professional it software & services

ORACLE Gold Partner

How to integrate Oracle BI Publisher via Web Services in Oracle Forms 11g

Version 4.0

White Paper, May 2013

Authors: Axel Harsch, PITSS
Jürgen Menge, Oracle
Florin Serban, PITSS
Rainer Willems, Oracle

Contributors: Mireille Duroussaud, Oracle

Introduction

This paper describes in a step-by-step way the generation of a client for the Web Services provided by Oracle BI Publisher and how to import and use this client in Oracle Forms 11g. This document is providing an update to the version 2.2, "How to integrate Oracle BI Publisher via Web Services in Oracle Forms"[1], including supplemental information regarding the integration between Oracle Forms 11g and BI Publisher 11g.

Used versions:

- Oracle Forms 11.1.2.0
- Oracle JDeveloper 11.1.2.3
  http://www.oracle.com/technetwork/developer-tools/jdev/downloads/index.html
- BI Publisher Enterprise 11.1.1.6 on WebLogic Server 10.3.5

The example included in this document should also work with Forms 11g, Rel. 2 (11.1.2.x) and JDeveloper 11g, Rel. 2 (11.1.2.x)

Oracle BI Publisher Web Service API

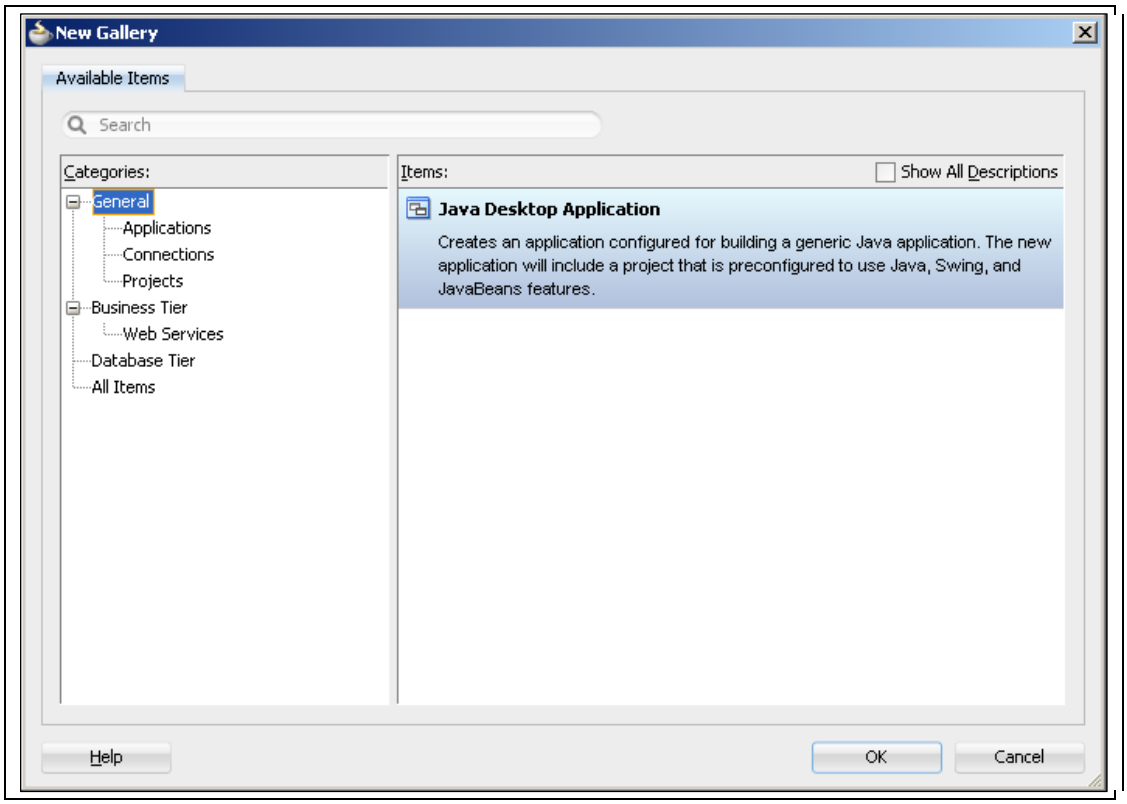
With Oracle BI Publisher 10.1.3.3.1 a Public Web Service API was firstly introduced into the product. After this initial implementation several versions of web services were integrated into the product. With the current release it is recommended a set of v2 web services which are documented here http://docs.oracle.com/cd/E23943_01/bi.1111/e22259/toc.htm

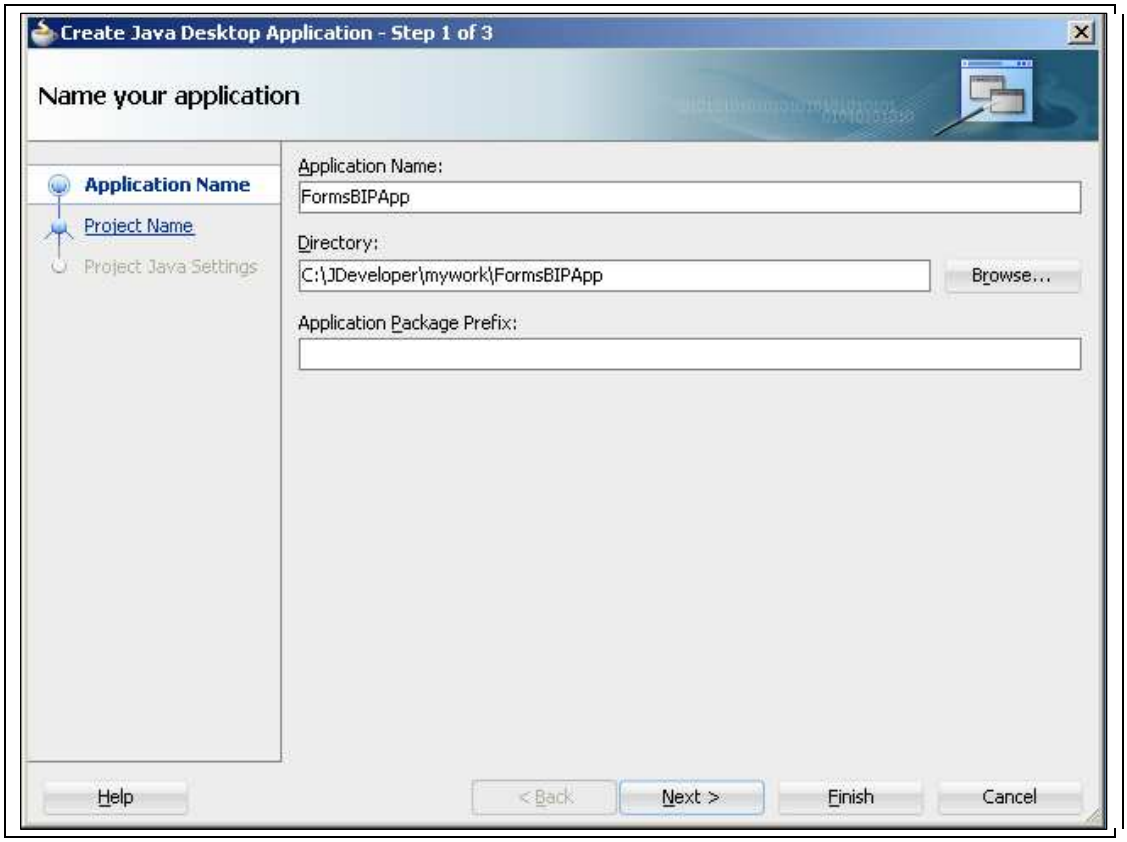
If Oracle BI Publisher is installed on a server the WSDL for the reporting web service can be found at: http://<host>:<port>/xmlpserver/services/v2/ReportService?wsdl

[1] http://www.oracle.com/technetwork/middleware/bi-publisher/overview/forms-bip-v22-129995.pdf

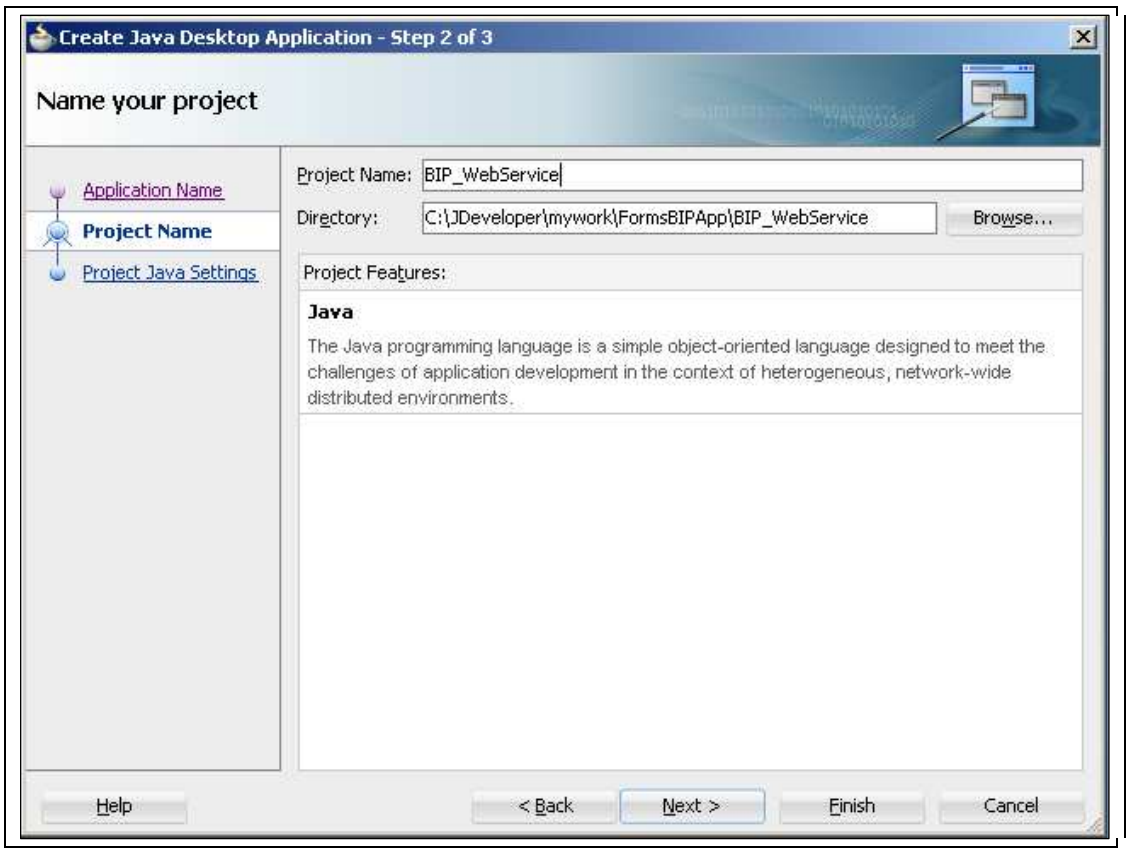
Creating a client for the Web Services with Oracle JDeveloper

We need to create a new application with Oracle JDeveloper, by selecting **Java Desktop Application** in the **New Gallery** dialog and entering the desired application name:





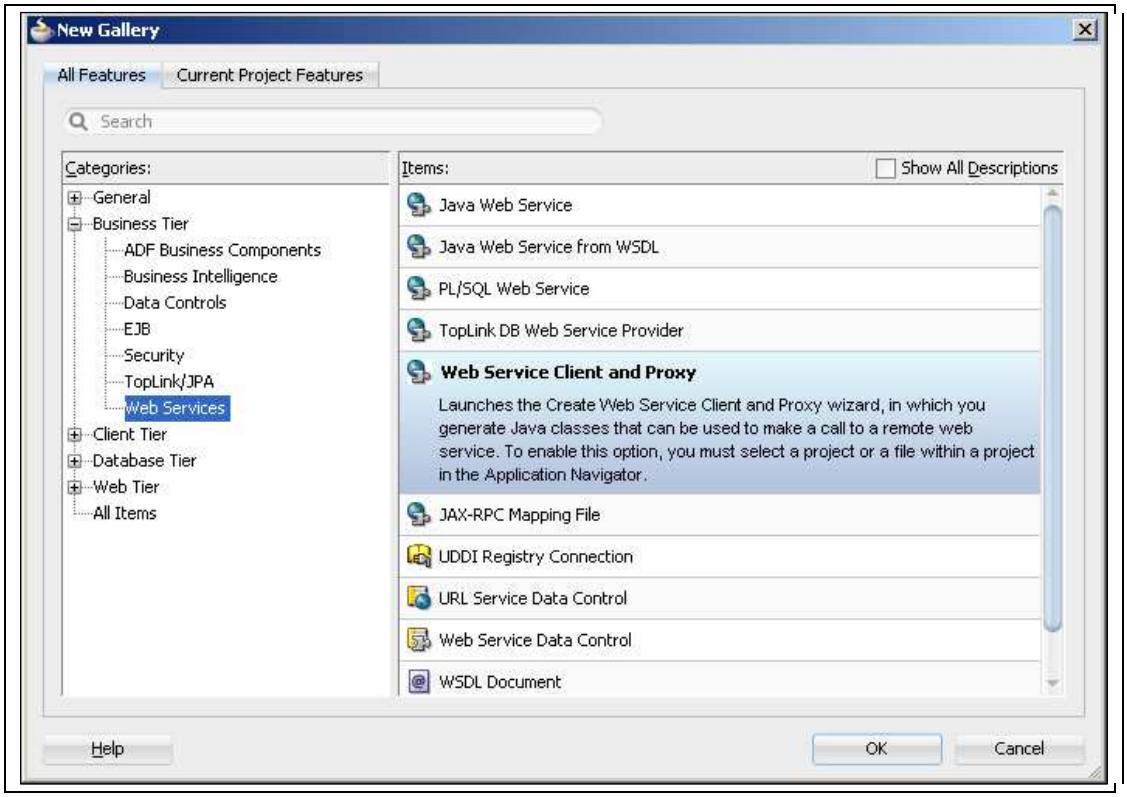
We then name the project as *BIP_WebService* and press the **Finish** button.



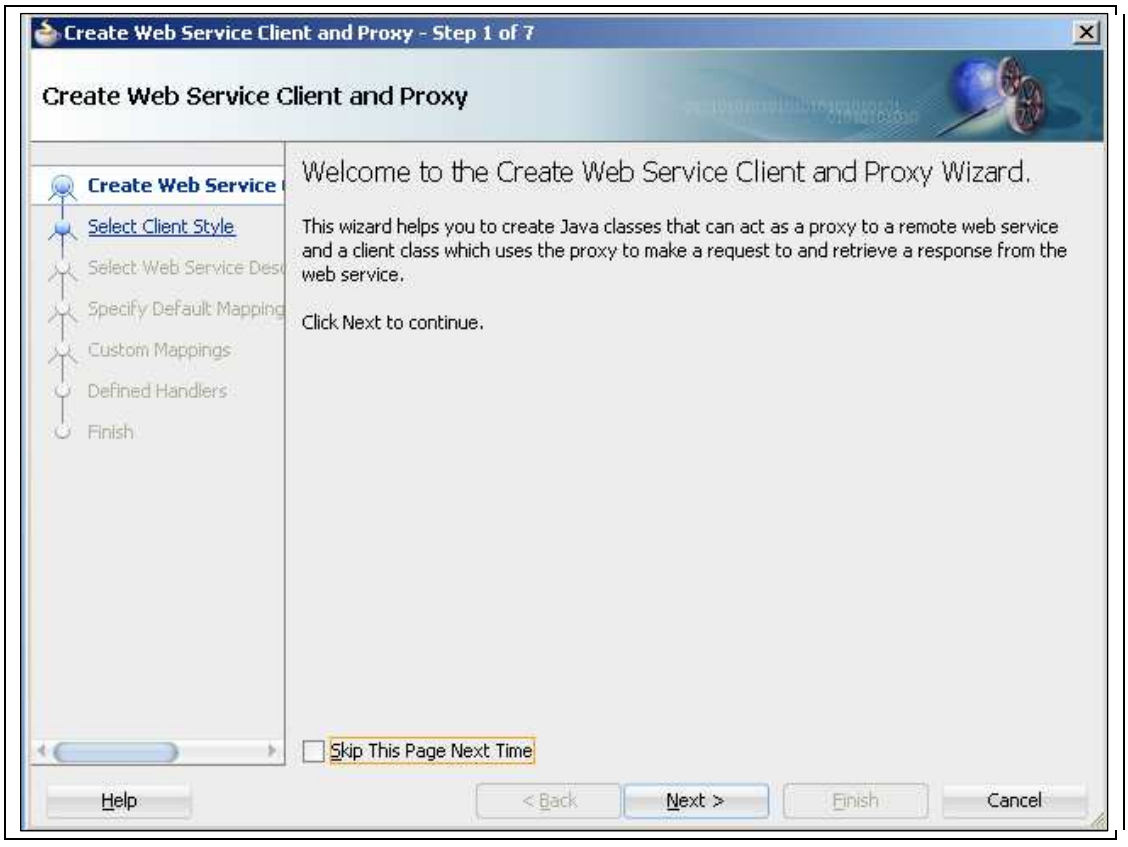
Note: The code examples in this document are referring this project name, *BIP_WebService*. In case another name is needed, the code will have to be adjusted accordingly.

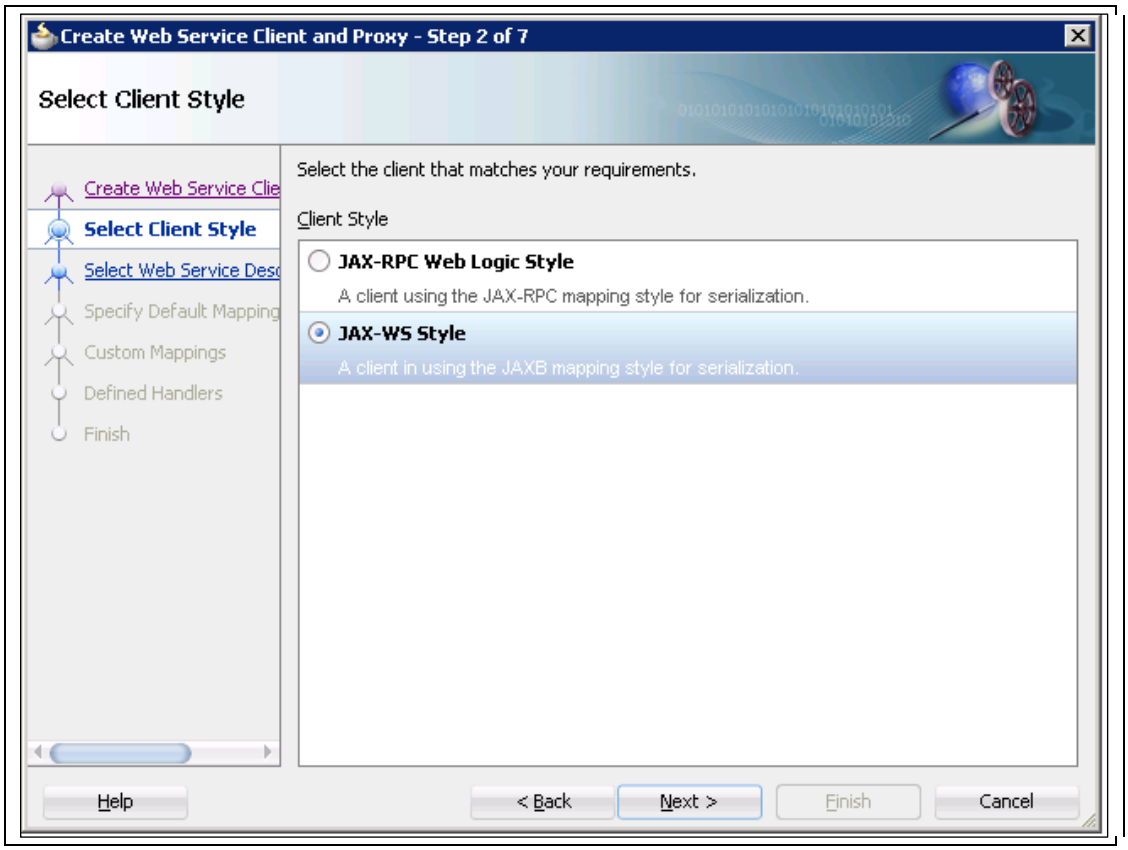
Note: In case a HTTP-Proxy is necessary, please define this in the **Tools** menu under **Preferences** at **Web Browser and Proxy**.

After the project is saved, right-click the project, choose **New** and in the **All Features** tab page select the Item **Web Services Client and Proxy** in the Category **Business Tier – Web Services**.



Press **OK** and the Web Service wizard is opened. Accept the first two screens.

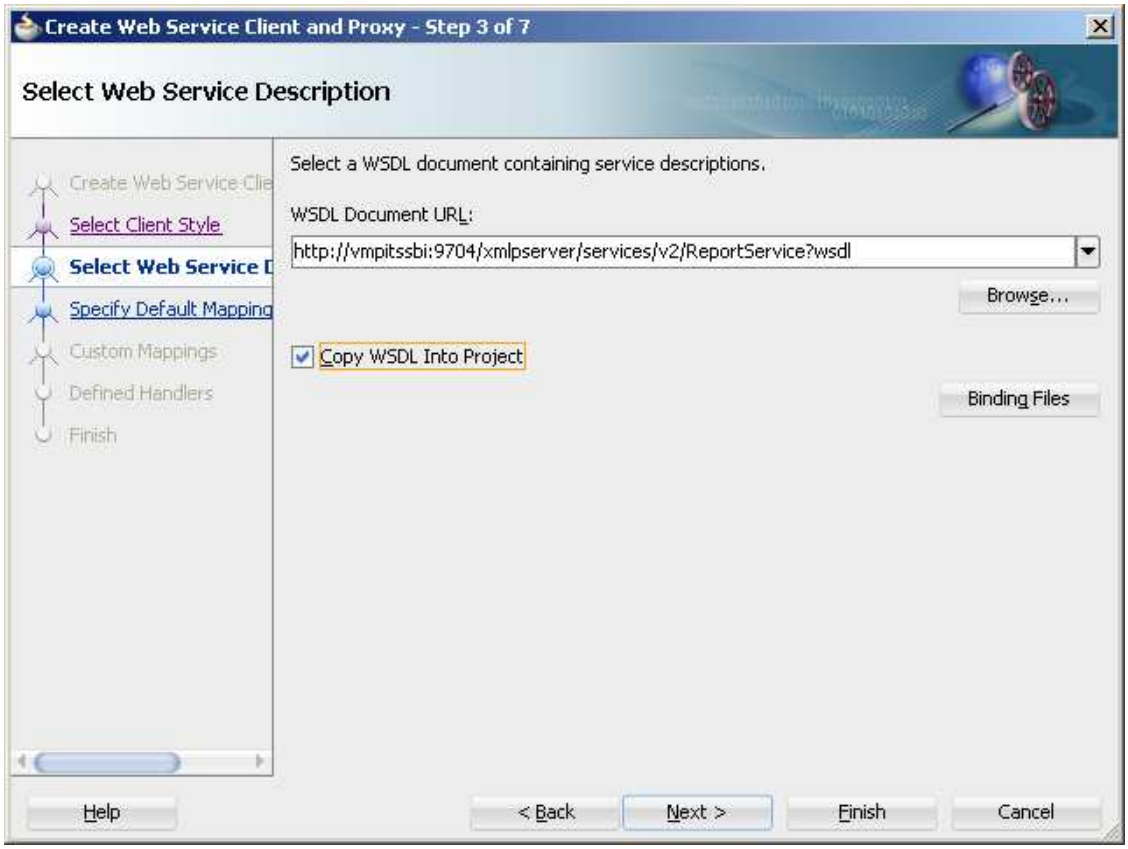




Choose the appropriate URL of the web service endpoint of your BI Publisher Server in the **WSDL Document URL**:

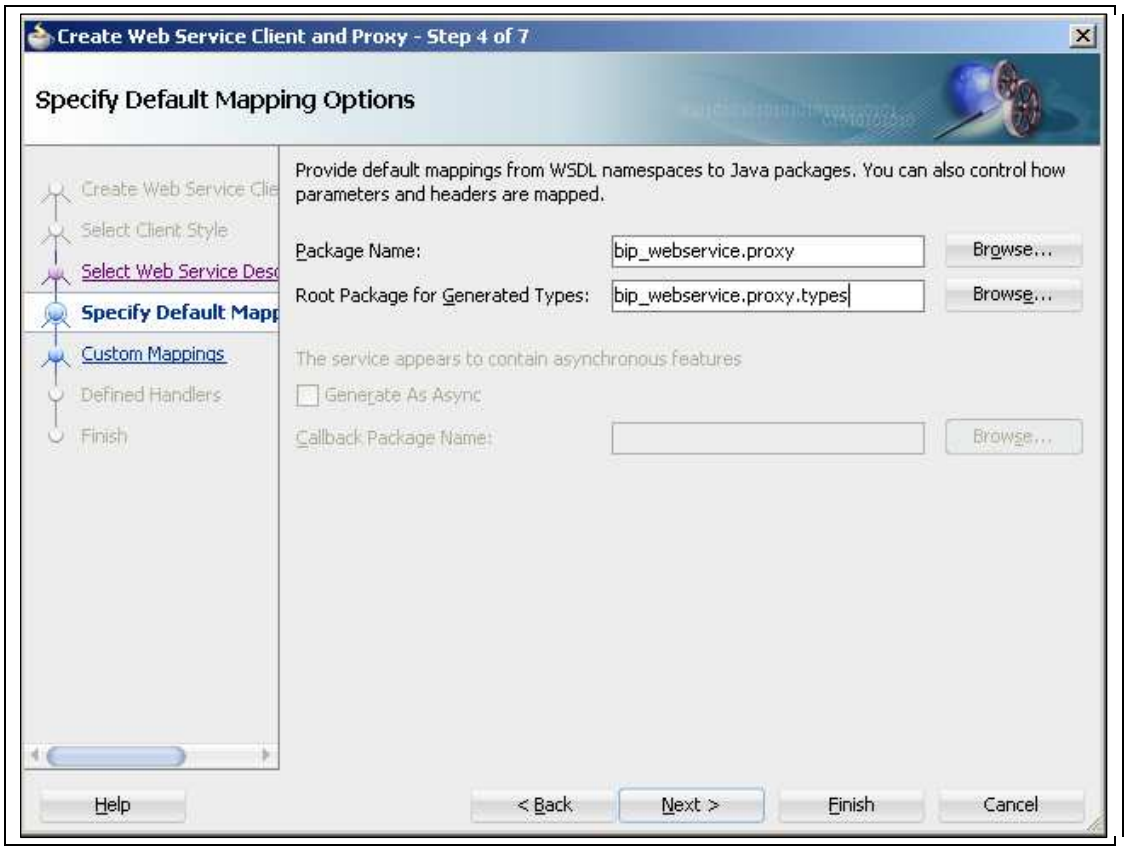
http://<machine>:<port>/<yourappname>/services/v2/ReportService?wsdl

(Default for <yourappname> is *xmlpserver*)

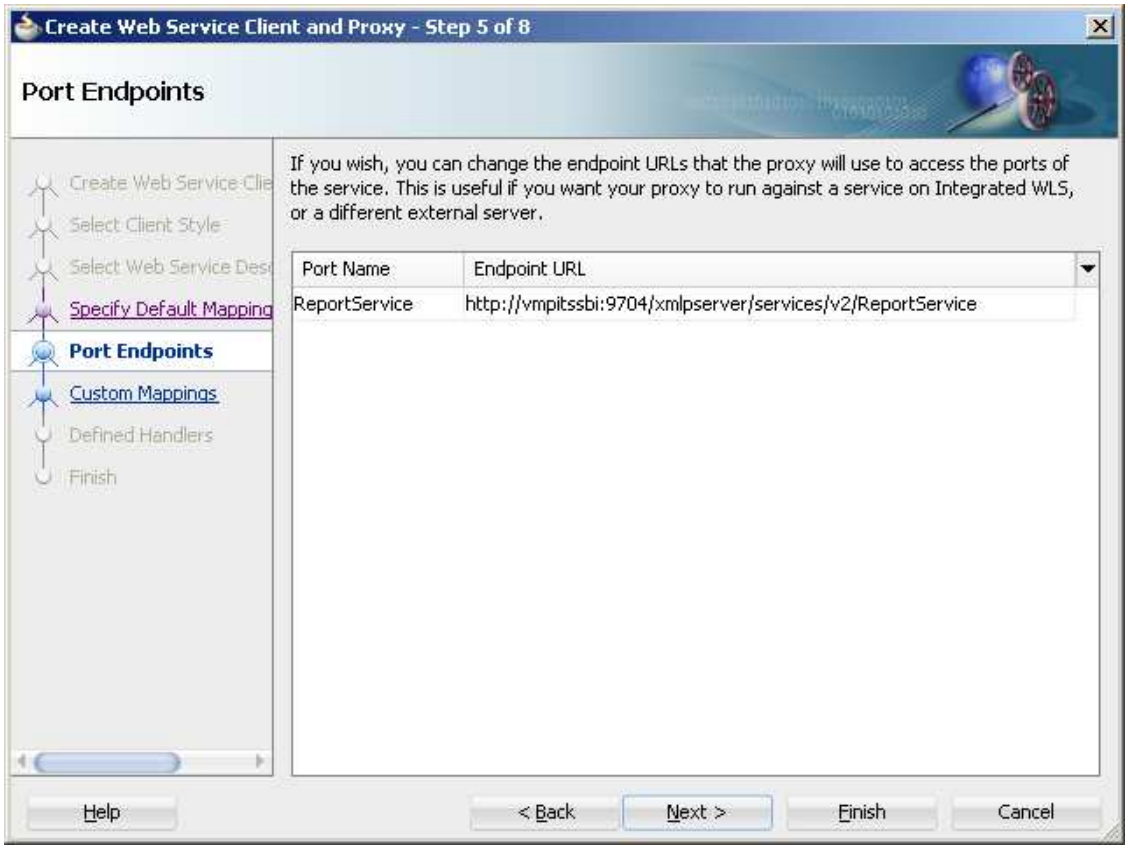


Note: If for any reason a connection to this endpoint URL from JDeveloper is not possible, you can download the WSDL file in a web browser and copy it into a local directory of your application. Afterwards point to this local copy and proceed as described here.

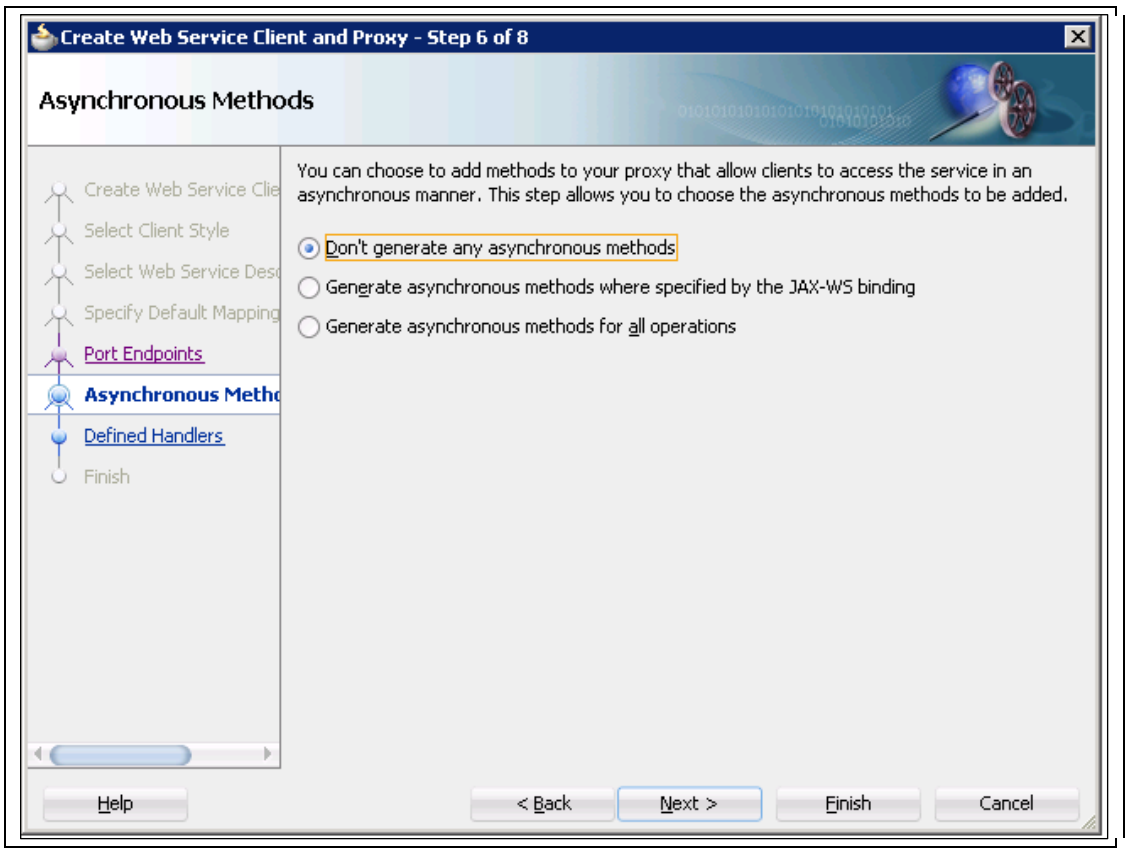
Set the desired **Package Name** (here *bip_webservice.proxy*).



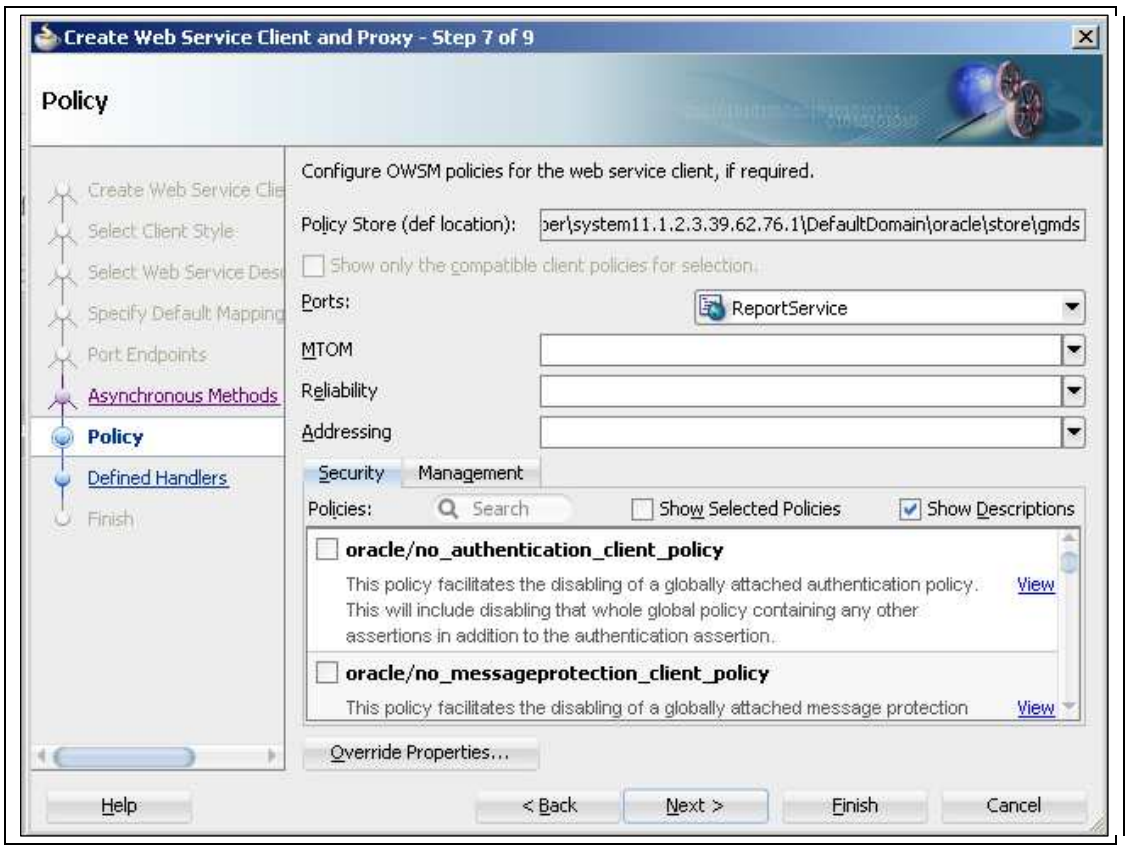
Accept the next screen.

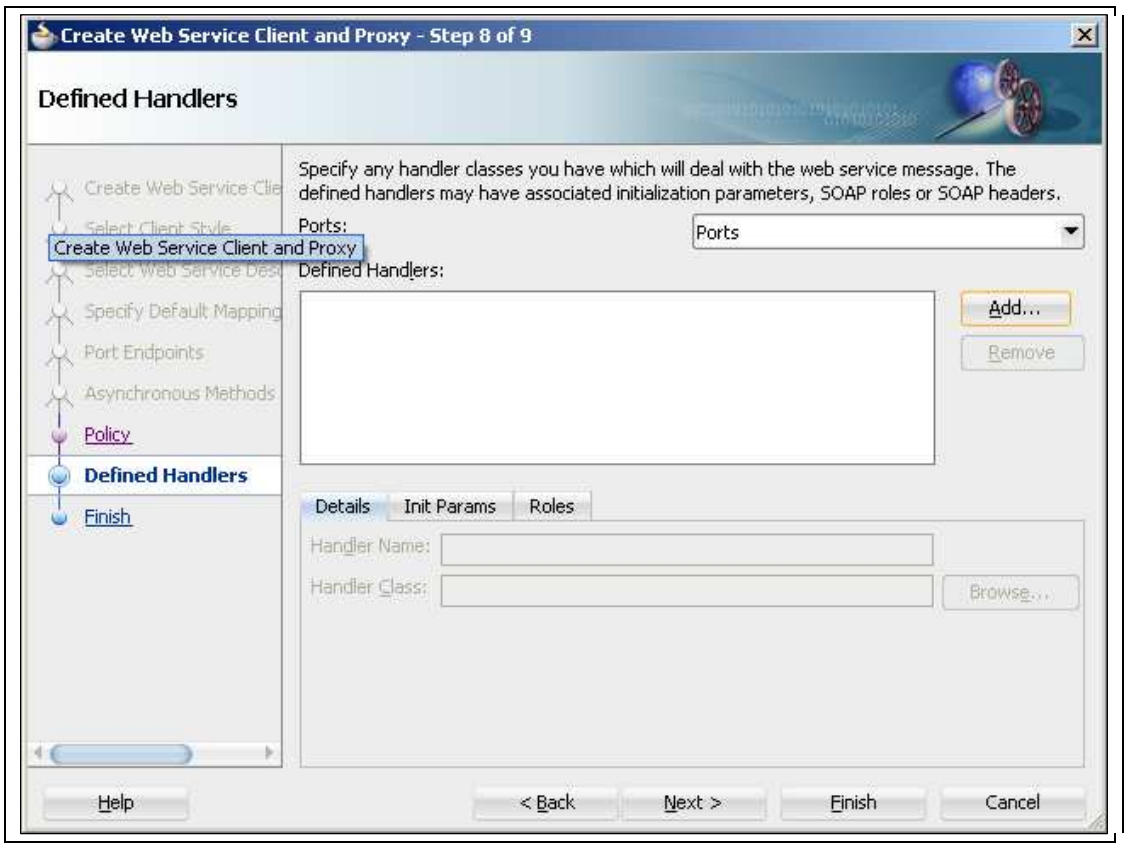


Select **Don't generate any asynchronous methods** in the next screen.

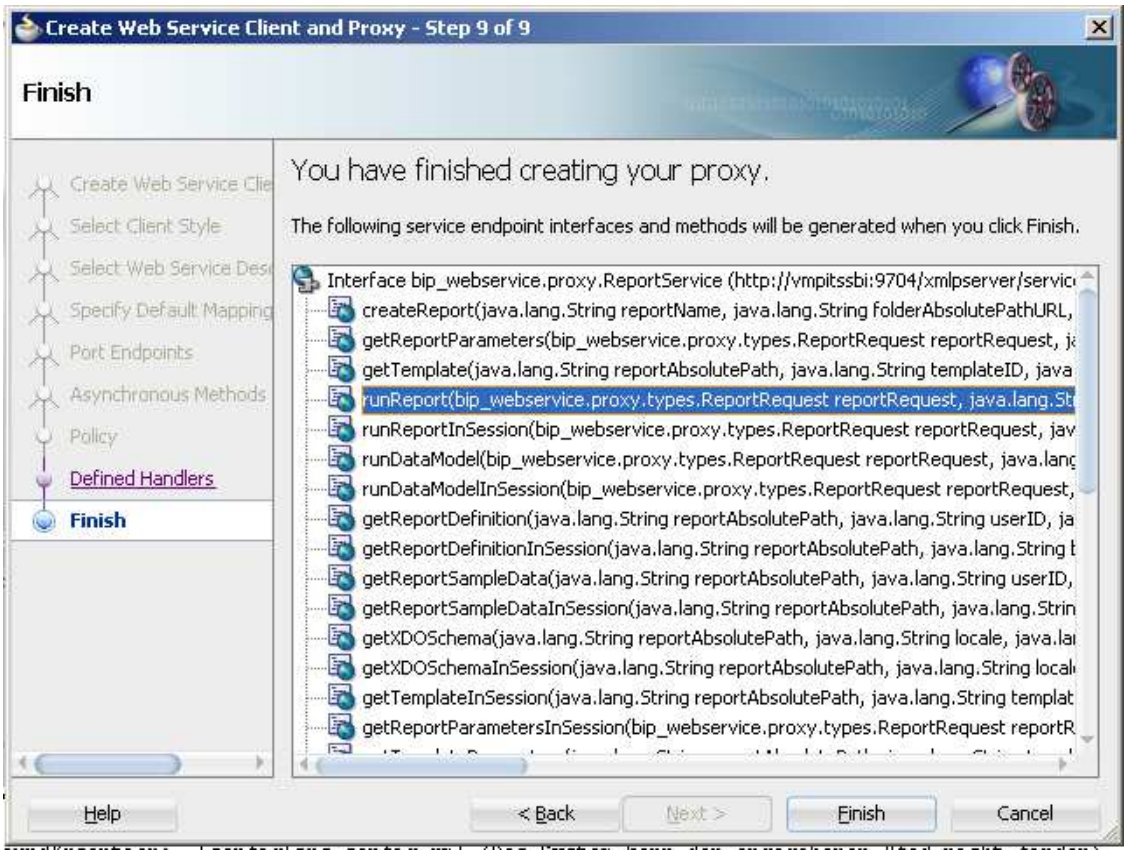


Accept the following two screens.

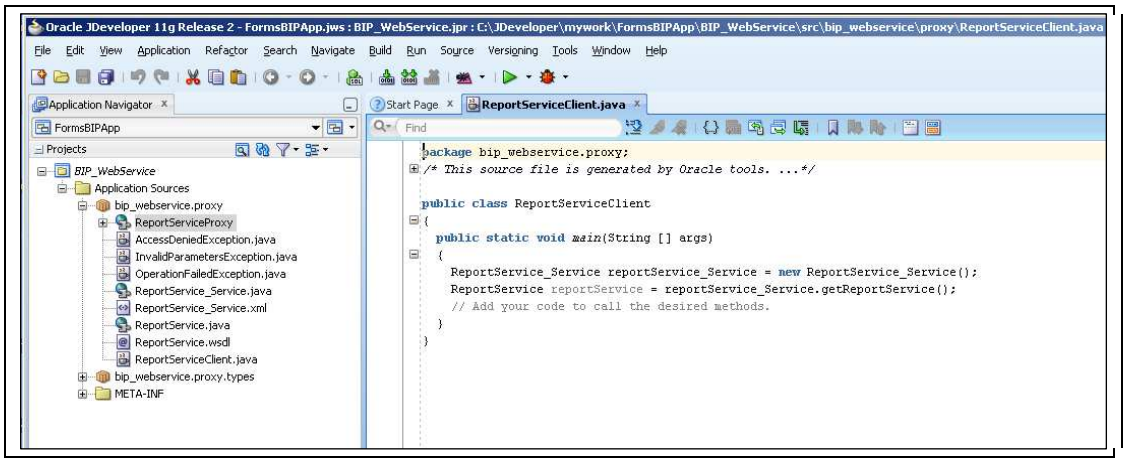




Check that the *runReport()* method is available in the **Finish** screen.



JDeveloper creates a proxy stub (proxy client) to call the selected web service methods.



Now the web service needs to be tested. For this, the *main*() method needs to be edited in the *ReportServiceClient* class (please find the line „// Add your code here“), as in the following code fragment. This will call the BI report which was created with the help of the following document: Getting Started with Oracle Business Intelligence Publisher
http://docs.oracle.com/html/E28374_01/toc.htm

Note: Adjust username, passwords, directory names accordingly to your environment.

```
public static void main(String [] args) throws AccessDeniedException,
InvalidParametersException, OperationFailedException, IOException
  {
    try
    {
      ReportService_Service reportServiceService = new
ReportService_Service();
      ReportService         reportService        =
reportServiceService.getReportService();

      // Add your code to call the desired methods.
      final String username = "weblogic";
      final String password = "Welcome1";
      final String reportAbsolutePath = "~weblogic/My Data Models/My
Reports/Employee Report.xdo";

      // Testing runReport
      System.out.println("Testing runReport Service");
      ReportRequest repRequest = new ReportRequest();

      //Set general Report Parameters
      repRequest.setReportAbsolutePath(reportAbsolutePath);
      repRequest.setAttributeTemplate("Chart and Table Layout");
      repRequest.setAttributeFormat("pdf");
      repRequest.setAttributeLocale("en-US");
      repRequest.setSizeOfDataChunkDownload(-1);

      //Set User Parameter P_DEPT
      ArrayOfParamNameValue arrayOfParamNameValue = new
ArrayOfParamNameValue();
      ParamNameValues paramNameValues = new ParamNameValues();
      ParamNameValue  paramNameValue  = new ParamNameValue();
      ArrayOfString arrayOfString = new ArrayOfString();
      paramNameValue.setName("P_DEPT");
      arrayOfString.getItem().add("RESEARCH");
      paramNameValue.setValues(arrayOfString);
      arrayOfParamNameValue.getItem().add(paramNameValue);
      paramNameValues.setListOfParamNameValues
(arrayOfParamNameValue);
      repRequest.setParameterNameValues(paramNameValues);

      ReportResponse repResponse = new ReportResponse();
      repResponse = reportService.runReport(repRequest, username,
password);
      String contentType = repResponse.getReportContentType();
      System.out.println(contentType);
      byte[] baReport = repResponse.getReportBytes();
      FileOutputStream fio = new FileOutputStream("C:\\test.pdf");
        fio.write(baReport);
        fio.close();
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
```

After following all the JDeveloper' hints to import necessary classes, the code should look as in the following screenshot:

```
package bip_webservice.proxy;
import bip_webservice.proxy.types.ArrayOfParamNameValue;
import bip_webservice.proxy.types.ArrayOfString;
import bip_webservice.proxy.types.ParamNameValues;
import bip_webservice.proxy.types.ParamNameValue;
import bip_webservice.proxy.types.ReportRequest;
import bip_webservice.proxy.types.ReportResponse;

import java.io.FileOutputStream;
import java.io.IOException;
// This source file is generated by Oracle tools.
// Contents may be subject to change.
// For reporting problems, use the following:
// Generated by Oracle JDeveloper 11g Release 2 11.1.2.3.0.6276

public class ReportServiceClient
{
  public static void main(String [] args) throws AccessDeniedException, InvalidParametersException,
                                                  OperationFailedException, IOException
  {
    try
    {
      ReportService_Service reportServiceService = new ReportService_Service();
      ReportService        reportService        =  reportServiceService.getReportService();

      // Add your code to call the desired methods.
      final String username = "weblogic";
      final String password = "Welcome1";
      final String reportAbsolutePath = "~weblogic/My Data Models/My Reports/Employee Report.xdo";

      // Testing runReport
      System.out.println("Testing runReport Service");
      ReportRequest repRequest = new ReportRequest();

      //Set general Report Parameters
      repRequest.setReportAbsolutePath(reportAbsolutePath);
      repRequest.setAttributeTemplate("Chart and Table Layout");
      repRequest.setAttributeFormat("pdf");
      repRequest.setAttributeLocale("en-US");
      repRequest.setSizeOfDataChunkDownload(-1);

      //Set User Parameter P_DEPT
      ArrayOfParamNameValue arrayOfParamNameValue = new ArrayOfParamNameValue();
      ParamNameValues paramNameValues = new ParamNameValues();
      ParamNameValue  paramNameValue  = new ParamNameValue();
      ArrayOfString arrayOfString = new ArrayOfString();
      paramNameValue.setName("P_DEPT");
      arrayOfString.getItem().add("RESEARCH");
      paramNameValue.setValues(arrayOfString);
      arrayOfParamNameValue.getItem().add(paramNameValue);
      paramNameValues.setListOfParamNameValues(arrayOfParamNameValue);
      repRequest.setParameterNameValues(paramNameValues);

      ReportResponse repResponse = new ReportResponse();
      repResponse = reportService.runReport(repRequest, username, password);
      String contentType = repResponse.getReportContentType();
      System.out.println(contentType);
      byte[] baReport = repResponse.getReportBytes();
      FileOutputStream fio = new FileOutputStream("C:\\test.pdf");
      fio.write(baReport);
      fio.close();
    } catch (Exception ex) {
      ex.printStackTrace();
    }
  }
}
```

Starting with BI Publisher 10.1.3.4 there is a new and important parameter in the complex type *ReportRequest - SizeOfDataChunkDownload*. According to the documentation this parameter should be set to **-1** if you don't want to split the resulting data into chunks. Otherwise BI Publisher will produce an empty file.

The Web Service Client may be tested now by clicking **Run** in the context menu of *ReportServiceClient.java* or in the toolbar. During the test the method *main()* is used.

To call the *runReport()* method from our Forms application we need some additional code because we should not:

- use the method *main()* from outside
- modify the generated method *runReport()*.

There are two conceptual ways to do that:

1. to write a wrapper class around *ReportServiceClient.java*
2. to write an additional, customized method in the *ReportServiceClient.java* which will be called from outside

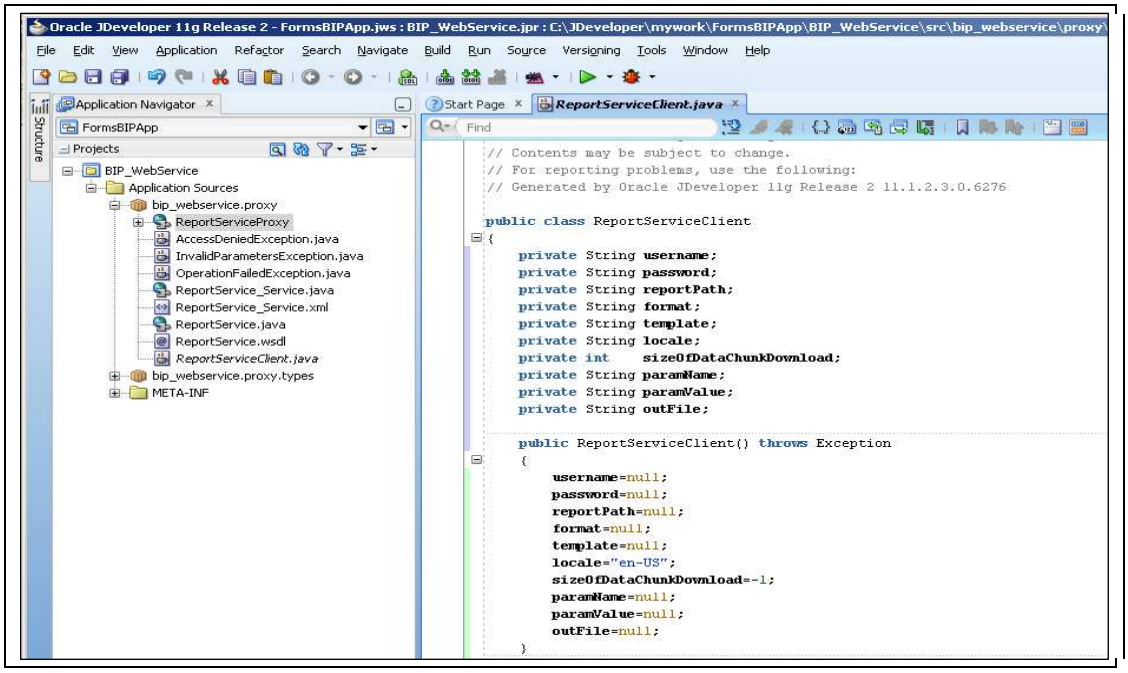
For reasons of simplicity we will select here the second solution and add a method *callRunReport()* in the *ReportServiceClient.java*.

First we define the parameters in the class *PublicReportServiceClient*:

```
public class ReportServiceClient {
private String username;
private String password;
private String reportPath;
private String format;
private String template;
private String locale;
private int    sizeOfDataChunkDownload;
private String paramName;
private String paramValue;
private String outFile;
```

and initialize them

```
public ReportServiceClient() throws Exception
{
  username = null;
  password = null;
  reportPath = null;
  format = null;
  template = null;
  locale = "en-US";
  sizeOfDataChunkDownload = -1;
  paramName = null;
  paramValue = null;
  outFile = null;
}
```



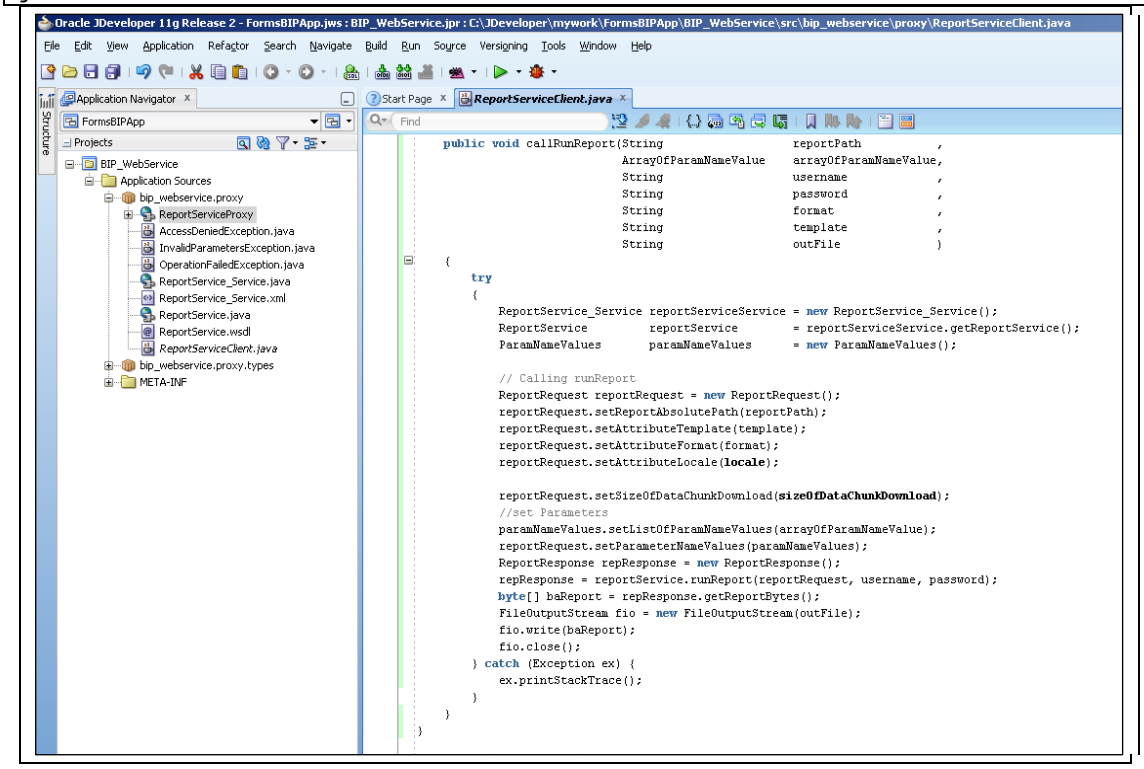
Now we will add a public method *callRunReport()* that will call the generated method *runReport()* in the web service client. The key argument of this method is called *arrayOfParamNameValue* and is used to transport the information regarding the user parameters (pairs of name and values).

```
public void callRunReport(String                    reportPath           ,
                          ArrayOfParamNameValue     arrayOfParamNameValue,
                          String                    username             ,
                          String                    password             ,
                          String                    format               ,
                          String                    template             ,
                          String                    outFile              ){
  try {
    ReportService_Service reportServiceService = new ReportService_Service();
    ReportService         reportService        =
reportServiceService.getReportService();
    ParamNameValues       paramNameValues      = new ParamNameValues();

    // Calling runReport
    ReportRequest reportRequest = new ReportRequest();
    reportRequest.setReportAbsolutePath(reportPath);
    reportRequest.setAttributeTemplate(template);
    reportRequest.setAttributeFormat(format);
    reportRequest.setAttributeLocale(locale);
    reportRequest.setSizeOfDataChunkDownload(sizeOfDataChunkDownload);

    //set Parameters
    paramNameValues.setListOfParamNameValues(arrayOfParamNameValue);
    reportRequest.setParameterNameValues(paramNameValues);
    ReportResponse repResponse = new ReportResponse();
    repResponse = reportService.runReport(reportRequest, username, password);
    byte[] baReport = repResponse.getReportBytes();
    FileOutputStream fio = new FileOutputStream(outFile);
    fio.write(baReport);
    fio.close();
  } catch (Exception ex) {
    ex.printStackTrace();
  }
```

```
}
```

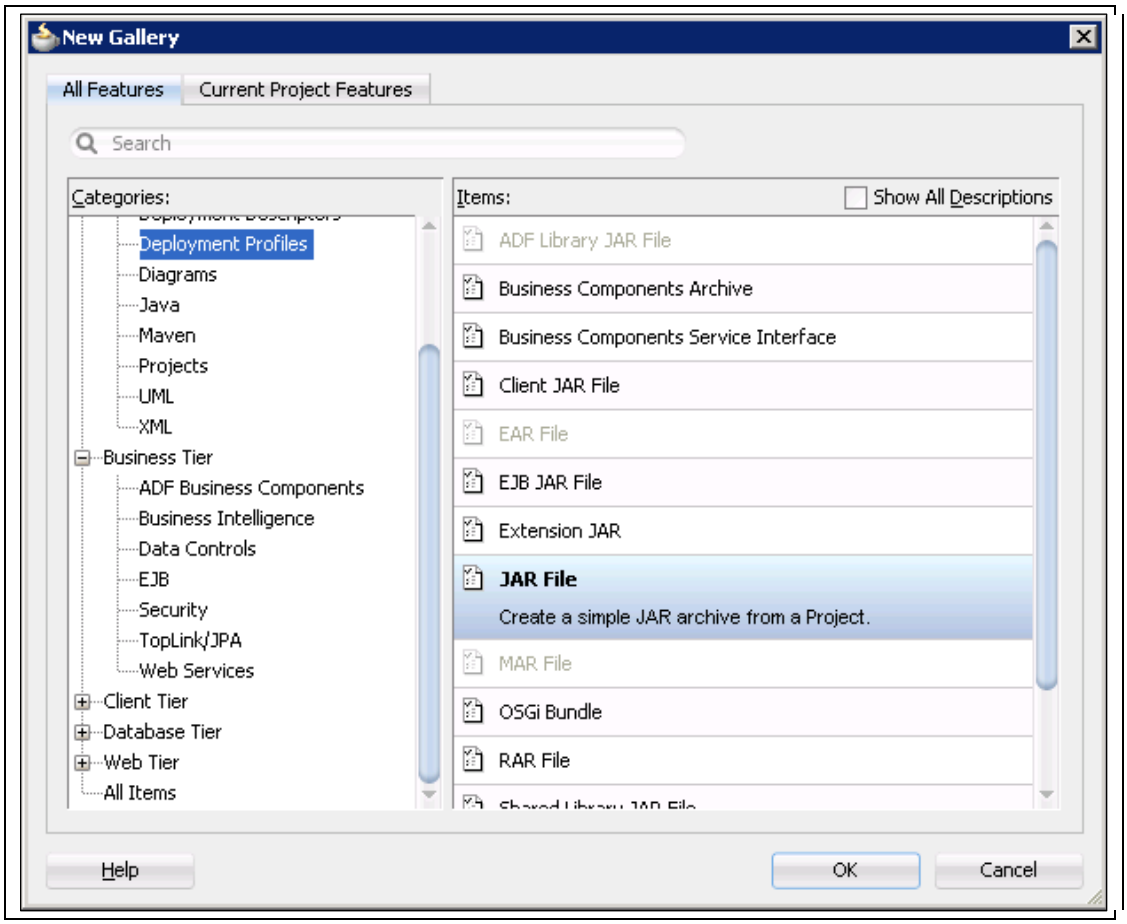


Hint:

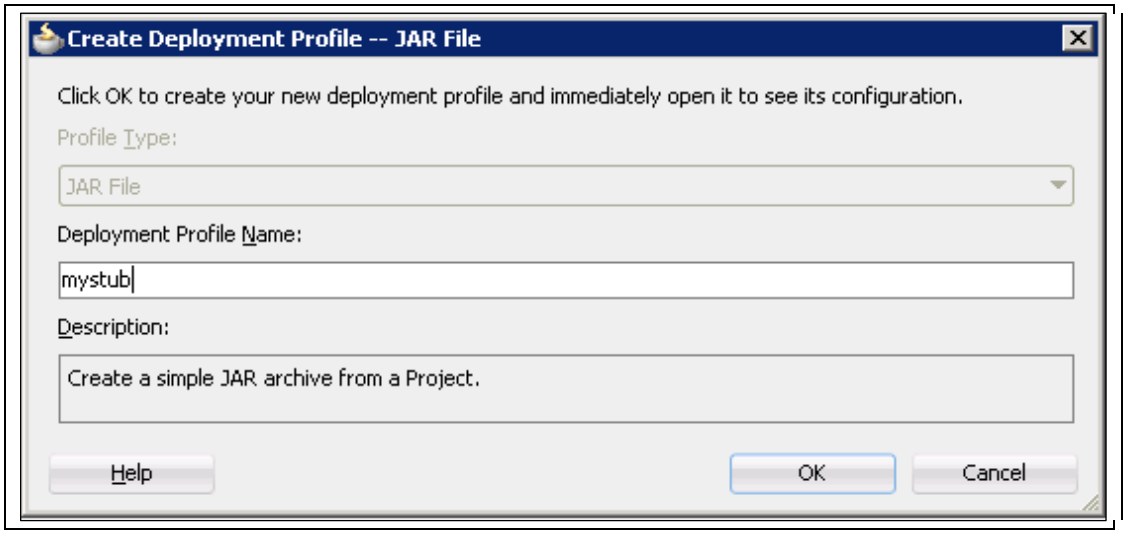
The address of the webservice endpoint (i.e. your host, port and application) is defined in the file *ReportService_Service.java.*

Now we have built and tested the Web Service Client from within JDeveloper. In order to call the Web Service from Forms, the Web Service Client must be deployed to the file system as a JAR file. (As an alternative to a JAR file which is a type of archive it is possible to use the generated files directly).

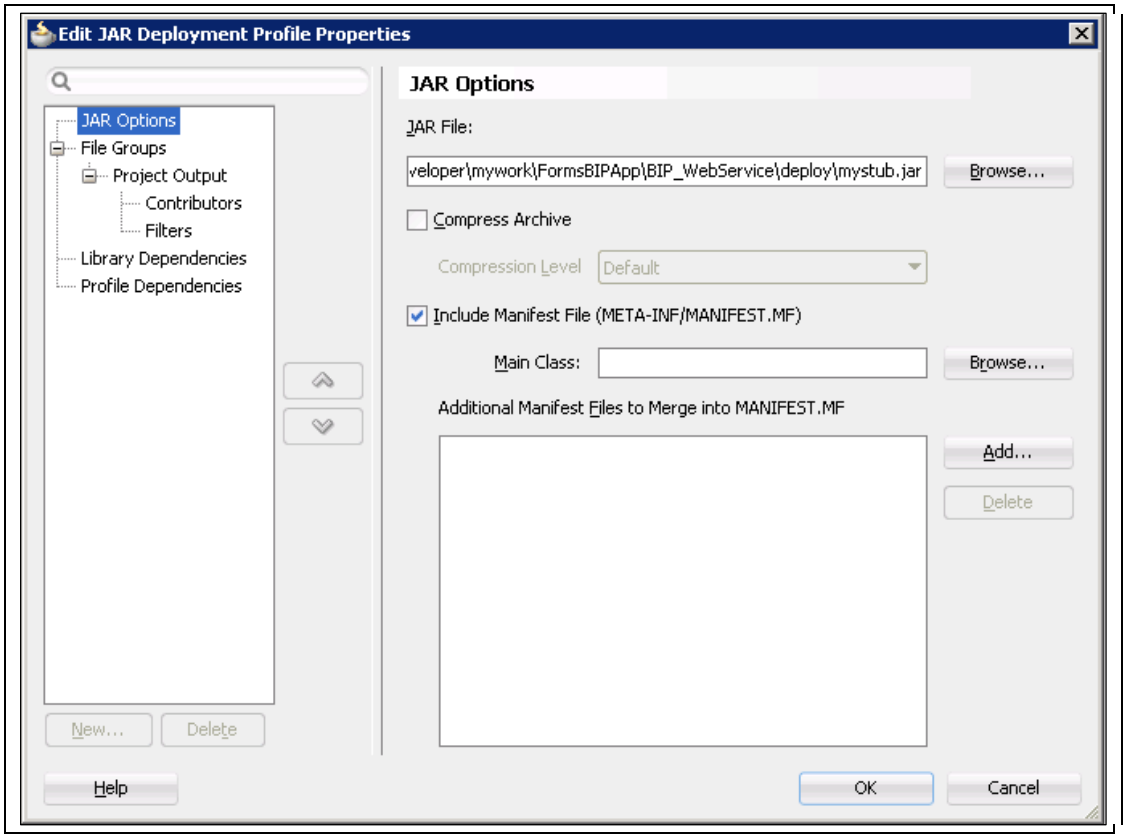
By right-clicking the project and selecting **New** from the context menu, the **New Gallery** will open. We need to select the **Category: General – Deployment Profiles** and then the **JAR File** item.



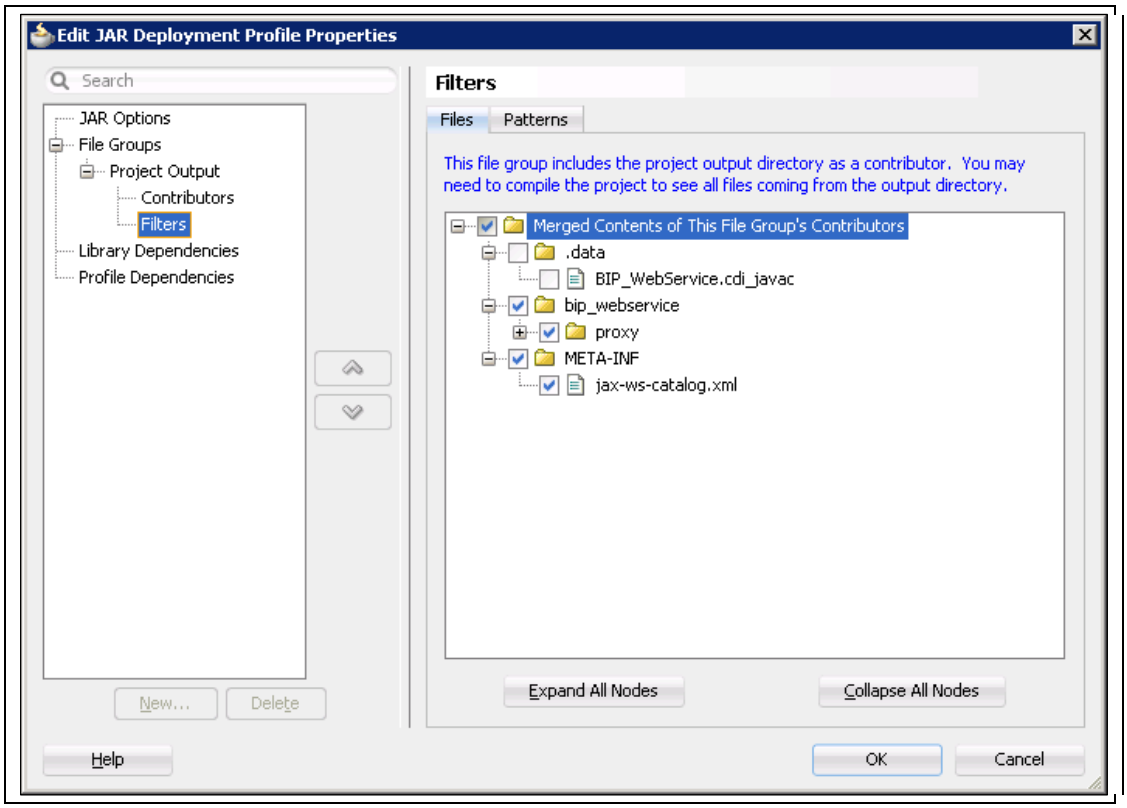
We can indicate the deployment profile name (here *mystub*).



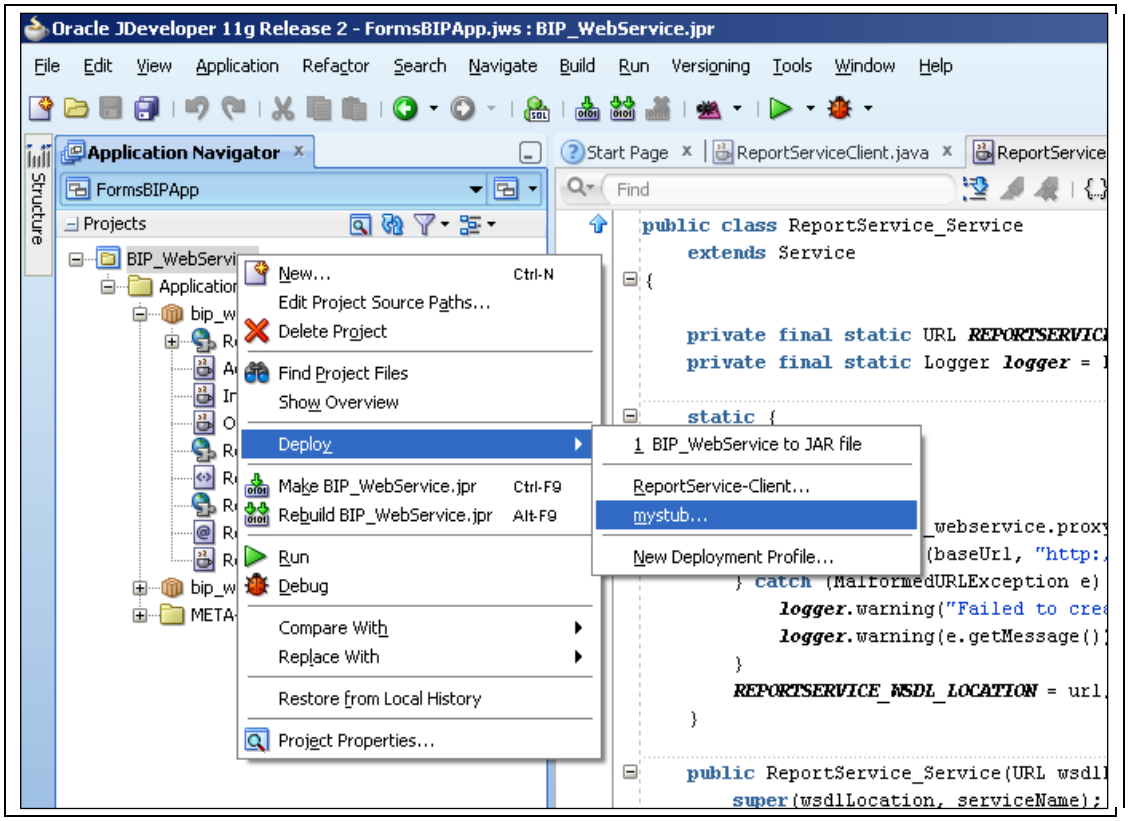
Then we will accept the JAR Deployment Profile properties.



By selecting the **Filters** option we can verify that all the necessary files are selected.



Right-click the archive file under the **Resources** node and select **Deploy to JAR file**. This will deploy the JAR file to the file system.


Oracle JDeveloper 11g Release 2 - FormsBIPApp.jws : BIP_WebService.jpr
File Edit View Application Refactor Search Navigate Build Run Versioning Tools Window Help
Application Navigator
FormsBIPApp
Projects
Start Page
ReportServiceClient.java
Find
public class ReportService_Service
extends Service
{
private final static URL REPORTSERVICE
private final static Logger logger =
static {
New... Ctrl-N
Edit Project Source Paths...
Delete Project
Find Project Files
Show Overview
Deploy
Make BIP_WebService.jpr Ctrl-F9
Rebuild BIP_WebService.jpr Alt-F9
Run
Debug
Compare With
Replace With
Restore from Local History
Project Properties...
1 BIP_WebService to JAR file
ReportService-Client...
mystub...
New Deployment Profile...
logger.warning("Failed to cre
logger.warning(e.getMessage()
REPORTSERVICE_WSDL_LOCATION = url
public ReportService_Service(URL wsdl
super(wsdlLocation, serviceName);

Creating the Oracle Forms application

Importing the Java client in Oracle Forms Builder

Oracle Forms needs to be able to see the relevant Java files in the Forms Builder during design time. This includes at least the generated jar-File (*mystub.jar*) with the Web Service Client. For this, all the Java classes that need to be imported in Forms must be visible in the Forms Builder Classpath. There are two possible ways to expose the necessary classes to Forms Builder:

1. By entering all the necessary jar-Files including their absolute path in the registry key FORMS_BUILDER_CLASSPATH for the Oracle Home of your Developer Suite installation.
2. By setting the environment variable FORMS_BUILDER_CLASSPATH in a script and calling the Forms Builder from this script.

Here is an example:

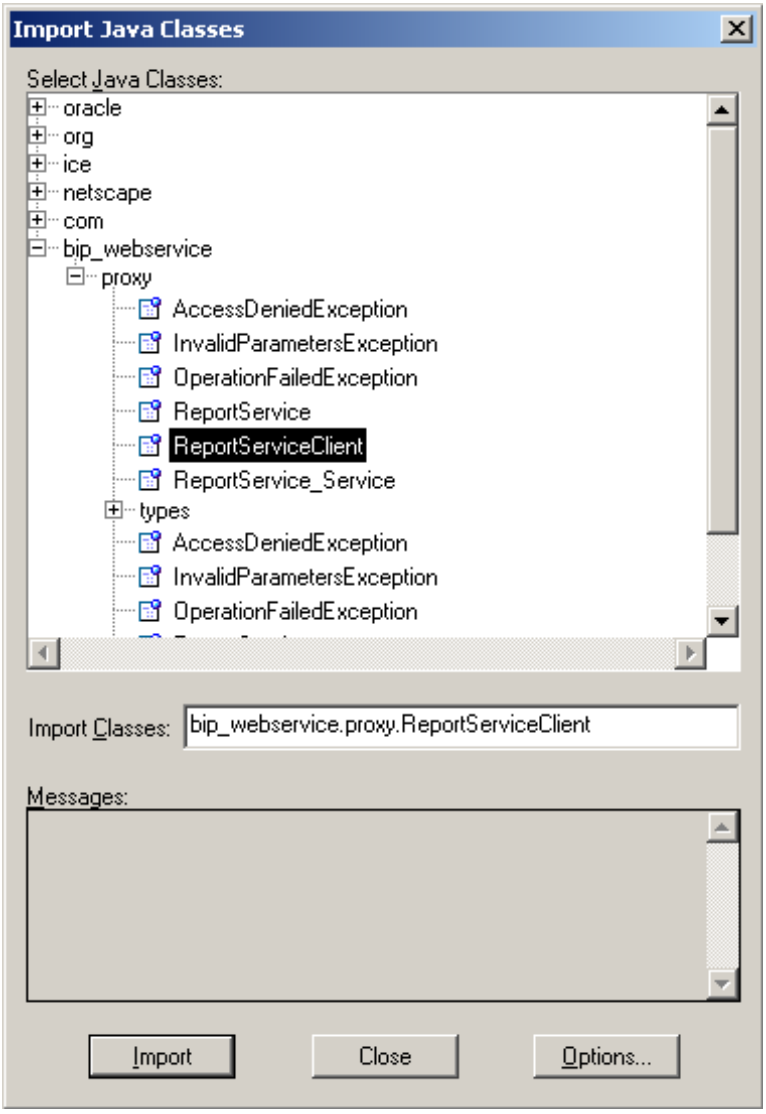
```
set FORMS_BUILDER_CLASSPATH =
   E:\Oracle\Middleware\as1\jlib\frmbld.jar;
   E:\Oracle\Middleware\as1\jlib\importer.jar;
   E:\Oracle\Middleware\as1\jlib\debugger.jar;
   E:\Oracle\Middleware\as1\jlib\utj.jar;
   E:\Oracle\Middleware\as1\jlib\ewt3.jar;
   E:\Oracle\Middleware\as1\jlib\share.jar;
   E:\Oracle\Middleware\as1\jlib\dfc.jar;
   E:\Oracle\Middleware\as1\jlib\ohj.jar;
   E:\Oracle\Middleware\as1\jlib\help-share.jar;
   E:\Oracle\Middleware\as1\jlib\oracle_ice.jar;
   E:\Oracle\Middleware\as1\jlib\jewt4.jar;
   E:\Oracle\Middleware\as1\forms\java\frmwebutil.jar;
   E:\Oracle\Middleware\as1\forms\java\frmall.jar;
   E:\Oracle\Middleware\as1\forms\java\mystub.jar

E:\Oracle\Middleware\as1\bin\frmbld.exe
```

Building the Forms application

In the Forms Builder we will create a new Forms module and use the Java Importer from the menu **Program - Import Java Classes** to import the following classes:

- *bip_webservice.proxy.ReportServiceClient*
- *bip_webservice.proxy.types.ArrayOfParamNameValue*
- *bip_webservice.proxy.types.ArrayOfString*
- *bip_webservice.proxy.types.ParamNameValue*
- *java.lang.String*
- *java.util.List*



If the classes are not accessible we have to modify the classpath and restart the Forms Builder.
If the classes are accessible, in case we get error messages during the import, there may be some referenced classes that are missing in the classpath. In this case, we need to add them and restart the Forms Builder.

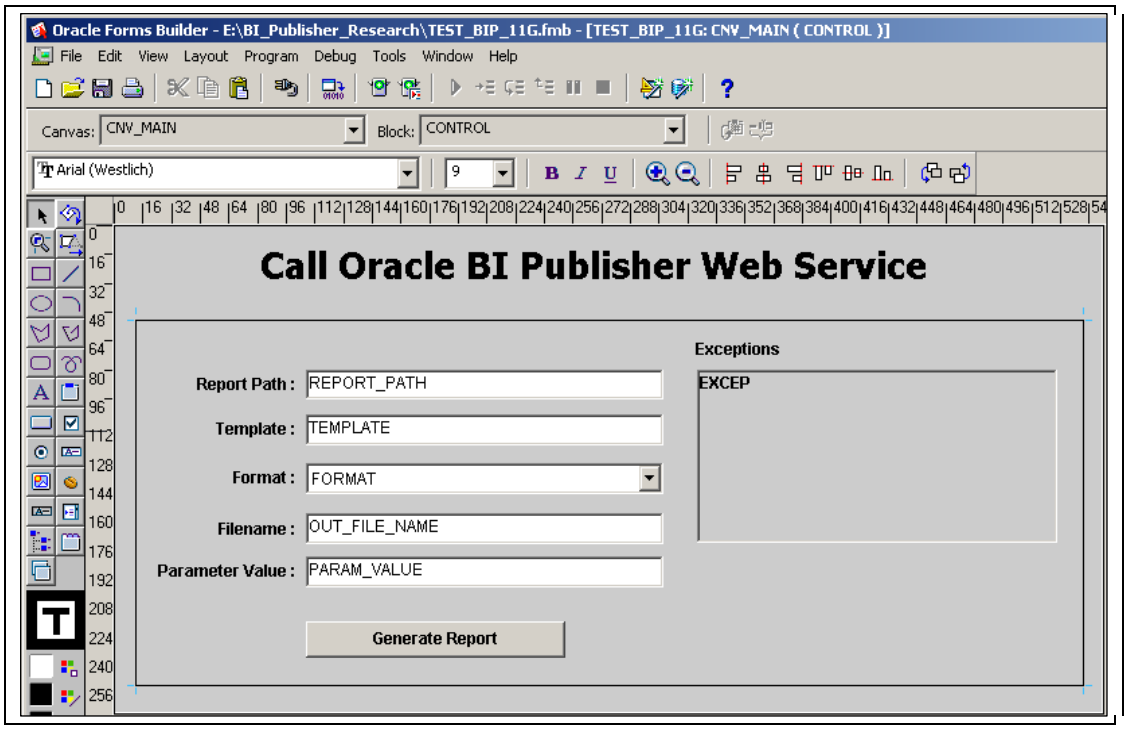
If the import succeeds we will get 6 new packages:

- *ArrayOfParamNameValue*
- *ArrayOfString*
- *List*
- *ParamNameValue*
- *ReportServiceClient*
- *String*

We need to rename the package *List* to *Java_List* and the package *String* to *Java_String*.

Now we can define the application logic in Forms and call the BI Publisher web service by using the *ReportServiceClient.callRunReport()* PL/SQL procedure.

Here is an example how the code may look like. The code assumes that there is a block named *CONTROL* with the items *REPORT_PATH, TEMPLATE, FORMAT, OUT_FILE_NAME, PARAM_VALUE*, a *GENERATE_REPORT* button and an optional item *EXCEP* in case we need to implement exception handling.



Here is the program code of the WHEN-BUTTON-PRESSED trigger:

Note: Adjust username, passwords, directory names accordingly to your environment.

```
Declare
lBIUsername             Varchar2(200);
lBIPassword             Varchar2(200);
lOutputFile             Varchar2(200);
lOutputFormat           Varchar2(10);
lFileExtension          Varchar2(10);
```

```
lParameterName          Varchar2(200);
lParameterValue         Varchar2(200);
lReportPath             Varchar2(200);
lReportTemplate         Varchar2(200);
lObj                    Ora_Java.JObject;
lArrayOfParamNameValue  Ora_Java.JObject;
lParamNameValue         Ora_Java.JObject;
lArrayOfString          Ora_Java.JObject;
lDone                   Boolean;
Begin

  lObj := ReportServiceClient.New();

  -- Login information for BI Publisher server
  lBIUsername := 'weblogic';
  lBIPassword := 'Welcome1';

  -- Set the report path and report template
  lReportPath     := :control.report_path;
  lReportTemplate := :control.template;

  -- Set the format and the extension for the output file
  If :format ='excel' Then
    lFileExtension := 'xls';
  Else
    lFileExtension := :control.format;
  End If;
  lOutputFormat := :control.format;

  -- Set the download directory
  lOutputFile := 'E:\Oracle\Middleware\as1\forms\java\' ||
                 :control.out_file_name || '.' || lFileExtension;

  -- Pass the user parameter
  lParameterName  := 'P_DEPT';
  lParameterValue := :control.param_value;
  lArrayOfParamNameValue := ArrayOfParamNameValue.New();
  lParamNameValue := ParamNameValue.New();
  ParamNameValue.SetName(lParamNameValue, lParameterName);
  lArrayOfString := ArrayOfString.New();
  lDone := Java_List.Add(ArrayOfString.GetItem(lArrayOfString),
                         Java_String.New(lParameterValue));
  ParamNameValue.SetValues(lParamNameValue, lArrayOfString);
  lDone :=
Java_List.Add(arrayOfParamNameValue.GetItem(lArrayOfParamNameValue),
              lParamNameValue);

  -- Call callRunReport
  ReportServiceClient.CallRunReport(lObj                ,
                                    lReportPath         ,
                                    lArrayOfParamNameValue,
                                    lBIUsername         ,
                                    lBIPassword         ,
                                    lOutputFormat       ,
                                    lReportTemplate     ,
                                    lOutputFile         );
  --Show Report in browser--
  Web.Show_Document( 'http://vmpitssbi:8090/forms/java/' ||
                :control.out_file_name || '.' || lFileExtension, '_blank');
End;
```

The report will be created as a file on the application server. Afterwards this file can be displayed with the built-in *web.show_document()*. For a multi-user environment we need to generate unique filenames.
The file in this simple example is generated into $ORACLE_HOME\forms\java folder (*E:\Oracle\Middleware\as1\forms\java*) which is mapped to the *"/forms/java/"* virtual directory.

The sample code has no error handling (which would be a good idea to have).

Parameters

ReportPath:	the location of the report definition file relative to the BI Publisher repository (example: */HR Manager/Employee Salary Report/Employee Salary Report.xdo*).
ParameterName:	The name of the parameter passed to the BI Publisher web service.
ParameterValue:	The value of the parameter passed to the BI Publisher web service.
BIUsername / BIPassword:	Username and Password for authentication against the BI Publisher server Note: Instead of passing username and password to the BI Publisher server it is recommended to use the method *login()* or *impersonate()* to get a valid session cookie from BI Publisher server. With a valid session cookie the method *runReportInSession()* has to be used instead.
OutputFormat:	The desired format of the document. Valid values are *pdf*, *rtf*, *excel*, *xml* and *html*. If in the BI Publisher environment a specific format is not allowed, it cannot be generated via web services, too.
ReportTemplate:	The name of the template (Example: *Table and Chart Layout*) (we need to use the logical name of the template, not the physical one).
OutputFile:	Directory path and filename that needs to be generated.

Synchronous vs. asynchronous call

The example uses a synchronous call of the BI Publisher Web Service, in order to call the document via *web.show_document()* directly after the call of the method *callRunReport()*. For long running reports it would be better to call the reports asynchronous, which could be done via a multi-threated web service client.

Configuring the Oracle Forms Runtime

In the Forms runtime environment we need to specify all the used and referenced classes via the CLASSPATH variable of the *default.env* environment file. It should contain a list of all required jar-files, including their absolute path on the application server machine.
Here is an example of the CLASSPATH variable in the *default.env* file:

```
CLASSPATH = E:\Oracle\Middleware\as1\forms\j2ee\frmsrv.jar;
            E:\Oracle\Middleware\as1\jlib\ldapjclnt11.jar;
            E:\Oracle\Middleware\as1\jlib\debugger.jar;
            E:\Oracle\Middleware\as1\jlib\ewt3.jar;
            E:\Oracle\Middleware\as1\jlib\share.jar;
            E:\Oracle\Middleware\as1\jlib\utj.jar;
            E:\Oracle\Middleware\as1\jlib\zrclient.jar;
            E:\Oracle\Middleware\as1\reports\jlib\rwrun.jar;
            E:\Oracle\Middleware\as1\forms\java\frmwebutil.jar;
            E:\Oracle\Middleware\as1/jlib/start_dejvm.jar;
            E:\Oracle\Middleware\as1\opmn\lib\optic.jar;
```

It is recommended to have a separate configuration in the *formsweb.cfg* and to define here a specific environment file for this application.

```
[bip_webservice]
envFile=bip_webservice.env
...
```

In the newly created *bip_webservice.env* file we need to append to the CLASSPATH variable the path to the *mystub.jar* as in the following example:

```
CLASSPATH = E:\Oracle\Middleware\as1\forms\j2ee\frmsrv.jar;
            E:\Oracle\Middleware\as1\jlib\ldapjclnt11.jar;
            E:\Oracle\Middleware\as1\jlib\debugger.jar;
            E:\Oracle\Middleware\as1\jlib\ewt3.jar;
            E:\Oracle\Middleware\as1\jlib\share.jar;
            E:\Oracle\Middleware\as1\jlib\utj.jar;
            E:\Oracle\Middleware\as1\jlib\zrclient.jar;
            E:\Oracle\Middleware\as1\reports\jlib\rwrun.jar;
            E:\Oracle\Middleware\as1\forms\java\frmwebutil.jar;
            E:\Oracle\Middleware\as1/jlib/start_dejvm.jar;
            E:\Oracle\Middleware\as1\opmn\lib\optic.jar;
            E:\Oracle\Middleware\as1\forms\java\mystub.jar
```

In our case *E:\Oracle\Middleware\as1* represents the ORACLE_HOME environment variable.

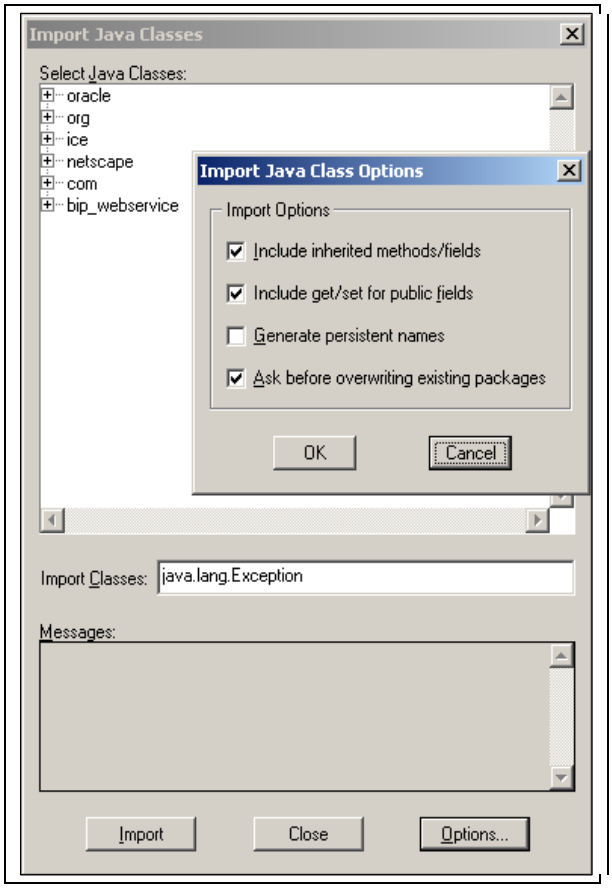
Now we may test the form by running the named configuration and calling the BI Publisher report from the application:
http://<host>:<port>/forms/frmservlet?config=bip_webservice& ...

Debugging

When running the Form for the first time we may get an unhandled exception ORA-105100 or ORA-105101. This indicates that Java classes are still missing in the runtime environment or that a Java error occurred during the execution of the code.
It would be helpful to have some more information about what happened by using some kind of exception handling. There is a way to display the Java error stack in the Form which could be of great help finding the cause of the error.

For this, we need to import the *java.lang.Exception* and *java.lang.StackTraceElement* classes and to activate the following options in the Forms-Java Importer:

- Include inherited methods/fields
- Include get/set for public Fields



In the code example below a nested exception catches the errors during the execution of the *Exception_.toString()*.
We can add this code to our WHEN-BUTTON-PRESSED Trigger after importing the necessary classes.

```
Declare
..........
raisedException   Ora_Java.JObject;
stack_trace       Ora_Java.JObject;
stackTrcElement   Ora_Java.JObject;
Begin
..........

-- Exception handling for Java-errors
Exception
  -- Check for ORA-105100
  When Ora_Java.Java_Error Then
    Message('Unable to call out to Java, ' || Ora_Java.Last_Error);
    Return;
  -- Check for ORA-105101
  When Ora_Java.Exception_Thrown Then
    raisedException := Exception_.New(Ora_Java.Last_Exception);
    Begin
      :control.excep := 'Exception: ' ||
                        Exception_.ToString(raisedException);
    Exception
      When Ora_Java.Java_Error Then
        Message('Unable to call out to Java, ' || Ora_Java.Last_Error);
        Return;
    End;
    -- Get an array of StackTraceElement from the Exception
    stack_trace := Exception_.GetStackTrace(raisedException);
    -- Loop over all the Elements
    For i In 0 .. Ora_Java.Get_Array_Length(stack_trace) Loop
      -- Get each Element
      stackTrcElement := Ora_Java.Get_Object_Array_Element(stack_trace,
i);
      --Make a string out of it and add it to the error field
      :control.excep := :control.excep || Chr(10) ||
                        stackTraceElement.ToString(stackTrcElement);
    End Loop;
    Ora_Java.Clear_Exception;
    Return;
  When Others Then
    Message('Problem!');
    Return;
End;
```

Outlook

The example could be extended in many ways:

- It is possible to pass also complex parameter structures with more than one parameter and more than one value per parameter (multiple selection)
- Another direction could be to separate the processing of the report from the Forms application by using an asynchronous call of BI Publisher Web Service.
- It is also possible to write the generated report as a byte stream into a CLOB column in the database which would give us the opportunity to use features from the database (security, stored procedures, AQ etc.).

In case of questions please feel free to contact us
Find our contacts in the lower right
We like to support you

Please also have a look at PITSS' services page

http://www.pitss.eu/services/

About PITSS

PITSS is the leading provider of software & services for modernizing and effectively managing Oracle applications. The PITSS Group was established in 1999 and has gained international recognition with over 1,000 customers and a multitude of successful Oracle projects. PITSS is an Oracle Gold partner and, as a member of the Oracle Modernization Alliance (OMA), is the only Oracle Forms Migration partner for automated migrations. With sites in Stuttgart, Munich (Germany) and Troy (USA) as well as certified international partners, the company successfully provides support for IT projects of medium sized companies, large enterprises and public contractors across the globe.

ORACLE Gold Partner

How to integrate Oracle BI Publisher via Web Services in Oracle Forms 11g

May 2013

Authors:
Axel Harsch, PITSS
Jürgen Menge, Oracle
Florin Serban, PITSS
Rainer Willems, Oracle

Contributors:
Mireille Duroussaud, Oracle

PITSS in Europe
Germany
+49-711-728.752.00
info@PITSS.com
www.PITSS.com

PITSS in Americas
USA
248.740.0935
info@PITSSamerica.com
www.PITSSamerica.com