ATG Repositories UI Integration

Presenter’s Name
Presenter’s Title

Enablement 2.0
DEVELOP · SELL · IMPLEMENT

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

- RQL
- Use Repos on Page

Learning Objectives

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

- Understand RQL and express queries in RQL
- Select based on ID, Properties and multi-valued properties using RQL
- Use Order By and Range expressions
- Parameterize RQL Queries to improve efficiency
- Use ATG OOTB Droplets and FormHandlers to operate on repositories

Section 1:
RQL

RQL Overview

- ATG’s Repository Query Language (RQL) is a generic language for formulating queries that map to any repository implementation.
- It is Repository independent.
- RQL is similar to SQL:
  - Describes conditions to match items of a given type.
  - Keywords are case-insensitive.
  - Supports all standard comparison operators and logical operators.
- RQL is higher level than SQL, it is object-based, not table-based query language.

Use of RQL Queries

- RQL Queries can be used in RQL Servlet Beans in a JSP to perform repository queries.
- They can be used to define an RQL Filter that is implicitly applied to all queries performed by the repositories.
- RQL Queries can be included using the query-items tag in the XML repository definition file.
- You can use them directly by creating a RqlStatement object and running a query to return items.

Examples of RQL Queries

- A simple comparison query:
  - Query all items whose age property is greater than 30
  - age>30
- An RQL query with logical operators:
  - age>20 AND (lastName = “jones” OR paymentOverdue = true)
- Text comparison queries:
  - firstName STARTS WITH “h” AND lastName ENDS WITH “son”
- Date and timestamp queries:
  - submittedDate > date(“2012-12-24”)
- Multi-valued property queries:
  - Interests INCLUDES “biking”
- Null check and count queries:
  - phoneNumber IS NULL AND COUNT(addresses) > 1

Features of RQL Queries

- ID-based Queries
- Comparison Queries
- Text Comparison Queries
- Date and Timestamp Queries
- Property of Property Queries
- Logical Operators
- Multi-Valued Property Queries
- Full Text Search Queries

Logical Operators

- Query expressions can be combined with the following:
  - AND, OR and NOT.
- Parentheses can be used to affect grouping and precedence.
- The order of precedence:
  - NOT has the highest precedence.
  - AND has the next precedence.
  - OR has the lowest precedence.
- Example:

```
name = “joe” OR NOT phone ENDS WITH “7” AND age>30
```

- This is read as:

```
(name = "joe" OR ((NOT phone ENDS WITH "7") AND age > 30))
```

ID Based Queries

- This simple query is based on repository IDs.
- Repository IDs are not portable across repository implementations, so they should be used with care.

```
ID = “sku10123” OR ID = “sku10124”
```

- Can use the IN operator:

```
ID IN { “sku10123”, “sku10124” }
```

- In the case of content repositories:

```
IN FOLDERS { “10001”, “10002” }
```

- Composite ID can be specified with “[“:

```
ID = [“dept2”, “emp345”]
```

Comparison Queries

- Simplest RQL queries, where a property’s value is compared against another, or against a constant:

```
age > 18
```

- The following operators can be used:

=	!=
<	<=
>	>=

- They can also be used on string properties where case ordering is determined by lexical order of strings.
- They are only for scalar values.

Text Comparison Queries

- Applied to String properties to determine if a portion or all of a property’s value matches a given comparison value.

```
firstName STARTS WITH "H"
lastName ENDS WITH “son”
phoneNumber CONTAINS “33”
State EQUALS “Utah”
```

- To perform case insensitive comparison, use IGNORECASE directive.

```
firstName STARTS WITH IGNORECASE “jo”
Sports CONTAINS IGNORECASE “ball”
```

Date and Timestamp Queries

- Use date and timestamp to create date literals.

```
date("yyyy-MM-dd")
datetime("yyyy-MM-dd HH:mm:ss zzz")
```

- These can be compared to properties.

```
birthDay > date(“2000-01-01”)
shipDate > datetime(“2012-12-24 00:00:01 EDT”)
```

Property of Property Queries

- Used when a property refers to another repository item.
  - Single value properties:

```
address.zip = “90210"
```

  - Multi value properties:

```
favoriteArtists INCLUDES Artist(name STARTS WITH "B")
```

  - Multiple levels of "property-of-property" expressions:

```
department.manager.address.state = "LA"
```

Multi-Valued Property Queries (1)

- Multi-valued queries are applied to arrays or collections of scalar values:
  - INCLUDES,
  - INCLUDES ANY,
  - INCLUDES ALL (not valid for GSARepository queries).
- For example, if a property on an author is a set of strings that contains genre, you can use:

```
genre INCLUDES “sci-fi"
```

Multi-Valued Property Queries (2)

- INCLUDES:

```
interests INCLUDES "biking"
```

- INCLUDES ANY:

```
interests INCLUDES ANY { "biking", "swimming" }
```

This is equivalent to:

```
(interests INCLUDES "biking") OR (interests
INCLUDES "swimming")
```

- INCLUDES ALL:

```
interests INCLUDES ALL{ "biking", "swimming" }
```

This is equivalent to:

```
(interests INCLUDES "biking") AND (interests
INCLUDES "swimming")
```

IS NULL, COUNT, and ALL

- IS NULL is used to check if an item is null:

```
phoneNumber IS NULL
```

- COUNT operator can be used to query the size of a multi-valued property:

```
COUNT (addresses) > 1
```

- ALL returns all items in a particular item descriptor:

```
ALL
```

ORDER BY Operator

- Orders the results by item properties.
  - Ascending order:

```
age > 30 ORDER BY firstName
```

  - Descending order:

```
age > 30 ORDER BY firstName SORT DESC
```

  - Ordered by multiple properties:

```
age > 30 ORDER BY lastName, firstName SORT DESC
```

This means:

Orders results by lastName, if multiple results have the same lastName, within their group they are ordered by firstName in descending order.

  - Case insensitive sort:

```
age > 30 ORDER BY lastName SORT DESC CASE IGNORECASE
```

RANGE Operator

- Must come after the ORDER BY directive.
- Has three forms:

```
age > 30 RANGE +10
```

Returned first 10 results. If the result set is already less than 10, all results are returned.

```
age > 30 RANGE 10+
```

Skipped first 10 results, then returned remaining results.

```
age > 30 RANGE 40+10
```

Skips the first 40 results, then returns up to the next 10 results.

Parameterized RQL Queries

- The values can be substituted with parameter expressions:

```
age > ?0 AND firstName CONTAINS ?1 RANGE ?2+10
```

- Parameters can be used wherever constant values are used including RANGE expressions.
- They cannot be used in array expressions such as ID IN and IN FOLDERS.
- Parameters cannot be used to replace property names. They must be hardcoded.

Section 1 ✓
Check Your Understanding

What is the RQL Query to return all items sorted by the property firstName and return only top 10 results?

Answer: ALL ORDER BY firstName RANGE +10.

Section 1 ✓
Check Your Understanding

What is the RQL Query to check if a multi-valued property “interests” has biking and golfing in it?

Answer: interests INCLUDE ALL {"biking", "golfing"}

Section 1 ✓
Check Your Understanding

What is the RQL Query to check if a user dateOfBirth is greater than 1/1/1970 and first name is Joe or joe?

Answer: dateOfBirth > date('1970-01-01') AND firstName EQUALS IGNORECARE "Joe"

Section 1 ✓
Check Your Understanding

In the parameterized RQL Query 'interests INCLUDE ALL {?1, ?2}, what does ?1 and ?2 mean?

Answer: They are parameters passed to the Query and are replaced before execution.

Section 1 ✓
Check Your Understanding

Explain what the RQL Query 'phoneNumber IS NULL OR NOT phoneNumber CONTAINS "310" gives?

Answer: Gives all users whose phoneNumbers are null or do not contain 310 in them.

Summary

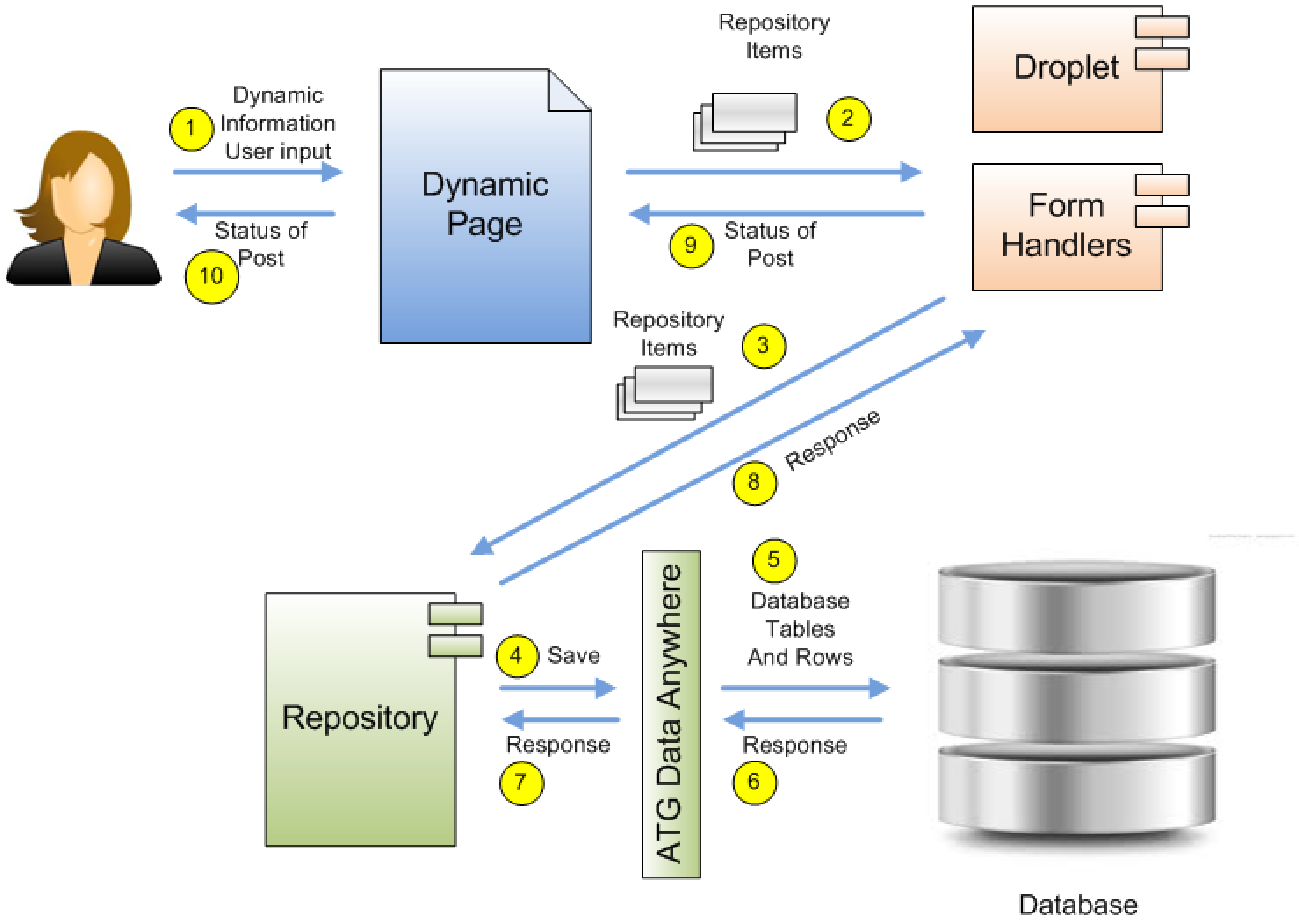
- Repository Query Language or RQL is a generic langugage for formulating queries.
- RQL Queries are higher level and return objects.
- You can use logical operators and grouping such as AND or OR.
- You can query by ID, text, or comparison type operators.
- You can query properties or repository items that are themselves properties.
- You can query multi-valued properties.
- You can order and sort results. You can define the range of the results that you are interested in.

Section 2:
Using the Repository on a JSP Page

Using ATG Repositories

- Web Applications require access to a database to display dynamic elements on a page or to store input from users.
- ATG Provides droplets and form handlers to make this task easier.
- ATG Ships with the following Droplets:
  - ItemLookupDroplet,
  - RQLQueryForEach Droplet.
- The following form handler enables data access for manipulation:
  - RepositoryFormHandler.

Using ATG Repositories



Droplets for Repository Handling

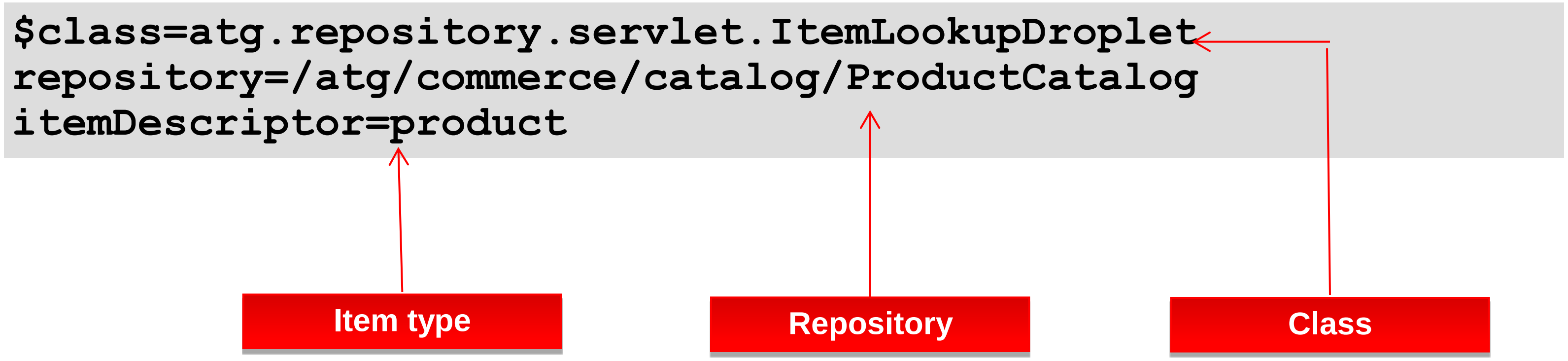
- ItemLookupDroplet:
  - Looks up an item in one or more repositories based on the item's ID.
  - Renders the item on the page.
- RQLQueryForEach:
  - Constructs an RQL query.
  - Renders its output parameter once for each element returned by the query.
- Other droplets:
  - RQLQueryRange: Renders a selected subset of the elements returned by the query.
  - ItemLink: Takes a repository item and generates a static or dynamic URL.
  - NamedQueryForEach: Used for Named Queries.

ItemLookupDroplet

- Looks up an item in one or more repositories based on the item’s ID.
  - Instance of class atg.repository.servlet.ItemLookupDroplet.
  - Configure one instance for one repository.
  - Configure one instance to define mapping from a key to an alternate set of repositories.
  - Search the default repository if it can not find item in alternate set of repositories.
- Renders the item on the page.

ItemLookupDroplet Example - Component

- OOTB droplet ProductLookup is an example of using ItemLookupDroplet.
  - Full path /atg/commerce/catalog/ProductLookup



```
$class=atg.repository.servlet.ItemLookupDroplet
repository=/atg/commerce/catalog/ProductCatalog
itemDescriptor=product
```

ItemLookupDroplet Example - JSP Code

- Get a product repository item by product id.

```
<dsp:droplet name="/atg/commerce/catalog/ProductLookup">
  <dsp:param name="id" param="productId"/>
  <dsp:setvalue param="product" paramvalue="element"
  <dsp:oparam name="output">
    <dsp:getvalueof var="pUrl" param="product.template.url"
                              vartype="java.lang.String">
      <dsp:a href="${pUrl}">
        <dsp:param name="prod_id"
             param="product.repositoryId"/>
        <dsp:valueof param="product.displayName"/>
      </dsp:a>
    </dsp:getvalueof>
  </dsp:oparam>
</dsp:droplet>
```

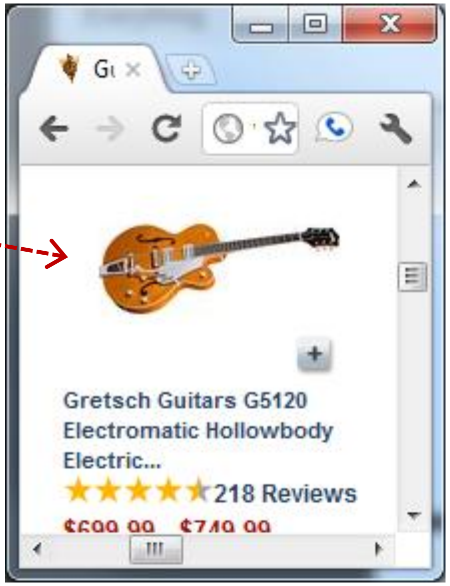
Dynamic Beans

- Repository items are dynamic beans.
- Dynamic Beans extend the standard JavaBeans specification.
- Properties can be determined at runtime:
  - Properties are defined when the XML file is parsed.
  - No setter and getter methods for each property.
  - Use DynamicPropertyMapper to get and set properties.
- Access dynamic bean properties:

```
getPropertyValue(propertyName);
setPropertyValue(beanName, beanValue);
```

Dynamic Property Mapper

Product.displayName gets mapped to getDisplayName as Product is a repository item and a dynamic Bean



```
<dsp:droplet name="/atg/commerce/catalog/ProductLookup">
  <dsp:param name="id" param="productId"/>
  <dsp:setvalue param="product" paramvalue="element"
  <dsp:oparam name="output">
    <dsp:getvalueof var="img" param="product.image"/>
    <img src="${img}" />
    <dsp:valueof param="product.displayName"/>
  </dsp:oparam>
</dsp:droplet>
```

RQLQueryForEach

- Constructs an RQL query and renders its output parameter once for each element returned by the query.
  - Required Input parameters:
    - repository (The repository to query),
    - itemDescriptor (The name of the item type to query),
    - queryRQL (The RQL query to execute).
  - Output parameters:
    - index (The zero-based index of the returned row),
    - count (The one-based number of the returned row),
    - element.

RQLQueryForEach

- The syntax for specifying a parameter:
  - RQL in RQLQueryForEach

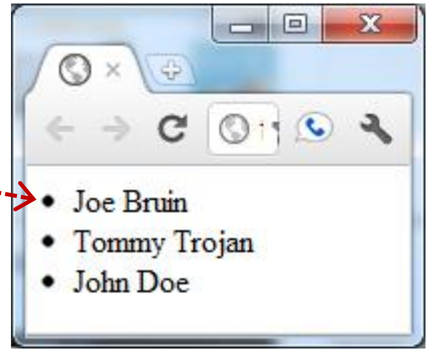
```
<dsp:param name="queryRQL"
           value="age >:whatAge"/>
```

The value for the whatAge argument is supplied through a parameter, for example:

```
<dsp:a href="http://www.example.com/over35.jsp">
   Click here if you are over 35
   <dsp:param name="whatAge" value="35"/>
</dsp:a>
```

Example: RQLQueryForEach

firstName can be accessed as repository parameters as repository items are dynamic beans.



- Joe Bruin
- Tommy Trojan
- John Doe

```
<dsp:droplet name="/atg/dynamo/droplet/RQLQueryForEach">
   <dsp:param name="itemDescriptor" value="user"/>
   <dsp:param name="repository"
              value="profileAdapterRepository"/>
   <dsp:param name="queryRQL" value="receiveEmail=true"/>
   <dsp:oparam name="output">
      <li><dsp:valueof param="element.firstName "/>
                   <dsp:valueof param="element.lastName"/>
   </dsp:oparam>
</dsp:droplet>
```

RepositoryFormHandler Overview

- Dynamic web sites commonly receive user input and display values in the database to the user.
- The RepositoryFormHandler provides methods for working with repository items.
- It is used to add/update/delete repository items.
- Offers several benefits:
  - Requires only repository item type for updates,
  - Supports any repository type such as HTML, XML, LDAP, and SQL,
  - Caches repository data,
  - Optimizes data access.

RepositoryFormHandler Properties

- Required properties:
  - class: set as atg.repository.servlet.RepositoryFormHandler
  - repository: full name of repository to be operated
  - itemDescriptorName : name of item type to be operated

```
$class=atg.repository.servlet.RepositoryFormHandler
$scope=request
repository=/atg/userprofilling/ProfileAdapterRepository
itemDescriptorName=user
```

- Several optional properties are available to control the behavior of the form handler.

Updating Items from the Value Dictionary (1)

- To update item property values from a form, the item property can be referenced as:

```
<dsp:input type="input-type"
  bean="atg/dynamo/droplet/MyRepositoryFormHandler.value.name"/>
```

- The value property is a dictionary of property/value pairs in a map like data structure.
- It is used to store all pending item property values.
- Writes all values from this dictionary as a single transaction, preventing profile properties from being left in an inconsistent state.

Value Dictionary (2)

- Hierarchical properties can be referenced in a dot notation as follows.

```
<dsp:input type="text"
        bean="MyRepositoryFormHandler.value.address.city"/>
```

- The above implies that the address item has its own property called city.

Submit Operations

- RepositoryFormHandler supports the following:
  - create: create a repository item
  - delete: delete a repository item
  - update: update a repository item
- You should associate one of these operations with submit input tag:

```
<dsp:input type="submit"
       bean="MyRepositoryFormHandler.create"/>
<dsp:input type="submit"
       bean="MyRepositoryFormHandler.delete"/>
```

- Each operation is associated with a SuccessURL and ErrorURL for navigation.
  - Example: create is associated with createSuccessURL and createErrorURL

Create Items Operation

- Create operation on RepositoryFormHandler can:
  - Create a repository item based on the value set in the form.
  - Add the created item to the repository.
- If a repositoryId property is specified, it is used. Otherwise an auto-generated ID is used.
- Create operation navigation properties:
  - createSuccessURL
  - createErrorURL

Sample of Creating New Items

- Register the RepositoryFormHandler Component:

```
$class=atg.repository.servlet.RepositoryFormHandler
$scope=request
repository=/atg/userprofileing/ProfileAdapterRepository
itemDescriptorName=user
requireIdOnCreate=false
```

- Use form tags to create a new item:

```
<dsp:form action="createForm.jsp" method="post">
   <dsp:input type="hidden" value="success.jsp"
        bean="MyRepositoryFormHandler.createSuccessURL" />
    ... add other input values
    <dsp:input type="submit"
          bean="MyRepositoryFormHandler.create" />
</dsp:form>
```

Delete Items Operation

- Delete method of the Repository form Handler deletes the item specified by repositoryID.
- Specify the item’s repositoryId in the form.
- Add delete operation redirection.
- Add delete submit input tag.

```
<dsp:form action="deleteForm.jsp" method="post">
   <dsp:input bean="MyRepositoryFormHandler.repositoryId"
        type="hidden" value="${itemId}"/>
   <dsp:input type="hidden"
        bean="MyRepositoryFormHandler.deleteSuccessURL"
        value="/deleteSuccess.jsp"/>
   <dsp:input type="submit"
        bean="MyRepositoryFormHandler.delete"/>
</dsp:form>
```

Update Items Method

- Update method updates the item described by the repositoryID.
- Set the repositoryId of the Form Handler before calling the form tag.
- Set the updated values in the form tag.
- Use the update operation.

Update Items Example

- Note the setvalue before the form tag.

```
<dsp:setvalue
  bean="MyFormHandler.repositoryId" paramvalue="itemId"/>

<dsp:form action="updateForm.jsp" method="post">
   <dsp:input bean="MyFormHandler.repositoryId"
        type="hidden" paramvalue="itemId" name="itemId"/>
   <dsp:input type="input-type"
    bean="MyRepositoryFormHandler.value.name"/>
    ... Other inputs for the item update ...
   <dsp:input type="submit" value="Update"
        bean="MyRepositoryFormHandler.update" />
</dsp:form>
```

Section 2 ✓
Check Your Understanding

What provisions does ATG provide to handle repository items from the UI?

Answer: ATG ships with droplets and form handlers to handle data from the UI.

Section 2 ✓
Check Your Understanding

Name a few servlet beans that can be used to access repository items.

Answer: ItemLookupDroplet, RQLQueryForEach droplet.

Section 2 ✓
Check Your Understanding

What is the purpose of the RepositoryFormHandler?

Answer: It is used to create, update, and delete repository items from the UI.

Section 2 ✓
Check Your Understanding

What are the required properties for an Item Lookup Droplet?

Answer: Repository and itemDescriptor are required for the item lookup droplet. In addition you can provide the id for the lookup.

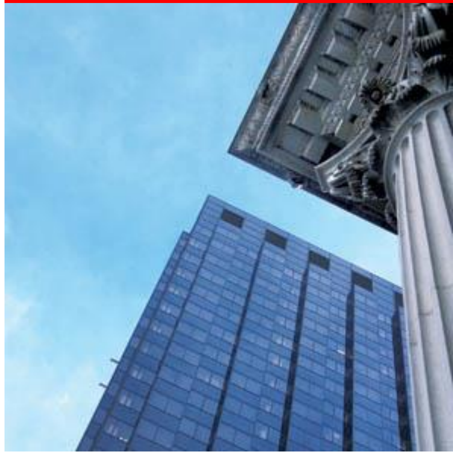
Section 2 ✓
Check Your Understanding

An Item lookup droplet returned product in the output parameter element. How do you access the name of the product on the page?

Answer: You can use element.name to access the element's name property.

Summary

- ATG provides servlet beans and form handlers to access repositories from the UI.
- The ItemLookupDroplet can be used to look up an item from the repository. The RQLQueryForEach droplet can be used to run an RQL query and display the results on the page.
- The Repository Form Handler can be used to create, delete, and update repository items.
- Repository items are Dynamic JavaBeans and can be used on the page through property values.



Q&A

ORACLE IS THE INFORMATION COMPANY

ORACLE®