**Customizing ADF UI components’ default texts**

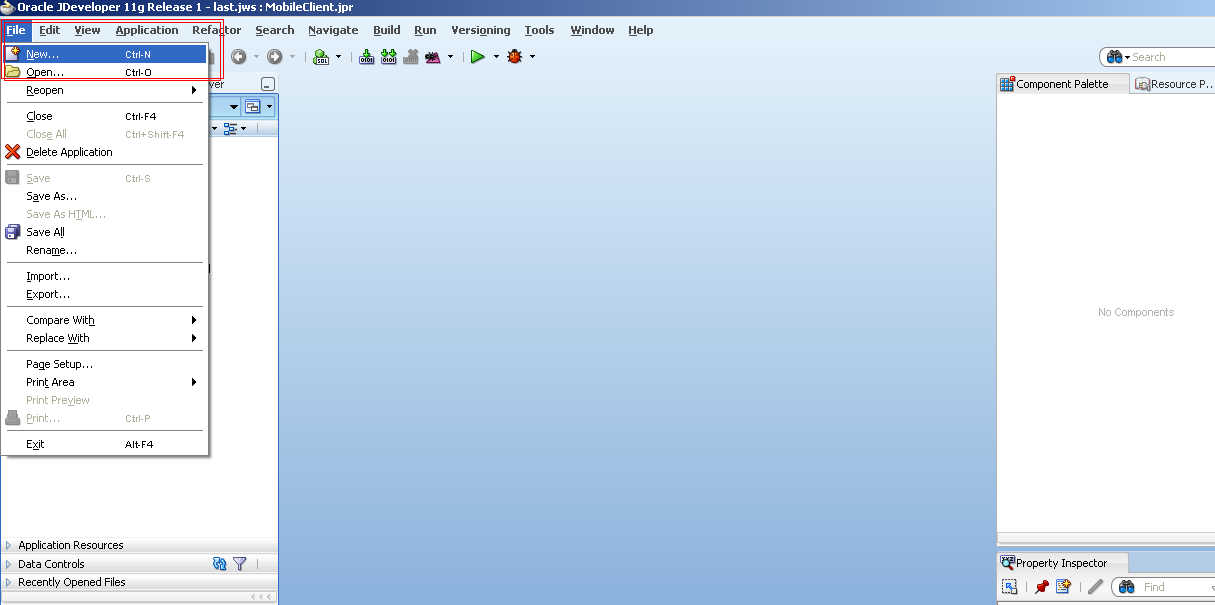
**Understanding Skinning and Resource Bundles**

Skinning refers to changing the default properties of the UI components which may include the colors, fonts, positions, icons and texts etc. It allows us to define and configure a skin in Trinidad-config.xml and trinindad-skins.xml files to be used by the UI components in the application.

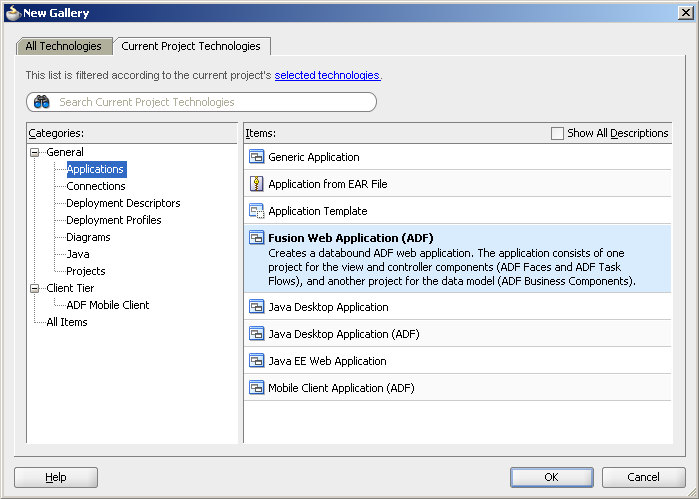
A skin can further be configured to refer a CSS file or a resource bundle class. The CSS file would contain and be used for the layout properties like color, fonts, icons, positions etc whereas the Resource Bundle shall contain the text values which shall override the default string values of the components. Both the CSS and Resource Bundles are common to the application and shall be all the screens. Here in this document, we shall focus on overriding the default String values of UI components i.e. using resource bundles in a skin.

### **Creating the ADF application**

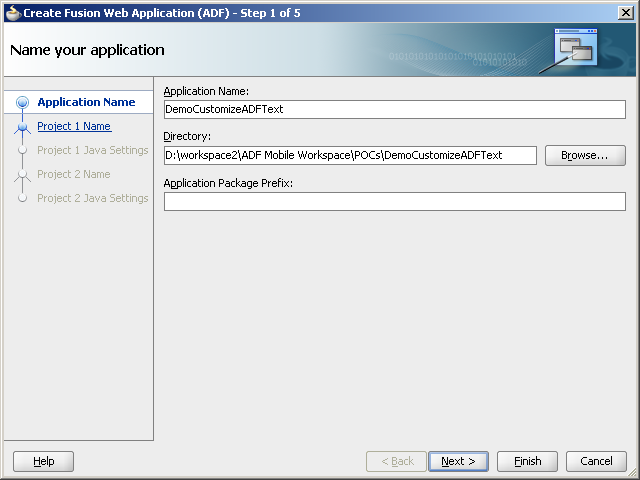
* 1. Click on ‘File’ menu and click on ‘New’.



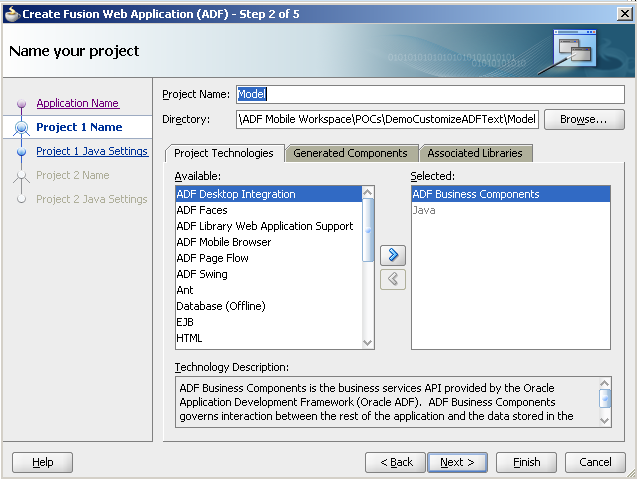
1.2 Select ‘Applications’ from the left pane in the ‘Current Project Technologies’ tab and then select ‘Fusion Web Application (ADF)’. Click OK.



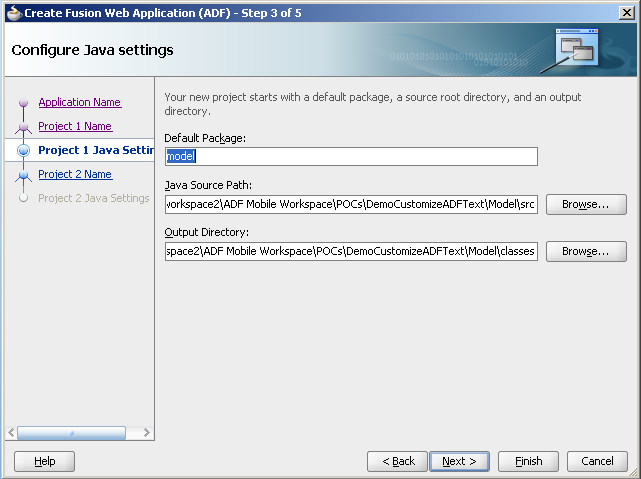
* 1. Name the new application to be created as ‘DemoCustomizeADFText’. You will notice that the Directory field also changes accordingly. Click ‘Next’.



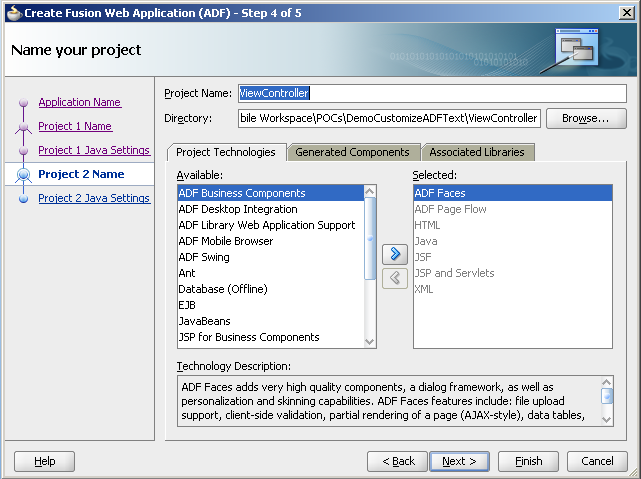
* 1. Accept the default and click on ‘Next’.



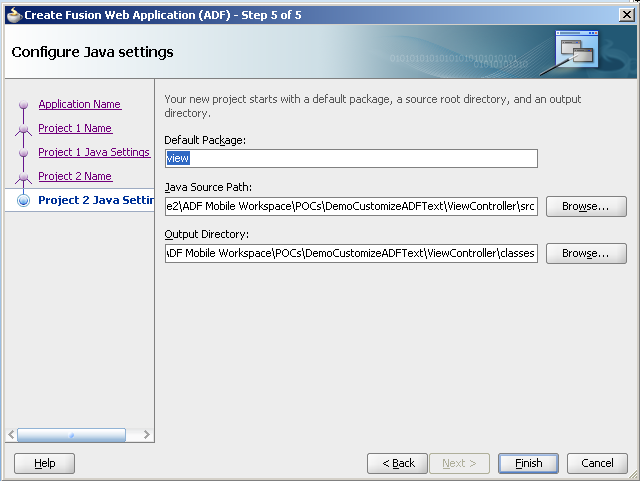
* 1. Accept the default and click ‘Next’.



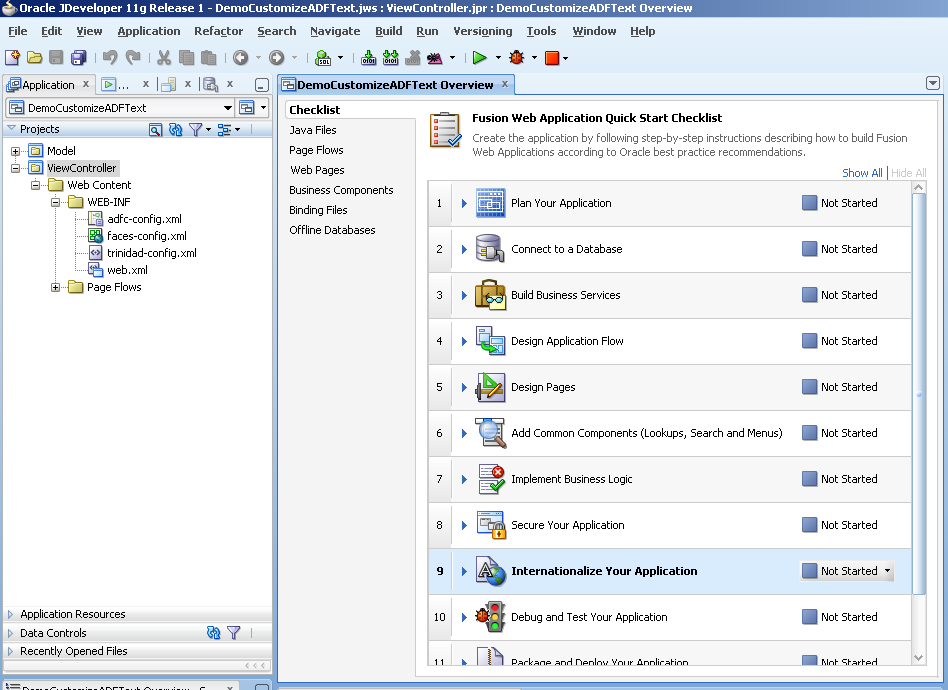
* 1. Ensure that the Project Technologies have ‘ADF Faces’, ‘ADF Page Flow’, ‘HTML’, ‘Java’, ‘JSF’, ‘JSP and Servlets’, ‘XML’ selected. Click ‘Next’.



* 1. Accept the default and click ‘Finish.

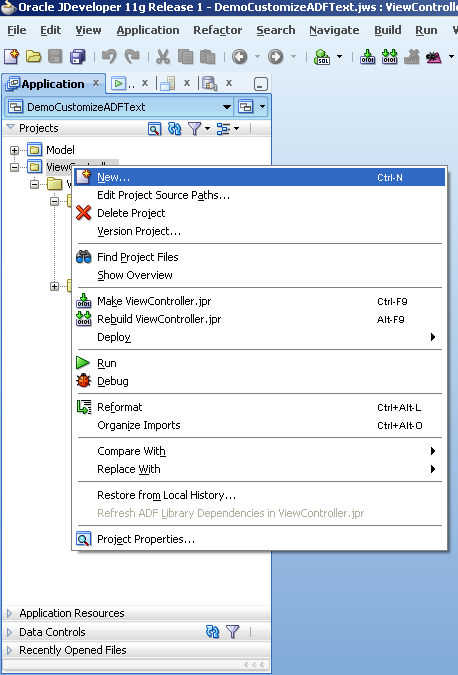


* 1. A new application has been created. At the left is the Application Navigator containing the application file structure.

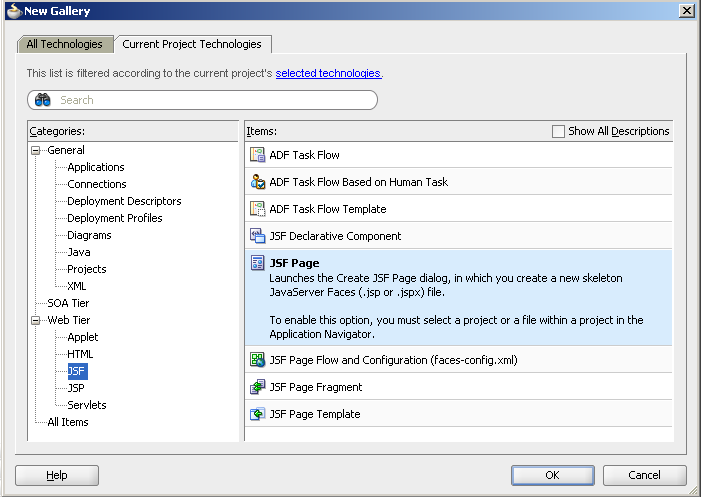


### **Creating the JSP page with required component**

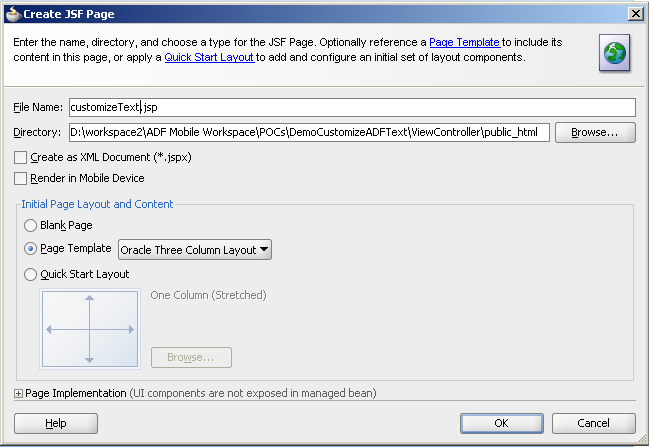
* 1. Right click on the ‘View Controller’ project and click on ‘New’



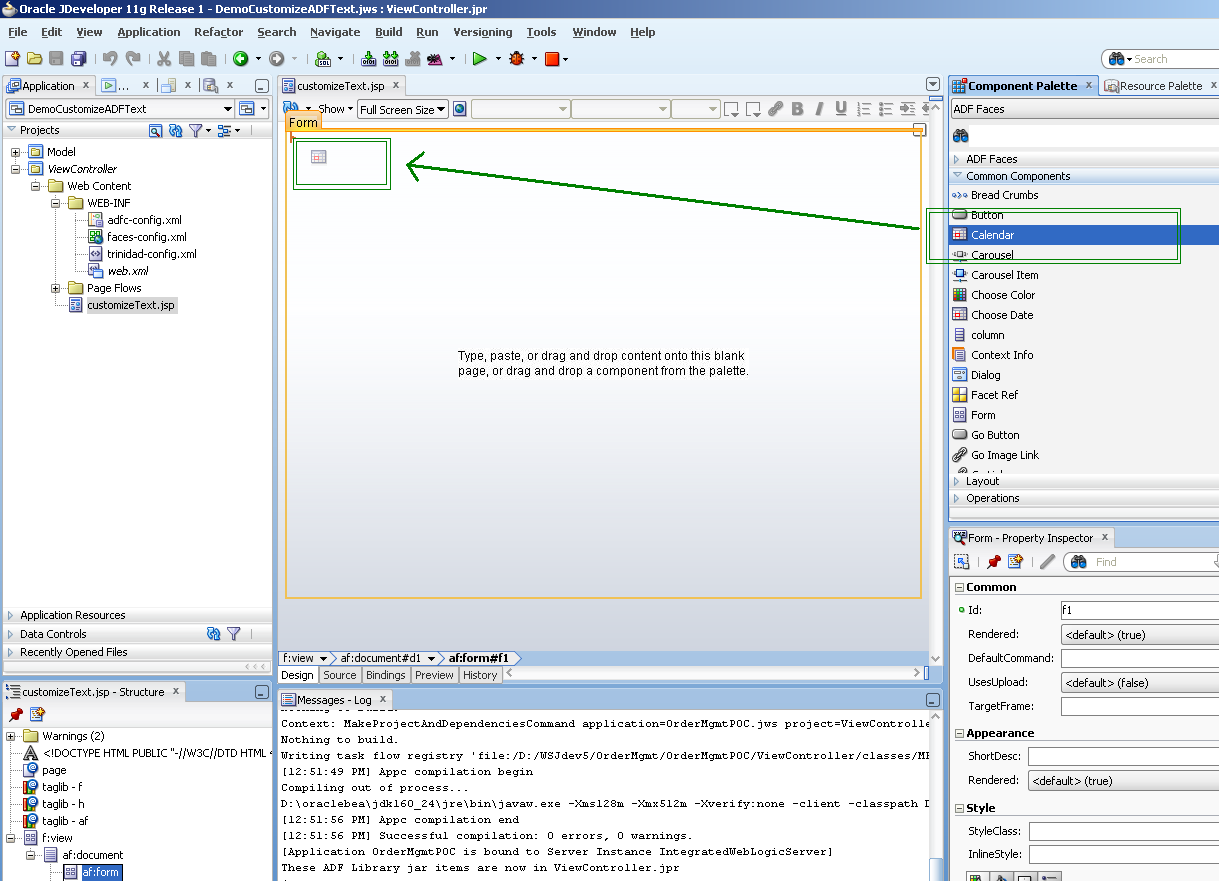
2.2 Select ‘JSF’ in the left pane in the ‘Current Project Technologies’ tab. Select ‘JSF Page’ in the right pane and click on ‘OK’.



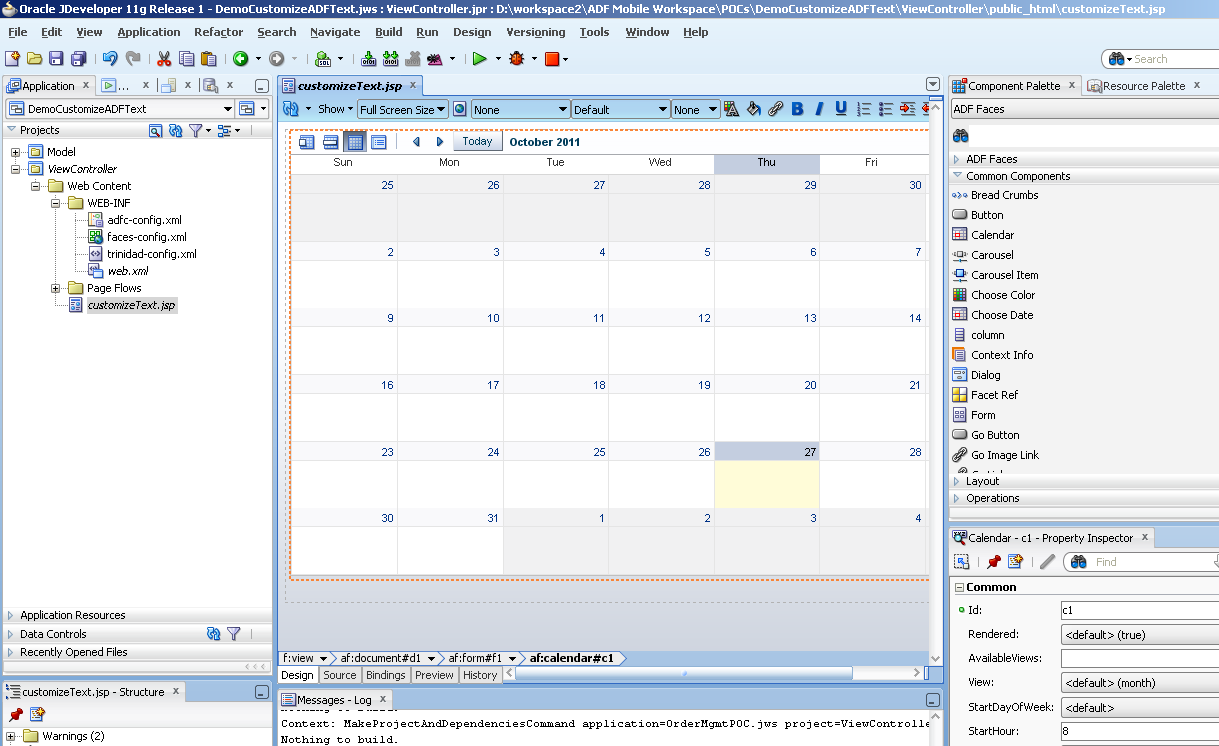
2.3 Name the file as ‘customizeText.jsp’ and click on ‘OK’



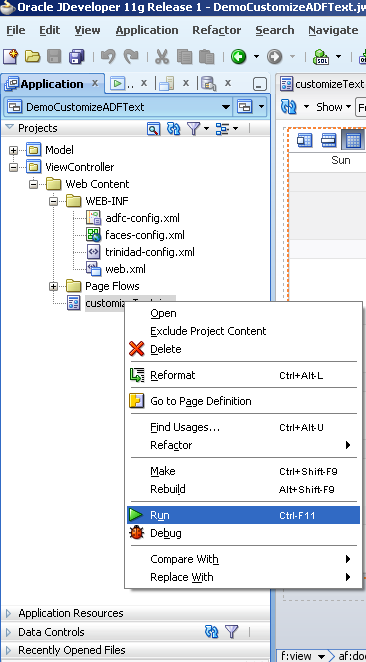
2.4 Drag a ‘Calendar’ component from the component pallete and drop it to the JSP page just created.



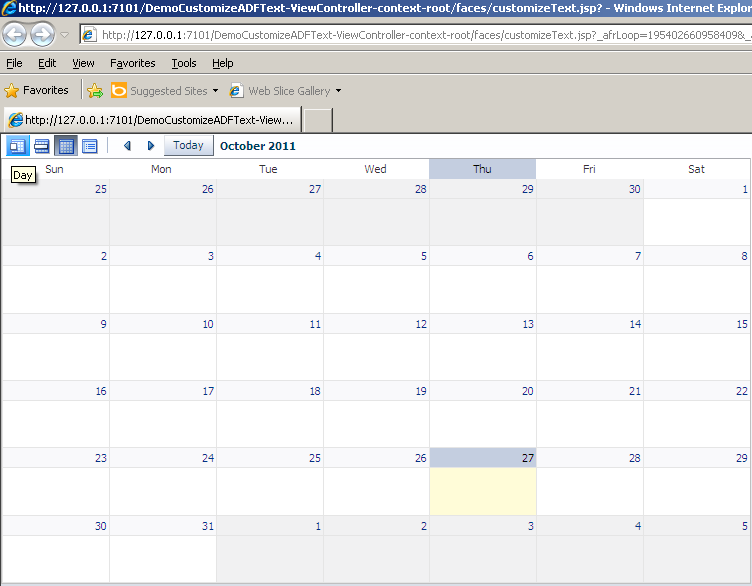
2.5 The JSP page will be created as below

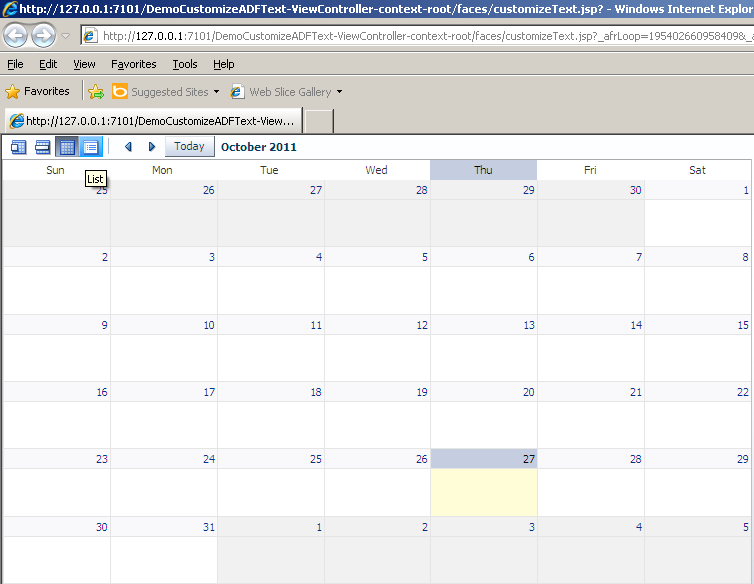


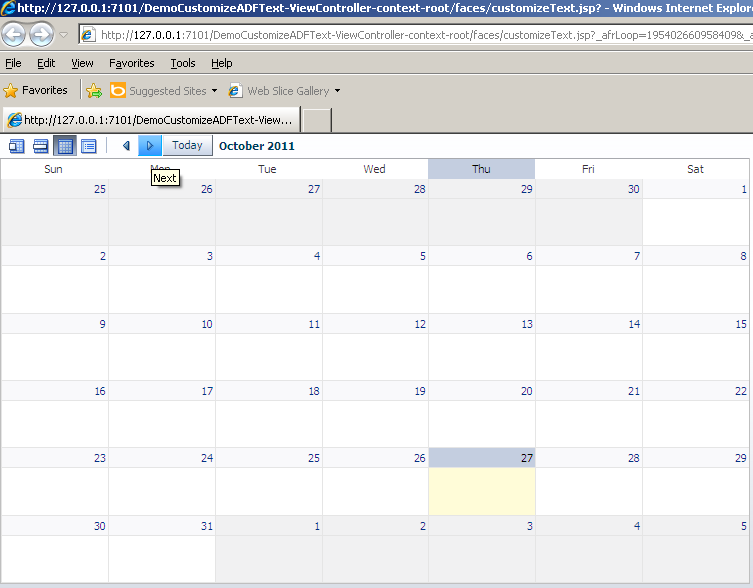
2.6 Right click on the JSP page just created and click on ‘Run’.



2.7 Observe the default strings of the component i.e. ‘>’ tooltip as ‘Next’, button text as ‘Today’ etc.

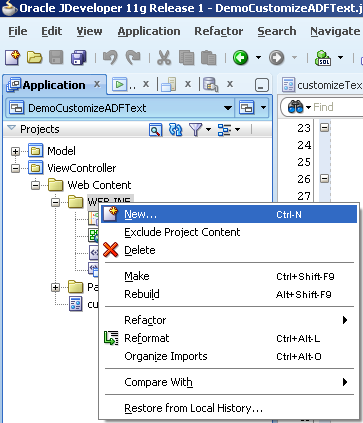


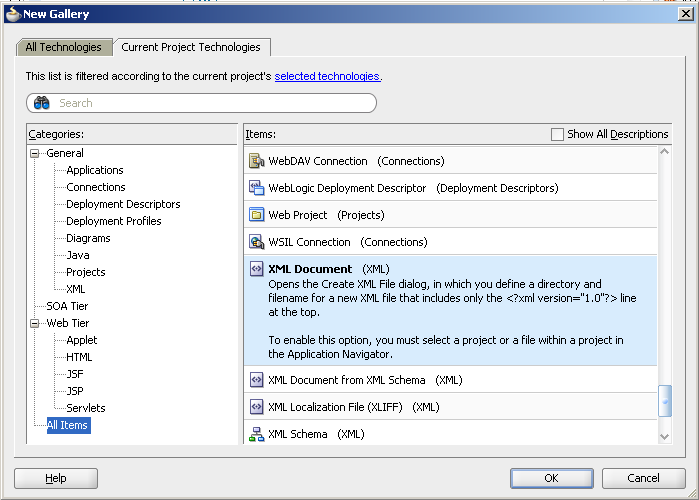


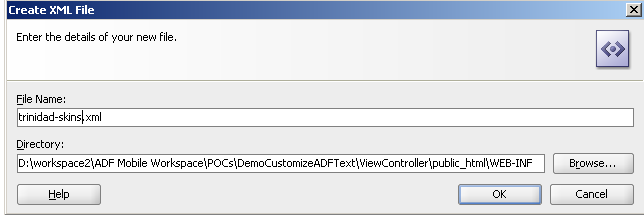


### **Customizing the text values**

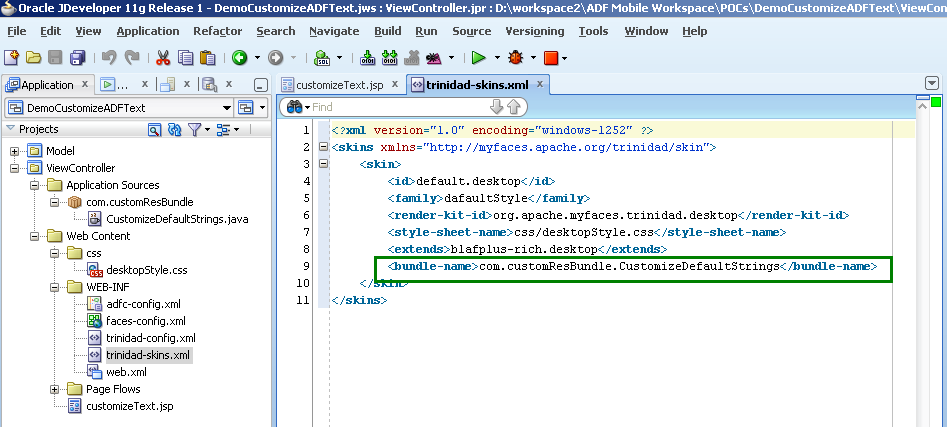
3.1 Create a new XML file ‘tirinidad-skins.xml’ in the WEB-INF folder of ViewController. This file contains the skinning configurations of the ADF application so as to customize the UI design(color, shapes, sizes of components) and default strings.







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



**Code for reference:**

<?xml version="1.0" encoding="WINDOWS-1252"?>

[<skins xmlns="http://myfaces.apache.org/trinidad/skin">](file:///C:\Users\abhishek_gupta54\Downloads\%3cskins%20xmlns=%22http:\myfaces.apache.org\trinidad\skin%22%3e)

[<skin>](file:///D:\ADFApplications\mobile\POCs\CodeForBOK\DemoCustomizeADFText\ViewController\public_html\WEB-INF\trinidad-skins.xml)

<id>default.desktop</id>

<family>dafaultStyle</family>

<render-kit-id>org.apache.myfaces.trinidad.desktop</render-kit-id>

<style-sheet-name>css/desktopStyle.css</style-sheet-name>

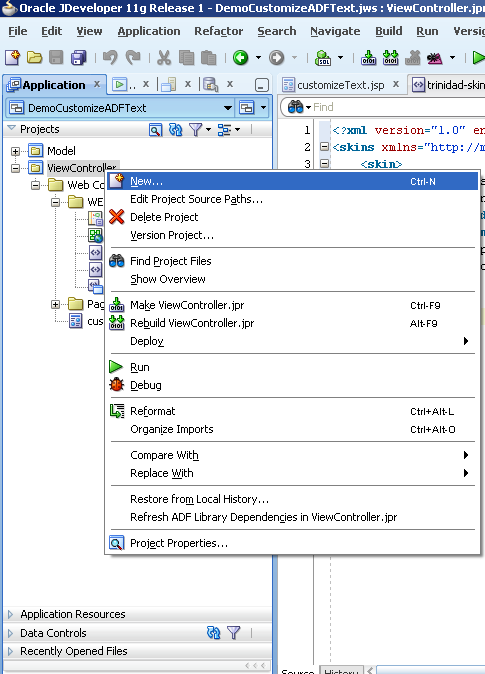
<extends>blafplus-rich.desktop</extends>

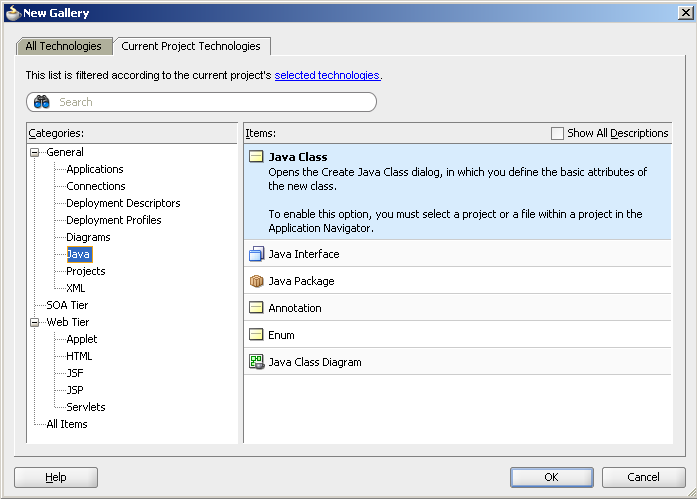
<bundle-name>com.customResBundle.CustomizeDefaultStrings</bundle-name>

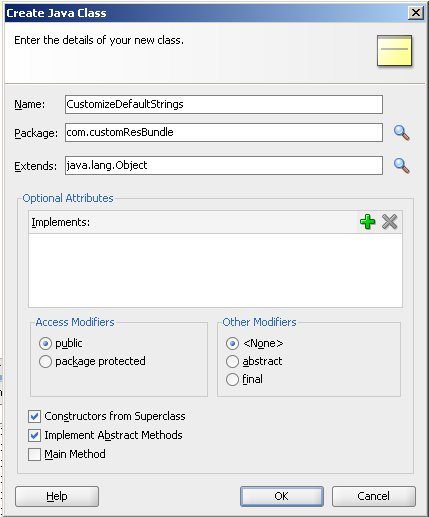
</skin>

</skins>

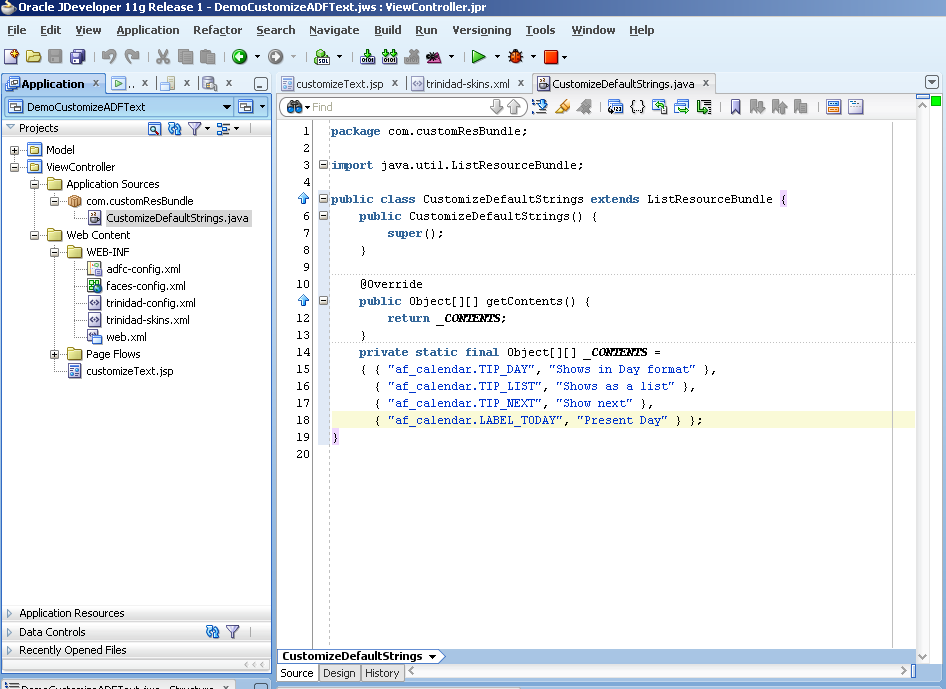
3.3 Create a new Java class named as ‘CustomizeDefaultStrings’ in the package com.customResBundle. If you happen to change the class name or the package name, remember to update the correct path in the <bundle-name> tag of Trinidad-skins.xml file (as in step 3.2).







3.4 Update the class code as below.



**Code for reference:**

package com.customResBundle;

import java.util.ListResourceBundle;

public class CustomizeDefaultStrings extends ListResourceBundle {

public CustomizeDefaultStrings() {

super();

}

@Override

public Object[][] getContents() {

return \_CONTENTS;

}

private static final Object[][] \_CONTENTS =

{ { "af\_calendar.TIP\_DAY", "Shows in Day format" },

{ "af\_calendar.TIP\_LIST", "Shows as a list" },

{ "af\_calendar.TIP\_NEXT", "Show next" },

{ "af\_calendar.LABEL\_TODAY", "Present Day" } };

}

**Understanding the code for overriding the string values:**

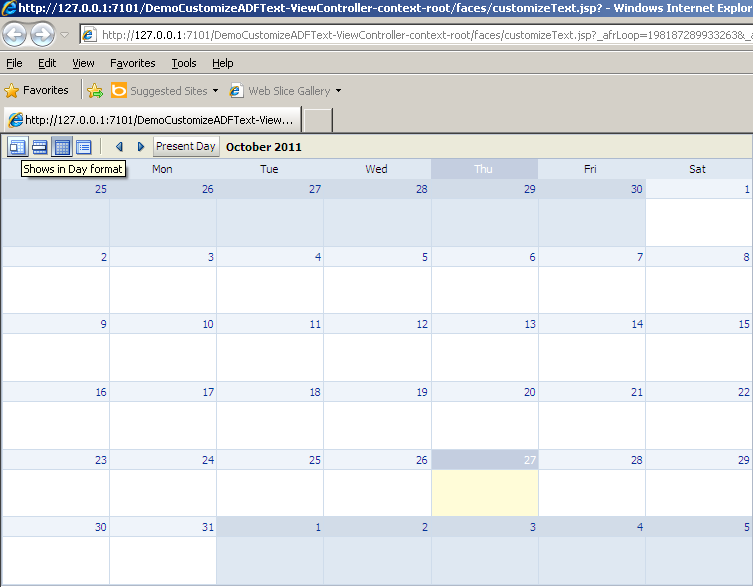
1. \_CONTENTS named constant variable here stores the texts we need to replace the default texts with. These key value pairs should be comma separated as in the code.

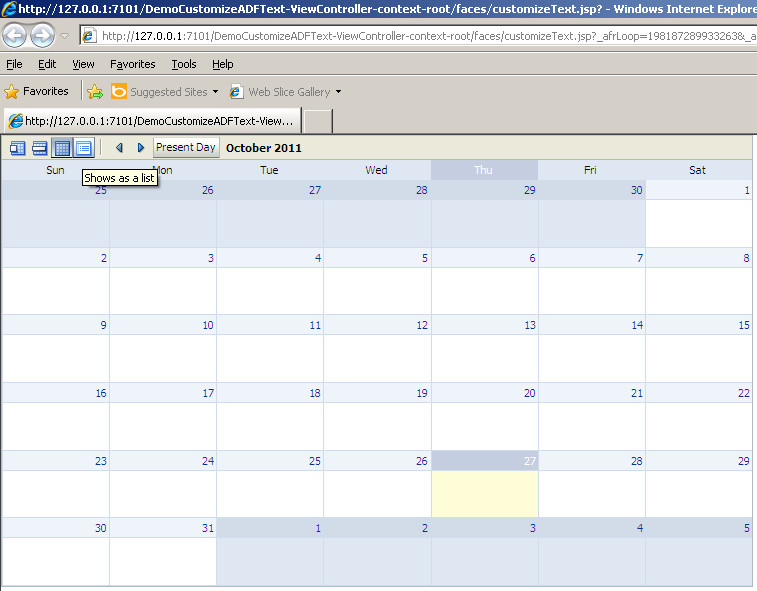
Each default text property is represented by the name of format af\_COMPONENT.PROPERTY where COMPONENT is the ADF component whose default strings are to be changed and PROPERTY represents the value that has to be changed.

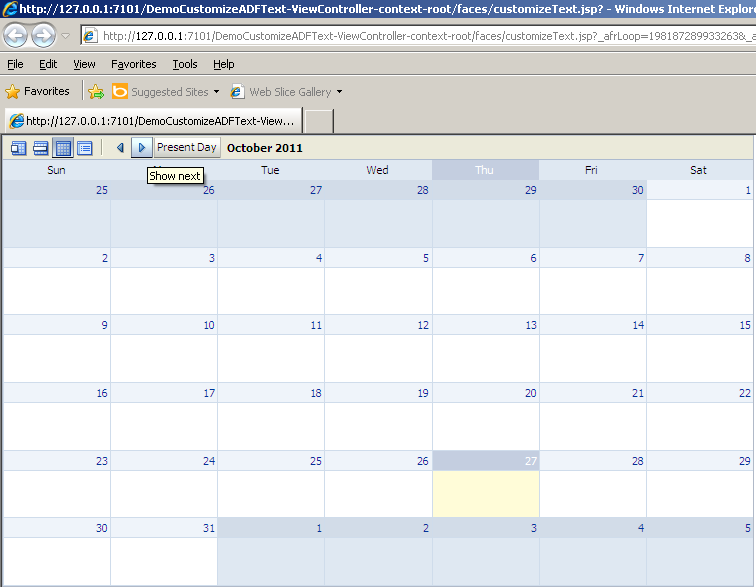
For example, af\_calendar.TIP\_NEXT represents the tooltip of the ‘>’ link in the calendar component. Similarly, to change the value for page loading message, the name should be used as af\_document.LABEL\_SPLASH\_SCREEN.

1. getContents() method is the default ADF method which is being overridden here. This method returns the 2-D object array storing the key value pairs of the component names and its corresponding string values in the UI. The properties mentioned in the \_CONTENTS get overridden whereas others retain their default values.
2. As this class is referenced from the skin, the default values of UI components’ strings get overridden and are then updated while displaying in the UI screens.

3.5 Observe the change in the string values as required.







**Sample Application source code for reference:**



**Reference:**

<http://download.oracle.com/docs/cd/E15523_01/apirefs.1111/e15862/toc.htm>