**ADF Integration With Java Webservice**

**Introduction**

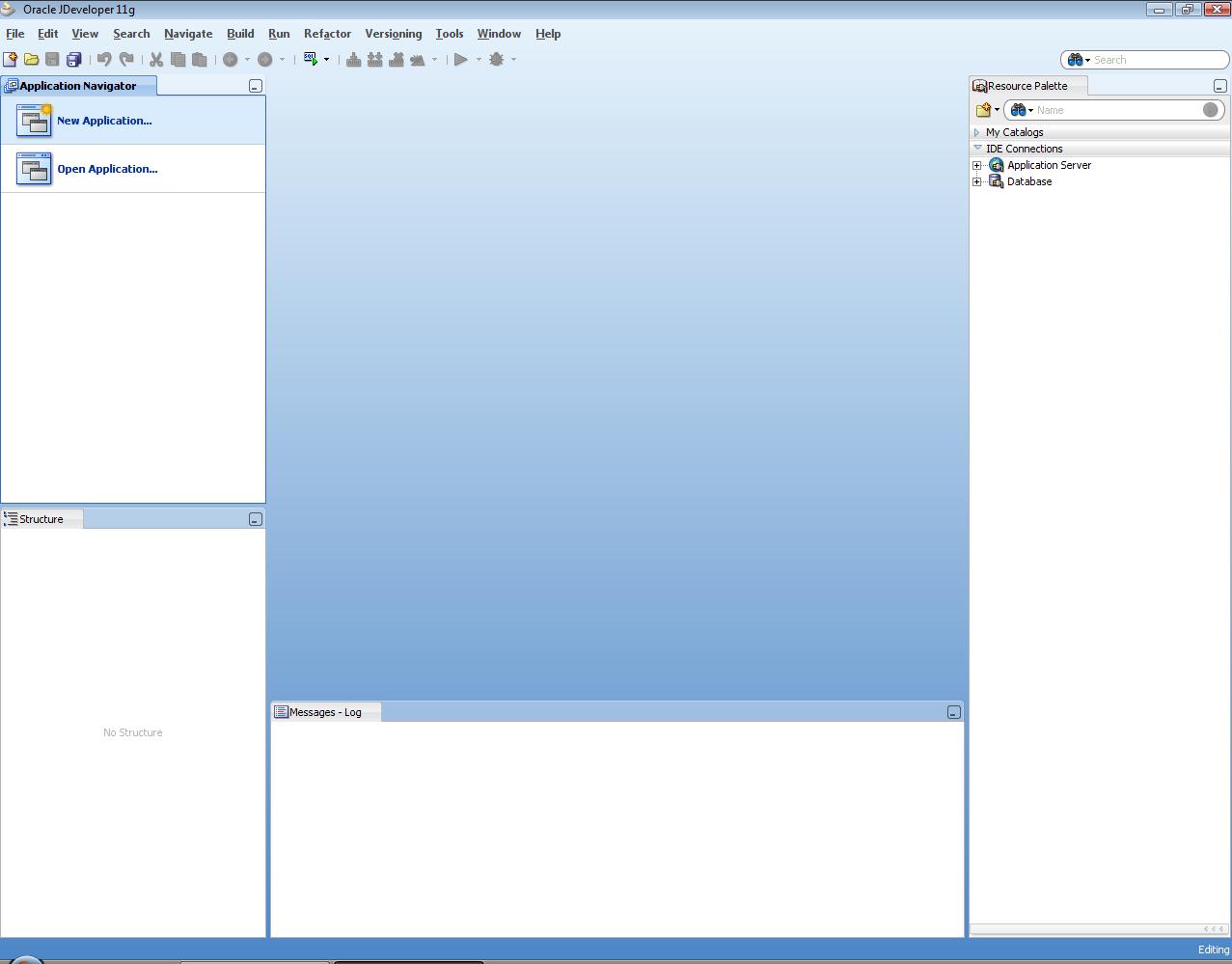
ADF is based on the J2EE design pattern Model-View-Controller (MVC). The Fusion Web application consists of two projects. One project for the data model (ADF Business Component), and another for the view and controller components (ADF Faces and ADF task flows). Here we are using view and controller components from ADF and integrate it with java web service using a java proxy client.

Bok is divided into 3 parts

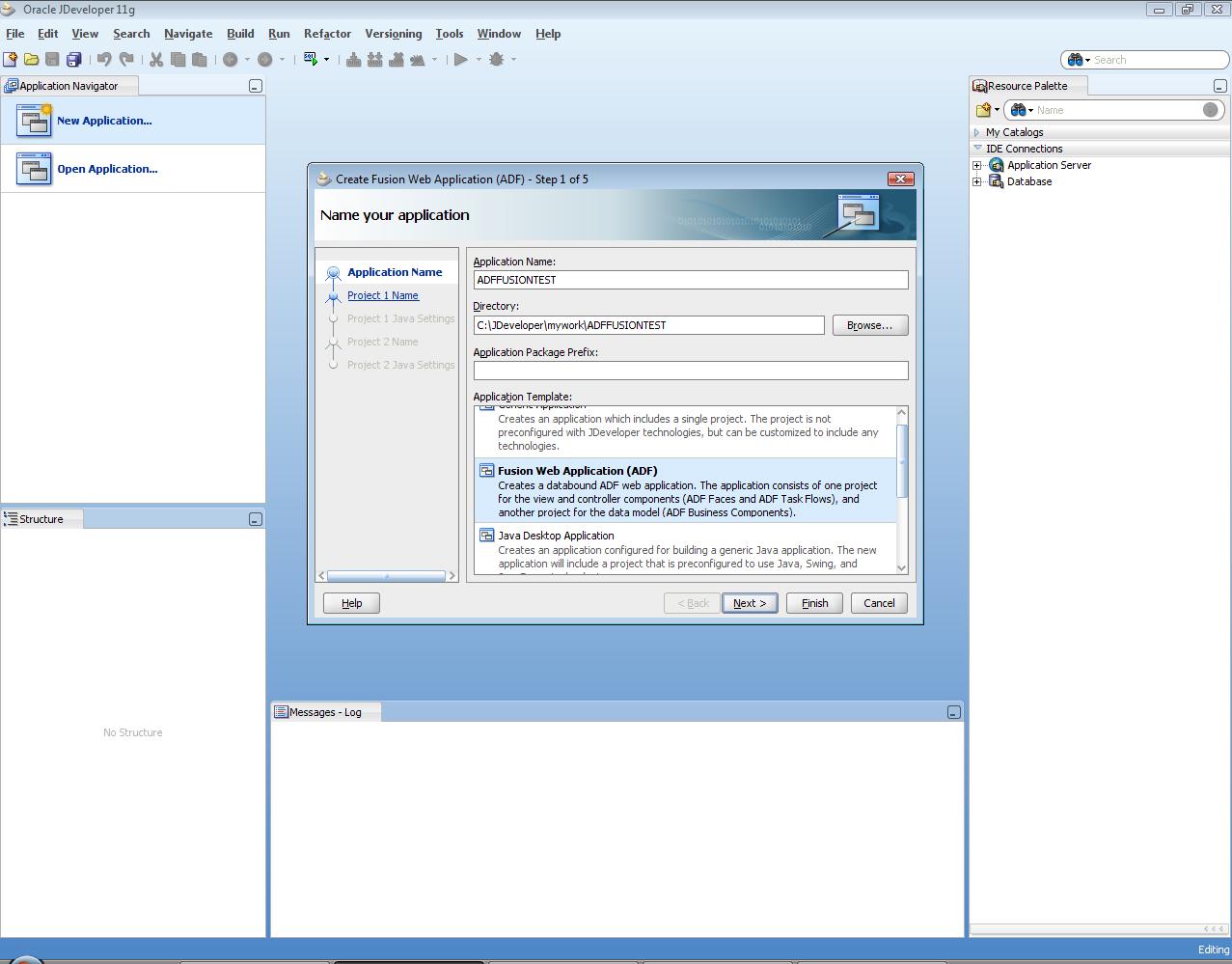
1. Steps required for creating Java Web Service.
2. Steps required for creating Java proxy client.
3. Steps required for creating ADF pages and integrate it with Web service.

**Creating new application and Project**

1. In order to create a new application we need to open a create application wizard. To open the wizard, in the application navigator, choose **New Application**.

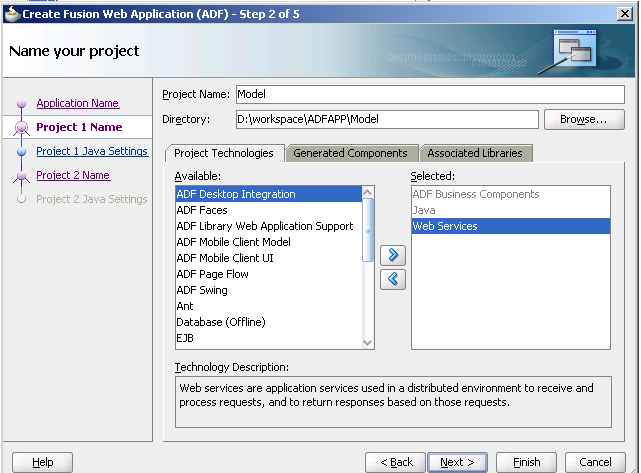


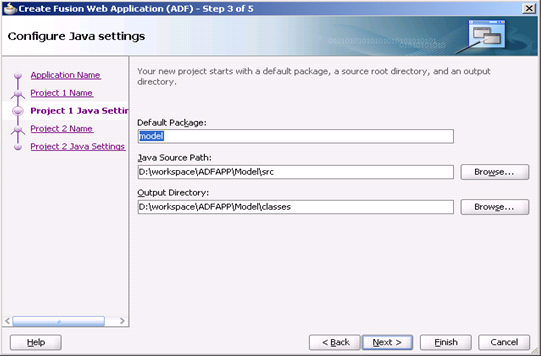
1. When you click on the **New Application** a wizard starts running and a dialog appears.

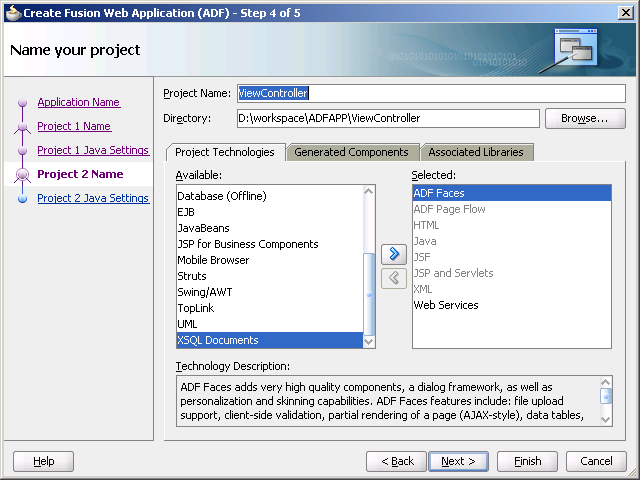


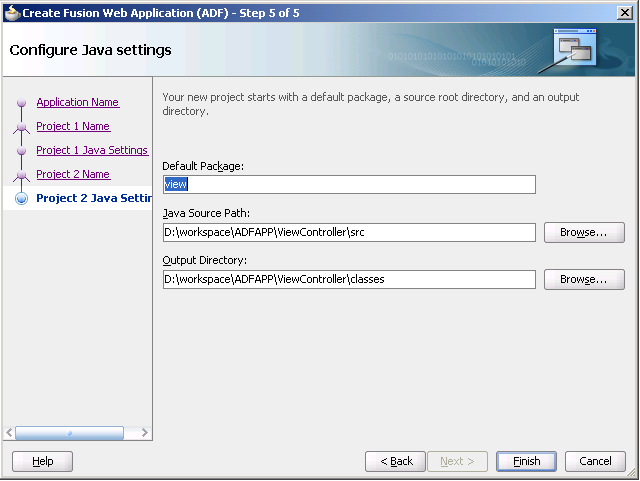
1. Choose technology. Enter the application name as *ADFAPP.* This will be the workspace and will be stored as .jws file

Select model technology: select ADF FACES for view and Web service for model technology. Specified the view and model project name.



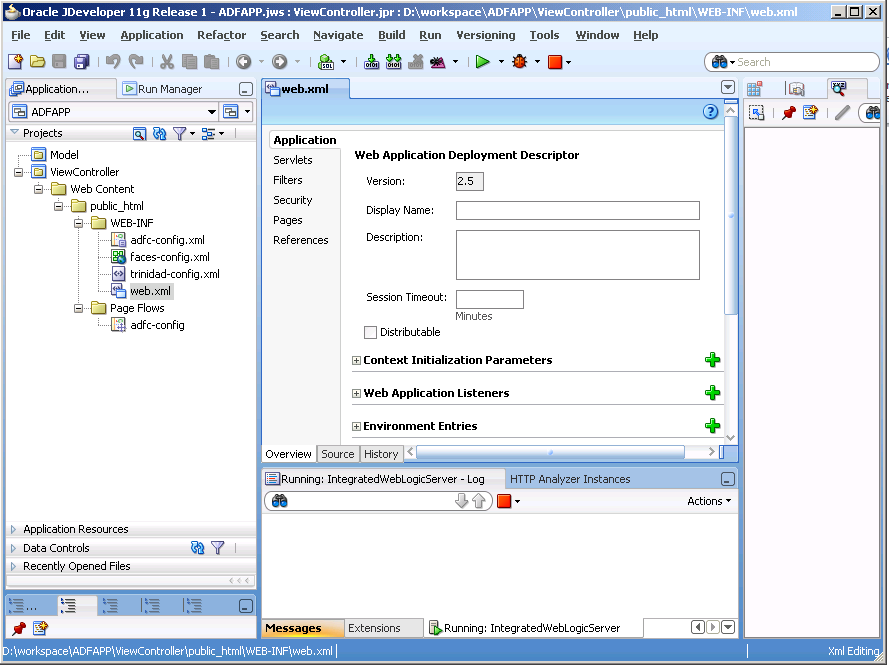






Click finish to create a project

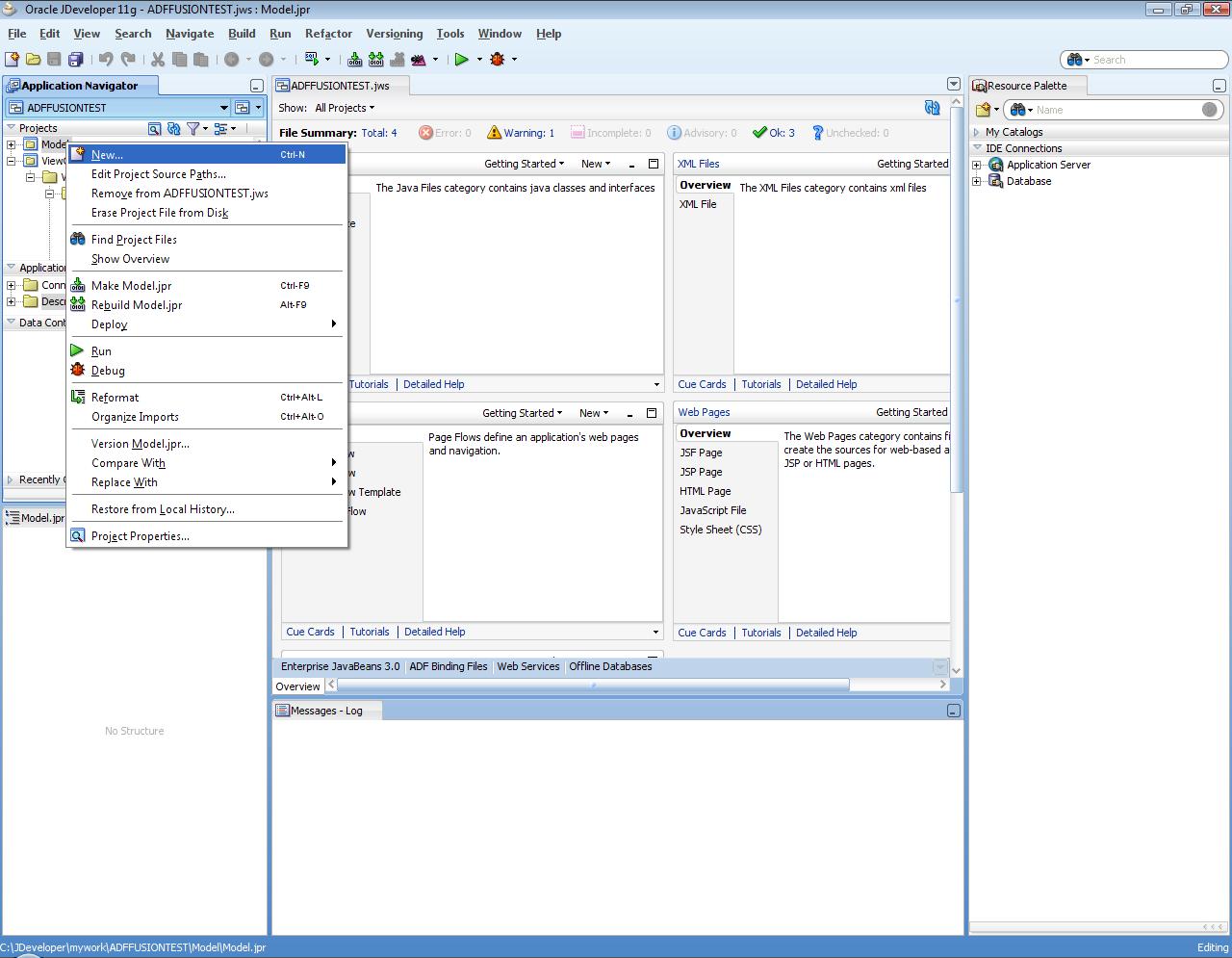
The application is created and the following screen appears having two folder ViewController for creating ADF webtier and Model for creating a WebService.



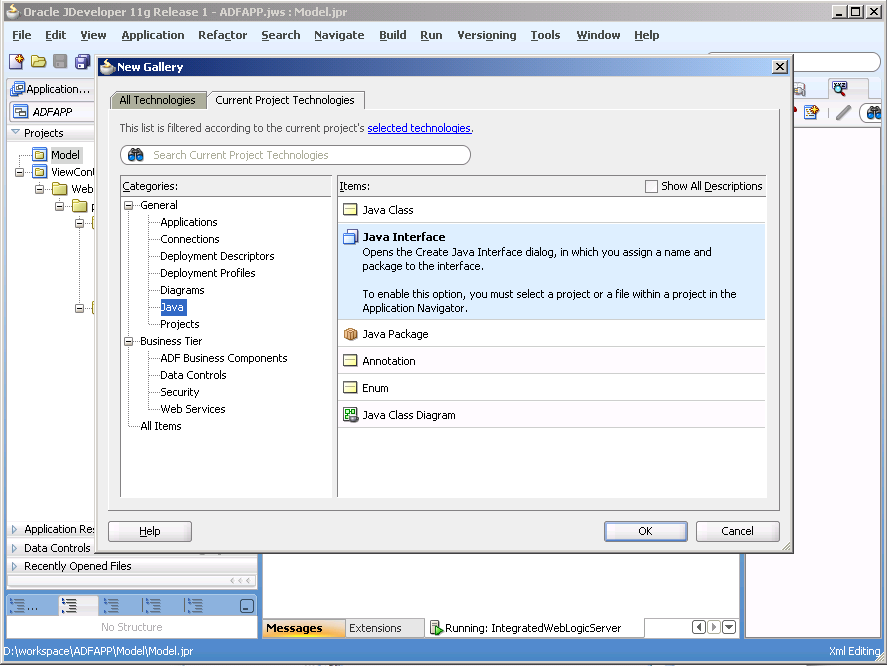
**Creating Java Web-Service Component**

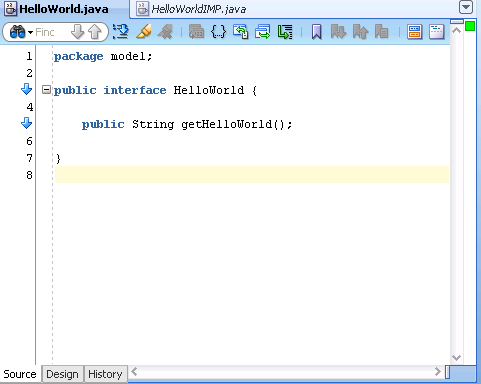
**Steps**

1)Since we know that Business component are created in Model. So, in the Application Navigator, Select Model, right click on it and choose New from the menu.

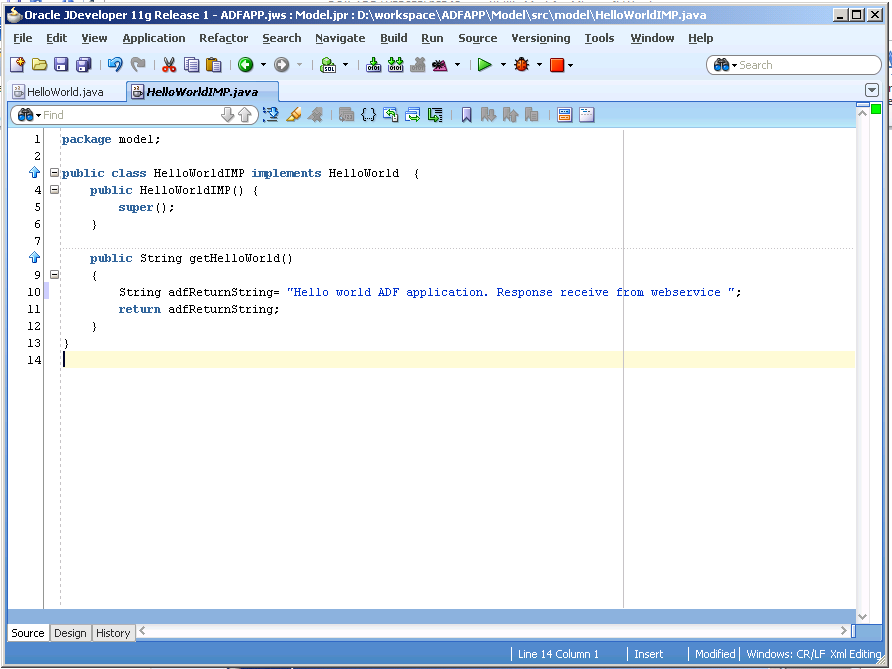


2) Select java interface which exposed to outside world. Create a HelloWorld interface class and define a getHelloWorld() method.



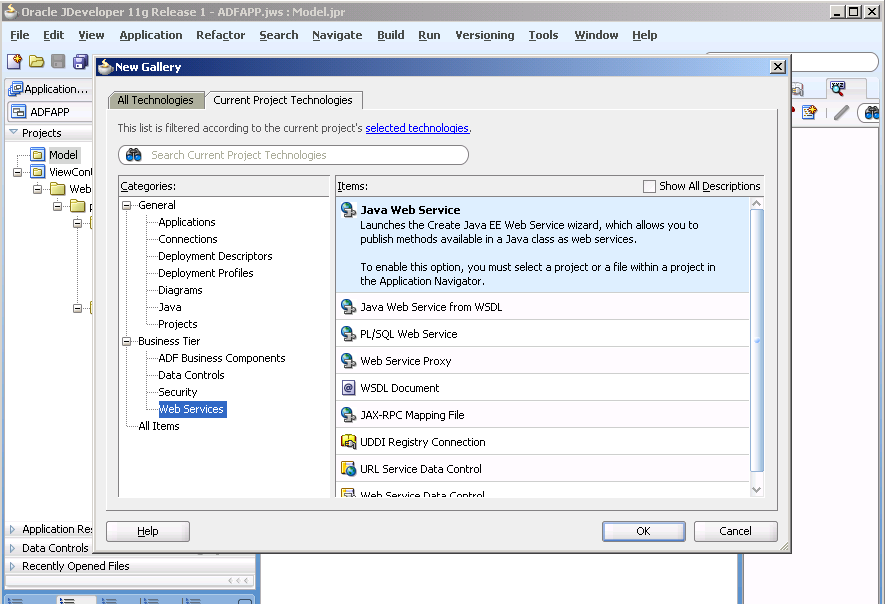


2) Create HelloWorld.java Implementation class having business logic.

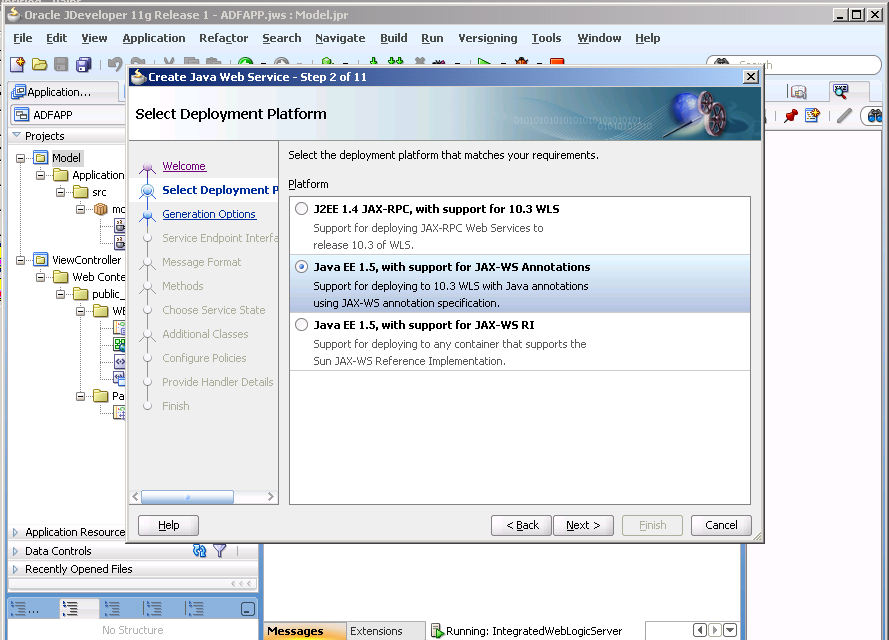


Follow are the following steps to create a java based webservice.

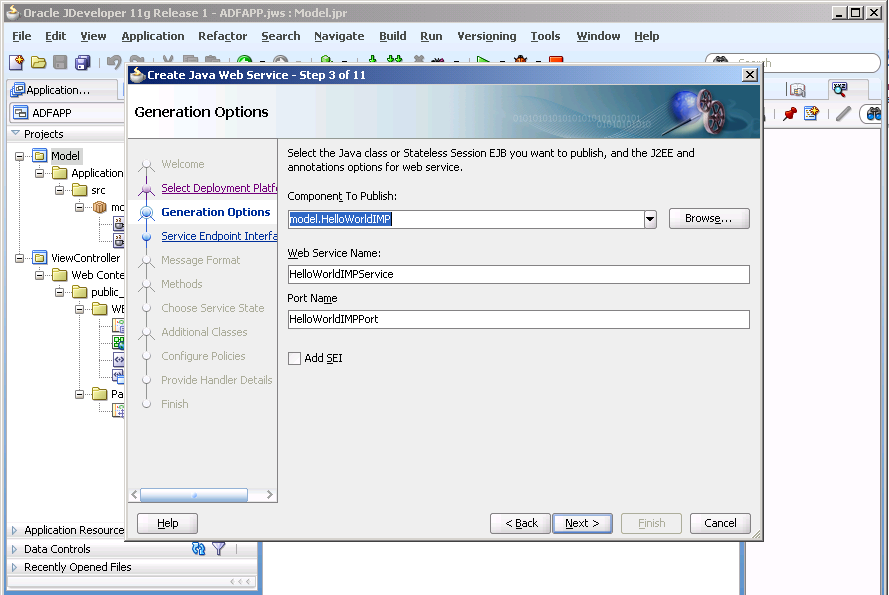
1. Choose Web Service as a Technology.



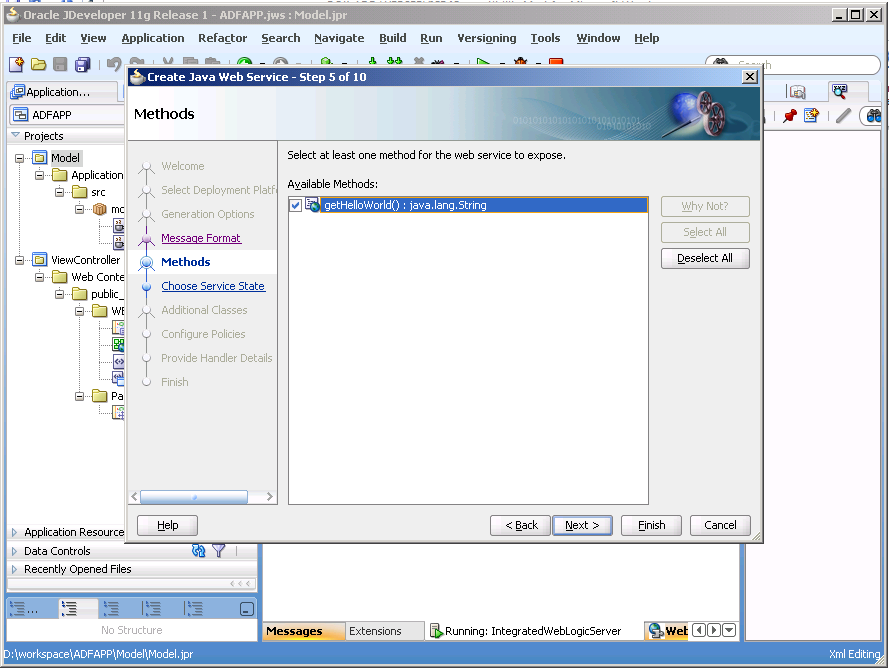
1. Choose the deployment platform



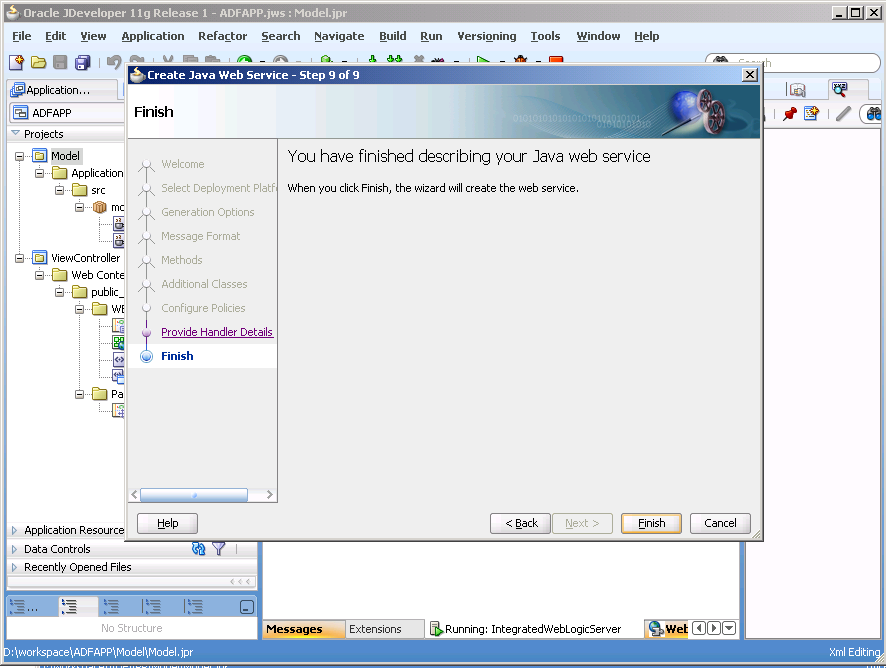
5) Select the java class that you want to expose as web service. Select HelloWorld Implementation class for publishing it as web service.



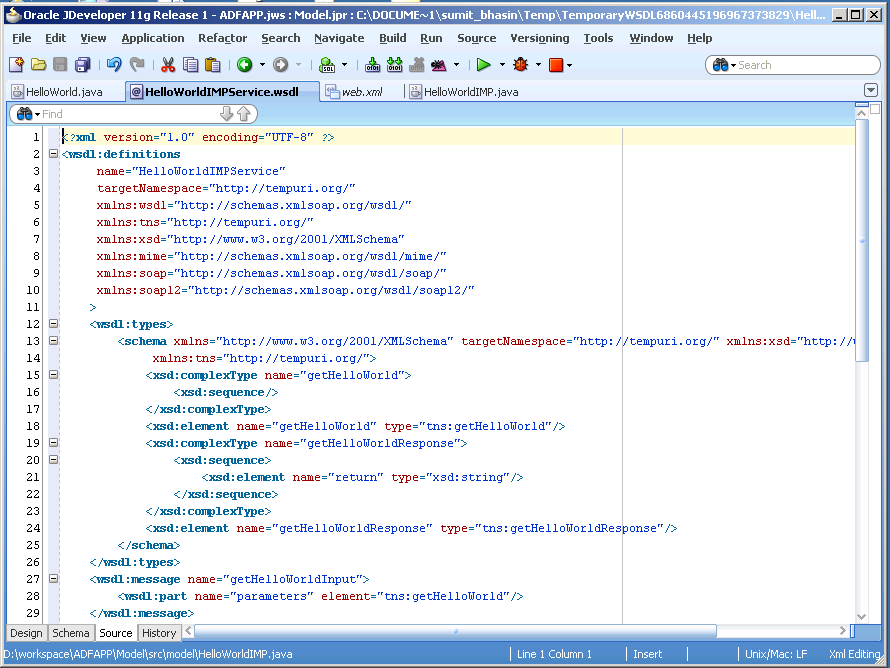
6) Select method that needs to be exposed as web service.



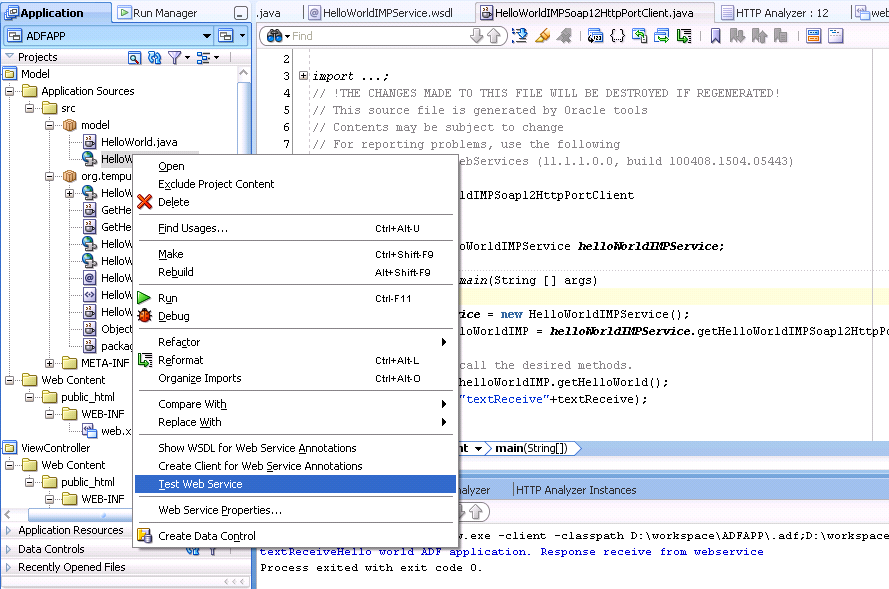
7) Click on finish and it will create a java based web service for the method exposed previously.



8) WSDL for exposed web service is created. Now next step is to launch the Web Service and create a proxy client

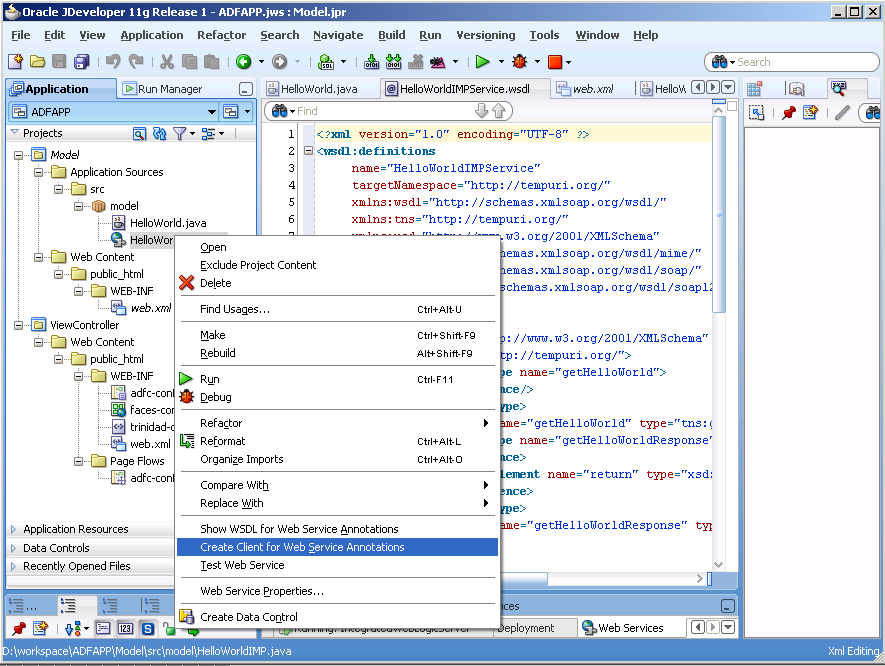


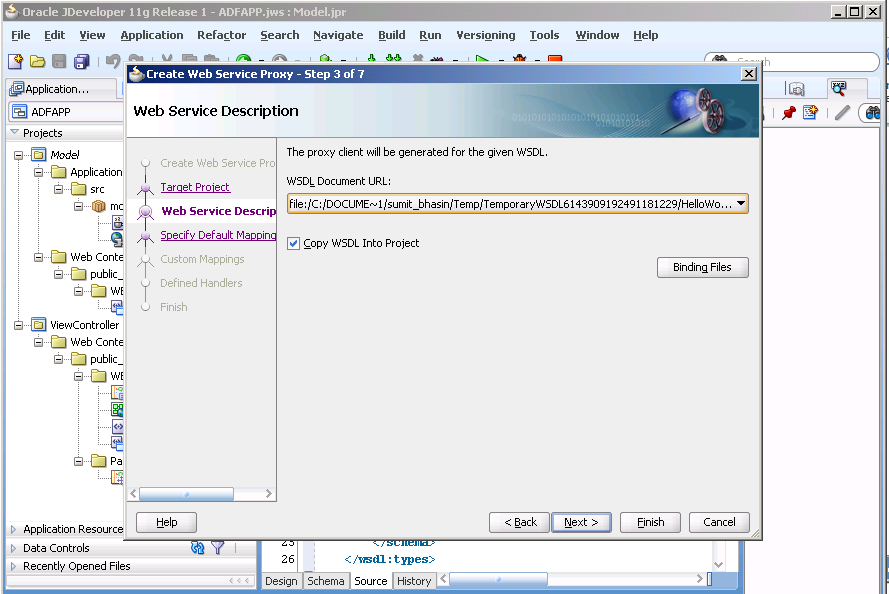
In-order to launch web service right clicks on Hello world Implementation class and chooses test web service option. It will deploy the service on to the application server.



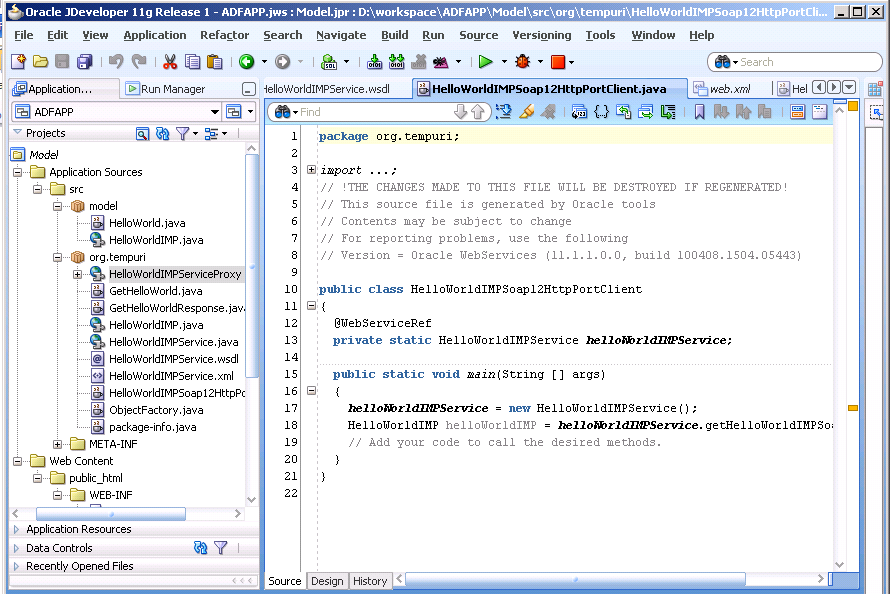
**Creating the java Proxy client for Web service**

Now next step is to create a proxy client. In order to create a client for the web service. Select the web service interface that is exposed and Choose create client for web service annotation option. It will show popup screen with set of steps need to follow.

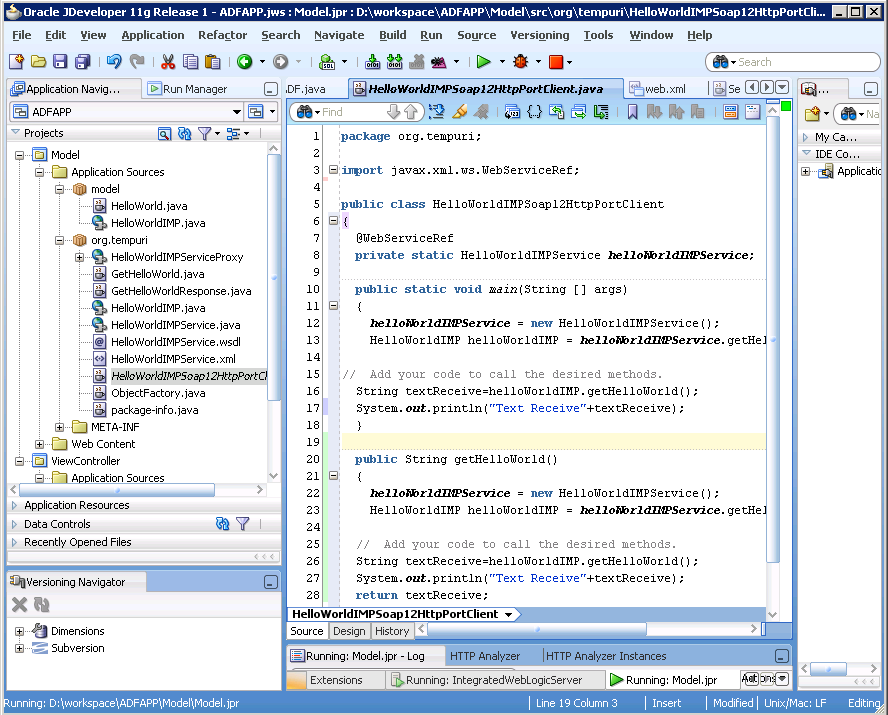




On click of finish. You will see in application navigation dialog set of class is been create along with java client class to access the web Service.



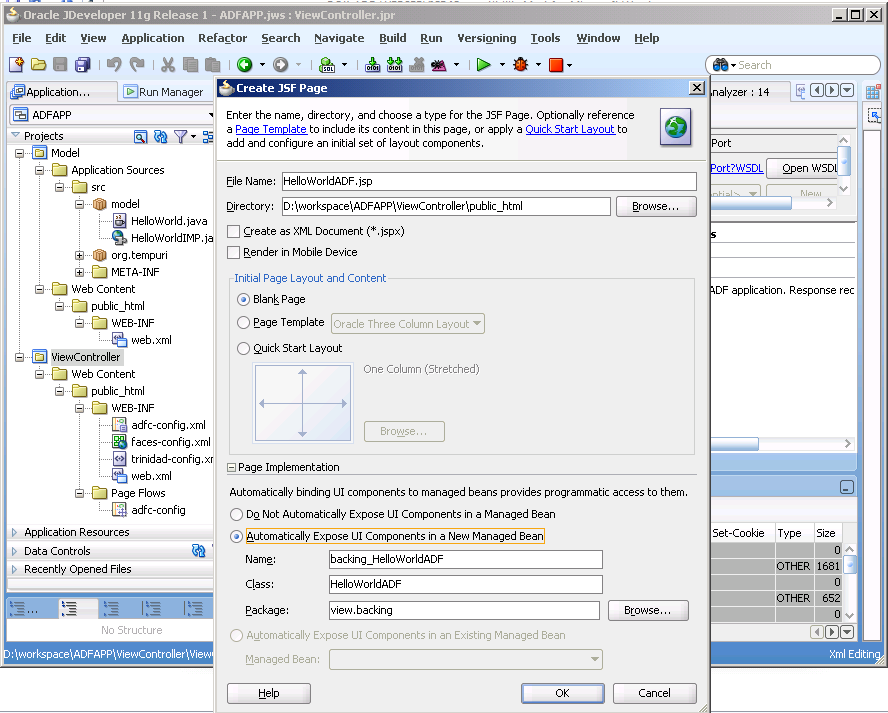
Add another method getHelloWorld() for interacting with webservice and which is called by manage bean.



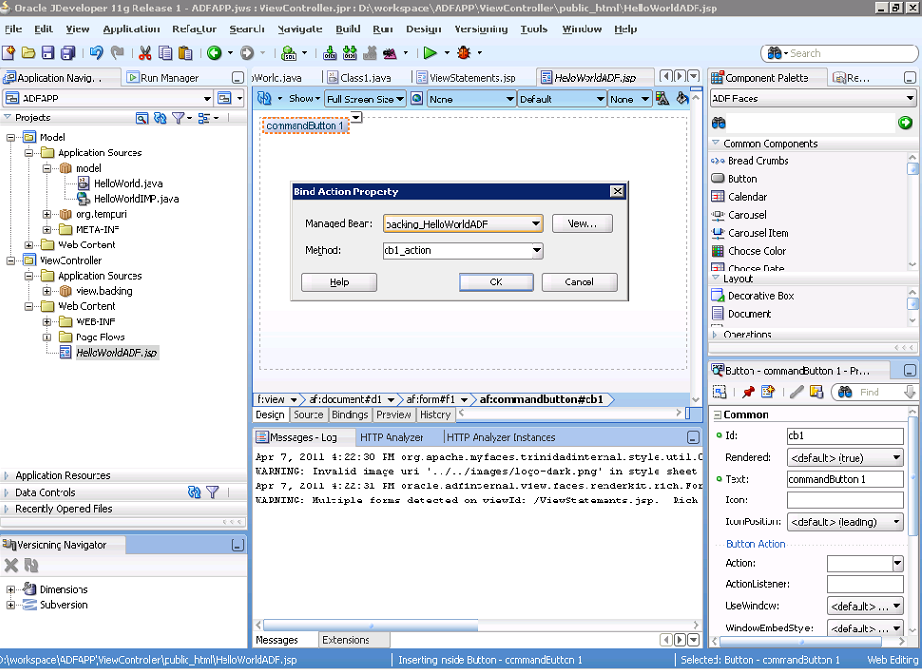
**Creating the ADF Web User Interface**

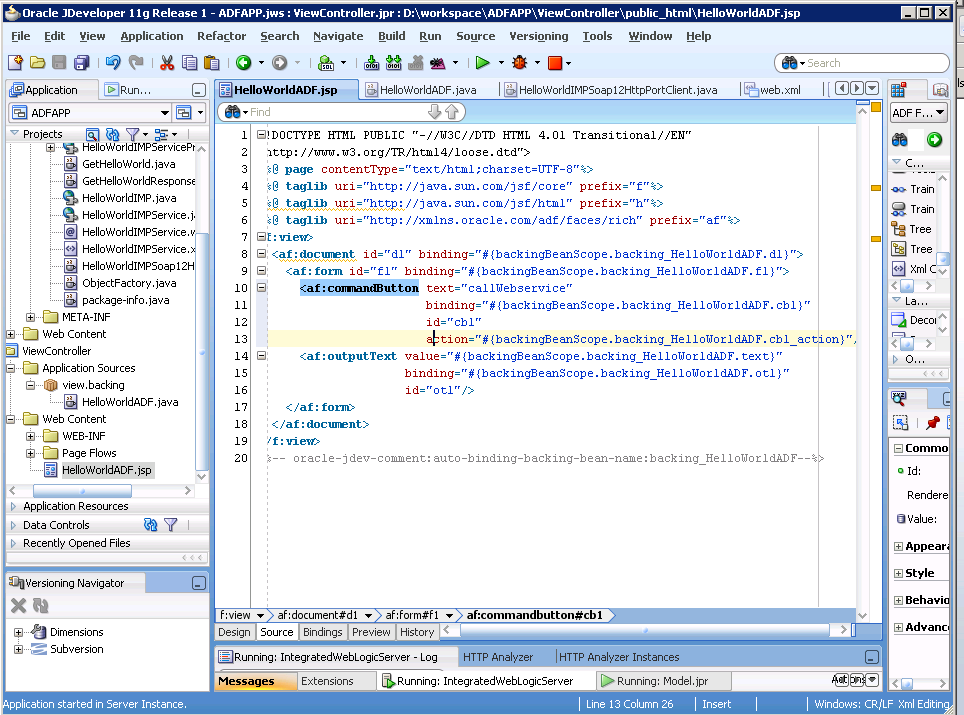
In Application Navigator, right click on **View Controller;** from the popup menu choose new. The New Gallery window appears, in this window create a JSF page HelloWorldADF.jsp and choose automatic expose UI component in a new managed bean and click on Ok button.

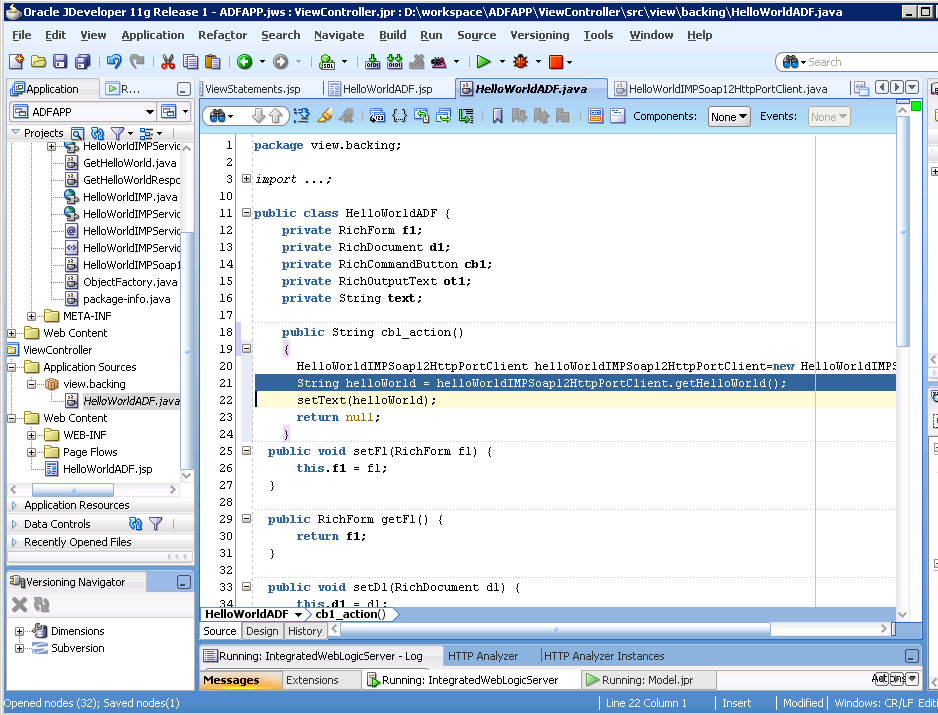
1. Create HelloWorldADF.jsp and select automatically expose UI components in a new managed bean option. It creates a backing bean for dynamic UI support.



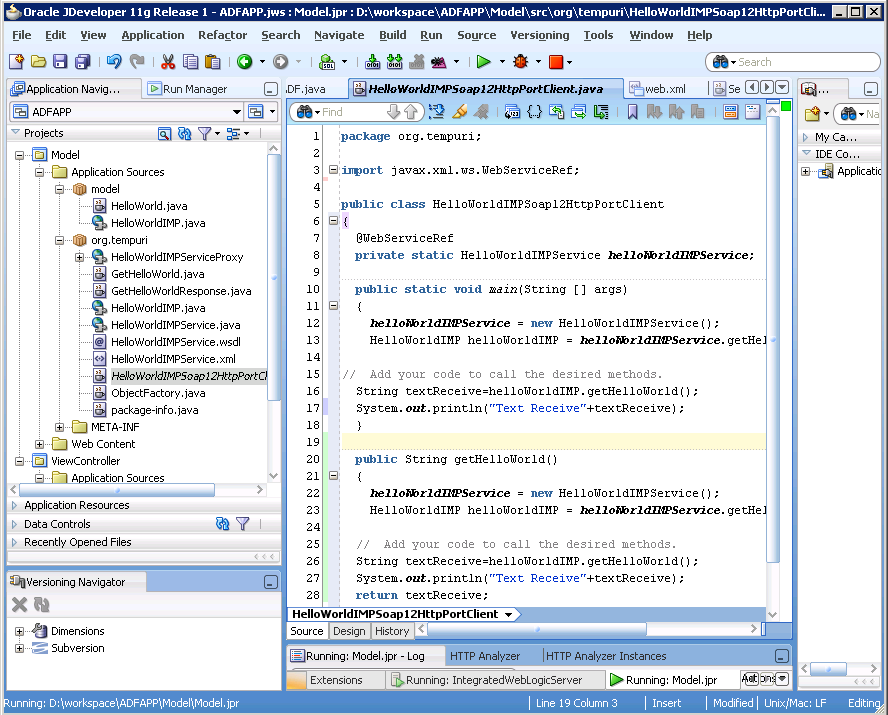
1. Now create a command button and the output text box on jsp page and associate it with method of manage bean.



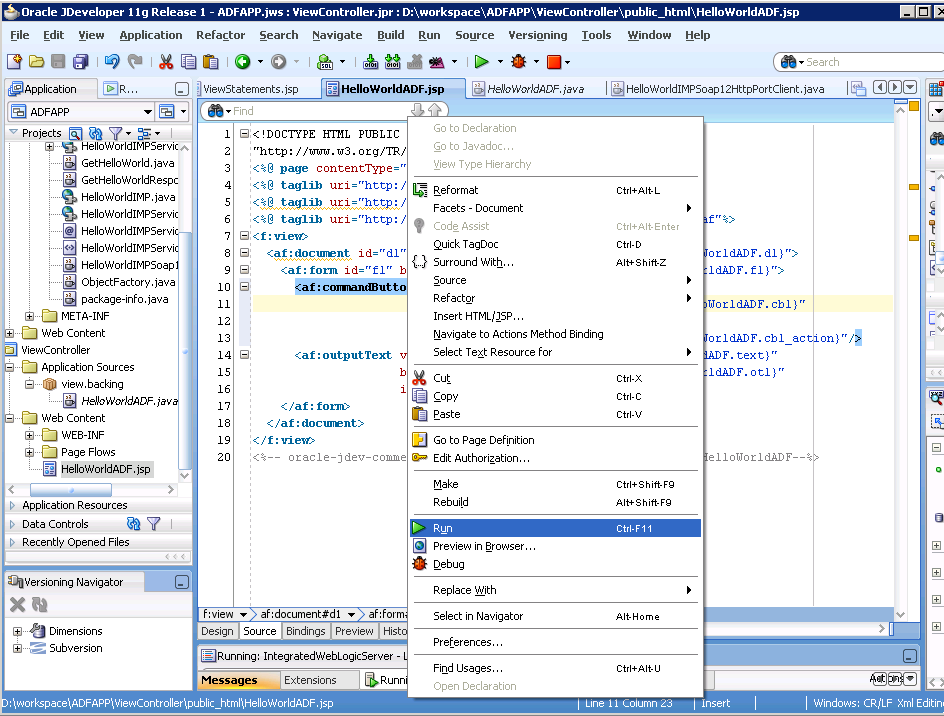




1. Associate the button with managed bean that interact with the web service using cb1\_action method which internally call the getHelloWorld() of proxy client class to get the response back from the web service and display that to the screen.

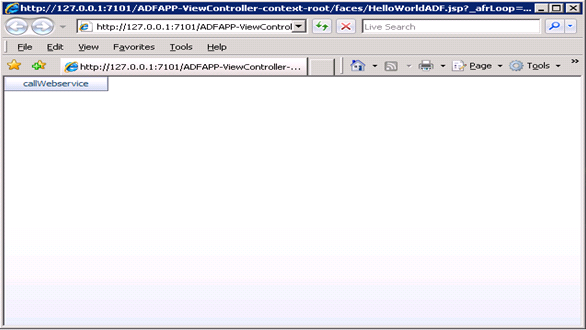


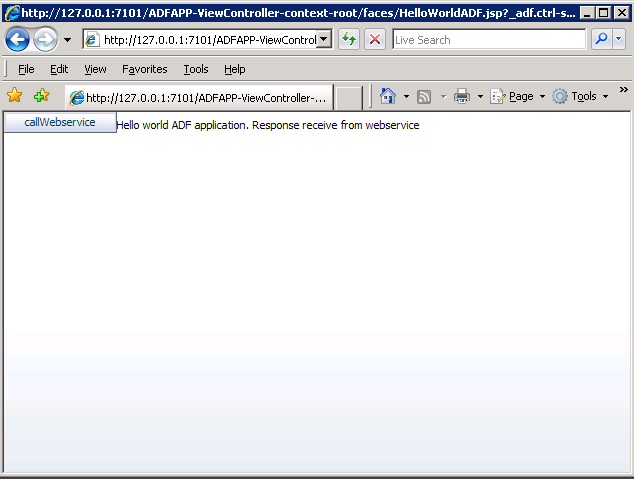
1. Run the jsp page



**Output Flow**

On click of “callWebservice” button control goes to managed bean which passes control to the proxy client and further proxy client interact with the exposed web service and return the response back to the client. Proxy client passes response back to managed bean and in manage bean text value is assign to the output box which display on screen as below.





**Reference(s)**

1. Oracle Jdeveloper 10g Handbook, Dr. Avrom Roy-Faderman, Peter Koletzke, Dr. Paul Dorsey, 2004

2. Jdeveloper 11g HELP menu