Introduction to Product Catalog

Presenter's Name
Presenter's Title

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Training Materials – Usage Agreement

Use of this Site (“Site”) or Materials constitutes agreement with the following terms and conditions:

1. Oracle Corporation (“Oracle”) is pleased to allow its business partner (“Partner”) to download and copy the information, documents, and the online training courses (collectively, “Materials") found on this Site. The use of the Materials is restricted to the non-commercial, internal training of the Partner’s employees only. The Materials may not be used for training, promotion, or sales to customers or other partners or third parties.

2. All the Materials are trademarks of Oracle and are proprietary information of Oracle. Partner or other third party at no time has any right to resell, redistribute or create derivative works from the Materials.

3. Oracle disclaims any warranties or representations as to the accuracy or completeness of any Materials. Materials are provided "as is" without warranty of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

4. Under no circumstances shall Oracle or the Oracle Authorized Delivery Partner be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this Site of Materials. As a condition of use of the Materials, Partner agrees to indemnify Oracle from and against any and all actions, claims, losses, damages, liabilities and expenses (including reasonable attorneys' fees) arising out of Partner’s use of the Materials.

5. Reference materials including but not limited to those identified in the Boot Camp manifest can not be redistributed in any format without Oracle written consent.

Agenda

- Overview of a Catalog
- Using the Product Catalog
- Catalog Templates

- Extending the Product Catalog
- Internationalization of Product Catalog
- Administering the Product Catalog

Learning Objectives

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

- Extend the product catalog to meet business requirements
- Add new properties to existing catalog elements
- Create new sub types from existing repository items
- Create new item types to hold custom data
- Internationalize the product catalog
- Administer the product catalog from the BCC

Section 1:

Extending the Product Catalog

Extending the Product Catalog Elements

- By default, there is only one type of category and one type of product in the catalog.
- Most sites need to modify (extend) the existing types in the catalog.
- You can:
  - Create additional properties for existing item types.
  - Create an entirely new type, whose definition is independent of existing types.
  - Create an item type that is a sub-type of an existing type.

Adding New Properties to Existing Catalog Elements

- Existing product catalog elements such as products, SKUs, and categories can be extended to add additional properties.
- Adding additional properties relies on XML combine and normal repository extension that we studied in the repository section of this guide.
- Use case:
  - Your product needs display specifications for products.
  - The specifications between products should be comparable side-by-side.
  - All products on the site require this field.

Steps to Adding a New Property (1)

- Step 1: Create a database field to hold the new value.
  - You can create a new database table and bundle all your new fields in it.

```
CREATE TABLE PRJ_PRODUCT (
  ID VARCHAR2(40 BYTE) NOT NULL REFERENCES
                             DCS_PRODUCT(PRODUCT_ID),
  . . .
  SPECIFICATION VARCHAR(255),
  . . .
  PRIMARY KEY (ID)
);
```

- It is always a good idea to create your own table instead of changing ATG tables (dcs_product).
- This allows for easy upgrade between versions.

Steps to Adding a New Property (2)

- Add new property specification to the product item definition in customCatalog.XML .

```
<item-descriptor name="product">
  <table name="PRJ_PRODUCT" type="auxiliary"
         id-column-name="id">
    …
    <property name="specification" data-type="string"
              column-name="SPECIFICATION"
              display-name="Specification"/>
    …
  </table>
</item-descriptor>
```

- The customCatalog.XML should be placed at config/atg/commerce/catalog/custom/customCatalog.XML .

Creating a New Sub-Type

- When only a subset of items require a new property and the property is not applicable to the other items in your catalog, choose to create a sub-type.
- As an example:
  - You are trying to model an electronic gift card.
  - It requires minimum and maximum balance you can put on it.
  - You can create a sub-type of product and add the new properties to it.
- Sub-type is like class extension in Java. The sub-type repository item inherits the parent repository item properties and adds to it.

Steps to Create a Product Sub-Type (1)

- The default product item definition includes an enumerated property named type can be used to create item sub-types.

```
<property name="type" data-type="enumerated"
          column-name="product_type" writable="false"
          hidden="true">
</property>
```

- To create a product sub-type, add an option value to the product type enumeration.

```
<property name="type" data-type="enumerated"
          column-name="product_type" writable="false"
          hidden="true">
  <option value="product"/>
  <option value=“electronicGiftCard"/>
</property>
```

Steps to Create a Product Sub-Type (2)

- Create a database table called PRJ_EGIFTCARD with appropriate database fields.
- Add a new item-descriptor to the repository definition file, and set its super-type attribute to the name of the parent item type.

```
<item-descriptor name="electronicGiftCard"
                 super-type="product"
                 sub-type-value="electronicGiftCard">
  <table name="PRJ_EGIFTCARD" type="auxiliary"
                      id-column-name="PRODUCT_ID">
     <property name="minCardValue" data-type="double"
            column-name="MIN_CARD_VALUE"/>
     <property name="maxCardValue" data-type="double"
            column-name="MAX_CARD_VALUE" />
  </table>
</item-descriptor>
```

Steps to Create a Product Sub-Type (3)

- The /atg/commerce/CatalogTools component needs to be updated with the new sub-type.
- The OOTB Properties are:

```
catalogFolderItemTypes=catalogFolder
catalogItemTypes=catalog
productItemTypes=product
categoryItemTypes=category
SKUItemTypes=sku,configurableSku
```

- Add new sub-type to the productItemTypes property of the CatalogTools component:

```
productItemTypes+=electronicGiftCard
```

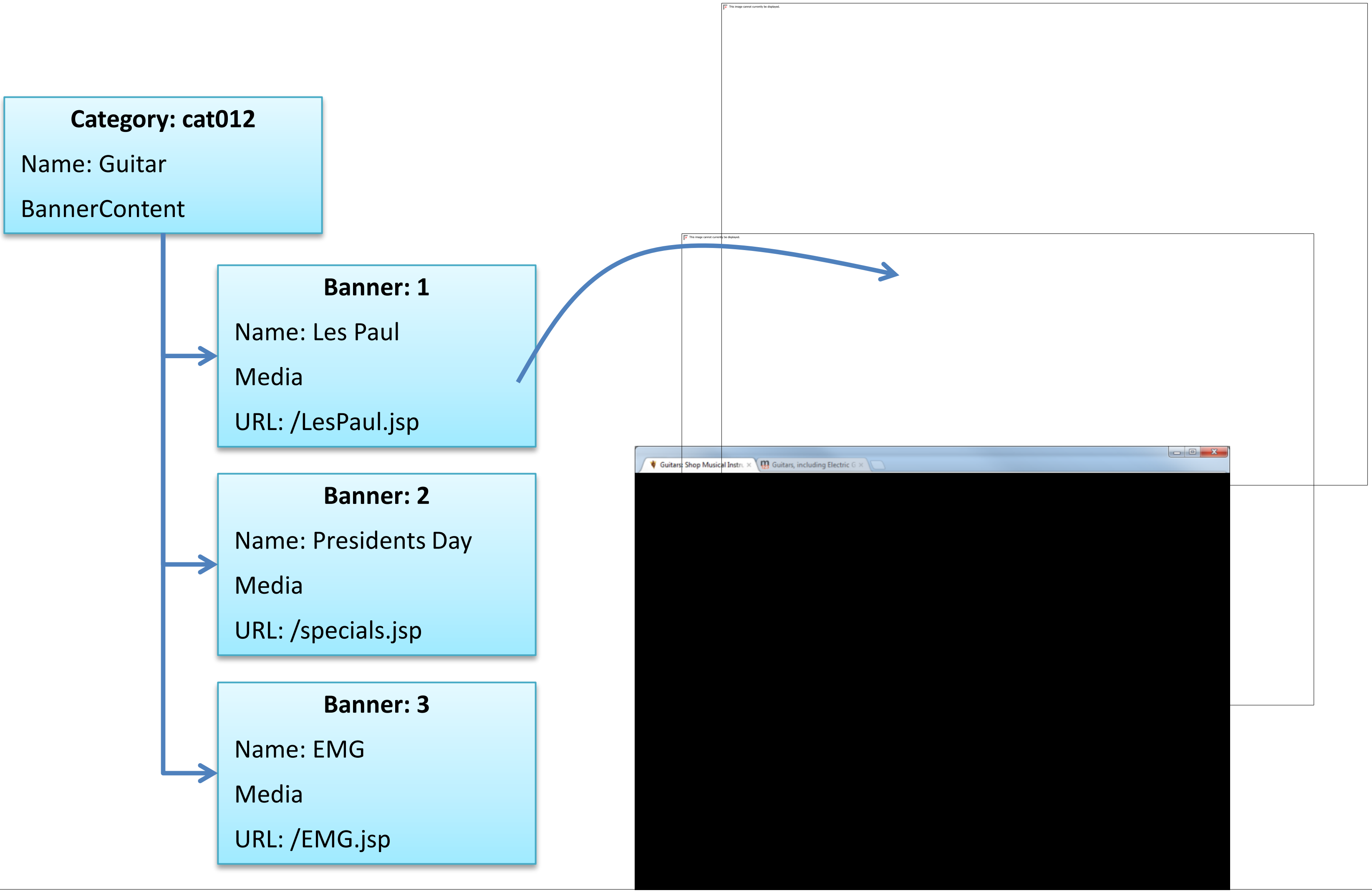
Extending Categories, SKUs, and Media Items

- The procedure outlined earlier for product item extensions and sub-types can be used for categories, SKUs, and media items as well.
- Use the appropriate property in the catalog tools to add the new sub-type.
- Generally avoid using too many levels of inheritance as queries to items will require multiple joins and affects performance.
- Create sub-types only when some instances require the property.
- Create additional properties when all instances require the property.

Adding a New Type

- In some cases you would need to add an entirely new item type to the product catalog.
- The new item type will be managed from the Business Control Center by the merchandisers.
- The new item type will not be changed by the web users by their actions.
- Use case:
  - You need a list of banners in your category repository item.
  - These banners will rotate on the page.
  - Each banner needs an image, description, and URL to go to when clicked.

Adding New Types



Creating a New Item Type (1)

- Add a primary table for bannerContent Item:

```
CREATE TABLE PRJ_BANNER_CONTENT (
  ID VARCHAR2(40) NOT NULL  PRIMARY KEY,
  TITLE VARCHAR2(40 CHAR),
  LINK_URL VARCHAR2(256 CHAR),
  MEDIA_ID VARCHAR2(40),
  FOREIGN KEY (MEDIA_ID) REFERENCES DCS_MEDIA(MEDIA_ID)
);
```

- In the pub schema you will need to add additional columns for versioning and publishing.

Creating a New Item Type (2)

- Add repository item definition for bannerContent in customCatalog.XML .

```
<item-descriptor name="bannerContent"
                 display-name="Promotional Content" …>
  <table name="PRJ_BANNER_CONTENT" type="primary"
         id-column-name="id">
    <property name="id" data-type="string"
                 column-name="ID"></property>
    <property name="title" data-type="string"
           column-name="TITLE"></property>
    <property name="linkURL" data-type="string"
           column-name="LINK_URL"></property>
    <property name="media" item-type="media-external"
           column-name="MEDIA_ID"></property>
  </table>
</item-descriptor>
```

Creating a New Item Type (3)

- Create a new table to hold the list of banners:

```
CREATE TABLE PRJ_CATEGORY_BANNERS (
  CATEGORY_ID VARCHAR2(40 BYTE) NOT NULL ENABLE,
  BANNER_ID VARCHAR2(40 BYTE),
  SEQUENCE_NUM NUMBER NOT NULL ENABLE
);
```

- Add new property banners for category item:

```
<item-descriptor name="category">
  <table name="PRJ_CATEGORY_BANNERS" type="multi"
         multi-column-name="SEQUENCE_NUM"
         id-column-name="CATEGORY_ID">
    <property name="banners" display-name="Banners"
              data-type="list" column-name="BANNER_ID"
              component-item-type="bannerContent"/>
  </table>
</item-descriptor>
```

Section 1
Check Your Understanding

What are the three ways to extend the product catalog?

Answer: Add more properties, add new item types, and create sub-types of existing product types.

Section 1
Check Your Understanding

What mechanism do you rely on to add more properties to an already defined item descriptor in the product catalog?

Answer: You use XML combine.

Section 1
Check Your Understanding

When you add a new property to an item, where should you place the database columns used to hold the data?

Answer: You should create your own auxiliary table.

Section 1
Check Your Understanding

To create a new item sub-type, what property does ATG provide in the product catalog?

Answer: ATG provides type property that can be used to define sub-types of items.

Section 1
Check Your Understanding

When should you add more properties as opposed to creating a sub-type?

Answer: When only a subset of items requires a property, use sub-type extension.

Summary

- If the default product catalog does not meet your needs, you can create more properties, add new item types, and create sub-types.
- Additional properties can be added to product catalog using XML combine.
- If only a subset of items requires a new property, you can create a sub-type of an item.
- You can create an entirely new item type that refers to or is referred by other product catalog items.
- Using these three techniques, the product catalog can be adapted to satisfy the business requirements.

Section 2

Internationalization of Product Catalog

Internationalizing the Product Catalog

- Some eCommerce websites need to serve a wider audience speaking multiple languages.
- The process of making the site available in multiple languages in multiple countries is termed as internationalization of the product catalog.
- The best practices for internationalization is to extend the commerce repository to support translated versions of required properties.
- This best practice offers:
  - The application can switch between sites, without requiring any JSP page changes.
  - No database schema changes are required to add additional languages after the internationalization framework is set up.

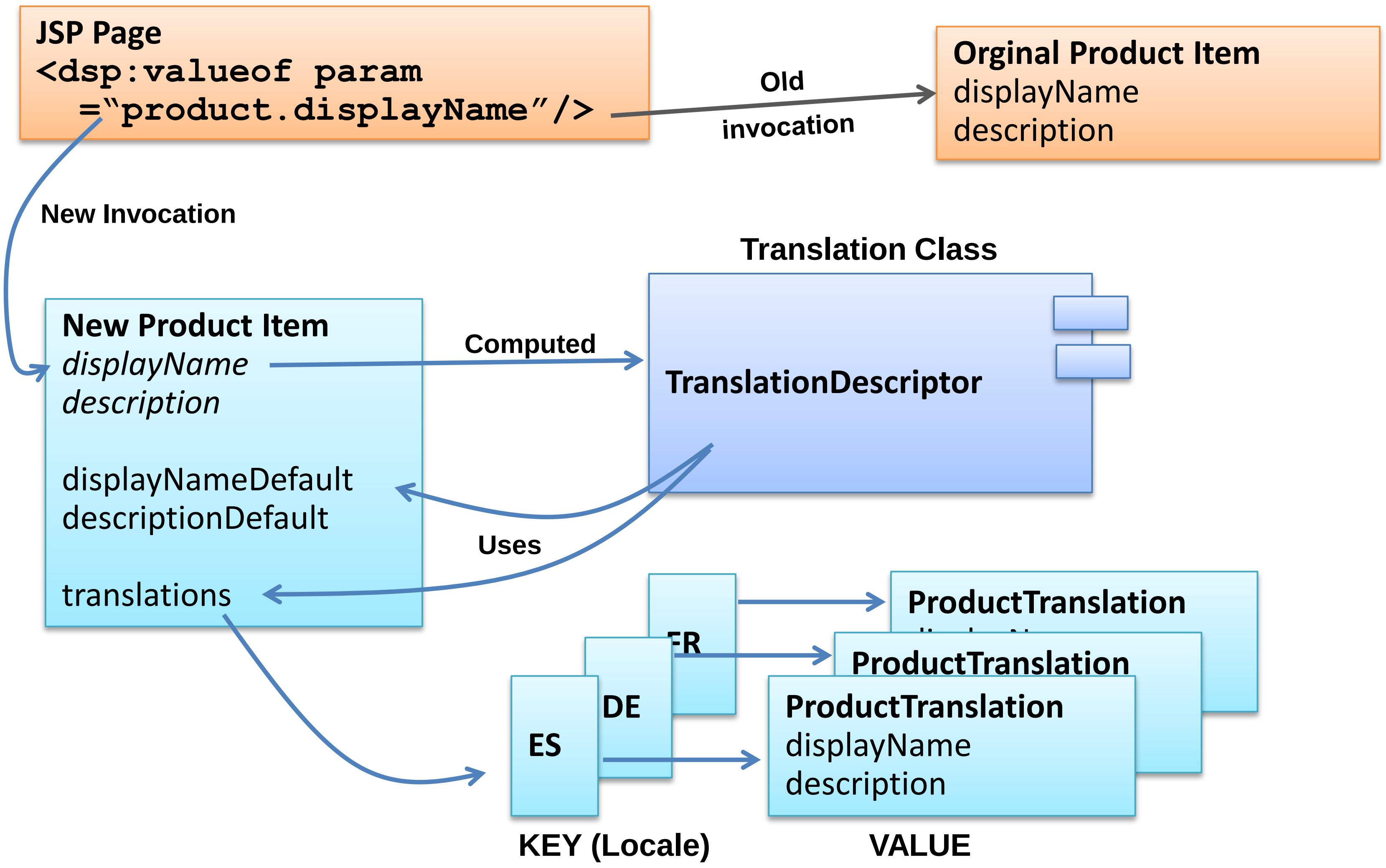
ATG Internationalizing Strategy Example

- Decide which properties of which item types should be translated.
- Category, product, SKU, or other repository items can be selected.
- As an example, we will configure translation for the displayName and description of product.
- The fundamental concept is that we will replace the original properties displayName and description to now refer to computed properties that return the locale-specific name and description.
- The JSPs then do not have to change as they can continue to refer to displayName by its old name.

Steps for Internationalization

- For each property you want to internationalize:
  - Add a new definition called displayNameDefault that points to the same column as the original definition.
  - Add a map of translations, with locale as key and the translation as value.
  - Replace the original definition of property displayName to now point to a computed property that will do the translation using the default and the map of translations.
- To hold the translations, we will create a productTranslation item descriptor that will hold the displayName and description in a different language.
- The map will have the value point to this productTranslation repository item.

Internationalization Strategy



Defining the Product Item Descriptor (1)

```
<item-descriptor name="product" display-property="displayNameDefault"
        xml-combine="append">
    <table name="dcs_product" type="primary" id-column-name="product_id">
       <property name="displayName" xml-combine="remove" />
       <property name="description" xml-combine="remove" />

       <property name="displayNameDefault" data-type="string"
              column-name="display_name" display-name="Display Name" />
       <property name="descriptionDefault" data-type="string"
              column-name="description"  display-name="Description" />

    </table>
    . . .
```

- Use XML combine to remove the original definition.
- Remap the database fields to propertyDefault format.
- We will then put the translation map in.

Defining the Product Item Descriptor (2)

```
. . .

 <table name="prj_product_product_xlate" type="multi"
       multi-column-name="locale" id-column-names="product_id">
    <property name="translations" display-name="Translations"
             column-name="translation_id" data-type="map"
             component-item-type="productTranslation"
             display-name="Translations">
     </property>
 </table>

. . .
```

- Define a map with key as locale and productTranslation as the value.
- The custom property descriptor will use this translation table as well as the default.

Defining the Product Item Descriptor (3)

```
 . . .
   <property name="displayName" data-type="string"
              property-type="atg.repository.TranslationDescriptor"
              writable="false" hidden="true" queryable="false">
      <attribute name="translationView" value="productTranslation"/>
      <attribute name="defaultProperty" value="displayNameDefault"/>
      <attribute name="defaultLocale" value="en_US"/>
      <attribute name="useStoreDefaultLocale" value="false"/>
   </property>
   <property name="description" data-type="string"
              property-type="atg.repository.TranslationDescriptor"
              writable="false" hidden="true" queryable="false">
      <attribute name="translationView" value="productTranslation"/>
      <attribute name="defaultProperty" value="descriptionDefault"/>
      <attribute name="defaultLocale" value="en_US"/>
      <attribute name="useStoreDefaultLocale" value="false"/>
    </property>
</item-descriptor>
```

- Map the properties to custom properties that will perform the locale lookup and use the appropriate translation.

Defining the Product Item Descriptor (4)

```
 <item-descriptor name="productTranslation">
    <table name="prj_product_xlate" type="primary"
            id-column-name="translation_id">
       <property name="displayName" column-name="display_name"
               data-type="string"  display-name="Display Name"/>
       <property name="description" column-name="description"
               data-type="string"  display-name="Description"/>
    </table>
</item-descriptor>
```

- Define the productTranslation repository item.
- Each productTranslation item contains the displayName and description in a specific language.
- The key (locale) in the map object of product item defines the language for which this is the translation.

Modify and Create Tables

```
CREATE TABLE PRJ_PRODUCT_PRODUCT_XLATE (
  PRODUCT_ID      VARCHAR2(40),
  LOCALE          VARCHAR2(40),
  TRANSLATION_ID  VARCHAR2(40),
);

CREATE TABLE PRJ_PRODUCT_XLATE (
  TRANSLATION_ID VARCHAR2(40),
  DISPLAY_NAME   VARCHAR2(254),
  DESCRIPTION    VARCHAR2(254),
);
```

- PRJ_PRODUCT_PRODUCT_XLATE holds the map of locale vs. the translation object.
- The PRJ_PRODUCT_XLATE holds the actual translation in another language.

Internationalizing Site Content

- Site static content should be located in two places:
  - In property file WebAppResources.properties,
  - In the database access as siteContent repository item.
- If the site content is from the database and is accessed as repository item, use the strategy already discussed in the earlier part of this section.
- This rest of this section covers internationalization when the site content is from WebAppResources .
- We will use the I18N JSTL tag libraries to achieve this.

Procedure to Internationalize Static Content

- Step 1: Create a WebAppResources.properties file.
  - This is the resource file used by the I18N module.
- Step 2: Configure the fmt setBundle function.
  - Include the command in JSP pages.
- Step 3: Use the configured values.
  - Use the fmt message tag to internationalize the web pages.

Step 1: Create the WebAppResources.properties file

- Create the file WebAppResources.properties:

```
# Localized resources for Web Application

common.search=Search
common.contactUs=Contact Us

common.button.buy=Buy

myAccount.changeMyPassword=Change my password
order.orderNumber=Your order number is {0}.
order.orderTotal=Your order total is {0}.
```

- The Spanish version of this file would be called WebAppResources_es.properties .
- It would contain the keys translated into Spanish.

Step 2: Configure the fmt setBundle

- The syntax for the fmt:setBundle call is:

```
<fmt:setBundle basename="store.web.WebAppResources" />
```

- This assumes that the file is located in /store/web/webAppResources.properties or with an additional suffix.
- Include this statement at the top of all JSP pages.
- Typically, if you have a header included, you should place it there.
- This call sets a default resource bundle for use by the fmt message tag.
- The basename attribute is required and should include the suffix or the extension.

Step 3: Use the Configured Values (1)

- In your JSP Page, instead of coding as:

```
<input type="submit" value="Search"/>
```

Code as:

```
<input type="submit"
   value='<fmt:message key="common.search"/>' />
```

- Or you can use JSTL as:

```
<fmt:message var="searchText" key="common.search">
       <dsp:input type="submit" value="${searchText}"/>
</fmt:message>
```

Step 3: Use the Configured Values (2)

- A more complex example passing parameters is:

```
# Localized resources for Web Application
order.orderTotal=Your order total is {0}.
```

- The JSP page could use this as:

```
<fmt:message key="order.orderTotal">
  <fmt:param>
    <fmt:formatCurrency value=”${orderTotal}”/>
  </fmt:param>
</fmt:message>
```

- The fmt:param value is printed instead of {0} .

Section 2
Check Your Understanding

What is internationalization?

Answer: The process of making the site available in multiple languages in multiple countries.

Section 2
Check Your Understanding

What are the advantages of the ATG internationalization strategy?

Answer: The JSP pages don’t need any changes and no database schema changes are need to add a new language.

Section 2
Check Your Understanding

What is the itemTranslation table used for?

Answer: It is used to hold the translations for each language.

Section 2
Check Your Understanding

What is the purpose of the translations map in the item descriptor?

Answer: The translation property in the item descriptor is a map that holds the key value pair of locale vs. the actual language translation.

Section 2

Check Your Understanding

What is the purpose of setBundle tag?

Answer: The fmt setBundle tag sets a default resource bundle for use by the fmt message tag.

Section 2
Check Your Understanding

What is the JSTL tag used to render content from a property file in a locale-specific manner?

Answer: The fmt message tag.

Summary

- The process of making the site available in multiple languages in multiple countries is termed internationalization of the product catalog.
- Using the ATG internationalization strategy, you can ensure that the JSP pages do not need to be recoded or data schema changed when a new language is added.
- The fundamental concept is that we will replace the original properties to refer to computed properties that return the locale-specific values.
- The item descriptor for the item being internationalized should be changed to add a key/ value translation property and a computed property that returns the translation.
- I18N JSTL tag libraries are used to internationalize static content on the site.

Section 3

Administering the Product Catalog

ATG Business Control Center

- ATG Business Control Center is a complete application for creating and maintaining the content required for ATG web applications.
- ATG BCC streamlines the process of developing site content by letting merchandisers manage all stages of the content lifecycle:
  - Creation,
  - Approval,
  - Deployment.
- ATG BCC is a browser-based interface that needs to be set up and enabled on a website.
- This section goes through the basics of managing some of the commerce elements we learned in this module.

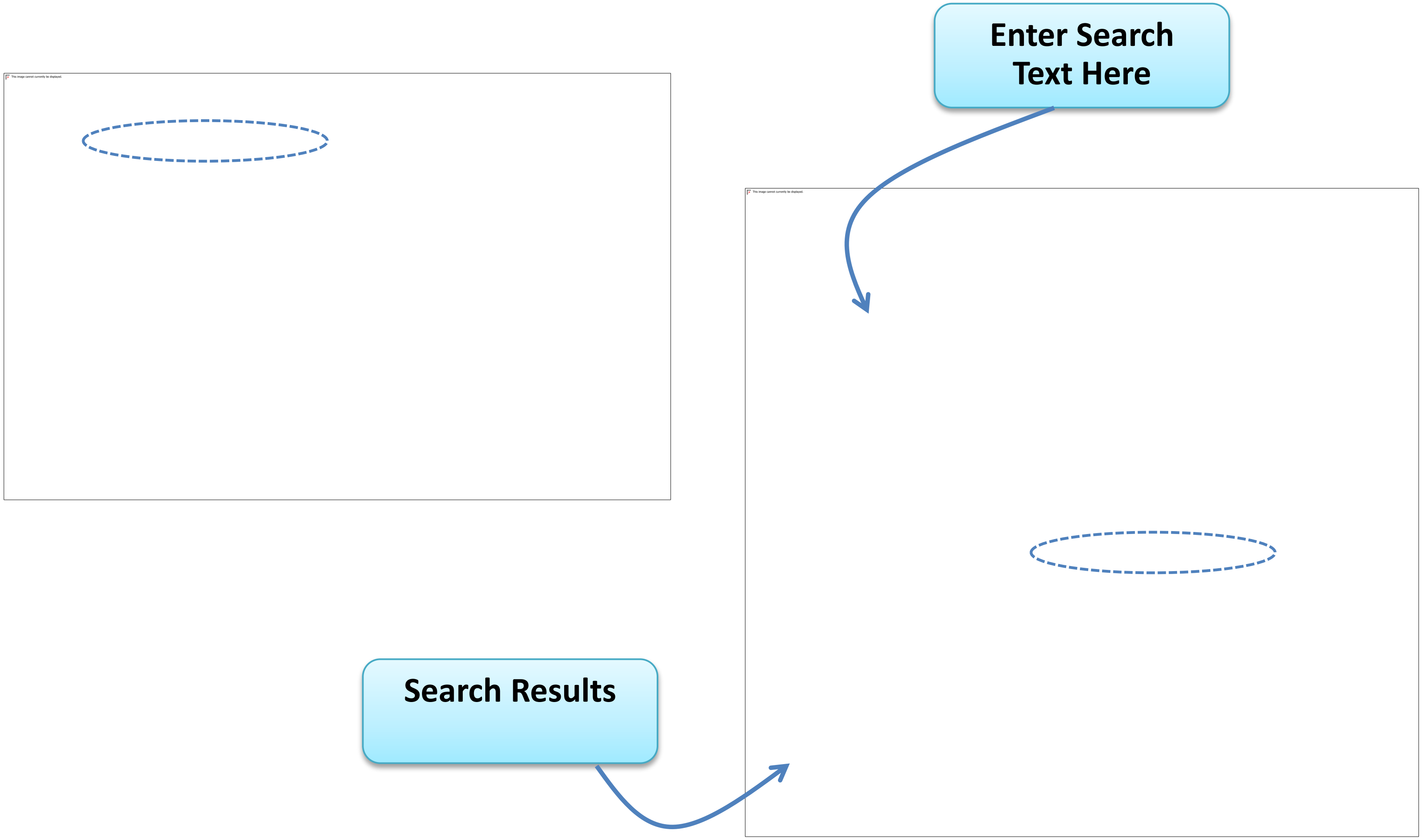
Project in the Context of BCC

- A project is a set of tasks that defines the stages of creation and publishing lifecycle.
- Typical tasks include authoring, editing, approval, and deployment.
- It is a logical grouping of items relating to a particular business goal.
- In this section, we will create a project that edits an item, creates a new item, and deploys it.
- The unit of work (project) is deployed in one transaction and can be reverted before it is accepted.

Creating a Project

This image cannot currently be displayed.
This image cannot currently be displayed.

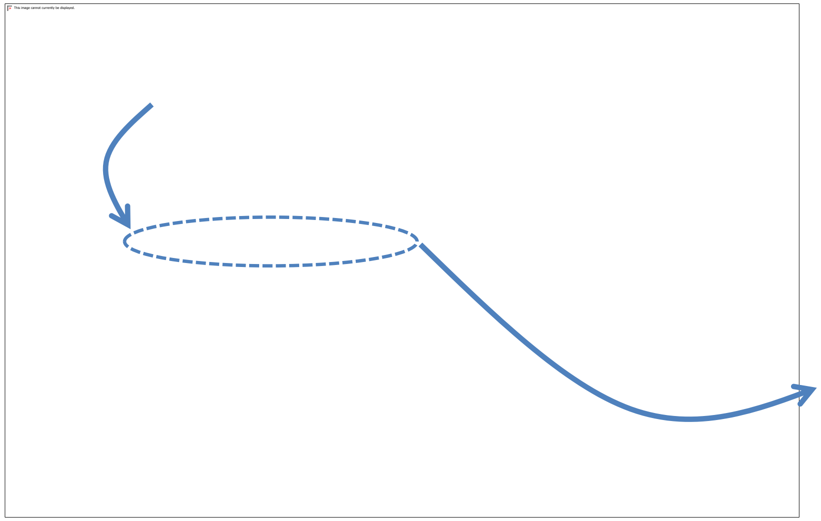
Finding Products, Categories, and SKUs

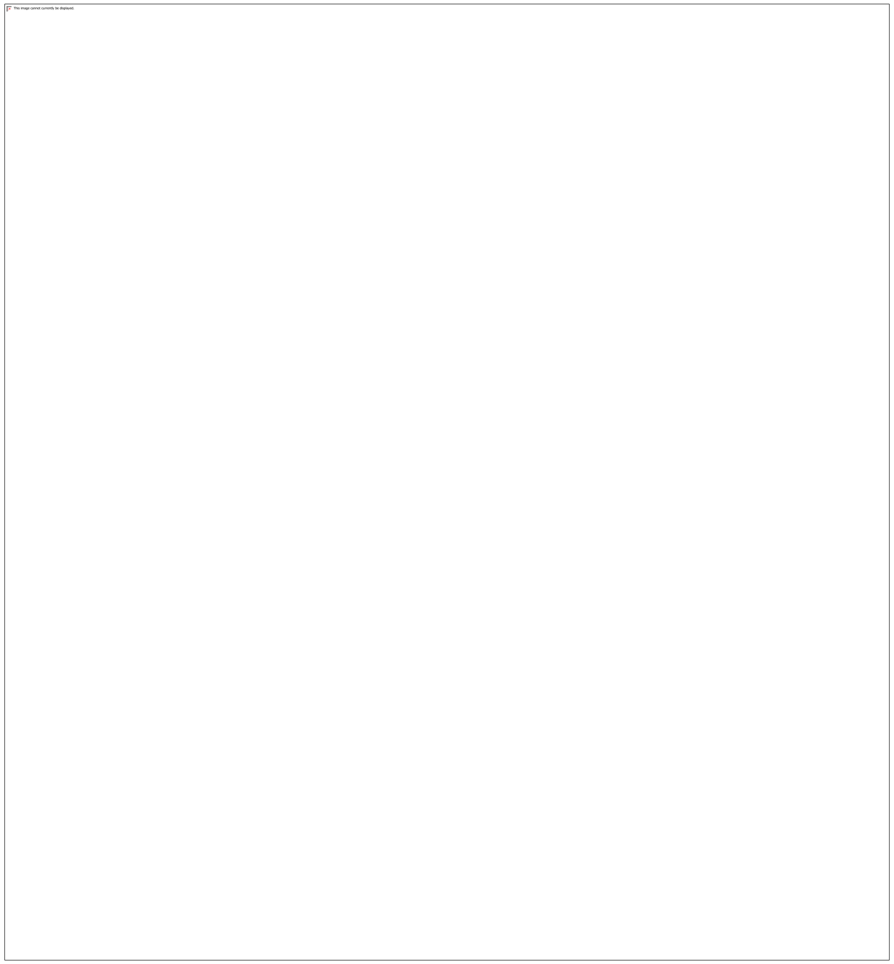


Editing Items

Edit Fields

Adding New Items





Deploying Changes

- The changes done in the project can be deployed like other projects created using the Business Control Center.
- The deployed projects go to the CATALOG schema in the commerce site.
- Items are first deployed to the inactive schema.
- The site is then switched to point to the inactive schema making it the active schema.
- Items are then deployed to the other schema which is inactive.
- This helps prevent outage to the web user.

Section 3
Check Your Understanding

The product catalog changes go to which schema in the eCommerce website?

Answer: They typically go to the CATALOG schema.

Section 3
Check Your Understanding

Does publishing the product catalog cause downtime for the customers?

Answer: Publishing product catalog changes does not cause downtime for the customer.

Section 3
Check Your Understanding

What is a project?

Answer: A project is a set of tasks that define the stages of creation and publishing lifecycle.

Section 3
Check Your Understanding

What is the smallest unit of work that can be reverted after deploying to production?

Answer: A project is the smallest unit of work that can be reverted.

Summary

- ATG Business Control Center is a complete application for creating and maintaining the content required for ATG web applications.
- Product catalog items such as catalogs, categories, products, and SKUs are managed using the BCC.
- A project is a set of tasks that define the stages of creation and publishing lifecycle.
- You can search for product catalog items using the BCC UI.
- You can edit the items in the context of a project.
- You can create new product catalog items.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

<Insert Picture Here>

Q&A

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.

This image cannot currently be displayed.