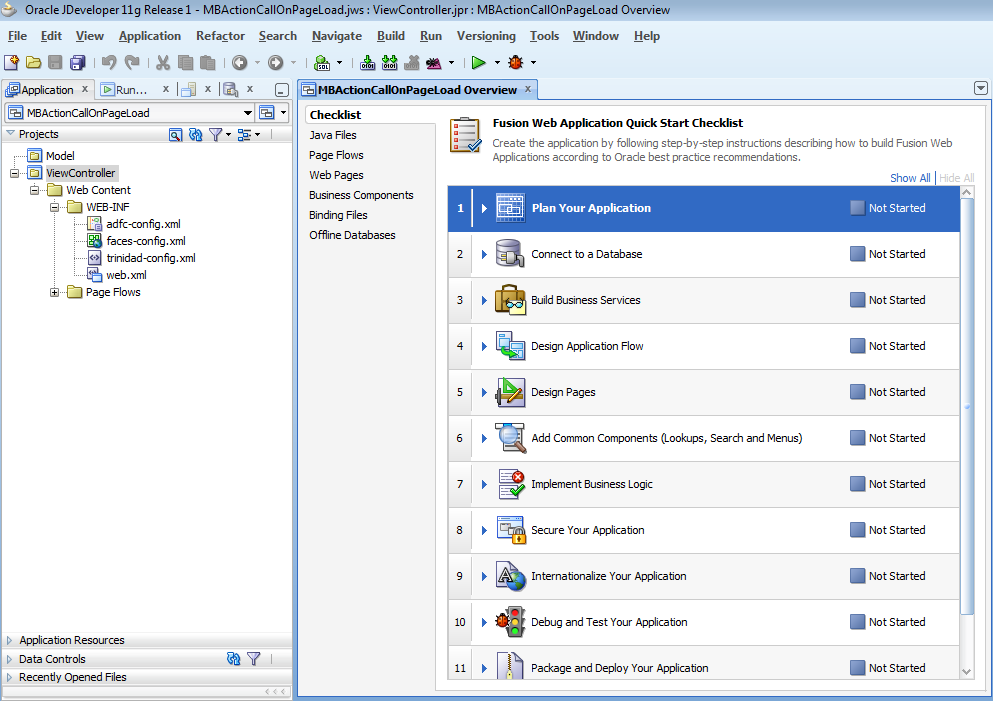
**Invoking A Managed Bean Method On ADF PageLoading**

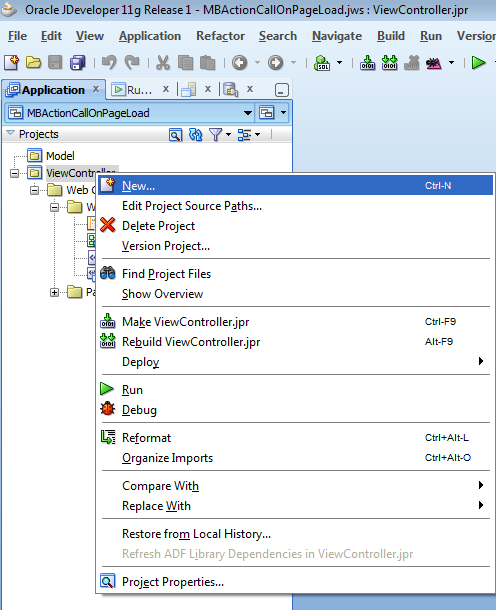
### Creating the ADF application

* 1. Create a new ADF application named ‘MBActionCallOnPageLoad’

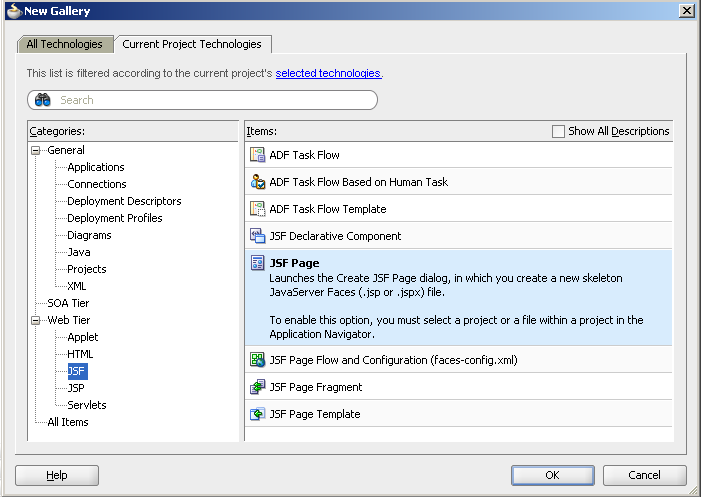


### Creating a simple jspx page

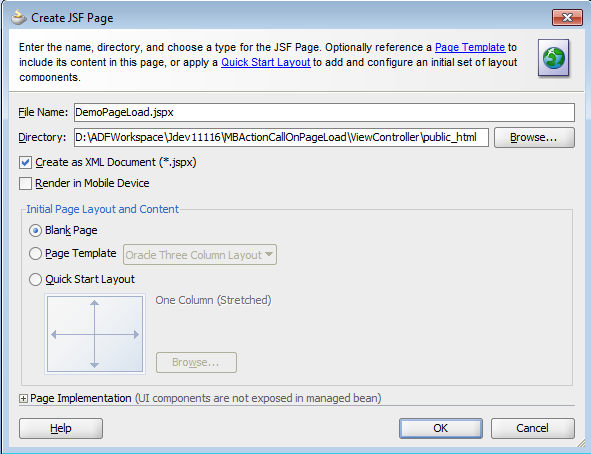
* 1. Right click on the ‘View Controller’ project and click on ‘New’



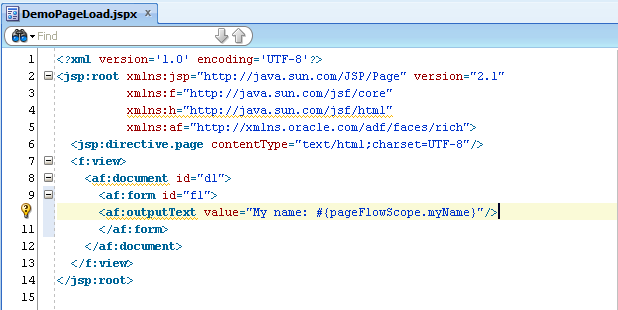
2.2 Select ‘JSF’ in the left pane in the ‘Current Project Technologies’ tab. Select ‘JSF Page’ in the right pane and click on ‘OK’.



2.3 Name the file as ‘DemoPageLoad.jspx’ and click on ‘OK’



2.4 Add an output text to the jspx page as follows:



**Source code for reference:**

<?xml version='1.0' encoding='UTF-8'?>

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.1"

xmlns:f="http://java.sun.com/jsf/core"

xmlns:h="http://java.sun.com/jsf/html"

xmlns:af="http://xmlns.oracle.com/adf/faces/rich">

<jsp:directive.page contentType="text/html;charset=UTF-8"/>

<f:view>

<af:document id="d1">

<af:form id="f1">

<af:outputText value="My name: #{pageFlowScope.myName}"/>

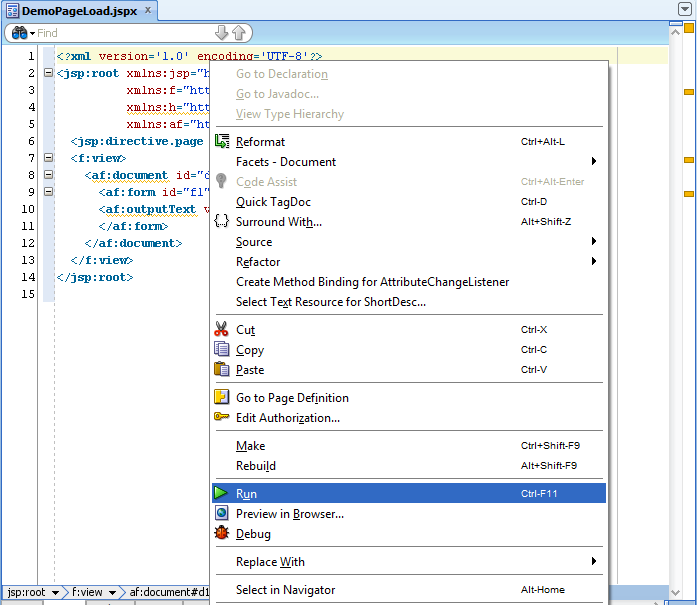
</af:form>

</af:document>

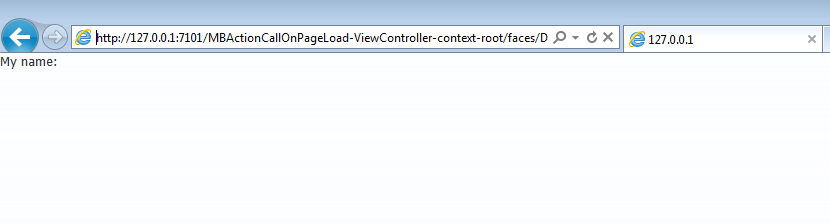
</f:view>

</jsp:root>

2.5 Right click on the jspx page just created and click on ‘Run’.

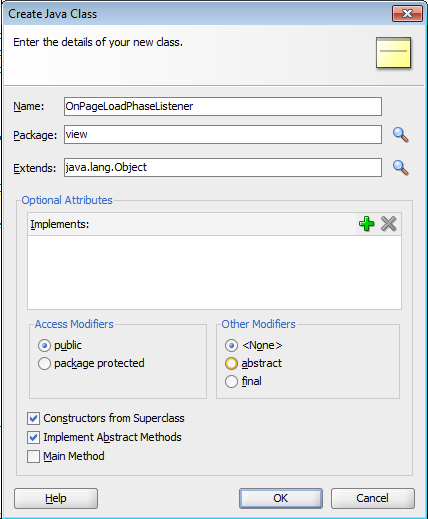


2.6 Observe the output name as blank in the resultant screen as till now, no method has been called and hence, no value set.

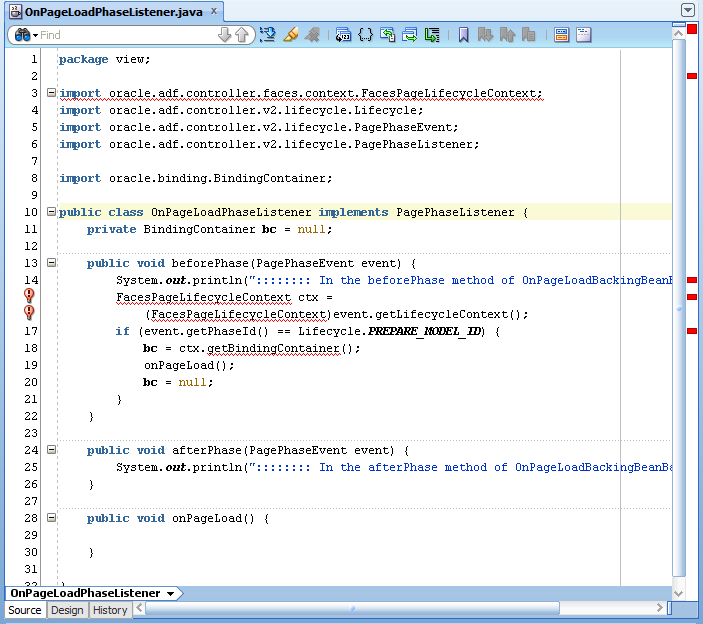


### Creating Phase Listener class

3.1 Create a Java class ‘OnPageLoadPhaseListener’ as below



3.2 Write the class code as below. This class will act as a Phase Listener whose beforePhase and afterPhase methods will be called before and after the phases of ADF application lifecycle.



**Source code for reference:**

package view;

import oracle.adf.controller.faces.context.FacesPageLifecycleContext;

import oracle.adf.controller.v2.lifecycle.Lifecycle;

import oracle.adf.controller.v2.lifecycle.PagePhaseEvent;

import oracle.adf.controller.v2.lifecycle.PagePhaseListener;

import oracle.binding.BindingContainer;

public class OnPageLoadPhaseListener implements PagePhaseListener {

private BindingContainer bc = null;

public void beforePhase(PagePhaseEvent event) {

System.out.println("beforePhase method called" );

FacesPageLifecycleContext ctx =

(FacesPageLifecycleContext)event.getLifecycleContext();

if (event.getPhaseId() == Lifecycle.PREPARE\_MODEL\_ID) {

bc = ctx.getBindingContainer();

onPageLoad();

bc = null;

}

}

public void afterPhase(PagePhaseEvent event) {

System.out.println("afterPhase method called" );

}

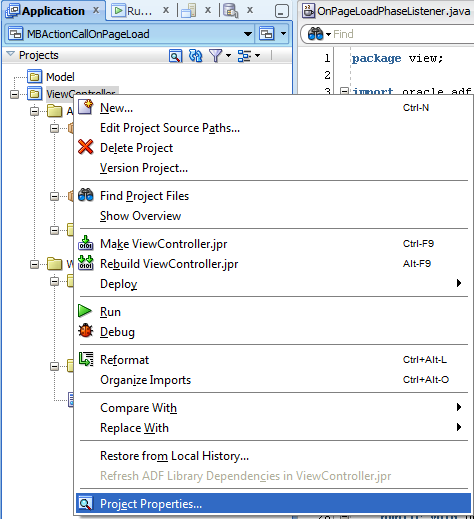
public void onPageLoad() {

}

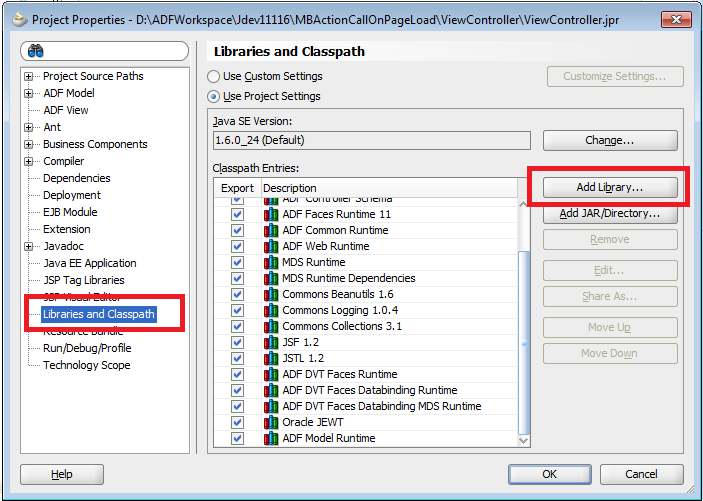
}

3.3 If there are errors noticed in step 3.6 i.e. FacesPagesLifeCycleContext class is not found, follow the steps below:

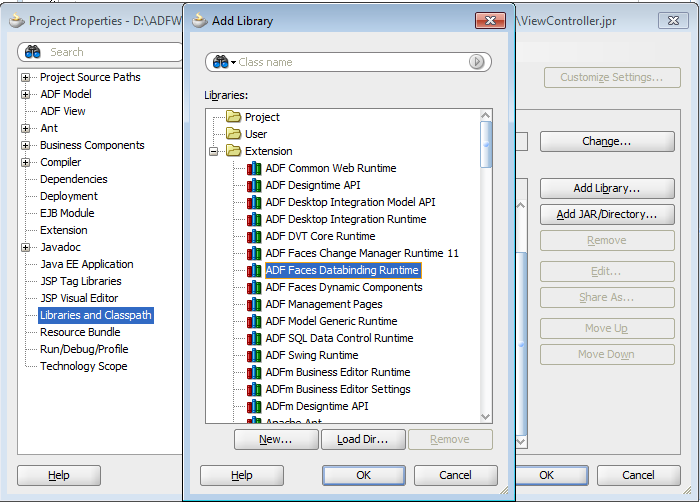
a) Right click on ViewController and go to Project properties.



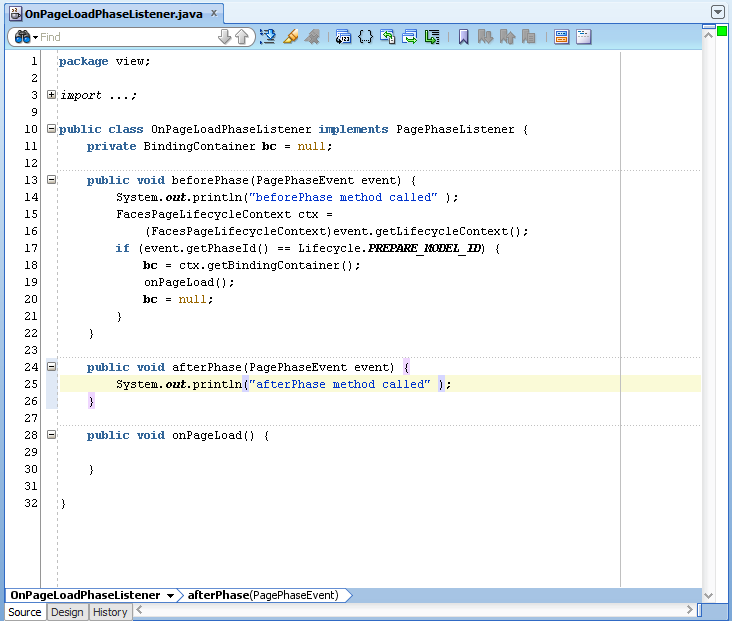
1. Select ‘Libraries and Classpath’ and click on ‘Add Library’ button.



1. Select the library ‘ADF Faces Databinding Runtime’ and click on OK.

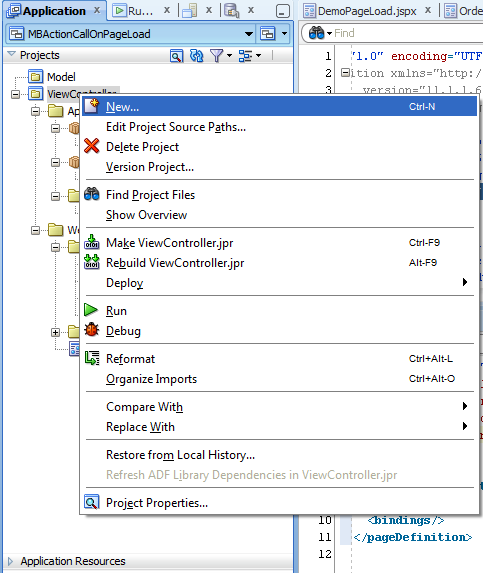


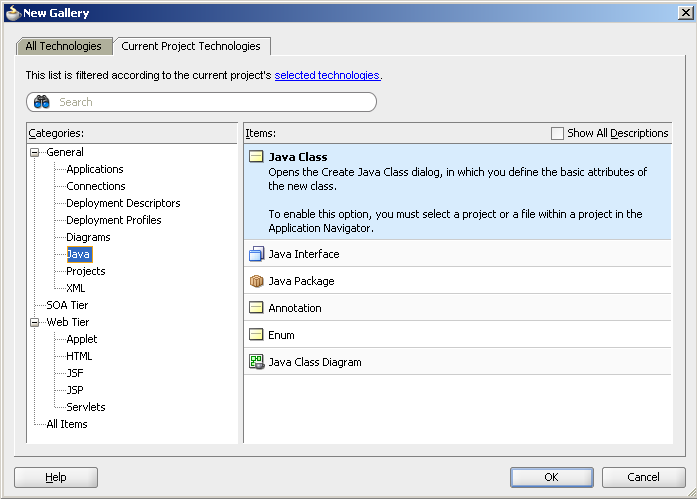
1. Notice that no errors now appear in the Phase Listener class.

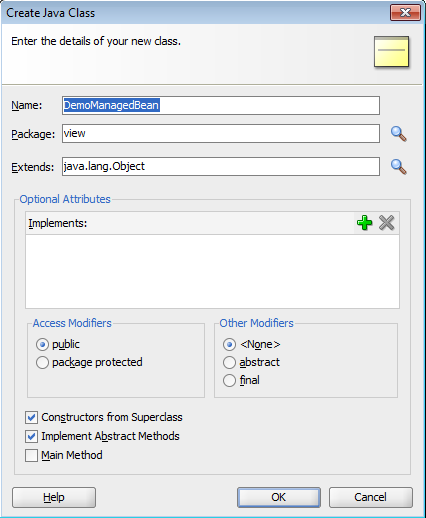


### Creating Managed Bean with the method to be called on page load

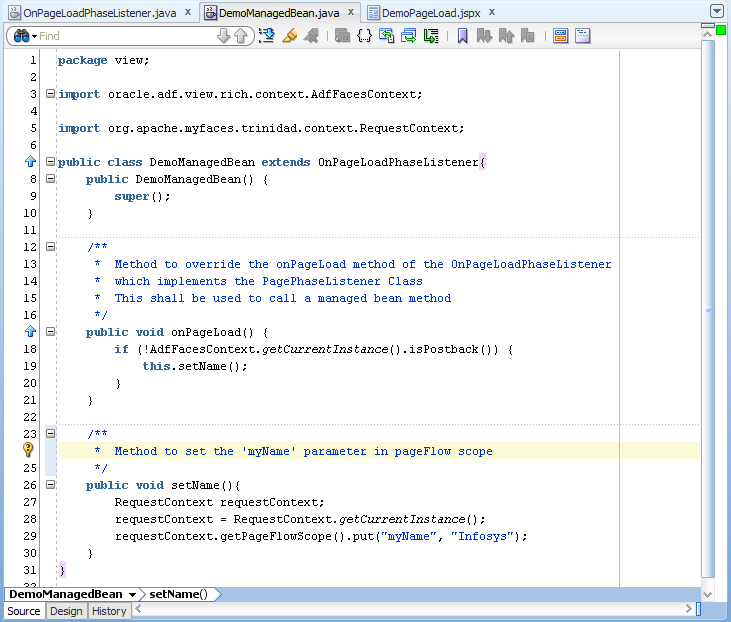
4.1 Create a new Java class named ‘DemoManagedBean.java’ in the ViewController project.







4.2 Write the managed bean code as below. This managed bean class should implement the OnPageLoadPhaseListener class just created and should override the onPageLoad method within which the method of managed bean can be called which we intended for. Say the method which we wanted to call on page load is ‘setName’. Here we have a simple code which puts the value ‘Infosys’ to a variable ‘myName’



**Source code for reference:**

package view;

import oracle.adf.view.rich.context.AdfFacesContext;

import org.apache.myfaces.trinidad.context.RequestContext;

public class DemoManagedBean extends OnPageLoadPhaseListener{

public DemoManagedBean() {

super();

}

/\*\*

\* Method to override the onPageLoad method of the OnPageLoadPhaseListener

\* which implements the PagePhaseListener Class

\* This shall be used to call a managed bean method

\*/

public void onPageLoad() {

if (!AdfFacesContext.getCurrentInstance().isPostback()) {

this.setName();

}

}

/\*\*

\* Method to set the 'myName' parameter in pageFlow scope

\*/

public void setName(){

System.out.println("inside setname...");

RequestContext requestContext;

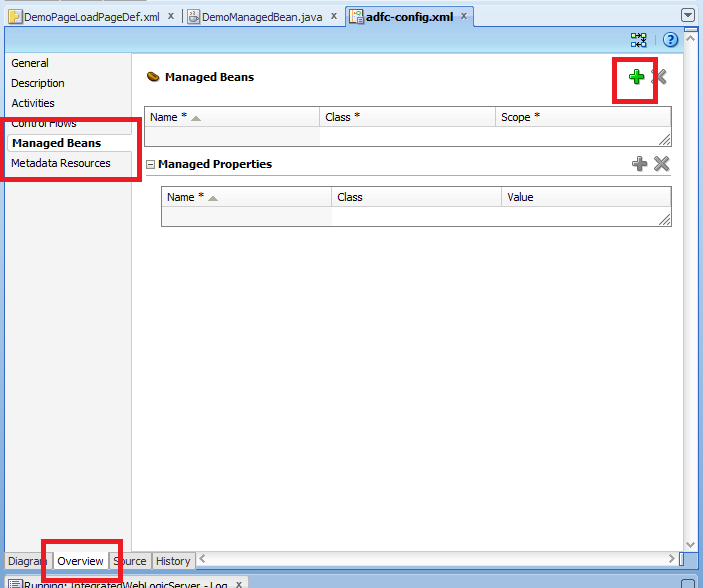
requestContext = RequestContext.getCurrentInstance();

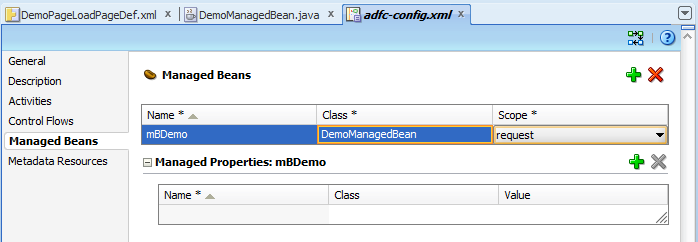
requestContext.getPageFlowScope().put("myName", "Infosys");

}

}

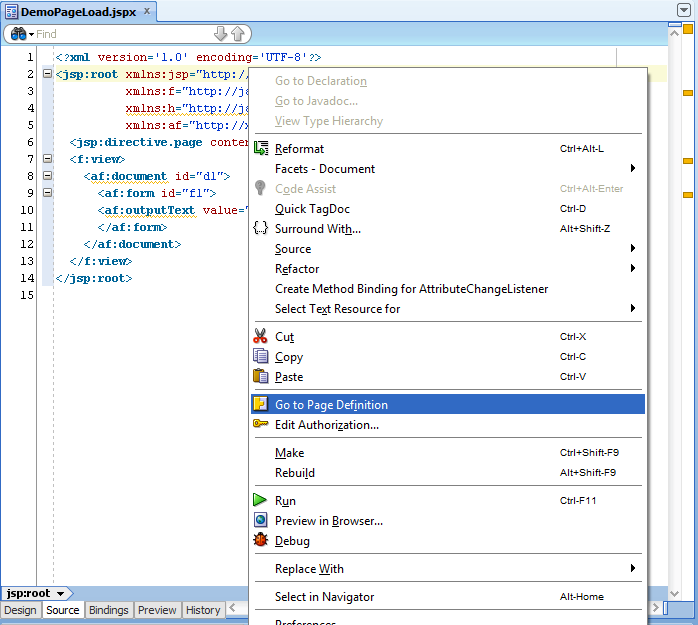
4.3 Register this Java class as the managed bean in adfc-config.xml file.



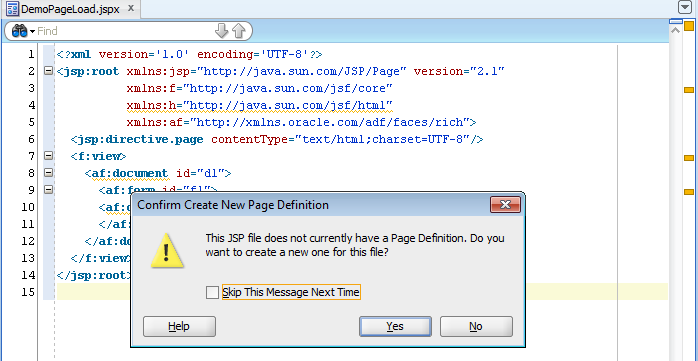


### Calling the method on page load

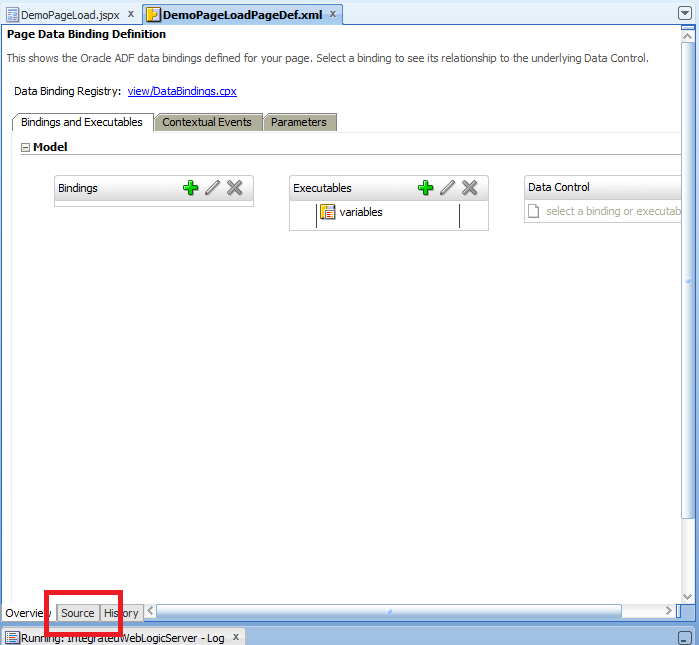
5.1 Right click on the jspx screen just created add click on ‘Go to Page Definition’ so as to go to the page definition that contains the bindings, executables, iterators being used in this page.



5.2 If it’s a newly created page with no bindings added, it will prompt for the creation of the Page Definition. Click on Yes to allow this. Thereafter, a new file named DemoPageLoadPageDef.xml shall be created in the project.

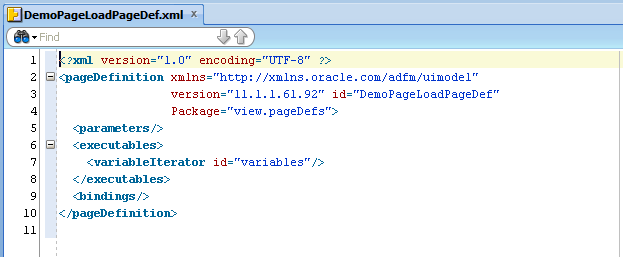


5.3 Click on the Source tab of this Page definition.

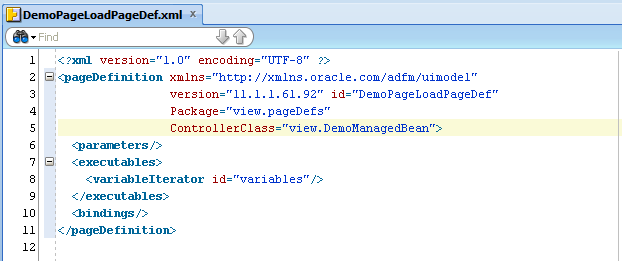


5.4 Update the xml file just created with the code as below.

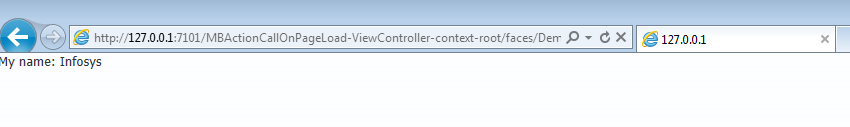
1. Code as in the default page definition xml file:



1. Code should be changed as shown below i.e. the path of the ManagedBean class has to be added into the ControllerClass attribute of the pageDefinition tag.



5.5 Run the jsp page again and see that the name value now shows the value as ‘Infosys’ which we saved indicating that the managed bean method was called as soon as the page was loaded.



**Sample Application source code for reference:**



**Reference:**

<http://udayarocks.wordpress.com/2011/07/22/how-to-invoke-a-method-from-managed-bean-when-jspx-page-loads-in-adf/>