Multiple Forms from Single JSPDyn Page

Applies to:

SAP EP 6.0 and above. For more information, visit the Portal and Collaboration homepage.

Summary

This article is about the utility of the JSPDynpage when multiple forms are required in the application.

Author: Atul Agrawal

Company: Keane, an NTT Data Company

Created on: 01 July 2011

Author Bio

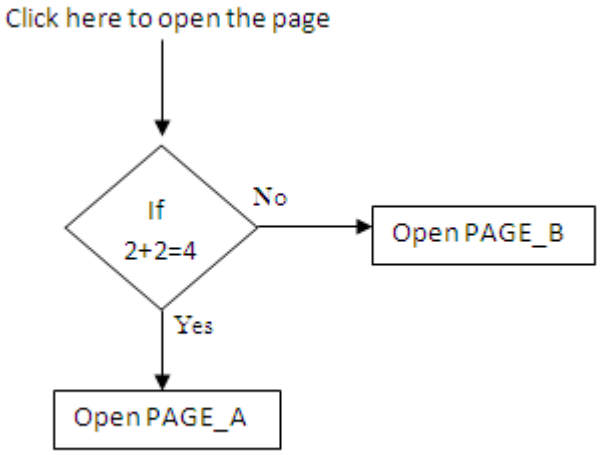
Atul Agrawal is a SAP EP consultant with Keane, an NTT Data Company, Gurgaon. He is Proficient on SAP Netweaver Portal in developing Custom Java iViews, System Administration, Content Administration, User Management, NWDS based PDK Development, Customization of portal themes, desktop rules and master role collection, customization of standard Portal components like Logon PAR, Mast Head and Top level navigation. Atul has the experience in Integrating BW, Web Applications and BO reports using Appinegrator services of SAP Portal. He is proficient in web technologies ExtJS 2.3, JQuery and Blue print CSS and also holds experience in end to end implementation of external facing Portal and integrating it with SAP BW via SAP Logon ticket based mechanism.

Table of Contents

Introduction

The problem can be best understood with the following example.

We have a link on a page. Click on this link will open PAGE_A or PAGE_B depending on some condition.



This can be achieved by the following two solutions

Solution 1

The simplest and obvious approach-

1. Put conditional logic before generating the link to click.
2. As per the condition, required link for the respective page will be displayed.
3. Each page would be a JSPDynpage.
4. Create separate iviews for each page.

The problem is handled by putting the logic before generating the link.

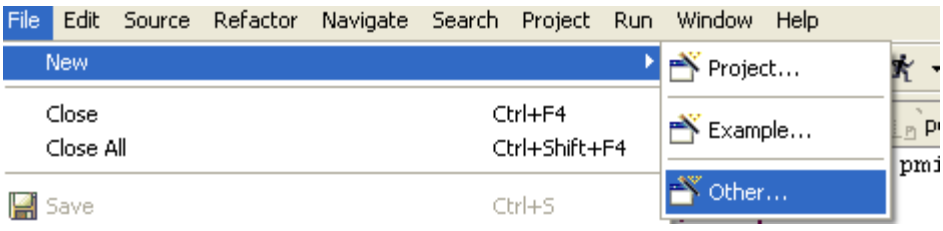
Solution 2

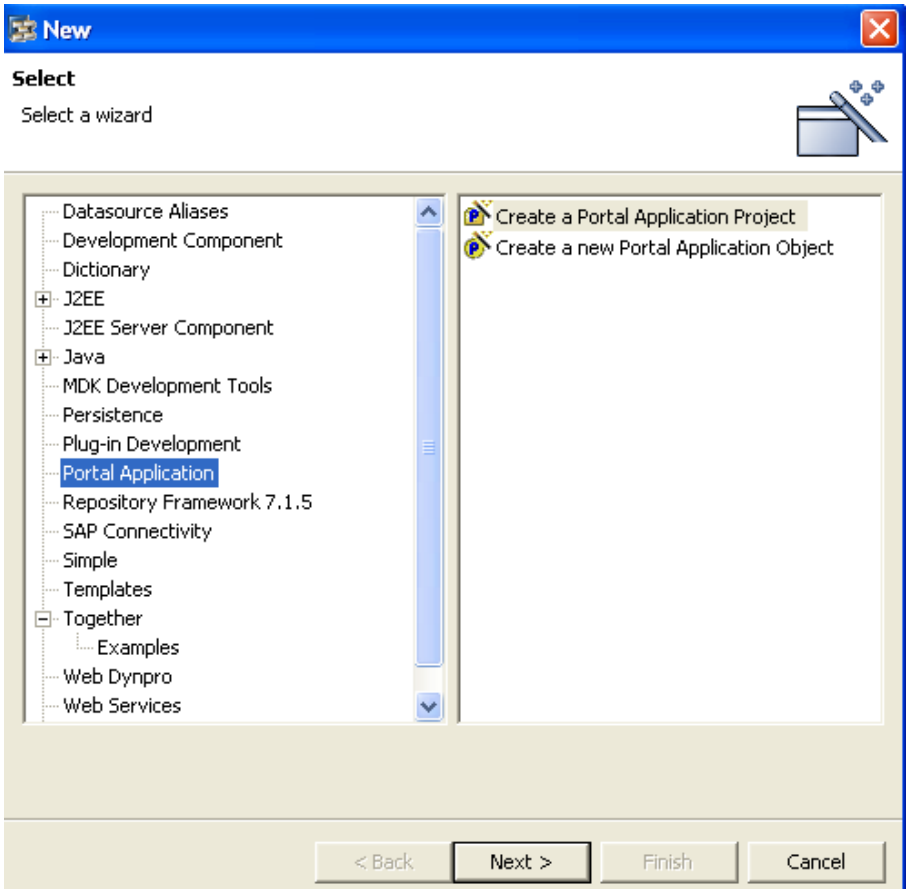
The better solution would be- if the creation of new iview can be avoided and the same par file can be used to render two pages. Now let's see the step by step process to achieve the same.

Create New Project

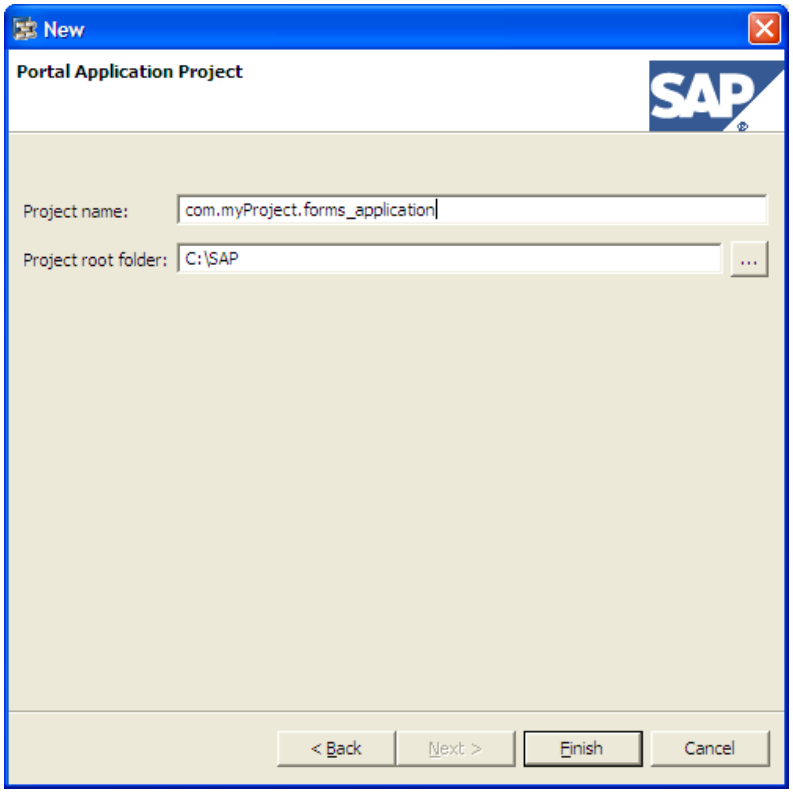
Start with creating a new project.

Open NWDS and go to File -> New ->Other





Select Portal Application > Create a Portal Application Project and click next.

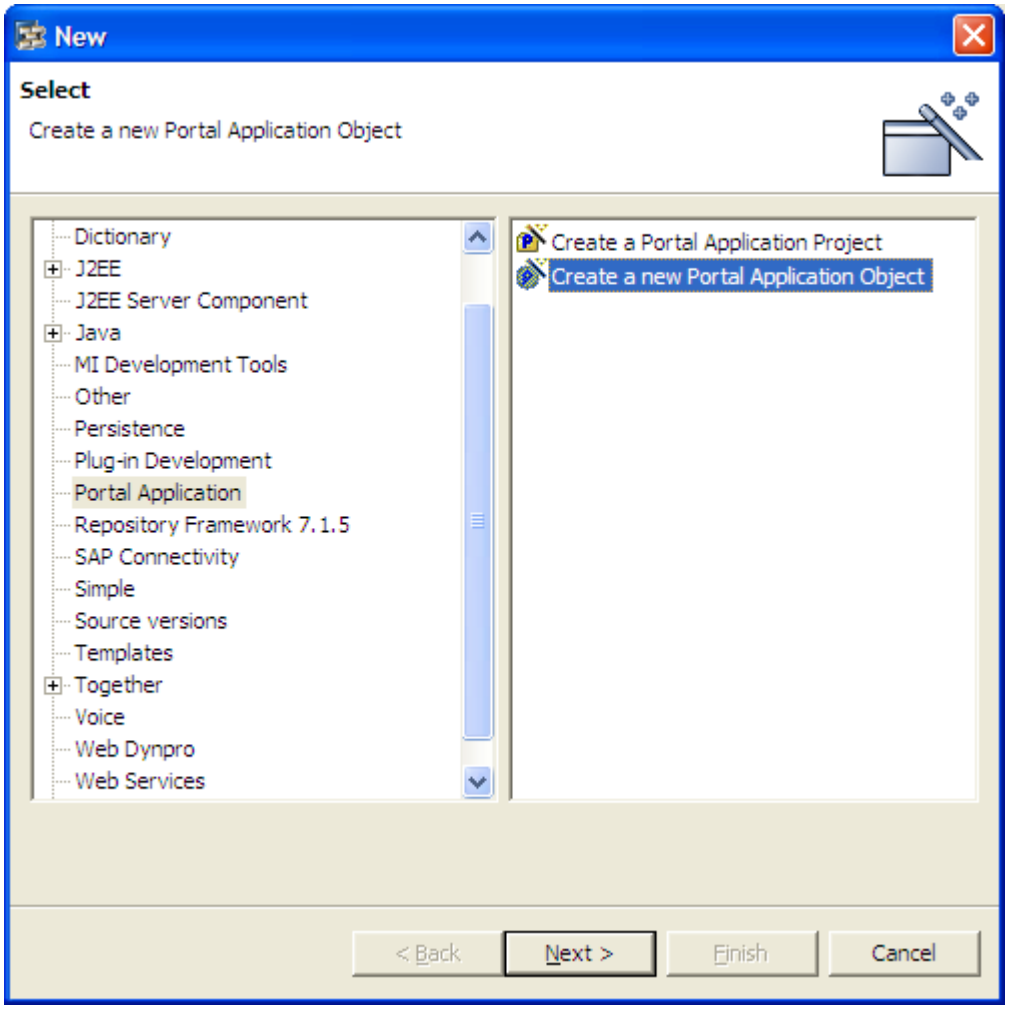


Enter the desired Project name and click finish. Here we will be using com.myProject.forms_application.

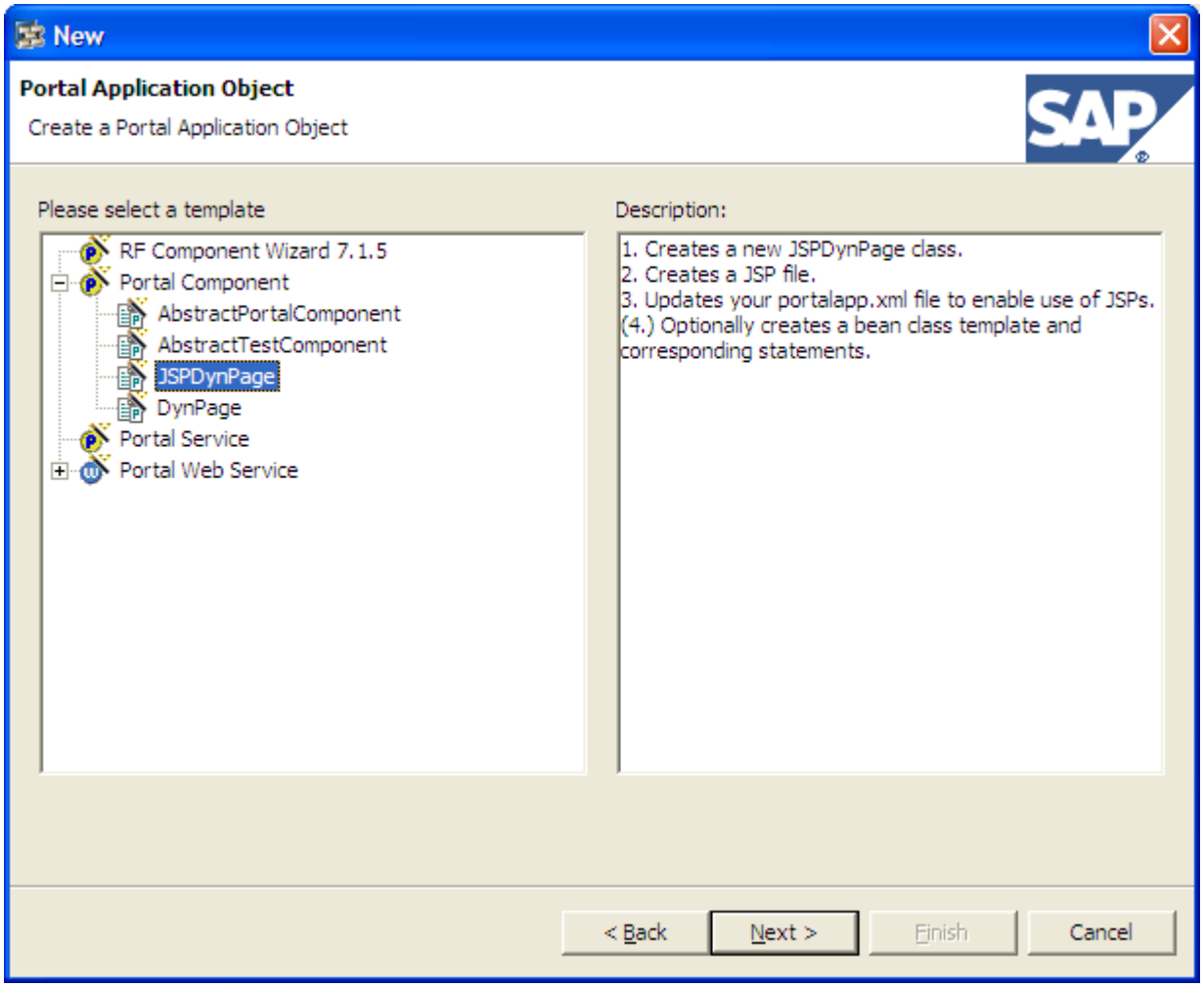
Create Portal Object

Next we need to create the portal object.

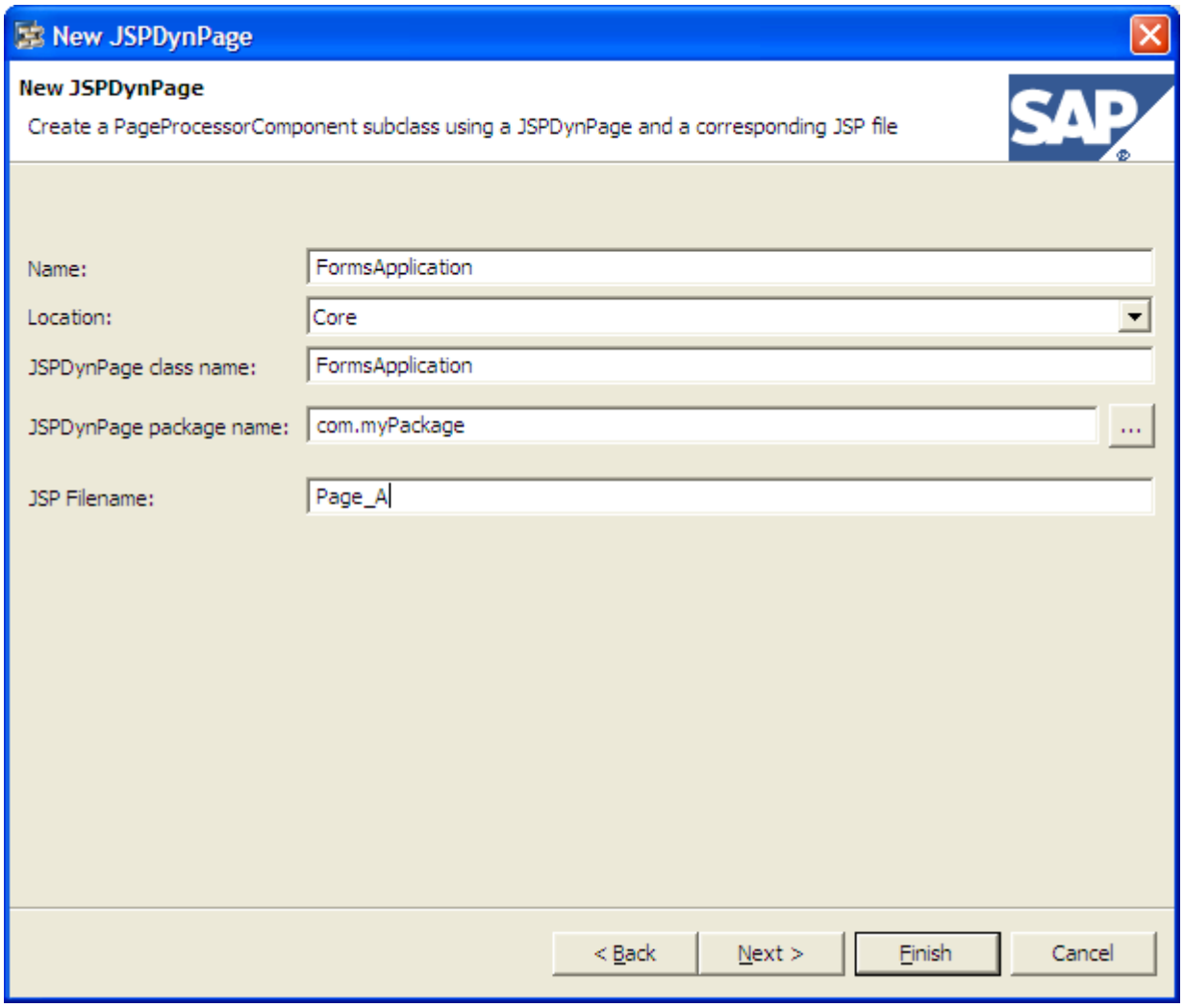
Right Click on the Project and choose-> New -> Other -> Create a new Portal Application Object



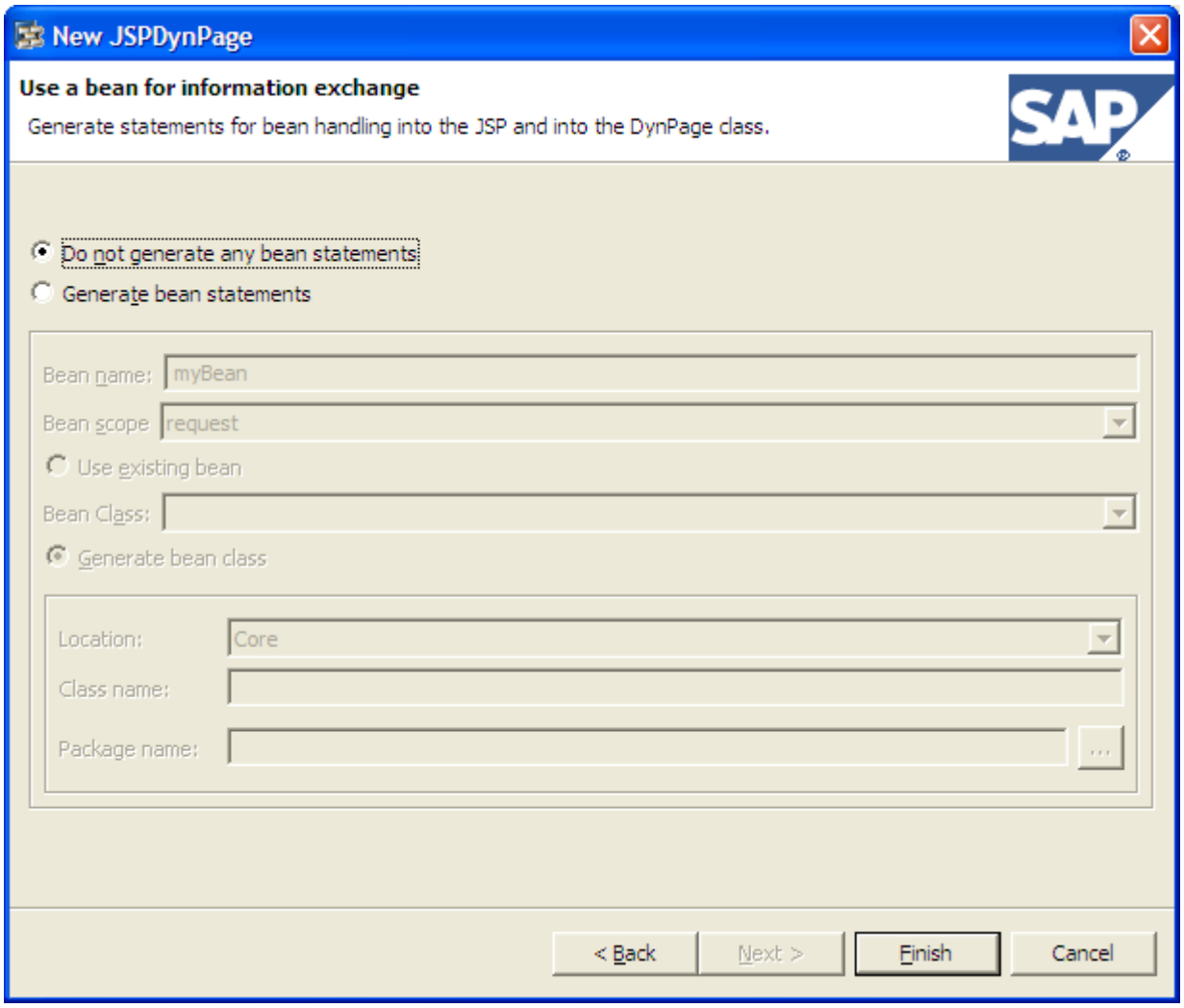
Click Next to choose your par file com.myProject.forms_application. Click next again to see the following



Choose JSPDynPage and click next



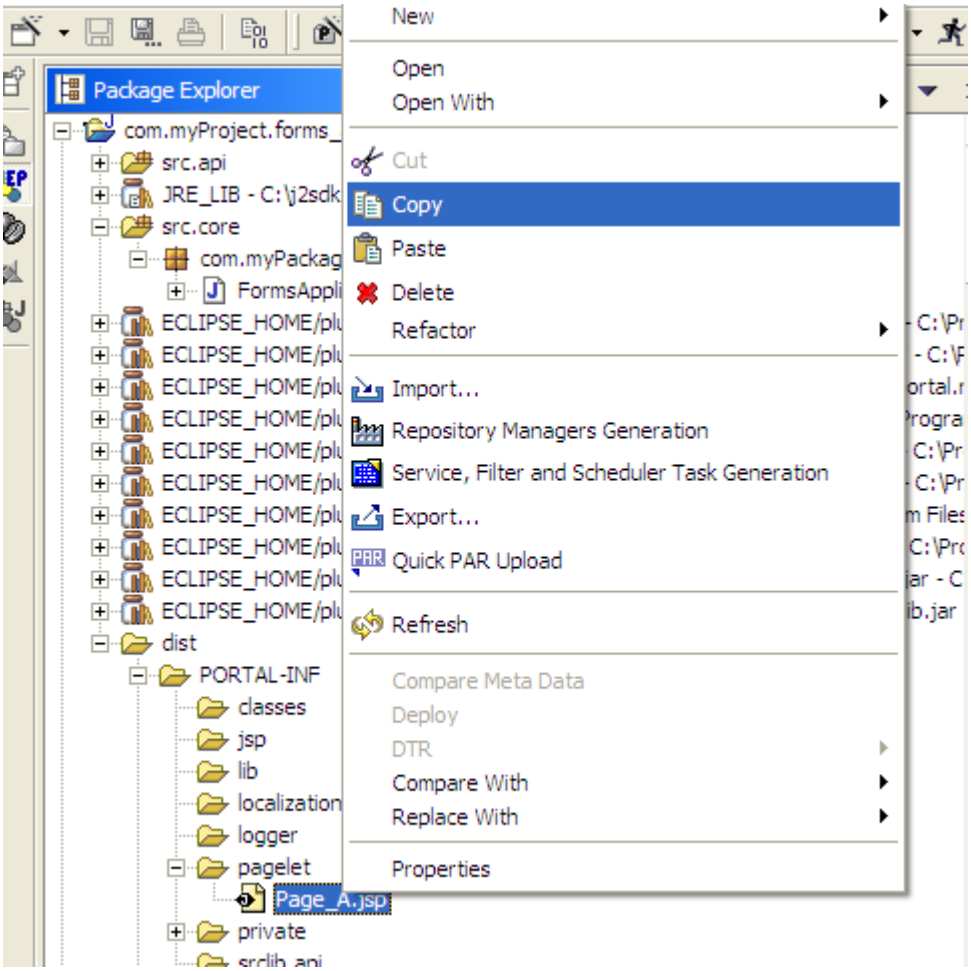
Fill in the desired class name and JSP file name and choose next.



Choose do not generate any bean statement and click finish.

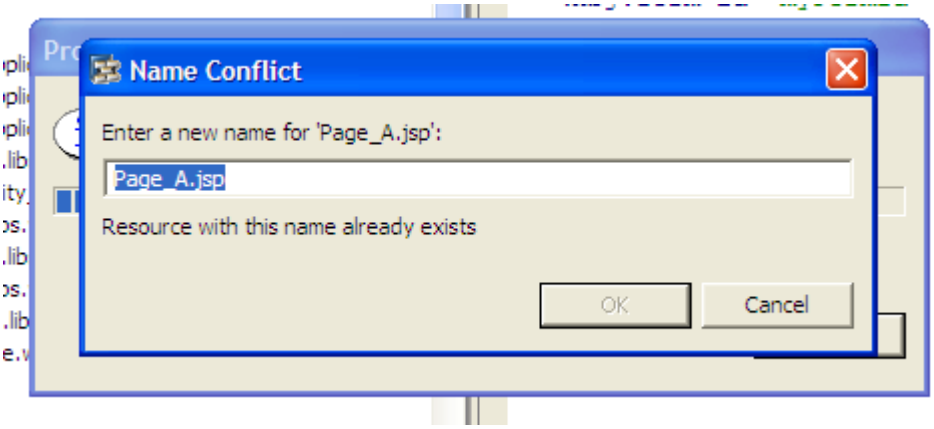
By now you have successfully created a new project and a portal object inside the project, which shall need to be modified.

Multiple Jsp files



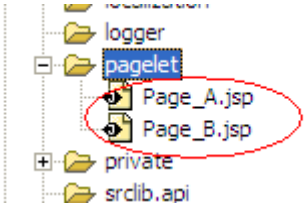
Navigate to dist > PORTAL-INF > pagelet > Page_A.jsp in NWDS, right click and copy the jsp file.

Right click on pagelet and paste the copied file. The following message would be seen.



Change the name to Page_B.jsp and press OK.

Now you have two jsp pages



Open Page_A.jsp and put in the sample code.

<HTML>

<head>

</head>

<body>

<B>**This is the First Page*****</B>***

</body>

</HTML>

Open Page_B.jsp and put in the sample code.

<HTML>

<head>

</head>

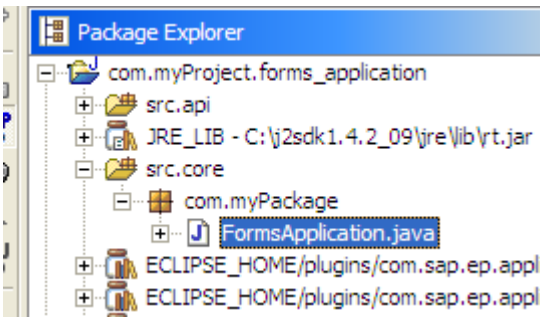
<body>

<B>**This is the Second Page*****</B>***

</body>

</HTML>

Java File

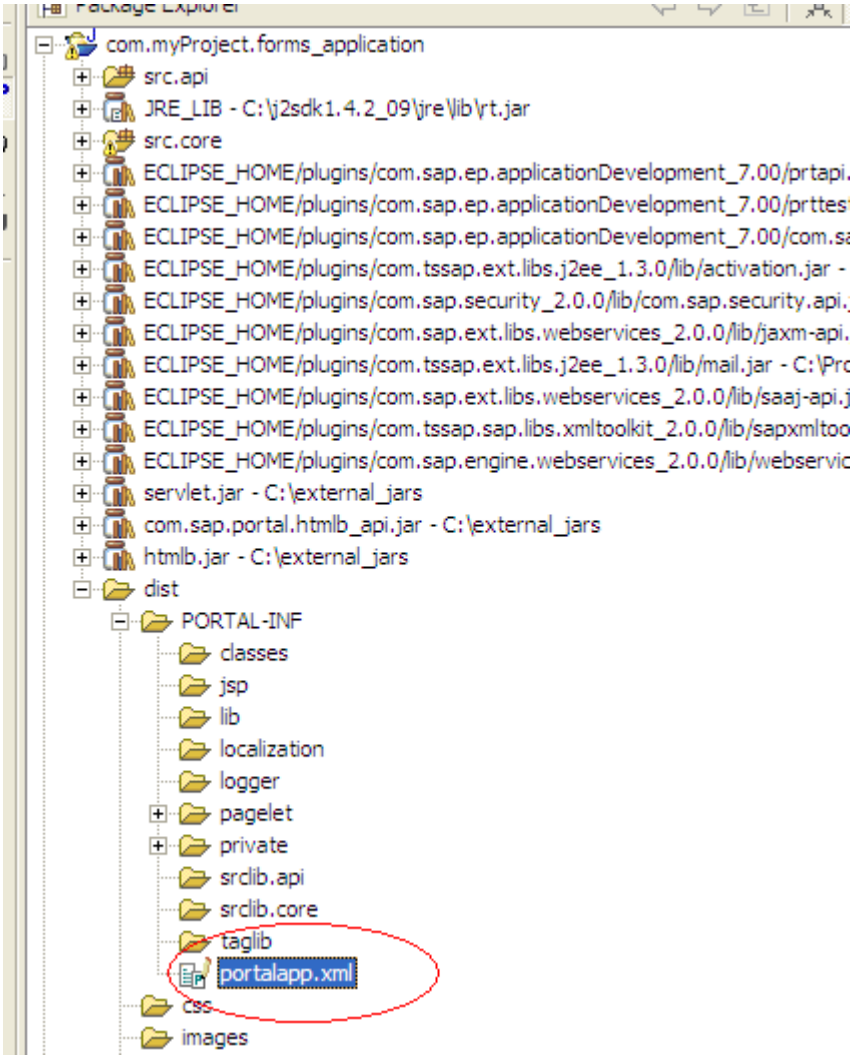


Open FormsApplication.java and edit it as following

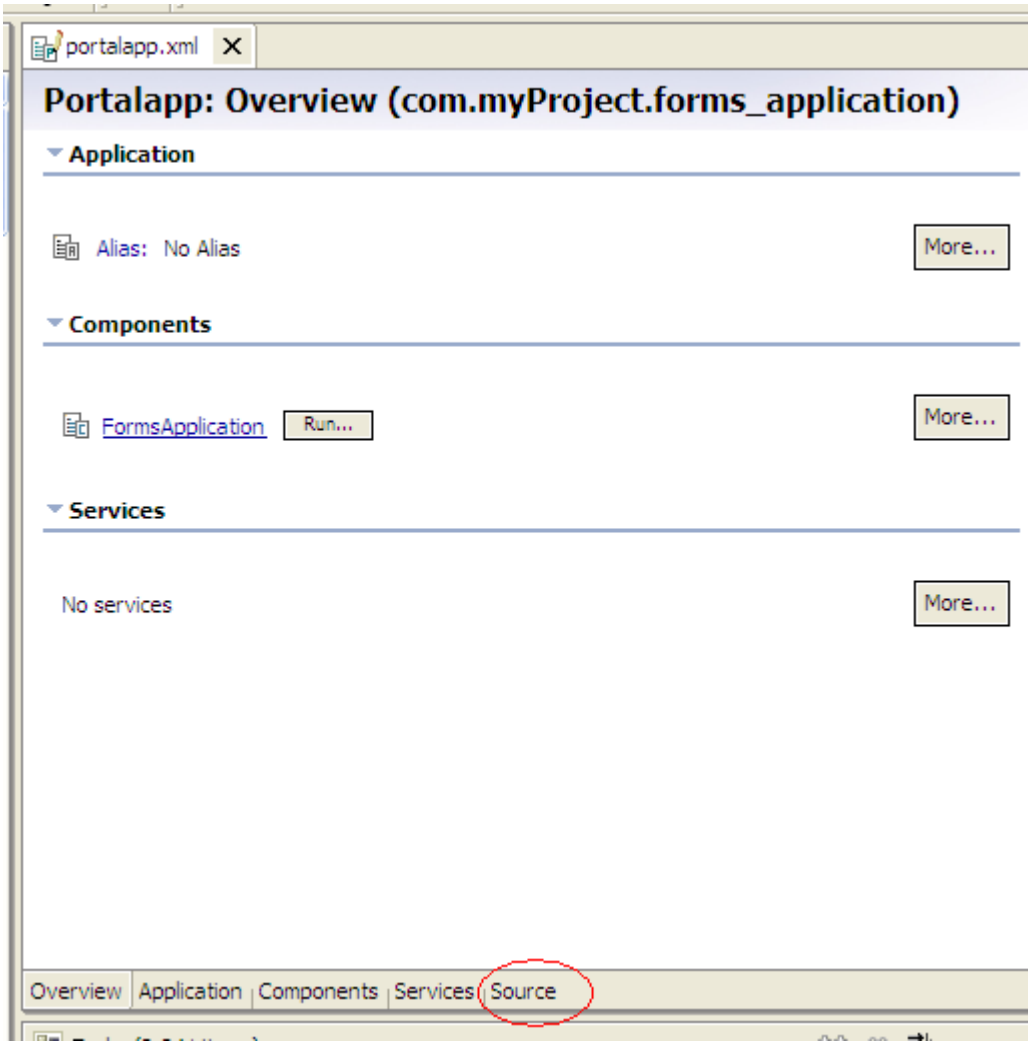
```
public static class FormsApplicationDynPage extends JSPDynPage{
  boolean flag = true;
  int myCondition = 4;
  public void doInitialization(){
      if (myCondition==4)
          flag = true;
      else
          flag = false;
  }

  public void doProcessAfterInput() throws PageException {
  }

  public void doProcessBeforeOutput() throws PageException {

    if(flag)
   { this.setJspName("Page_A.jsp"); }
    else
    {this.setJspName("Page_B.jsp");}
  }
}
```

We are doing two things here. Firstly put the condition under doInitialization method and set the flag so that the condition is identified for which page needs to be displayed. Secondly, depending on this flag, set the required jsp page to be displayed on click of the link, in the method doProcessBeforeOutput.

Portalapp.xml file



Open the portalapp.xml in NWDS and check the source. By default, overview is selected.




```
portalapp.xml
<?xml version="1.0" encoding="utf-8"?>
<application>
  <application-config>
    <property name="PrivateSharingReference" value="com.sap.portal.htmlb"/>
  </application-config>
  <components>
    <component name="FormsApplication">
      <component-config>
        <property name="ClassName" value="com.myPackage.FormsApplication"/>
        <property name="ComponentType" value="jspnative"/>
        <property name="JSP" value="pagelet/Page_A.jsp"/>
      </component-config>
      <component-profile/>
    </component>
  </components>
  <services/>
</application>
```

Here you can see that the *application-config* tag has the property name set to “PrivateSharingReference”. Change it to “SharingReference”

Under the *component-config* tag, property name “ComponentType” and “JSP” has to be removed. So the new xml would look like as following

Save this xml and deploy your par onto the server.

Note: It is assumed that the link saying “Click here to open the page” is in another par file and already exists. Hence I have not covered the logic for how to open the above mentioned par (com.myProject.forms_application.FormsApplication) with this link.

Conclusion

Single par file has now two pages which can be opened through a single iview. Single link would be displayed on a page, on click of which, required jsp page is rendered depending on the condition. This reduces the work of creating new iview for another jspDynpage and altering the par for displaying the link based on the condition.

Related Content

Tutorial III: Building a JSP DynPage

Minimizing Use of Public Sharing References

Overview of the Content Development Process

Java development methodologies (Part II)

For more information, visit the Portal and Collaboration homepage.

Disclaimer and Liability Notice