ORACLE®

Introduction to Droplets

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.

Learning Objectives

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

- Identify the use of common ATG droplets.
- Embed OOTB ATG Droplets on a page passing parameters to it.
- Design custom droplets
- Create and use custom droplets
- Identify a good droplet implementation from a bad implementation
- Recognize and deal with limitations of ATG droplets.z

Agenda

- Droplet
- Custom Droplets

Section 1:
Droplets

ATG Droplet

- Most applications require a way to generate HTML from a Java Object.
- A Droplet is an ATG Servlet Bean that satisfies this requirement.
- It is a nucleus component that can be configured and used to encode business logic that should not be in the JSP (view) layer.
- ATG Droplets allow for clear separation of model, view, and controller.
- For example, it can be used to access the database for the user profile and enumerate users' preferences stored in the database.

Using ATG Droplets

- ATG ships with a range of droplets to perform common tasks.
- These droplets can be categorized as:
  - Standard servlet Beans.
  - Database and repository access servlet beans.
  - Multisite servlet beans.
  - XML servlet beans.
  - Transaction servlet beans.
  - Personalization servlet beans.
  - Business process tracking servlet beans.

Some Frequently Used Droplets

Class Name	Component Path	Description
atg.droplet.ForEach	/atg/dynamo/droplet/ForEach	Displays each element of an array.
atg.droplet.IsNull	/atg/dynamo/droplet/IsNull	Displays one of two possible outputs, depending on whether its input parameter is null.
atg.droplet.Switch	/atg/dynamo/droplet/Switch	Displays one of a set of possible outputs, depending on the value of its input parameter.
atg.droplet.CurrencyFormatter	/atg/dynamo/droplet/CurrencyFormatter	Displays a numeric value as a currency amount, formatting it based on the locale.

The ForEach Droplet

- The ForEach Droplet is used to iterate over the elements of an array property.
- ForEach droplet is a nucleus component.
  - Class Name: atg.droplet.ForEach
  - Component: /atg/dynamo/droplet/ForEach
- Specify the desired HTML for displaying each element.
  - Specify the HTML to display before and after array processing.
  - Specify the HTML to display if the array has no elements.

Example - ForEach Droplet (Cont.)

- The example below uses the ForEach droplet to loop through the elements in the array property.
- The param array is the input property.
- The param element is the output property.
- The param output and outputStart are the open parameters.

```
<dsp:droplet name="/atg/dynamo/droplet/ForEach">
  <dsp:param name="array" bean="/samples/StudentArray"/>
  <dsp:oparam name="outputStart">
    <p>The student is registered for these courses:</p>
  </dsp:oparam>
  <ul>
  <dsp:oparam name="output">
    <li><dsp:valueof param="element"></dsp:valueof></li>
  </dsp:oparam>
  </ul>
</dsp:droplet>
```

Embedding Droplets into JSPs

- **dsp:droplet** tag is used to embed the droplet in the JSP.

```
<dsp:droplet name="/atg/dynamo/droplet/ForEach">
  …
</dsp:droplet>
```

- Encapsulates programming logic in a server-side JavaBean.
- Makes logic accessible to the jsp page that calls it.
- Important attribute name:
  - name: Sets the Nucleus path and droplet to invoke.
  - param: Used to pass in/out params.
  - oparam: Used to pass open parameters.

Droplet Parameter Types

- Each droplet has a predefined set of parameters that controls the droplet's behavior. Droplets have three types of parameters:
  - Input
    - Passed into the droplet .
  - Output
    - Set by the droplet.
  - Open
    - Different stages of droplet processing.

Input Parameters

- dsp:param
  - Identifies a droplet input parameter.
  - Defines a page parameter.
- Example:
  - Input parameters accepted by ForEach Droplet which must be set by dsp:param tag:
    - array
    - sortProperties

```
<dsp:droplet name="ForEach">
  <dsp:param name="array" bean="/samples/StudentArray"/>
   …
</dsp:droplet>
```

Output Parameters

- Droplet can render data to JSP page.
- Example:
  - Output parameters provided by ForEach Droplet which can be accessed JSP/DSP Tags:
    - index
    - count
    - key
    - element
    - size

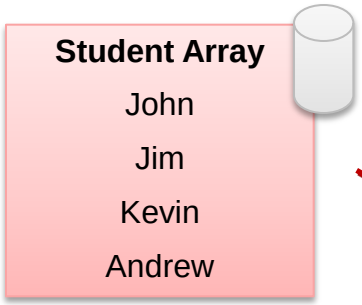
```
<li>
   <dsp:valueof param="element"></dsp:valueof>
</li>
```

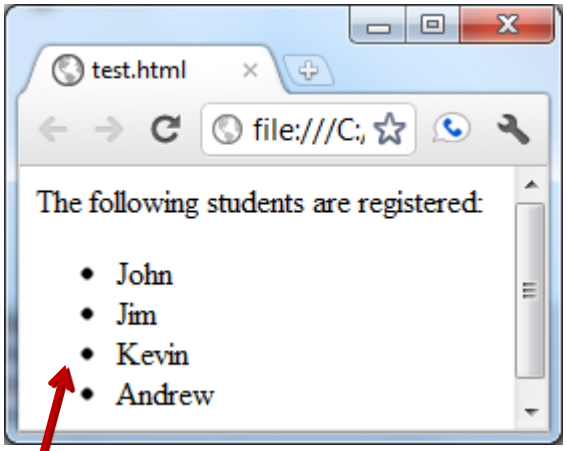
Open Parameters

- dsp:oparam
  - Supplies a parameter to the current droplet.
  - Gets processed and rendered by the droplet.
- Example:
  - Open parameters supported by ForEach Droplet are set using dsp:oparam tag:
    - output
    - outputStart
    - outputEnd
    - empty

```
<dsp:droplet name="ForEach">
  . . .
  <dsp:oparam name="output">
   ...
  </dsp:oparam>
</droplet>
```

Putting it all together






```
<dsp:droplet name="/atg/dynamo/droplet/ForEach">
  <dsp:param name="array" bean="/samples/StudentArray"/>
  <dsp:oparam name="outputStart">
    <p>The following students are registered:</p>
  </dsp:oparam>
  <ul>
  <dsp:oparam name="output">
    <li><dsp:valueof param="element"></dsp:valueof></li>
  </dsp:oparam>
  </ul>
</dsp:droplet>
```

Using the IsNull Droplet

- The parameter value is the input parameter.
- The oparam true only renders if value is not null.
- In addition oparam false is also supported.

```
<dsp:droplet name="IsNull">
  <dsp:param bean="MyProfile.email" name="value"/>
  <dsp:oparam name="true">
    <dsp:form action="address_book.jsp" method="POST">
       My email address:
       <dsp:input type="text"
                bean="MyProfileFormHandler.email"/>
       <dsp:input type="submit"
         bean="MyProfileFormHandler.update value="Update"/>
    </dsp:form>
  </dsp:oparam>
</dsp:droplet>
```

Section 1 
Check Your Understanding

What nucleus component can be embedded on a page to generate dynamic HTML content?

Answer: Droplets are nucleus components that encapsulate business logic and can be embedded on JSP Pages to generate dynamic HTML content.

Section 1 ✔
Check Your Understanding

Name a few commonly used droplets.

Answer: ForEach, IsNull, ItemLookdroplet are some commonly used examples of droplets.

Section 1 
Check Your Understanding

What is the ForEach droplet used for?

Answer: The ForEach droplet is used to iterate through an array or objects and display them.

Section 1 
Check Your Understanding

What type of parameters does any droplet accept?

Answer: A droplet typically accepts some or all of input, output, and open parameters.

Section 1 
Check Your Understanding

What tag do you use to embed a droplet on a page?

Answer: The dsp:droplet tag is used to embed a droplet on the page.

Summary

- ATG Servlet Beans or Droplet are used to encapsulate business logic.
- They can be embedded on the page to generate html dynamically.
- ATG ships with many droplets that can facilitate various activities.
- Droplets have input parameters, output parameters, and open parameters.
- Common droplets include ForEach, IsNull, IsEmpty, ItemLookupDroplet, etc.

Section 2:
Custom Droplets

Why create Custom Droplets?

- ATG ships with many droplets that provide a convenient mechanism to perform typical actions.
- Custom droplets allow you to extend this capability.
- Custom droplets can perform any action that Java Code can achieve.
- They provide an excellent mechanism to separate business logic from the view (JSP).
- By using the droplet tag and the input, output, and open parameters, complex business logic can be extracted out and reused.

Creating a New Custom Droplet

- We will execute the following steps to create a custom droplet:
  - Create a java class.
  - Register the class as a nucleus component.
  - Use the droplet tag to embed it on the java page.

- This example will use the following:
  - Send an input parameter called storename to the droplet.
  - Use the out.println method to print HTML on the page.

DropletTest.java Java Class

```
package test;

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import atg.servlet.*;

public class DropletTest extends DynamoServlet {
  public DSBTest () {}
  public void service (DynamoHttpServletRequest request,
                       DynamoHttpServletResponse response)
                  throws ServletException, IOException {
    String storename = request.getParameter ("storename");
    if (storename == null) storename = "No-Name's";

    ServletOutputStream out = response.getOutputStream ();
    out.println ("<h1>Welcome to " + storename + "</h1>");
  }
}
```

The DynamoServlet

- Note that the Droplet class:
  - extended DynamoServlet class.
  - The service method took DynamoHttpServletRequest and DynamoHttpServletResponse objects as parameters.
- DynamoServlet , DynamoHttpServletRequest, and DynamoHttpServletResponse are subclasses of Servlet, HttpServletRequest, and HttpServletResponse classes.
- They extend and add several functions to access ATG Servlet functionality.
- DynamoServlet extends GenericService which allows it to be a nucleus component.
- Any class implementing the standard Servlet interface can be invoked using the droplet tag.

Registering the Droplet

- Droplets should be registered as nucleus components.
- Create a folder called “Test” in a folder that is in the CONFIGPATH of ATG.
- The class name is test.DropletTest.
- The scope is global as no state is being kept by the servlet.

/test/DropletTest.properties

```
### /test/DropletTest file
$class=test.DropletTest
$scope=global
```

The DropletTestJSP Page


```
<%@ taglib uri="/dspTaglib" prefix="dsp" %>
<dsp:page>
<html>
  <head><title>DSB Store Test</title></head>
  <body><h1>DSB Store Test</h1>
    <dsp:droplet name="/test/DropletTest">
      <dsp:param name="storename" value="Joe's Hardware"/>
    </dsp:droplet>
  </body>
</html>
</dsp:page>
```

Passing Parameters to Droplets

- String and object parameters can be passed to droplets with the param tag.

```
<dsp:droplet name="MyDroplet">
  <dsp:param name="storename" value="Joe's Store"/>
  <dsp:param name="userAge" bean="/test/Person.age"/>
<dsp:droplet>
```

- String parameters can be read in droplet as:

```
String name = request.getParameter ("storename");
```

- Object parameters can be read in droplet as:

```
Object age = request.getObjectParameter ("userAge");
```

Displaying Open Parameters

- A good way to separate code and display markup is to use open param servicing.

```
<dsp:droplet name="MyDroplet">
  <dsp:param name="storename" value="Joe's Store"/>
  <dsp:oparam name="welcome"/>
    <h1>Welcome to Joe's store</h1>
  <dsp:oparam>
<dsp:droplet>
```

- In the droplet code you can render this as:

```
request.serviceParameter("welcome", request, response);
```

- The service method obtains the value of the given parameter and displays it.
- A droplet can contain multiple open parameters. The java code can conditionally service one or more of these.

Setting Parameters

- When your ATG Servlet bean displays open parameters (oparam), dynamic elements can be passed to it.

```
<dsp:droplet name="MyDroplet">
  <dsp:param name="storename" value="Joe's Store"/>
  <dsp:oparam name="welcome"/>
    <h1>Welcome to <dsp:valueof param="storename" />.
    </h1>
  <dsp:oparam>
<dsp:droplet>
```

- A Droplet can add or change parameters using:

```
request.setParameter("storename", "Mike's Store");
```

- The above setParameter call overrides the original definition and sets a new value to it.
- These can be used as output parameters.

Local Parameters

- getLocalParameter and serviceLocalParameter can be used to create parameters that are visible only locally to an ATG Droplet.

```
<dsp:param name="notLocalForA" value="Some Store"/>
<dsp:droplet name="DropletA">
  <dsp:param name="localForA" value="Joe's Store"/>
<dsp:droplet>
```

- notLocalForA above is not local for the DropletA above.
- Local Parameters are useful to contain scope and ensure that you are reading and setting values that are specific for your droplet.

Best Practice: Separation of Java and JSP Code

- Ability to set parameters and render open parameters discussed in the previous slides provides a mechanism to separate HTML Markup from Java Code.
- JSP Formatting should be separated from Java Code.
- This allows JSP designers and Java Codes to work together and maintain autonomy.
- This also allows for reuse of droplet code.

Example of a Bad Droplet

- The code below includes HTML Markup that can change.
- This forces the designer to ask a Java Developer to change and recompile the Java Class.

```
public void service (DynamoHttpServletRequest request,
                     DynamoHttpServletResponse response)
        throws ServletException, IOException {
  ServletOutputStream out = response.getOutputStream ();
  out.println ("<ul>");
  for (int i = 0; i < 10; i++) {
    out.println ("<li>This is number " + i);
  }
  out.println ("</ul>");
}
```

A Better Way to Write the Droplet

- The Java Class

```
public void service (DynamoHttpServletRequest request,
                     DynamoHttpServletResponse response)
        throws ServletException, IOException {
  ServletOutputStream out = response.getOutputStream ();
  for (int i = 0; i < 10; i++) {
    request.setParameter ("number", new Integer (i));
    request.serviceParameter ("lineformat", request, response);
  }
}
```

- The JSP Code

```
<ul>
  <dsp:droplet name="/test/Counter2">
    <dsp:oparam name="lineformat">
       <li>This is number <dsp:valueof
             param="number"/>
    </dsp:oparam>
  </dsp:droplet>
</ul>
```

Limitations of ATG Custom Droplets

- Open parameters are not executed precisely when their service Parameter is called.
- Open parameters are called after the service method returns.
- In general avoid:
  - Setting a global or thread-state variable that is accessed by code invoked from an open parameter.
  - Opening or closing a socket or JDBC that is accessed by code invoked from an open parameter.
  - Replacing the output stream/print writer in the response with your own designed to capture the output of an open parameter.

Section 2 ✔
Check Your Understanding

What are the steps for creating a custom droplet?

Answer: Write the Java Class, register the droplet with nucleus, and embed it on the page.

Section 2
Check Your Understanding

How do you pass a parameter to the droplet from the page?

Answer: Use the dsp:param tag.

Section 2 
Check Your Understanding

What types of parameters can be passed to the droplet?

Answer: You can pass string or any other object types. getParameter can access the string parameters and getObjectParameter can access the object parameters.

Section 2 
Check Your Understanding

What are open parameters?

Answer: Open parameters are parameters that can be conditionally rendered from the Java code.

Section 2 
Check Your Understanding

What method is used to output parameters ?

Answer: SetParameter method is used to set a parameter that is visible to the open params.

Summary

- Custom droplets extend the capability of ATG droplets.
- Custom droplets are used to separate code from markup.
- Custom droplets are nucleus components.
- Parameters can be passed to and from droplets.
- Open parameters can be rendered in java code.

ORACLE IS THE INFORMATION COMPANY


ORACLE®