

Creating a WebCenter Portal Page Template Based on a Bootstrap Theme

ORACLE WHITE PAPER | APRIL 2016





Table of Contents

| | |
|---|----|
| Introduction | 2 |
| Downloading a Bootstrap Theme | 3 |
| Create and Deploy a Web Application to Host Bootstrap Theme Resources | 3 |
| Creating a New Portal Page Template | 3 |
| Adding Bootstrap Theme Components and Features to a Page Template | 4 |
| Adding Portal-Specific Items to a Page Template | 6 |
| Portal Icon and Name | 7 |
| Portal Links | 7 |
| Portal Search or Language | 7 |
| Portal Navigation | 8 |
| Creating a New Skin for a Page Template | 8 |
| Styling Components and Task Flows for a Page Template | 9 |
| Examples | 10 |
| Example 1 | 10 |
| Adding static Bootstrap theme HTML to a WebCenter Portal page template | 10 |
| Example 2 | 11 |
| Using a Bootstrap theme to style the portal icon, name, and description | 11 |
| Example 3 | 12 |
| Using jQuery, ADF skinning, a font-awesome icon, and Bootstrap theme CSS to style a portal Login link | 12 |
| Testing a Page Template in WebCenter Portal | 14 |



Introduction

Page templates provide the structure for common areas in portal pages. They define how individual pages and groups of pages display on a user's screen, and help to ensure that the pages in a portal are consistent in structure and layout. If you change a page template, then all pages that reference that template automatically inherit the changes.

Bootstrap is currently one of the most popular HTML5, CSS3, and JavaScript frameworks for developing modern, responsive, mobile-first web pages and sites. With Bootstrap themes, you can build responsive page templates for portal pages very quickly by saving development time in the initial styling and skinning of a web site. You can select from thousands of available Bootstrap themes to customize your page template to the unique needs of your portal.

WebCenter Portal includes two built-in Bootstrap-themed page templates: Mosaic and Unicorn. You can build WebCenter Portal page templates using these built-in page templates as a starting point. Or, you can purchase your own Bootstrap themes from any Bootstrap marketplace site and use them to create custom WebCenter Portal page templates.

This white paper describes how to build a page template that uses a Bootstrap theme. The following are the main steps involved in this process:

1. Download a Bootstrap theme
2. Create and deploy a web application to host the resources of the Bootstrap theme
3. Create a new portal page template
4. Add Bootstrap theme components and features to the page template
5. Add portal specific items to the page template
6. Create a skin for the page template
7. Style components and task flows for a page template

Downloading a Bootstrap Theme

You may already know the name of the Bootstrap theme that you wish to use for your new portal page template, or you may want to browse for ideas.

To search and download a Bootstrap theme:

1. From any commercial Bootstrap theme provider, purchase and download the theme you wish to use for your portal page template.
2. In a folder on your local computer, unzip the downloaded Bootstrap theme. The directory will include HTML pages and CSS/JavaScript files. The directory structure will look something like the following:

```
/HTML
  /assets
    /css
    /img
    /plugins
    /js
    /php
    /ajax
```

Create and Deploy a Web Application to Host Bootstrap Theme Resources

The size of a Bootstrap theme's web resources, such as Javascript, images, and CSS, could be of approximately 40MB. To keep the page template archive small, and for Bootstrap resources to be served faster, these resources can be put on a CDN or the web server's static HTML directory.

Another approach that can be used during the development of a page template is to put these resources in a web application and deploy the application to the application server. You can create an application based on the *ADF Fusion Web Application* template. Since the resources of a Bootstrap theme reside under the `/HTML/assets` directory of the downloaded zip file, this directory can be copied as-is to the `public_html` directory of the newly created application's `ViewController` project. After deploying the application to the application server, the Bootstrap theme's assets can be referenced using the following URL format:

```
http://<host>:<port>/<context root>/HTML/assets/
```

Where, `<host>:<port>` refers to the host and the port where the web application is deployed, and `<context root>` refers to the context root specified for the web application. In the sample code given in this whitepaper, the context root of the application is set to `/bootstrap`.

Creating a New Portal Page Template

While you can develop page templates in WebCenter Portal, the editing capabilities are limited. Oracle recommends that you use JDeveloper to develop page templates for portals. When fully developed, you can publish a page template directly to WebCenter Portal (the portal server) for immediate use or for testing. Alternatively, you can export the page template to a file and upload the page template to WebCenter Portal later.

To create a new page template in JDeveloper, see [“Creating a Page Template”](#) in *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

Adding Bootstrap Theme Components and Features to a Page Template

To incorporate the components and features you require from the Bootstrap theme you downloaded:

1. In JDeveloper, edit the page template you have created.

See [“Editing a Page Template”](#) in *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

2. At the beginning of the page template file, in the component section, add a content facet if it does not already exist. For example:

```
<component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
  <display-name>Template Display Name</display-name>
  <facet>
    <facet-name>content</facet-name>
  </facet>
  ...
</component>
```

3. In the main content area of the page template file, add coding for the general structure of the page template (you can refer to the built-in Mosaic or Unicorn page templates for examples of this section). For example:

```
<af:pageTemplateDef var="attrs">
  <af:xmlContent>
    <component xmlns="http://xmlns.oracle.com/adf/faces/rich/component">
      <display-name>template</display-name>
      <facet>
        <description>The content of the page goes here</description>
        <display-name>Page Content</display-name>
        <facet-name>content</facet-name>
      </facet>
    </component>
  </af:xmlContent>
  <af:panelGroupLayout id="mainpگلwrapper" layout="default"
styleClass="WCNonEditableArea">
    <meta content="width=device-width, initial-scale=1" name="viewport"/>
    <!-- Begin Bootstrap CSS section -->
    ...
    <!-- End Bootstrap CSS section -->
    <!-- Begin required hidden portal template components-->
    ...
    <!-- End required hidden portal template components -->
    <!-- Begin required visible portal template components -->
    ...
    <!-- End required visible portal template components -->
    <div class="WCSTemplateRoot" id="wrapper">
      <!-- Begin template content section -->
      <div class="wrapper">
        <!-- Begin header section -->
        <div class="header">
          <div class="container">
            <!-- Begin Logo -->
            ...
            <!-- End Logo -->
            <!-- Begin Topbar -->
            ...
            <!-- End Topbar -->
            <!-- Begin Toggle for better mobile display -->
            ...
          </div>
        </div>
      </div>
    </div>
  </af:panelGroupLayout>
```

```

        <!-- End Toggle -->
    </div>
    <!-- Begin Navigation Section -->
    ...
    <!-- End Navigation Section-->
</div>
<!-- End header section -->
<!-- Begin Page Content Section -->
...
<!-- End PAGE Content Section -->
</div>
<!-- End Template Content Section -->
<!-- Begin footer section -->
...
<!-- End footer section -->
</div>
<!-- Begin Bootstrap JavaScript section -->
...
<!-- End Bootstrap JavaScript section -->
...
</af:pageTemplateDef>

```

Note: font-awesome 4.1.0, bootstrap 3.2.0, and jQuery 1.11.1 CSS and JavaScript files are already included in WebCenter Portal, so do not need to be copied or uploaded.

4. In the CSS section of your page template file, add CSS references for your Bootstrap theme. For example:

```

<!-- Begin Bootstrap CSS section -->
<!-- Web Fonts -->
<link rel='stylesheet' type='text/css'
href='//fonts.googleapis.com/css?family=Open+Sans:400,300,600&subset=cyrillic,latin' />
<!-- CSS Global Compulsory -->
<link rel="stylesheet"
href="/bootstrap/unify/HTML/assets/plugins/bootstrap/css/bootstrap.min.css"/>
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/css/style.css"/>
<!-- CSS Header and Footer -->
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/css/headers/header-
default.css"/>
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/css/footers/footer-v1.css"/>
<!-- CSS Implementing Plugins -->
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/plugins/animate.css"/>
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/plugins/line-icons/line-
icons.css"/>
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/plugins/font-awesome/css/font-
awesome.min.css"/>
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/plugins/parallax-
slider/css/parallax-slider.css"/>
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/plugins/owl-carousel/owl-
carousel/owl.carousel.css"/>
<!-- CSS Customization -->
<link rel="stylesheet" href="/bootstrap/unify/HTML/assets/css/custom.css"/>
<!-- End Bootstrap CSS section -->

```

5. In the JavaScript section of your page template file, add JavaScript references for your Bootstrap theme. For example:

```

<!-- Begin Bootstrap JavaScript section -->
<!-- JS Global Compulsory -->
<script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/jquery/jquery.min.js"></script>
<script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/jquery/jquery-migrate.min.js"></script>
<script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/bootstrap/js/bootstrap.min.js"></script>
<!-- JS Implementing Plugins -->

```

```

    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/back-to-top.js"></script>
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/smoothScroll.js"></script>
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/parallax-
slider/js/modernizr.js"></script>
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/plugins/parallax-
slider/js/jquery.cslider.js"></script>
    <script type="text/javascript"
        src="/bootstrap/unify/HTML/assets/plugins/owl-carousel/owl-
carousel/owl.carousel.js"></script>
    <!-- JS Customization -->
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/js/custom.js"></script>
    <!-- JS Page Level -->
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/js/app.js"></script>
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/js/plugins/owl-carousel.js"></script>
    <script type="text/javascript"
src="/bootstrap/unify/HTML/assets/js/plugins/parallax-slider.js"></script>
    <script type="text/javascript">
        jQuery(document).ready(function ()
        {
            App.init();
            OwlCarousel.initOwlCarousel();
            ParallaxSlider.initParallaxSlider();
        });
    </script>
    <!-- End Bootstrap JavaScript section -->

```

6. From the downloaded Bootstrap theme HTML files, copy HTML code snippets for desired page elements and paste into your new page template file (.jspx or .jsf). Page elements may include a logo, title, description, navigation, or footer. See “Example 1: Adding static Bootstrap theme HTML to a WebCenter Portal page template”.
7. Test the page template in WebCenter Portal to see the styling of the downloaded Bootstrap theme. See “Testing a Page Template in WebCenter Portal”.

Adding Portal-Specific Items to a Page Template

After you copy HTML code snippets from the downloaded Bootstrap theme files for desired page elements into your new page template, you can modify those page elements so that they use WebCenter Portal-specific content built with ADF faces components. Replacing Bootstrap theme page elements with ADF faces components allows your page template to expose dynamic content retrieved from the portal for the following elements:

- » Portal Icon, Name, or Description
- » Portal Links
- » Portal Search or Language
- » Portal Navigation

Note: You can find the full source of the code snippets provided in this section in the built-in Mosaic and Unicorn page templates.

Portal Icon and Name

To add a portal icon and name to the page template (all linked to the portal Home page), which are set in the portal settings, to your page template:

1. Copy the following `af:switcher` placeholder code snippet to your page template file (jspx or jsf):

```
<af:switcher facetName="#{WCApPContext.currentScope.default ? 'defaultLogo' :
    'spaceLogo'}">
    ...
</af:switcher>
```

2. Refer to “Example 2” on page 11 to modify this placeholder code to use the skinning of your selected Bootstrap theme.

Portal Links

To add links such as Administration, Favorites, Portals, Profile, Preferences, Login, Logout, Self-Registration, and Help to your page template:

1. Copy the following `wcdc` utility tags placeholder code snippet to your page template file (jspx or jsf). As shown in the following example for the Login link:

```
<wcdc:spacesAction id="wcLoginLink"
shortDesc="#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}"
text="#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}" type="login"/>
```

2. Refer to “Example 3” on page 12 to modify this placeholder code to use the skinning of your selected Bootstrap theme.

Portal Search or Language

To add Search or Language task flows to your page template:

1. Copy the following `af:region` code snippets to your page template file (jspx or jsf):

```
<af:region id="searchbox" value="#{bindings.localToolbarSearch.regionModel}"/>
...
<af:region id="lang" value="#{bindings.languageSelectorRegion.regionModel}"
    styleClass="text"/>
```

2. Add the search and language task flow definitions to your page template `pagedef.xml` file:

```
<taskFlow activation="deferred" id="localToolbarSearch"
...
</taskFlow>
<taskFlow Refresh="ifNeeded" activation="conditional"
    active="#{!security.authenticated}" id="languageSelectorRegion"
...
</taskFlow>
```

3. Use the same techniques illustrated in “Example 2” on page 11 and “Example 3” on page 12 to modify this placeholder code to use the skinning of your selected Bootstrap theme.

Portal Navigation

To add portal navigation to your page template:

1. Copy the following `af:iterator` code snippets to your page template file (jsp or jsf):

```
<af:iterator id="il" var="side_menu_item"
  value="#{navigationContext.defaultNavigationModel.listModel
  ['startNode=/, includeStartNode=false']}">
</af:iterator>
```

2. Use the same techniques illustrated in “Example 2” on page 11 and “Example 3” on page 12 to modify this placeholder code to use the skinning of your selected Bootstrap theme, as desired.

Creating a New Skin for a Page Template

The Bootstrap theme CSS selectors are very general and could have undesirable effects on the styling of ADF Faces components. Instead of changing the Bootstrap CSS style, a custom skin can be created to restore the existing styling of some ADF Faces components or to change the styling of existing UI components to match the Bootstrap theme style.

Creating and editing a skin requires knowledge of CSS and how the skin is used in page templates. Oracle recommends that you use JDeveloper to develop a skin for portal page templates. When fully developed, you can publish a skin directly to WebCenter Portal (the portal server) or to a specific portal for immediate use or for testing. Alternatively, you can export the skin to a file and upload the skin to WebCenter Portal later.

For a Bootstrap-themed page template skin, extend the WebCenter Portal Alta built-in skin (in Shared Assets) or copy all WebCenter Portal Alta skin selectors as the starting point for your new page template. This establishes the default portal Alta look for portal components. It is also important for the custom skin to be based on WebCenter Portal Alta skin so the Portal Builder works correctly.

To create a new skin in JDeveloper, see [“Creating a Skin”](#) in *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

For information about how to copy a skin (a WebCenter Portal asset), see [“Copying an Asset”](#) in *Building Portals with Oracle WebCenter Portal*.

Styling Components and Task Flows for a Page Template

To style the components and task flows in your page template:

1. In JDeveloper, edit the skin you have created.
See [“Editing a Skin”](#) in *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.
2. Add selectors to the skin to style ADF faces components as required, such as the administration links, portal name and icon, or any page content user interface elements.
3. (Optional) Copy the CSS styles that you added to your page template to the new skin. You can then remove these CSS styles from the page template, allowing you to more easily edit them with the portal skin editor at runtime.
4. For portal user interface elements that are difficult to style directly, such as Administration links, use jQuery to locate the element by the unique ID, and apply style classes to them to tune their styling. For example, the following code shows the Unicorn page template, using jQuery to style portal user interface elements:

```
<script type="text/javascript">
  //Script block
  if (window.addEventListener) {
    /* Modern browsers */
    window.addEventListener("load", onLoad, false)
  }
  else if (window.attachEvent) {
    /* IE */
    window.detachEvent("onload", onLoad)
    window.attachEvent("onload", onLoad)
  }
  else {
    window.onload = onLoad
  }

  function onLoad() {
    //Work around for styling ADF elements and overriding the ADF default styles.
    $('body').addClass('#{attrs.themeType}');

    //Use Jquery to style portal admin link components such as portals,
    favorites, administration to use font-awesome icon and bootstrap css styles
    $('#portals .ptext span').html('&lt;i class="icon fa fa-users"&gt;&lt;/i>
    &lt;span class="text hidden-xs1 hidden-sml">
    #{uib_o_w_s_r_Spaces.LABEL_COMMUNITIES}&lt;/span>')
    $('#favb .ftext span').html('&lt;i class="icon fa fa-heart"&gt;&lt;/i>
    &lt;span class="text hidden-xs1 hidden-sml">
    #{uib_o_w_r_WebCenter.LABEL_FAVORITES}&lt;/span>')
    $('#wcadmin .adtext span').html('&lt;i class="icon fa fa-gears"&gt;&lt;/i>
    &lt;span class="text hidden-xs1 hidden-sml">
    #{uib_o_w_r_WebCenter.GLOBAL_LINK_ADMINISTRATION}&lt;/span>');

    ...
  }

  // Javascript force onLoad if it is not loaded, this is needed in case
  of PPR reloading -->
  if (!$('body').hasClass('#{attrs.themeType}'))
  {
    onLoad();
  }
</script>
```

For another example of using jQuery for styling, see “Example 3: Using jQuery, ADF skinning, a font-awesome icon, and Bootstrap theme CSS to style a portal Login link” on page 12:

Note: By default, a portal skin is compressed. If an ADF faces component has a `styleClass` applied to it, the `styleClass` may be compressed. If an HTML component has the same `styleClass` applied to it, the `styleClass` will not get compressed. Therefore, if a skin selector related to this `styleClass` also relies on another parent `styleClass`, it is advisable that the parent `styleClass` and child `styleClass` are both applied to ADF faces components, or both non-ADF faces components. Then, if you use link references to your downloaded CSS files, those styles will not be compressed.

Examples

The following examples from the built-in Mosaic page template illustrate how you can use bootstrap, font-awesome, and jQuery libraries to style ADF faces components and task flows for a page template. This allows you to expose dynamic values set by portal settings in a page template, styled according to a Bootstrap theme you have selected.

Example 1

Adding static Bootstrap theme HTML to a WebCenter Portal page template

1. From the downloaded Bootstrap theme HTML files, copy the HTML code snippets for the icon, name and description to the target page template file (`.jspx` or `.jsf`).

For example, these page elements in the built-in Mosaic page template are:

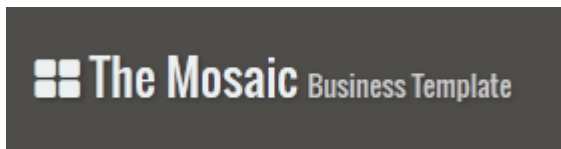
```
<!-- portal link with css style -->
<a class="navbar-brand" href="index.html">

  <!-- portal icon generated with font-awesome library -->
  <i class="fa fa-th-large"></i>

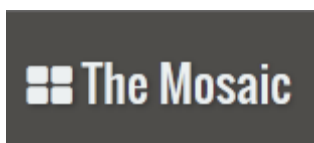
  <!-- portal name -->
  The Mosaic

  <!-- portal description with bootstrap css style -->
  <span class="hidden-sm">Business Template</span>
</a>
```

2. Test the page template (see “Testing a Page Template in WebCenter Portal” on page 14) to see the HTML code snippets render as shown in the following example screenshot:



When the browser screen is sized smaller, the description is hidden due to the `hidden-sm` style of the Bootstrap theme:



This example illustrates how you can expose the styling of a Bootstrap theme in a WebCenter portal page template. However, the page elements are static, specific to the Bootstrap theme. The next example illustrates how to apply Bootstrap theme styling to ADF faces components in a WebCenter Portal page template. This means that you can expose dynamic portal settings in your page template, using the styling of a Bootstrap theme rather than the default styling of the page template.

Example 2

Using a Bootstrap theme to style the portal icon, name, and description

1. Identify the styling tags in the HTML code snippets of the Bootstrap theme (from Example 1):

```
<!-- portal link with css style -->
<a class="navbar-brand" href="index.html">

  <!-- portal icon generated with font-awesome library -->
  <i class="fa fa-th-large"></i>

  <!-- portal name -->
  The Mosaic

  <!-- portal description with bootstrap css style -->
  <span class="hidden-sm">Business Template</span>
</a>
```

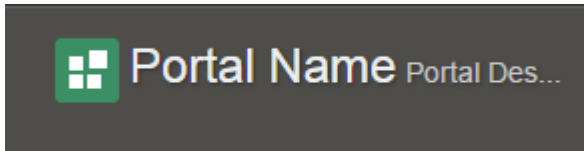
2. In your page template file, modify the `af:switcher` placeholder area (from “Adding Portal-Specific Items to a Page Template: Portal Icon, Name, or Description” on page 6), using the Bootstrap theme styling tags to style the portal icon, name, and description that are defined in the portal settings, as shown in the following example:

```
<!-- apply the "navbar-brand" css style class to the top parent container of the
af:switcher area -->
<div class="navbar-brand">
<af:switcher facetName="#{WCAppContext.currentScope.default ? 'defaultLogo' :
'spaceLogo'}">
...
  <f:facet name="spaceLogo">
    <af:goLink destination="#{boilerBean.globalLogoURIInSpace}"
      disabled="#{changeModeBean.inEditMode or serviceCtx.scope.spaceTemplate}"
      id="splggl2">
    ...
  <!-- portal display name -->
    <af:outputText value="#{serviceCtx.scope.spaceTemplate ?
      WCTruncator[spaceTemplateManager.template
        [WCAppContext.currentScope.name].GSMetadata.displayName]['10'] :
      !(empty spaceContext.currentSpace) ? WCTruncator[spaceContext.space
        [serviceCtx.scope.name].GSMetadata.displayName]['10'] :
      WCTruncator[WCAppContext.application.applicationConfig.title]['10']}" />

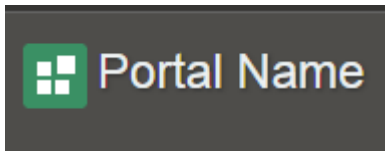
  <!-- apply the "hidden-sm" style class to the portal description -->
    <af:outputText inlineStyle="padding-left:5pxp styleClass="hidden-sm"
      value="#{serviceCtx.scope.spaceTemplate ?
      WCTruncator[spaceTemplateManager.template
        [WCAppContext.currentScope.name].GSMetadata.description]['10'] : !
      (empty spaceContext.currentSpace) ? WCTruncator[spaceContext.space
        [WCAppContext.currentScope.name].GSMetadata.description]['10'] : ''}" />
    </af:goLink>
  </f:facet>
</af:switcher>
</div>
```

3. Test the page template (see "Testing a Page Template in WebCenter Portal" on page 14) to see the Bootstrap theme styling applied to the portal components.

The following screenshot shows the styling of the built-in Mosaic page template applied to a WebCenter portal page template file, where the portal settings define the display name as "Portal Name", the portal description as "Portal Description", and the logo as the graphic shown.



When the browser screen is sized smaller, the description is hidden due to the `hidden-sm` style of the Bootstrap theme:



4. If the Bootstrap theme styling does not take effect on the ADF faces component, it is possible that it is overridden by ADF and portal default skin behavior. In this case, you may need to use ADF skinning in the skin file.

For example, to make the portal name link white without underline when the cursor hovers over it, when it has focus, or when it is visited, add the following style coding in your skin file.

```
navbar-dark .navbar-brand a:hover,  
navbar-dark .navbar-brand a:focus  
navbar-dark .navbar-brand a:visited{  
  color: #fff;  
  text-decoration:none;  
}
```

Example 3

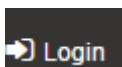
Using jQuery, ADF skinning, a font-awesome icon, and Bootstrap theme CSS to style a portal Login link

1. From the downloaded Bootstrap theme HTML files, copy the HTML code snippets for the Login link and font-awesome icon to the target page template file (`jspx` or `jsf`).

For example, these elements in the built-in Mosaic page template are:

```
<a href="sign-in.html" class="pull-right"><i class="fa fa-sign-in"></i> Login</a>
```

2. Test the page template (see "Testing a Page Template in WebCenter Portal" on page 14) to see the HTML code snippet render as shown in the following example screenshot:



3. Identify the styling tags in the HTML code snippets of the Bootstrap theme:

```
<a href="sign-in.html" class="pull-right"><i class="fa fa-sign-in"></i> Login</a>
```

- In your page template file, modify the `wcdc` placeholder area (from “Adding Portal-Specific Items to a Page Template: Portal Links” on page 6) using jQuery, ADF skinning, a font-awesome icon, and Bootstrap theme CSS to style a portal Login link, as shown in the following example:

Before:

```
<wcdc:spacesAction id="wcLoginLink"
shortDesc="#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}"
text="#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}" type="login"/>
```

After:

```
<!-- apply class on parent HTML element, use inlineStyle and styleClass to apply
CSS and ADF skinning to page component -->
<div class="pull-right WCPortalAdminLinks" id="user-nav">
  <wcdc:spacesAction id="wcLoginLink" inlineStyle="white-space:nowrap"
shortDesc="#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}" styleClass="litext"
text="#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}" type="login"/>
</div>
```

- (Optional) Modify the `text` and `shortDesc` values to specify the desired link label (such as “Login”) and hint text when the cursor hovers over the link.
- Add the Bootstrap theme styles to your CSS or skin file. For example:

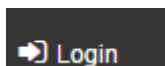
```
WCPortalAdminLinks a, .WCPortalAdminLinks a:active,
WCPortalAdminLinks a:visited, .WCPortalAdminLinks a:hover {
  color: #FFFFFF !important;
  font-size: 12px;
}
```


- For `wcdc` components and task flows that contain elements that are difficult to style directly, you can use jQuery to locate the element by the unique ID, and apply style classes to them to tune their styling. For example, a font-awesome icon and Bootstrap theme styling before the Login text cannot be directly passed in attributes for the `wcdc` component. However, you can use jQuery to locate the Login element within the `wcdc` component and insert the font-awesome and Bootstrap theme CSS HTML code snippets during runtime page rendering. **Tip:** Use browser tools such as firebug to inspect target HTML tags.

In your page template file, modify the JavaScript placeholder area (from “Adding Portal-Specific Items to a Page Template” on page 6) to use jQuery code to insert a font-awesome icon and styling before the Login link, as shown in the following example from the built-in Mosaic page template:

```
<script type="text/javascript">
...
function onLoad() {
...
$('#user-nav .litext span').html('&lt;i class="icon fa fa-sign-in">
&lt;/i>&lt;span class="text hidden-xs hidden-sm">
#{uib_o_w_w_r_WebCenter.GLOBAL_LINK_LOGIN}&lt;/span>');
...
}
</script>
```

- Test the page template (see “Testing a Page Template in WebCenter Portal” on page 14) to see the Login link render with a font-awesome icon (generated by the jQuery HTML tag `<i class="icon fa fa-sign-in">`), as shown in the following example screenshot:





When the browser screen is sized smaller, the Login text is hidden and only the icon shows due to the `hidden-sm1` style, which is an enhanced portal style modified from the `hidden-sm` style of the Bootstrap theme:



Testing a Page Template in WebCenter Portal

As you make changes to a portal during development, such as modifying the page template and skin, you can iteratively publish and test it in WebCenter Portal to view the results.

For instructions on how to publish a page template and skin to WebCenter Portal as a shared asset, or to a specific portal as a portal asset, see [“Publishing WebCenter Portal Assets”](#) in *Developing WebCenter Portal Assets and Custom Components with Oracle JDeveloper*.

When you test your portal in WebCenter Portal, be sure to verify the following areas:

- » Navigation
- » Responsiveness on different devices
- » Portal-specific components and links




Oracle Corporation, World Headquarters

500 Oracle Parkway
Redwood Shores, CA 94065, USA

Worldwide Inquiries

Phone: +1.650.506.7000
Fax: +1.650.506.7200

CONNECT WITH US

-  blogs.oracle.com/oracle
-  facebook.com/oracle
-  twitter.com/oracle
-  oracle.com

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. 0416

Creating a WebCenter Portal Page Template Based on a Bootstrap Theme
April 2016
Authors: Ingrid Snedecor, Doris Zhang, Ken Young, Savita Thakur
Contributing Authors: Igor Polyakov, Anup Chatterjee