WebCenter Portal Task Flow Customization in 12c
Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
# Table of Contents

Introduction 1

Understanding WebCenter Task Flow Customization Process 1

Preparing a Customizable WebCenter Portal Application 1
  - Creating Customization Classes 2
  - Making Your Customization Classes Available to JDeveloper 4
  - Enabling Seeded Customizations for View Projects 5
  - Configuring Design Time Customization Layers for JDeveloper 6
  - Configuring the adf-config.xml File 7

Customizing WebCenter Portal Task Flows 8
  - Setting up for WebCenter Portal Task Flows 8

Applying Customizations 11
  - Applying Customization to the Deployed WebCenter Portal Application 11
  - Removing Customization 12
Introduction

WebCenter Portal exposes its common functionalities using task flows (ADF Libraries). With the JDeveloper Customization Developer role, customers can customize WebCenter Portal task flows to extend or alter the out-of-the-box look and feel and functionality. The beauty of the JDeveloper Customization Developer role is that look and feel of the out-of-the-box shipped task flows of WebCenter Portal can be altered without actually writing any code. This document describes the steps to alter the look and feel of the out-of-the-box shipped task flows using the JDeveloper Customization Developer role.

Understanding WebCenter Task Flow Customization Process

When task flow customizations are applied to a deployed WebCenter Portal, customizations are applied to all instances of the task flow. Hence, you need not repeat these customizations for each specific page, where the task flow is being used.

For WebCenter Portal task flow customizations, create seeded customizations for task flows in an ADF Fusion technology application and package the task flow customizations in a JAR archive using JDeveloper. Import the seeded customizations later to the MDS repository attached to the WebCenter Portal instance.

Preparing a Customizable WebCenter Portal Application

Create an ADF Fusion application with a default role as shown in Figure 1.

Figure 1. Create ADF Fusion Application
Creating Customization Classes

A customization class is the interface that MDS uses to define which customization applies to the base definition metadata. Each customization class defines a customization layer (for example, site or user) and can contain multiple layer values. For a task flow customization, use the siteCC layer. The customization classes that are used in the application should be available in JDeveloper when customizing the application.

This white paper uses the approach of creating the CC classes in a sample application. Later, the CC classes are packaged in a JAR and placed under the JDeveloper classpath.

To create a customization class

1. In the application ViewController project of your newly created ADF Fusion application, add a new java class named WCSiteCC, extending "oracle.mds.cust.CustomizationClass" (see Figure 2).

![Create Java Class](image)

Figure 2. Create Customization Classes

2. Enter the details of your new class.

   - **Name**: WCSiteCC
   - **Package**: view
   - **Extends**: oracle.mds.cust.CustomizationClass
   - **Optional Attributes**:
     - **Implements**: 
     - **Access Modifiers**: public
     - **Other Modifiers**: <None> abstract final
     - **Constructors from Superclass**: checked
     - **Implement Abstract Methods**: checked
     - **Main Method**: unchecked
   - **Messages**: 

3. Click **OK** to create the customization class.
2. In WCSiteCC.java, add the code. The following is a sample WCSiteCC Class code.

```java
package view;

import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
import oracle.mds.core.MetadataObject;
import oracle.mds.core.RestrictedSession;
import oracle.mds.cust.CacheHint;
import oracle.mds.cust.CustomizationClass;

public class WCSiteCC extends CustomizationClass {
    public WCSiteCC() {
        super();
    }

    @Override
    public CacheHint getCacheHint() {
        // TODO Implement this method
        return null;
    }

    @Override
    public String getName() {
        // TODO Implement this method
        return "site";
    }

    @Override
    public String[] getValue(RestrictedSession mdsSession, MetadataObject mo) {
        // TODO Implement this method
        return new String[] {"webcenter"};
    }
}
```

3. Rebuild the ViewController project.
Making Your Customization Classes Available to JDeveloper

After you create the customization classes, you must make them available to JDeveloper so that you can use them when implementing customizations. When working with the Customization Developer role, your customization classes must be available on Project’s classpath.

To make the customization classes available to JDeveloper:

1. In the Application window of the JDeveloper, right-click the ViewController project and select Project Properties.
2. In the Project Properties dialog, select Deployment, then click New Profile.
3. In the Create Deployment Profile dialog (Figure 3), select JAR File from the Profile Type drop-down list.
4. In Deployment Profile Name, enter the deployment profile name, for example myCC, and click OK.

Click OK to create your new deployment profile and immediately open it to see its configuration.

- **Profile Type:** JAR File
- **Deployment Profile Name:** myCC
- **Description:** Create a simple JAR archive from a Project.

Figure 3. Create a Deployment Profile

5. Edit the deployment profile to add the WCSiteCC class (Figure 4).
6. Click OK to save the profile.
7. Click OK to close project properties.
8. From the File menu, click Save All.
9. In the Application window of the JDeveloper, right-click the ViewController project and select Deploy, then select the newly created deployment profile, for example, myCC.
10. In the Deploy dialog, select Deploy to JAR File, and click Finish.
    This deploys the jar profile under <Application>/ViewController/deploy/myCC.jar. where <Application> is the name of the application.
11. In the Application window of the JDeveloper, right-click the ViewController project, and select Project Properties.
12. In the Project Properties dialog, select Libraries and Classpath, and then select Add Jar/Directory to add myCC.jar to view Controller’s Classpath.

**Enabling Seeded Customizations for View Projects**

To enable seeded customization for view projects

1. In the Application window of the JDeveloper, right-click the ViewController project and select Project Properties.
2. In the Project Properties dialog, select ADF View.
3. Select the Enable Seeded Customizations check box (Figure 5).
Figure 5. Enable Seeded Customization for View Projects

4. Click OK.
5. From the File menu, click Save All.

Configuring Design Time Customization Layers for JDeveloper

For customizing WebCenter Portal task flows, configure the CC layer values that can be used in the JDeveloper Customization role. In JDEV_HOME/jdev/CustomizationLayerValues.xml, you can configure the CC layer values.

For WC task flow customizations, we use Site layer with value as WebCenter.

For example,

```xml
<cust-layers xmlns="http://xmlns.oracle.com/mds/dt">
  <cust-layer name="site" id-prefix="s">
    <cust-layer-value value="webcenter" display-name="WebCenter"/>
  </cust-layer>
</cust-layers>
```
Based on the above setting, the JDeveloper Customization role considers **WebCenter** as the layer value for **Site** layer.

**Note:** The site layer value is case sensitive.

Configuring the adf-config.xml File

The **adf-config.xml** file of the application must have an appropriate **cust-config** element in the **mds-config** section. The **cust-config** element allows clients to define an ordered and named list of customization classes. You can use the overview editor for the **adf-config.xml** file to add customization classes.

To add the customization classes

1. Open the **adf-config.xml** file of your application in the Overview editor.
2. On Overview editor, select **MDS**.
3. Add the **WCSiteCC** class to generate MDS Customization configuration (see **Figure 6**).

![Figure 6: Add Customization Classes](image)

4. From the File menu, click **Save All**.
The following is the example of the customization class order in the adf-config.xml file:

```xml
<adf-config xmlns="http://xmlns.oracle.com/adf/config">
  <adf-mds-config xmlns="http://xmlns.oracle.com/adf/mds/config">
    <mds-config xmlns="http://xmlns.oracle.com/mds/config" version="11.1.1.000">
      <cust-config>
        <match path="/">
          <customization-class name="view.WCSiteCC"/>
        </match>
      </cust-config>
    </mds-config>
  </adf-mds-config>
</adf-config>
```

Customizing WebCenter Portal Task Flows

Out-of-the-box all the WebCenter Portal task flows have well defined UI and behaviour, that is, business logic. In certain customer deployments, out-of-the-box look and feel or behaviour of the task flows may not be suitable. You may want to modify the look and feel of the portal server or may want to add your own task flow on one of the existing WebCenter Portal task flow or you may want to modify the UI of a WebCenter Portal task flow by eliminating few buttons on the toolbar or add your own button. This can be accomplished without opening the WebCenter Portal task flow JARs and modifying the shipped code of WebCenter Portal.

The JDeveloper Customization role is a powerful mechanism available, which allows customization of ADF Library without changing the code in the base library JAR. Since WebCenter Portal and ADF layer are built on top of MDS, the JDeveloper Customization role can be leveraged to extend WebCenter Portal task flows. All WebCenter Portal task flows are packaged as ADF Library so task flow customization is possible in JDeveloper design time.

Once the setup is done (see Configuring Design Time Customization Layers for JDeveloper) is ready for allowing task flow customization for WebCenter Portal task flows.

Setting up for WebCenter Portal Task Flows

This Setup is a onetime activity in JDeveloper used to set up the workspace to allow the customization of WebCenter Portal task flows. This step is mandatory before working on any of the task flow use cases mentioned above.

To setup workspace for WebCenter portal task flow

1. For customizing WebCenter Portal task flows, you need to include WebCenter Portal task flow view JAR in your project. Perform the following steps:
   a. Navigate to WC_ORACLE_HOME/webcenter/modules/oracle.webcenter.framework/ directory on the machine where WebCenter Portal server is installed.
   b. Copy the task flow containing the JAR that you intend to customize to a location visible from JDeveloper install. For example, if you want to customize search taskflow, you would copy search-service-view.jar.
2. Run the JDeveloper Studio in “Default Role”.

---

8 | WEBCENTER PORTAL TASK FLOW CUSTOMIZATION IN 12C
3. In the application window’s Projects palette select, select Libraries & Classpath.


5. Navigate to the WebCenter Portal task flow directory and, select the task flow view JAR you added in step 8 above.

6. Click Select and then click OK on the project properties.

7. From the File menu, click Save All.

8. In the application’s Projects palette select the Navigate Display Options icon and then click Show Libraries (Figure 7).
You can see All Libraries that have been included in the ViewController project in navigator.

![Figure 7. Application Projects Palette](image)

9. Select Tools, then Switch Roles, then Customization Developer to start JDeveloper in Customization role.

10. In JDeveloper Customization role, verify that the layer site is selected with WebCenter as layer value in the Customization Context window (Figure 8).

---

**WEBCENTER PORTAL TASK FLOW CUSTOMIZATION IN 12C**
5. Select jsf or jspx from the WebCenter Portal task flow and start the customization. 

Figure 9 shows an example for customizing createTopic.jsff in Discussion Forum service. 

All customizations will be saved under the following directory:

Application Directory/ViewController/libraryCustomizations
Applying Customizations

The output of the task flow customization is the generated MDS customization. The customizations show up as the .xml or .jspx.xml or .jsff.xml file in the project. These customization documents are essentially the instructions for MDS to apply delta on top of the base document that is shipped to show the customized behaviour at runtime.

Applying Customization to the Deployed WebCenter Portal Application

In order to view the customization done in Customizing WebCenter Portal Task Flows, these customizations need to be transferred to the MDS repository of the deployed WebCenter Portal instance. We recommend you to back up the MDS schema before performing this step, as this step updates runtime WebCenter Portal metadata repository. You can restore back to original state in case of any incorrect customization that can make the task flow non-functional.

To apply customization to deployed portal application

1. Rebuild the application in JDeveloper as a default user.
2. Create a JAR profile and package the library customizations under ViewController project in a JAR archive. For customization to get in Jar, add directory for ViewController/libraryCustomizations to Contributors in the Edit JAR Deployment Profile dialog as shown in Figure 10.

![Figure 10. Edit JAR Deployment Profile Properties](image-url)
3. Run the new deployment profile and take the generated JAR that contains customizations to a machine where WebCenter portal is deployed. Extract the jar content to a temporary directory on the machine, for example, /tmp/wc-cust.

4. Use the importMetadata WLST command to import these task flow customizations into the WebCenter Portal application’s MDS repository.

The following is the example command to import customization:

```wls:/weblogic/serverConfig>importMetadata(application='webcenter',
server='WC_Portal',
fromLocation='/tmp/wc-cust',
docs="/**")
```

For more details on the importMetadata WLST commands, see WLST Command Reference for Infrastructure Components.

Removing Customization

The customization applied in Applying Customizations can be removed if you choose. Once removed, the WebCenter Portal task flow behaviour or look and feel reverts to the original shipped product.

Use the deleteMetadata WLST command to remove the applied customization.

**Note:** Execute the deleteMetadata WLST command with caution, as incorrect use of this command may cause loss of metadata documents.

The following is the example for the deleteMetadata WLST command:

```deleteMetadata(application, server, docs, [restrictCustTo], [excludeAllCust],
[excludeBaseDocs], [excludeExtendedMetadata], [cancelOnException],
[applicationVersion])
```

For more information about the deleteMetadata WLST command, WLST Command Reference for Infrastructure Components.