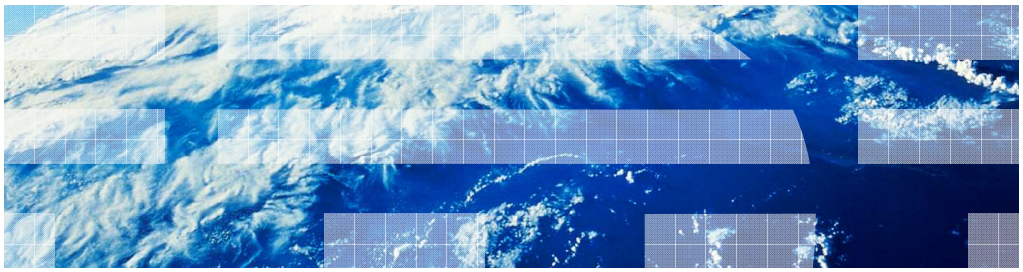
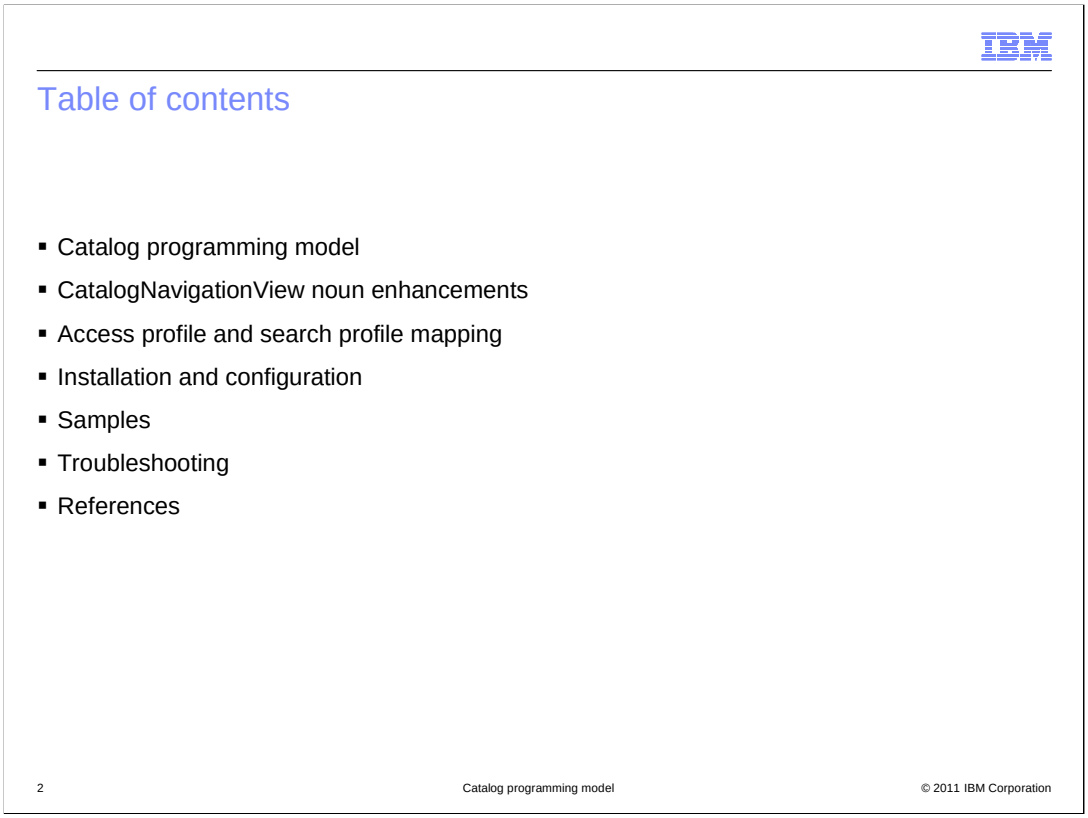
WebSphere Commerce V7 Feature Pack 3

Catalog programming model



WebSphere software

This presentation covers the new catalog programming model that is included in WebSphere Commerce V7 Feature Pack 3. The catalog programming model is a new service for accessing storefront catalog data.

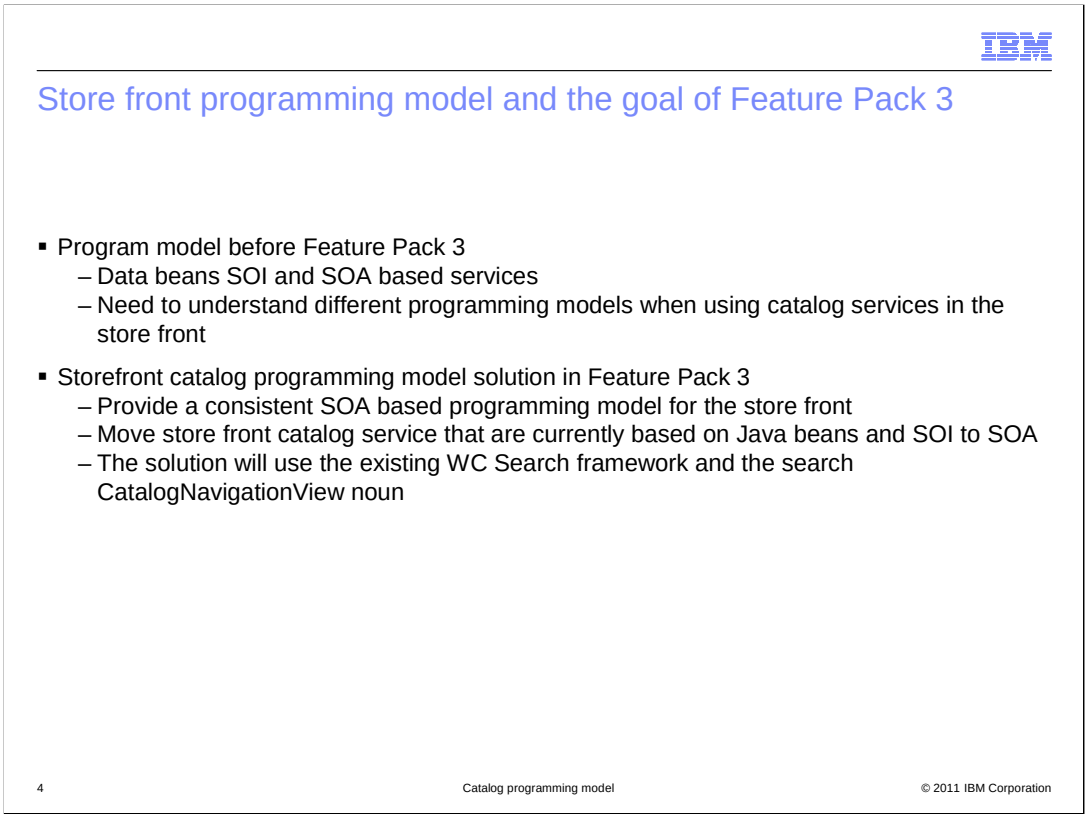


This presentation will cover the new catalog programming model included in Feature Pack 3 and walk through the updates to the CatalogNavigationView noun and the new access and search profiles. The installation and configuration steps needed to use the new model and some samples and troubleshooting information is also included.

Section

Catalog programming model

This section will cover the existing store front programming model, the goals of the Feature Pack 3 new catalog programming model, and the enhancements needed for the model.



The existing store front catalog programming model consists of a mix of technologies including data beans SOI and SOA based services. This results in you needing to understand different programming models when using catalog services in the store front. As part of the ongoing transition to SOA, Feature Pack 3 introduces new services for accessing storefront catalog data. This provides a consistent programming model across the catalog component. The catalog service is built on top of the WebSphere Commerce search solution so some of the information for the catalog entries is retrieved from the search engine and other information from the database. The CatalogNavigationView noun that was introduced in feature pack 2 has been extended in Feature Pack 3 and contains information from both the search index and database. This design simplifies storefront development since a single noun can be used for accessing all catalog data.

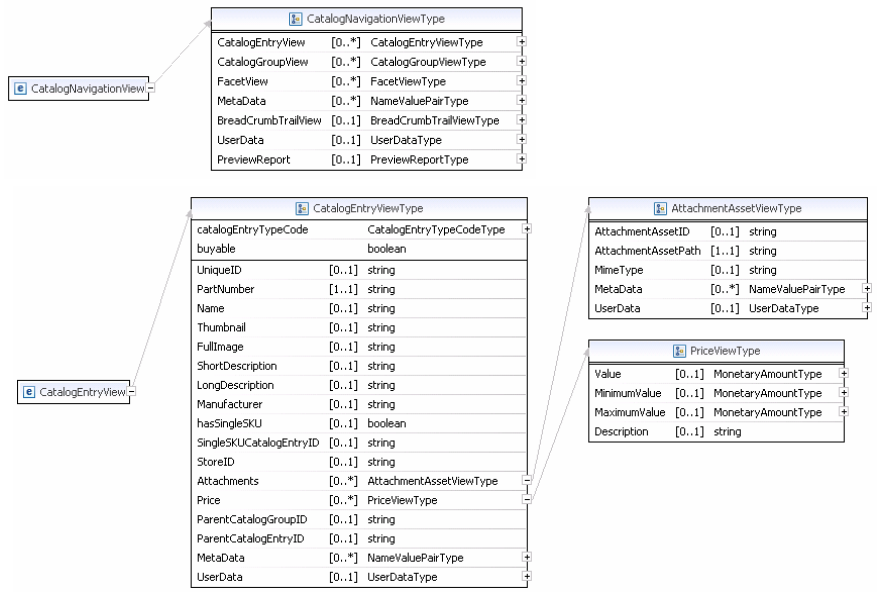
Catalog data beans are currently used by store front

- In Feature Pack 3 only catalog entry related data bean are moved to SOA service
- The store front currently uses these catalog data beans

Data beans	Addressed areas in FEP3
CatalogEntryDataBean	Y
ProductDataBean	Y
PackageDataBean	Y
DynamicKitDataBean	Y
BundleDataBean	Y
ItemDataBean	Y
InterestItemListDataBean	N
CatalogDataBean	N
CategoryDataBean	N

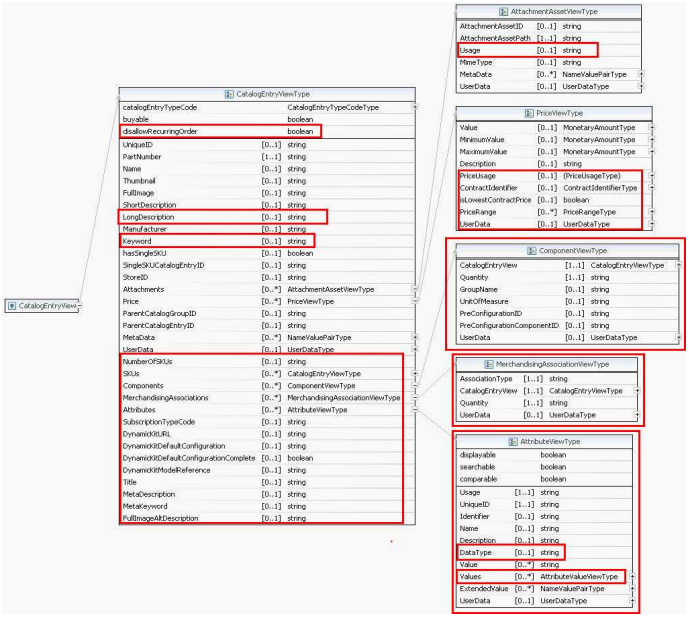
In Feature Pack 3, the focus is on catalog entry data. Category and catalog details are still accessed using data beans.

CatalogNavigationView noun for Feature Pack 2

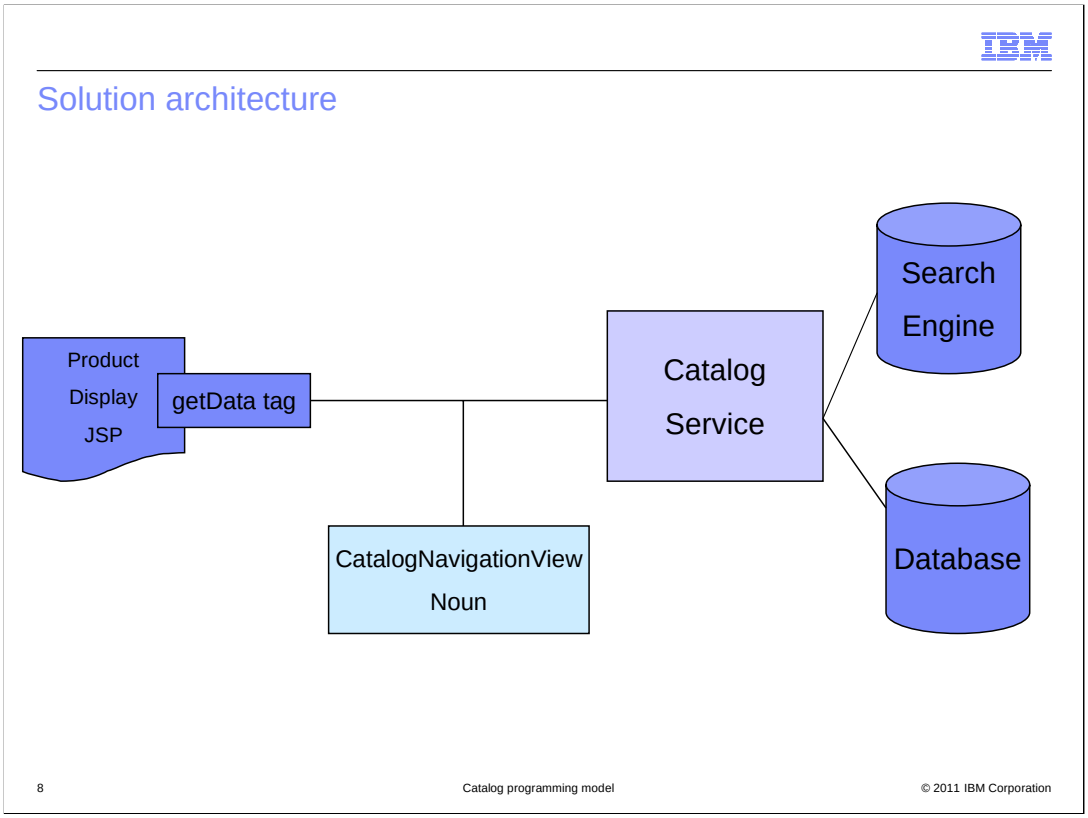


In Feature Pack 2 the CatalogNavigationView noun was introduced as part of the new Search feature.

CatalogNavigationView noun enhancements for Feature Pack 3

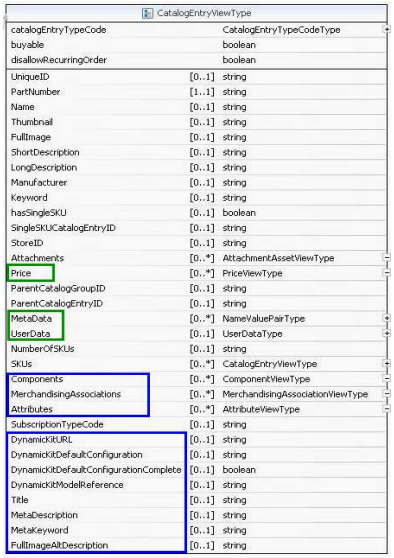


The CatalogNavigationView function in Feature Pack 2 does not replace all the elements provided by the data beans listed on the previous slide. The areas missing in the CatalogNavigationView noun include Additional data specific to the catalog entry, Components associated with the catalog entry, Merchandising associations for the catalog entry, Pricing information relating to quantities and contracts, and Catalog entry attributes. Also the noun does not include Search Engine Optimization (SEO) elements, and Sterling Commerce configurator support elements. The CatalogNavigationView noun is enhanced in Feature Pack 3 allowing for the replacement of the catalog entry related data beans in the store front with new Catalog SOA based services. The goal is to minimize the learning curve for store front services, so that regardless of whether the store front navigation is driven by search or database query, the store front is using a consistent noun. Using the CatalogNavigationView noun will reduce the number of levels that you must navigate to reach the required data.



Here is an architectural overview of the catalog service. The product display page uses the getData tag and makes a call to the catalog service. Since the solution is built on top of the WebSphere Commerce search solution, some of the information for the catalog entry is retrieved from the search engine and other information from the WebSphere Commerce database. The CatalogNavigationView noun is populated with data from both sources and returned in the service response. That data will then be used to populate the product display page in the storefront.

Data comes from search versus the database



CatalogEntryViewType		
catalogEntryTypeCode		CatalogEntryTypeCodeType
buyable		boolean
disallowRecurringOrder		boolean
UniqueID	[0..1]	string
PartNumber	[1..1]	string
Name	[0..1]	string
Thumbnail	[0..1]	string
FullImage	[0..1]	string
ShortDescription	[0..1]	string
LongDescription	[0..1]	string
Manufacturer	[0..1]	string
Keyword	[0..1]	string
hasSingleSKU	[0..1]	boolean
SingleSKUCatalogEntryID	[0..1]	string
StoreID	[0..1]	string
Attachments	[0..*]	AttachmentAssetViewType
Price	[0..*]	PriceViewType
ParentCatalogGroupID	[0..1]	string
ParentCatalogEntryID	[0..1]	string
MetaData	[0..*]	NameValuePairType
UserData	[0..1]	UserDataType
NumberOfSKUs	[0..1]	string
SKUs	[0..*]	CatalogEntryViewType
Components	[0..*]	ComponentViewType
MerchandisingAssociations	[0..*]	MerchandisingAssociationViewType
Attributes	[0..*]	AttributeViewType
SubscriptionTypeCode	[0..1]	string
DynamicKitURL	[0..1]	string
DynamicKitDefaultConfiguration	[0..1]	string
DynamicKitDefaultConfigurationComplete	[0..1]	boolean
DynamicKitModelReference	[0..1]	string
Title	[0..1]	string
MetaDescription	[0..1]	string
MetaKeyword	[0..1]	string
FullImageAltDescription	[0..1]	string

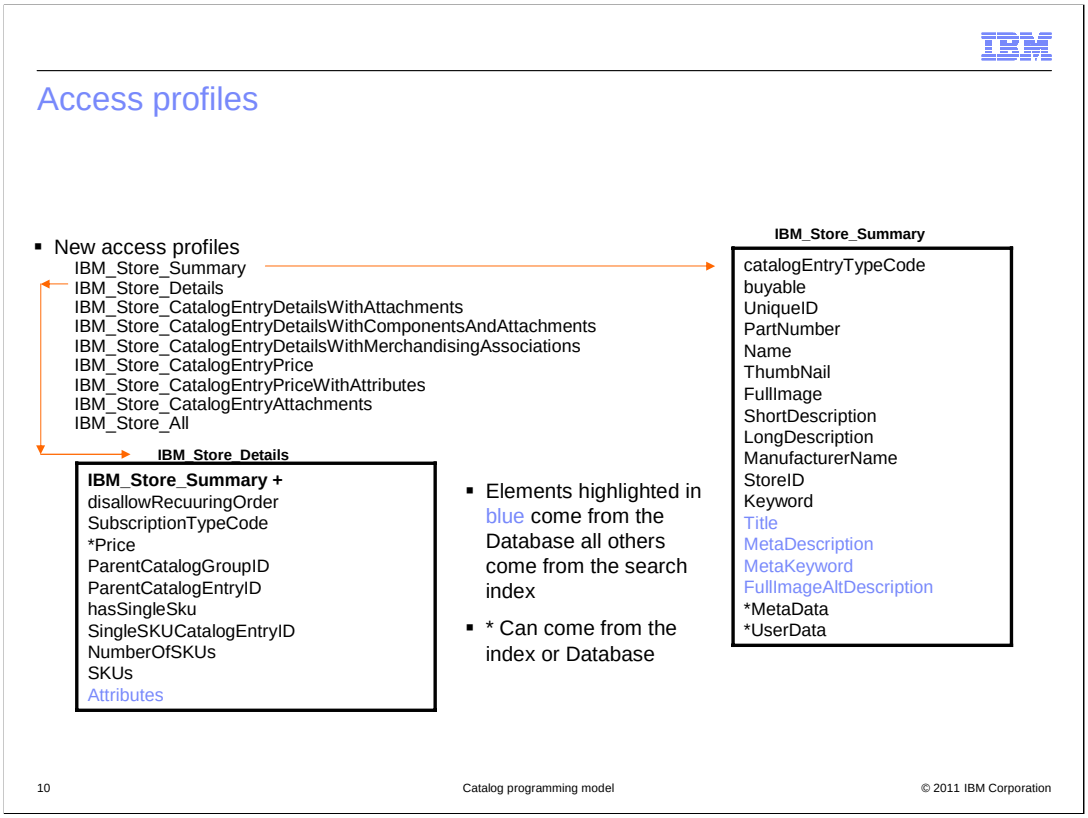
- Each element in the noun will either come from the search index or the Database
- Elements highlighted in blue come from the Database
  - Some elements themselves contain CatalogEntryView elements and therefore some of that information comes from the index
- Elements highlighted in green can come from the index or the DB
  - Where Price data comes from depends on which price display mode is selected in wc-search.xml
  - MetaData is used internally
  - UserData is for you to perform customization
- All other elements come from the search index
- Elements are added to the noun by either a mediator or results filters
  - New filters were added to pull in the extra data required by the product display page

9 Catalog programming model

As the catalog programming model builds on the WebSphere Commerce Search solution, each new element added to the noun will either come from the search index or the database. In general, if the data is fairly static and does not change often, such as Name or Keyword, it will come from the index. If the data is very dynamic or needs to be computed, it will come from the database. Price data can come from either the index or the database depending on which price display mode is selected in wc-search.xml. UserData is a place holder element for you to re-use for your own data. This element allows you to add your own data without having to change the logical model.

To support the new noun, long description, keywords, disallow recurring order, and subscription type were added to the Catalog Entry index. The Unstructured Content index now indexes all attachments and is sorted according to attachment relationship sequence number (instead of relationship id). The rulename element is now mapped to attachment Usage.

Elements are added to the noun by either a mediator or results filters. New filters and mediators were added to pull in extra data that is required by the product display page. The results filters use a combination of index queries, command calls and service calls to add data to the new noun elements. When possible the results filters will use search profiles to retrieve the required information from the index or database. These profiles can then be easily modified to return additional information with no code changes.



In previous releases you learned that you specify an XPath with a corresponding access profile that defines what data is returned by the XPath query. In order to preserve this familiar pattern Feature Pack 3 defines access profile that are mapped with a search profile. This allows you to specify only an access profile with the XPath query. The access profile will provide access control and determine what information is coming from the database. Here is a list of access profiles and details for the IBM_Store_Summary and IBM_Store_Details profiles fields returned by each profile.

Search profile

- wc-component.xml contains the mapping between the access profile and the related search profile

ccess Profile	Search Profile
BM_Store_Summary	BM_findCatalogEntrySummary
BM_Store_Details	BM_findCatalogEntryDetails
BM_Store_CatalogEntryAttachments	BM_findCatalogEntryAttachments
BM_Store_CatalogEntryDetailsWithAttachments	BM_findCatalogEntryDetailsWithAttachments
BM_Store_CatalogEntryPrice	BM_findCatalogEntryPrice
BM_Store_CatalogEntryPriceWithAttributes	BM_findCatalogEntryPriceWithAttributes
BM_Store_CatalogEntryDetailsWithComponentsAndAttachments	BM_findCatalogEntryDetailsWithComponentsAndAttachments
BM_Store_CatalogEntryDetailsWithMerchandisingAssociations	BM_findCatalogEntryDetailsWithMerchandisingAssocDetails
BM_St _All	BM_fi dC t l E t All

- Each access profile has a new search profile defined in the wc-search.xml

```
- <_config:profile indexName="CatalogEntry" name="IBM_findCatalogEntrySummary">
  + <_config:query>
  - <_config:result>
      <_config:field name="catentry_id" />
      <_config:field name="storeent_id" />
      <_config:field name="buyable" />
      <_config:field name="partNumber_ntk" />
      <_config:field name="name" />
      <_config:field name="thumbnail" />
      <_config:field name="fullImage" />
      <_config:field name="shortDescription" />
      <_config:field name="longDescription" />
      <_config:field name="keyword" />
      <_config:field name="mfName_ntk" />
      <_config:field name="catenttype_id_ntk_cs" />
    </_config:result>
  </_config:profile>
```


The access profiles need to be mapped with a search profile. Here is a list of the new search profiles and the corresponding access profiles. wc-compontnet.xml configuration property links the access profile to their respective search profiles. The associated search profiles will control information coming from the index. The new search profiles are defined in wc-search.xml. An example search profile IBM_findCatalgoEntrySummary is shown here listing the elements that will come from the index when using this profile.

Expression builder

- One expression builder for each access profile and for both category entry ID and part number
  - For example: getCatalogEntryViewDetailsByID and getCatalogEntryViewDetailsByPartnumber expression builders for IBM_Store_Details access profile
- Override the associated search profile
  - Specify a search profile on the expression builder definition or the getData call in the JSP files
  - Note: This is not recommended as best practice
  - For example:

```
<wcf:getData type="com.ibm.commerce.catalog.facade.datatypes.CatalogNavigationViewType"
        var="catalogNavigationView“ expressionBuilder="getCatalogEntryViewDetailsByID"
        varShowVerb="showCatalogNavigationView" scope="request">
                <wcf:param name="UniqueID" value="${catEntryID}" />
                <wcf:contextData name="storeId" data="${WCParam.storeId}" />
                <wcf:contextData name="catalogId" data="${WCParam.catalogId}" />
                <wcf:param name="searchProfile" value="MySearchProfile"/>
 </wcf:getData>
```


Expression builders can be used for the new catalog programming model. One expression builder exists for each access profile and for both catentryID and partNumber. For example the getCatalogEntryViewDetailsByID and getCatalogEntryViewDetailsByPartnumber expression builders were created for the IBM_Store_Details access profile.

For convenience, but not recommended as best practice, you can still specify a search profile on the expression builder definition or the getData call in the JSP files to override the associated search profile. The UniqueIDsExpressionBuilder class was modified to support the searchProfile parameter which can be passed in addition to the access profile. An example of a getData call with the getCatalogEntryViewDetailsByID expression builder is shown here. MySearchProfile is used instead of the associated IBM_Store_Details access profile which is linked to the IBM_findCatalogEntryDetails search profile.

XPath expression

- The **CatalogNavigationView** noun XPath is extended
  - Support returning results for one or more catalog entries by catalog entry ID and part number

/CatalogNavigationView[CatalogEntryView[(UniqueID=)]]

/CatalogNavigationView[CatalogEntryView[(PartNumber=)]]

- The XPath to Solr search query mappings are defined in **wc-search.xml** using the new elements:

```
<_config:queryMapping indexName='CatalogEntry'
  xpathKey='/CatalogNavigationView[CatalogEntryView[(UniqueID=)]]'
  searchQuery='catentry_id:(?UniqueID?)'/>

<_config:queryMapping indexName='CatalogEntry'
  xpathKey='/CatalogNavigationView[CatalogEntryView[(PartNumber=)]]'
  searchQuery='partNumber_ntk:(?PartNumber?)'/>
```

- Control parameters for access control (_wcf.ap) is used

The solution extends the existing **CatalogNavigationView** noun XPath to support returning results for one or more catalog entries by catalog entry ID or part number. UniqueID or PartNumber is used to specify the catalog entry for which to return results. The queries use the explicit UniqueID or partnumber passed in on the XPath for the search. The XPaths shown here are supported. The XPath to Solr search query mappings are defined in **wc-search.xml** using the new element shown above. The existing control parameters for access control (_wcf.ap) is used to control access and details returned.

Installation and configuration

- The server code component is installed as part of enabling the 'foundation' feature
- Enable WebSphere Commerce search

In order to use the new catalog programming model to retrieve catalog data you need to enable the 'foundation' feature and enable WebSphere Commerce search.

Samples

- Sample JSP files based on the default JSP pages are provided to showcase the new noun and noun subparts for Product display
  - **components/store-enhancements/samples/SOACatalogStorefrontServices/stores/**
- The new Sterling Configurator solution default JSP files for DynamicKits and configurator use the new service
- The SEO Solution default JSP files use the new noun for SEO information
- Tutorial to display warranty information on the storefront using Catalog navigation noun

The Madisons and Elite stores were not updated to use the new catalog programming model. Documentation and sample updated store front default JSP files are provided to aid in the understanding of how to use the new storefront services and customize it. The sample JSP files are placed in the toolkit located in the path shown here. The sample JSP files will cover product, item, bundle, and kit displays in the storefront including pricing, attributes, merchandising associations, bundles and package components and attachments. In addition to the sample JSP files, the new Sterling Configurator solution for DynamicKits and SEO default JSP files use the new service.

In addition to the samples, the existing warranty tutorial is enhanced to show how to use and customize the new service. The tutorial shows the steps to create additional catalog data and display it in the storefront using the new Catalog navigation noun.

Display warranty information on the storefront using catalog navigation noun (1 of 2)

- Create new warranty tables and insert test data, Extend the search schema to support the warranty fields, Configure the Data Import Handler to extract data from the relational table
- Extend the IBM_findCatalogEntrySummary search profile to return warranty data

```
<_config:profile name="X_findCatalogEntryWarranty“ extends="IBM_findCatalogEntrySummary">
        <_config:result>
        <_config:field name="warterm" />
```

- Customize the Business Object Mediator to map the new warranty columns from the search index to the user data field of the CatalogEntryView noun

```
<_config:mediator-property name="CatalogEntryView/UserData[(Name='Warranty Term')]"
value="warterm" />
```

- Create a new access profile and map to the new search profile

```
<ActionGroupAction Name="GetCatalogNavigationView.IBM_Store_Warranty"/>
        <_config:configgrouping name="AccessProfileToSearchProfileMapping">
                <_config:property name="IBM_Store_Warranty"
value="X_findCatalogEntryWarranty"/>
```


The updated Warranty tutorial shows the steps to create additional catalog data and display it in the storefront using the new Catalog navigation noun. It starts off with the existing warranty steps of creating new warranty tables and inserting test data, extending the search schema to support the warranty fields, and configuring the Data Import Handler to extract data from the relational table. A new search profile is needed when you want to retrieve additional information from the search index. So it walks through the steps to extend the IBM_findCatalogEntrySummary search profile to return warranty data by creating a new search profile. In order to display warranty information on the storefront, you then have to map the new search results to the user data field of the CatalogEntryView noun. This is done by extending the business object mediator configuration to include the new warranty fields. An expression builder is used on the storefront to display the data. This builder expects an access profile as a parameter, so a new access profile is created and mapped to the new search profile created.

Display warranty information on the storefront using catalog navigation noun (2 of 2)

- Extend the getCatalogEntryViewSummaryByID expression builder to use the new profile

```
<name>accessProfile</name>
<value>IBM_Store_Warranty</value>
```

- Add another tab to the product display page and Populate the new warranty tab with content

```
<wcf:getData
type="com.ibm.commerce.catalog.facade.datatypes.CatalogNavigationViewType"
var="catalogNavigationView" expressionBuilder="getCatalogEntryViewSummaryByID"
varShowVerb="showCatalogNavigationView" maxItems="1" recordSetStartNumber="0"
scope="request">
…
       <c:forEach var="userDataField"
items="${catEntry.userData.userDataField}">
                <b>${userDataField.typedKey}: </b>${userDataField.typedValue}
<br>
       </c:forEach>
```


Then the expression builder is extended to use the new access profile. The final step is modifying the storefront JSP to display the warranty information and populate it with the warranty information retrieved using the getData tag that invokes the expression builder.

Section

Troubleshooting

This next section covers some troubleshooting scenarios.

Error scenario 1: XPath is mapped to a Solr query that is not valid

- In the example below, the catentry_id1 field does not exist in the index.

```
<_config:queryMapping indexName="CatalogEntry"
       searchQuery="catentry_id1:(?UniqueID?)"
xpathKey="/CatalogNavigationView[CatalogEntryView[(UniqueID=)]]"/>
```

- Steps to troubleshoot:
  - Enable the trace for the com.ibm.commerce.foundation.* and com.ibm.commerce.catalog.* components
  - Find the first exception in the log:

```
[6/2/11 11:54:05:343 EDT] 0000004a SolrCore      E org.apache.solr.common.SolrException log
org.apache.solr.common.SolrException: undefined field catentry_id1
        at org.apache.solr.schema.IndexSchema.getDynamicFieldType(IndexSchema.java:1136)
        at org.apache.solr.schema.IndexSchema$SolrQueryAnalyzer.getAnalyzer(IndexSchema.java:389)
        at org.apache.solr.schema.IndexSchema$SolrIndexAnalyzer.reusableTokenStream(IndexSchema.java:364)
```

  - Search for the preceding Solr query causing the issue (search for "Final Solr query expression: "):

```
[6/2/11 11:54:05:343 EDT] 0000004a solr          1
com.ibm.commerce.foundation.internal.server.services.search.processor.solr.SolrSearchExpressionProcessor
getEntityObjects Final Solr query expression: q=(*:*) AND
(catentry_id1:(10459))&fq=storeent_id:("10101")&fq=published:1&fl=catentry_id,storeent_id,buyable,partNumber_ntk,n
ame,thumbnail,fullImage,shortDescription,longDescription,keyword,mfName_ntk,catenttype_id_ntk_cs,parentCatgroup
_id_facet,parentCatentry_id,subscripType,disallowRecOrder,price_USD&start=0&rows=50&timeAllowed=5000&debug
Query=true
```


An example error scenario is shown here. The error situation is that XPath is mapped to a Solr query that is not valid. The field used in the query does not exist in the index. When troubleshooting enable the trace shown here and find the exception searching for the Solr query that caused the issue.

More error scenarios (1 of 2)

- Error scenario 2: The XPath to Solr query mapping is missing from wc-search.xml
  - A query is formed to return everything
  - Instead of getting items requested, you will just get the first N items from the index
  - No exceptions in the log
- Error scenario 3: JSP specifies an access profile that is not mapped to a search profile in the wc-component.xml
  - The request will go through DataServiceFacade rather than SearchServiceFacade
  - The access profile is not defined in any of the template files and you will see an exception complaining about a missing query template

Two more error scenarios are shown here. These include missing steps such as the XPath to Solr query mapping is missing in the wc-search.xml and the access profile is not mapped to a search profile in the wc-component.xml.

More error scenarios (2 of 2)

- Error scenario 4: JSP specifies an access profile mapped to a search profile that does not exist
  - Example error:
    com.ibm.commerce.foundation.server.services.dataaccess.exception.ConfigurationException: CWXFS3001E: Search profile name "X_findCatalogEntryWarrantys" is not valid

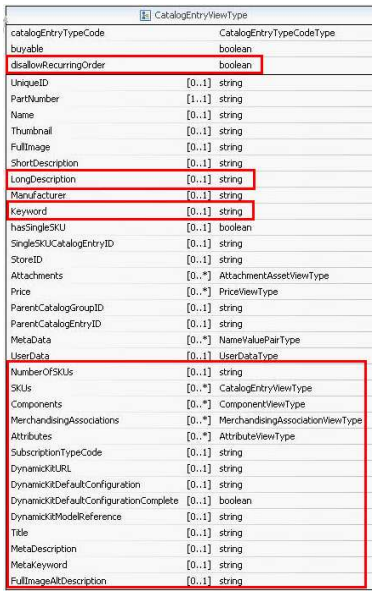
The last error scenario is shown here, an access profile that is being used in the JSP does not exist.

Section

References

The following slides are here for reference.

New element types for CatalogEntryViewType

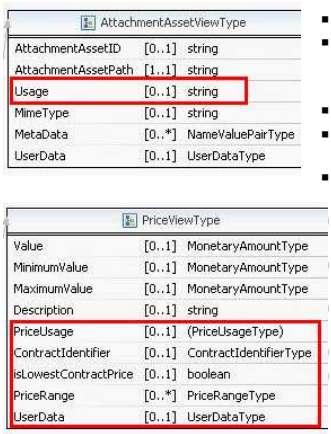


CatalogEntryViewType		
catalogEntryTypeCode		CatalogEntryTypeCodeType
buyable		boolean
disallowRecurringOrder		boolean
UniqueID	[0..1]	string
PartNumber	[1..1]	string
Name	[0..1]	string
Thumbnail	[0..1]	string
FullImage	[0..1]	string
ShortDescription	[0..1]	string
LongDescription	[0..1]	string
ManuFacturer	[0..1]	string
Keyword	[0..1]	string
hasSingleSKU	[0..1]	boolean
SingleSKUCatalogEntryID	[0..1]	string
StoreID	[0..1]	string
Attachments	[0..*]	AttachmentAssetViewType
Price	[0..*]	PriceViewType
ParentCatalogGroupID	[0..1]	string
ParentCatalogEntryID	[0..1]	string
MetaData	[0..*]	NameValuePairType
UserData	[0..1]	UserDataType
NumberOfSkUs	[0..1]	string
SkUs	[0..*]	CatalogEntryViewType
Components	[0..*]	ComponentViewType
MerchandisingAssociations	[0..*]	MerchandisingAssociationViewType
Attributes	[0..*]	AttributeViewType
SubscriptionTypeCode	[0..1]	string
DynamicKitURL	[0..1]	string
DynamicKitDefaultConfiguration	[0..1]	string
DynamicKitDefaultConfigurationComplete	[0..1]	boolean
DynamicKitModelReference	[0..1]	string
Title	[0..1]	string
MetaDescription	[0..1]	string
MetaKeyword	[0..1]	string
FullImageAltDescription	[0..1]	string

- **disallowRecurringOrder** indicates if this catalog entry can be ordered for multiple deliveries. Used for subscriptions.
- **LongDescription** is new in that it is now populated with the long description
- **Keyword** is the keyword attribute of a catalog entry
- **NumberOfSKUs** is number of entitled items for the catalog entry. Only applies to products
- **SKUs** is the entitled SKUs associated with the catalog entry when NumberOfSkus is > 0. Only applies to products.
- **Components** is the component catalog entries for the catalog entry. Applies to kits, bundles and packages
- **MerchandisingAssociations** is the merchandising associations associated with the catalog entry such as upsell, crossSell, replacement and accessory.
- **Attributes** is the defining and descriptive attributes associated with the catalog entry. For products only the defining attributes used by entitled SKUs are returned.
- **SubscriptionTypeCode** is type of subscription that this catalog entry is associated with
- **DynamicKitURL** is the URL location of an external dynamic kit configurator
- **DynamicKitDefaultConfiguration** is the reference information for an external dynamic kit configurator. If multiple values, only the first one found is returned.
- **dynamicKitDefaultConfigurationComplete** indicates if the predefined configuration is complete. Only applies to dynamic kits. If multiple values, the first one found is returned.
- **DynamicKitModelReference** will either contain a Sterling Commerce model reference or XML for a default configuration. Only applies to dynamic kits.
- **Title**, **MetaDesc**, **MetaKeyword**, and **FullImageAltDescription** is the SEO title, description, keyword and alternate description for a catalog entry image

This slide shows the new elements that have been added to CatalogEntryViewType to support the storefront service:

New or enhanced element types for AttachementAssetViewType and PriceViewType



AttachmentAssetViewType		
AttachmentAssetID	[0..1]	string
AttachmentAssetPath	[1..1]	string
Usage	[0..1]	string
MimeType	[0..1]	string
MetaData	[0..*]	NameValuePairType
UserData	[0..1]	UserDataType

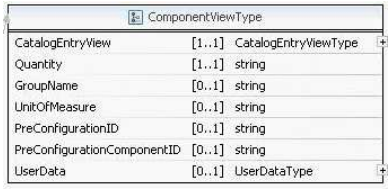
PriceViewType		
Value	[0..1]	MonetaryAmountType
MinimumValue	[0..1]	MonetaryAmountType
MaximumValue	[0..1]	MonetaryAmountType
Description	[0..1]	string
PriceUsage	[0..1]	(PriceUsageType)
ContractIdentifier	[0..1]	ContractIdentifierType
isLowestContractPrice	[0..1]	boolean
PriceRange	[0..*]	PriceRangeType
UserData	[0..1]	UserDataType

- **AttachementAssetViewType**
- **Usage** is the usage of the attachment for example ANGLEIMAGES_FULLIMAGE

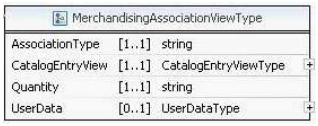
- **PriceViewType**
- **Value** is the display or offer price for quantity 1. The type of price is indicated by PriceUsage
- **MinimumValue** is the minimum unit price in the present currency. This will only be populated for Products and Bundles that have multiple SKUs.
- **MaximumValue** is the maximum unit price in the present currency. This will only be populated for Products and Bundles that have multiple SKUs.
- **Description** indicates the type of price being returned for example List/Display (“L”), Calculated Offer (“O”)
- **PriceUsage** is the price type. The supported values are "Display" for display price and "Offer" for offer/contract price
- **ContractIdentifier** is the contract identifier for the contract
- **isLowestContractPrice** is for contract offer prices. This indicates that this contract has the lowest offer price amongst all the contracts for this catalog entry
- **PriceRange** is the price ranges and associated offer prices for the contract. If the contract has only one unbounded (1-n) price range, this element is not returned
- **UserData**

This slide shows the new and enhanced elements for the AttachementAssetViewType and PriceViewType to support the storefront service:

New ComponentViewType and MerchandisingAssociationViewType



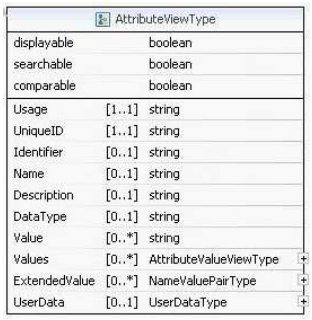
- **ComponentViewType** represents a component catalog entry
- **CatalogEntryView** is the component catalog entry
- **Quantity**
- **GroupName** is the group name for the component catalog entry
- **UnitOfMeasure** is the unit of measure for the component catalog entry. Only applies to components that are dynamic kits.
- **PreConfigurationID** is the predefined dynamic kit configuration id. Only applies to components that are dynamic kits
- **PreConfigurationComponentID** is the ID of the component in the predefined configuration. Only applies to components that are dynamic kits
- **UserData**



- **MerchandisingAssociationViewType** represents a merchandising association
- **AssociationType** is the type of association for example upsell
- **CatalogEntryView** is the associated catalog entry for this merchandising association
- **Quantity**
- **UserData**

This slide shows the new ComponentViewType and MerchandisingAssociationViewType and their elements to support the storefront service:

New AttributeViewType



AttributeViewType		
displayable		boolean
searchable		boolean
comparable		boolean
Usage	[1..1]	string
UniqueID	[1..1]	string
Identifier	[0..1]	string
Name	[0..1]	string
Description	[0..1]	string
DataType	[0..1]	string
Value	[0..*]	string
Values	[0..*]	AttributeValueViewType
ExtendedValue	[0..*]	NameValuePairType
UserData	[0..1]	UserDataType

- **AttributeViewType** represents an attribute
- **displayable** indicates attribute is displayable
- **searchable** indicates attribute is searchable
- **comparable** indicates attribute is comparable
- **Usage** is the attribute usage for example "Descriptive" or "Defining"
- **UniqueID** is the ID for attribute
- **Identifier, Name, Description** of the attribute
- **DataType** is the attribute's data type for example string
- **Value** element is deprecated and replaced with **Values**
- **Values** is an **AttributeValueViewType** which is an element that represents information about an attribute's values. For descriptive attributes there can be multiple values. For defining attributes, product can have multiple values and SKUs will have at most one. For Products, only the defining attribute values used by the entitled SKUs are returned.
  - **Value** is the attribute's value
  - **UniqueID** is the ID for attribute value
  - **Identifier** is the identifier for attribute value
  - **ExtendedValue** is the extended data for the attribute value for example Image
  - **UserData** is the user data for the attribute value
- **ExtendedValue** is the extended data for the attribute
- **UserData**

This slide shows the new AttributeViewType and its elements that have been added to support the storefront service:

Access profiles (1 of 3)

IBM_Store_Summary	catalogEntryTypeCode buyable UniqueID PartNumber Name ThumbNail FullImage ManufacturerName ShortDescription LongDescription StoreID Keyword Title MetaDesc MetaKeyword FullImageAltDescription MetaData U D
IBM_Store_Details	**IBM_Store_Summary +** disallowRecuuringOrder SubscriptionTypeCode Price ParentCatalogGroupID ParentCatalogEntryID hasSingleSku SingleSKUCatalogEntryID NumberOfSKUs SKUs A ib

Here is a list of the access profiles and the fields returned by them.

Access profiles (2 of 3)

IBM_Store_CatalogEntryDetailsWithAttachments	**IBM_Store_Details +** Attachments
IBM_Store_CatalogEntryDetailsWithComponentsAndAttachments	**IBM_Store_CatalogEntryDetailsWithAttachments +** Components DynamicKitURL DynamicKitDefaultConfiguration dynamicKitDefaultConfigurationComplete
IBM_Store_CatalogEntryDetailsMerchandisingAssociations	**IBM_Store_Details +** MerchandisingAssociations
IBM_Store_All	**IBM_Store_CatalogEntryDetailsWithComponentsAndAttachments +** MerchandisingAssociations

This list continues with the access profiles and the fields returned by them.

Access profiles (3 of 3)

IBM_Store_CatalogEntryDetailsWithAttachments	UniqueID Attachments
IBM_Store_CatalogEntryPrice	catalogEntryTypeCode UniqueID Price
IBM_Store_CatalogEntryPrcieWithAttributes	**IBM_Store_CatalogEntryPrice +** buyable StoreID PartNumber Name ThumbNail ManufacturerName ShortDescription Attributes

This list continues with the access profiles and the fields returned by them.

Expression builders and the associated access profiles

Expression Builder	Access Profile
▪getCatalogEntryViewSummaryByID ▪getCatalogEntryViewSummaryByPartnumber	IBM_Store_Summary
▪getCatalogEntryViewDetailsByID ▪getCatalogEntryViewDetailsByPartnumber	IBM_Store_Details
▪getCatalogEntryViewAttachmentsByID ▪getCatalogEntryViewAttachmentsByPartnumber	IBM_Store_CatalogEntryAttachments
▪getCatalogEntryViewPriceByID ▪getCatalogEntryViewPriceByPartnumber	IBM_Store_CatalogEntryPrice
▪getCatalogEntryViewPriceWithAttributesByID ▪getCatalogEntryViewPriceWithAttributesByPartnumber	IBM_Store_CatalogEntryPriceWithAttributes
▪getCatalogEntryViewDetailsWithAttachmentsByID ▪getCatalogEntryViewDetailsWithAttachmentsByPartnumber	IBM_Store_CatalogEntryDetailsWithAttachments
▪getCatalogEntryViewDetailsWithComponentsAndAttachmentsByID ▪getCatalogEntryViewDetailsWithComponentsAndAttachmentsByPartnumber	IBM_Store_CatalogEntryDetailsWithComponentsAndAttachments
▪getCatalogEntryViewDetailsWithMerchandisingAssociationsByID ▪getCatalogEntryViewDetailsWithMerchandisingAssociationsByPartnumber	IBM_Store_CatalogEntryDetailsWithMerchandisingAssociations
▪getCatalogEntryViewAllByID ▪getCatalogEntryViewAllByPartnumber	IBM_Store_All

Here is a table of all expression builders and the associated profiles. There is one expression builder for each access profile for both catentryID and partNumber.

Sample JSP files (1 of 2)

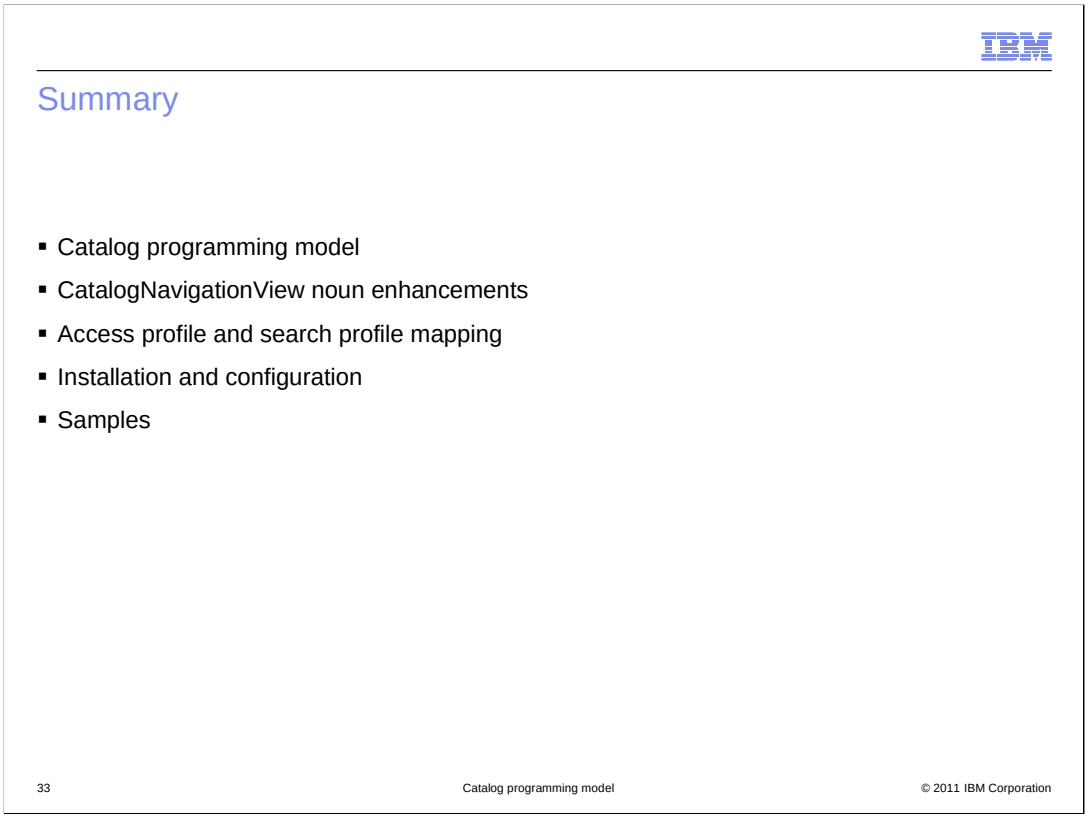
- The following sample JSP files are provided and are located under these directories in the toolkit and server code
  - **components\store-enhancements\samples\SOACatalogStorefrontServices\stores\Madisons\ShoppingArea\CatalogSection\CatalogEntrySubsection**
- Main product display JSP files call the new service using IBM_Store_All profile
  - **BundleDisplay.jsp, ItemDisplay.jsp, PackageDisplay.jsp, ProductDisplay.jsp**
- Product compare JSP files call the new service using the IBM_Store_Details profile
  - **CompareProductsDisplay.jsp**
  - **components\store-enhancements\samples\SOACatalogStorefrontServices\stores\Madisons\Snippets\Catalog\Attachments**
- The attachment display JSP uses the new service only for the site map
  - **CatalogAttachmentAssetsDisplay.jsp**
- The following JSP is new and calls the service using the IBM_Store_CatalogEntryAttachments profile to return attachments for the product display attachments tab
  - The name, short description and long description displayed in the JSP will come from the attachment target table (meta data) instead of the relationship table
  - **CatalogEntryAttachmentAssetsDisplay.jspf**
  - **components\store-enhancements\samples\SOACatalogStorefrontServices\stores\Madisons\Snippets\Catalog\CatalogEntryDisplay**

Here is some information about the sample JSP files that showcase the new noun and noun subparts for product display. These are based on the JSP files. The sample updated store front JSP files are provided to aid in understanding how to use the new storefront services and customizing it.

Sample JSP files (2 of 2)

- Product display JSP files use the new noun passed in from the calling JSP
  - Display all the main elements of the product display including price, merchandising, components, attachments and attributes on the product display page
  - **CachedBundleDisplay.jsp, CachedItemDisplay.jsp, CachedPackageDisplay.jsp, CachedProductOnlyDisplay.jsp, CatalogEntryThumbnailDisplay.jsp**
- The following JSPF is new and is used to add angle view images to the product display JSP files above
  - **AttachmentImagesDisplay.jspf**
  - **components\store-enhancements\samples\SOACatalogStorefrontServices\stores\Madisons\Snippets\Catalog\MerchandisingAssociations**
- The following merchandising display JSP uses the new service noun passed in from the calling JSP
  - Displays merchandising information on the product display page
  - **MerchandisingAssociationsDisplay.jsp**
  - **components\store-enhancements\samples\SOACatalogStorefrontServices\stores\Madisons\Snippets\ReusableObjects**
- The following price display JSPF uses the new service noun passed in from the calling JSP
  - Displays pricing information on the product display page
  - **CatalogEntryPriceDisplay.jspf**
  - **components\store-enhancements\samples\SOACatalogStorefrontServices\stores\Elite\Snippets\Catalog\CatalogEntryDisplay**
- The following contract price JSP uses the new service noun passed in from the calling JSP to display contract information in the Elite store
  - Displays contract and pricing information on the product display page in Elite
  - **B2BCatentryContractSelectExt.jspf**

Here is a continuation of the sample JSP files.



Feature Pack 3 introduces a new Catalog Programming Model service for accessing storefront catalog data. The catalog service is built on top of the WebSphere Commerce search and the CatalogNavigationView noun has been extended to support this new model. New access profiles and search profiles were created for the XPath expression. Samples were created to show you how to get your storefront catalog data following this new model.

Reference links

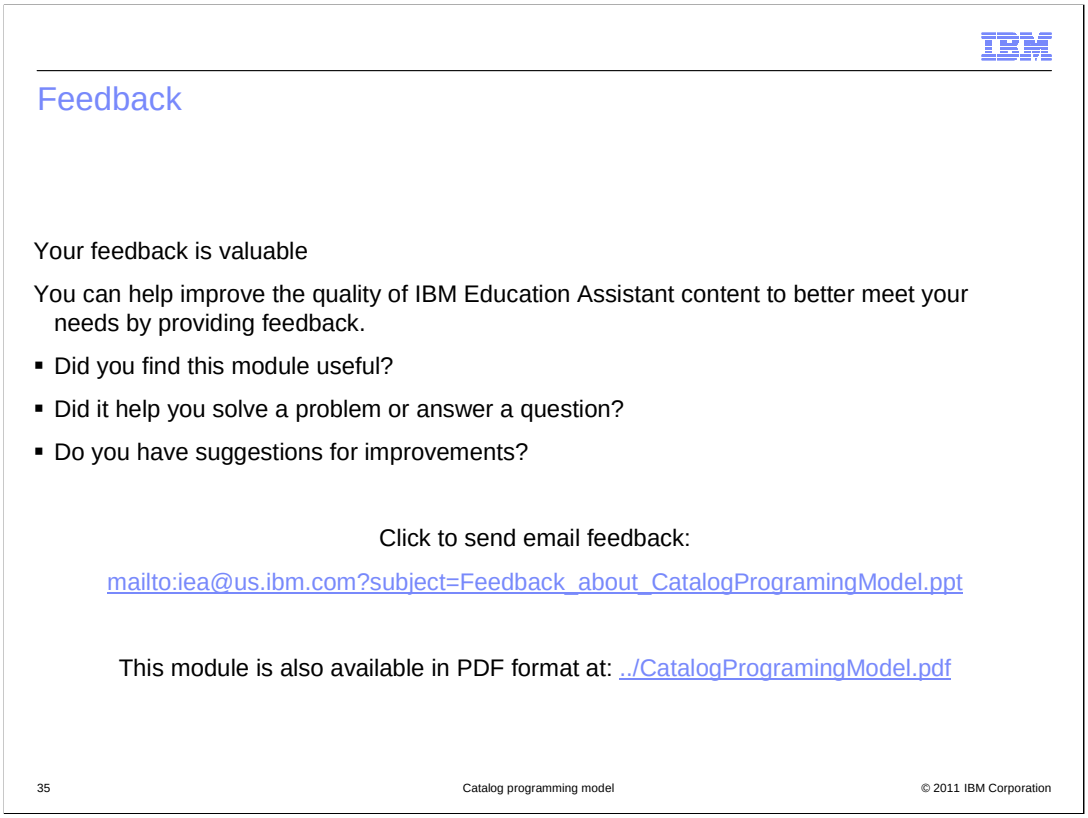
- Catalog programming model

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

- Search framework

http://publib.boulder.ibm.com/infocenter/wchelp/v7r0m0/topic/com.ibm.commerce.developer.doc/concepts/csdsearch.htm

Here are some useful links.



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

Trademarks, disclaimer, and copyright information

IBM, the IBM logo, ibm.com, and WebSphere are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of other IBM trademarks is available on the web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. Sterling Commerce, Sterling Commerce, Sterling Commerce, is a registered trademark of Sterling Commerce, Inc., an IBM Company, in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY. WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. IN ADDITION, THIS INFORMATION IS BASED ON IBM'S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE. IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION. NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, NOR SHALL HAVE THE EFFECT OF, CREATING ANY WARRANTIES OR REPRESENTATIONS FROM IBM (OR ITS SUPPLIERS OR LICENSORS), OR ALTERING THE TERMS AND CONDITIONS OF ANY AGREEMENT OR LICENSE GOVERNING THE USE OF IBM PRODUCTS OR SOFTWARE.