

ADF Faces : View data from session EJB (Part1)

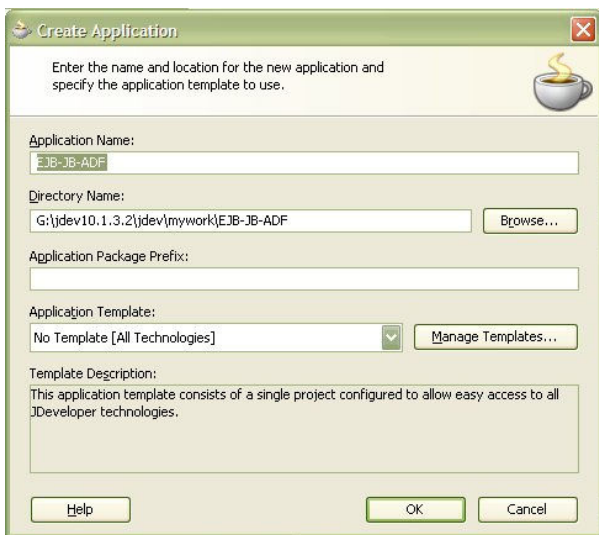
By: Dr. Ahmad Taufik Jamil, Pusat Teknologi Maklumat, HUKM

This tutorial will show you how to pass parameter value originated from a session EJB and view them in a JSF page. The tutorial will show how to get data from session EJB using JavaBean. The tutorials will show how to develop User Interface using ADF Faces.

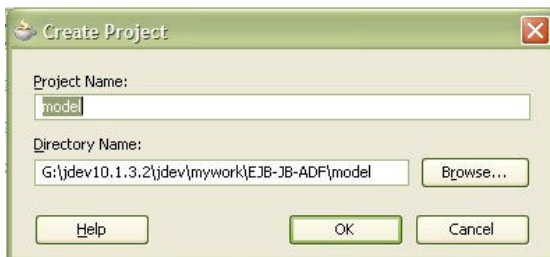
Part 1

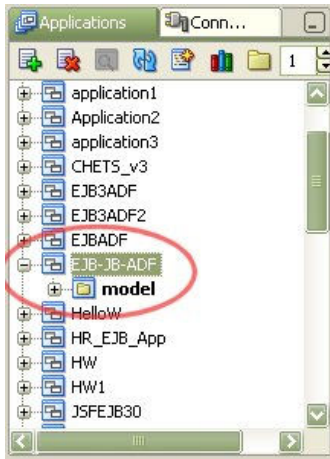
Using JavaBean (backing bean)

1. Creating new workspace and project folder.
 - a. Create a new workspace. Right click at Applications, click New Application... At *Create Application* dialog box, Enter **EJB-JB-ADF** for *Application Name:*, and **No Template [All Technologies]** for *Application Template:*, then click *OK*.



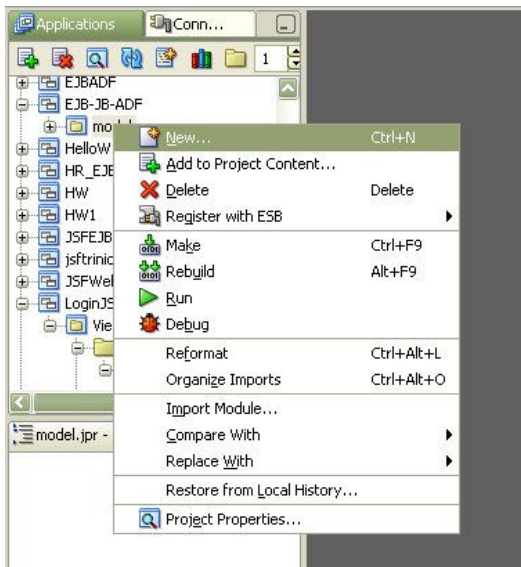
- b. Create a project folder. Enter **model** for *Project Name:*, then click *OK*.



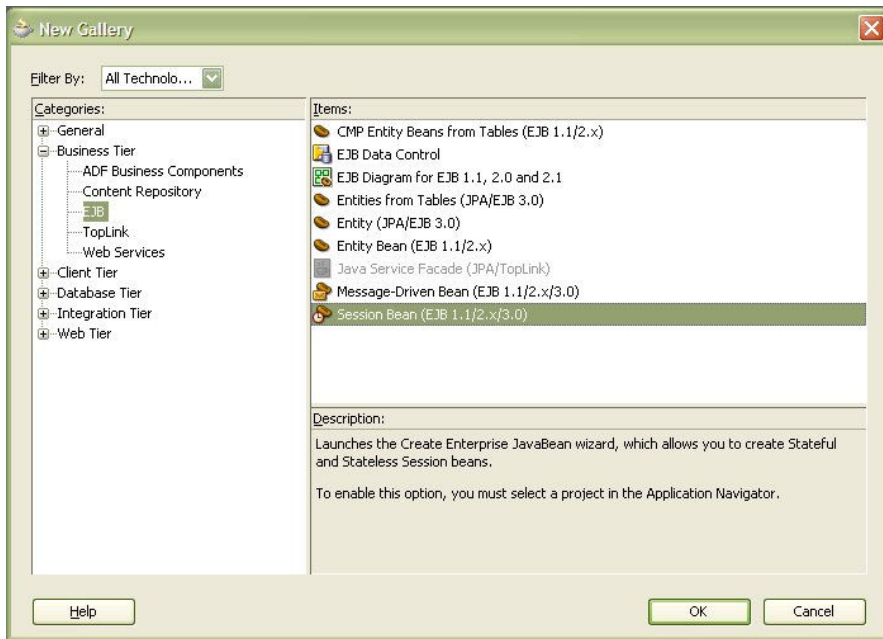


2. Creating a new session EJB

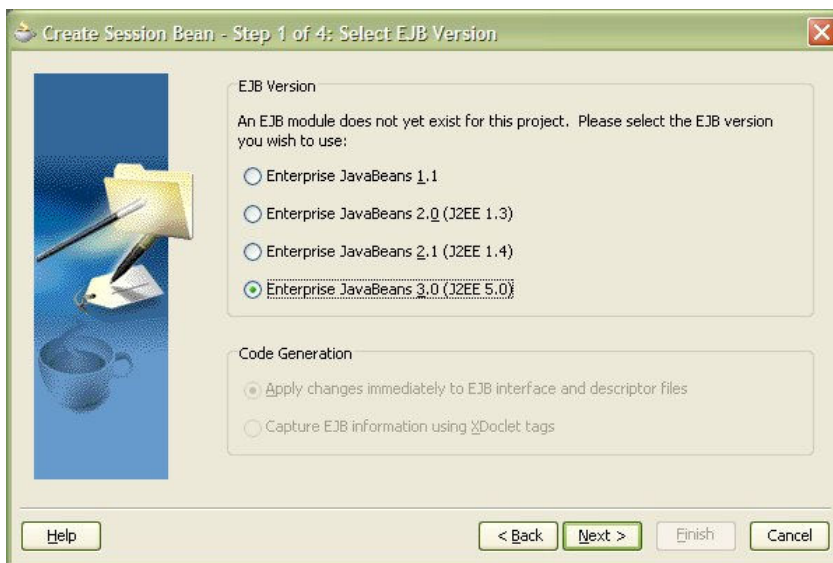
- a. Right click at project folder *model* and click *New...*



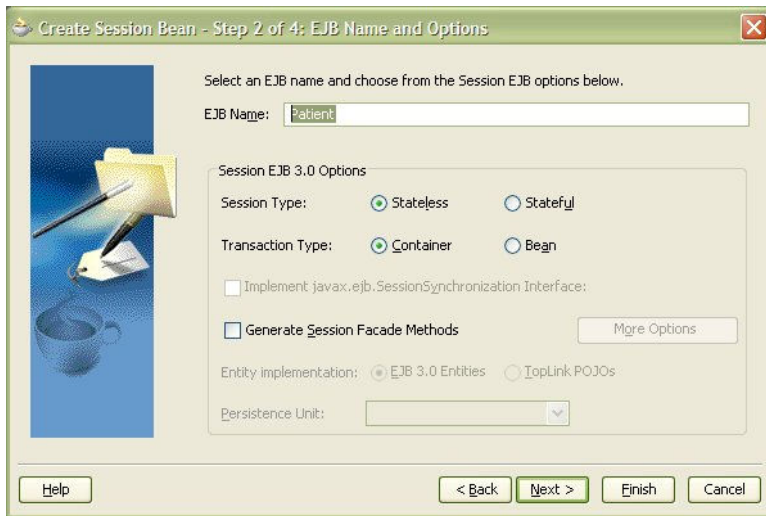
- b. At New Gallery dialog box, inside *Categories:*, under **Business Tier**, click **EJB**, at *Items* : click **Session EJB(EJB1.1/2.x/3.0)**, then click **OK**.



- c. At create session Bean wizard, click *Next*, at step 1 of 4, select **Enterprise JavaBean 3.0 (J2EE 5.0)**, then click *Next*.



- d. At step 2 of 4, enter **Patient** for *EJB Name:*, then click *Next*.



- e. Accept all default value at *step 3 of 4* and *4 of 4*, and click *Finish*.
- f. Now, source editor for ***PatientBean.java*** is seen. Add source code below, below constructor ***PatientBean()*** { } after the curly bracket (}).

```

public ArrayList viewPatient() {
    ArrayList patientList= new ArrayList(10);
    String[] patient1;
    patient1 = new String[]{"Ahmad Zam zam","0122345678","700628045678","Selangor"};
    String[] patient2;
    patient2 = new String[] {"Arin Sujak","0177745678","890628045671","Perlis"};
    String[] patient3;
    patient3 = new String[]{"Baharin Ujang","0195675678","880628045672","Pahang"};
    String[] patient4;
    patient4 = new String[]{"Zulhairi Hasan","0136345671","560628045674","Trengganu"};
    String[] patient5;
    patient5 = new String[] {"Hasan Kamal","0159345673","450628045633","Johor"};
    String[] patient6={"Mohamad Kamal","0188345632","670628045687","Sarawak"};

    patientList.add(patient1);
    patientList.add(patient2);
    patientList.add(patient3);
    patientList.add(patient4);
    patientList.add(patient5);
    patientList.add(patient6);

    return patientList;
}

```

```

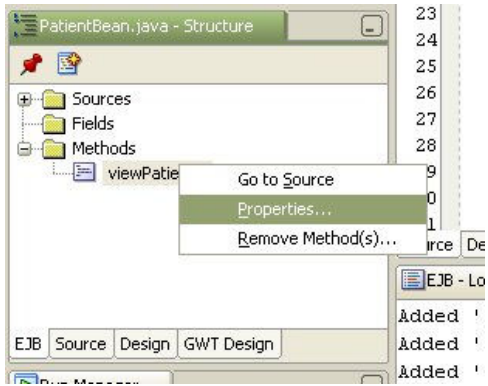
PatientBean.java PatientBean.java
7 @Stateless(name="Patient")
8 public class PatientBean implements Patient, PatientLocal {
9     public PatientBean() {
10    }
11
12    public ArrayList viewPatient(){
13        ArrayList patientList= new ArrayList(10);
14        String[] patient1;
15        patient1 = new String[]{"Ahmad Zam zam","0122345678","700628045678","Selangor"};
16        String[] patient2;
17        patient2 = new String[] {"Arin Sujak","0177745678","890628045671","Perlis"};
18        String[] patient3;
19        patient3 = new String[]{"Baharin Ujang","0195675678","880628045672","Pahang"};
20        String[] patient4;
21        patient4 = new String[]{"Zulhairi Hasan","0136345671","560628045674","Trengganu"};
22        String[] patient5;
23        patient5 = new String[] {"Hasan Kamal","0159345673","450628045633","Johor"};
24        String[] patient6={"Mohamad Kamal","0188345632","670628045687","Sarawak"};
25
26        patientList.add(patient1);
27        patientList.add(patient2);
28        patientList.add(patient3);
29        patientList.add(patient4);
30        patientList.add(patient5);
31        patientList.add(patient6);
32
33        return patientList;
34    }
35 }
36

```

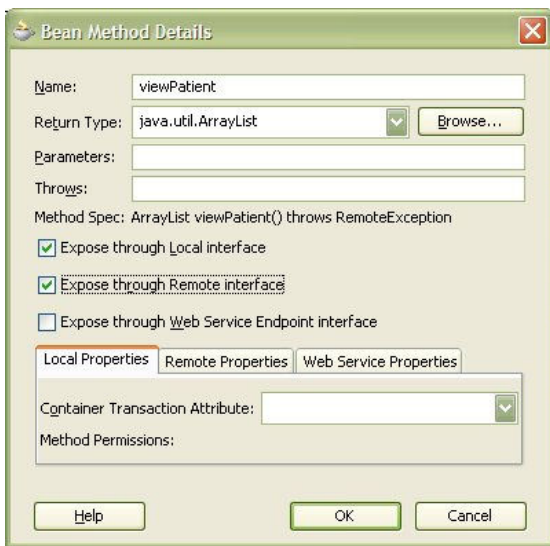
- g. Expand project folder *model*, until you see *PatientBean.java* and click to mark it. Under structure below it, you will see *viewPatient()* under folder *method*.



- h. Right click at *viewPatient()*, and click Properties...



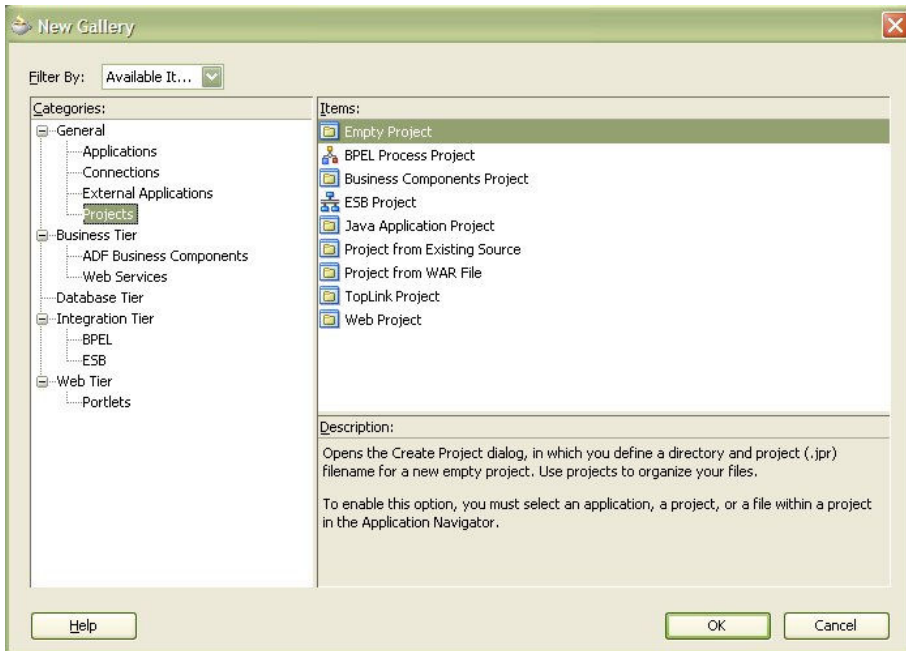
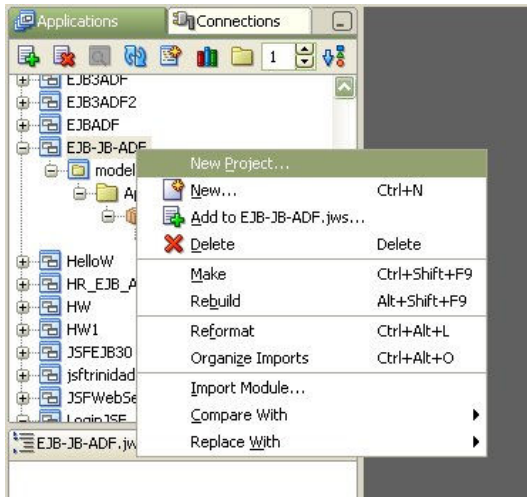
- i. At *Bean Method details* dialog box, check both *Expose through Local Interface* and *Expose through Remote Interface*, and click *OK*.



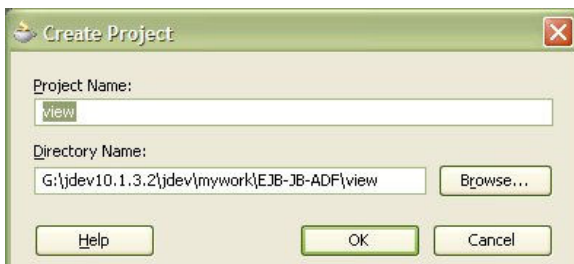
- j. Save all and compile (Make) file *PatientBean.java*

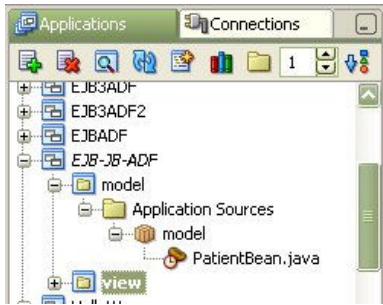
3. Creating new project folder for view layer

- a. Right click at *EJB-JB-ADF* workspace, click *New Project...* At *New Gallery* dialog box, at column *Categories:*, under *General*, select *Projects*, then click *Empty Project* at column *Items:* then click *OK*.

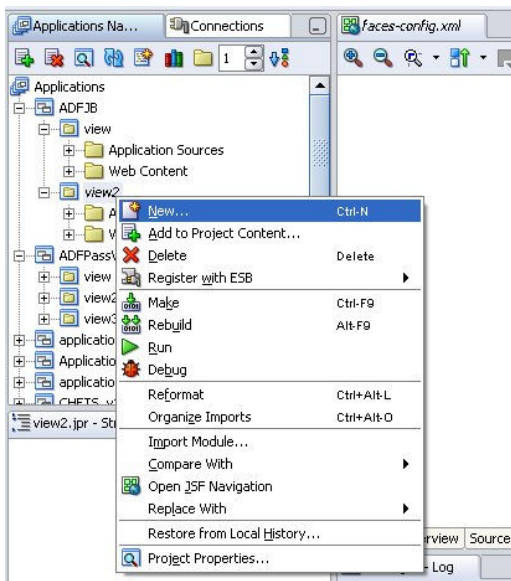


b. At *Create Project* dialog box, enter **view** for *Project Name:* and click *OK*. New **view** project folder is created.

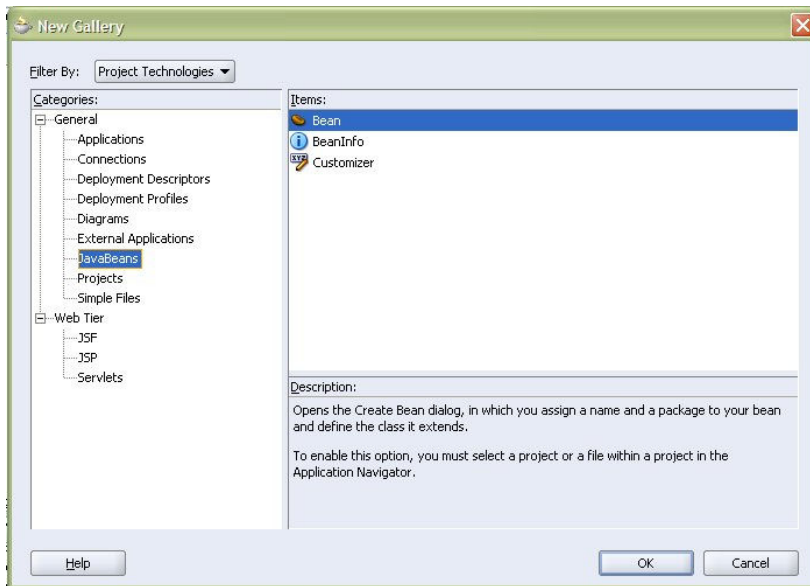




4. Creating new Managed Bean
 - a. Right click at *view* project and click *New...*



- b. At *New Gallery* dialog box, at *Categories:*, expand **General**, find and click **JavaBeans**. Inside *Items*, click **Bean** and then click **OK**.



- c. At *Create Bean* dialog box, enter **PatientJB** for *Name*., **view** for *Package*., and choose **java.lang.Object** for *Extends*., then click *OK*. Source editor for **PatientJB.java** is now opened.



- d. Add the following lines inside the **PatientJB** class, immediately below **PatientJBt** constructor (`public PatientJB() {`).

```
private String name;
private String telno;
private String icno;
private String state;

public PatientJB(String name, String telno, String icno, String state) {

    this.name=name;
    this.telno=telno;
    this.icno=icno;
    this.state=state;
}

public String getName(){
    return name;
}

public String getTelno(){
    return telno;
}

public String getIcno(){
    return icno;
}
```

```

}

public String getState(){
    return state;
}

```

The screenshot shows an IDE window with the following tabs: faces-config.xml, borang.jsp, Borang.java, and PatientJB.java. The PatientJB.java file is open and displays the following code:

```

1 package view;
2
3 public class PatientJB {
4     public PatientJB() {
5     }
6
7     private String name;
8     private String telno;
9     private String icno;
10    private String state;
11
12    public PatientJB(String name, String telno, String icno, String state) {
13
14        this.name=name;
15        this.telno=telno;
16        this.icno=icno;
17        this.state=state;
18    }
19
20    public String getName(){
21        return name;
22    }
23
24    public String getTelno(){
25        return telno;
26    }
27
28    public String getIcno(){
29        return icno;
30    }
31
32    public String getState(){
33        return state;
34    }
35

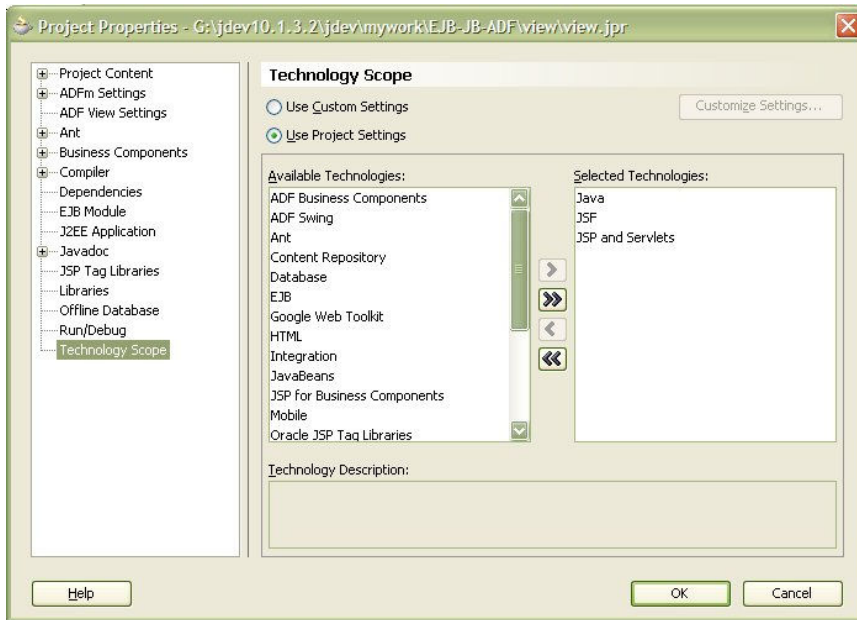
```

The IDE interface includes a 'Source' tab and a 'Design' tab, with a 'GWT Design' and 'History' panel visible at the bottom.

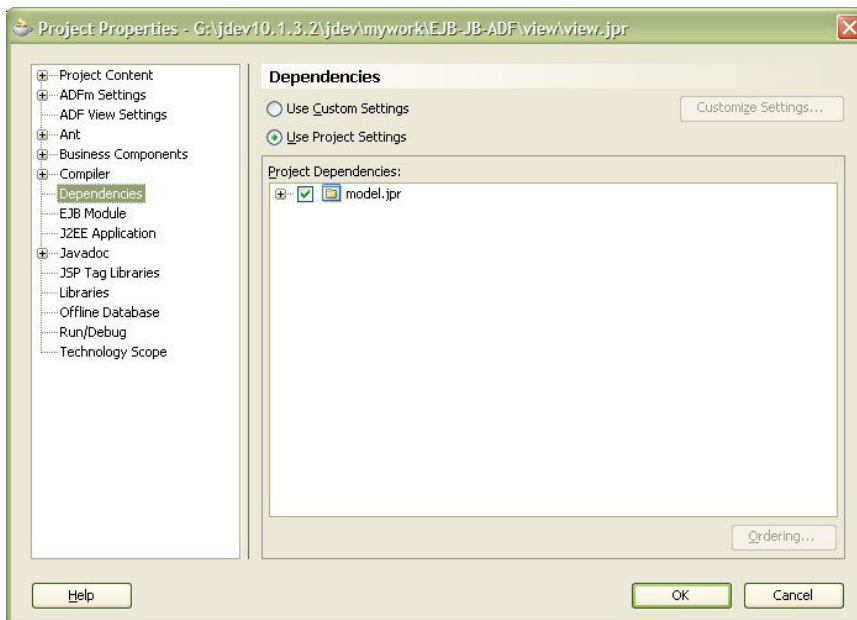
e. Save all and compile(Make).

5. Creating page navigation using face-config.xml.

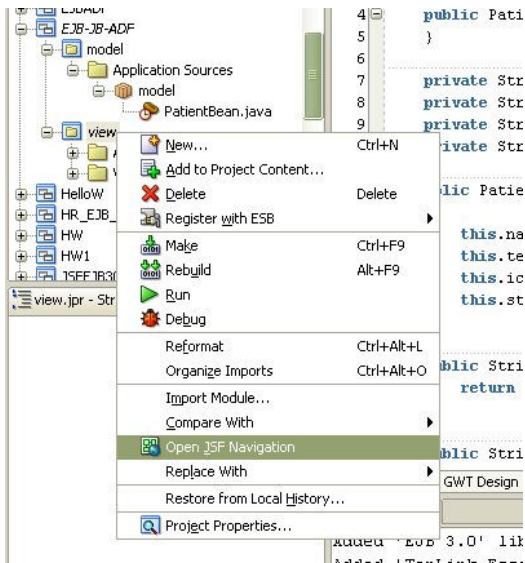
- a. Right click at **view** project and click *Project Properties....* In *Project Properties* dialog box, at left column; click **Technology Scope**. Find and click **JSF** at *Available Technologies* and move to right column (*Selected Technologies*)-**Java, JSP and Servlets** will be moved as well.



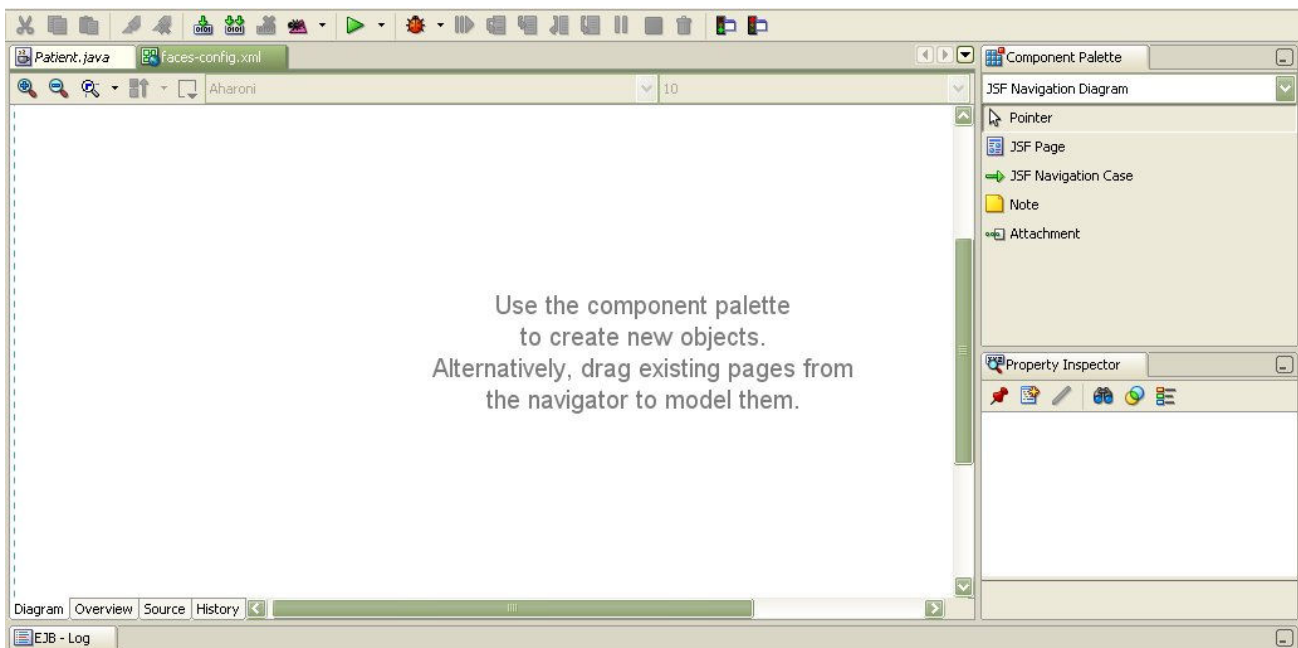
- b. At the same dialog box, click at **Dependencies** (at left column) and check **model.jpr** (at right column), then click **OK**.



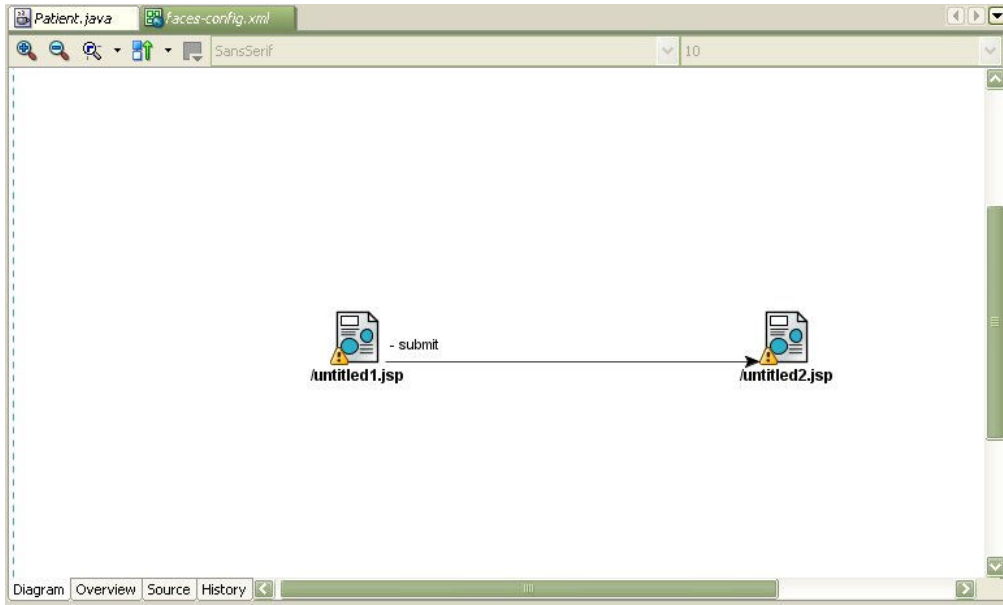
- c. Right click at **view** project, and click **Open JSF navigation**.



d. File *faces-config.xml* is opened in Diagram view.



d. Create new page navigation. Click, drag & drop **JSF Page** from *Component Palette*, onto the diagram view of *faces-config.xml*, and then drop another **JSF page** beside it. Click **JSF Navigation Case** from *Component Palette*, then click at */untitled1.jsp* and then click at */untitled2.jsp*. An arrow is created and pointing to */untitled2.jsp*. Change – **success** to – **submit**.



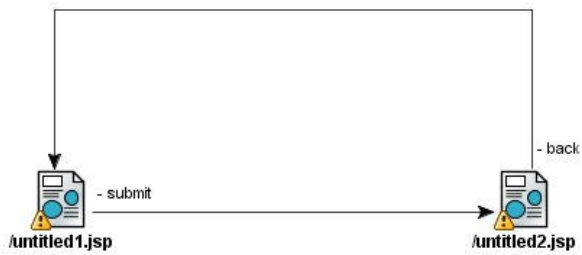
e. Create another JSF Navigation from */untitled2.jsp* to */untitled1.jsp*. Rename – **success** to – **back**.



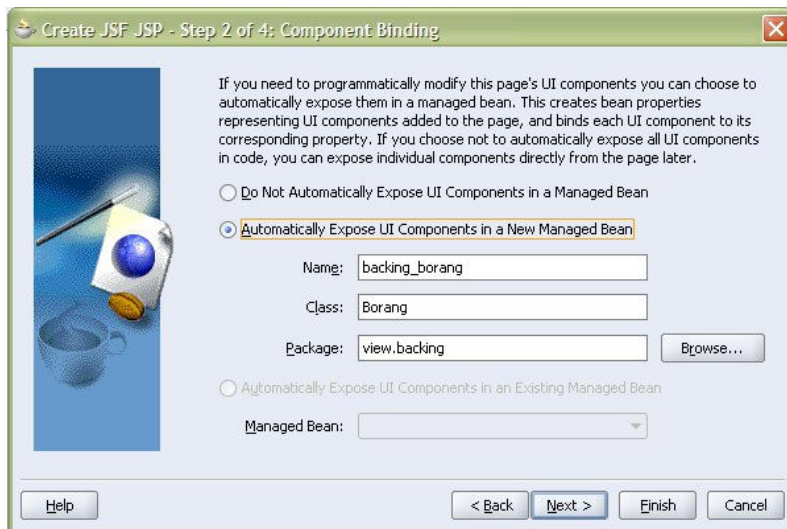
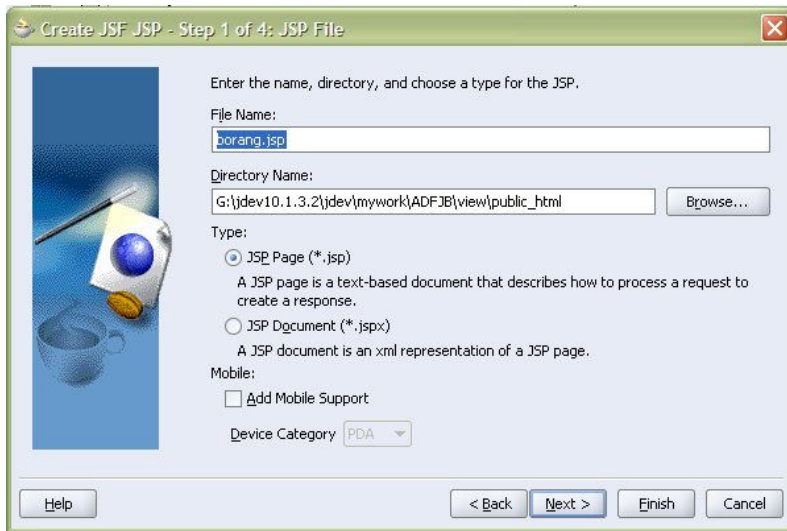
f. Click Save All

6. Creating JSF form and backing beans.

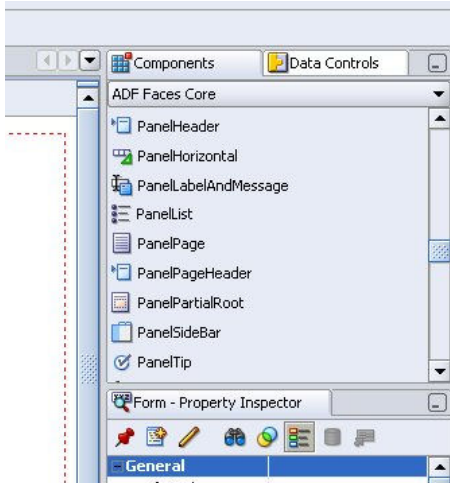
a. Go back to *faces-config.xml* Diagram view.



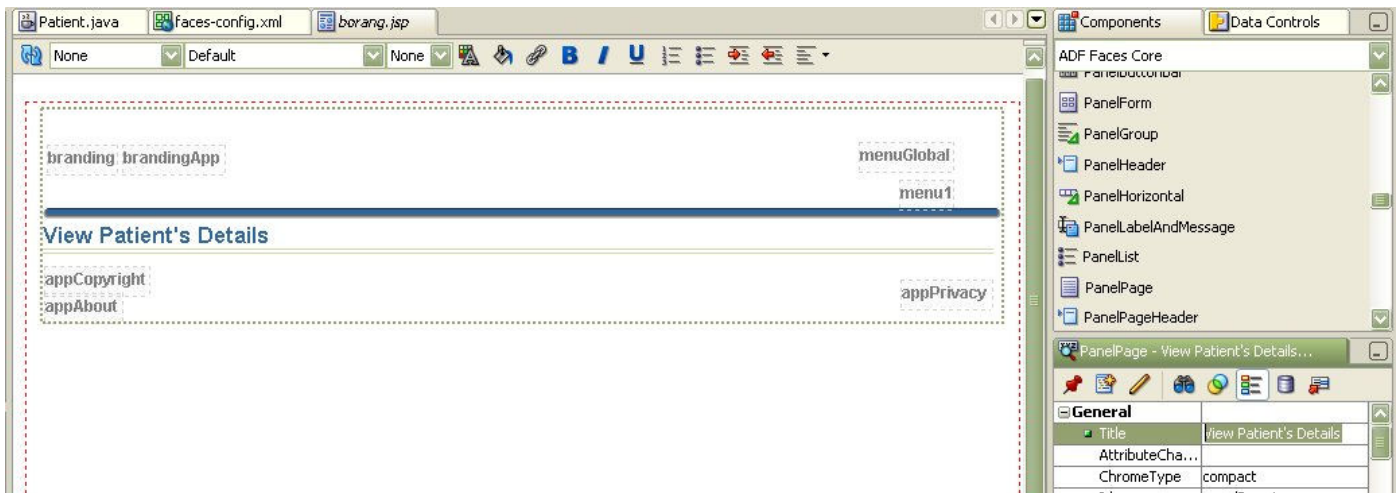
- b. Double click icon */untitled1.jsp*. At *Create JSF JSP* wizard click *Next*. At Step 1 of 4, enter **borang.jsp** for *File name*:. Click *Next*. Choose ***Automatically Expose UI Components in a New managed Beans***. Click *Finish*.



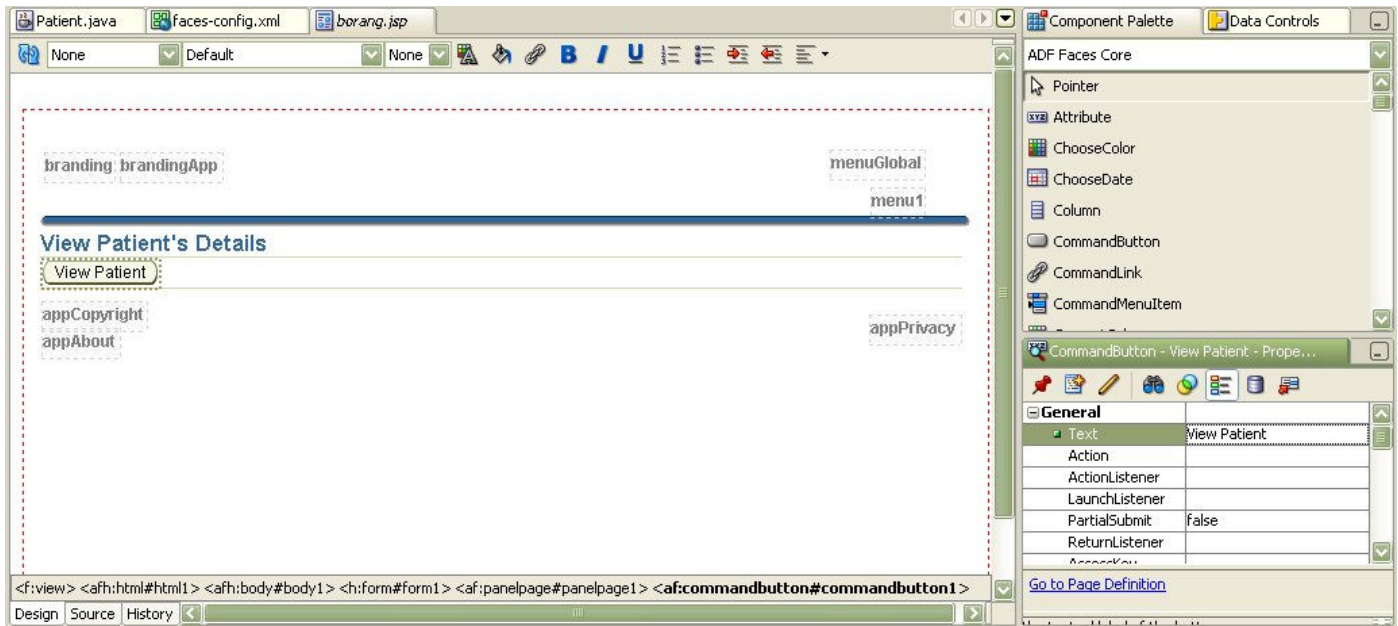
- c. At the *borang.jsp* design area, click, drag and drop ***PanelPage*** from *Components Palette*, to *borang.jsp* design area.



d. Using *Property Inspector*, change Title to **View Patient's Details**. Now the title of the page has changed.



e. Click, drag & drop **CommandButton** from *Component Palette* to the **panelPage**, and change the *Text* value in *Property Inspector* from **CommandButton 1** to **View Patient** and press Enter.



7. Editing backing bean (*Borang.java*).

- a. Double click button *View Patient*, *Bind Action Property* dialog box will appear, accept all the default value and click *OK*.



- b. You will see backing bean *Borang.java* is opened in the editor. Add the following lines to *Borang.java*, after the variable declaration (after this line `private CoreCommandButton commandButton1;`).

```
//Declare new variable for patient
String namex=null;
String telnox=null;
String icnox=null;
String statex=null;
ArrayList <PatientJB> patientjlist = new ArrayList <PatientJB>();

//Declare new constructor to connect to EJB using initialcontext and get values for patient
public Borang(){
    try {

        final Context context = getInitialContext();
        Patient patient = (Patient)context.lookup("Patient");
        ArrayList patientList= new ArrayList(10);
        patientList = patient.viewPatient();

        for(int i=0;i<patientList.size();i++){
            String[] s = (String[])patientList.get(i);
            namex=s[0];
            telnox=s[1];
            icnox=s[2];
            statex=s[3];
        }
    }
}
```



```

        patientjlist.add(new PatientJB(name, telno, icno, state));
    }

    } catch (Exception ex) {
        ex.printStackTrace();
    }

    }

//Method for initial context
private static Context getInitialContext() throws NamingException {
    // Get InitialContext for Embedded OC4J
    // The embedded server must be running for lookups to succeed.
    return new InitialContext();
}

//method to return value to JSF page
public ArrayList <PatientJB> getPatientjlist(){

    return patientjlist;

}

```

```

18 import oracle.adf.view.faces.component.html.HtmlHtml;
19
20 import view.PatientJB;
21
22 public class Borang {
23
24     private HtmlHtml html1;
25     private HtmlHead head1;
26     private HtmlBody body1;
27     private HtmlForm form1;
28     private CorePanelPage panelPagel;
29     private CoreCommandButton commandButton1;
30     //Declare new variable for patient
31     String name=null;
32     String telno=null;
33     String icno=null;
34     String state=null;
35     ArrayList<PatientJB> patientjlist = new ArrayList<PatientJB> ();
36
37     public Borang(){
38         try {
39
40
41             final Context context = getInitialContext();
42             Patient patient = (Patient)context.lookup("Patient");
43             ArrayList patientList= new ArrayList(10);
44             patientList = patient.viewPatient();
45
46             for(int i=0;i<patientList.size();i++){
47                 String[] s = (String[])patientList.get(i);
48                 name=s[0];
49                 telno=s[1];
50                 icno=s[2];
51             }
52         }
53     }
54 }

```

- c. Press ALT+ENTER few times if necessary to insert line import below, or you can manually add the line below to file **Borang.java**.

```
import view.PatientJB;
import java.util.ArrayList;

import javax.faces.component.html.HtmlForm;

import javax.naming.Context;

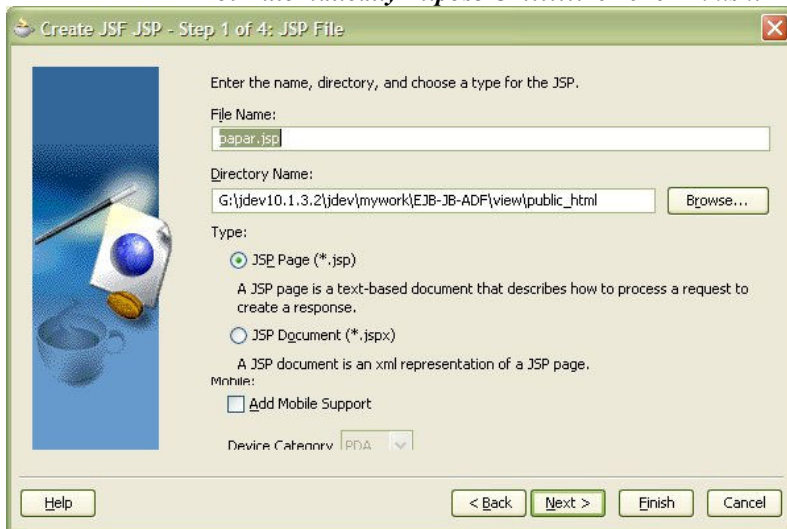
import javax.naming.InitialContext;
import javax.naming.NamingException;

import model.Patient;
```

- d. Find method `public String commandButton1_action()`, in file **Borang.java**, and change the line `return null;` to `return "submit";`
e. Save All and compile (Make).

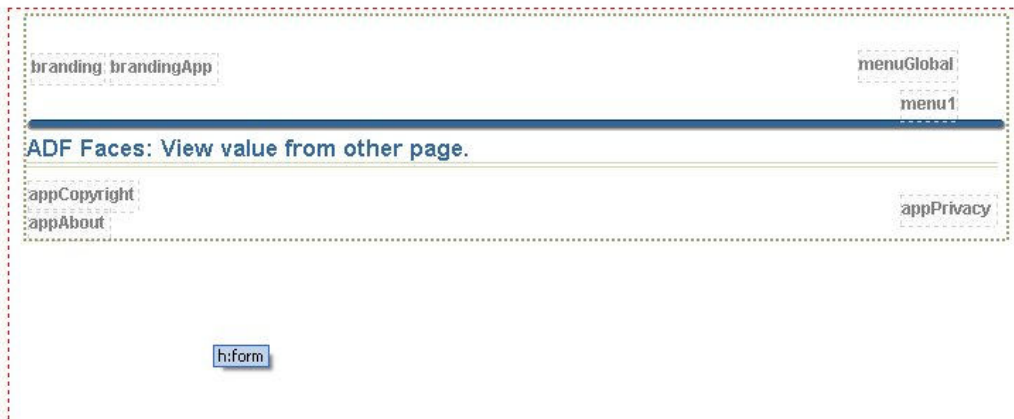
8. Creating JSF page to view data.

- a. Go back to diagram view of **faces-config.xml**. Double click icon **/untitled2.jsp**
b. On **Create JSF JSP wizard** click **Next**, at Step 1 of 4, change file name to **papar.jsp** and click **Next**. Choose **Do Not Automatically Expose UI.....** then click **Finish**.

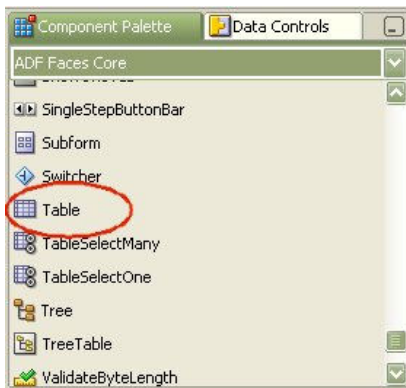




- c. At Design view for file *papar.jsp*, click, drag & drop **PanelPage** from *Components Palette* tab to design area, and change the title to **ADF Faces: View value from other page.**



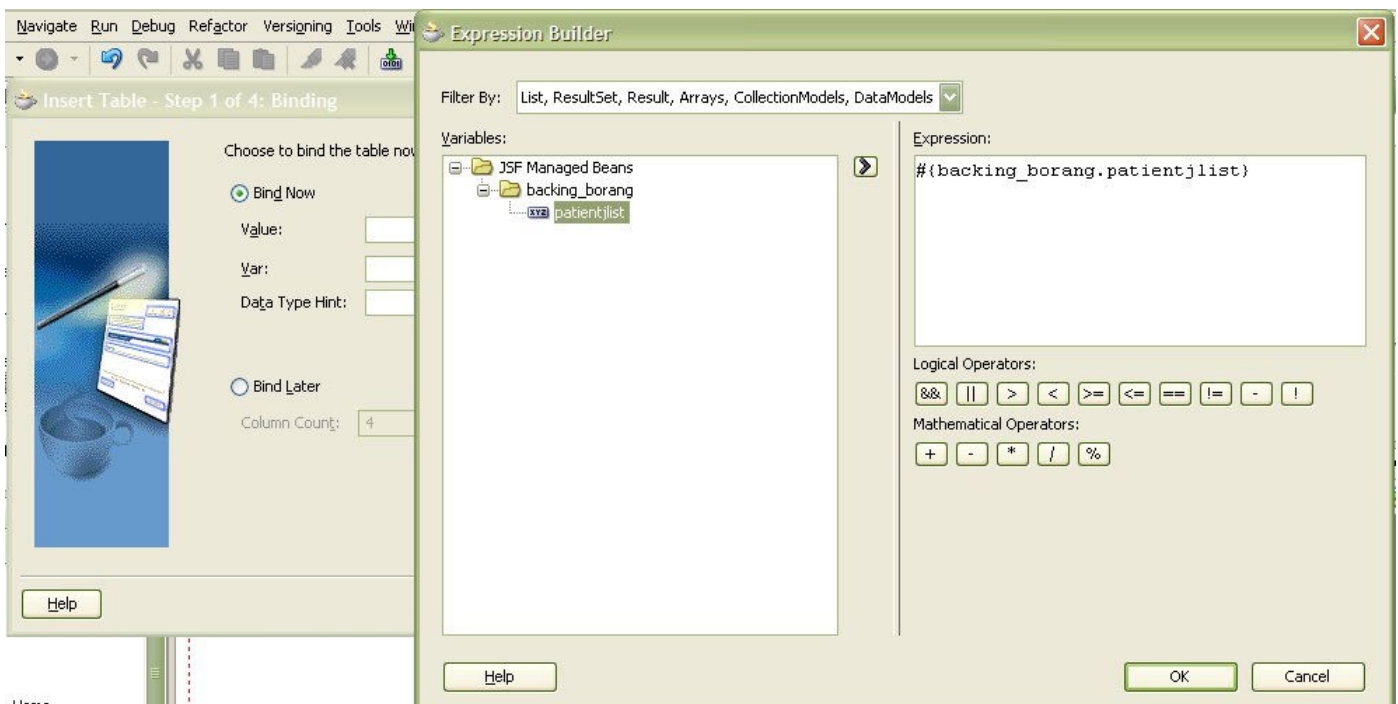
- d. Find component **Table** from *Component Palette*, click, and drag & drop Table to *papar.jsp* (inside *panelPage*).



- e. Click *Next* on the welcome page of *Create Table* wizard.



f. At Step 1 of 4, choose **Bind Now**, and click button *Bind...*

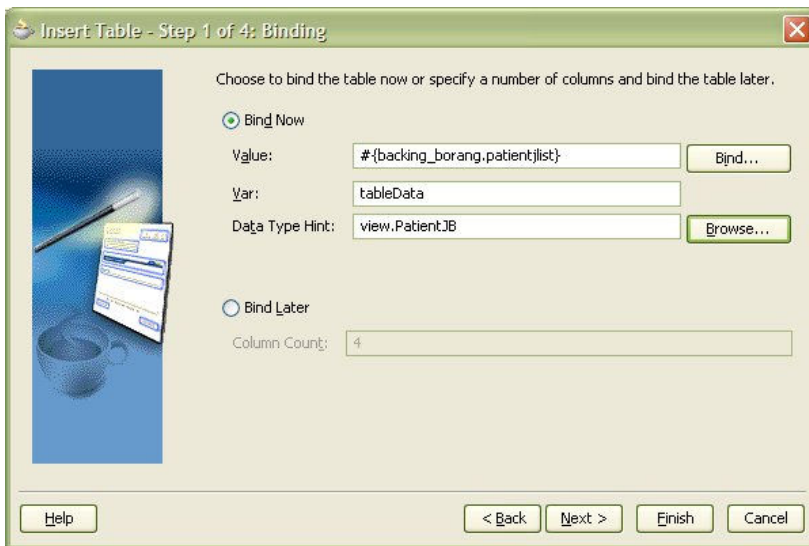


g. At *Expression Builder*, under *backing_borang* and *JSF managed Beans*, click **patientjlist**, and click button > to move it to the right column. Then click *OK*.

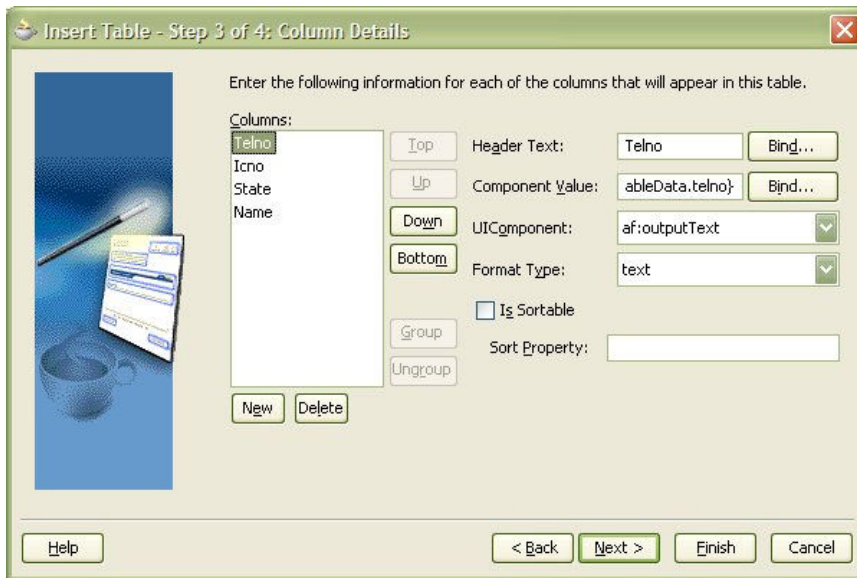
h. Enter **tableData** for field *Var:*, and click button *Browse...*



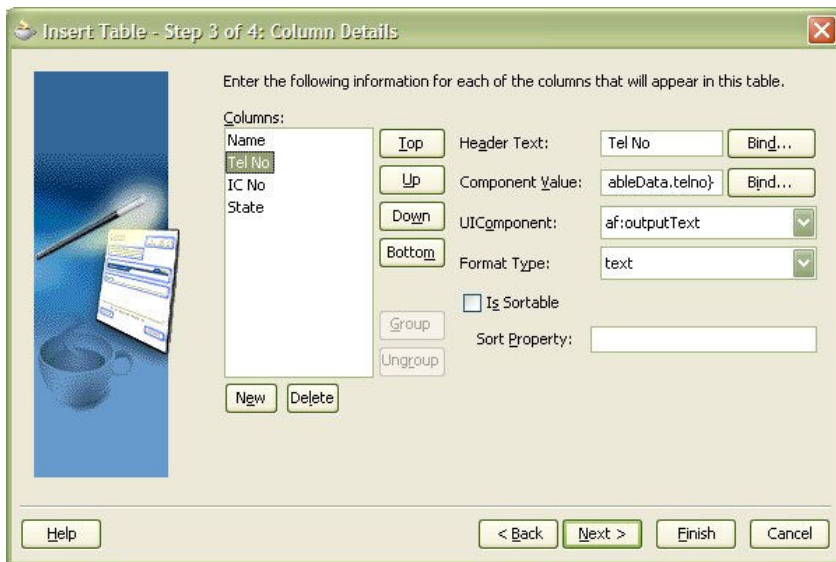
- i. At Class Browser, at *Hierarchy* tab, expand *view* and click *PatientJB* and click *OK*.



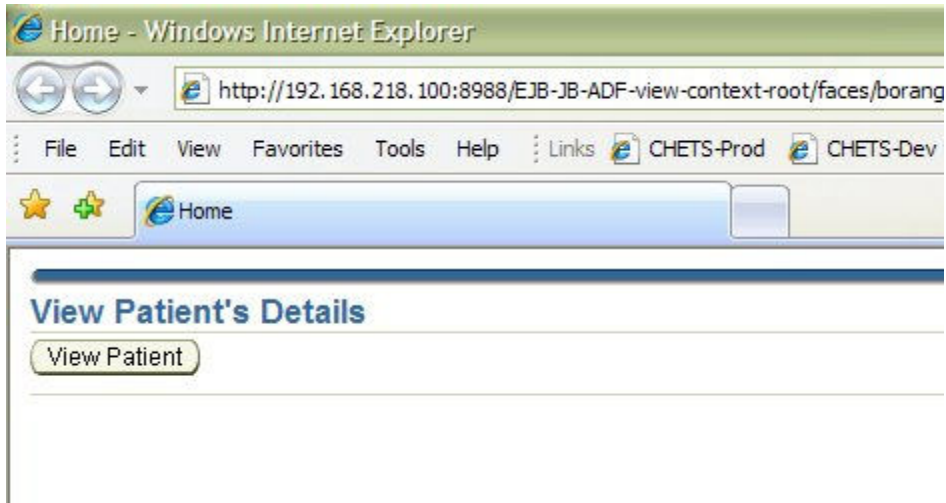
- j. Click Next at Step 1 of 4, then click Next at Step 2 of 4.



- k. At Step 3 of 4, click **Name** inside *Columns:*, and click button *Top* to move name to the top column. Change *Header Text:* to **Name**, **Tel No**, **IC No** and **State** for **Name**, **Telno**, **Icno** and **State** respectively, then click *Next*.



- l. At Step 4 of 4, click *Finish*.
 m. Click Save All and compile(Make).
 n. Run file **borang.jsp**



- o. Click button View Patient. You will see the result below.

