**Adding BI Publisher Content to an ADF Project**

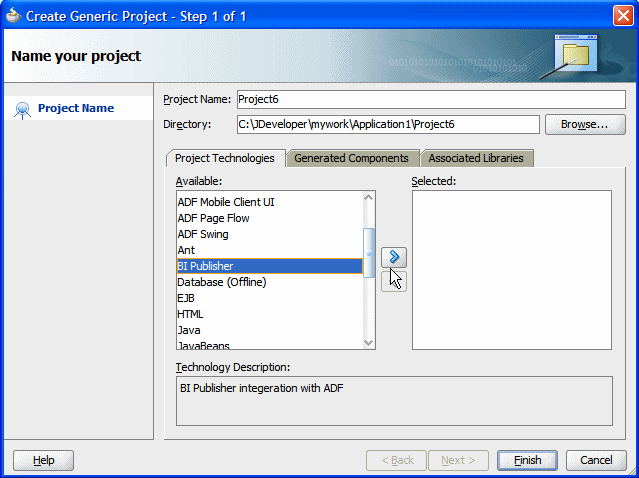
After you install the BI Publisher extension you can create a project that includes a BI Publisher report region. The following procedures describe how to add BI Publisher content to an ADF project:

**1 Adding the BI Publisher Technology Scope**

To display the BI Publisher options, add the BI Publisher technology scope to your project.

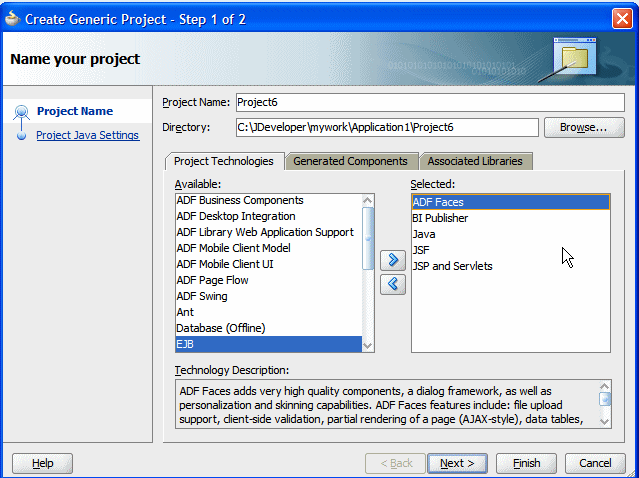
1. In Oracle JDeveloper, go to the Projects Pane and right-click the project to which you want to add the technology scopes and select Project Properties.
2. Select Technology Scope.
3. In the Available Technologies list, select BI Publisher as shown in [Figure 10-2](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACBHIBH):

***Figure 10-2 Selecting the BI Publisher Technology Scope***



1. Click the right shuttle button to add BI Publisher and its dependent technologies to your project. Dependent technologies are moved together. [Figure 10-3](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACFHBFE) shows the results of this action.

***Figure 10-3 BI Publisher and Dependent Technologies***



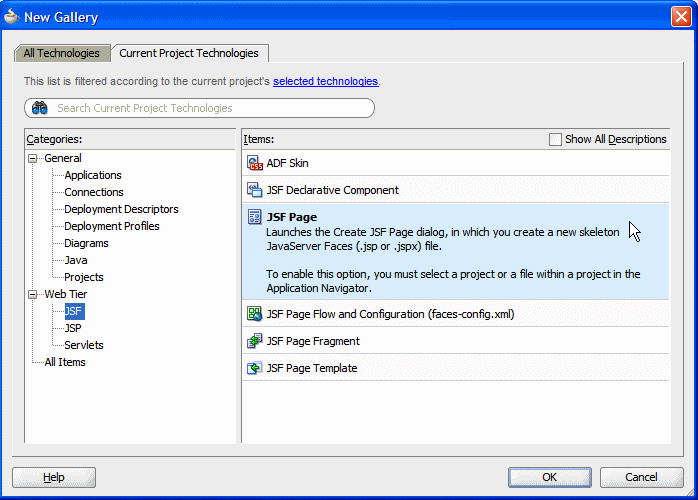
1. Click **Finish**.

**2 Adding a BI Publisher Region to the JSF Page**

The BI Publisher region must reside on a JSF page. To add a JSF page to your project:

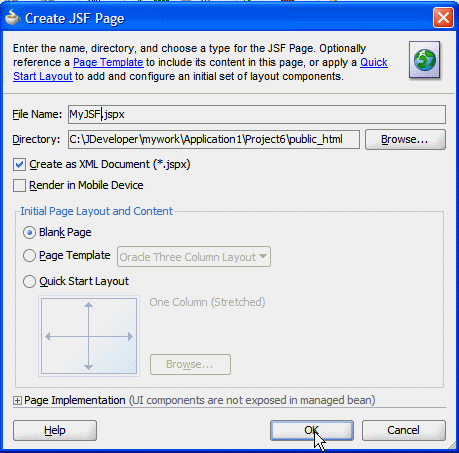
1. Right-click your project and select New.
2. From the Gallery dialog, on the Categories pane, under Web Tier, select JSF.
3. From the items presented in the right pane, select JSF Page, as shown in [Figure 10-4](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACCIAJH).

***Figure 10-4 Selecting a JSF Page***



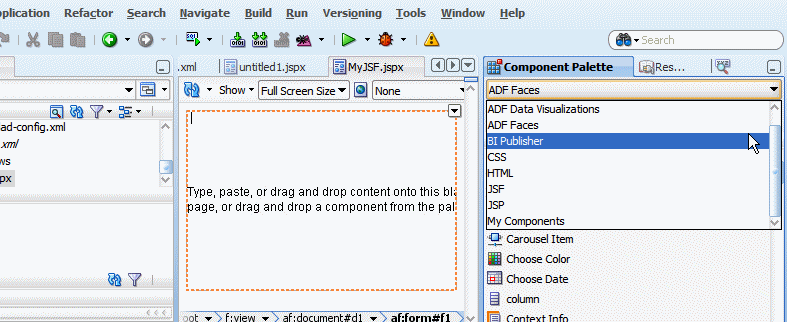
1. On the Create JSF Page dialog, enter a name for your JSF page. Oracle recommends selecting Create as XML Document to create an XML-based JSP document (extension .jspx).

***Figure 10-5 Create JSF Page Dialog***



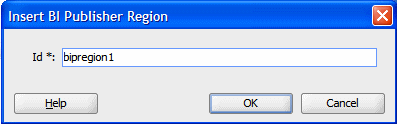
1. From the Component Palette list, select BI Publisher, as shown in [Figure 10-6](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACDDIBD).

***Figure 10-6 Component Palette List***



1. Drag and drop the BI Publisher Region component from the palette to the page.
2. Once the BI Publisher Region is dropped to the JSPX page, the **Insert BI Publisher Region** dialog, shown in [Figure 10-7](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACBCAAC), prompts you to enter a region ID for the BI Publisher Region. The region ID must be unique and must not include the underscore "\_" character.

***Figure 10-7 BI Publisher Region Dialog***

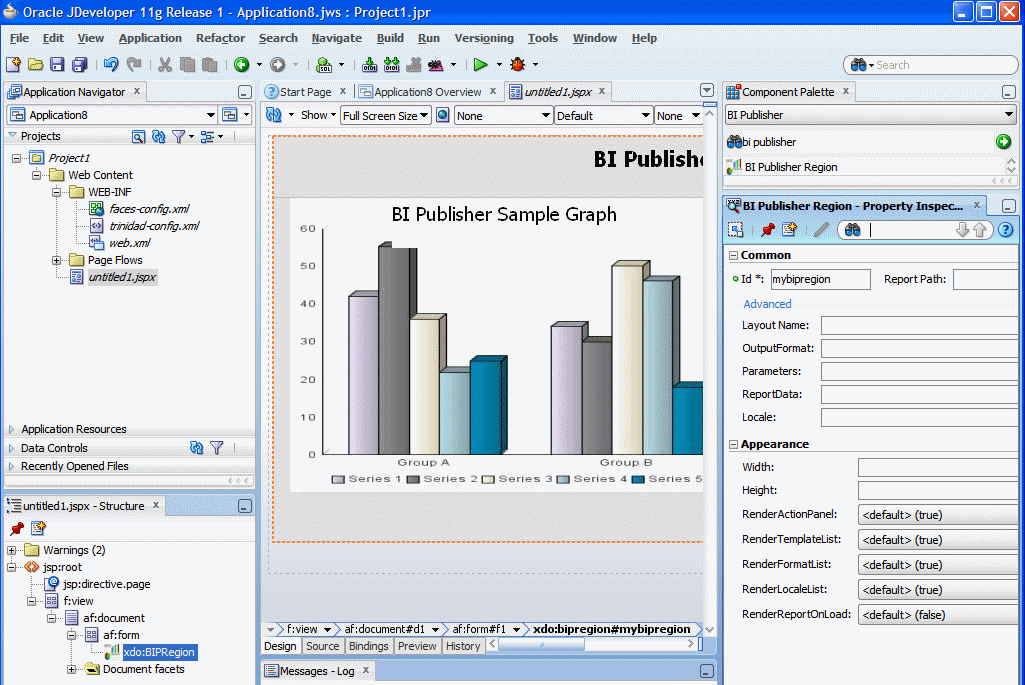


Click OK. When you insert the BI Publisher region notice the following:

* + A placeholder image displays in the BI Publisher region of your page.
  + The BI Publisher configuration file, /WEB-INF/xmlp-client-config.xml is created. Click **Refresh** on the Projects pane if it does not display immediately.
  + The Property Inspector presents the BI Publisher region properties when the BI Publisher region is selected.

[Figure 10-8](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACDDIIF) shows these items:

***Figure 10-8 Report Image, Configuration File, and Property Inspector***



1. In the Property Inspector, under the **Appearance** region, enter the report region Width and Height in px to specify the size of the report region in your application page. For example 1000 px and 800 px. No defaults are assigned for height and width, so ensure that you set these fields.

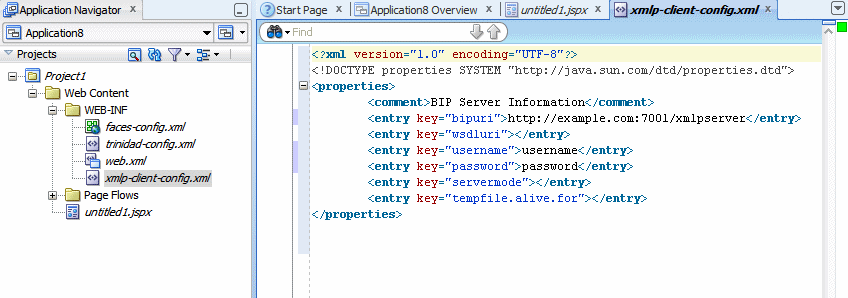
**3 Configuring Connection to the BI Publisher Server**

To establish a connection to the BI Publisher server, update the xmlp-client-config.xml file under Project > Web Content > WEB-INF with the connection information as follows:

1. Double-click **xmlp-client-config.xml** to open the file for editing.

[Figure 10-9](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACDDBEG) shows the sample **xmlp-client-config.xml** file.

***Figure 10-9 Entering the Connection Keys***



1. Update the properties (keys) as shown in the following example. Add the properties not found in the default file.

The sample **xmlp-client-config.xml** file:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">

<properties>

<comment>BIP Server Information</comment>

<entry key="bipuri">http://example.com:7001/xmlpserver</entry>

<entry key="sso.bipuri">http://example.com:7001/xmlpserver</entry>

<entry key="username">username</entry>

<entry key="policy">oracle/wss\_username\_token\_client\_policy</entry>

</properties>

* + bipuri - enter the URL for the BI Publisher server. This may be behind a firewall. For example: http://hostname.com:7001/xmlpserver
  + sso.bipuri - enter the external SSO-enabled BI Publisher URL, which is accessible by a client browser. In Oracle Fusion Applications Business Intelligence domains this property is already configured. For other implementations, you must enter this property.
  + username - enter the secure user name to connect to BI Publisher. This user is the authenticated user used to connect from your application to BI Publisher.
  + policy- enter the OWSM web service security policy to use when connecting to the BI Publisher web service. See [Section 10.4, "Configuring a Security Policy"](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHBFJDH) for more information.

**4 Setting Properties for the BI Publisher Region**

The following properties are available in the Property Inspector:

***Table 10-1 Common Properties***

| **Property** | **Description** |
| --- | --- |
| Id | The unique ID for the BI Publisher region. You assign this ID when you insert the BI Publisher region to your page. The ID must not contain spaces. |
| Report Path | The path to the report in the BI Publisher catalog. Reports have the extension ".xdo". Begin the path from the first level beneath "Shared Folders," but do not include "Shared Folders." For example: "/Samples/1.+Overview/Balance+Letter+Report.xdo" |
| Rendered | The control to show or hide the BI Publisher region when the page is rendered.  The default is true (show). |

***Table 10-2 Advanced Properties***

| **Property** | **Description** |
| --- | --- |
| Layout Name | The layout from the report definition to apply to the report data. For example, "My Layout". If you do not specify a value here the default layout from the report definition is used. |
| OutputFormat | Sets the default output format. See [Table 10-4](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHBIJFJ) for the list of valid values. If you do not specify a value here the default output format from the report definition is used. |
| Parameters | To pass parameters from your application page to the BI Publisher report, use this property to specify the backing bean that captures the parameter values. Use the Expression Builder to populate this field. For example:  #{UIBackingBean.parameters}  See [Section 10.6, "Passing Parameters from the Application Page"](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHJDDHD) for information. |
| ReportData | If your report uses a push data model, use this property to specify the backing bean that contains the method that defines the report data location. Use the Expression Builder to populate this field. For example:  #{UIBackingBean.reportData}  See [Section 10.7, "Using a Push Data Model"](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHICJFI) for more information. |
| Locale | Enter a default locale format using the ISO language code-country code combination, for example en-US. |

***Table 10-3 Appearance Properties***

| **Property** | **Description** |
| --- | --- |
| Width | Sets the report region width. Enter the value in pixels, for example, 1000 px. |
| Height | Sets the report region height. Enter the value in pixels, for example, 800 px. |
| RenderActionPanel | Controls whether to show or hide the report viewer action panel. The default is true.  Note that if you set this to false, you must set renderReportOnLoad to true. |
| RenderFormatList | Controls whether to show or hide the output Format List. The default is true. |
| RenderLocaleList | Controls whether to show or hide the Locale List. The default is true. |
| RenderReportOnLoad | When set to true, the report is generated when the ADF page is launched. The default is true. |

***Table 10-4 Valid Values for OutputFormat***

| **Output Format** | **Value to Enter for OutputFormat Property** | **Template Types That Can Generate This Output Format** |
| --- | --- | --- |
| Interactive | N/A | Not supported |
| HTML | html | BI Publisher, RTF, XSL Stylesheet (FO) |
| PDF | pdf | BI Publisher, RTF, PDF, Flash, XSL Stylesheet (FO) |
| RTF | rtf | BI Publisher, RTF, XSL Stylesheet (FO) |
| Excel (mhtml) | excel | BI Publisher, RTF, Excel, XSL Stylesheet (FO) |
| Excel (html) | excel2000 | BI Publisher, RTF, Excel, XSL Stylesheet (FO) |
| Excel (\*.xlsx) | xlsx | BI Publisher, RTF, XSL Stylesheet (FO) |
| PowerPoint (mhtml) | ppt | BI Publisher, RTF, XSL Stylesheet (FO) |
| PowerPoint (.\*pptx) | pptx | BI Publisher, RTF, XSL Stylesheet (FO) |
| MHTML | mhtml | BI Publisher, RTF, Flash, XSL Stylesheet (FO) |
| PDF/A | pdfa | BI Publisher, RTF, XSL Stylesheet (FO) |
| PDF/X | pdfx | BI Publisher, RTF, XSL Stylesheet (FO) |
| Zipped PDFs | pdfz | BI Publisher, RTF, PDF, XSL Stylesheet (FO) |
| FO Formatted XML | xslfo | BI Publisher, RTF, XSL Stylesheet (FO) |
| Data (XML) | xml | BI Publisher, RTF, PDF, Excel, Flash, XSL Stylesheet (FO), Etext, XSL Stylesheet (HTML XML/Text) |
| Data (CSV) | csv | BI Publisher, RTF, PDF, Excel, Flash, XSL Stylesheet (FO), XSL Stylesheet (HTML XML/Text), Etext |
| XML | xml | XSL Stylesheet (HTML XML/Text) |
| Text | text | XSL Stylesheet (HTML XML/Text), Etext |
| Flash | flash | Flash |

**4 Configuring a Security Policy**

For your application to run securely you must ensure that both server-side and client-side web service security policies are configured. In Enterprise installs of Business Intelligence, the server-side policies may be set up by provisioning scripts; in these cases you only configure the client-side service policy.

For more information about the Oracle WebLogic Server web service policies, see "[Using Oracle Web Services Manager Security Policies](http://www.oracle.com/pls/topic/lookup?ctx=fa106&id=WSSOV386)" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server*.

Once the server-side policy is defined and attached as a global policy or a local policy, you enable it on the client-side by updating the xmlp-server-config.xmlfile.

**Note:**

In a provisioned Oracle Fusion Applications environment the tasks in this section are not required because authentication is configured by the provisioning scripts.

For standalone environments, SSO must be set up and the credentials for login must be identical on the server side and the client side or the user will be challenged for credentials to log in to BI Publisher.

**4.1 Viewing and Configuring the Server Policy**

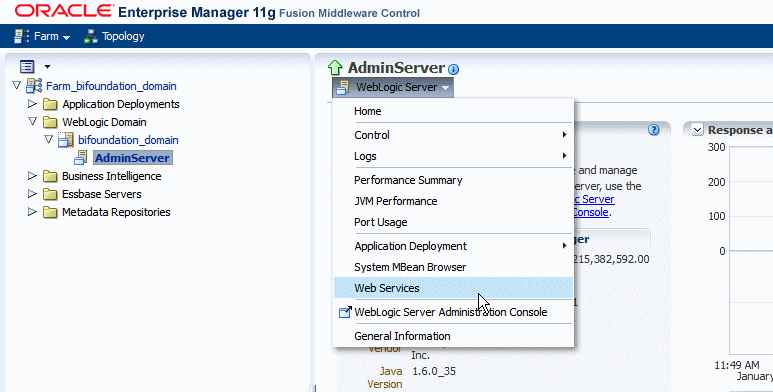
If your environment already includes a global policy or a local policy attached to the BI Publisher web service you do not need to add a policy. Use this procedure to verify the policy on the server side.

To view or set up the server policy:

1. Log in to Oracle Fusion Middleware Control. The URL is typically: http://hostname.domain:port/em
2. Open the home page for the server instance where BI Publisher is installed. In simple installations this is the AdminServer.

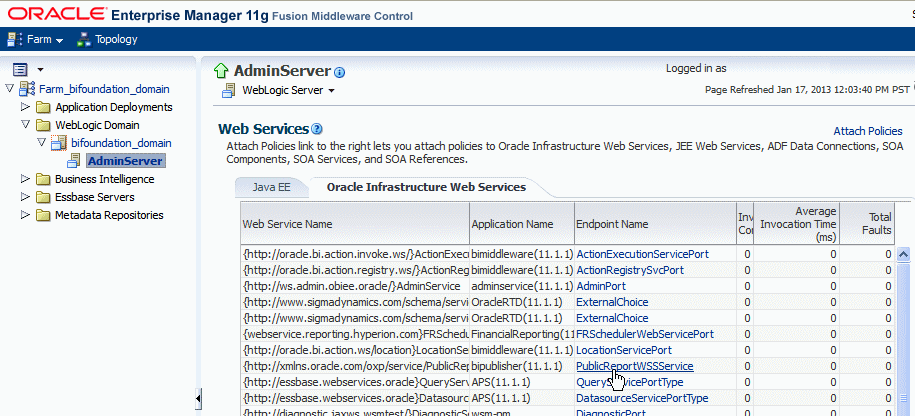
From the home page menu select Web Services as shown in [Figure 10-10](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHICFIG).

***Figure 10-10 Selecting Web Services***



1. On the Web Services page, select the BI Publisher (bipublisher(11.1.1)) service PublicReportWSService as shown in [Figure 10-11](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHCICJC).

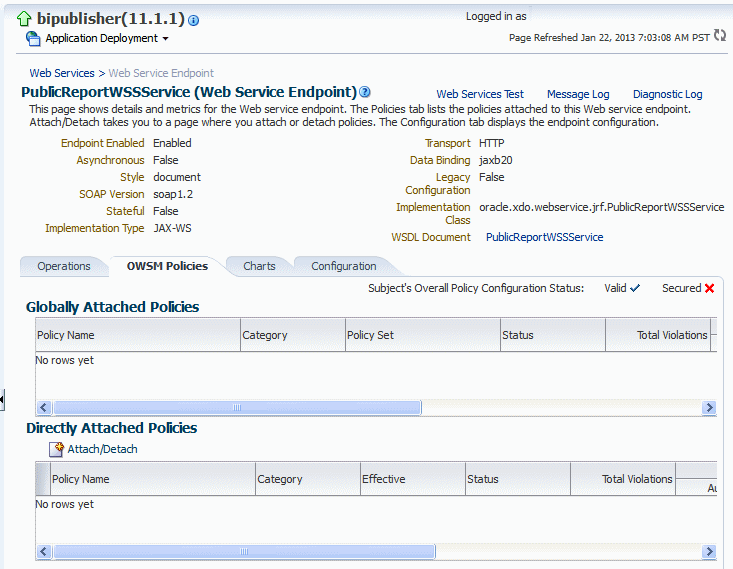
***Figure 10-11 Selecting the BI Publisher Web Service***



1. In the detail page for the web service, attached global policies display in the Global Policy table. Local policies display in the Directly Attached table. If a policy is already attached, proceed to [Section 10.4.2, "Configuring the Client-Side Service Policy."](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHJDDAB)

If there are no policies attached, attach a local (directly attached) policy by clicking Attach/Detach.

***Figure 10-12 Page Showing Attached Security Policies***



1. From the Available Policies table, select the policy to attach to the BI Publisher web service and click **Attach**. For example,oracle/wss\_saml\_or\_username\_token\_service\_policy.
2. Click OK then restart the BI Publisher application.
3. Proceed to [Section 10.4.2, "Configuring the Client-Side Service Policy."](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHJDDAB)

**4.2 Configuring the Client-Side Service Policy**

Configure the client-side service policy by making the appropriate entries in the xmlp-client-config.xml file.

On your client:

1. Navigate to the xmlp-client-config.xml in your project under Project > Web Content > WEB-INF.
2. To the properties listing, add the entry key="policy" and enter the client policy name, for example:
3. <?xml version="1.0" encoding="UTF-8"?>
4. <!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
5. <properties>
6. <comment>BIP Server Information</comment>
7. <entry key="bipuri">http://example.com:7001/xmlpserver</entry>
8. <entry key="sso.bipuri">http://example.com:7001/xmlpserver</entry>
9. <entry key="username">username</entry>
10. <entry key="policy">oracle/wss\_username\_token\_client\_policy</entry>
11. </properties>

**5 Deploying and Running the Application**

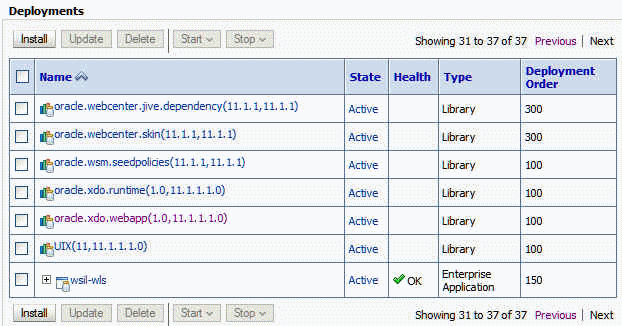
To deploy and run the application, first copy the required libraries to the client where JDeveloper is installed and then follow the deployment steps.

* [Copy Libraries](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHGGICE)
* [Deploying the Application](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHHAHDI)

**Copy Libraries**

1. Ensure that Oracle Business Intelligence Publisher shared libraries are installed on your target WebLogic Server instance.
   * oracle.xdo.runtime: $*MW\_HOME*/jdeveloper/xdo/lib/xdoruntime.ear
   * oracle.xdo.webapp: $*MW\_HOME*/jdeveloper/xdo/lib/xdowebapp.war
2. When the shared libraries are installed, you can see them in the WebLogic Server console as shown in [Figure 10-13](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHIIFBE).

***Figure 10-13 Libraries Shown in WebLogic Server Console***

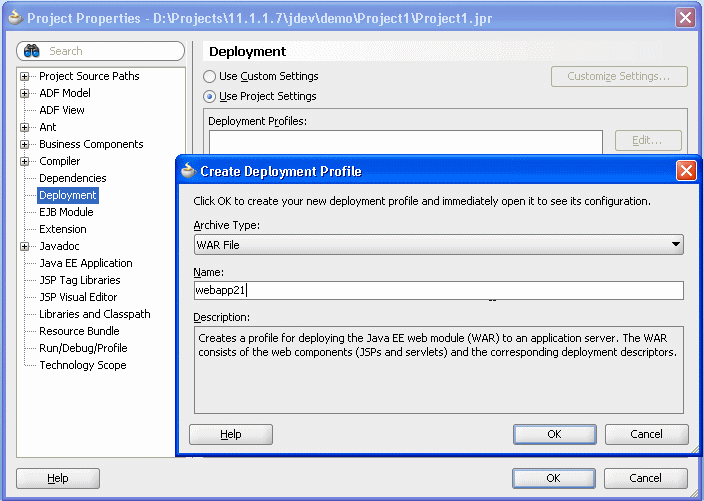


**Deploying the Application**

To deploy and run the application:

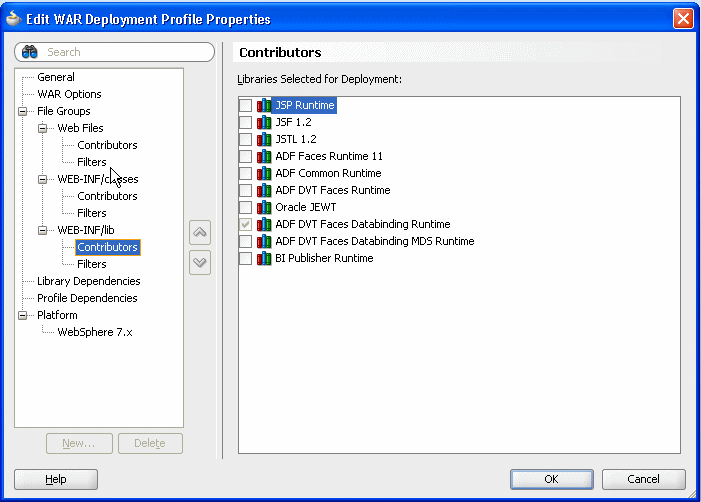
1. From Project Properties, click **Deployment**, click **New** and enter a value for the J2EE application name. [Figure 10-14](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACHJIGA) shows the Create Deployment Profile dialog.

***Figure 10-14 Create Deployment Dialog***



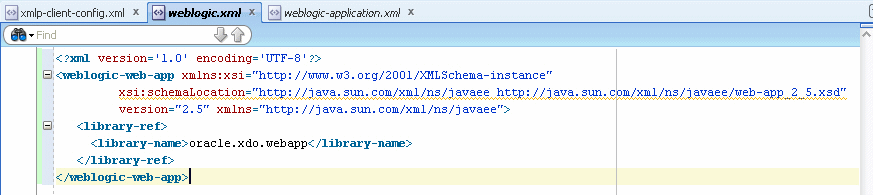
1. Click **WEB-INF/lib**. BI Publisher Runtime does not have to be included so long as **weblogic.xml** and **weblogic-application.xml** are properly configured. [Figure 10-15](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACFAGED) shows libraries selected for deployment.

***Figure 10-15 Libraries Selected for Deployment***

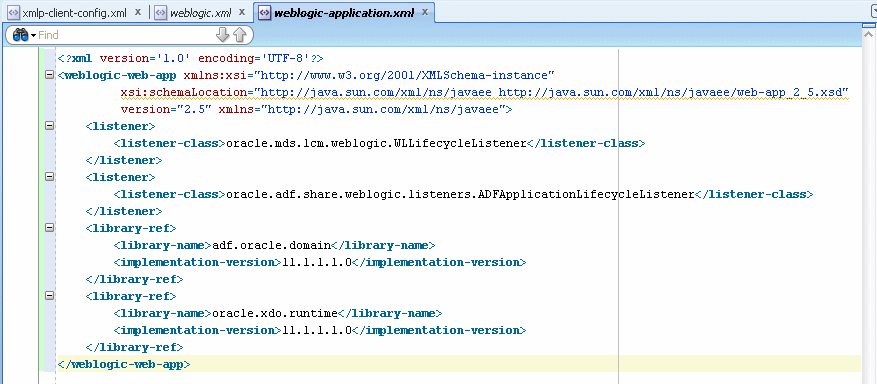


1. Add a shared library reference to weblogic.xml and weblogic-application.xml as shown in [Figure 10-16](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACBJDHH) and [Figure 10-17](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACFGBCC).

***Figure 10-16 Library Entry in weblogic.xml***

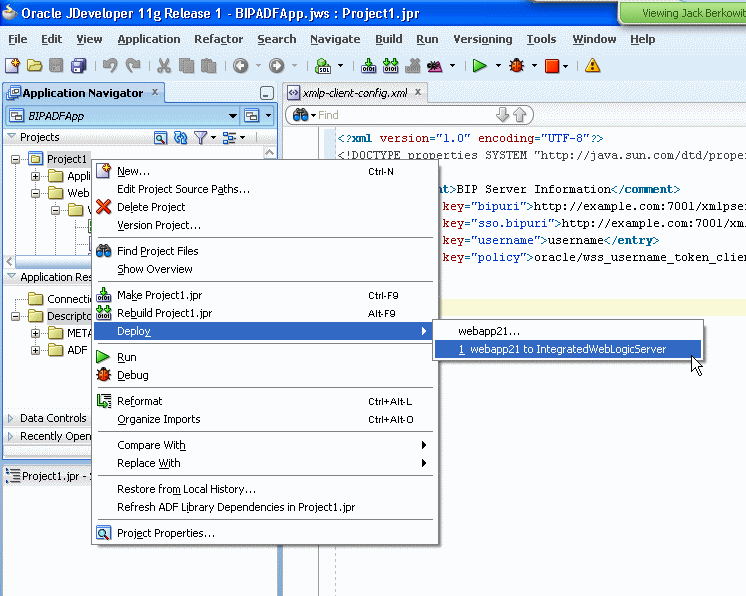


***Figure 10-17 Example weblogic-application.xml***



1. From the JDeveloper menu, select **Run > Start Server Instance**. Do not run the JSPX page directly.
2. Right-click the application and select **Deploy - (your BIP ADF Application name) - to IntegratedWLSConnection** as shown in [Figure 10-18](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACHHFGA). Make sure the BIP ADF Application deployment profile has your J2EE application name **(webapp21** in this example) selected on Application Assembly.

***Figure 10-18 Deploying an Application to a Connection***

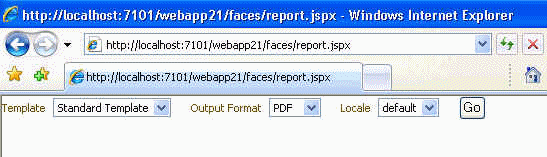


The deployment completed message displays in the Log window.

1. Open your browser and enter http://<hostname>:<port>/j2eeWebAppName/faces/report.jspx in the address field. When the page launches, the contents display according to the settings you defined for RenderActionPanel, RenderFormatList, RenderLocaleList, and RenderReportOnLoad.

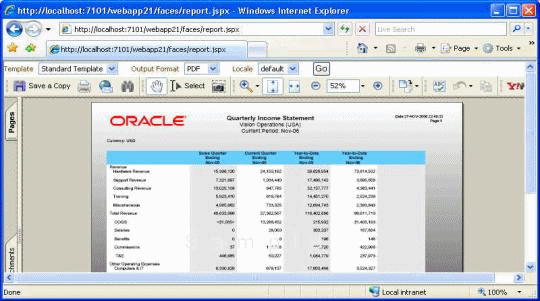
[Figure 10-19](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACFIAAF) shows an example display when RenderActionPanel, RenderFormatList, and RenderLocaleList are set to true and RenderReportOnLoad is set to false. The Template list, Output Format list, and Locale list display and can be updated by the user. To render the report, the user must click Go.

***Figure 10-19 Displayed Fields***



1. Click **Go**. Your report displays, as shown in [Figure 10-20](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CACFFFBJ).

***Figure 10-20 Generated Report***



**6 Passing Parameters from the Application Page**

You can pass parameters entered from your application page to the report using a backing bean. The BIP region tag must define the parameters property and bind it to the backing bean method as shown:

<?xml version='1.0' encoding='UTF-8'?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:af="http://xmlns.oracle.com/adf/faces/rich"

xmlns:xdo="http://xmlns.oracle.com/xdo/faces">

<jsp:directive.page contentType="text/html;charset=UTF-8"/>

<f:view>

<af:document id="d1">

<af:form id="f1">

<p>

<af:inputText label="Employee ID" id="it1"

binding="#{UIBackingBean.empId}"/>

</p>

<p>

?[34m|

</p>

<p>

<af:separator id="s1"/>

</p>

<xdo:BIPRegion id="bipregion1" reportId="/Samples/Overview/Balance+Letter+Report.xdo"

width="1000px" height="800px"

**parameters="#{UIBackingBean.parameters}**" reportData=""/>

<f:facet name="second"/>

</af:form>

</af:document>

</f:view>

</jsp:root>

Specify the backing bean in the Parameters property of the Property Inspector. See [Table 10-2, "Advanced Properties"](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHFFDAI) for more information.

**7 Using a Push Data Model**

Some reports require XML data from a source other than a BI Publisher data model. For this requirement, you can push data to the BI Publisher server to use as input for your report. For this scenario, store the data in a location that is accessible by your application and create a backing bean that defines the getter method to retrieve the report data from its stored location. You still define the Report Path, Layout Name and other properties for the report as defined on the BI Publisher server.

The BIPRegion tag must define the reportData property for the tag, and bind it to the backing bean method. The following sample code shows this binding:

<?xml version='1.0' encoding='UTF-8'?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:af="http://xmlns.oracle.com/adf/faces/rich"

xmlns:xdo="http://xmlns.oracle.com/xdo/faces">

<jsp:directive.page contentType="text/html;charset=UTF-8"/>

<f:view>

<af:document id="d1">

<af:form id="f1">

<xdo:BIPRegion id="id3"

reportId="/Samples/Overview/Balance+Letter+Report.xdo"

width="1000px" height="800px"

**reportData="#{UIBackingBean.reportData}**"/>

</af:form>

</af:document>

</f:view>

</jsp:root>

The following code sample shows the UIBackingBean class referenced in the previous sample:

package fusionApp;

import java.io.RandomAccessFile;

import java.util.Hashtable;

import java.util.Properties;

import oracle.adf.view.rich.component.rich.input.RichInputText;

public class UIBeackingBean {

private RichInputText empId;

private Properties mParameters;

private byte[] reportData;

public UIBeackingBean() {

mParameters = new Properties();

reportData = null;

}

public void setEmpId(RichInputText empId) {

this.empId = empId;

}

public RichInputText getEmpId() {

return empId;

}

public Hashtable getParameters() {

if(empId != null && empId.getValue() != null)

{ mParameters.put("userid", new String[]{empId.getValue().toString()});

String[] values = (String[])mParameters.get("userid");

System.out.println("getParameters() is called : " + values[0]);

}

return mParameters;

}

public byte[] getReportData() {

String dataFile = "/tmp/reportData.xml";

try

{

RandomAccessFile raf = new RandomAccessFile(dataFile, "r");

reportData= new byte[(int)raf.length()];

raf.read(reportData);

raf.close();

//write this to temp

java.io.FileOutputStream outStream = new

java.io.FileOutputStream("/tmp/output\_reportData");

outStream.write(reportData);

outStream.close();

} catch (Exception e) {

System.out.println("Error reading file : " + e.getMessage());

}

return reportData;

}

}

**To define a push data model**:

1. Create a backing bean class that defines the getter method that retrieves your data from its stored location.
2. In JDeveloper, create the BIP Region for your page and specify all properties for the report that you wish to run on the BI Publisher server (Report Path, Layout Name, Output Format, and so on).
3. In the Property Inspector, for the ReportData property, enter the expression that defines the variable to pass to the backing bean method to call the appropriate getter method to retrieve the report data from its stored location. Use the Expression Builder to build the unified expression language (EL) syntax.

**8 Conditionally Required Settings**

Depending on the type of report you are running you may need to make other settings for your reports to run as expected.

This section contains the following topics:

* [Section 10.8.1, "Setting the MIME Types for Your Report Output Type"](http://docs.oracle.com/cd/E36909_01/fusionapps.1111/e26385/jdev.htm#CCHBAGBG)

**8.1 Setting the MIME Types for Your Report Output Type**

To ensure that all the output types can be successfully generated from your page, add the MIME-type mappings to the web.xml file. Typically you only need to add the MIME-type mappings for Excel output.

To add the MIME-type mappings:

1. Double-click the Project web.xml file to open it for editing.
2. Expand the MIME mappings region.
3. Click Add.
4. In the Property Inspector, enter the following:
   * extension - (Required) Enter the file name extension of the document type you want to map to a particular MIME type for your web application. For example, pdf. This field corresponds to the <extension> tag of the <mime-mapping> subelement.
   * mime-type - (Required) Enter the document MIME type that you want to map to the specified file name extension. For example, application/pdf for Adobe's Portable Document Format. This field corresponds to the <mime-type> tag of the <mime-mapping> subelement. Note, the official registry of Internet MIME types is managed by the Internet Assigned Numbers Authority (IANA) at www.iana.org.

The following table shows the required MIME types to generate the Microsoft Excel output options supported by BI Publisher:

| **Extension** | **Mime-Type** |
| --- | --- |
| xls | application/vnd.ms-excel |
| xlsx | application/vnd.openxmlformatsofficedocument.spreadsheetml.sheet |
| xlsm | application/vnd.ms-excel.sheet.macroEnabled.12 |
| mhtml | message/rfc822 |