

# WebCenter Portal Task Flow Customization in 12c

ORACLE WHITE PAPER | APRIL 2020





## 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

Disclaimer	1
Introduction	1
Understanding WebCenter Task Flow Customization Process	1
Preparing a Customizable WebCenter Portal Application	1
Creating Customization Classes	1
Making Your Customization Classes Available to JDeveloper	3
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	9
Setting up for WebCenter Portal Task Flows	9
Understanding Taskflow Customizations	11
Customizing the Search Service Task Flow Page Fragment	11
Customizing the Search Preferences Task Flow	13
Applying Task Flow Customizations	15
Applying Customization to the Deployed WebCenter Portal Application	17
Testing Page Fragment Customization	19
Testing Taskflow Customization	19
Removing Customization	20

## 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 below. The Studio Developer is the default role.

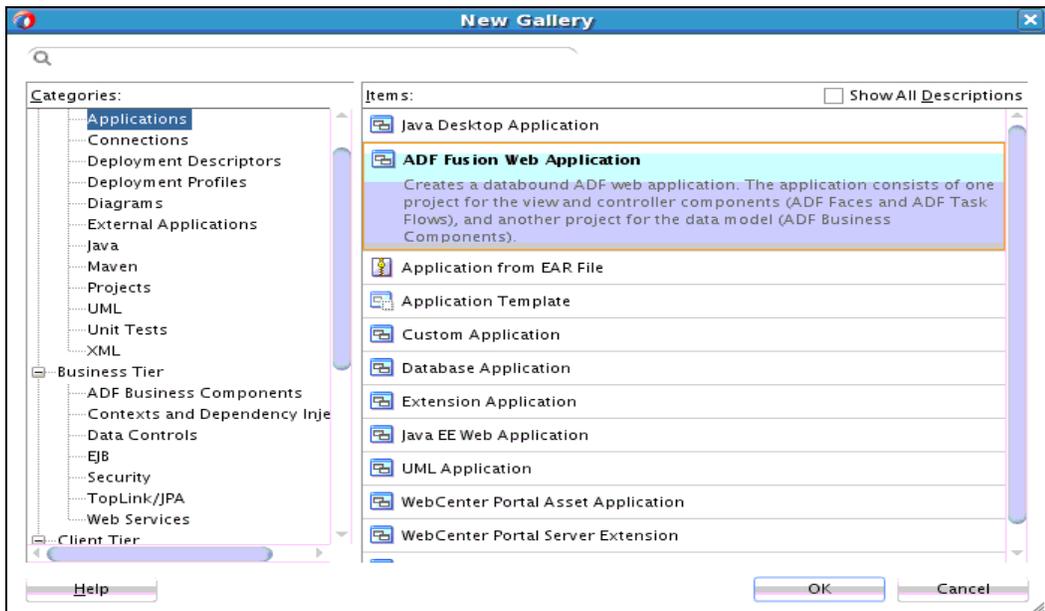


Figure 1. Create ADF Fusion Application

## 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. To create a Java class from the newly created ADF Fusion application, right-click ViewController project and select **New > From Gallery > Java > Class**.
2. In the application, ViewController project of your newly created ADF Fusion application, add a new java class named `WCSiteCC`, extending `oracle.mds.cust.CustomizationClass` as shown below.

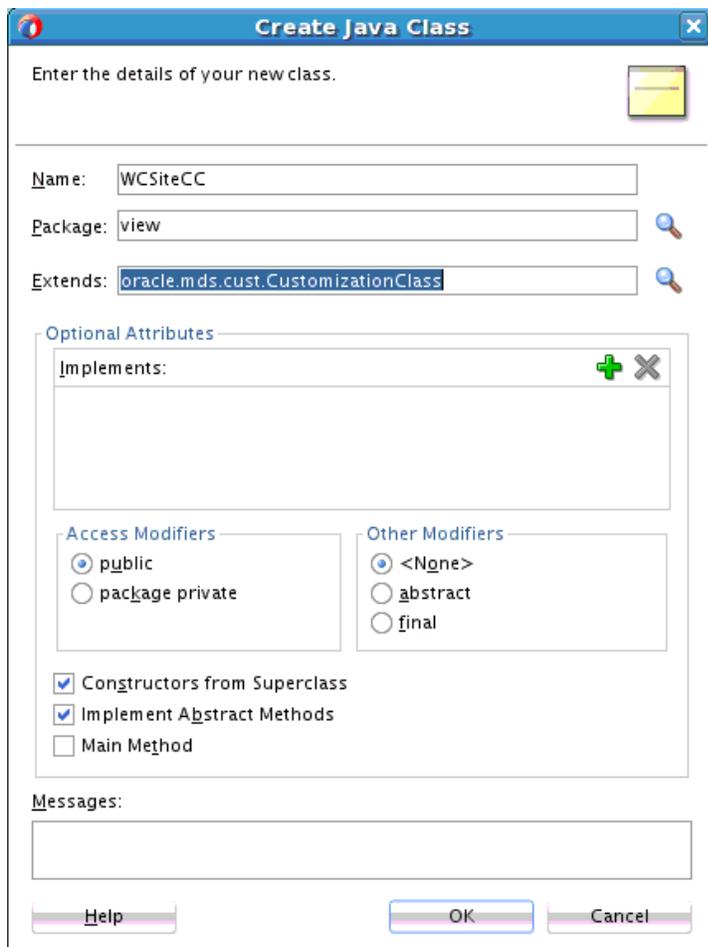


Figure 2. Create Customization Classes

3. In `WCSiteCC.java`, add the code. The following is a sample `WCSiteCC` Class code.

```

package view;

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() {
        return CacheHint.ALL_USERS;
    }

    @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"};
    }
}

```

4. 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:

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**, and click **New Profile**.
3. In the Create Deployment Profile dialog, 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**.



Figure 3. Create a Deployment Profile

5. In the Edit JAR Deployment Profile Properties, select **File Groups > Project Output > Filters**. Ensure that `WCsiteCC.class` is included as shown below.

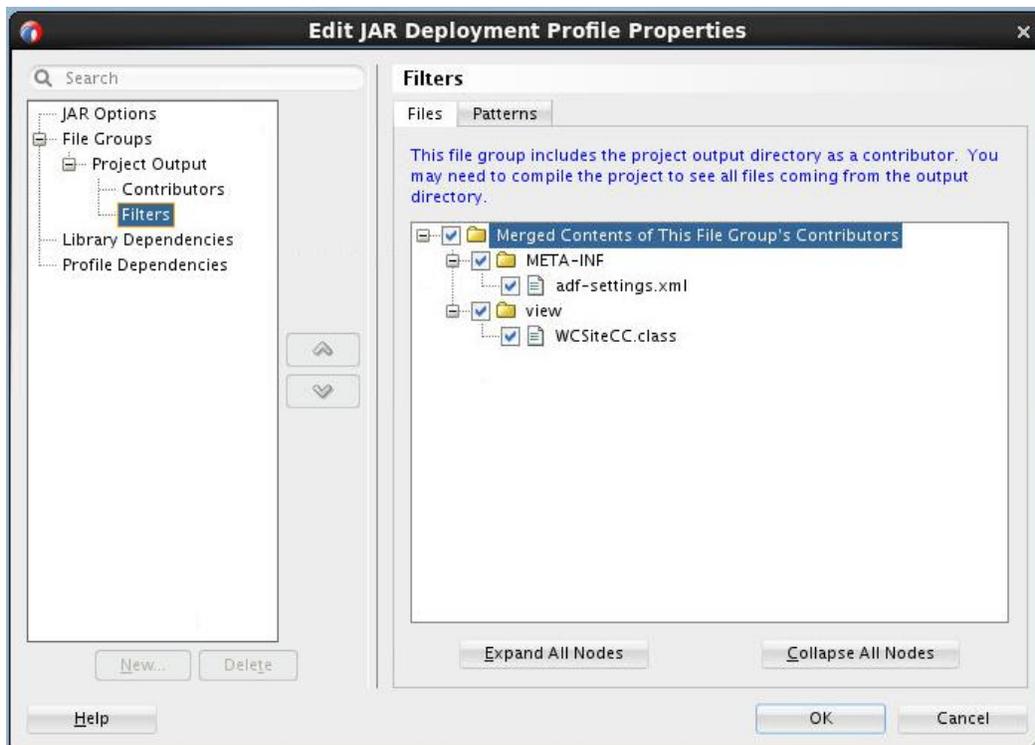


Figure 4. Edit the JAR Deployment Profile Properties

- 
6. Click **OK** to close.
  7. From the File menu, click **Save All**.
  8. 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`.
  9. 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.
  10. In the Application window of the JDeveloper, right-click the ViewController project, and select **Project Properties**.
  11. In the Project Properties dialog, select **Libraries and Classpath**, and then select **Add Jar/Directory** to add `myCC.jar` to view Controller's Classpath.
  12. Click **OK** to close the project properties.
  13. From the File menu, click **Save All**.

## 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 as shown below.

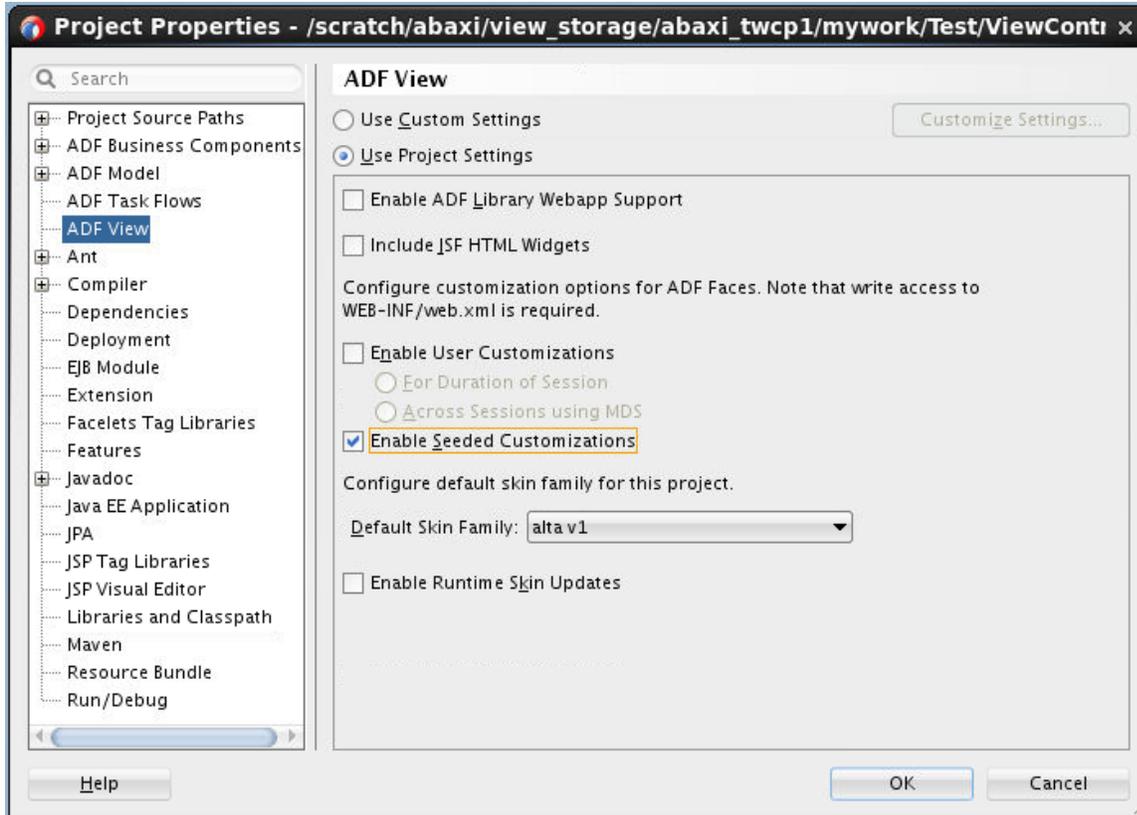


Figure 5. Enable Seeded Customization for View Projects

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

1. From the main menu, select **File** and then click **Open**.
2. Locate and open the file `CustomizationLayerValues.xml`, which is found in the following path:  
`<JDEV_HOME>/jdeveloper/jdev`
3. Add WebCenter value under site layer.

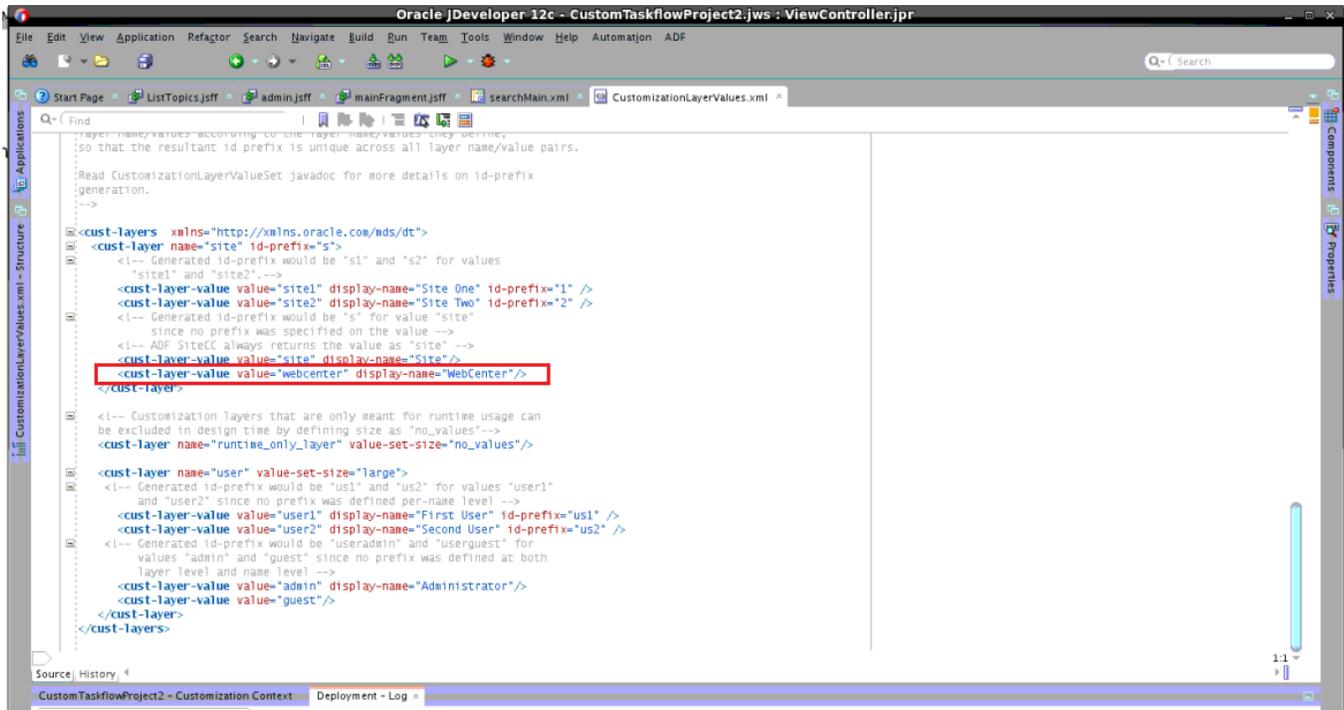


Figure 6. WebCenter Value under Site Layer

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.

4. From the **File Menu**, click **Save All**.

### 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 adf-config.xml file to add customization classes.

To add the customization classes:

1. The adf-config.xml file is located under **Application Resources > Descriptors > ADF META-INF** folder. Double click on the adf-config.xml file, it gets loaded on the right hand side.
2. On Overview editor, select **MDS**.
3. Add the WCSiteCC class to generate MDS Customization configuration.

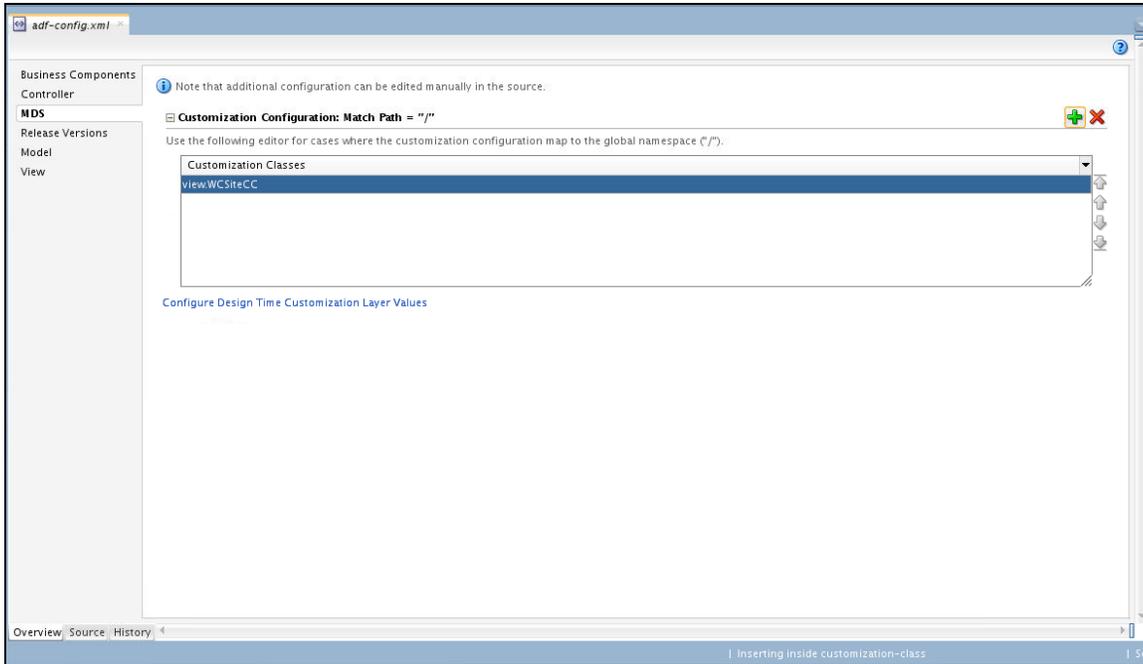


Figure 7. Add Customization Classes

4. From the File menu, click **Save All**.

The following is the example of the customization class order in the `adf-config.xml` file:

```
<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 one time 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 to your project. Perform the following steps:
  - a. Create a new directory, for example, TaskFlowLibraries under `<JDEV_HOME>/jdeveloper`.
  - b. Navigate to `<WC_ORACLE_HOME>/wcportal/webcenter/modules/oracle.webcenter.framework/` directory on the machine where WebCenter Portal server is installed.
  - c. Copy the jar of the taskflow that you want to edit to `<JDEV_HOME>/jdeveloper/TaskFlowLibraries` directory.
  - d. Now, we will be editing the Search Taskflows. To customize these taskflows, you have to copy the `search-service-view.jar` file to the directory `<JDEV_HOME>/jdeveloper/TaskFlowLibraries`.
2. Run the JDeveloper Studio in *Default Role* (Studio Developer).
3. In the application window's Projects palette, select **Libraries & Classpath**.
4. Click **Add Jar/Directory**.
5. Navigate to the `<JDEV_HOME>/jdeveloper/TaskFlowLibraries` directory and, select the `search-service-view.jar` file.
6. Ensure that the task flow jar files are included and then click **OK** to close 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**.

You can see all libraries that are included in the **ViewController** project in navigator.

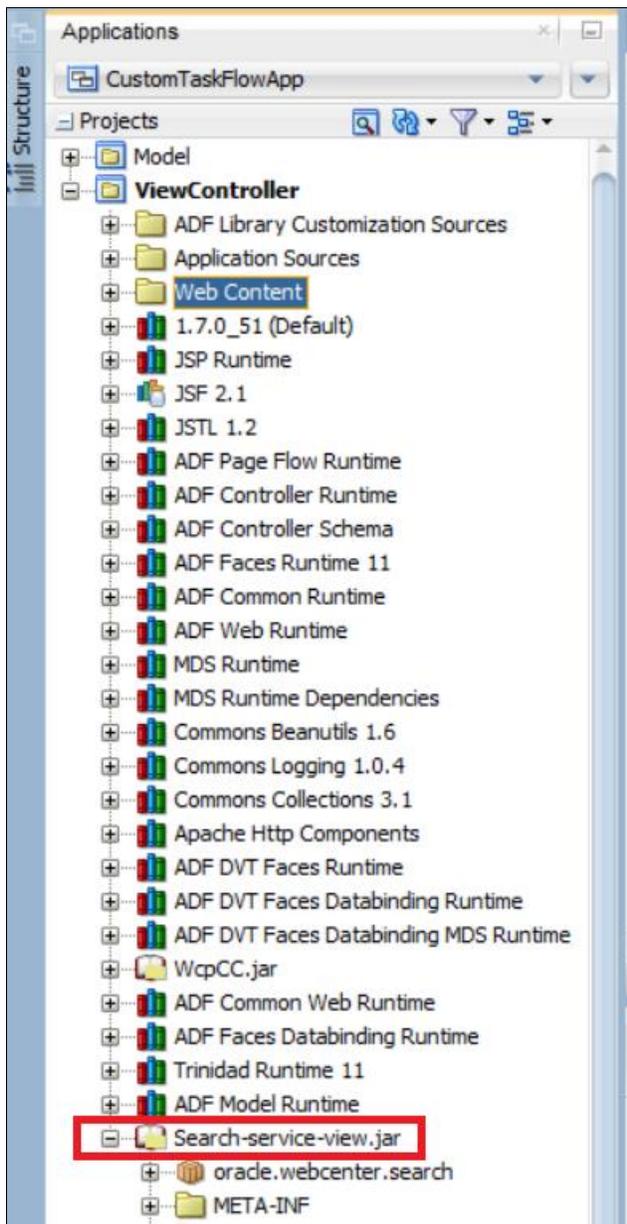


Figure 8. Application Projects Palette

9. Select **Tools > Switch Roles >Customization Developer** role.
10. Confirm if you want to restart JDeveloper in Customization role.
11. After JDeveloper opens in Customization role, select that the layer `site` is selected with `WebCenter` as layer value in the Customization Context window.

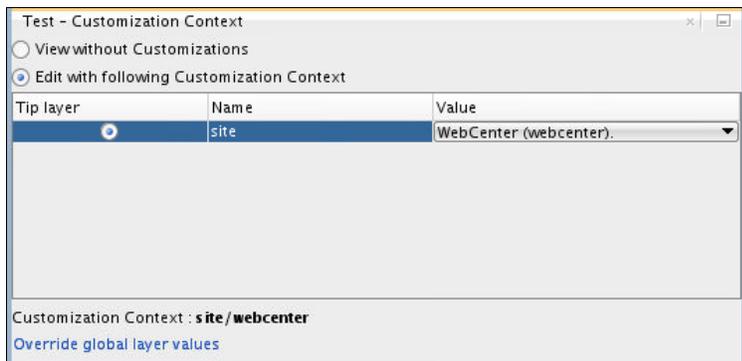


Figure 9. Customization Context

## Understanding Taskflow Customizations

To customize a task flow, open the task flow definition file.

Here are few things you can or can't do:

1. You can customize page fragments (.jsff) and taskflow definitions (.xml).
2. You can't customize classes even though you see them in the task flow jar files.
3. You need to do the updates from the Structure window (direct updates to the task flow file is disabled, for example, you can't do changes directly in a jsff file).
4. You can use the component property window to set values.
5. To do a customization, find the component you want to customize and select it so that it shows selected in the Structure window.  
Then using the Structure window add additional components or modify component properties.

## Customizing the Search Service Task Flow Page Fragment

The steps below gives us instructions to customize the page fragment. We will be adding a static text in the search service page fragment. To customize the page fragment:

1. Open the taskflow definition file of Search Taskflow (see [Figure 8](#), expand `oracle.webcenter.search` and navigate to `controller.taskflows.searchMain.xml`).
2. In the Diagram section, click **searchMain**, this will open the `mainFragment.jsff` file.  
In the `mainFragment.jsff` page, search for the tag `<af:outputFormatted>`. There are many occurrences of this tag. Select the tag one which has entry like this `<af:outputFormatted value="{pageFlowScope.serbb.searchMessage}" id="ot44"/>`
3. In the Structure pane, right-click `<af:outputFormatted>` tag, select **Insert Before Output Text (Formatted)**, then select **ADF Faces** as shown below.

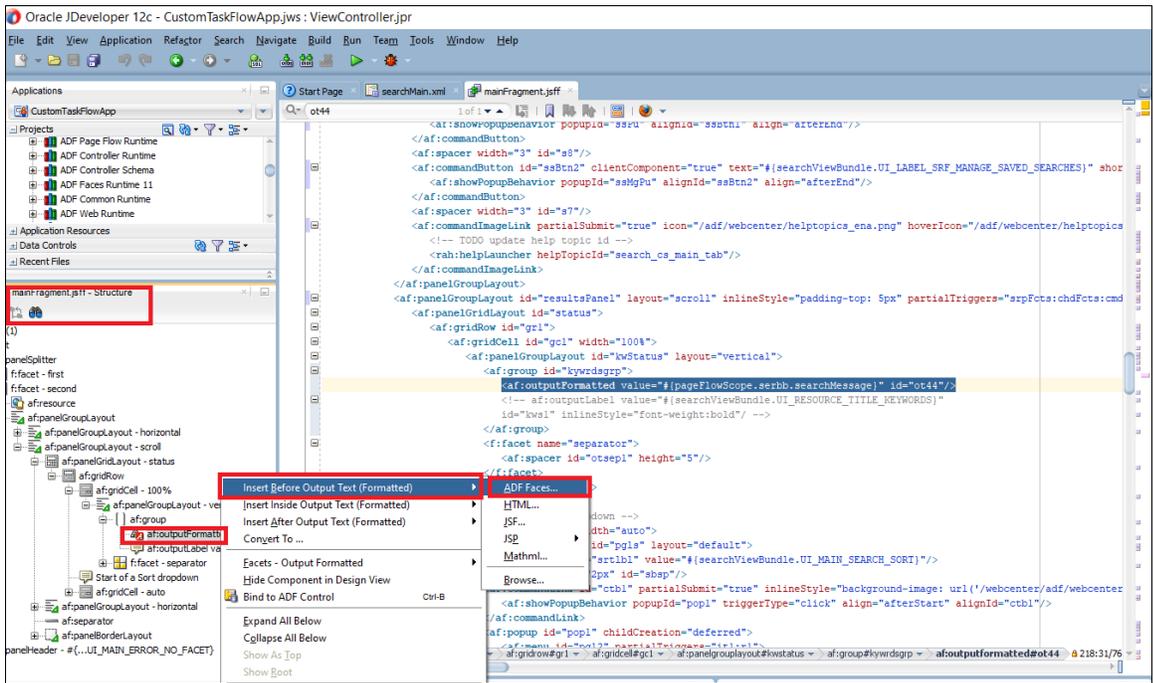


Figure 10. Adding Customization code

- Now, select Output Text (Label), the tag is dropped on the mainFragment.jsff page. In the Value section, click **Property Menu** and then select **Expression Builder** as shown below.

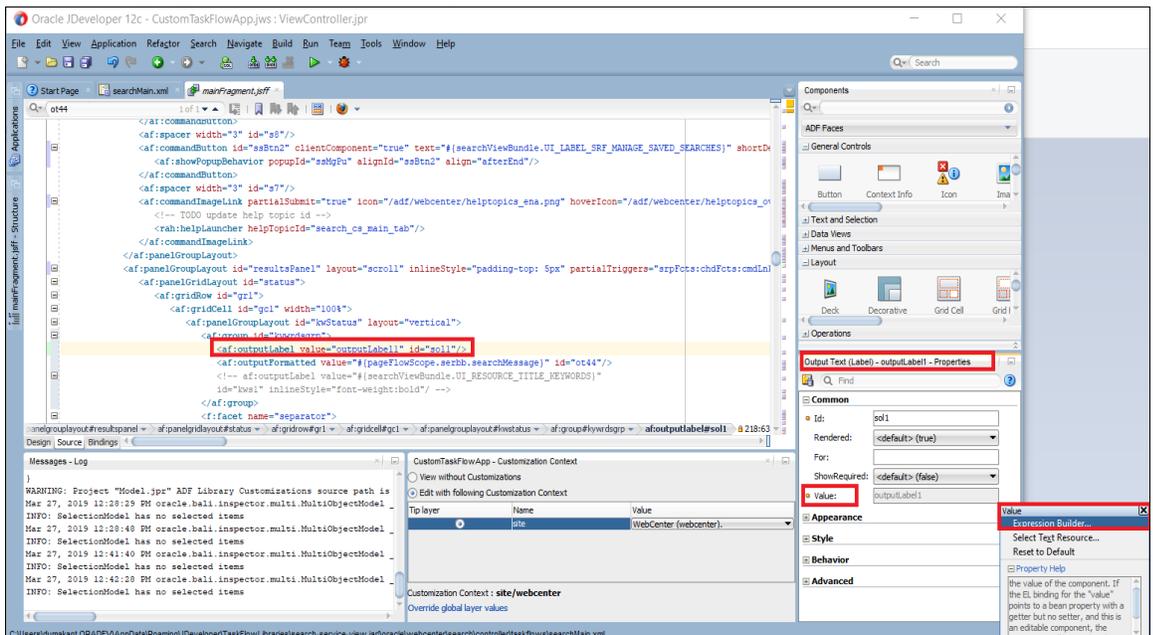
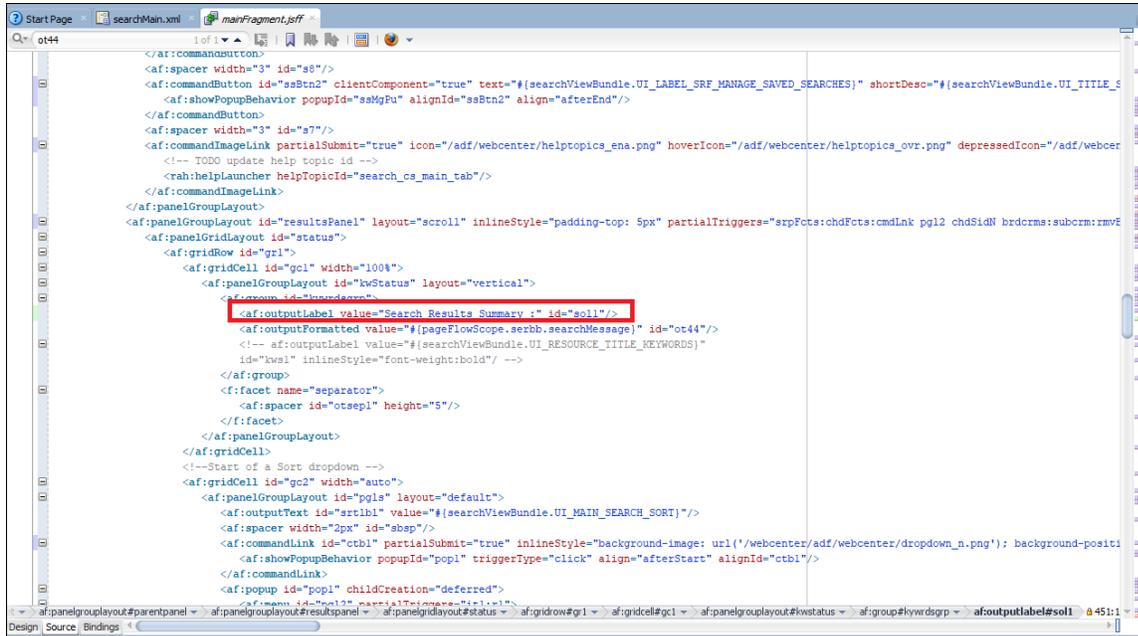


Figure 11. Populate the outputLabel

- Enter the value as *Search Results Summary:*.

6. You can see this value in the `mainFragments.jsff` as shown below.



```
</af:commandButton>
<af:spacer width="3" id="s8"/>
<af:commandButton id="ssBtn2" clientComponent="true" text="{searchViewBundle.UI_LABEL_SRF_MANAGE_SAVED_SEARCHES}" shortDesc="{searchViewBundle.UI_TITLE_
<af:showPopupBehavior popupId="ssMgPu" alignId="ssBtn2" align="afterEnd"/>
</af:commandButton>
<af:spacer width="3" id="s7"/>
<af:commandImageLink partialSubmit="true" icon="/adf/webcenter/helptopics_ena.png" hoverIcon="/adf/webcenter/helptopics_ovr.png" depressedIcon="/adf/webce
<!-- TODO update help topic id -->
<rah:helpLauncher helpTopicId="search_cs_main_tab"/>
</af:commandImageLink>
</af:panelGroupLayout>
<af:panelGroupLayout id="resultsPanel" layout="scroll" inlineStyle="padding-top: 5px" partialTriggers="srpFcts:cmdLnk pgL2 chdSidN brdcrms:subcrm:rmv
<af:panelGridLayout id="status">
<af:gridRow id="gr1">
<af:gridCell id="gcl" width="100%">
<af:panelGroupLayout id="kwStatus" layout="vertical">
<af:row id="kwStatus">
<af:outputLabel value="Search Results Summary" id="sol1"/>
<af:outputFormatted value="{pageFlowScope.serbb.searchMessage}" id="ot44"/>
<!-- af:outputLabel value="{searchViewBundle.UI_RESOURCE_TITLE_KEYWORDS}"
id="kws1" inlineStyle="font-weight:bold" -->
</af:group>
<f:facet name="separator">
<af:spacer id="otsepl" height="5"/>
</f:facet>
</af:panelGroupLayout>
</af:gridCell>
<!-- Start of a Sort dropdown -->
<af:gridCell id="gcl" width="auto">
<af:panelGroupLayout id="pgL3" layout="default">
<af:outputText id="srTb1" value="{searchViewBundle.UI_MAIN_SEARCH_SORT}"/>
<af:spacer width="2px" id="shsp"/>
<af:commandLink id="ctbl" partialSubmit="true" inlineStyle="background-image: url('/webcenter/adf/webcenter/dropdown_n.png'); background-positi
<af:showPopupBehavior popupId="pop1" triggerType="click" align="afterStart" alignId="ctbl"/>
</af:commandLink>
<af:popup id="pop1" childCreation="deferred">
</af:popup>
</af:panelGroupLayout>
</af:gridRow>
</af:gridLayout>
</af:panelGroupLayout>
</af:parentPanel>
```

Figure 12. The Customized `mainFragments.jsff`

7. From the **File Menu**, click **Save All**.

### Customizing the Search Preferences Task Flow

The steps below gives us instructions to customize the search preference taskflow. Here, we will be introducing a new taskflow parameter.

To customize the taskflow:

1. Open the taskflow definition file of Search Preferences. For example, `/oracle/webcenter/search/controller/taskflows/preferences.xml`.
2. In the Structure pane, right-click on **task-flow-definition-search-preferences**, select **Insert Inside Task Flow Definition**, then click **ADF Task Flow** as shown below.

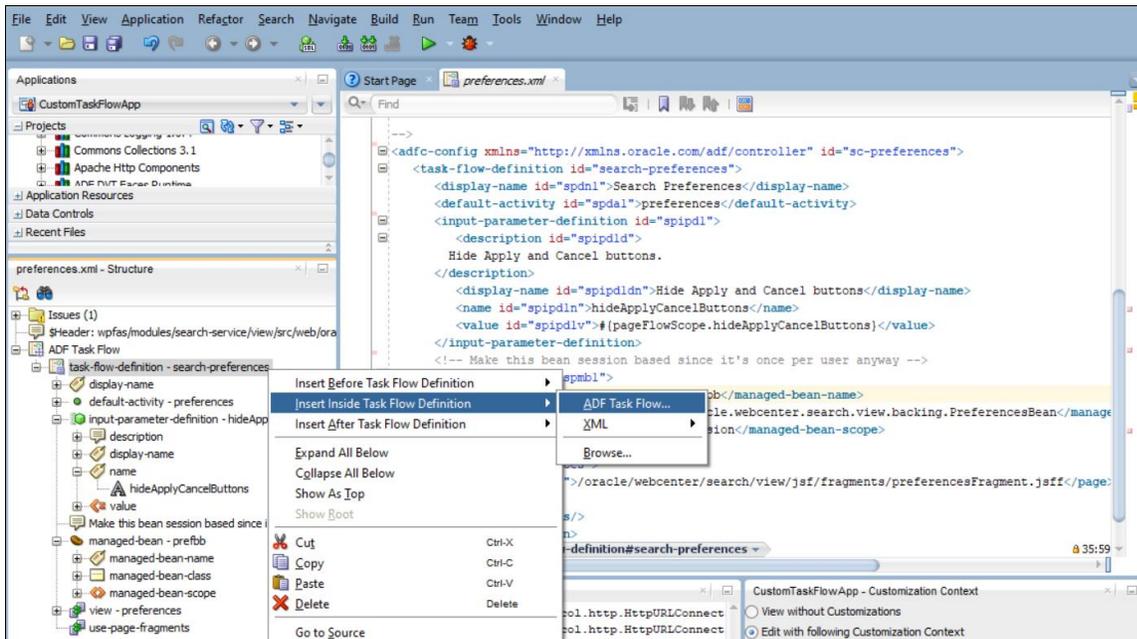


Figure 13. Adding Taskflow Input Parameter

3. Insert ADF Task Flow Item is launched. Select **Input Parameter Definition** and click **OK**.
4. Insert Input Parameter Definition popup will appear.
5. Enter the name, display name, description, and value fields. For example,
  - Name:** *testBooleanParam*
  - DisplayName:** *Test Boolean Parameter*
  - Value:** *true*
  - Description:** *Testing Addition of Input Parameter*
6. Click **OK**. You can see in the Source window that a new parameter is added as shown below.

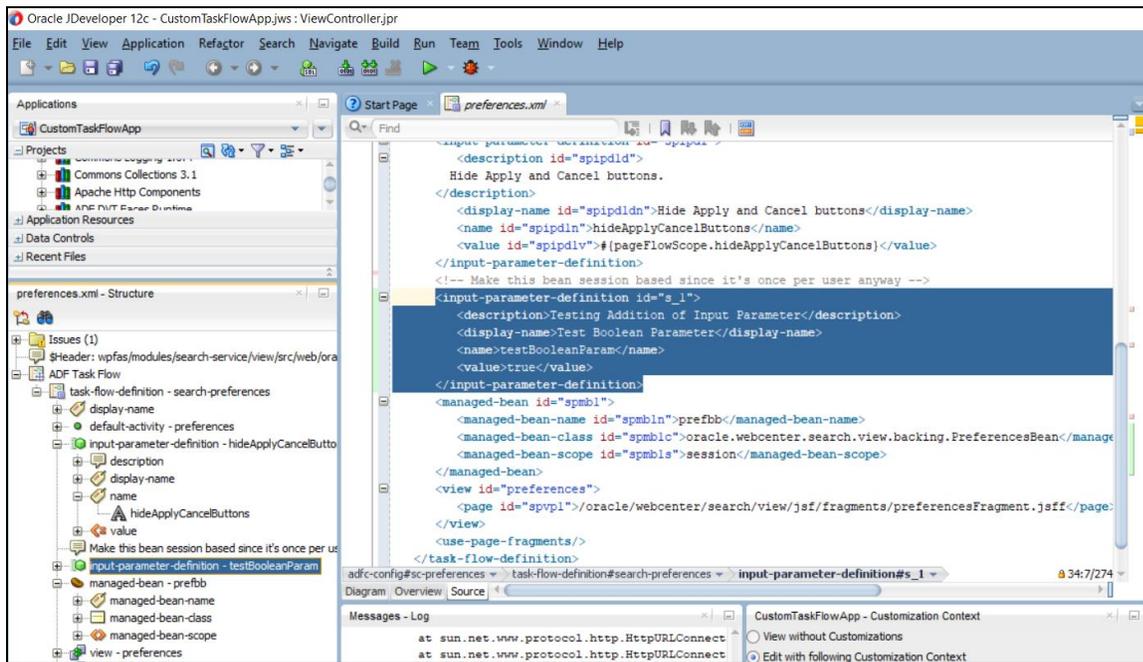


Figure 14. Taskflow Parameter Inserted

7. From the **File Menu**, click **Save All**.

## Applying Task Flow Customizations

The output of the task flow customization is the generated MDS customization. The customizations show up as the .xml.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.

To apply your task flow customizations:

1. Open JDeveloper in Default role (Studio Developer, All features).
2. Rebuild the application with your task flow customizations. To do this, right-click the ViewController project and select **Rebuild ViewController.jpr**.
3. Create a JAR deployment profile to package the task flow customizations.
  - a. Right-click the ViewController project and select **Project Properties**.
  - b. In the Project Properties dialog, select **Deployment**, and then click **New Profile**.
  - c. In the Create Deployment Profile dialog, select jar file from the **Profile Type** drop-down list. Enter the deployment profile name, for example, *TF-Customizations*. Click **OK**.
  - d. In the JAR Deployment Profile Properties, select **File Groups > Project Output > Contributors**.
  - e. Under Contributors, click **Add**.
  - f. Select the **ViewController > libraryCustomizations** directory and click **Select**.

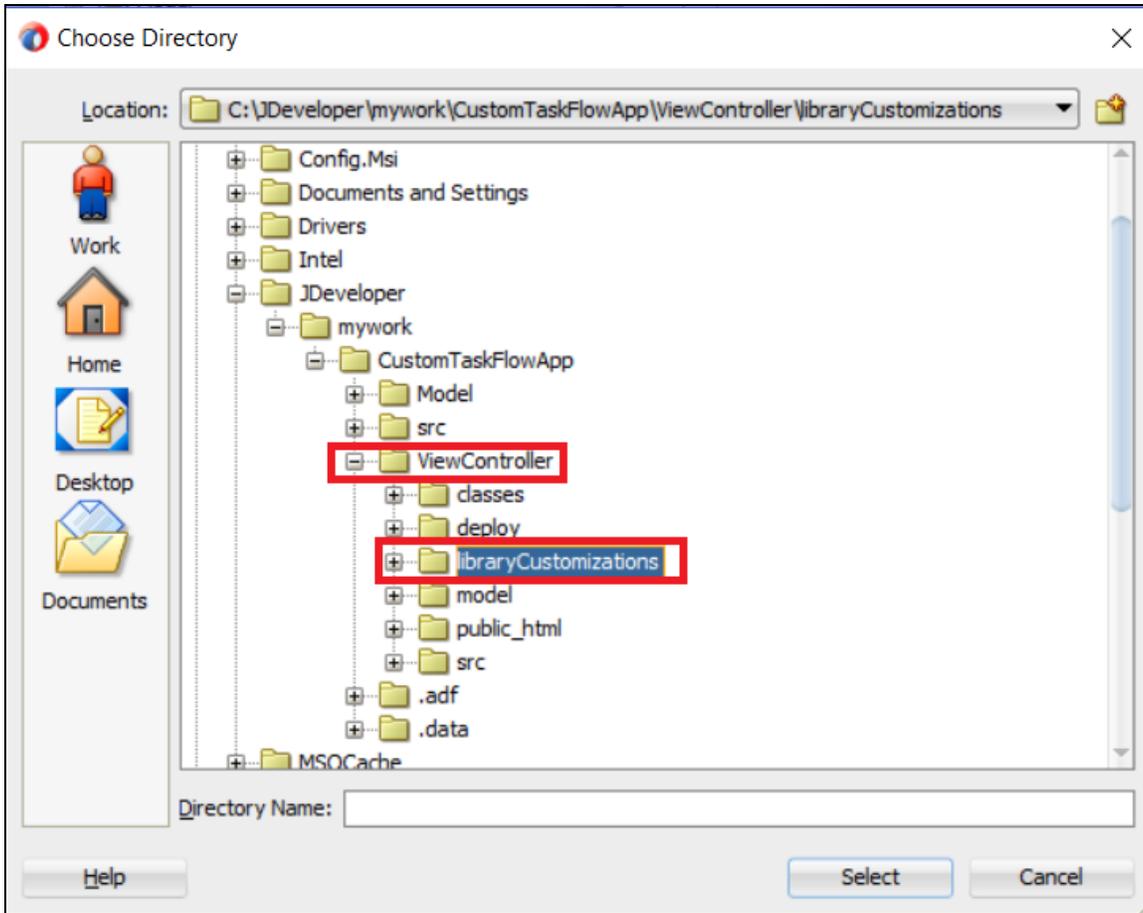


Figure 15. Add Customization Library

- g. Click **OK** to add the selected directory to the Contributors for the deployment profile.
- h. Confirm the *ViewController/libraryCustomizations* directory shows under **Contributors** and click **OK** to close the project properties as shown below.

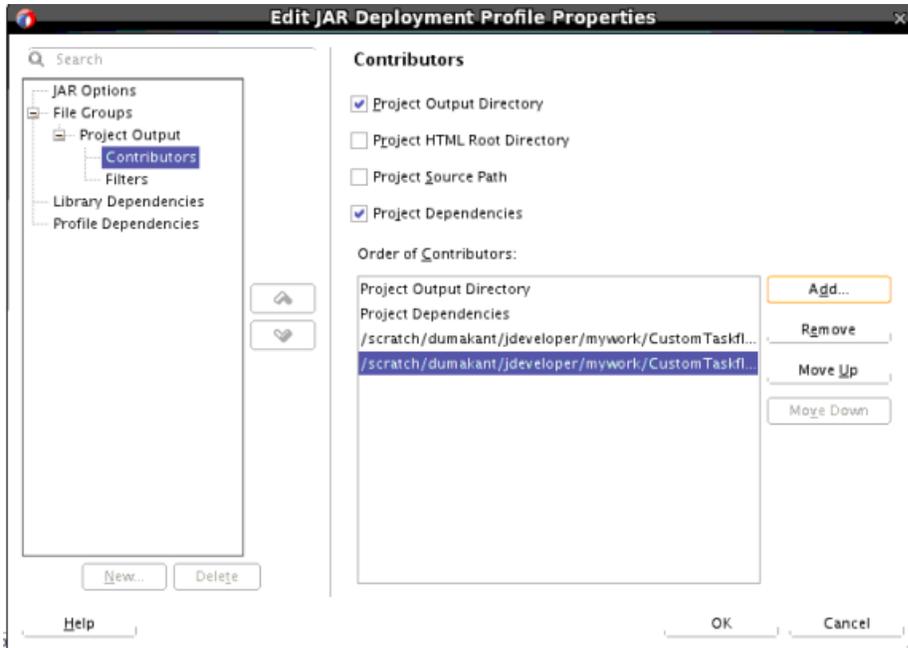


Figure 16. Edit JAR Deployment Profile Properties

- i. Right-click the ViewController project and select **Deploy**, then select the newly created deployment profile. For example, *TF-Customizations*.
- j. In the Deploy dialog, select **Deploy to JAR File** and click **Next**.
- k. Click **Finish** to complete the deployment to the JAR file.
- l. Note down the path of the customizations jar file,  
`<TaskFlowCustomizationApp>\ViewController\deploy\TF-Customizations.jar`

### 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.

1. Copy the `TF-Customizations.jar` file to the machine where you have the WebCenter domain and place in a stage folder. For example, `/tmp/wc-cust/TF-Customizations.jar`
2. Extract the contents of the task flow customization JAR file in the `/tmp/wc-cust` folder. To extract the contents of the jar file, do the following: `jar xvf TF-Customizations.jar`

```
inflating: META-INF/MANIFEST.MF
```

```
extracting: META-INF/adf-settings.xml
```

```
extracting:
```

```
oracle/webcenter/search/controller/taskflows/mdssys/cust/site/webcenter/preferences.xml.xml
```

```
extracting:
```

```
oracle/webcenter/search/view/jsf/fragments/mdssys/cust/site/webcenter/mainFragment.jsff.xml
```

```
extracting: view/WCSiteCC.class
```

You can remove the `TF-Customizations.jar` from the `/tmp/wc-cust` location.

3. 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='/**')
```

```
Executing operation: importMetadata.
```

```
Operation "importMetadata" completed. Summary of "importMetadata" operation is:  
5 documents successfully transferred. List of
```

```
documents successfully transferred from directory /refresh/oracle/stage/wc-cust:
```

```
/META-INF/MANIFEST.MF
```

```
/META-INF/adf-settings.xml
```

```
/oracle/webcenter/search/controller/taskflows/mdssys/cust/site/webcenter/preferences.xml.xml
```

```
/oracle/webcenter/search/view/jsf/fragments/mdssys/cust/site/webcenter/mainFragment.jsff.xml
```

```
/view/WCSiteCC.class
```

For details, see [WLST Command Reference for Infrastructure Components](#).

4. Restart is not required. You can now proceed for testing.

## Testing Page Fragment Customization

Access WebCenter instance and perform search, now notice the newly added String.

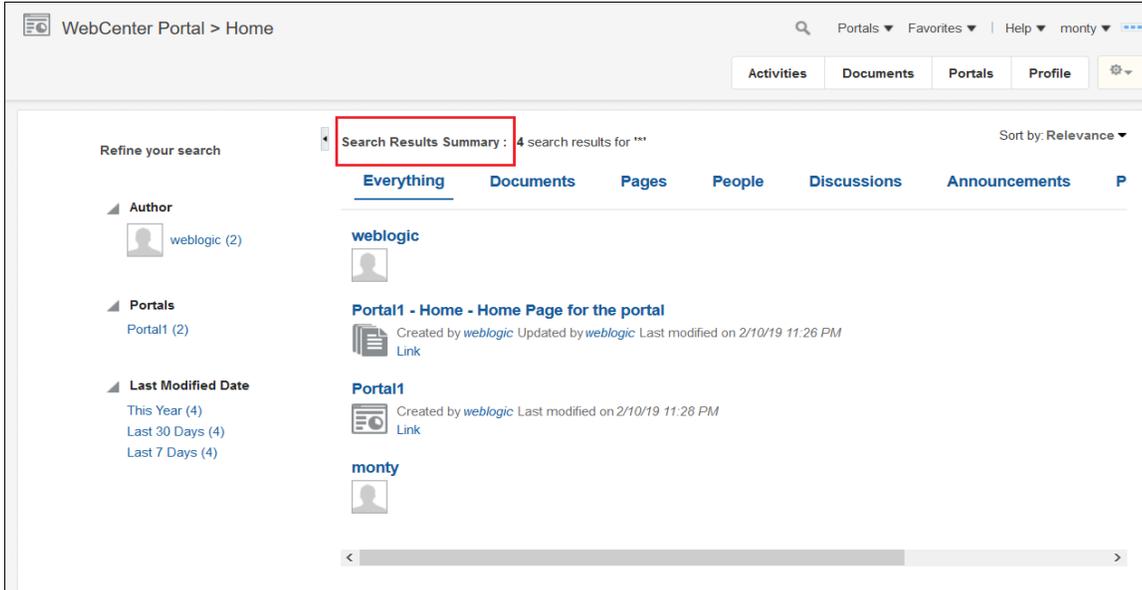


Figure 17. Customized Search Page

## Testing Taskflow Customization

1. Create a portal and then create a new page in portal. Click **Edit Portal**.
2. Now drop the Search Preferences taskflow on the page. If you do not see the search preferences taskflow in the resource catalog, you can follow the instructions given in the document to add this taskflow in the resource catalog. For information, see [Adding a Resource from the Resource Registry](#) and [Changing the Resource Catalogs in a Portal](#).

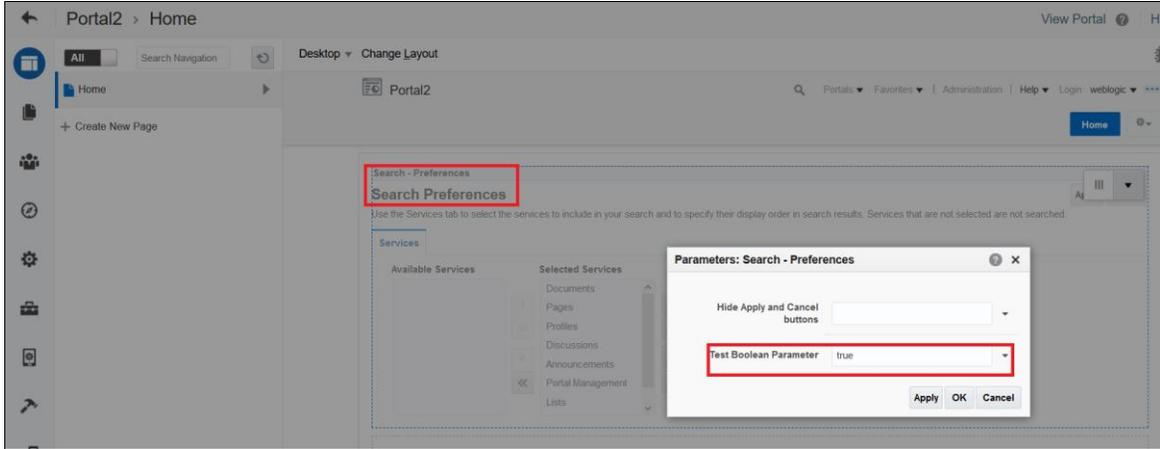


Figure 18. Taskflow Parameter

3. After the Search Preference Taskflow is dropped on the page, click **Actions Menu > Parameters** option.
4. You can see the newly created taskflow parameter in the Parameters popup.

### Removing Customization

The customization applied in (see [Applying Customizations](#)) can be removed. 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 `deleteMetadata` WLST command:

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

For example:

```
deleteMetadata(application='webcenter', server='WC_Portal', docs='/view/WCSiteCC.class')
```

For information about the `deleteMetadata` WLST command, see [WLST Command Reference for Infrastructure Components](#).



CONNECT WITH US

-  [blogs.oracle.com/oracle](https://blogs.oracle.com/oracle)
-  [facebook.com/oracle](https://facebook.com/oracle)
-  [twitter.com/oracle](https://twitter.com/oracle)
-  [oracle.com](https://oracle.com)

**Oracle Corporation, World Headquarters**

500 Oracle Parkway  
Redwood Shores, CA 94065, USA

**Worldwide Inquiries**

Phone: +1.650.506.7000  
Fax: +1.650.506.7200

**Hardware and Software, Engineered to Work Together**

Copyright © 2016, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group. 0420

White Paper Title  
April 2020  
Author: Nitin Shah  
Contributing Authors: Amit Baxi