Chapter

9

Embedding report content in a page

Slide 63 Embedding report content in a page

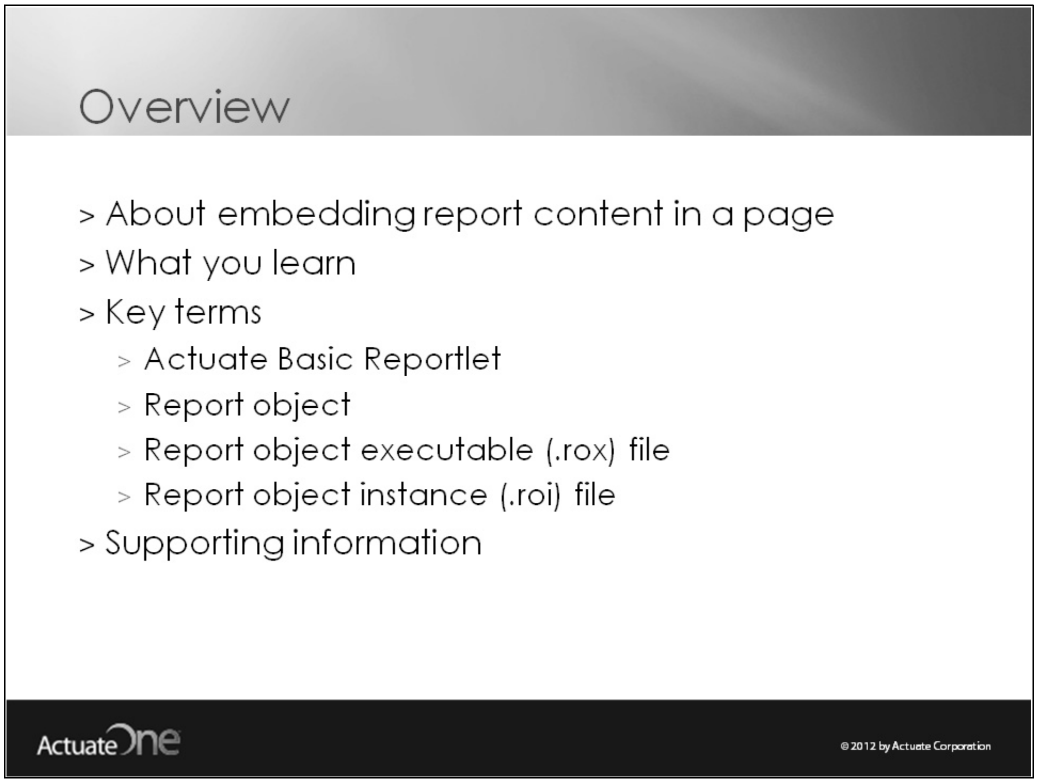
Actuate One

Embedding report content in a page

One Design
One Server
One User Experience

©2012 by Actuate Corporation

Slide 64 Overview


Overview

> About embedding report content in a page
> What you learn
> Key terms
  > Actuate Basic Reportlet
  > Report object
  > Report object executable (.rox) file
  > Report object instance (.roi) file
> Supporting information

Actuate One

© 2012 by Actuate Corporation

About embedding report content in a page

Information Console supports embedding report content in a page in a streamlined and modular way, using various mechanisms. These mechanisms include JSP built-in tags, Jakarta Struts custom tags, and Information Console custom tags. You studied JSP built-in tags in Chapter 6, "Accessing, modifying, and creating a page." You used Jakarta Struts template tags in Chapter 7, "Customizing presentation architecture." You used other Jakarta Struts tags in Chapter 8, "Customizing navigation." This chapter discusses how custom tags are made available in Information Console and how to call Information Console custom tags to display report content.

To customize pages in an Information Console web application, use Actuate custom tags. For example, use the Reportlet tag to include a portion of an Actuate Basic report in a web page. Most of the tags modularize functionality that is available within the Information Console web application. These tags are typically used to add Information Console functionality to an arbitrary web application. For example, Actuate custom tags support managing user data, internationalization, report viewing, and other common tasks.

What you learn

In this chapter, you learn to:

- Describe custom tags.
- Specify inclusion of custom tag libraries.
- Use Actuate custom tags in JSPs.
- Incorporate an Actuate Basic Reportlet in a JSP.

Key terms

Actuate Basic Reportlet
In Actuate Basic technology and Actuate BIRT, a portion of a report that can be embedded in a web page.

Report object
A component that contains all other components in a report. The report object is the root of the report structure in Actuate e.Report Designer Professional. The relevant Actuate Foundation Class is AcReport.

Report object executable (.rox) file
In Actuate Basic technology, a binary file type that contains the instructions for generating and viewing a report document. e.Report Designer Professional creates this executable form of a report design when the user compiles the Actuate Basic source (.bas) file for the report.

Report object instance (.roi) file
A set of persistent objects that represents a particular report.

Supporting information

For more information about using Actuate Basic Reportlets and Actuate custom tags in Information Console, see *Information Console Developer Guide.*

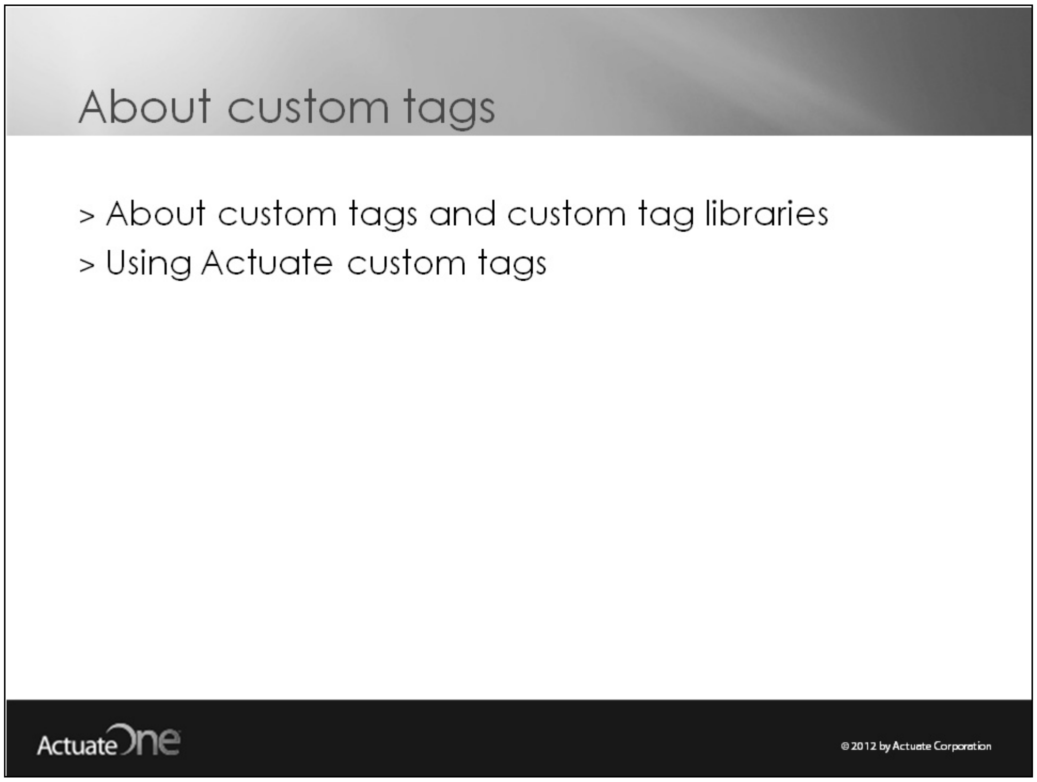
For more information about creating Actuate Basic Reportlets, see *Developing Reports using e.Report Designer Professional.*

For a description of all attributes for the tags, see *Information Console Developer Guide.*

For more information about how to specify components, see *Developing Reports using e.Report Designer Professional.*

For more information about report components, see *Developing Reports using e.Report Designer Professional.*

Slide 65 About custom tags


About custom tags

> About custom tags and custom tag libraries
> Using Actuate custom tags

Actuate One
© 2012 by Actuate Corporation

About custom tags and custom tag libraries

JavaServer Page technology includes built-in tags and tag libraries. JSP technology also supports creating custom tags and tag libraries. A custom tag library is a group of custom tags and a tag library descriptor (TLD). Tag libraries are XML files that specify the names of the tags, the handler classes associated with the tags, and other information about the tags.

Using Actuate custom tags

Information Console uses the built-in tag libraries, Jakarta Struts custom tag libraries, and Actuate custom tag libraries. These tag libraries are located in the following directory:

```
<context root>\WEB-INF
```

In the same directory, web.xml defines a URI and the location for each tag library that Information Console uses. To specify that Information Console use an additional tag library, add an entry for that tag library to web.xml. The JSP run-time engine uses web.xml to obtain the location of any TLDs requested in a taglib directive. The <taglib-uri> element provides the name of the tag library. The <taglib-location> element provides the path and file name for the tag library file. The following tag library entry in web.xml enables use of the Reportlet tag library:

```
<taglib>
   <taglib-uri>
      /reportlet
   </taglib-uri>
   <taglib-location>
      /WEB-INF/reportlet.tld
   </taglib-location>
</taglib>
```

The reportlet.tld tag library supports customizing a web application to include a portion of a report in a page. Most Actuate custom tags support backward compatibility.

Slide 66 Using custom tags

Using custom tags

> Specifying inclusion of a custom tag library
> Using a custom tag in a JSP

ActuateOne
© 2012 by Actuate Corporation

To use a custom tag, place a taglib directive in a JSP to access the custom tag library. Then, place that tag within that JSP.

Specifying inclusion of a custom tag library

The taglib directive declares that the JSP use a custom tag library. The taglib directive specifies the URI of the custom tag library defined in web.xml and the tag prefix used to reference a library tag in the JSP. The following taglib directive specifies the /reportlet tag library and uses the reportlet prefix within the JSP:

```
<%@ taglib uri="/reportlet" prefix="reportlet" %>
```

The prefix functions as a label to reference a tag in the library. The prefix parameter supports using identically named tags from two different libraries in a JSP without conflict. The taglib directive can also specify Jakarta Struts custom tags. For more information about these tags, see Chapter 7, "Customizing presentation architecture."

Using a custom tag in a JSP

After making a custom tag library accessible to a JSP, to use the tag, specify the prefix and the tag name. The simplest syntax for a custom tag is shown in the following example:

```
<library-prefix:tag-name/>
```

A tag has access to every object available to each JSP that calls the tag. A custom tag call can also include attributes that specify additional information for use by the tag. To assign a value to a tag attribute, structure the call to the custom tag using the following syntax:

```
<library-prefix:tag-name attribute-name="attributevalue"
   [attribute-name="attributevalue"]/>
```

The following example contains a call to a custom tag that determines page count for a report:

```
<viewer:getPageCount authID="<%= sAuthID %>" volume="<%= sVolume %>"
   serverURL="<%= sServerURL %>" locale="<%= sLocale %>"
   connectionHandle="<%= sConnectionHandle %>" id="<%= sId %>"
   name="<%= sName %>" type="<%= sType %>" version="<%= sVersionId %>" />
```

The TLD file for a custom tag library specifies required and optional attributes to use with its tags. For a list of tags and their attributes for the Actuate custom tag libraries, see *Information Console Developer Guide.*

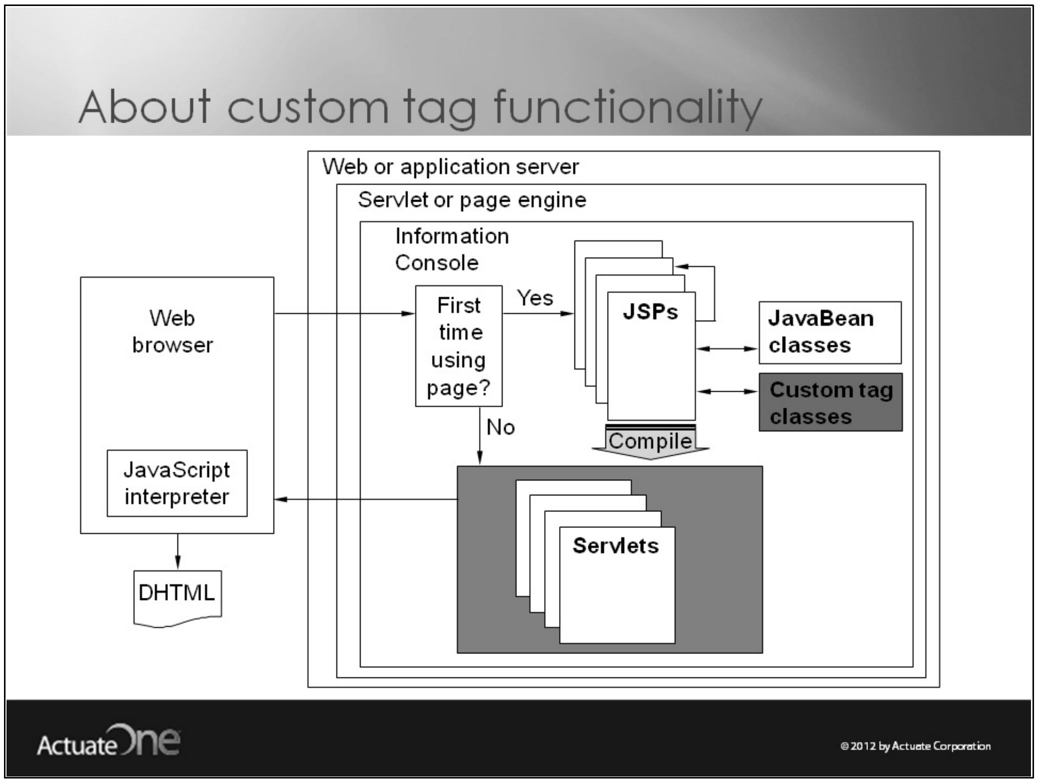
The body of a custom tag can include a JSP or other code. Unlike an attribute, the body of a custom tag can contain multiline values and reserved characters. To create a custom tag with a body, use the syntax shown in the following example:

```
<library-prefix:tag-name [attribute-name="attributevalue"]/>
   body-text
</library-prefix:tag-name>
```

For example, the following code shows part of the searchReport custom tag defined in the viewer.tld file. This part of the tag supports inclusion of a body containing JSP code for execution within the tag.

```
<tag>
      <name>searchReport</name>
      <tagclass>com.actuate.reportcast.tags.viewer.SearchReportTag
         </tagclass>
      <teiclass>com.actuate.reportcast.tags.viewer.SearchReportTEI
         </teiclass>
      <bodycontent>JSP</bodycontent>
      <info>
         This tag is used for fetching data corresponding to the specified
            search conditions in a report
      </info>
      ...
```

Slide 67 About custom tag functionality

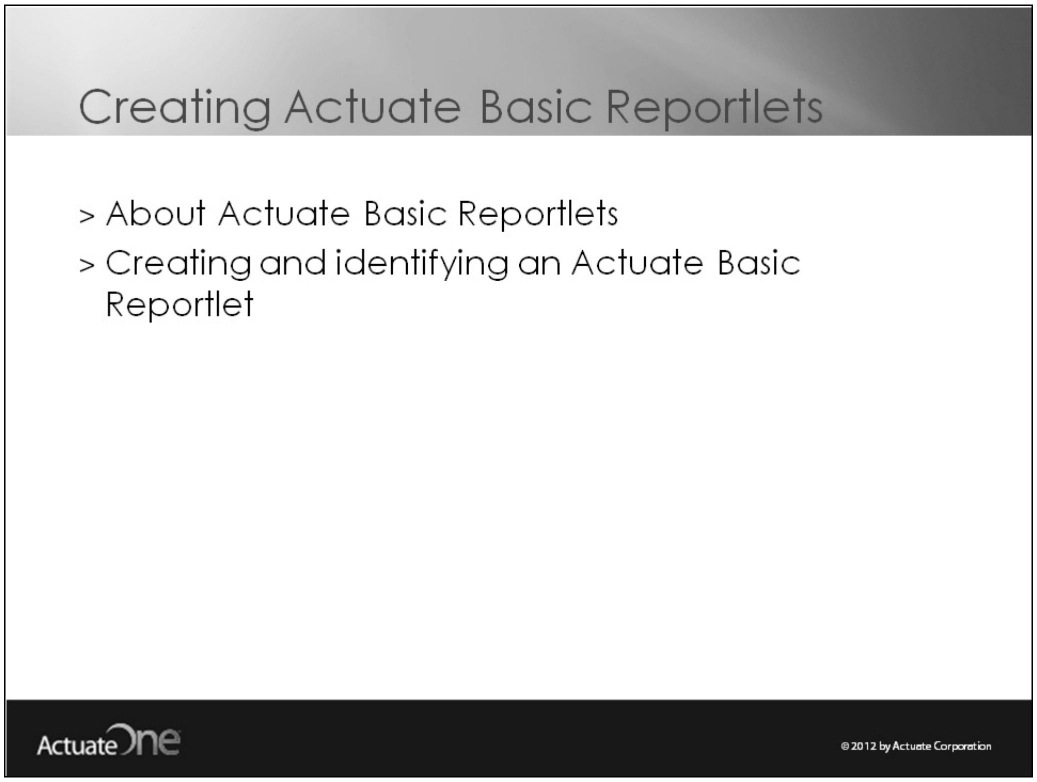


When a user performs an action in the web browser that requires a new Information Console page, the following actions result:

- The web browser requests a JSP or an action from the web or application server, such as the Actuate 11 Apache Tomcat for Information Console service.
- If a user requests an action, the Jakarta Struts framework controller handles the action. If the action uses a JavaBean, the controller calls the bean and evaluates any returned result. Then, the controller uses the logic in the action to forward the request to the appropriate JSP or action.
- If the JSP has not been used in this session or has not been used since its last modification, the application server locates the JSP. The server's servlet or page engine compiles the JSP, using any necessary JavaBean classes, Jakarta Struts custom tag libraries, and Actuate custom tag libraries:
  - The JSP run-time engine uses the URI parameter of each taglib directive to identify the tag libraries associated with the page.
  - The JSP run-time engine loads the libraries' TLDs, using web.xml to obtain the location of any TLDs specified in a taglib directive.
  - The JSP run-time engine parses each TLD and uses the information to decide which helper classes to use for populating instances of the helper classes and which methods to call on the handler.
  - The JSP run-time engine reads the rest of the JSP. When it reaches a custom tag call, the run-time engine uses the helper classes to verify whether the tag exists in the TLD, validate the tag's syntax, and create a Java stub for the tag to access the tag handler class that implements this tag. The JSP run-time engine now can use this tag handler class in the servlet for the page.

- The servlet runs. The web or application server returns the output to the web browser.
- The web browser interprets any JavaScript in the page and displays the DHTML result.

Slide 68 Creating Actuate Basic Reportlets



Creating Actuate Basic Reportlets

> About Actuate Basic Reportlets
> Creating and identifying an Actuate Basic Reportlet

ActuateOne
© 2012 by Actuate Corporation

About Actuate Basic Reportlets

An Actuate Basic Reportlet is a portion of a report that can be embedded in a web page. Using Actuate Basic Reportlets within a web page supports displaying a portion of a report, suppressing page breaks, and providing page-level security.

The appearance and behavior of a report changes when the content is extracted as an Actuate Basic Reportlet. For example, Actuate Basic Reportlets use the full width of the frame. In addition, the Viewer's navigation toolbar does not appear. The absence of the toolbar hides the table of contents, search, and PDF or Excel generation features.

An Actuate Basic Reportlet does not contain JavaScript. Also, several HTML formatting tags that would appear in a DHTML report do not appear in an Actuate Basic Reportlet. To support dynamically streaming content within an existing JSP, the Actuate Basic Reportlet HTML does not contain <HTML>, </HTML>, <HEAD>, </HEAD>, <BODY>, and </BODY> tags. In an Actuate Basic Reportlet, <DIV> and </DIV> tags designate divisions between specific portions of Reportlet content, and support adding the Reportlet to a web page.

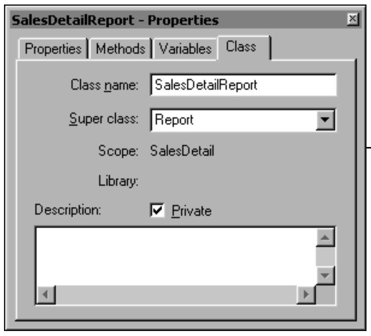
Creating and identifying an Actuate Basic Reportlet

To design an Actuate Basic Reportlet, begin by examining the design of the source report in e.Report Designer Professional. Optionally, extract a report section, group section, sequential section, page, or frame as an Actuate Basic Reportlet.

Specifying the component

If the component's class has global scope, specify the component by its name, such as ReportSummary. If the component's class does not have global scope, specify the component by the scope of the class, two colons, and the component's name. For example, if a class, NegativeChanges, is defined within the BudgetApp scope, specify the component as BudgetApp::NegativeChanges.

To determine the scope of the component, select the component in e.Report Designer Professional. In the Properties window, choose the Class page. The scope appears on the page, as shown in Figure 9-1.


SalesDetailReport - Properties
Properties
Methods
Variables
Class
Description:
The scope of SalesDetailReport, a report section, is the root SalesDetail report

Class name: SalesDetailReport
Super class: Report
Scope: SalesDetail
Library:
Private: ✓

Figure 9-1 The scope of a report section

Storing a report executable file and report instance file in a volume

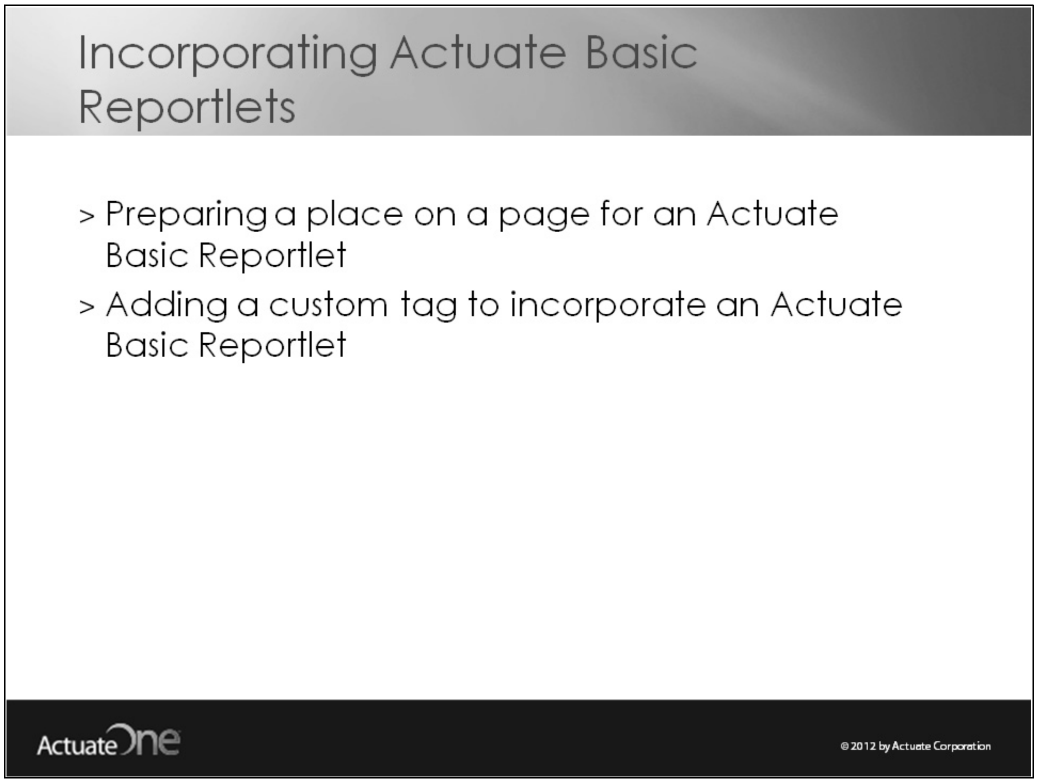
Ensure that the report object executable (.rox) file for the source report for the Actuate Basic Reportlet is stored in the Encyclopedia volume that contains the report object instance (.roi) file. If the instance file is uploaded without the executable file, the dependency for the ROX to the ROI must be set. Generating the ROI using Information Console or Management Console sets this dependency.

Implementing the Actuate Basic Reportlet using a URL

To implement an Actuate Basic Reportlet using a URL, create a search URL to extract a component as an Actuate Basic Reportlet. For example, the following search URL extracts page 1 of the Detail.roi report document located in the Reportlets folder on the Actuate iServer named siamese:

```
http://siamese:8700/iportal/servlet/ViewPage?page=1&format=Reportlet
   &name=/Reportlets/Detail.roi&version=001&type=roi
```

Slide 69 Incorporating Actuate Basic Reportlets


Incorporating Actuate Basic Reportlets

> Preparing a place on a page for an Actuate Basic Reportlet

> Adding a custom tag to incorporate an Actuate Basic Reportlet

ActuateOne
© 2012 by Actuate Corporation

Preparing a place on a page for an Actuate Basic Reportlet

To contain an Actuate Basic Reportlet within a JSP, create an IFRAME or an HTML table. Use an IFRAME to display a URL that accesses the servlet directly, as shown in the following example:

```
<IFRAME
   SRC="http://siamese:8700/iportal/servlet
      /ViewPage?page=1&format=Reportlet
      &name=/Reportlets/detail.roi&version=001&type=roi&scalingFactor=60"
      NAME="A" HEIGHT=250 WIDTH=80% HSPACE=10 SCROLLING=YES ALIGN=RIGHT>
      Your browser cannot display iFrames
</IFRAME>
```

To include the Reportlet within the same frame as the rest of the page, use an Actuate Basic Reportlet custom tag. In a typical page, place the Actuate Basic Reportlet within an HTML table.

For example:

```
<TABLE WIDTH=800 BORDER=0 CELLSPACING=0 CELLPADDING=10>
<TR>
<TD VALIGN="top">
<%-- The Reportlet tag goes here. --%>
</TD>
</TR>
</TABLE>
```

Adding a custom tag to incorporate an Actuate Basic Reportlet

To invoke an Actuate Basic Reportlet, use the ViewPage servlet in a URL, or use the JSP custom tag getReportletData. When using a custom tag, include the Actuate Basic Reportlet within the same frame as the rest of the page.

The Reportlet TLD contains two Reportlet custom tags, getReportletData and getReportlet. Both tags send a request for an Actuate Basic Reportlet to Actuate iServer, parse the response, and write the Reportlet to the JSP output stream.

The getReportlet tag is nearly identical to the getReportletData tag. In both tags, identify the report component from which to retrieve component data, either by the componentID or by the componentName, in conjunction with componentValue. The getReportlet tag gets the component as a page. getReportletData gets the entire component. To retrieve only part of the component as a Reportlet, use getReportlet with page numbers or search criteria.

The syntax for using either tag is shown in the following example:

```
<reportlet:<tag-name>
   authID="<%= sAuthID %>"
   volume="<%= sVolume %>"
   serverURL="<%= sServerURL %>"
   locale="<%= sLocale %>"
   timeZone="<%= tzTimeZone %>"
   id="reportlet"
   name="<report relative pathname>"
   componentID="<report design component ID>"
   reportletMaxheight="<reportlet height>"
   embeddedObjPath="../servlet/ViewEmbeddedObject?operation=" >
</reportlet:<tag-name>>
```

The authID is the authentication ID returned by Actuate iServer upon successful login. Specify the report component to retrieve in one of the following ways:

- By component ID, using the following syntax:

  ```
  componentID="<report design component ID>"
  ```

  The component ID uniquely identifies the component and, if the component occurs in the report multiple times, the instance of the component to use.

- By component name, using the following syntax:

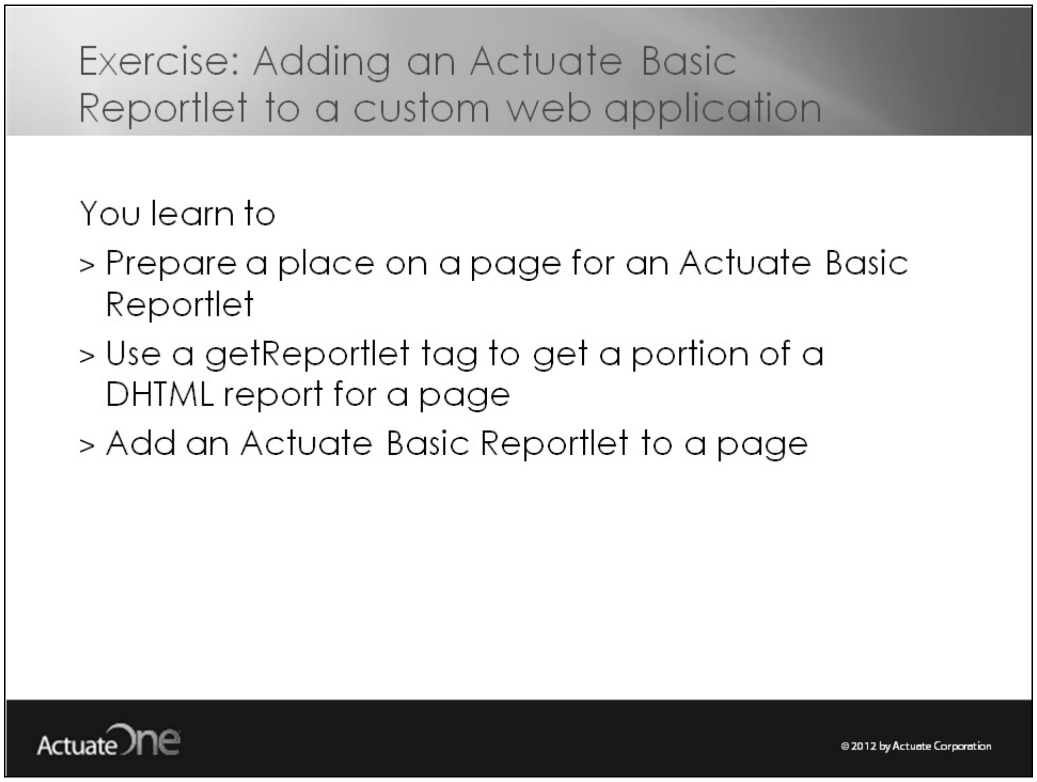
  ```
  componentName="<report design component name>"
  componentValue="<expression>"
  ```

  Use the name assigned to the report component during report design as the componentName. If no componentValue is specified, Information Console retrieves the first occurrence of the component. To access a different instance of the component, specify the occurrence of the component in componentValue. Use an expression such as the current user name or a search expression to set the value.

The following example shows a Reportlet tag that retrieves a customer's financial portfolio from the Portfolio report's Portfolio::FrameClientPortfolioHistory component:

```
<reportlet:getReportletData  authID="<%= sAuthID %>"
volume="<%= sVolume %>"
serverURL="<%= sServerURL %>" locale="<%= sLocale %>"
timeZone="<%= tzTimeZone%>"
id="reportlet" name="/Portfolio/portfolio.roi"
componentName="Portfolio::FrameClientPortfolioHistory"
componentValue="<%= sUserName%>" reportletMaxheight="300"  version="1"
embeddedObjPath="../servlet/ViewEmbeddedObject?operation=" >
```

Slide 70 Exercise 7



Exercise: Adding an Actuate Basic Reportlet to a custom web application

You learn to

> Prepare a place on a page for an Actuate Basic Reportlet

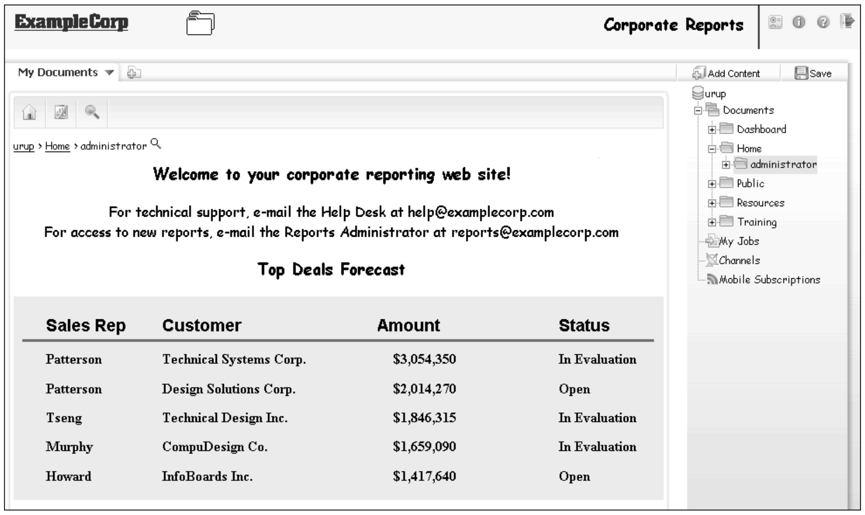
> Use a getReportlet tag to get a portion of a DHTML report for a page

> Add an Actuate Basic Reportlet to a page

ActuateOne
© 2012 by Actuate Corporation

Exercise 7 Adding an Actuate Basic Reportlet to a custom web application

Overview In this exercise, you add an Actuate Basic Reportlet to the Welcome page. After logging in, users see the Welcome page with a listing of the top deals currently forecast and the status of those deals. Welcome looks like the illustration in Figure 9-2.



Welcome to your corporate reporting web site!

For technical support, e-mail the Help Desk at help@examplecorp.com
For access to new reports, e-mail the Reports Administrator at reports@examplecorp.com

Top Deals Forecast

Sales Rep	Customer	Amount	Status
Patterson	Technical Systems Corp.	$3,054,350	In Evaluation
Patterson	Design Solutions Corp.	$2,014,270	Open
Tseng	Technical Design Inc.	$1,846,315	In Evaluation
Murphy	CompuDesign Co.	$1,659,090	In Evaluation
Howard	InfoBoards Inc.	$1,417,640	Open

Figure 9-2 An Actuate Basic Reportlet appearing in the Welcome page of the ExampleCorp web application

What you learn In this exercise, you learn to:

- Prepare a place on a page for an Actuate Basic Reportlet.
- Use a getReportlet tag to get a portion of a DHTML report for a page.
- Add an Actuate Basic Reportlet to a page.

What you do In this exercise, you complete the following tasks:

- Upload a report executable file to a volume
- Generate a report object instance (.roi) file
- Prepare web.xml to use a Reportlet library
- Prepare welcomecontent.jsp to use an Actuate Basic Reportlet
- Test an Actuate Basic Reportlet

Before you begin To complete this exercise, you need to have completed Exercise 1, “Creating an Information Console web application,” Exercise 2, “Customizing styles,” and Exercise 6, “Customizing navigation.” Complete Exercise 1, “Creating an Information Console web application,” Exercise 2, “Customizing styles,” and Exercise 6, “Customizing navigation,” before you begin this exercise.

If you have not completed Exercise 6, “Customizing navigation,” copy the files listed in Table 9-1 to the appropriate destination folder.

Table 9-1 Files to copy for Exercise 7

File name	Source folder	Destination folder
struts-config.xml	C:\ActuateTraining11 \IC-Day2\Finish \CustomizingNavigation\	<context root>\WEB-INF\

Table 9-1 Files to copy for Exercise 7

File name	Source folder	Destination folder
welcome.jsp	C:\ActuateTraining11 \IC-Day2\Finish \CustomizingNavigation\	<context root>\iportal \activePortal\private\filesfolders

To complete this exercise, you need access to the following resources:

- \ActuateTraining11\IC-Day2\welcomecontent.jsp
- \ActuateTraining11\IC-Day2\Forecast\Forecast.rox
- <context root>\iportal\activePortal\private\filefolders\welcomecontent.jsp
- <context root>\WEB-INF\web.xml

Task 1: Upload a report executable file to a volume

In this task you add a report executable file to the Training directory and change access rights to that file, using Actuate Management Console.

1 Choose Start➛All Programs➛Actuate 11➛BIRT iServer Management Console. The Management Console login appears in the web browser, as shown in Figure 9-3.

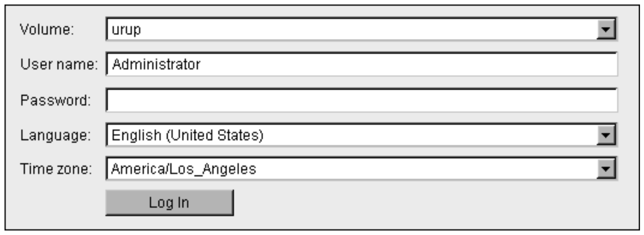

Volume: urup
User name: Administrator
Password:
Language: English (United States)
Time zone: America/Los_Angeles

Figure 9-3 Logging in to Management Console as Administrator

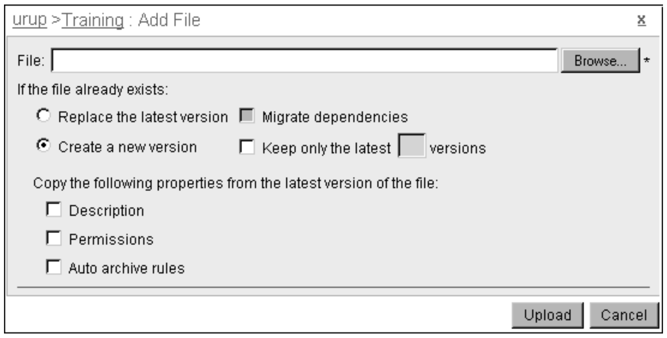
2 Log in to Management Console as Administrator without a password. The Administrator's home folder appears.

3 In the breadcrumb, choose the link for the top-level directory in the Encyclopedia volume. Then, choose Training.

4 To add Forecast.rox to the Training folder and assign all privileges, do the following steps:

1 Choose Add file.

2 In Add File choose Browse, as shown in Figure 9-4.


urup >Training : Add File

File: Browse... *

If the file already exists:

- Replace the latest version
- ☐ Migrate dependencies
- ● Create a new version
- ☐ Keep only the latest versions

Copy the following properties from the latest version of the file:

- ☐ Description
- ☐ Permissions
- ☐ Auto archive rules

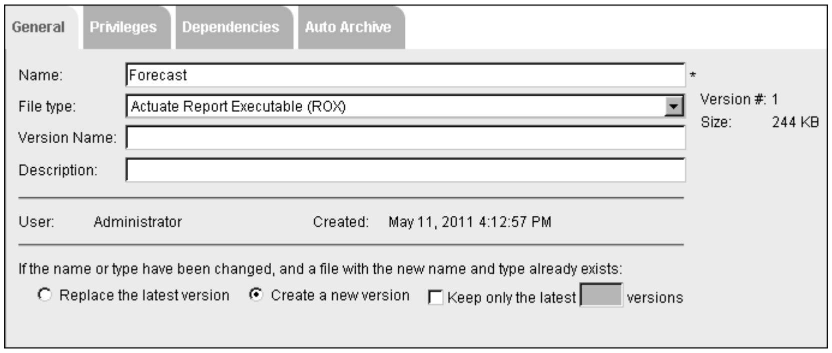
Figure 9-4 Browsing for a file to add

3 On Choose file, navigate to the following report executable file:

```
C:\ActuateTraining11\IC-Day2\Forecast\Forecast.rox
```

Choose Open.

4 On Add File, choose Upload. Properties—General appears, as shown in Figure 9-5.



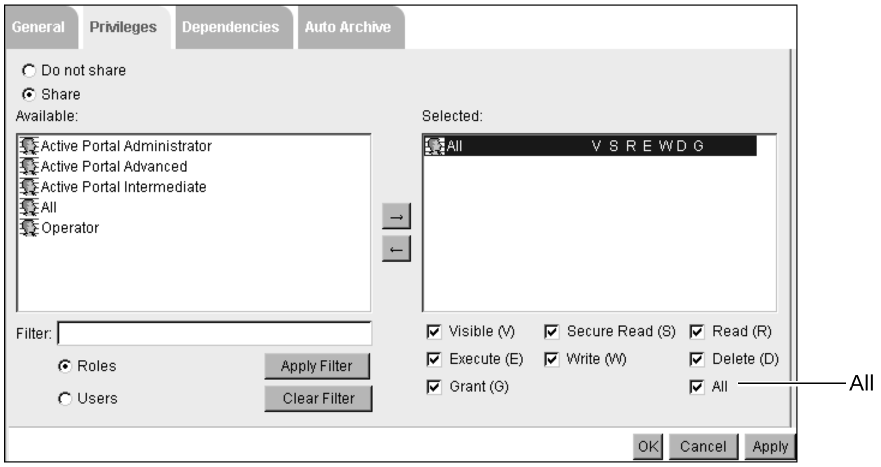
Name: Forecast *
File type: Actuate Report Executable (ROX)
Version Name:
Description:
Version #: 1
Size: 244 KB
User: Administrator Created: May 11, 2011 4:12:57 PM
If the name or type have been changed, and a file with the new name and type already exists:
( ) Replace the latest version (•) Create a new version [ ] Keep only the latest ___ versions

Figure 9-5 Properties—General

5 Choose Privileges.

6 On Properties—Privileges, in Available, select All. Choose the right arrow. All appears in Selected.

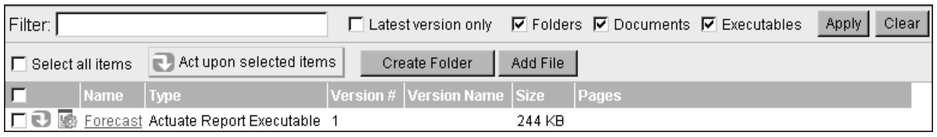
7 To assign all privileges to the All role, select All. Privileges for the All role appear in Selected, as shown in Figure 9-6.



( ) Do not share
(•) Share
Available:
Active Portal Administrator
Active Portal Advanced
Active Portal Intermediate
All
Operator
Selected:
All V S R E WD G
Filter:
(•) Roles
( ) Users
[✓] Visible (V) [✓] Secure Read (S) [✓] Read (R)
[✓] Execute (E) [✓] Write (W) [✓] Delete (D)
[✓] Grant (G) [✓] All

Figure 9-6 Selecting all privileges for the All role

8 Choose OK. Forecast appears as a report executable in Training, as shown in Figure 9-7.



Filter: [ ] Latest version only [✓] Folders [✓] Documents [✓] Executables
[ ] Select all items

	Name	Type	Version #	Version Name	Size	Pages
[ ]	Forecast	Actuate Report Executable	1		244 KB	

Figure 9-7 The Forecast report executable appearing in the Training folder

Task 2: Generate a report object instance (.roi) file

In this task, you run the Forecast report executable file using Actuate Management Console. This generates the Forecast report instance file, and creates a dependency between the two files.

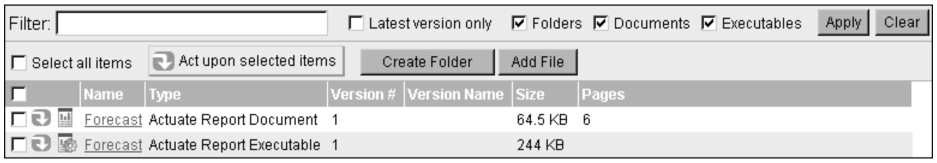
1 In the Training folder, choose Forecast.

2 In Forecast (ROX) (Version 1): Schedule, choose Output.

3 To save the report instance file in the /Training folder, in Folder, select Other.

4 Choose OK.

5 In the browser, choose Refresh. Forecast appears in Training, as shown in Figure 9-8.



Name	Type	Version #	Version Name	Size	Pages
Forecast	Actuate Report Document	1		64.5 KB	6
Forecast	Actuate Report Executable	1		244 KB	

Figure 9-8 Forecast.roi appearing in Training

6 Choose Log out.

7 Close the browser window.

Task 3: Prepare web.xml to use a Reportlet library

In this task, you modify the web application to access a Reportlet library.

1 Using JCreator or another code editor, open the following file:

```
<context root>\WEB-INF\web.xml
```

2 To enable your web application to access the Reportlet library, before the following lines:

```
</jsp-config>
<!-- configuration -->
```

add the following code:

```
<taglib>
   <taglib-uri>/reportlet</taglib-uri>
   <taglib-location>/WEB-INF/reportlet.tld</taglib-location>
</taglib>
```

3 Save and close web.xml.

Task 4: Prepare welcomecontent.jsp to use an Actuate Basic Reportlet

In this task, you modify the JSP that controls the Welcome page. These modifications set the page to display a message using a Reportlet.

1 Create a backup copy of <context root>\iportal\activePortal\private\filesfolders \welcomecontent.jsp called welcomecontent_backup.jsp.

2 Copy \ActuateTraining11\IC-Day2\welcomecontent.jsp into the following folder:

```
<context root>\iportal\activePortal\private\filesfolders\
```

3 Using JCreator or another code editor, open the following file:

```
<context root>\iportal\activePortal\private\filesfolders
   \welcomecontent.jsp
```

4 In welcomcontent.jsp, examine to the following line:

```
<%@ taglib uri="/reportlet" prefix="reportlet" %>
```

This line provides welcomecontent.jsp access to the Reportlet tag library.

5 Navigate to and examine the following code:

```
<jsp:useBean id="userinfobean"
   class="com.actuate.activeportal.beans.UserInfoBean" scope="session"/>
<%
   String sAuthID = userinfobean.getAuthid();
   String sServerURL = userinfobean.getServerurl();
   String sVolume = userinfobean.getVolume();
   com.actuate.common.util.AcLocale sLocale  =
      userinfobean.getAcLocale();
   com.actuate.reportcast.utils.AcTimeZone tzTimeZone =
      userinfobean.getTimezone();
%>
```

This code retrieves the necessary information to load an Actuate Basic Reportlet using the Reportlet tag library—authID, serverURL, volume, locale, and time zone.

6 Navigate to and examine the following code:

```
<h3 align="center">Top Deals Forecast</h3>
<TABLE WIDTH=650 BORDER=0 CELLSPACING=0 CELLPADDING=10>
   <TR>
      <TD VALIGN="top" bgcolor="#efebde">
         <reportlet:getReportletData
            authID="<%= sAuthID %>"
            serverURL="<%= sServerURL %>" volume="<%= sVolume %>"
            locale="<%= sLocale %>" timeZone="<%= tzTimeZone %>"
            id="reportlet" name="InsertFileName"
            componentName="InsertComponentName"
            reportletMaxheight="160" >
         </reportlet:getReportletData>
      </TD>
   </TR>
</TABLE>
```

This code contains two placeholders, InsertFileName and InsertComponentName.

7 To specify the file that contains the Reportlet component, in welcomecontent.jsp, replace the following text:

```
name="InsertFileName"
```

with:

```
name="/Training/Forecast.roi"
```

8 To specify the Reportlet component name, replace the following text:

```
componentName="InsertComponentName"
```

with:

```
componentName="ForecastApp::TopDeals"
```

The code looks like the following example:

```
<h3 align="center">Top Deals Forecast</h3>
<TABLE WIDTH=650 BORDER=0 CELLSPACING=0 CELLPADDING=10>
   <TR>
      <TD VALIGN="top" bgcolor="#efebde">
         <reportlet:getReportletData
            authID="<%= sAuthID %>"
            serverURL="<%= sServerURL %>" volume="<%= sVolume %>"
            locale="<%= sLocale %>" timeZone="<%= tzTimeZone %>"
            id="reportlet" name="/Training/Forecast.roi"
            componentName="ForecastApp::TopDeals"
            reportletMaxheight="160" >
         </reportlet:getReportletData>
      </TD>
   </TR>
</TABLE>
```

9 Save welcomecontent.jsp. Close the code editor.

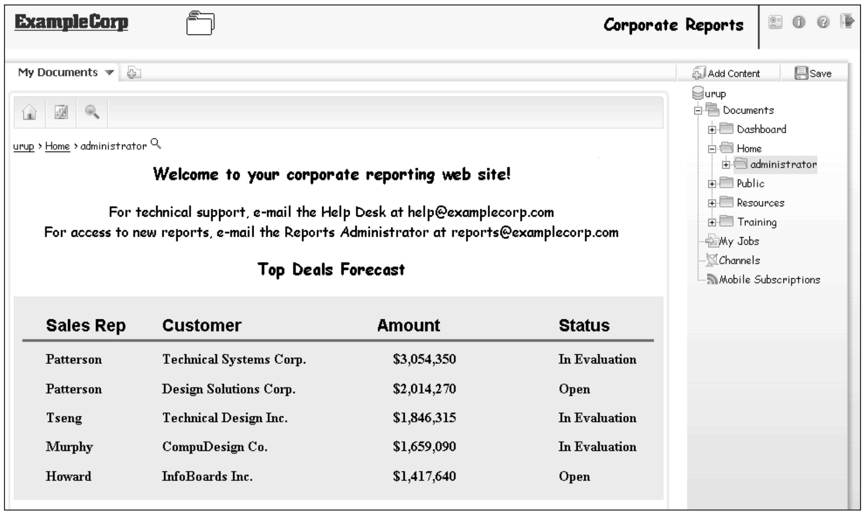
Task 5: Test an Actuate Basic Reportlet

In this task, you log in to the ExampleCorp web application to test the Reportlet.

1 To have Actuate 11 Apache Tomcat for Information Console service use the modified web.xml file, restart the service.

2 Choose the ExampleCorp Web application shortcut.

3 Log in to the ExampleCorp web application as Administrator. The Welcome page appears, as shown in Figure 9-9.



Welcome to your corporate reporting web site!

For technical support, e-mail the Help Desk at help@examplecorp.com
For access to new reports, e-mail the Reports Administrator at reports@examplecorp.com

Top Deals Forecast

Sales Rep	Customer	Amount	Status
Patterson	Technical Systems Corp.	$3,054,350	In Evaluation
Patterson	Design Solutions Corp.	$2,014,270	Open
Tseng	Technical Design Inc.	$1,846,315	In Evaluation
Murphy	CompuDesign Co.	$1,659,090	In Evaluation
Howard	InfoBoards Inc.	$1,417,640	Open

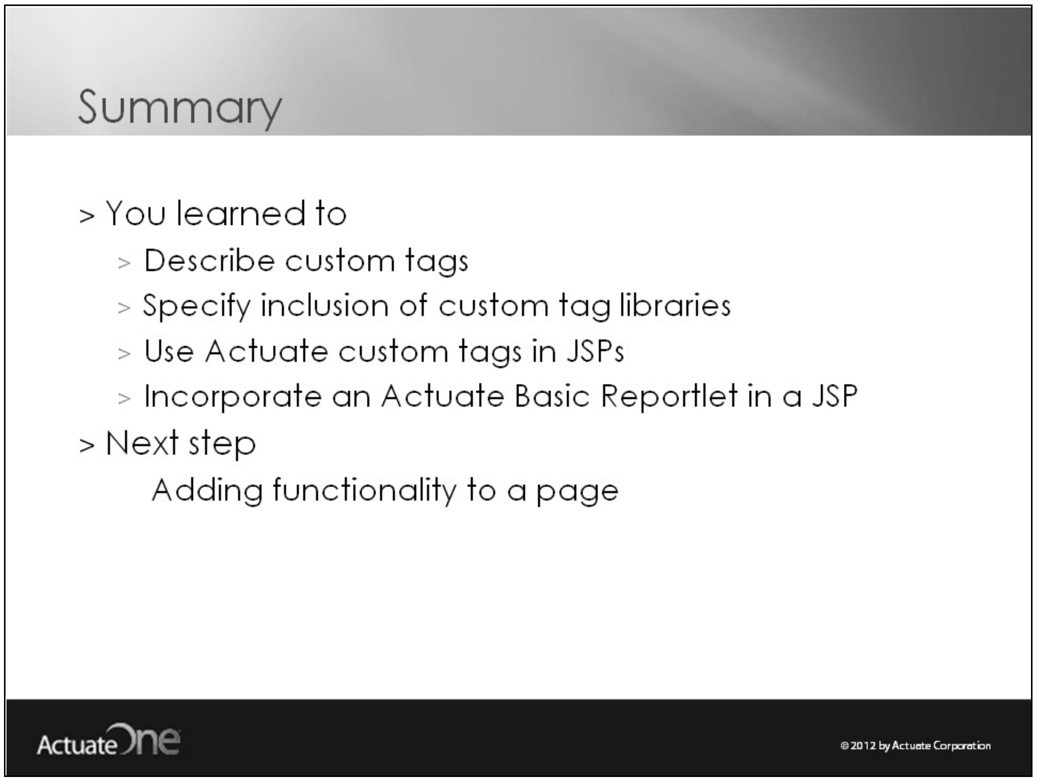
Figure 9-9 Examining the message displayed on the Welcome page for the ExampleCorp web application, using an Actuate Basic Reportlet

4 Close the browser window.

Summary In this exercise, you learned to:

- Prepare a place on a page for an Actuate Basic Reportlet.
- Use a getReportlet tag to get a portion of a DHTML report for a page.
- Add an Actuate Basic Reportlet to a page.

Slide 71 Summary



Summary

> You learned to
  > Describe custom tags
  > Specify inclusion of custom tag libraries
  > Use Actuate custom tags in JSPs
  > Incorporate an Actuate Basic Reportlet in a JSP
> Next step
    Adding functionality to a page

ActuateOne © 2012 by Actuate Corporation

Quiz

1 How do custom tags support separating presentation from content in JSPs?

2 In addition to the default JSP tag libraries, Information Console uses two sets of custom tag libraries. Who created each of these libraries?

3 What custom tag library is not used in the default Information Console web application but is frequently used in custom web applications? What is its purpose?

4 Can you use two tags with the same name within a JSP? If so, how do you specify which library to use for each tag?

5 How does the servlet or page engine use web.xml to process custom tags?

6 What is the difference between the getReportlet and getReportletData tags?