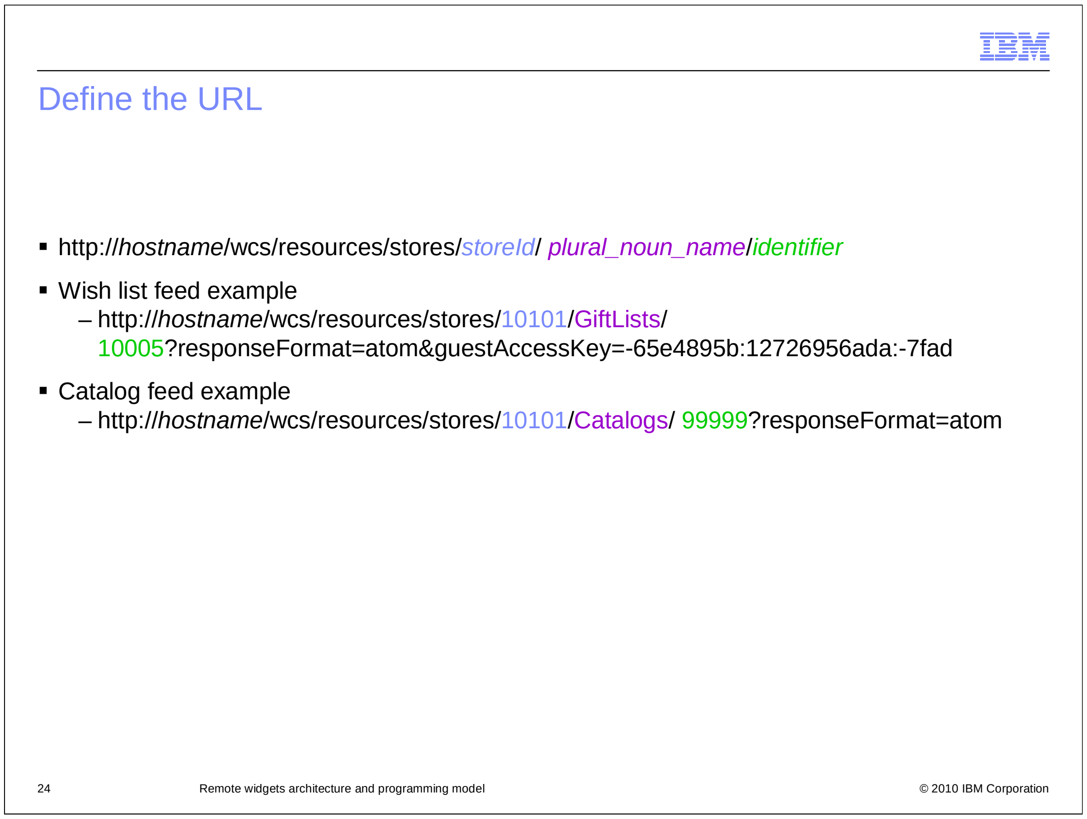


Creating a custom feed

- Define a URL using the defined RESTful syntax
- Create a JAX-RS resource
- Create the feed formatting JSP files

23 Remote widgets architecture and programming model © 2010 IBM Corporation

As shown in the architecture overview diagram earlier, there are three main customization points to consider when creating a custom feed. The first is the URL that defines the RESTful service request made to the Web application. This URL needs to conform to a specific format to be processed by the JAX-RS runtime. The second customization point is the JAX-RS resource. New resources can be created to call both existing and custom WebSphere Commerce Web services. The final customization point is to create new feed formatting JSP files that correspond to the SDO returned by the Web service call. These steps are described in more detail on the upcoming slides.



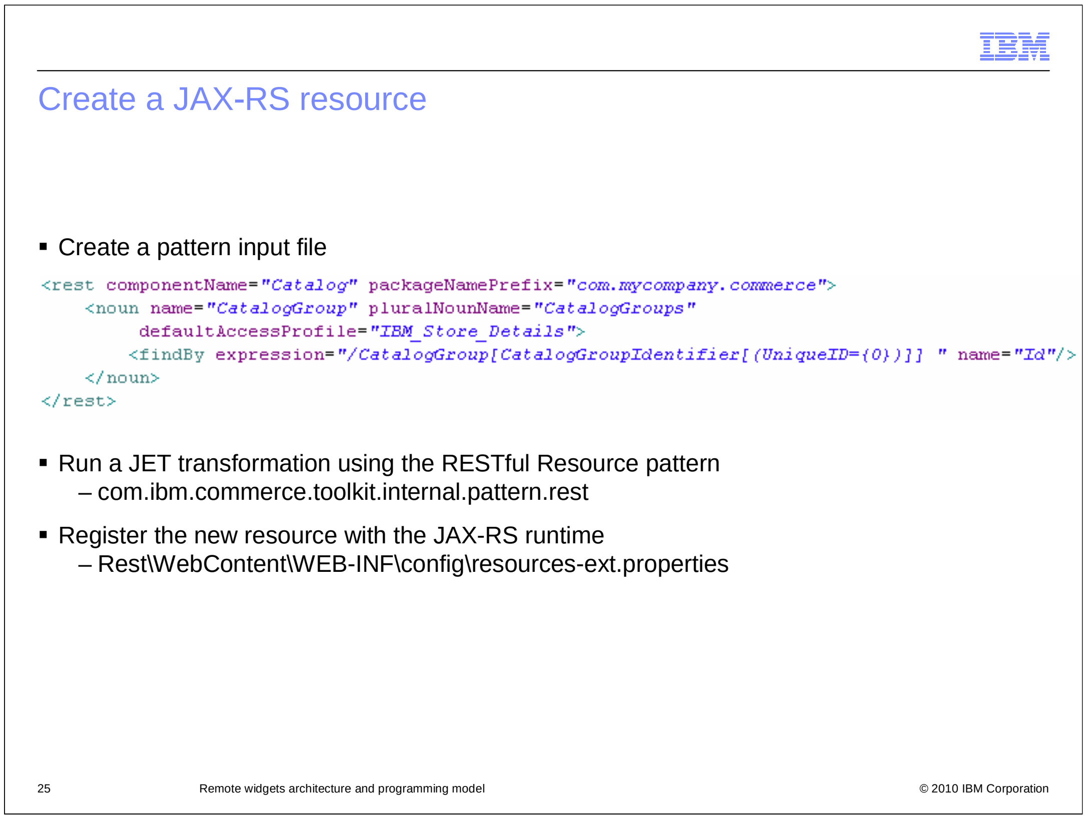
Define the URL

- http://*hostname*/wcs/resources/stores/*storeId*/ *plural_noun_name*/*identifier*
- Wish list feed example
  - http://*hostname*/wcs/resources/stores/10101/GiftLists/ 10005?responseFormat=atom&guestAccessKey=-65e4895b:12726956ada:-7fad
- Catalog feed example
  - http://*hostname*/wcs/resources/stores/10101/Catalogs/ 99999?responseFormat=atom

24 Remote widgets architecture and programming model © 2010 IBM Corporation

In order to be recognized by the JAX-RS runtime, URLs must conform to a specific syntax. You can define a URL for a new Web service feed by following the pattern shown here. The three key pieces of information you need to provide are storeId, plural_noun_name and an identifier. The plural_noun_name is the pluralized name of the noun to get from the Web service for the Atom feed. The identifier is the internal or external ID of the noun to get. Certain nouns might require additional name-value-pairs be added to the base URL structure.

Here you see two examples of URLs. The first is for the a Wish list feed which is provided as part of the remote widgets solution in feature pack 1. The second example is for a catalog feed. This is an example of defining a custom URL to create a feed for an existing WebSphere Commerce service.



Create a JAX-RS resource

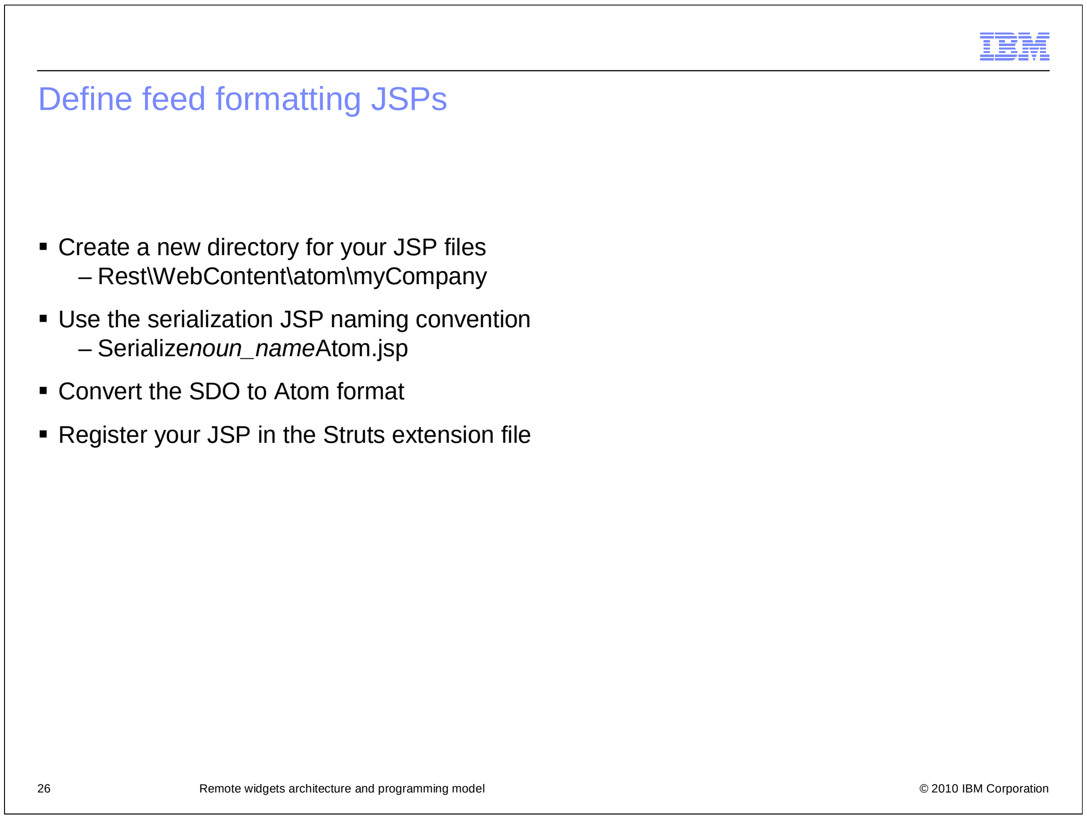
- Create a pattern input file

```
<rest componentName="Catalog" packageNamePrefix="com.mycompany.commerce">
    <noun name="CatalogGroup" pluralNounName="CatalogGroups"
         defaultAccessProfile="IBM_Store_Details">
        <findBy expression="/CatalogGroup[CatalogGroupIdentifier[(UniqueID={0})]] " name="Id"/>
    </noun>
</rest>
```

- Run a JET transformation using the RESTful Resource pattern
  – com.ibm.commerce.toolkit.internal.pattern.rest
- Register the new resource with the JAX-RS runtime
  – Rest\WebContent\WEB-INF\config\resources-ext.properties

25 Remote widgets architecture and programming model © 2010 IBM Corporation

Step two in defining a custom feed is generating the JAX-RS resource that will transform the URL into an OAGIS service call. A JET transformation is provided for you to simplify the creation of the JAX-RS resource. You create a pattern input file as described in the Information Center and then run the JET transformation using the provided pattern. The result is a complete JAX-RS resource Java class that you register by updating a properties file.



Define feed formatting JSPs

- Create a new directory for your JSP files
  - Rest\WebContent\atom\myCompany
- Use the serialization JSP naming convention
  - Serialize*noun_name*Atom.jsp
- Convert the SDO to Atom format
- Register your JSP in the Struts extension file

26 Remote widgets architecture and programming model © 2010 IBM Corporation

The final step in creating a new custom feed is defining the feed formatting JSP. You begin by creating a new directory to store your JSP and naming the file according to the serialization JSP naming convention shown on the slide. Next you create one or more JSP files that access data in an SDO and use it to populate an atom feed. If you are not familiar with the Atom feed format you can use one of the existing serialization JSP files as an example. Once your JSP has been created, you need to register it in the Struts configuration file.

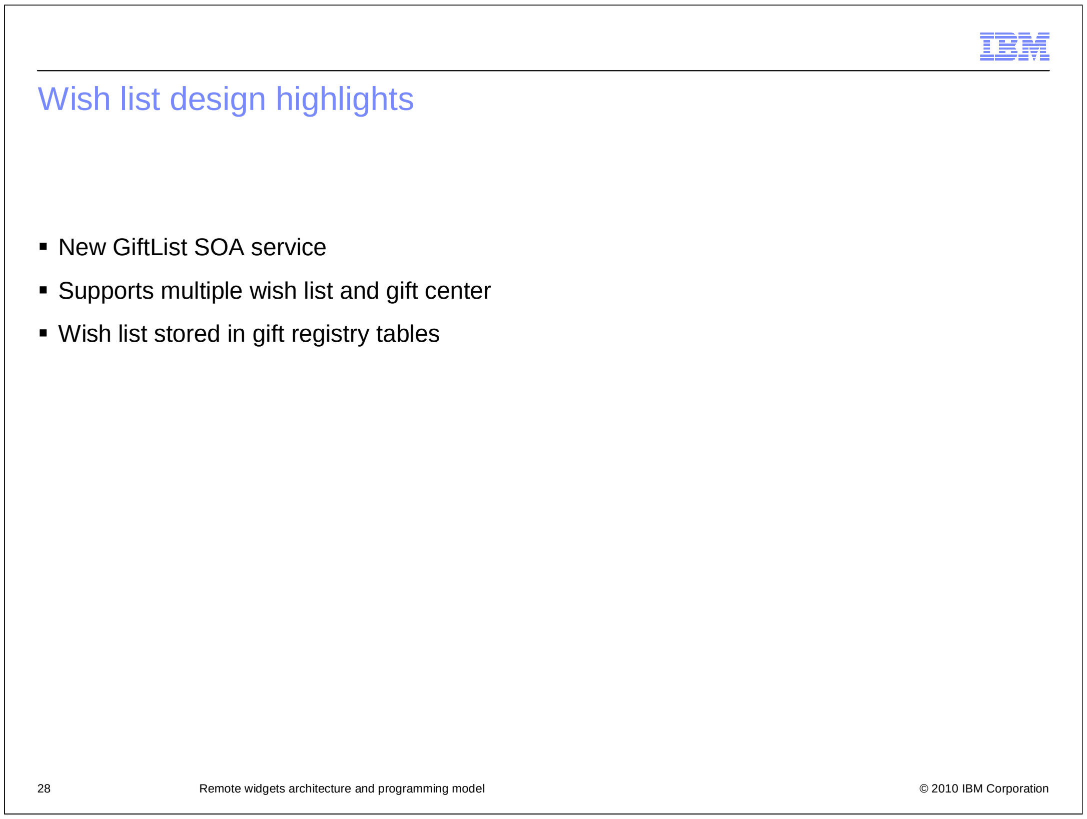
This step of the customization process can also apply to existing feeds. If you want to change the data provided by either the e-Marketing Spot or wish lists feeds, you can replace the default feed formatting JSP files with your own and register them in the Struts configuration file.

Section

Wish list / Gift center

27 Remote widgets architecture and programming model © 2010 IBM Corporation

This section covers the details of multiple wish lists and gift center.

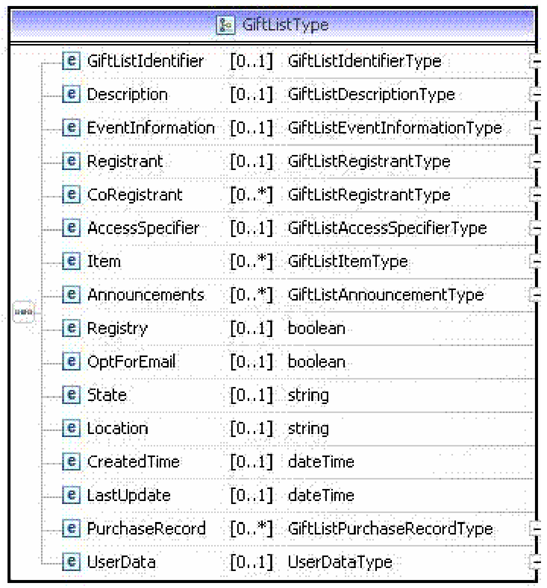


Wish list design highlights

- New GiftList SOA service
- Supports multiple wish list and gift center
- Wish list stored in gift registry tables

New in feature pack 1, the gift center feature is included with the WebSphere Commerce Enterprise and WebSphere Commerce Professional editions at no extra cost. A GiftList SOA service has been added to support both multiple wish list and gift center function. For gift registries, there is no change to the underlying database schema. The new services give you an alternate way of accessing the data and are used to support gift registry remote widgets. In sharing the GiftList service, the multiple wish list feature also shares the gift registry database tables and remote widget capability. Later slides will highlight these changes in more detail.

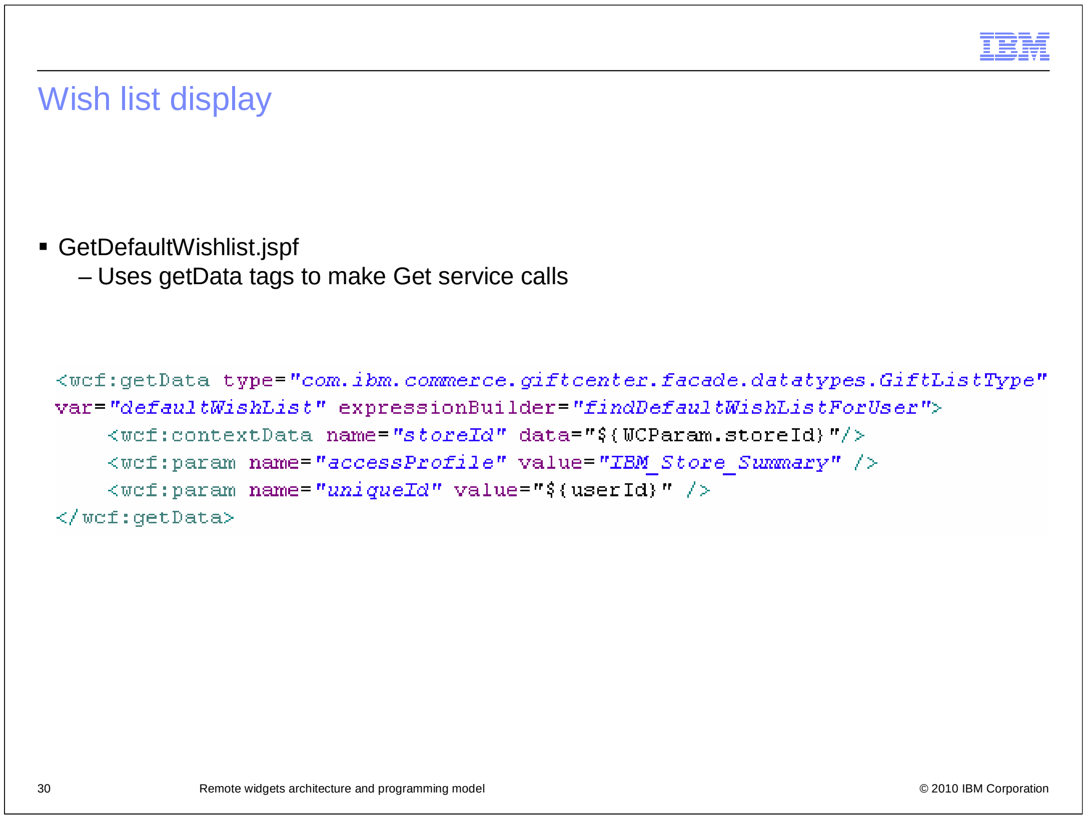
GiftList noun



GiftListType

GiftListIdentifier	[0..1]	GiftListIdentifierType
Description	[0..1]	GiftListDescriptionType
EventInformation	[0..1]	GiftListEventInformationType
Registrant	[0..1]	GiftListRegistrantType
CoRegistrant	[0..*]	GiftListRegistrantType
AccessSpecifier	[0..1]	GiftListAccessSpecifierType
Item	[0..*]	GiftListItemType
Announcements	[0..*]	GiftListAnnouncementType
Registry	[0..1]	boolean
OptForEmail	[0..1]	boolean
State	[0..1]	string
Location	[0..1]	string
CreatedTime	[0..1]	dateTime
LastUpdate	[0..1]	dateTime
PurchaseRecord	[0..*]	GiftListPurchaseRecordType
UserData	[0..1]	UserDataType

This slide shows the structure of the GiftList noun. A subset of these elements is used to represent wish lists. The Boolean element Registry indicates whether the noun represents a gift registry or wish list.

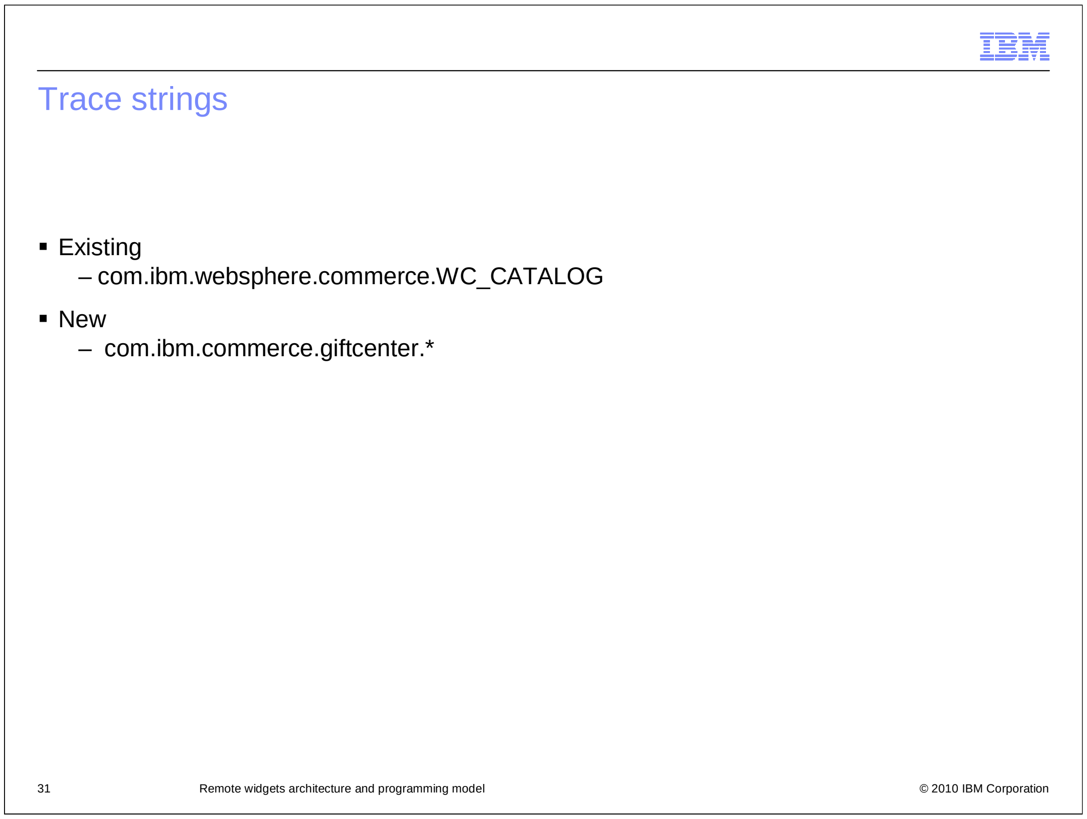


Wish list display

- GetDefaultWishlist.jspf
  - – Uses getData tags to make Get service calls

```
<wcf:getData type="com.ibm.commerce.giftcenter.facade.datatypes.GiftListType"
var="defaultWishList" expressionBuilder="findDefaultWishListForUser">
    <wcf:contextData name="storeId" data="${WCParam.storeId}"/>
    <wcf:param name="accessProfile" value="IBM_Store_Summary" />
    <wcf:param name="uniqueId" value="${userId}" />
</wcf:getData>
```

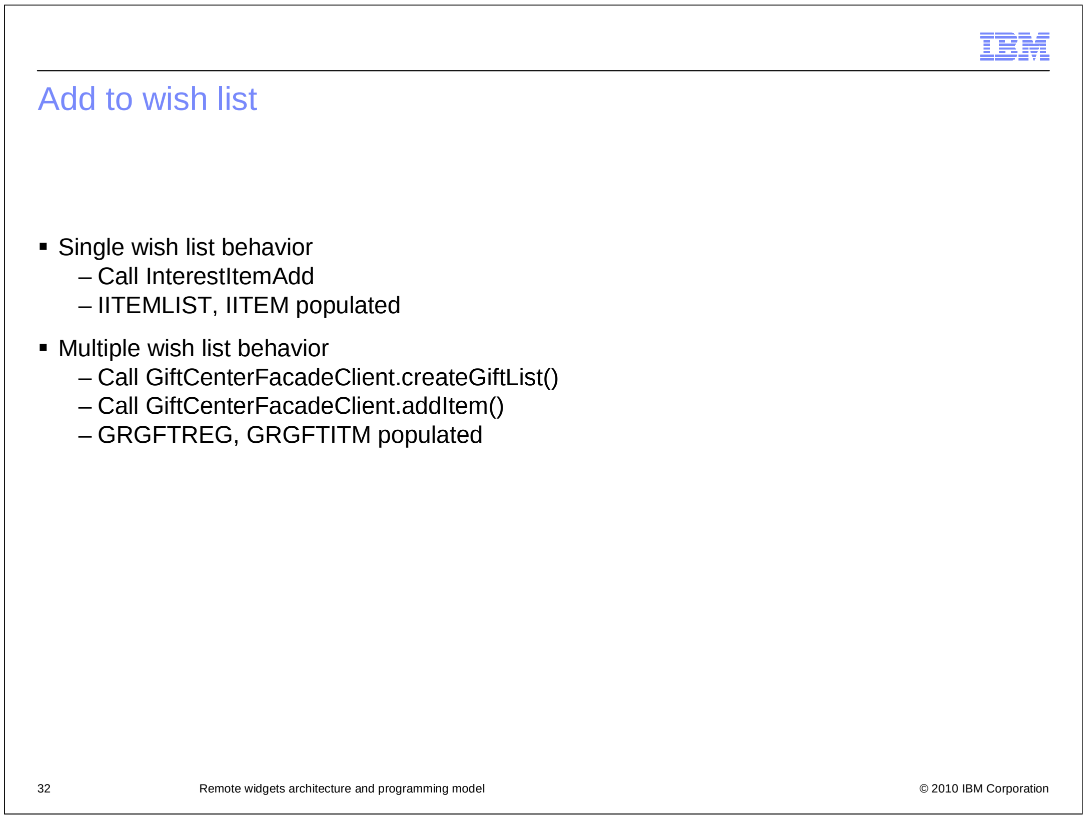
Here you see an example of some of the new JSP code to display a wish list. There is a separate set of JSP pages for displaying wish list data when multiple wish lists are enabled.



Trace strings

- Existing
  - com.ibm.websphere.commerce.WC_CATALOG
- New
  - com.ibm.commerce.giftcenter.*

For debugging purposes, there is a new trace string com.ibm.commerce.giftcenter.* that you can add to see wish list activity when the multiple wish list feature is enabled.

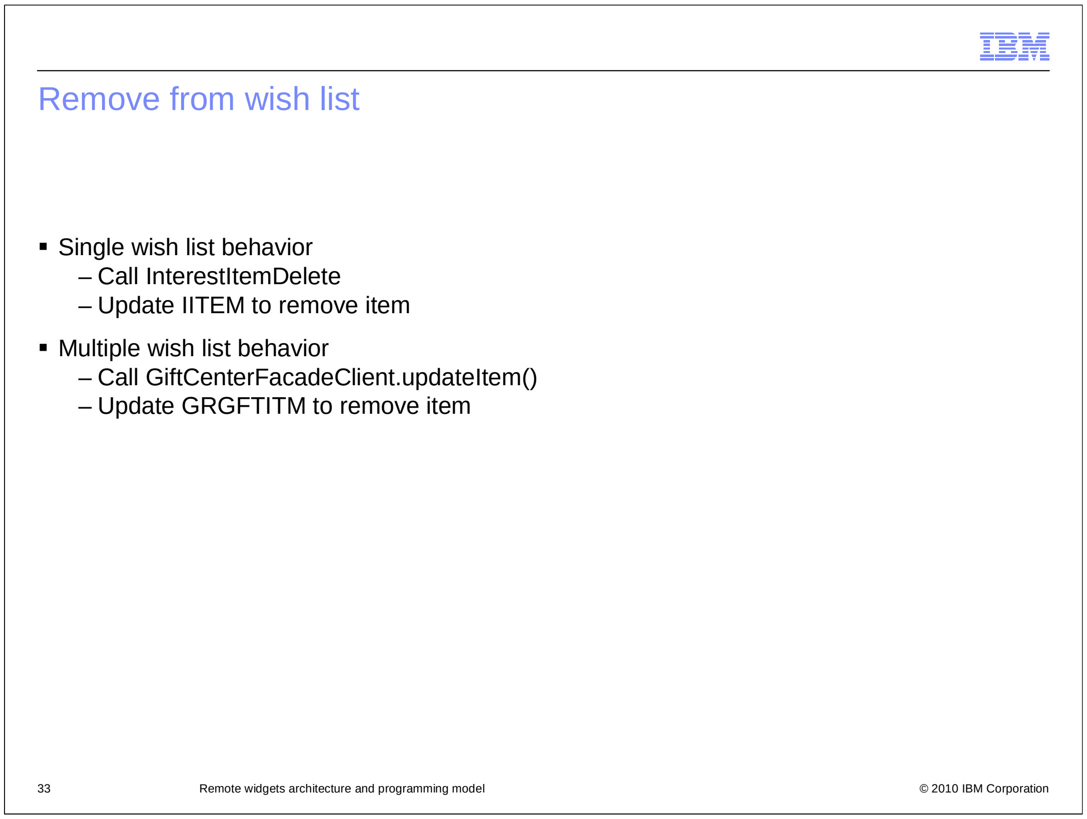


Add to wish list

- Single wish list behavior
  - Call InterestItemAdd
  - IITEMLIST, IITEM populated
- Multiple wish list behavior
  - Call GiftCenterFacadeClient.createGiftList()
  - Call GiftCenterFacadeClient.addItem()
  - GRGFTREG, GRGFTITM populated

32 Remote widgets architecture and programming model © 2010 IBM Corporation

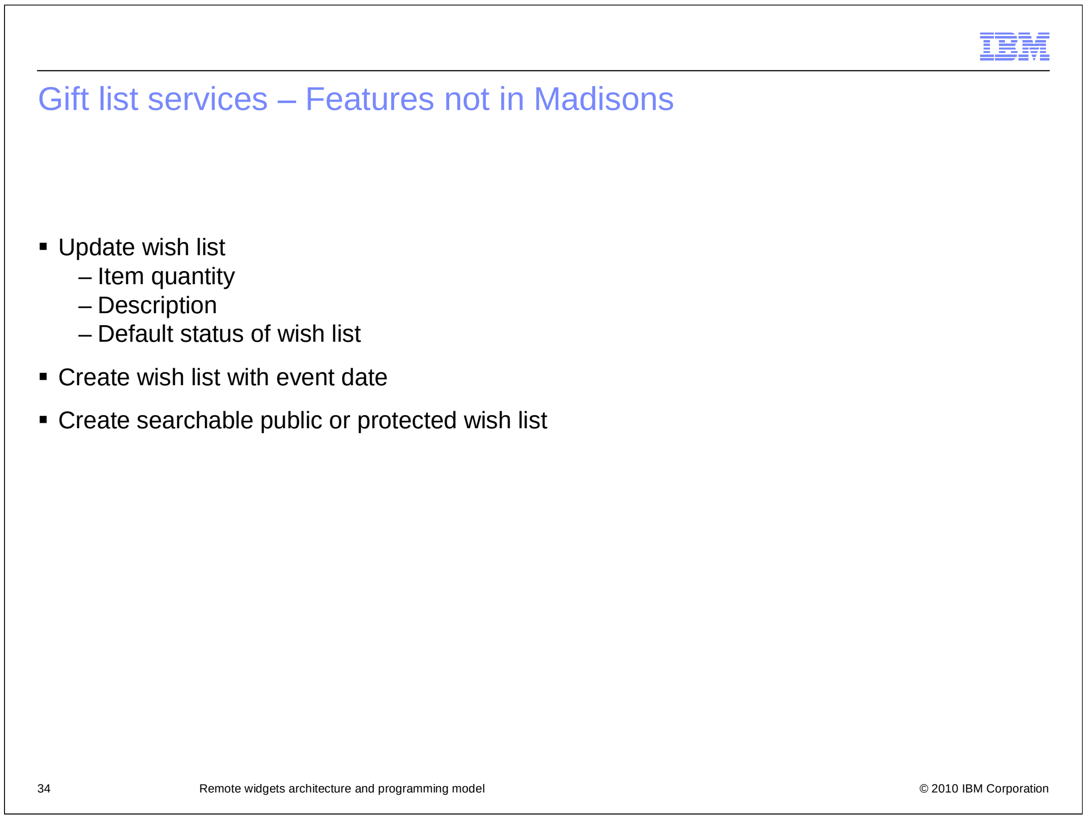
This slide highlights the differences in the add to wish list behavior between single and multiple wish lists. For single wish lists, the InterestItemAdd command is called and the interest item database tables are updated. When multiple wish lists are enabled, the GiftCenterFacadeClient is used to invoke the GiftList services. Wish list data is stored in the gift registry database tables.



Remove from wish list

- Single wish list behavior
  - Call InterestItemDelete
  - Update IITEM to remove item
- Multiple wish list behavior
  - Call GiftCenterFacadeClient.updateItem()
  - Update GRGFTITM to remove item

Removing items from a wish list has similar differences between the single and multiple wish list implementations. For a single wish list, the InterestItemDelete command removes items from the interest item table. When multiple wish lists are used the GiftCenterFacadeClient updates the gift registry item table to remove the item.

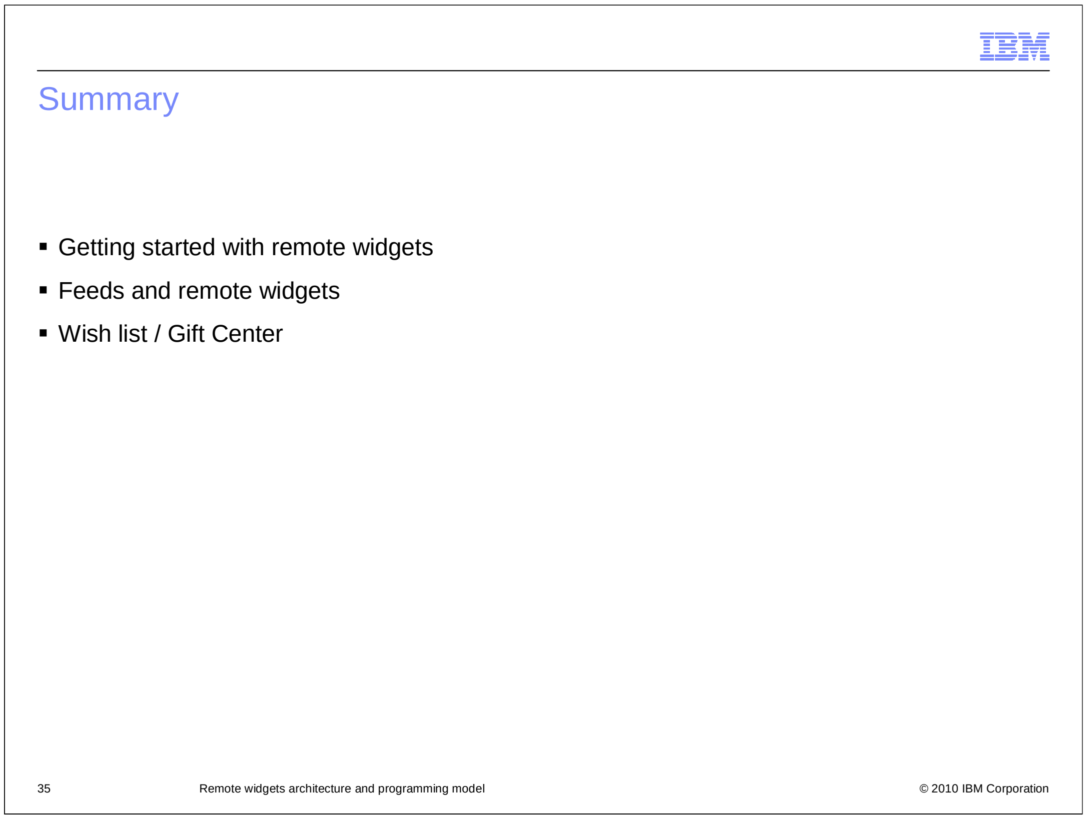


Gift list services – Features not in Madisons

- Update wish list
  - Item quantity
  - Description
  - Default status of wish list
- Create wish list with event date
- Create searchable public or protected wish list

34 Remote widgets architecture and programming model © 2010 IBM Corporation

The GiftList service allows multiple wish lists to share many of the same features as gift registries. This slide highlights features that are supported by the services but not demonstrated in the Madisons starter store. In the Madisons starter store you can update the name of a wish list. You can extend this to include updating item quantity, description or status of the list. You can also use the services to allow shoppers to specify an event date when creating a wish list and make wish lists searchable.

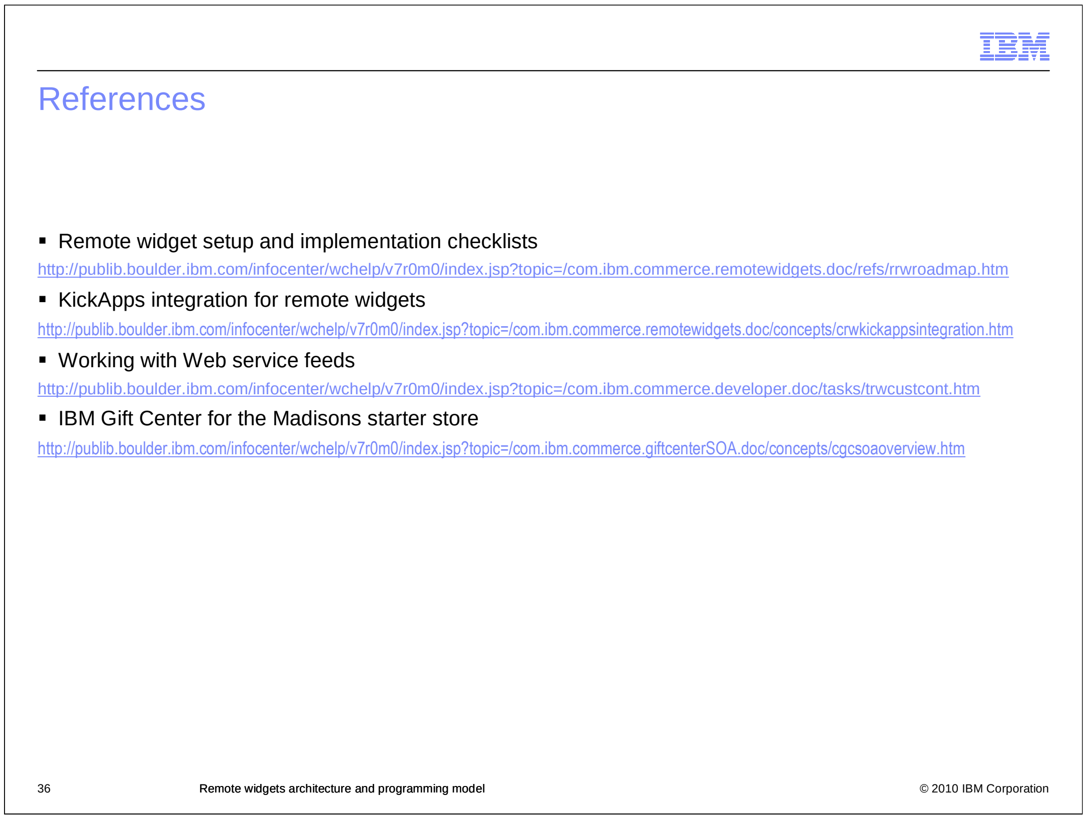


Summary

- Getting started with remote widgets
- Feeds and remote widgets
- Wish list / Gift Center

35 Remote widgets architecture and programming model © 2010 IBM Corporation

This presentation began with a summary of the packaging and configuration steps for the remote widgets and gift center solutions. The second section of the presentation covered the architecture and customization options for feeds and remote widgets. This presentation concluded with an introduction to the new GiftList service and its use in multiple wish lists and Gift Center.



References

- Remote widget setup and implementation checklists

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.remotewidgets.doc/refs/rrwroadmap.htm

- KickApps integration for remote widgets

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.remotewidgets.doc/concepts/crwkickappsintegration.htm

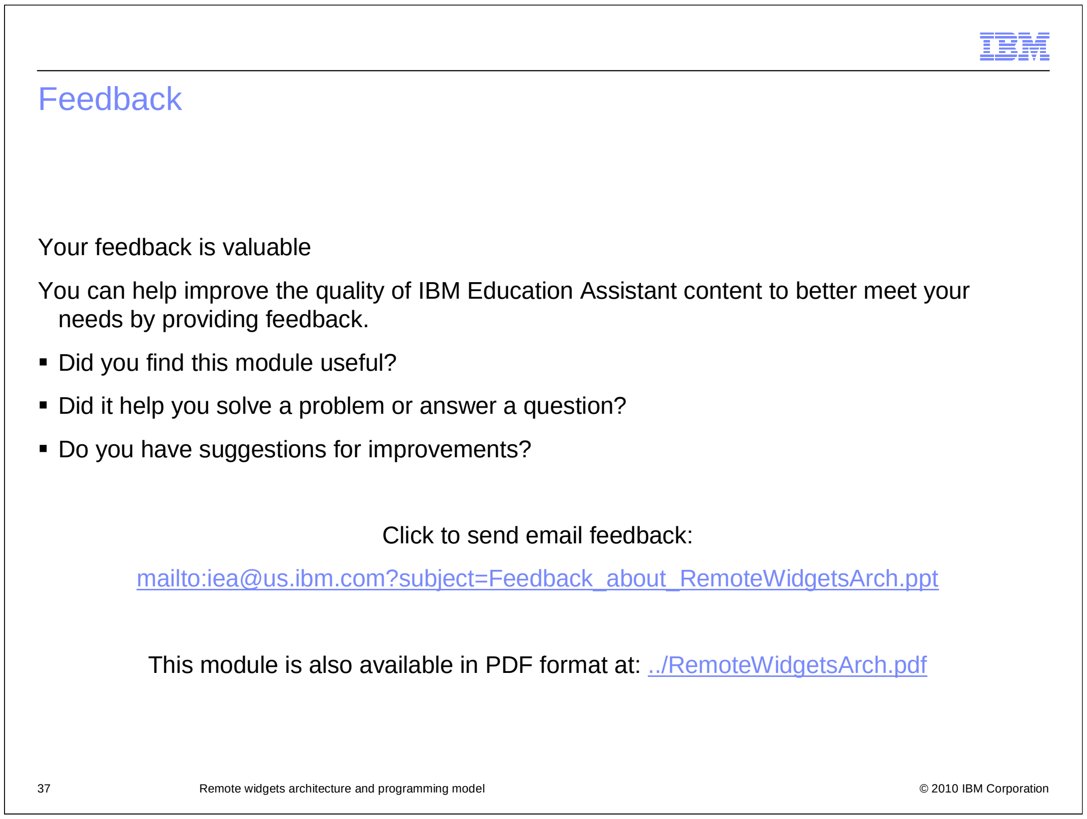
- Working with Web service feeds

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.developer.doc/tasks/trwcustcont.htm

- IBM Gift Center for the Madisons starter store

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/index.jsp?topic=/com.ibm.commerce.giftcenterSOA.doc/concepts/cgcsoaoverview.htm

This slide contains some useful references for further reading.


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 email feedback:

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

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

37 Remote widgets architecture and programming model © 2010 IBM Corporation

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

Trademarks, disclaimer, and copyright information