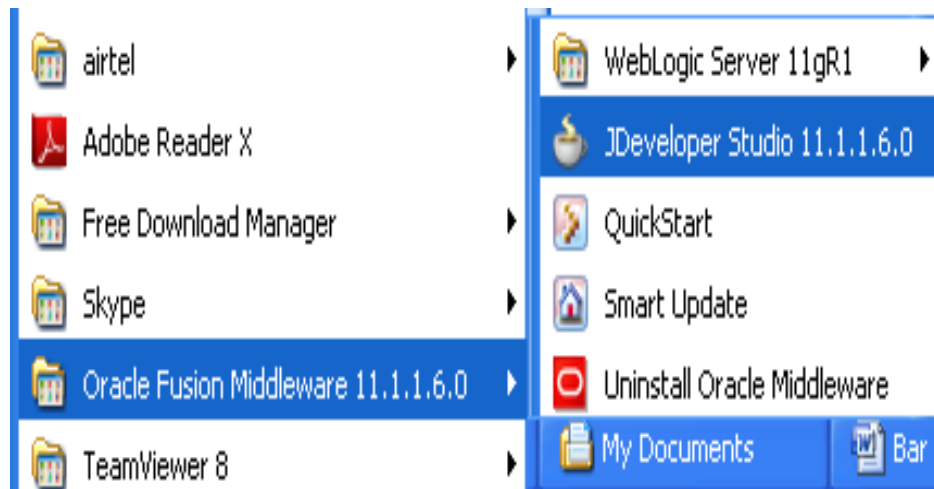


# Generate Weather ForeCast Report Using Third party wsdl in ADF

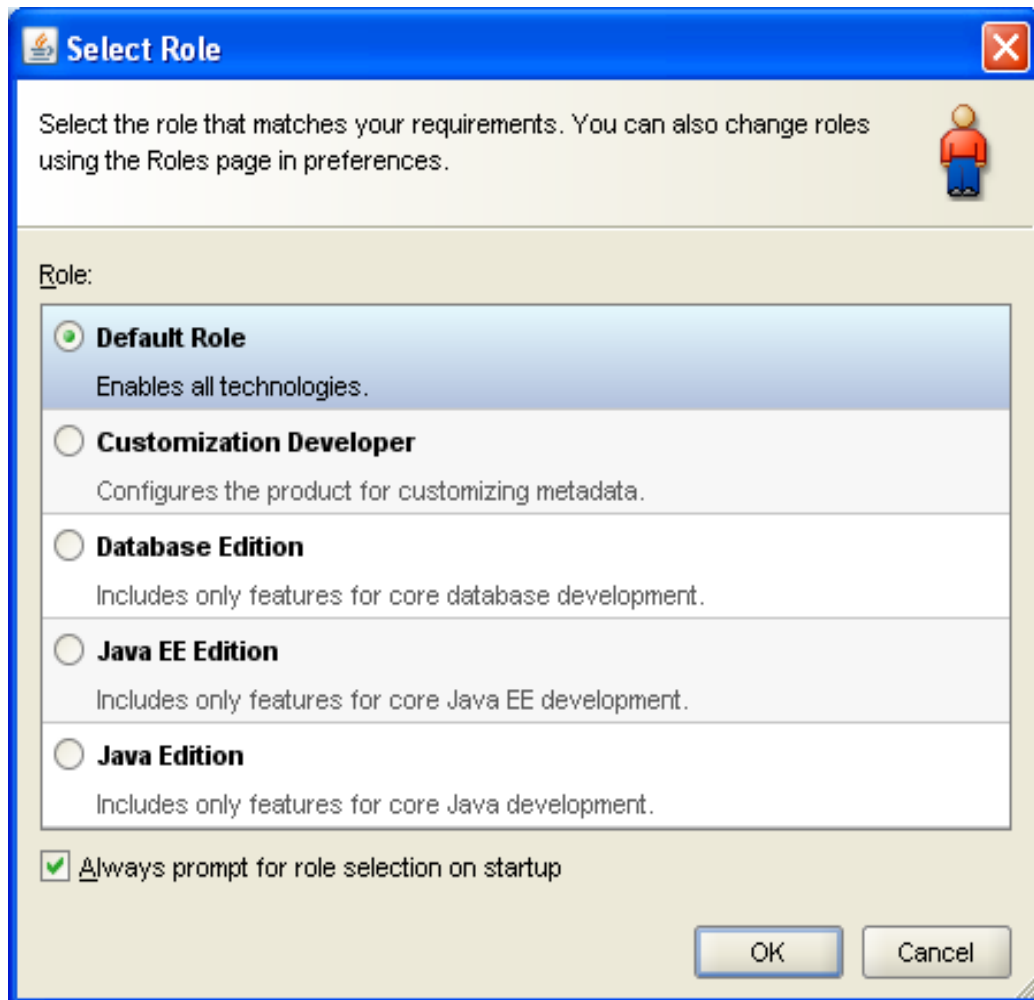
In this tutorial we are going to see how to generate weather report using third party wsdl

WsdI Path: <http://wsf.cdyne.com/WeatherWS/Weather.asmx?wsdl>

1.Start JDeveloper by selecting **Start > Programs > Oracle Fusion Middleware 11.1.6.0.0 >JDeveloper Studio 11.1.1.6.0**.



2. In the Select Role dialog, choose “**Default Role**” and click “**OK**”.



The image shows a Windows-style dialog box titled "Select Role". The title bar is blue with a small icon on the left and a close button (X) on the right. The main area has a light beige background. At the top, there is a text instruction: "Select the role that matches your requirements. You can also change roles using the Roles page in preferences." To the right of this text is a small icon of a person. Below the instruction is a section labeled "Role:" followed by a list of five roles, each with a radio button and a description. The first role, "Default Role", is selected. At the bottom left, there is a checked checkbox labeled "Always prompt for role selection on startup". At the bottom right, there are two buttons: "OK" and "Cancel".

**Select Role**

Select the role that matches your requirements. You can also change roles using the Roles page in preferences.

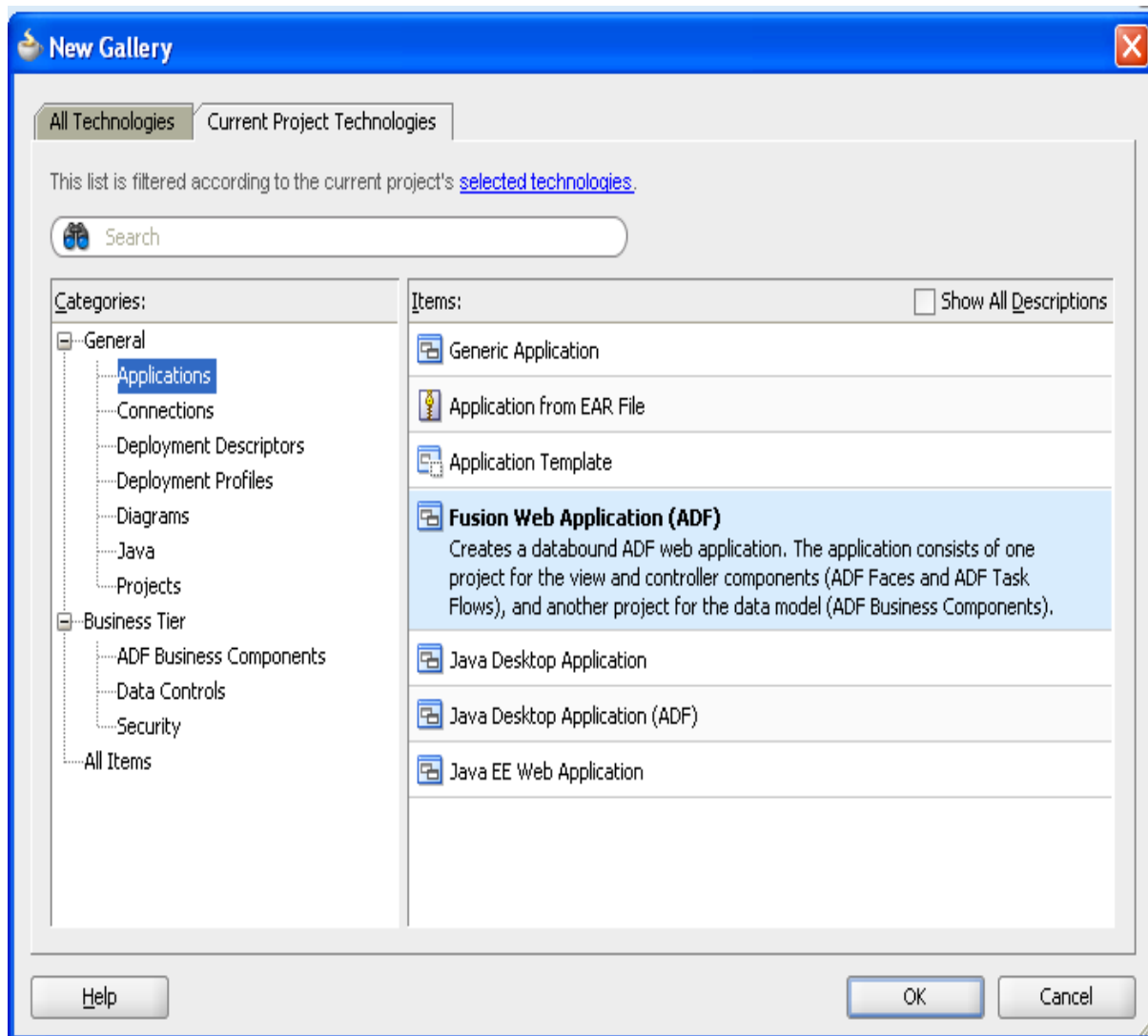
Role:

- ☒ **Default Role**  
Enables all technologies.
- ☐ **Customization Developer**  
Configures the product for customizing metadata.
- ☐ **Database Edition**  
Includes only features for core database development.
- ☐ **Java EE Edition**  
Includes only features for core Java EE development.
- ☐ **Java Edition**  
Includes only features for core Java development.

☒ Always prompt for role selection on startup

OK Cancel

3. **File > New** then selecting the Applications menu item in the left side of the new dialog, select the **Fusion Web Application (ADF)** type and click **“OK”**.



4. Create the application name as WeatherForeCast click “**Next**”.

**Create Fusion Web Application (ADF) - Step 1 of 5**

### Name your application

**Application Name**

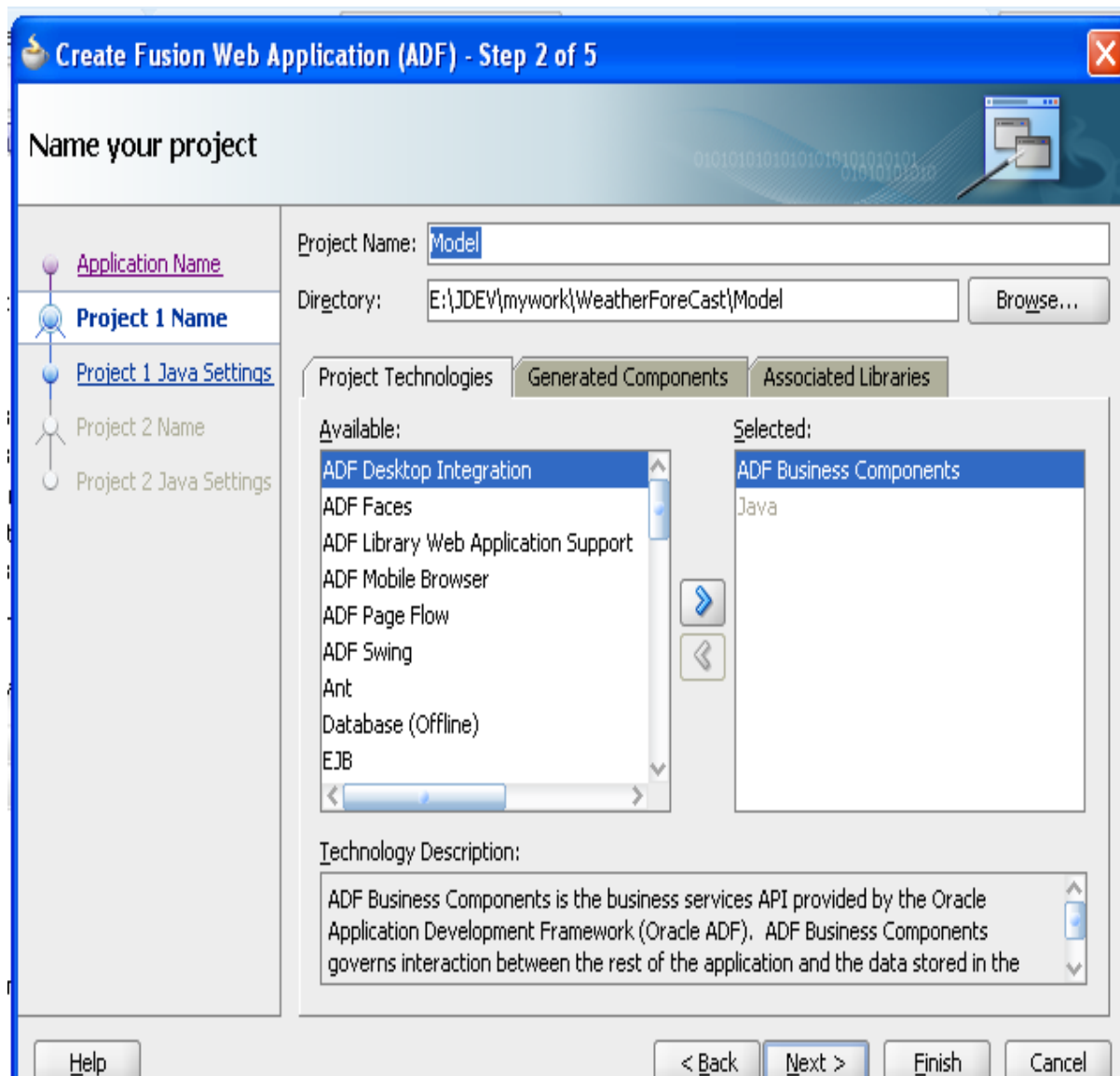
Application Name: WeatherForeCast

**Directory:**

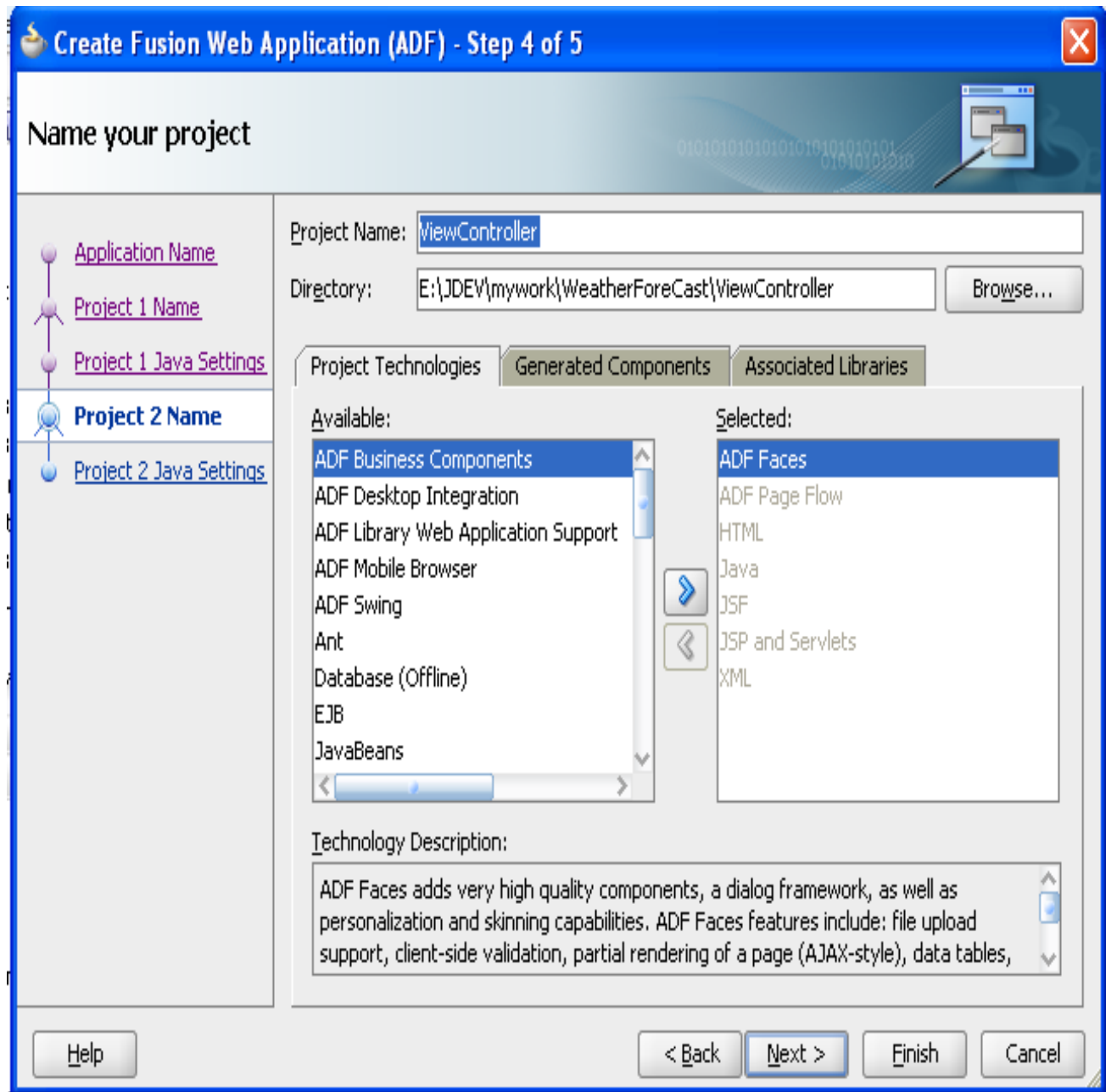
E:\JDEV\mywork\WeatherForeCast

**Application Package Prefix:**

5.It will create model project click “Next”.



6.It will create the View Controller project.



7.It will create javaSettings click “**Next**”.

**Create Fusion Web Application (ADF) - Step 5 of 5**

### Configure Java settings

Your new project starts with a default package, a source root directory, and an output directory.

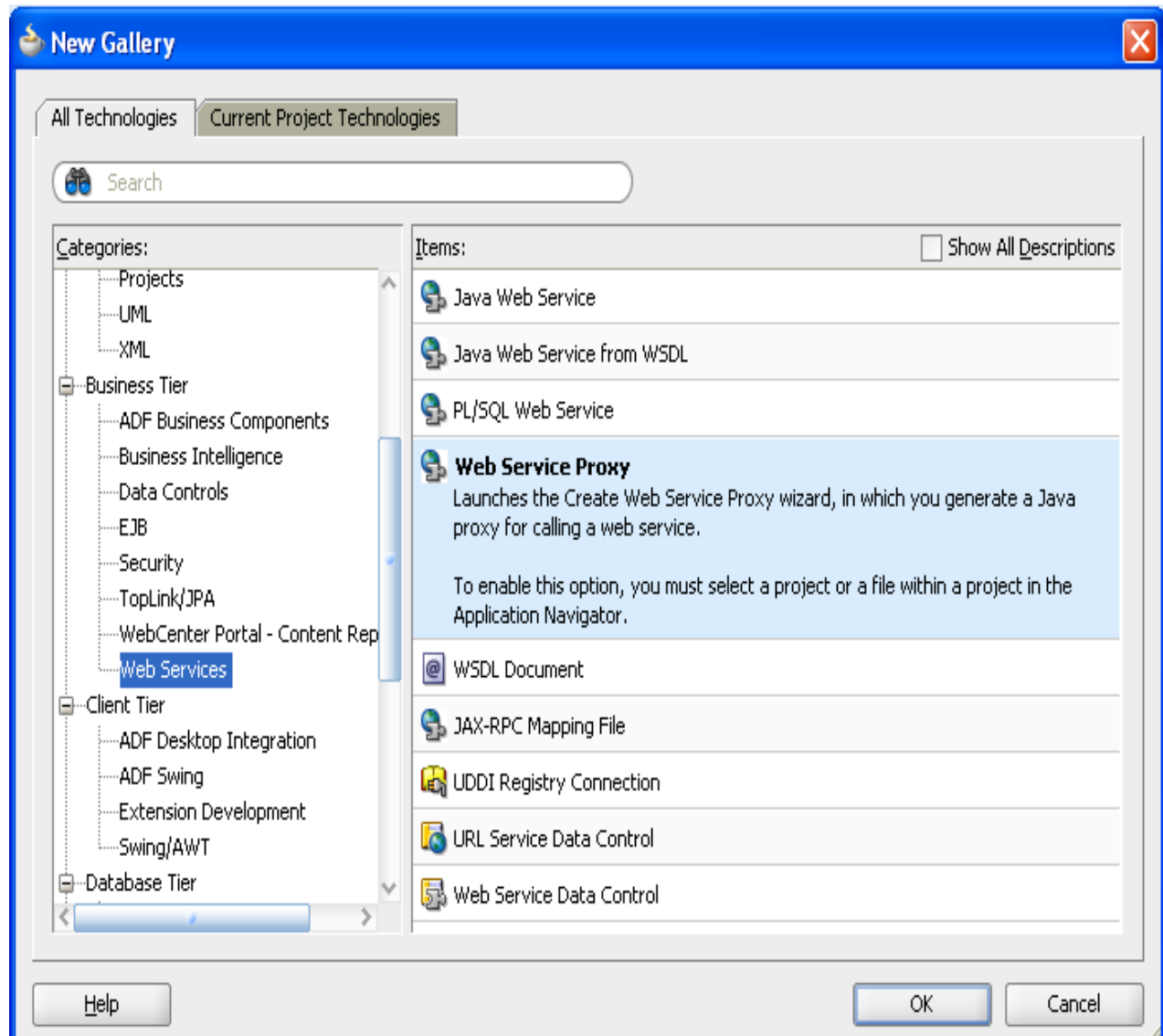
**Default Package:**  
view

**Java Source Path:**  
E:\JDEV\mywork\WeatherForeCast\ViewController\src Browse...

**Output Directory:**  
E:\JDEV\mywork\WeatherForeCast\ViewController\classes Browse...

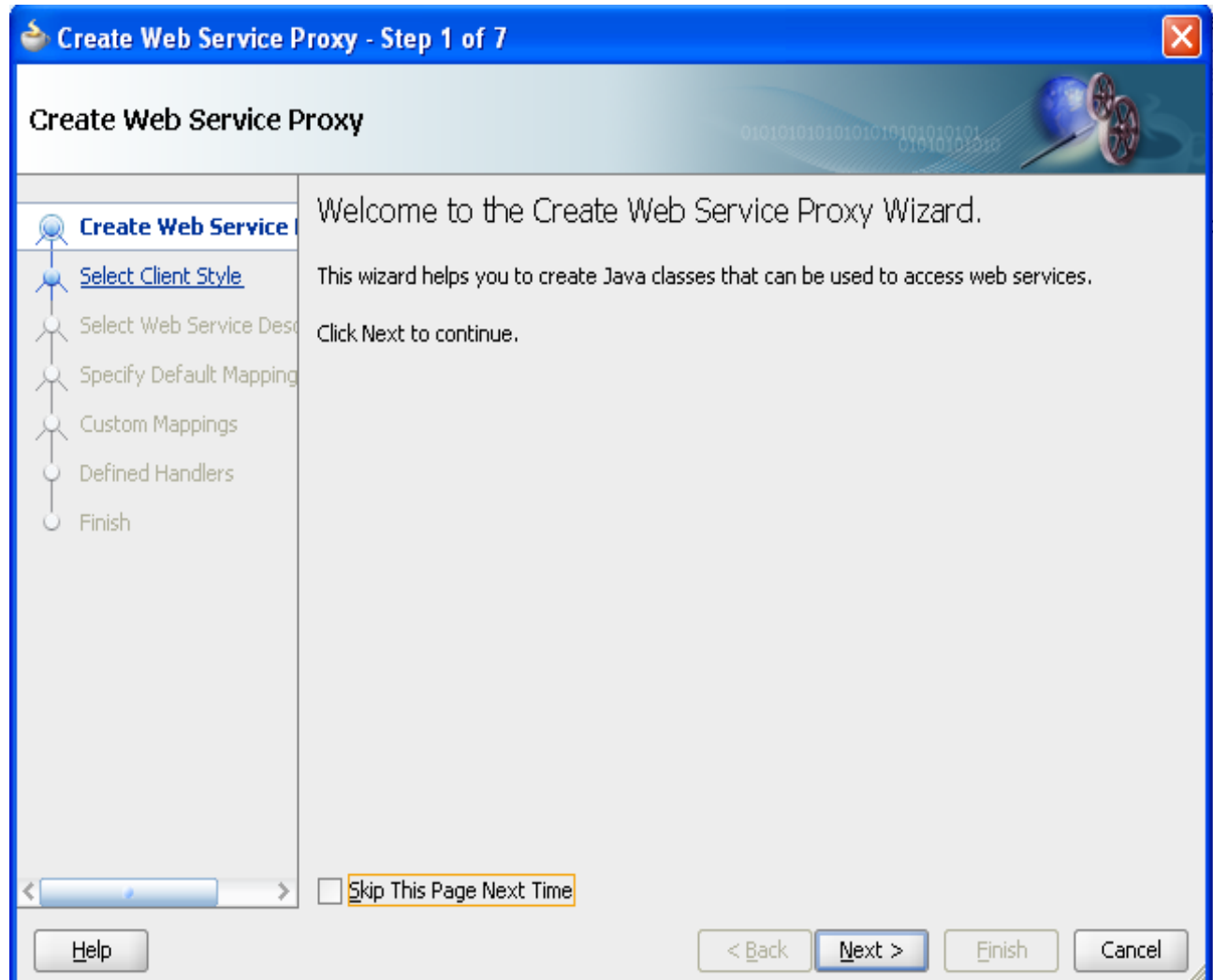
**Navigation:** Help, < Back, Next >, Finish, Cancel

8. Right click the model project select->New Select **WebServices** from the left side and **Web Services Proxy** on the right. Then click “OK”.

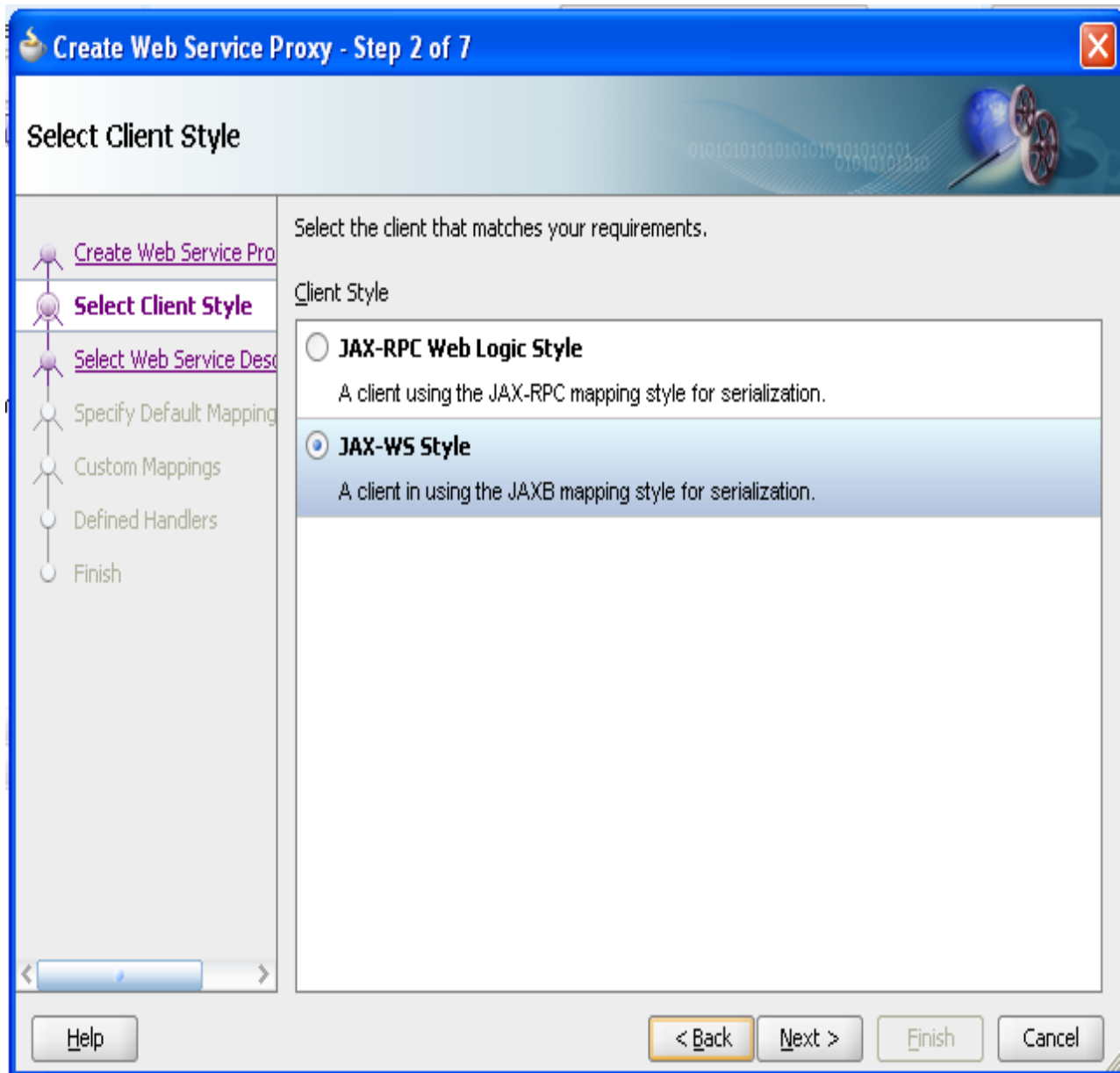




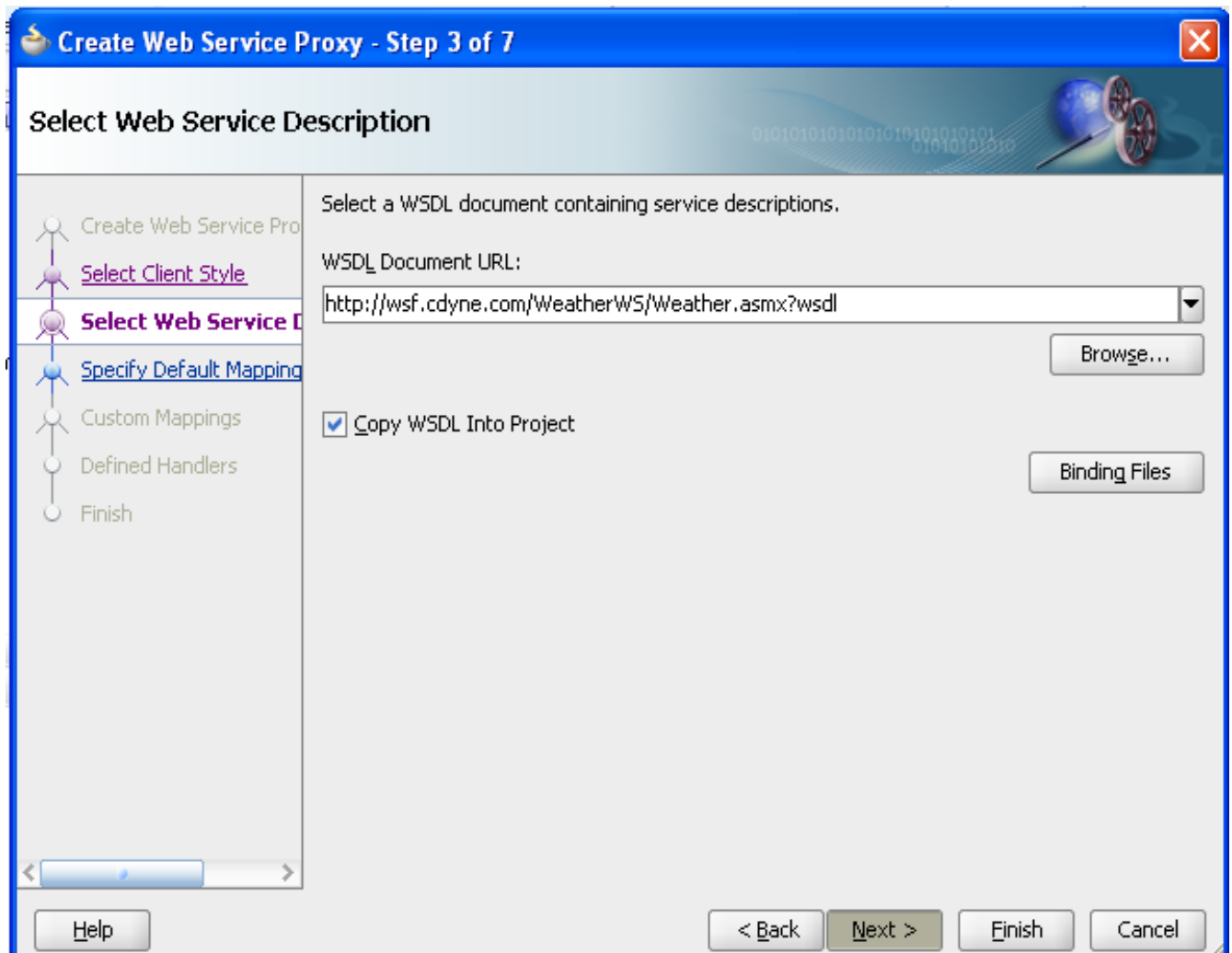
9. Click "Next".



10. Click "Next".



11. Now paste your WSDL url as shown below.



12. Click "Next".

**Create Web Service Proxy - Step 4 of 7**

### Specify Default Mapping Options

Provide default mappings from WSDL namespaces to Java packages. You can also control how parameters and headers are mapped.

Package Name:

Root Package for Generated Types:

The service appears to contain asynchronous features

☐ Generate As Async

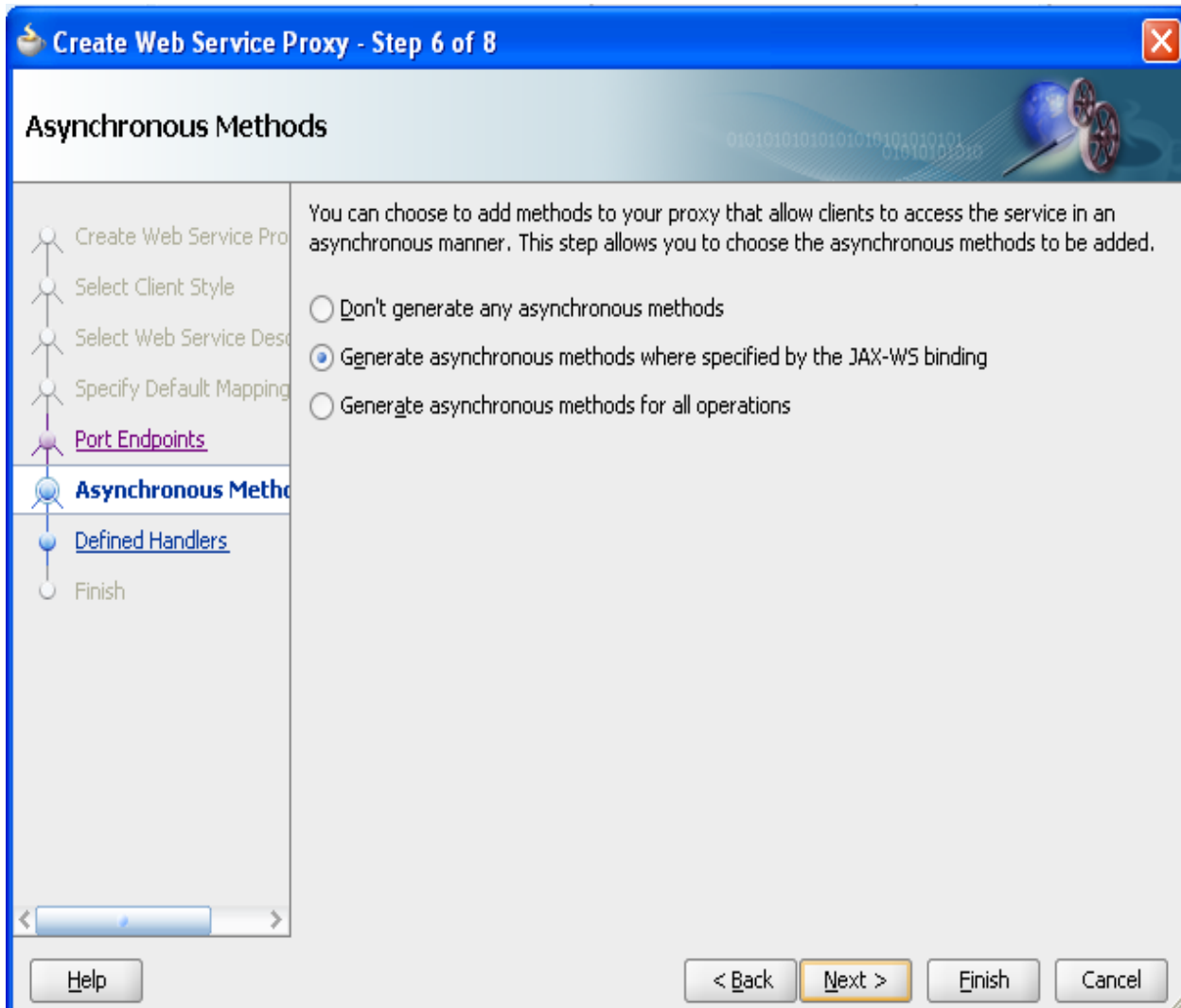
Callback Package Name:

☒ Unwrap Wrapped Parameters

**Progress Tree:**

- Create Web Service Proxy
- Select Client Style
- Select Web Service Description
- Specify Default Mapping**
- Custom Mappings
- Defined Handlers
- Finish

13. Click "Next".



**Create Web Service Proxy - Step 6 of 8**

### Asynchronous Methods

You can choose to add methods to your proxy that allow clients to access the service in an asynchronous manner. This step allows you to choose the asynchronous methods to be added.

☐ Don't generate any asynchronous methods

☒ Generate asynchronous methods where specified by the JAX-WS binding

☐ Generate asynchronous methods for all operations

**Navigation:**

- Create Web Service Proxy
- Select Client Style
- Select Web Service Description
- Specify Default Mapping
- Port Endpoints
- Asynchronous Methods**
- Defined Handlers
- Finish

**Buttons:** Help, < Back, Next >, Finish, Cancel

14. Click "Next".

**Create Web Service Proxy - Step 7 of 9**

### Policy

Configure OWSM policies for the web service client, if required.

Policy Store (def location):

☐ Show only the compatible client policies for selection.

Ports:

MTOM:

Reliability:

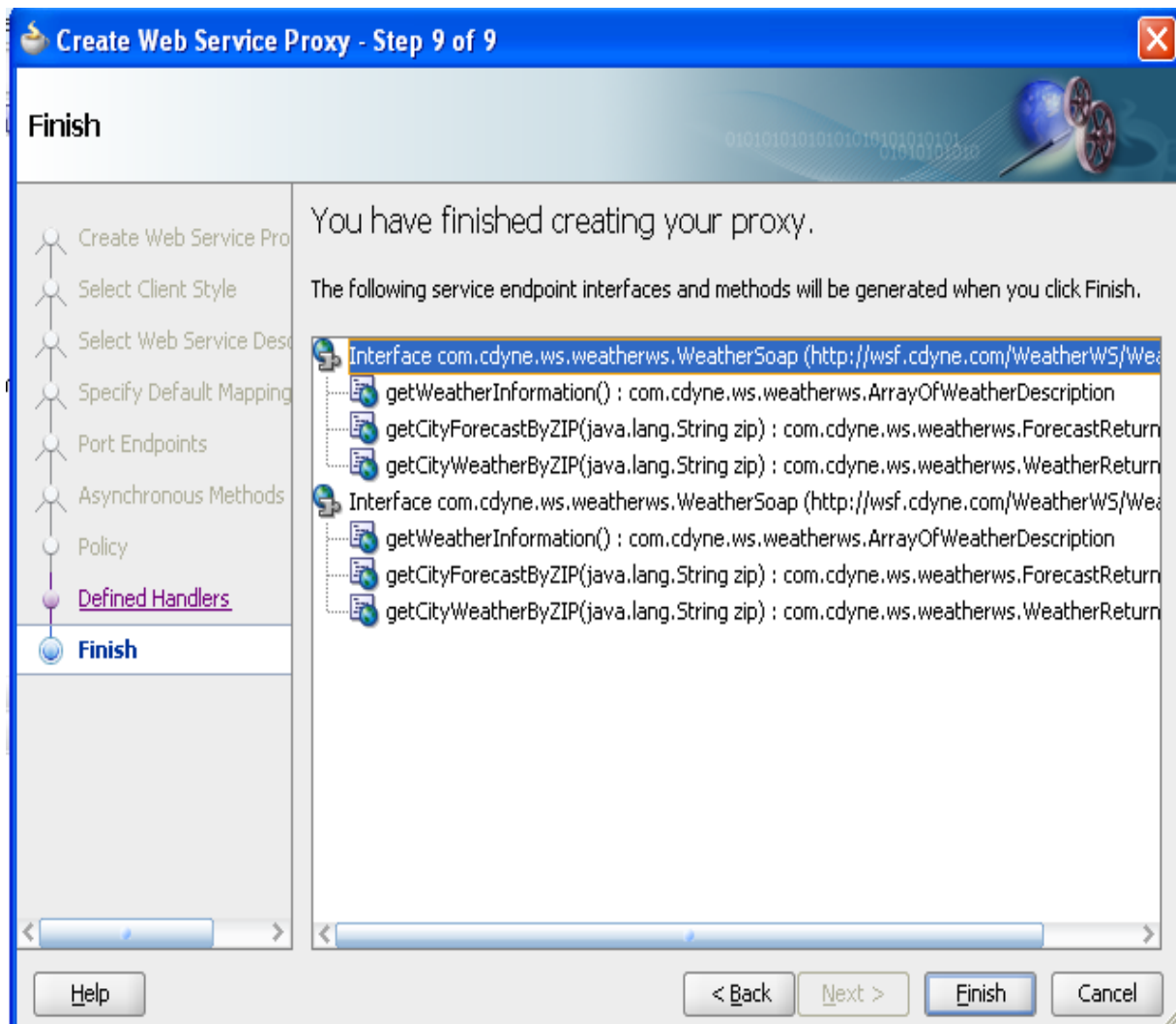
Addressing:

**Security** **Management**

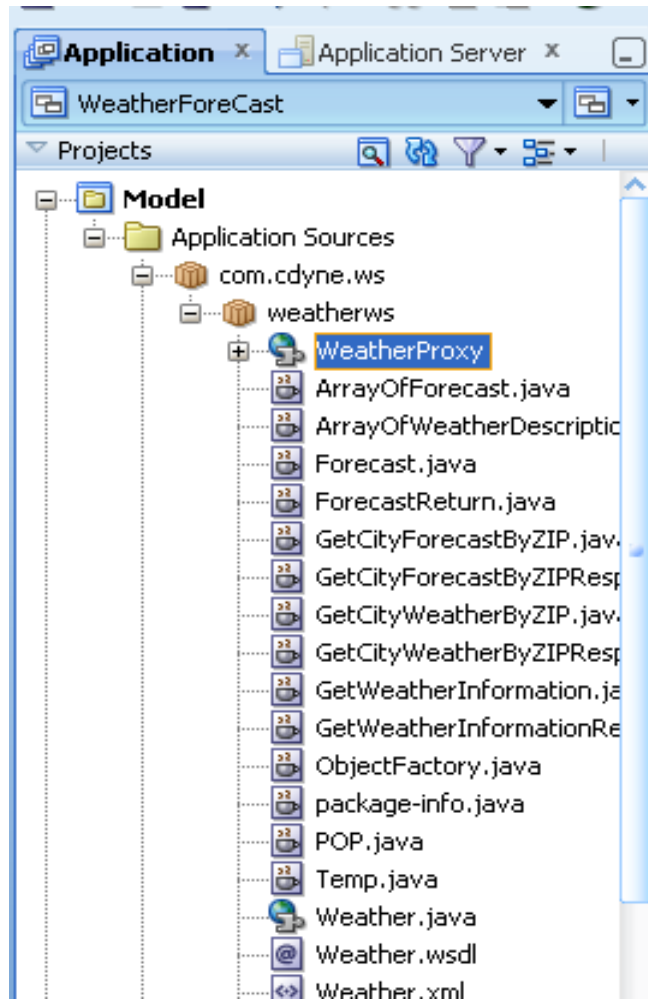
Policies: ☐ Show Selected Policies ☒ Show Descriptions

- ☐ **oracle/no\_authentication\_client\_policy**  
This policy facilitates the disabling of a globally attached authentication policy.  
This will include disabling that whole global policy containing any other assertions in addition to the authentication assertion. [View](#)
- ☐ **oracle/no\_messageprotection\_client\_policy**

15. Click "Next".



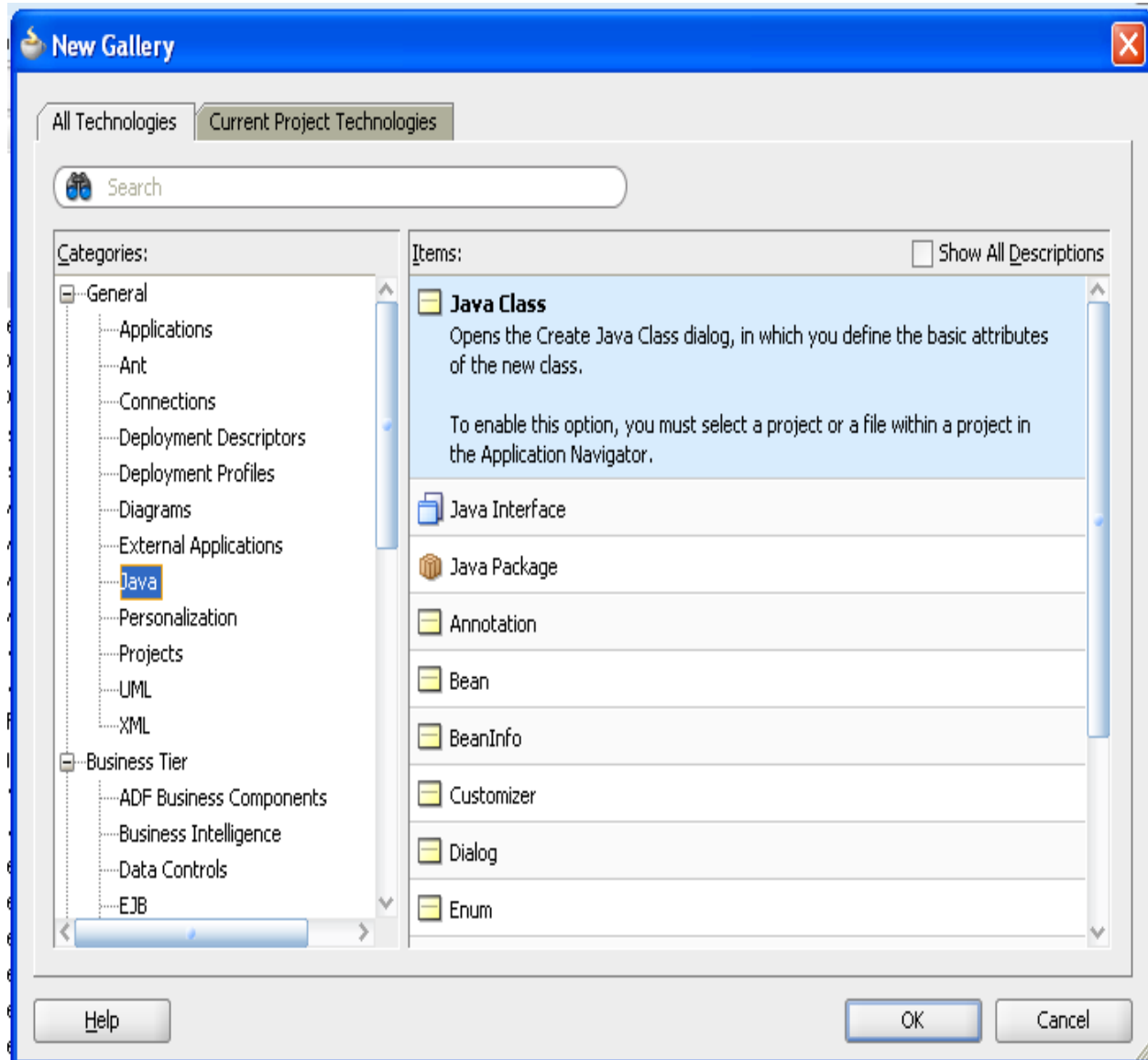
16. Now new Packages are created in your model project as shown below.



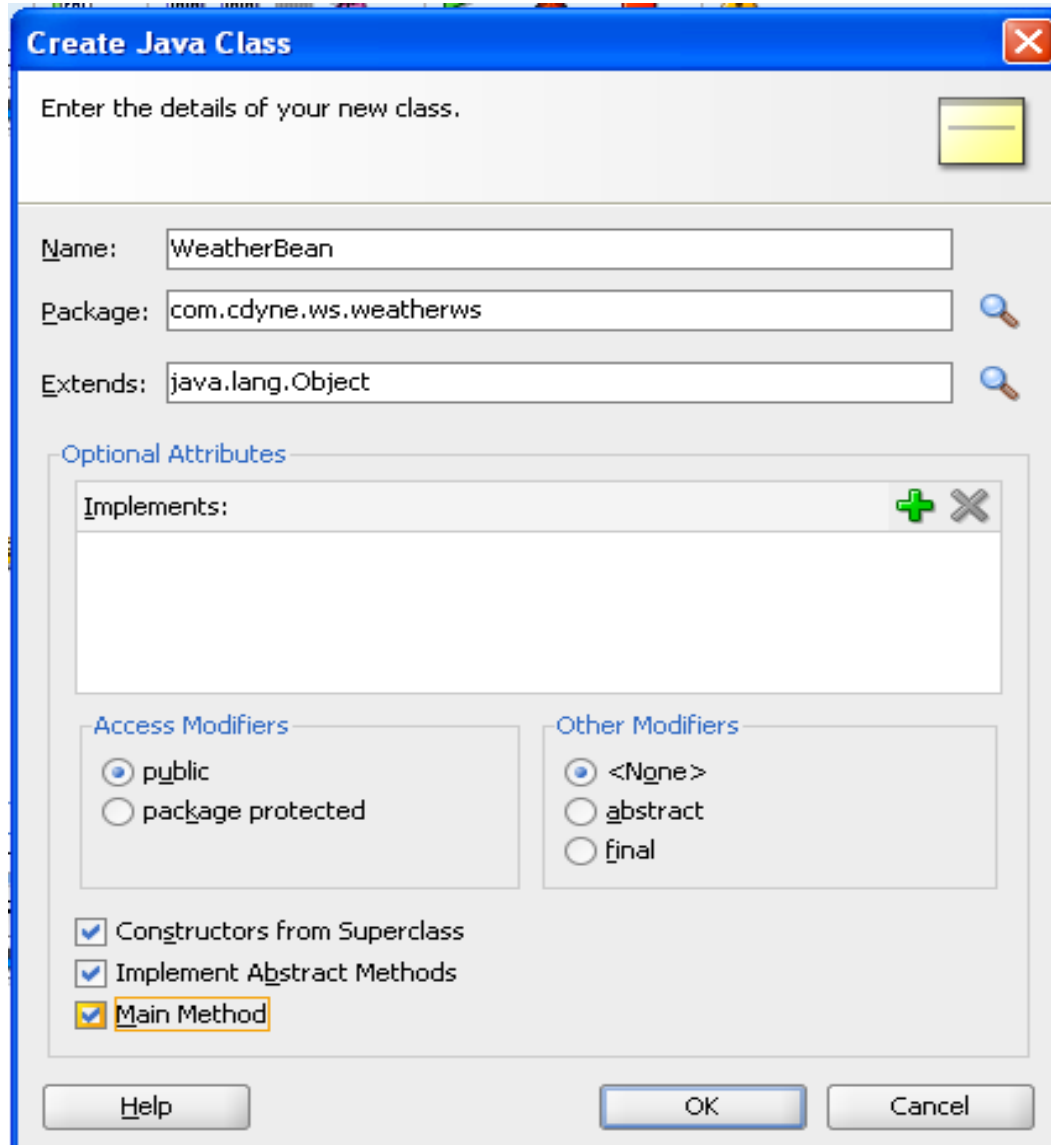


17. Right click the package “weatherws” select->New

Select **Java** from the left side and **Java Class** on the right. Then click “**OK**”.



18.Mention your class name as “WeatherBean”.



The image shows a 'Create Java Class' dialog box with a blue title bar and a red close button. The main area is light gray and contains the following elements:

- Enter the details of your new class.**: A text label at the top right of the main area.
- Name:**: A text field containing 'WeatherBean'.
- Package:**: A text field containing 'com.cdyne.ws.weatherws' with a magnifying glass icon to its right.
- Extends:**: A text field containing 'java.lang.Object' with a magnifying glass icon to its right.
- Optional Attributes**: A section with a light gray border containing:
  - Implements:**: A text field with a green plus icon and a gray X icon to its right.
  - Access Modifiers**: A group box containing two radio buttons: 'public' (selected) and 'package protected'.
  - Other Modifiers**: A group box containing three radio buttons: '<None>' (selected), 'abstract', and 'final'.
  - Three checked checkboxes: 'Constructors from Superclass', 'Implement Abstract Methods', and 'Main Method' (highlighted with a yellow border).
- Buttons**: 'Help', 'OK', and 'Cancel' buttons at the bottom.

19. In WeatherSoapClient.java file in under weatherws package ,we need to introduce following Method.

```
public static WeatherSoap getObj(){  
    weather = new Weather();  
    WeatherSoap weatherSoap = weather.getWeatherSoap();  
    return weatherSoap;  
}
```

**WeatherSoapClient.java Code:**

```
package com.cdyne.ws.weatherws;  
import javax.xml.ws.WebServiceRef;
```

```
public class WeatherSoapClient
```

```
{
```

```
    @WebServiceRef
```

```
    private static Weather weather;
```

```
    public static void main(String [] args)
```

```
{
```

```
        weather = new Weather();
```

```
        WeatherSoap weatherSoap = weather.getWeatherSoap();
```

```
}
```

```
    public static WeatherSoap getObj(){
```

```

        weather = new Weather();

        WeatherSoap weatherSoap = weather.getWeatherSoap();

        return weatherSoap;
    }
}

```

20. In WeatherBean.java file to add the following code:

```

public String getForeCast() {

    List<WeatherBean> list =new ArrayList<WeatherBean>();

    try{

        WeatherSoap ob;

        ob = WeatherSoapClient.getObj();

        String id=getZip();

        if(ob.getCityForecastByZIP(id).isSuccess()){

            System.out.println(ob.getCityForecastByZIP(id).city);

            for(int i=0;i<ob.getCityForecastByZIP(id).forecastResult().getForecast().size();i++){

                WeatherBean obj=new WeatherBean();

                this.city=ob.getCityForecastByZIP(id).city;
                this.date=ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).date.toString().substring(0
, 10);

                obj.setCity(ob.getCityForecastByZIP(id).city);

                obj.setDate(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).date.toString().substring
(0, 10));

                obj.setDis(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).description);

                obj.setHigh(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).getTemperatures().getD
aytimeHigh());
            }
        }
    }
}

```

```
obj.setLow(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).getTemperatures().getMorningLow());
```

```
short id1=ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).getWeatherID();
```

```
for(int j=0;j<ob.getWeatherInformation().weatherDescription.size();j++){
```

```
    if(ob.getWeatherInformation().getWeatherDescription().get(j).getWeatherID()==id1){
```

```
        obj.setImg(ob.getWeatherInformation().getWeatherDescription().get(j).getPictureURL());
```

```
    }
```

```
}
```

```
list.add(obj);
```

```
}
```

```
System.out.println("BEFORE LENGTH CALCULATION");
```

```
int n=list.size();
```

```
n=n-1;
```

```
System.out.println("Length "+n);
```

```
setLen(n+"");
```

```
setList1(list);
```

```
return "success";
```

```
}else{
```

```
    return "failure";
```

```
}
```

```
}
```

```
catch(Exception e){
```

```
    e.printStackTrace();
```

```
    return "failure";
```

```
} }
```

In our case ForeCast information and weather information are in different class ,in our case we need to Show forecast details and weather image.

So I match the Weather image for particular forecast using weather id.

**WeatherBean.java Code:**

```
package com.cdyne.ws.weatherws;

import java.util.ArrayList;

import java.util.List;

public class WeatherBean {

    public WeatherBean() {

        super();

    }

    private String name;

    private String msg;

    private String len;

    private String date;

    private String city;

    private String dis;

    private String high;

    private String low;

    private String img;

    private String zip;

    private List<WeatherBean> list1=null;

    private List<String> list2=new ArrayList<String>();
```

```
public void setName(String name) {  
    this.name = name;  
}  
  
public String getName() {  
    return name;  
}  
  
public void setMsg(String msg) {  
    this.msg = msg;  
}  
  
public String getMsg() {  
    return msg;  
}  
  
public void setLen(String len) {  
    this.len = len;  
}  
  
public String getLen() {  
    return len;  
}  
  
public void setDate(String date) {  
    this.date = date;  
}  
  
public String getDate() {  
    return date;  
}
```

```
public void setCity(String city) {  
    this.city = city;  
}  
  
public String getCity() {  
    return city;  
}  
  
public void setDis(String dis) {  
    this.dis = dis;  
}  
  
public String getDis() {  
    return dis;  
}  
  
public void setHigh(String high) {  
    this.high = high;  
}  
  
public String getHigh() {  
    return high;  
}  
  
public void setLow(String low) {  
    this.low = low;  
}  
  
public String getLow() {  
    return low;  
}
```



```

public void setImg(String img) {

    this.img = img;

}

public String getImg() {

    return img;

}

public void setList1(List<WeatherBean> list1) {

    this.list1 = list1;

}

public List<WeatherBean> getList1() {

    return list1;

}

public void setList2(List<String> list2) {

    this.list2 = list2;

}

public List<String> getList2() {

    return list2;

}

public String getForeCast() {

    List<WeatherBean> list =new ArrayList<WeatherBean>();

    try{

        WeatherSoap ob;

        ob = WeatherSoapClient.getObj();

        String id=getZip();

        if(ob.getCityForecastByZIP(id).isSuccess()){

```

```

        System.out.println(ob.getCityForecastByZIP(id).city);

        for(int i=0;i<ob.getCityForecastByZIP(id).forecastResult().getForecast().size();i++){

            WeatherBean obj=new WeatherBean();

            this.city=ob.getCityForecastByZIP(id).city;

            this.date=ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).date.toString().substring(0
, 10);

            obj.setCity(ob.getCityForecastByZIP(id).city);

            obj.setDate(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).date.toString().substring
(0, 10));

            obj.setDis(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).description);

            obj.setHigh(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).getTemperatures().getD
aytimeHigh());

            obj.setLow(ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).getTemperatures().getM
orningLow());

            short
            id1=ob.getCityForecastByZIP(id).getForecastResult().getForecast().get(i).getWeatherID();

            for(int j=0;j<ob.getWeatherInformation().weatherDescription.size();j++){

                if(ob.getWeatherInformation().getWeatherDescription().get(j).getWeatherID()==id1){

                    obj.setImg(ob.getWeatherInformation().getWeatherDescription().get(j).getPictureURL());

                }

            }

            list.add(obj);

```

```
    }

    System.out.println("BEFORE LENGTH CALCULATION");

    int n=list.size();

    n=n-1;

    System.out.println("Length "+n);

    setLen(n+"");

    setList1(list);

    return "success";

    }else{

        return "failure";

    }

}

}

catch(Exception e){

    e.printStackTrace();

    return "failure";

}

}

public void setZip(String zip) {

    this.zip = zip;

}

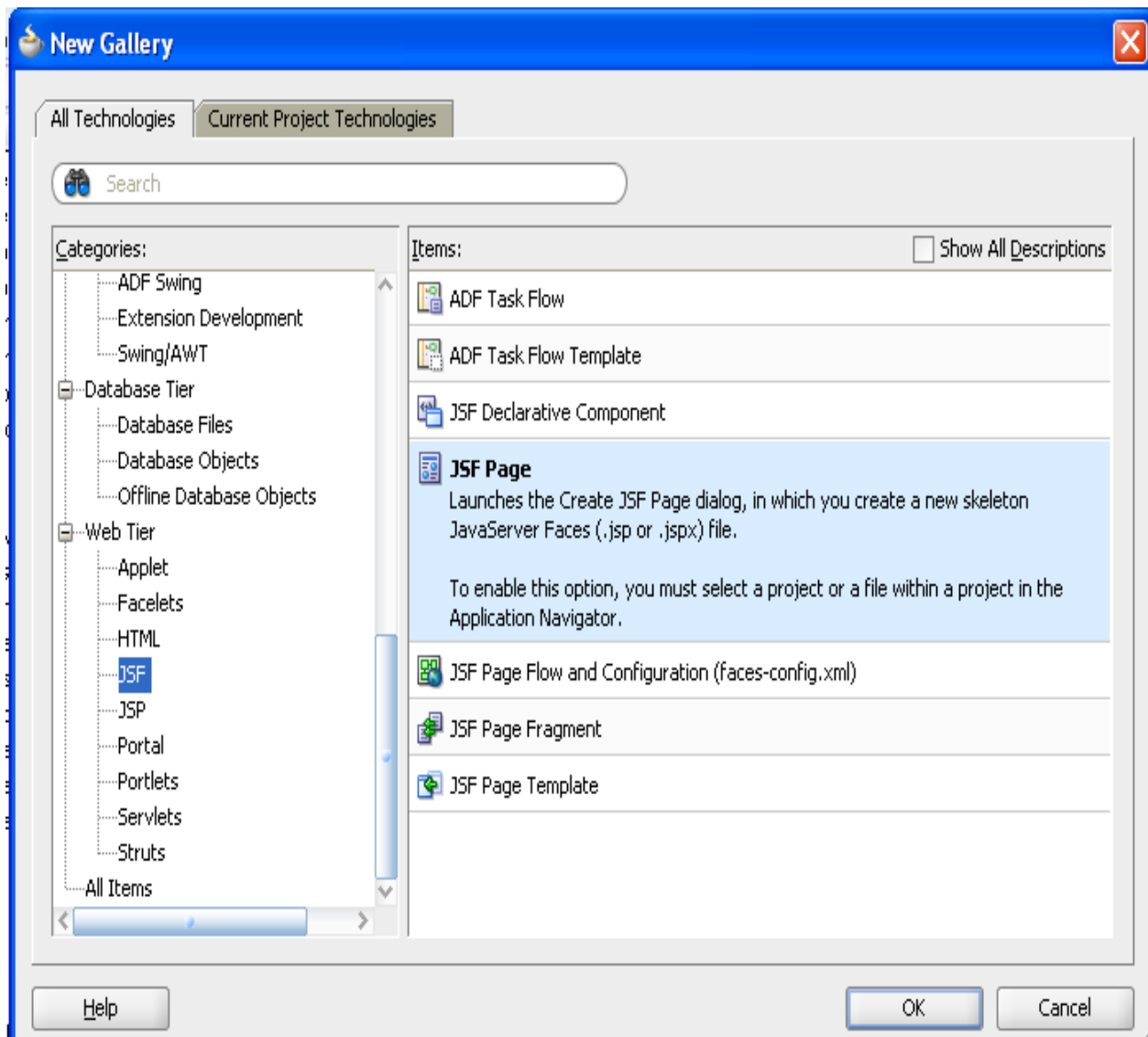
public String getZip() {

    return zip;

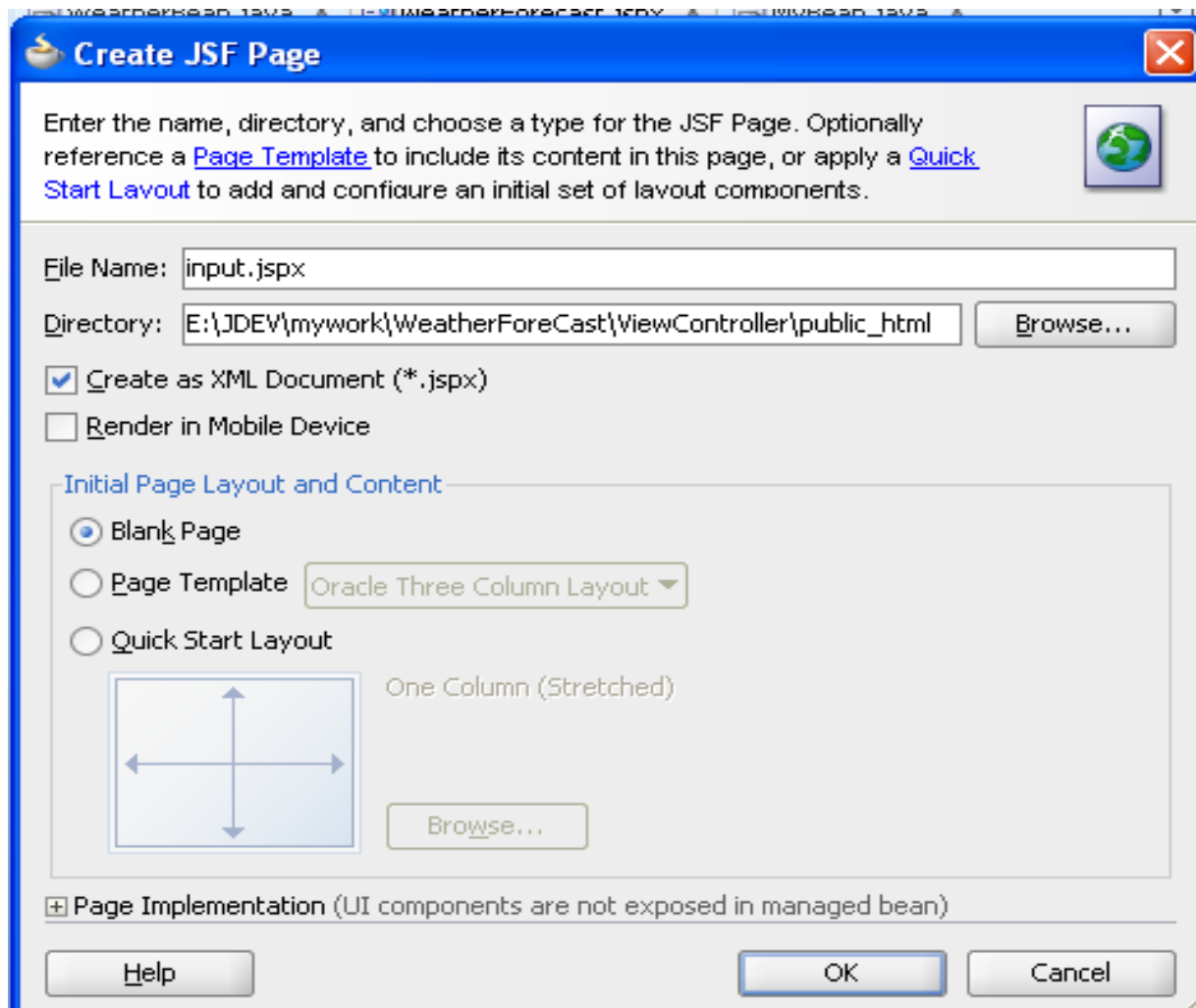
}

}
```

21. **Right-click** on the WebContent Folder and select **new**. Select **JSF** from the left side and **JSF Page** on the right. Then click “**OK**”.



22.Mention your page name as input.jspx.



The image shows a 'Create JSF Page' dialog box with a blue title bar and a close button. The main area has a light gray background. At the top, there is a text instruction: 'Enter the name, directory, and choose a type for the JSF Page. Optionally reference a [Page Template](#) to include its content in this page, or apply a [Quick Start Layout](#) to add and configure an initial set of layout components.' To the right of this text is a small globe icon. Below the instruction, there are three input fields: 'File Name:' with the text 'input.jspx', 'Directory:' with the text 'E:\JDEV\mywork\WeatherForeCast\ViewController\public\_html', and a 'Browse...' button to the right of the directory field. Below these fields are two checkboxes: 'Create as XML Document (\*.jspx)' which is checked, and 'Render in Mobile Device' which is unchecked. Below the checkboxes is a section titled 'Initial Page Layout and Content' with a light gray background. This section contains three radio buttons: 'Blank Page' (selected), 'Page Template' (with a dropdown menu showing 'Oracle Three Column Layout'), and 'Quick Start Layout'. Below the radio buttons is a diagram of a page layout with a central square and four arrows pointing outwards, labeled 'One Column (Stretched)'. To the right of the diagram is a 'Browse...' button. At the bottom of the dialog, there is a section titled 'Page Implementation (UI components are not exposed in managed bean)' with a plus icon. Below this section are three buttons: 'Help', 'OK', and 'Cancel'.

**Create JSF Page**

Enter the name, directory, and choose a type for the JSF Page. Optionally reference a [Page Template](#) to include its content in this page, or apply a [Quick Start Layout](#) to add and configure an initial set of layout components.

File Name:

Directory:

☒ Create as XML Document (\*.jspx)


☐ Render in Mobile Device

**Initial Page Layout and Content**

☒ Blank Page

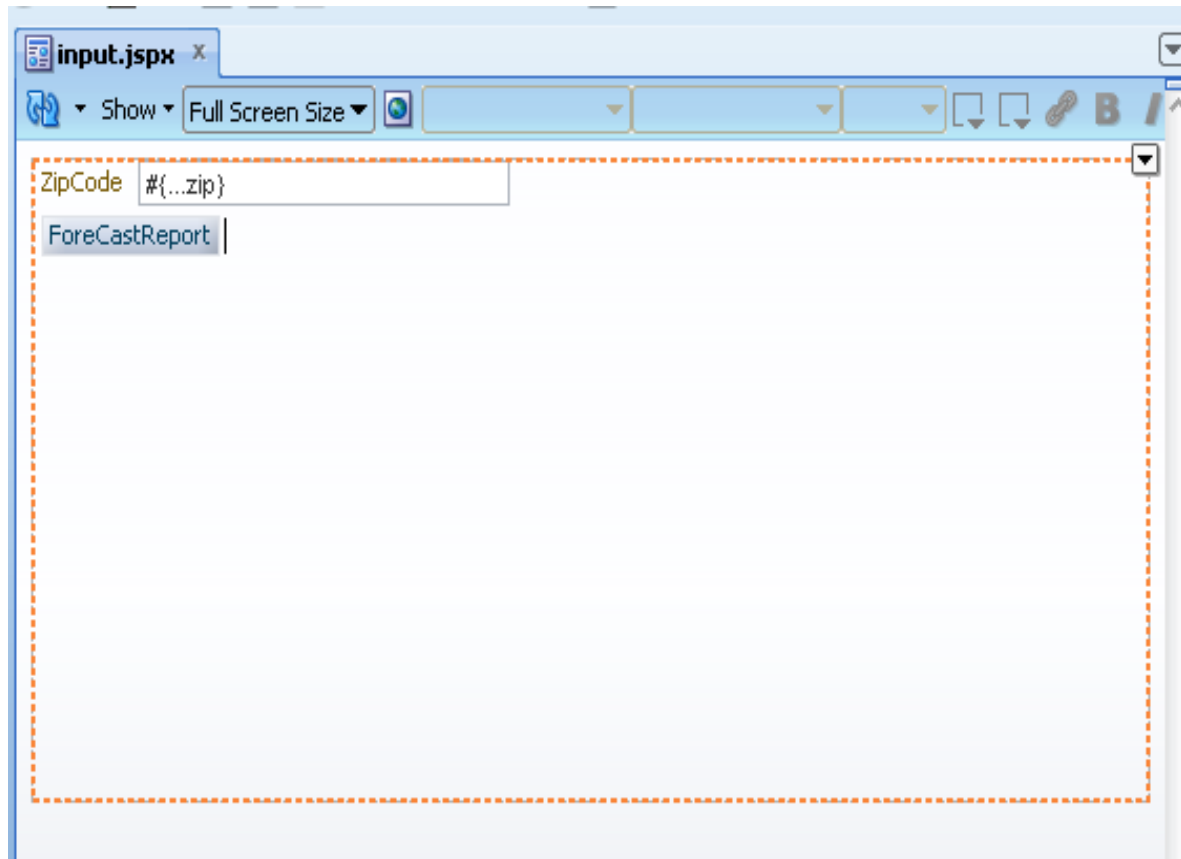
☐ Page Template

☐ Quick Start Layout

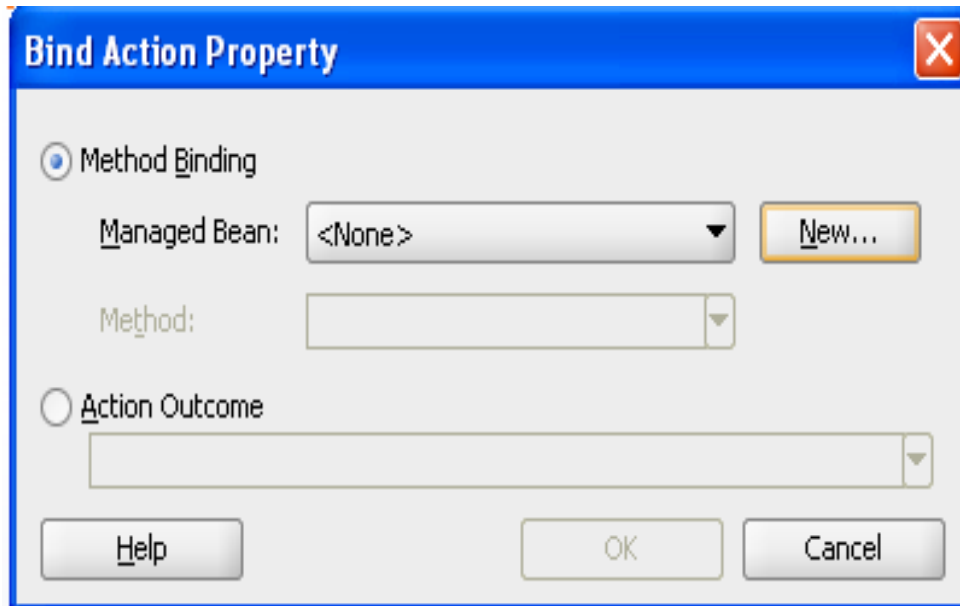
 One Column (Stretched)

**Page Implementation (UI components are not exposed in managed bean)**

23. In input.jspx page drag and drop a input text and button.



24. Double Click the button.



The image shows a 'Bind Action Property' dialog box with a blue title bar and a red close button. It contains two radio buttons: 'Method Binding' (selected) and 'Action Outcome'. Under 'Method Binding', there is a 'Managed Bean' dropdown menu showing '<None>' and a 'New...' button. Below that is a 'Method' dropdown menu. Under 'Action Outcome', there is a single dropdown menu. At the bottom are 'Help', 'OK', and 'Cancel' buttons.

**Bind Action Property**

☒ Method Binding

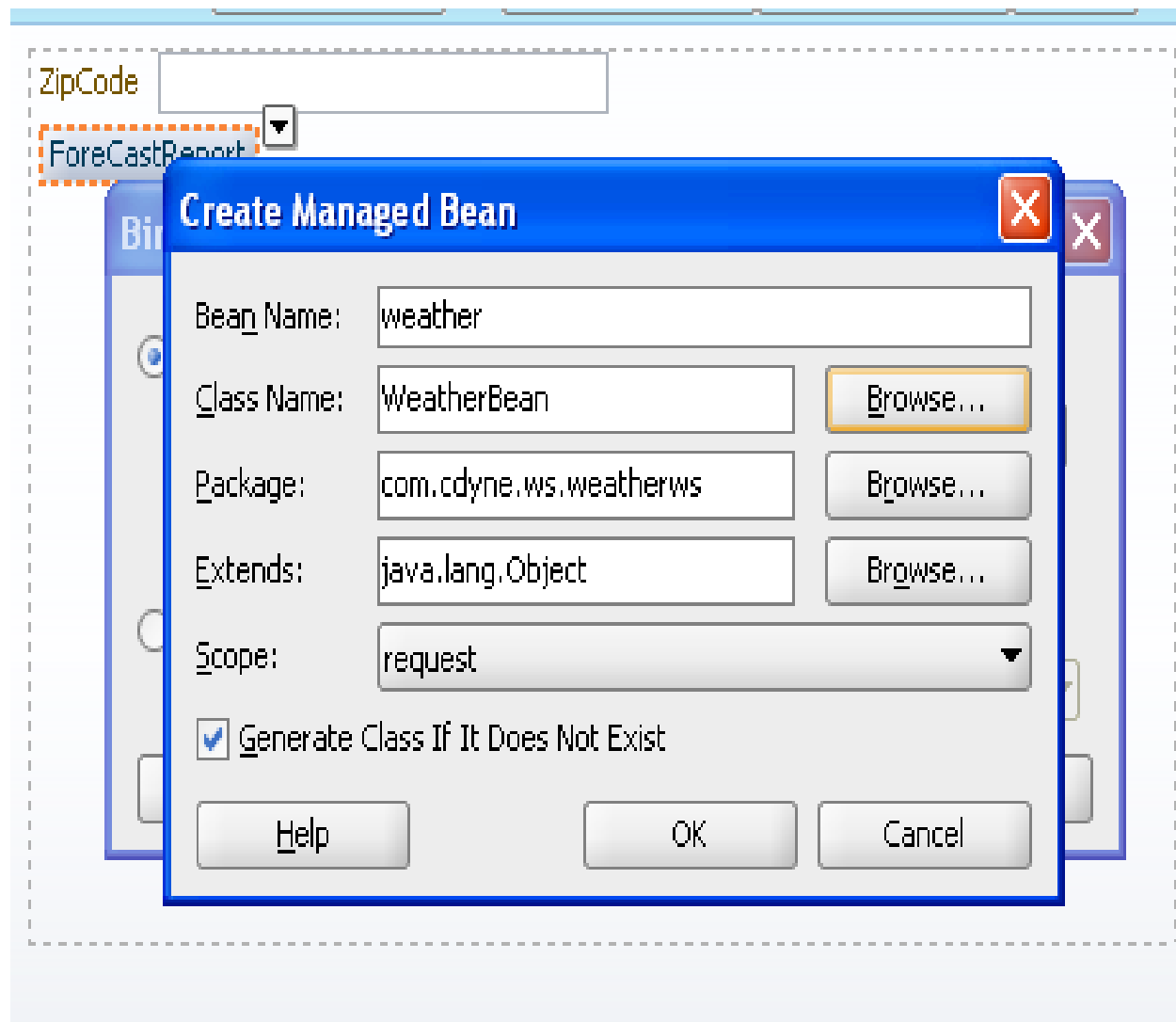
Managed Bean: <None> New...

Method:

☐ Action Outcome

Help OK Cancel

25. Give the Bean name as Weather, and click browse button select your WeatherBean







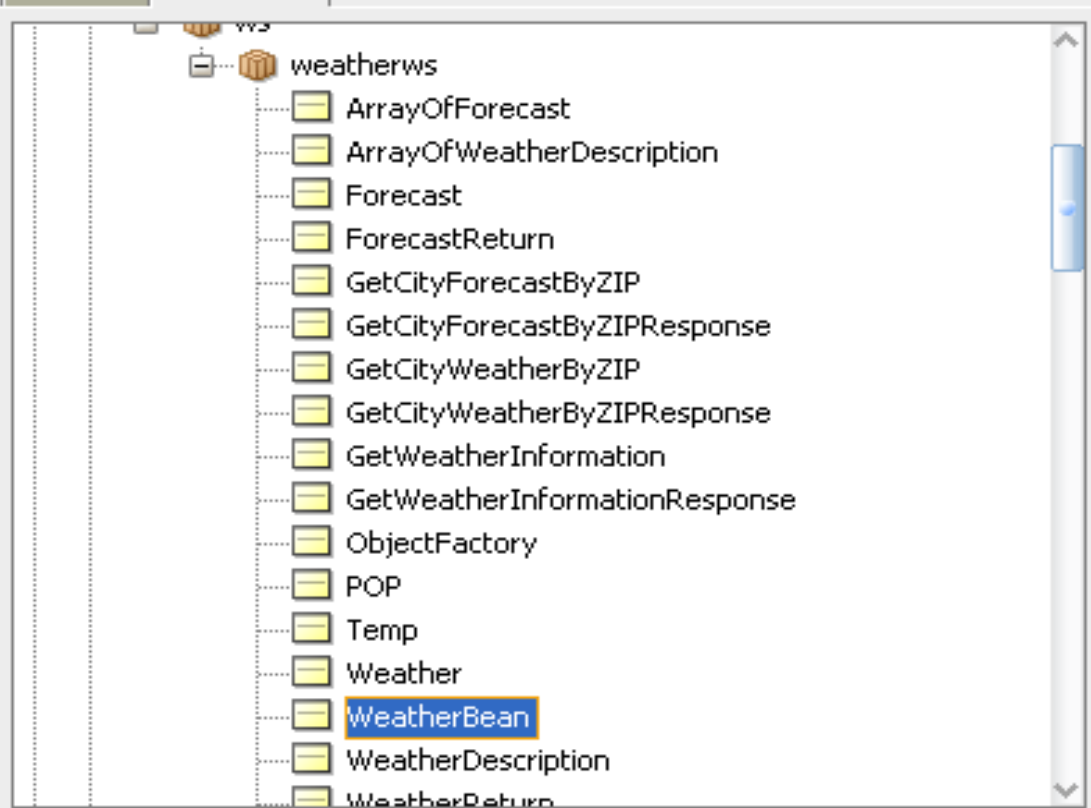
## Class Browser



To search, enter the simple class name, or the package prefix to search by package. Use a question mark (?) to match any single character, or an asterisk (\*) to match any number of characters.

Search

Hierarchy



Help

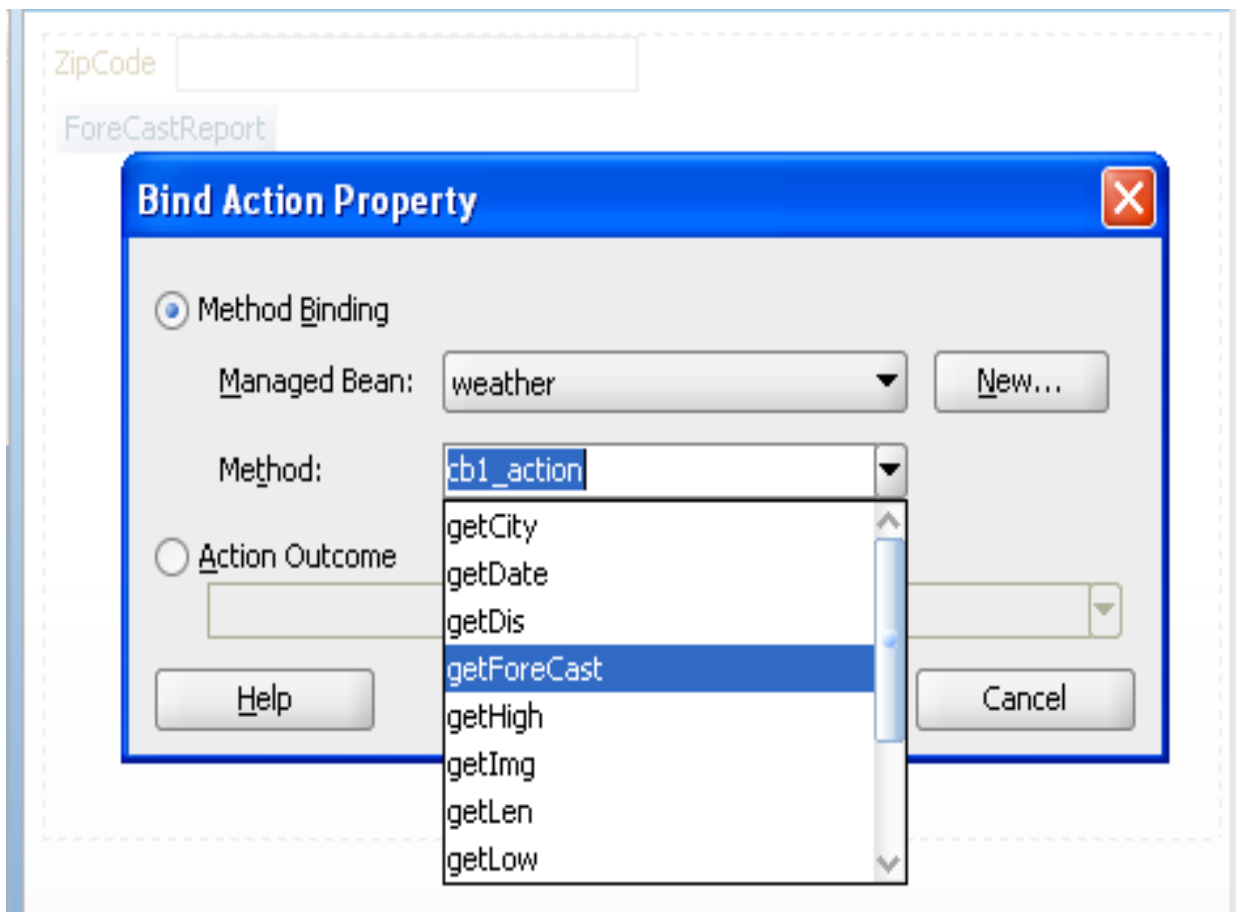
OK

Cancel

26.

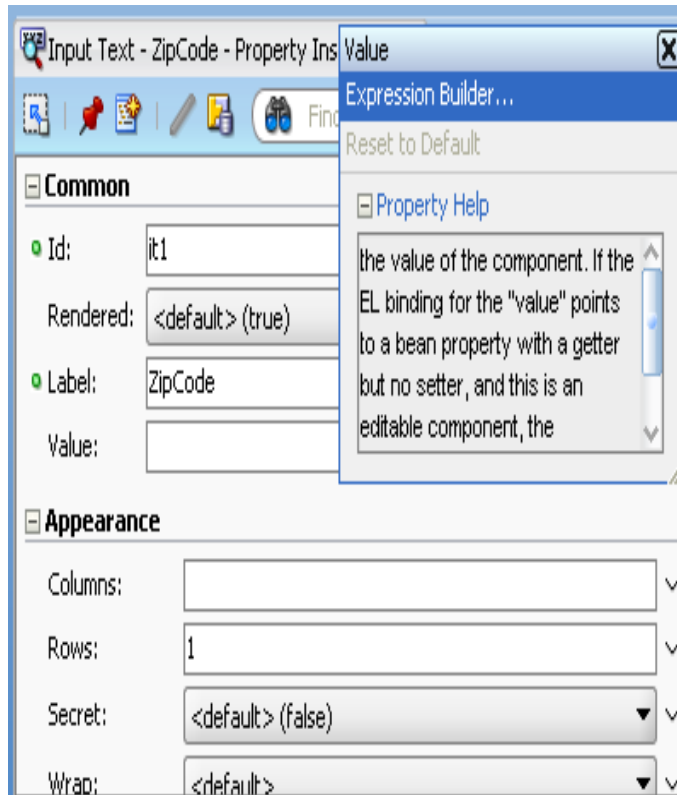
a) In Managed bean select your bean weather.

b) Method: select your getForeCast method.

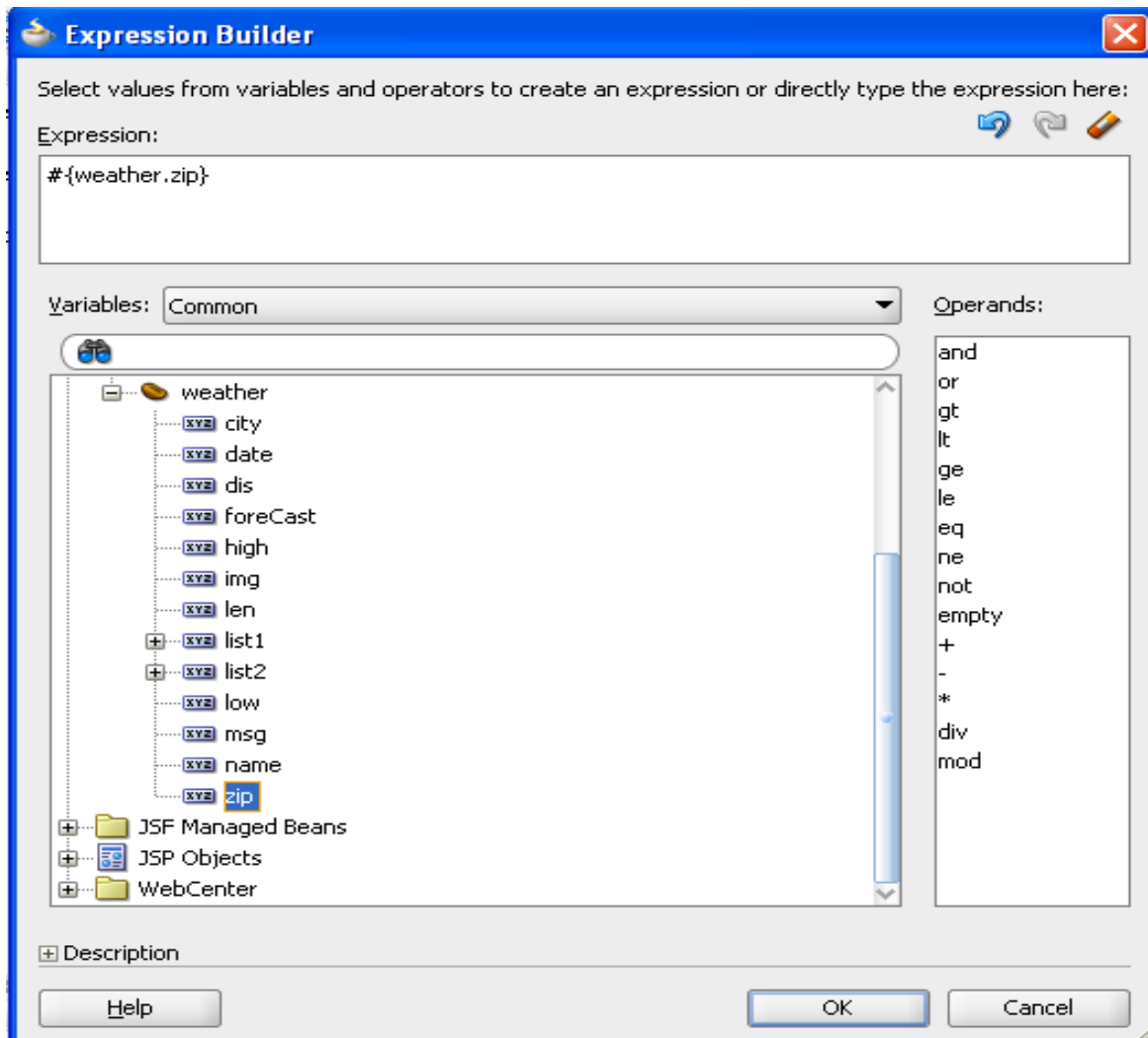


27. Right click the input text go to “properties” give the label name as ZipCode.

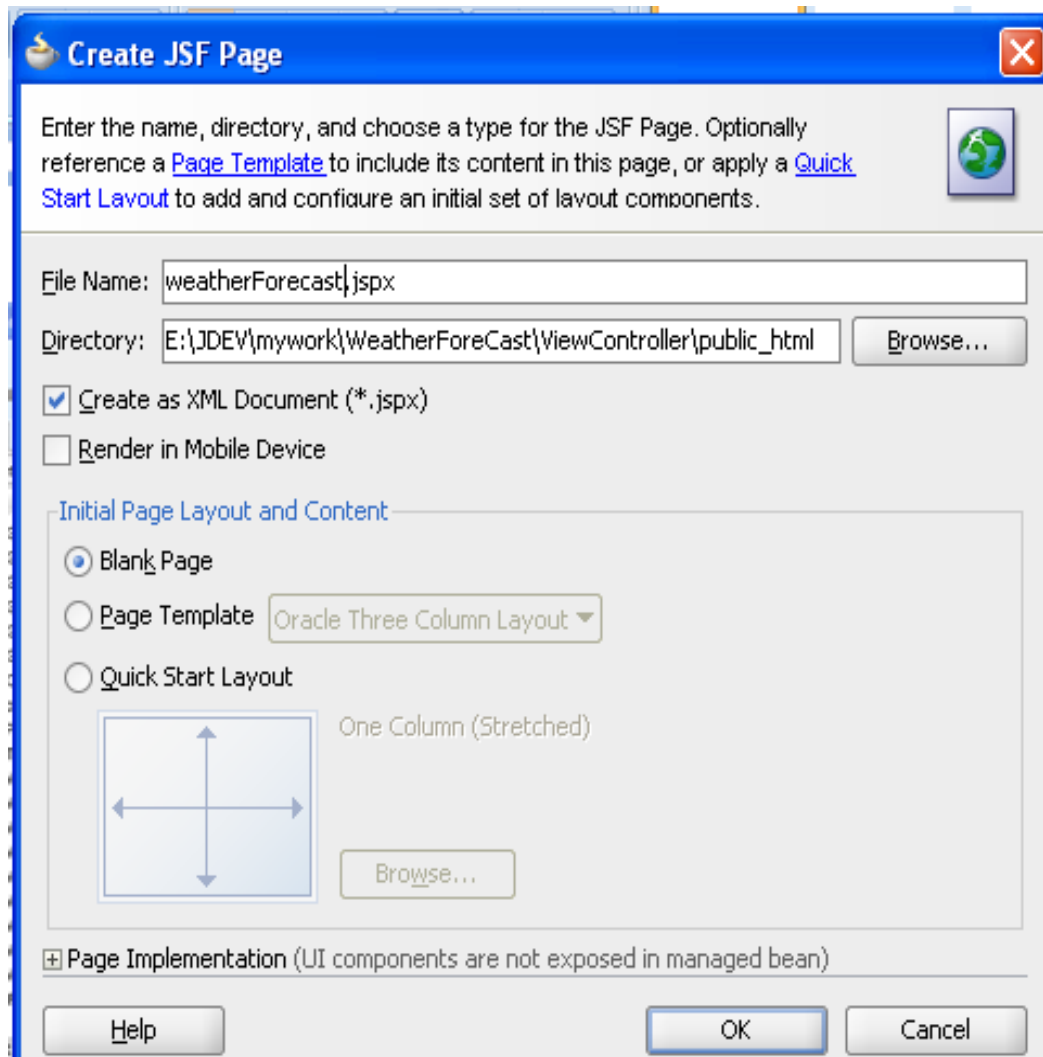
In Value attribute select expression builder



28.select ADF Managed Bean->weather->zip as shown below.



29. Create another jsp page name as weatherForeCast.jspx



30.To add following code in your weatherForeCast.jspx page:

```
<h:outputText id="msg" value="#{weather.msg}" style="width: 500px;color: red; " ></h:outputText>
```

```
    <h:dataTable value="#{weather.list1}" var="item" border="1">
```

```
    <h:column>
```

```
        <f:facet name="header">
```

```
            <h:outputText value="City" style="width: 130px; font-family: verdana;" />
```

```
        </f:facet>
```

```
            <h:outputText value="#{item.city}" style="width: 130px; font-family: verdana;" />
```

```
        </h:column>
```

```
        <h:column>
```

```
            <f:facet name="header">
```

```
                <h:outputText value="City" style="width: 130px; font-family: verdana;" />
```

```
            </f:facet>
```

```
                <h:outputText value="#{item.city}" style="width: 130px; font-family: verdana;" />
```

```
            </h:column>
```

```
            <h:column>
```

```
                <f:facet name="header">
```

```
                    <h:outputText value="Date" style="width: 130px; font-family: verdana;" />
```

```
                </f:facet>
```

```
                    <h:outputText value="#{item.date}" style="width: 130px; font-family: verdana;" />
```

```
                </h:column>
```

```
            <h:column>
```

```
                <f:facet name="header">
```

```
                    <h:outputText value="Discription" style="width: 130px; font-family: verdana;" />
```

```
</f:facet>

<h:outputText value="#{item.dis}" style="width: 130px; font-family: verdana;" />

</h:column>

<h:column>

<f:facet name="header">

<h:outputText value="High" style="width: 130px; font-family: verdana;" />

</f:facet>

  <h:outputText value="#{item.high}" style="width: 130px; font-family: verdana;" />

</h:column>

  <h:column>

<f:facet name="header">

<h:outputText value="Low" style="width: 130px; font-family: verdana;" />

</f:facet>

  <h:outputText value="#{item.low}" style="width: 130px; font-family: verdana;" />

</h:column>

  <h:column>

<f:facet name="header">

<h:outputText value="Image" style="width: 200px; font-family: verdana;" />

</f:facet>

  <af:image id="im" source="#{item.img}" ></af:image>

</h:column>

</h:dataTable>
```

31.Create another jsp page name as error.jsp

To add following code in error.jsp page:

```
<af:form id="f1">
```

```
    <br></br>
```

```
    <br></br>
```

```
    <br></br>
```

```
    <br></br>
```

```
    <center><b><font color="red" size="4">
```

Your ZipCode Does Not Point Out Any City

Pls Check Your ZipCode !....

```
    <br></br>
```

```
    <br></br>
```

```
    </font></b>
```

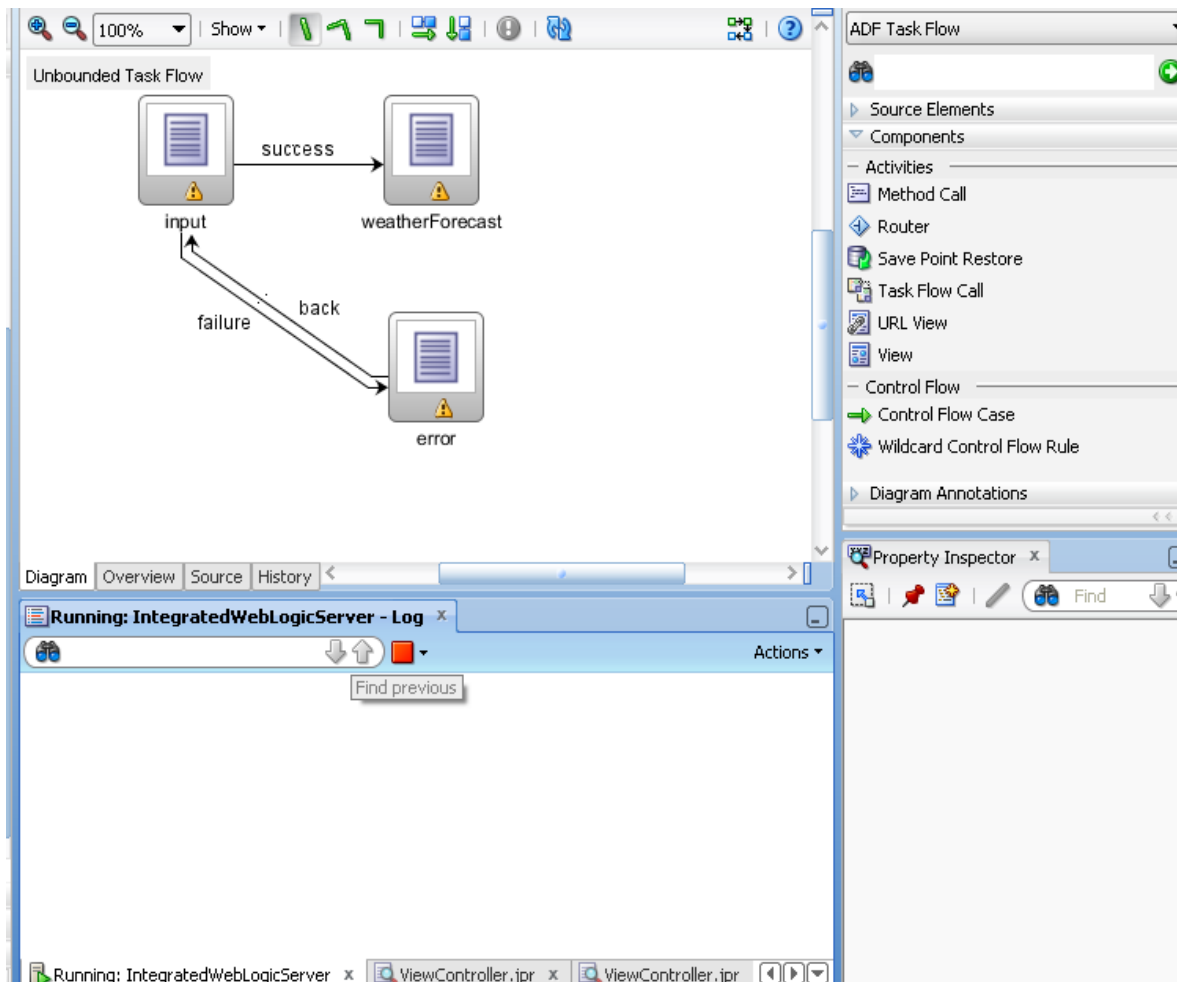
```
    <af:commandButton text="Back" id="cb1" action="back"/>
```

```
</center>
```

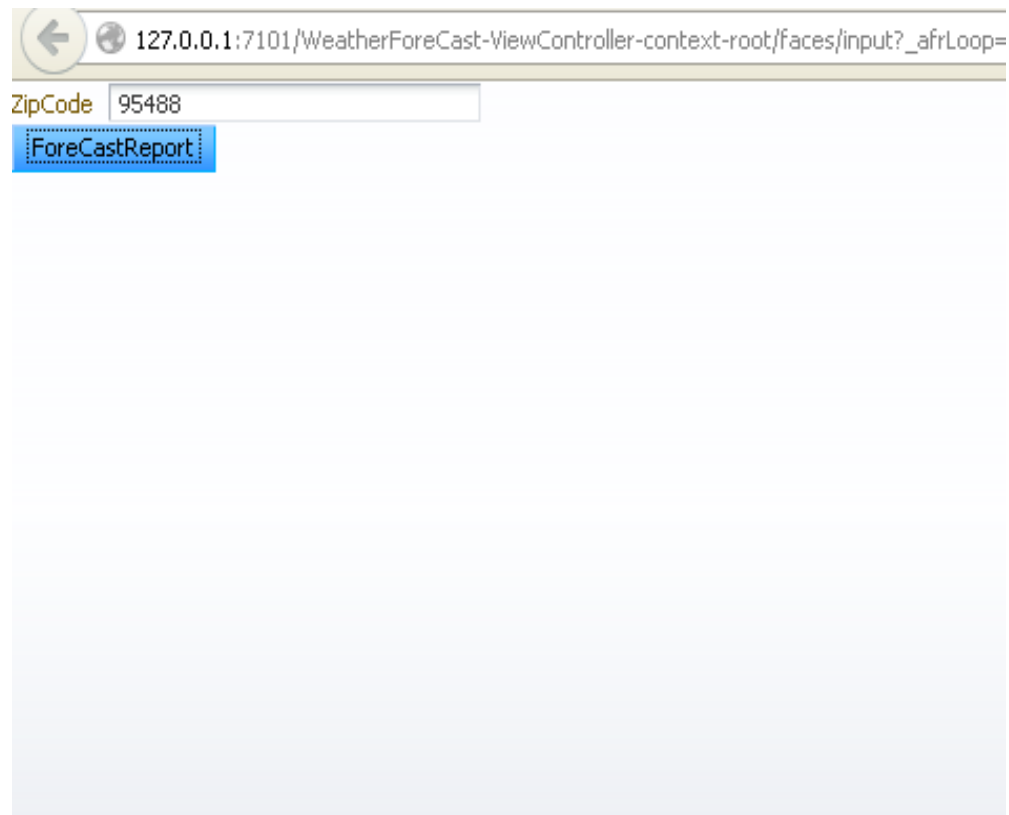
```
</af:form>
```



32. Drag and drop your three jspx in following format, and use control flow to connect these pages










OutPut:



127.0.0.1:7101/WeatherForeCast-ViewController-context-root/faces/input?\_afrcLoop=

ZipCode 95488

ForeCastReport

City	City	Date	Discription	High	Low	Image
Westport	Westport	2013-05-22	Sunny	55		
Westport	Westport	2013-05-23	Sunny	55	40	
Westport	Westport	2013-05-24	Partly Cloudy	57	45	
Westport	Westport	2013-05-25	Partly Cloudy	58	48	
Westport	Westport	2013-05-26	Partly Cloudy	56	49	
Westport	Westport	2013-05-27	Mostly Cloudy	58	47	
Westport	Westport	2013-05-28	Showers	58	48	

ZipCode

95489

ForeCastReport

**Your ZipCode Does Not Point Out Any City Pls Check Your ZipCode !....**

Back