BEA WebLogic Mobility Server

Mobilize Your Portal Guide

Version 3.3
December 2005

Contents

Introduction

About This Manual

This document explains how to mobilize a portal and then apply your own “Look & Feel” to it as required.

This involves a simple process of applying the “Mobilize Portal” template to the portal in order to mobilize it. Once you have met the pre-requisites outlined in chapter 1, proceed to the first section of chapter 2, which describes how to apply the template to a sample BEA portal. The second section of this chapter outlines how to deal with issues that may arise when mobilizing your own portal.

During mobilization, the template applies a Mobility Look & Feel to the portal. To incorporate your own Look & Feel into the portal, you will need to follow the instructions in chapter 3, which explains how to mobilize your framework.

Note: This document assumes that the reader is familiar with developing applications using BEA WebLogic Portal.

Terms Used in This Document

The term “menu-driven” refers to devices that, because of screen size and bandwidth limitations, divide content into smaller pieces and must use menus of links to allow users to navigate to different parts of the content. These devices are usually web-enabled smart phones.

The term “full browser” is used to describe conventional browsers—usually MS Internet Explorer or Mozilla—used on a PC.

The term “Look & Feel” refers to a combination of interrelated parts that determine a portal’s physical appearance. For more information, see the BEA WebLogic Workshop Online Help at http://e-docs.bea.com/workshop/docs81/doc/en/portal/buildportals/ifLF.html

- For a description of “skins” see http://e-docs.bea.com/workshop/docs81/doc/en/portal/buildportals/ifLFSkinsThemes.html
- For a description of “shells” see http://e-docs.bea.com/workshop/docs81/doc/en/portal/buildportals/ifLFShells.html
- For a description of “skeletons” see http://e-docs.bea.com/workshop/docs81/doc/en/portal/buildportals/ifLFSkeletons.html

Multi-Channel Portals

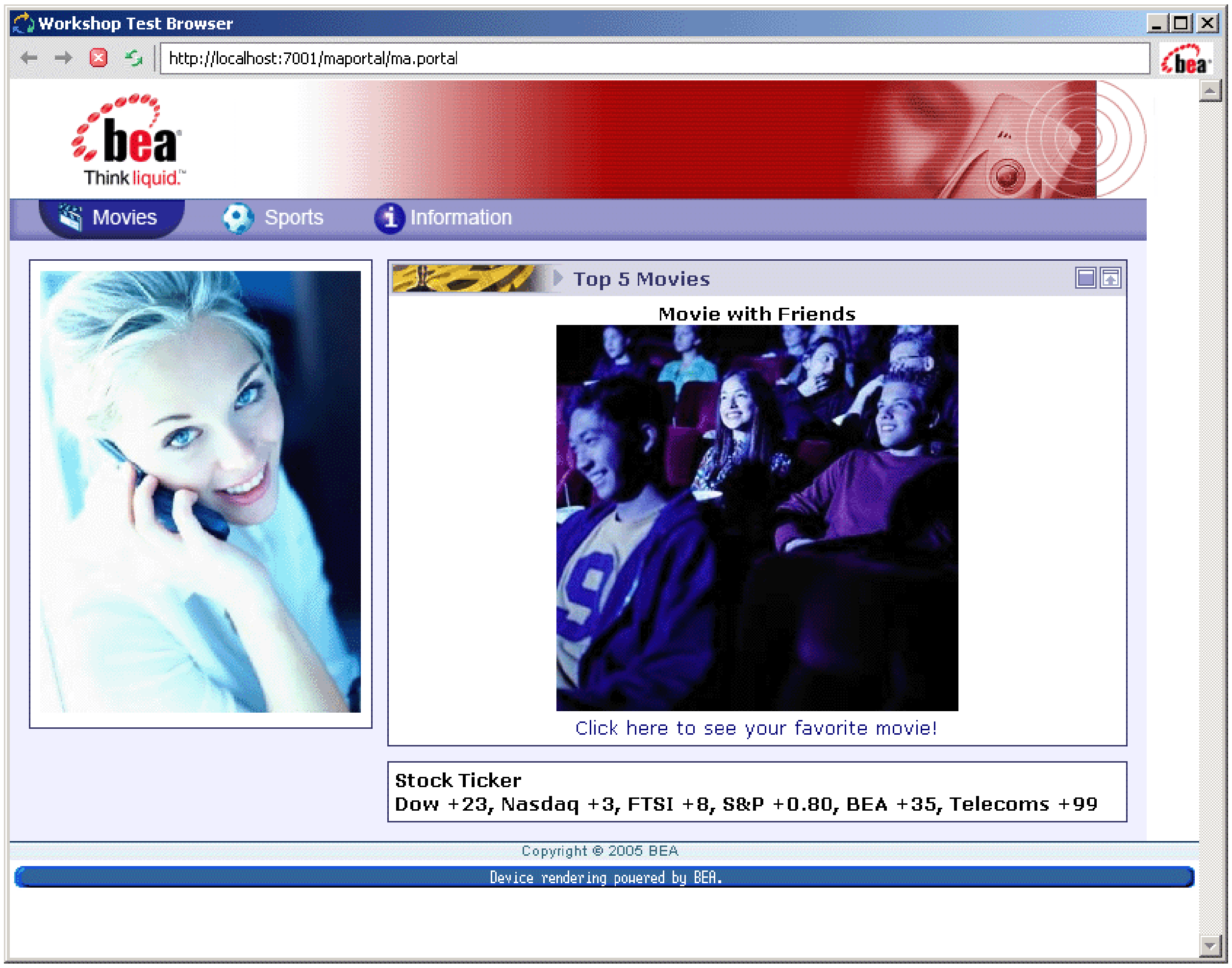
The Mobile Portal Framework extends the functionality of the BEA WebLogic Portal platform to allow developers to create mobile applications that run on handheld devices in addition to PCs. These are called multi-channel portals.

The mobility technology and mobility framework for BEA WebLogic Portal 8.1 ensure an optimal user experience on devices such as smart phones, PDAs, and so on.

Typical portals live in a two-dimensional PC-browser world with relatively large amounts of screen real-estate, which enables complex layouts and substantial content to display simultaneously. Content in this format is not generally suitable for direct delivery to handheld devices such as PDAs and smart phones. The multi-channel portal automatically restructures the content and provides new navigation mechanisms as necessary, tailoring it for the device making the request.

One of the most important principles of the multi-channel portal is that it allows developers to develop portal content (that is, portlets) without having to deal with the issues involved in delivery to, and navigation on, handheld devices. This is achieved by placing logic to handle these issues into a mobility framework.

The Sample Portal



Stock Ticker
Dow +23, Nasdaq +3, FTSI +8, S&P +0.80, BEA +35, Telecoms +99

Copyright © 2005 BEA
Device rendering powered by BEA.

1—Pre-Requisites

Ensure that you have installed BEA WebLogic Mobility Server™ and configured the device emulators as described in *BEA WebLogic Mobility Server Installation Guide*.

2—Mobilize the Portal

The first step that you will undertake in mobilizing your portal is applying the Mobilize Portal template to it. The “Apply the Mobilize Portal Template” section demonstrates how to do this, by describing a scenario in which the template is applied to a sample BEA portal.

Once you have read this section and are ready to mobilize your own portal, you may need to see the “Specific Mobilization Scenarios” section, which explains how to address issues that may arise when mobilizing a portal.

Apply the Mobilize Portal Template

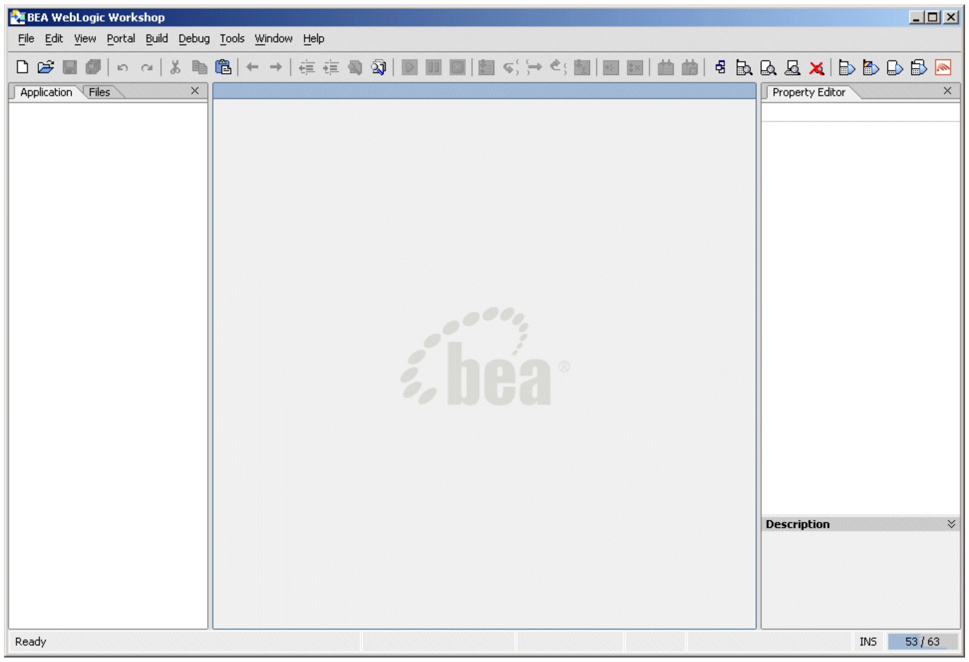
Applying the template involves the following steps:

1. Launch BEA WebLogic Workshop and open the portal that you want to mobilize.
2. Apply the template to the portal.
3. Select the “Mobility” values for the portal desktop attributes.
4. Save your changes.

The WebLogic Mobility Server Installer installs the Mobilize Portal template in your BEA Workshop templates directory. You will now apply this template to your portal to mobilize it. Here, we will use BEA Workshop to apply the template to a sample BEA portal.

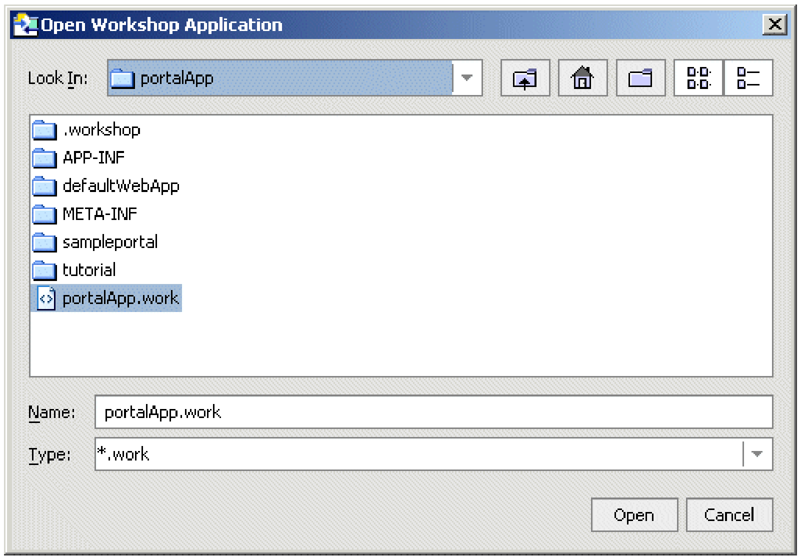
1. Choose **Start** → **Programs** → **BEA WebLogic Platform 8.1** → **WebLogic Workshop 8.1**. The following screen is displayed.

   BEA Workshop Opening screen



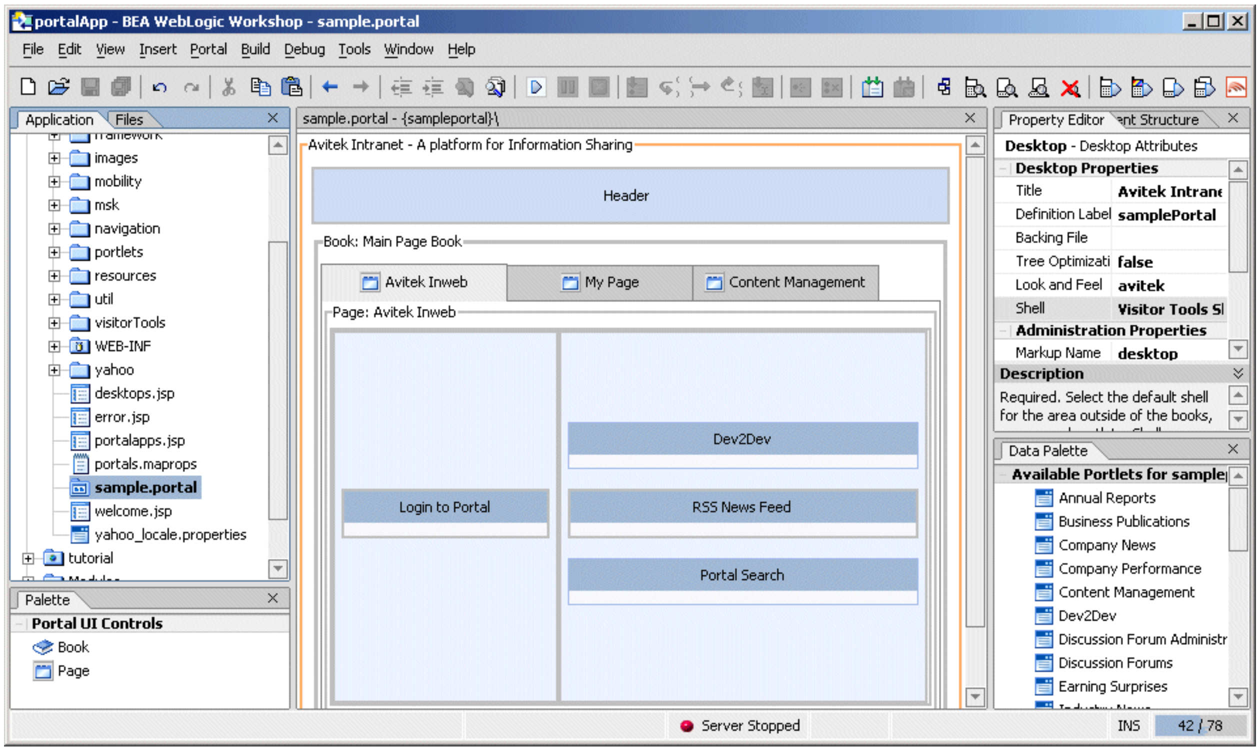
2. Choose **File** → **Open Application**. The Open Workshop Application dialog is displayed.

Open Workshop Application dialog



3. To open the sample BEA portal, navigate to **<your bea directory>\weblogic81\samples\portal\portalApp\** and open the *portalApp.work* file.
4. Select the **Application** tab to view the portalApp application. Navigate to the **sampleportal** folder within the **portalApp** folder.

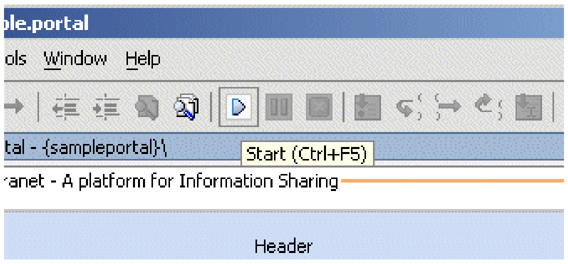
Design view of the portal



5. Double-click on the *sample.portal* file from within this folder to launch a design view of the portal.

6. Locate the start button on the toolbar across the top of the screen to view how the pre-mobilized sample portal will display.

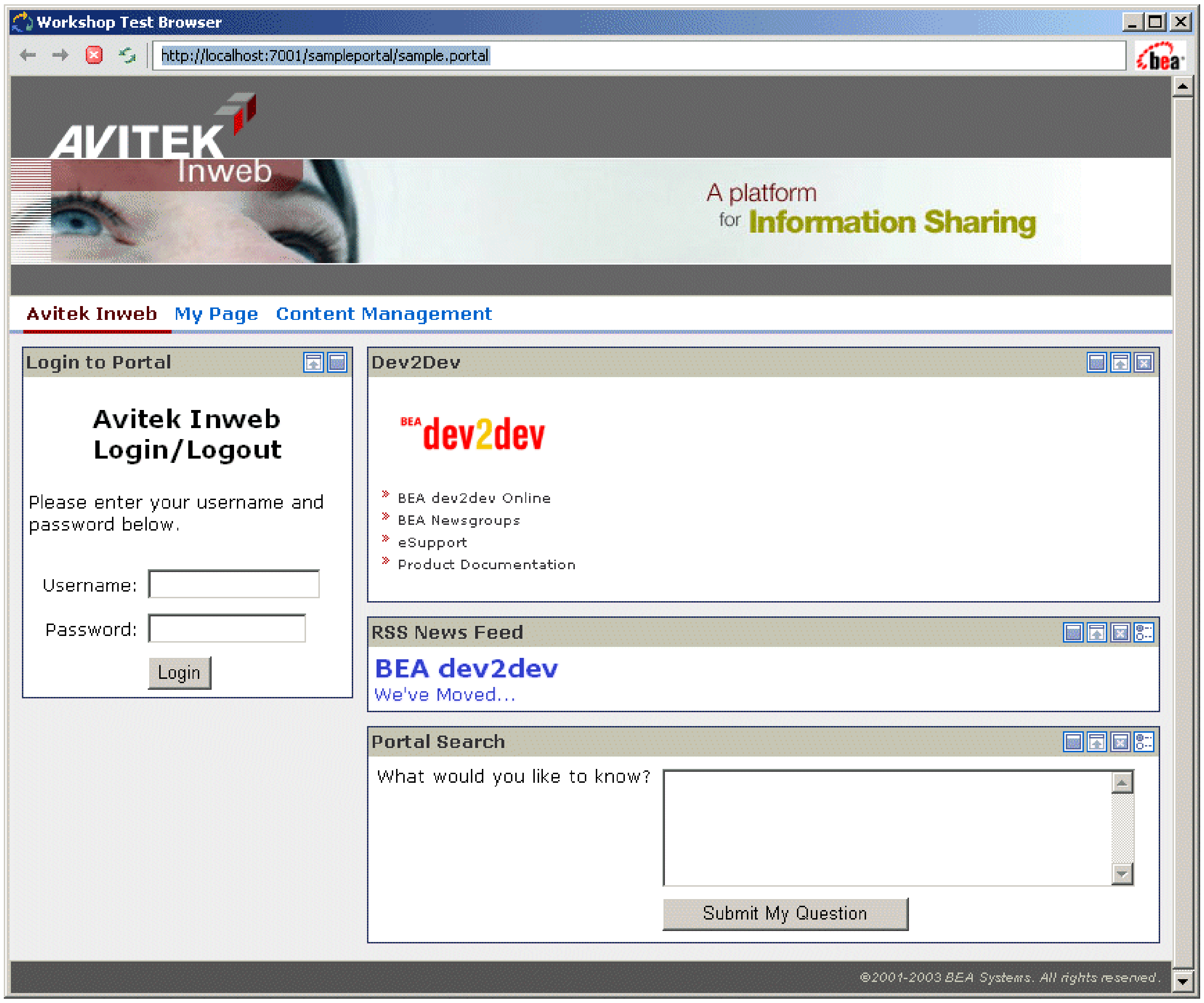
Start button



Click the start button to start the sample portal.

7. The pre-mobilized sample portal will display in the Workshop test browser.

Test browser



Avitek Inweb Login/Logout

Please enter your username and password below.

Username:

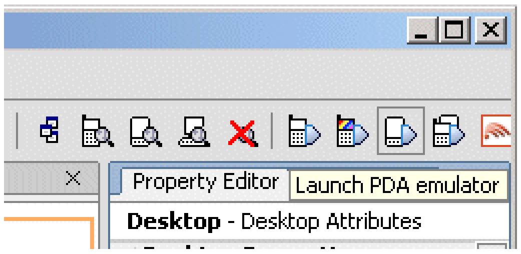
Password:

Login

What would you like to know?

8. Launch the PDA emulator from the toolbar to view how the portal would be rendered on a PDA device before it has been mobilized.

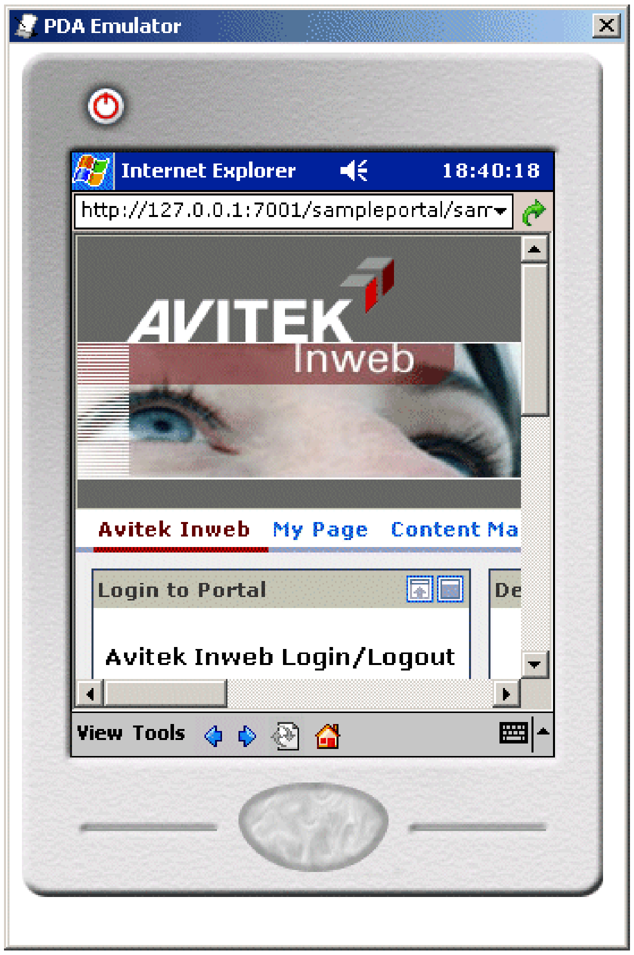
Launch PDA emulator



Click the Launch PDA emulator icon to launch the emulator.

9. The PDA emulator is displayed.

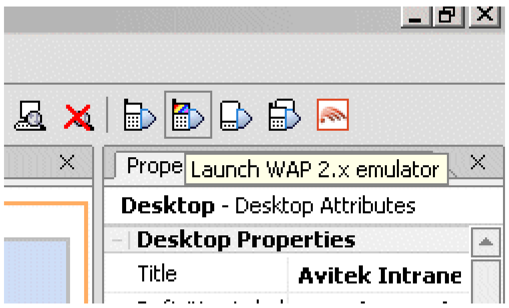
Pre-mobilized portal on PDA emulator



As you can see from the image generated using this example, the portal is rendered in a visually unattractive format and the non user-friendly navigation means that users must scroll extensively to view screen content.

10. Launch the Openwave emulator from the toolbar to view how the portal would be rendered on a mobile device before it has been mobilized.

    Launch Openwave emulator

    Click the Launch WAP 2.x emulator icon

11. The Openwave emulator is displayed.

    Pre-mobilized portal on Openwave emulator

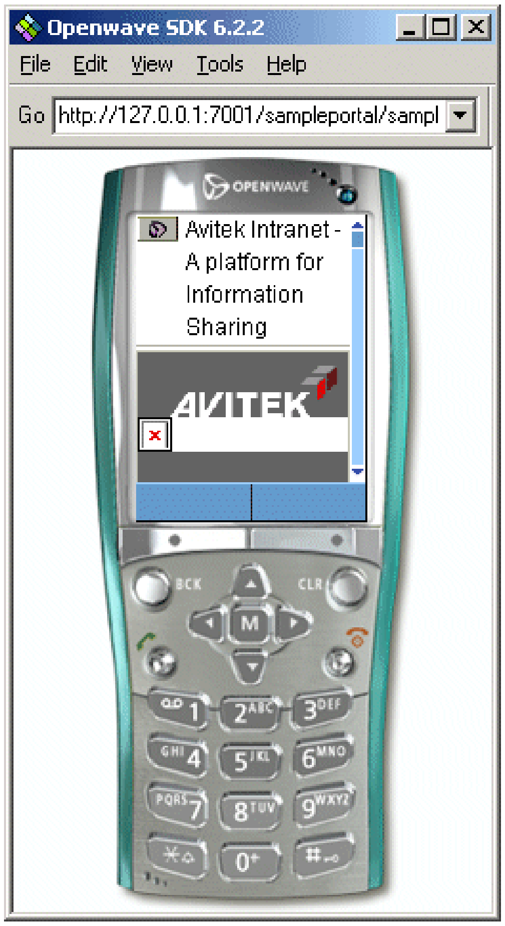
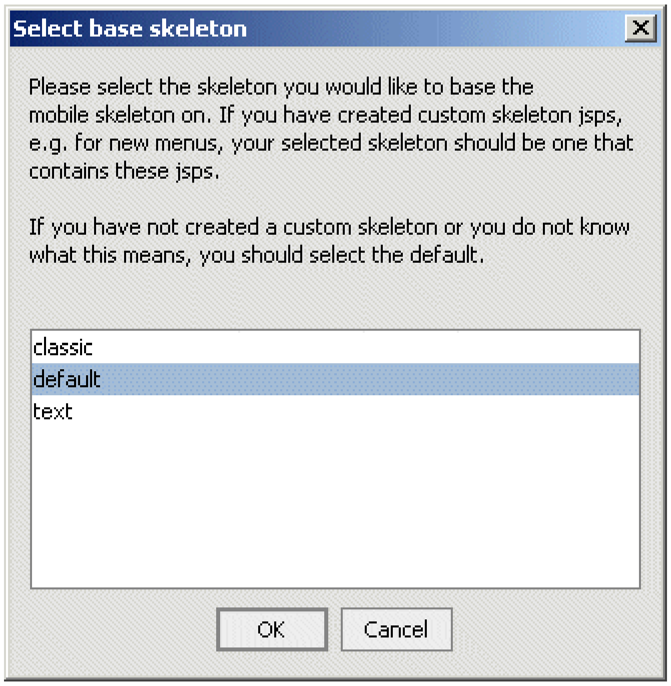
    
    Openwave SDK 6.2.2
File
Edit
View
Tools
Help
Go
http://127.0.0.1:7001/sampleportal/sampl
OPENWAVE
Avitek Intranet - A platform for Information Sharing
AVITEK
BCK
CLR
M

    Image Courtesy of Openwave Systems, Inc.

    If the portal is actually rendered on the mobile device, navigation links and images will not display correctly as you can see from the preceding graphic, and the non user-friendly navigation means that users must scroll extensively to view screen content.

12. You will now apply the Mobilize Portal template to mobilize the portal. Right-click on the **sampleportal** folder in the **Application** tab and choose **Install** → **Mobilize Existing Portal**.
13. The Select Base Skeleton dialog is displayed.

    Select Base Skeleton dialog

    Please select the skeleton you would like to base the mobile skeleton on. If you have created custom skeleton jsps, e.g. for new menus, your selected skeleton should be one that contains these jsps.

    If you have not created a custom skeleton or you do not know what this means, you should select the default.

    classic
    default
    text

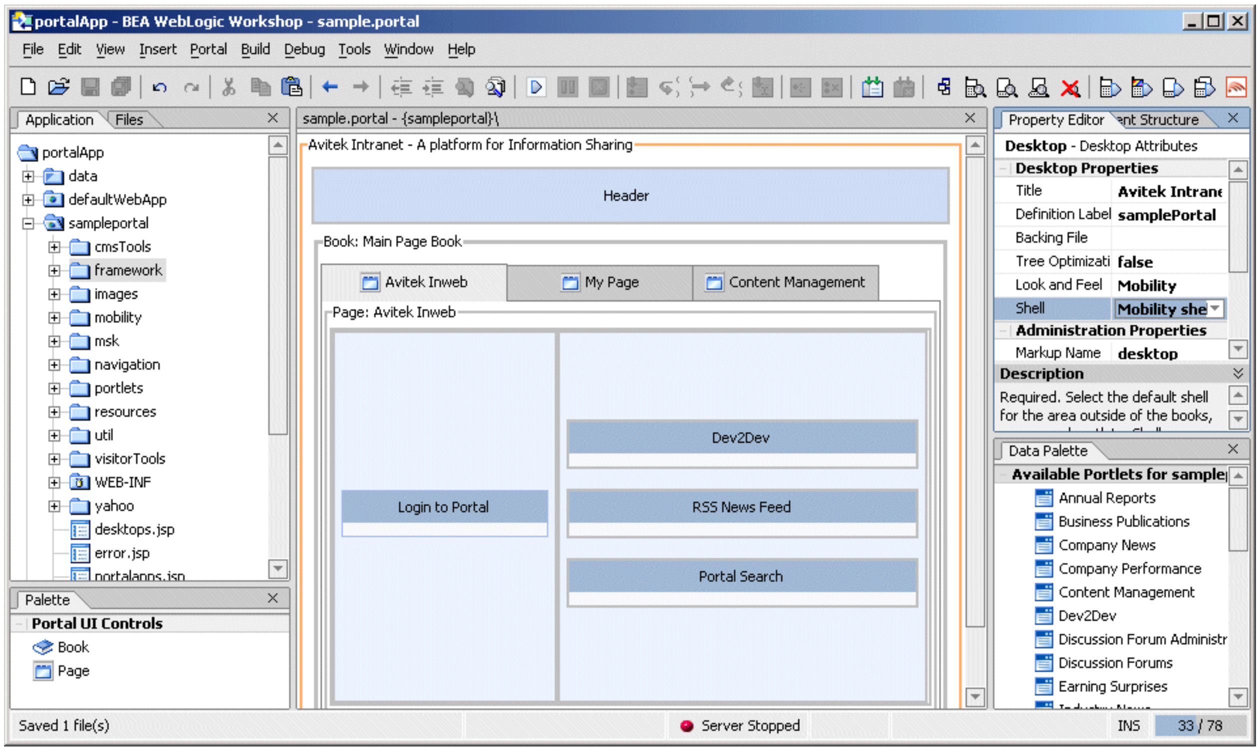
14. Select “default” and click **OK**. You have now applied the template.

15. Click the title of the portal in the designer (“Avitek Intranet - A platform for Information Sharing”) to display the “Look and Feel” and “Shell” desktop attributes in the **Property Editor** section to the right-hand-side of the screen.

16. Set the values of these properties as follows:

    - Look and Feel: Mobility
    - Shell: Mobility shell

    Property editor section



Desktop - Desktop Attributes

Desktop Properties

Title	**Avitek Intrane**
Definition Label	**samplePortal**
Backing File	
Tree Optimizati	**false**
Look and Feel	**Mobility**
Shell	**Mobility she**

Administration Properties

Markup Name	**desktop**

Description

Required. Select the default shell for the area outside of the books,

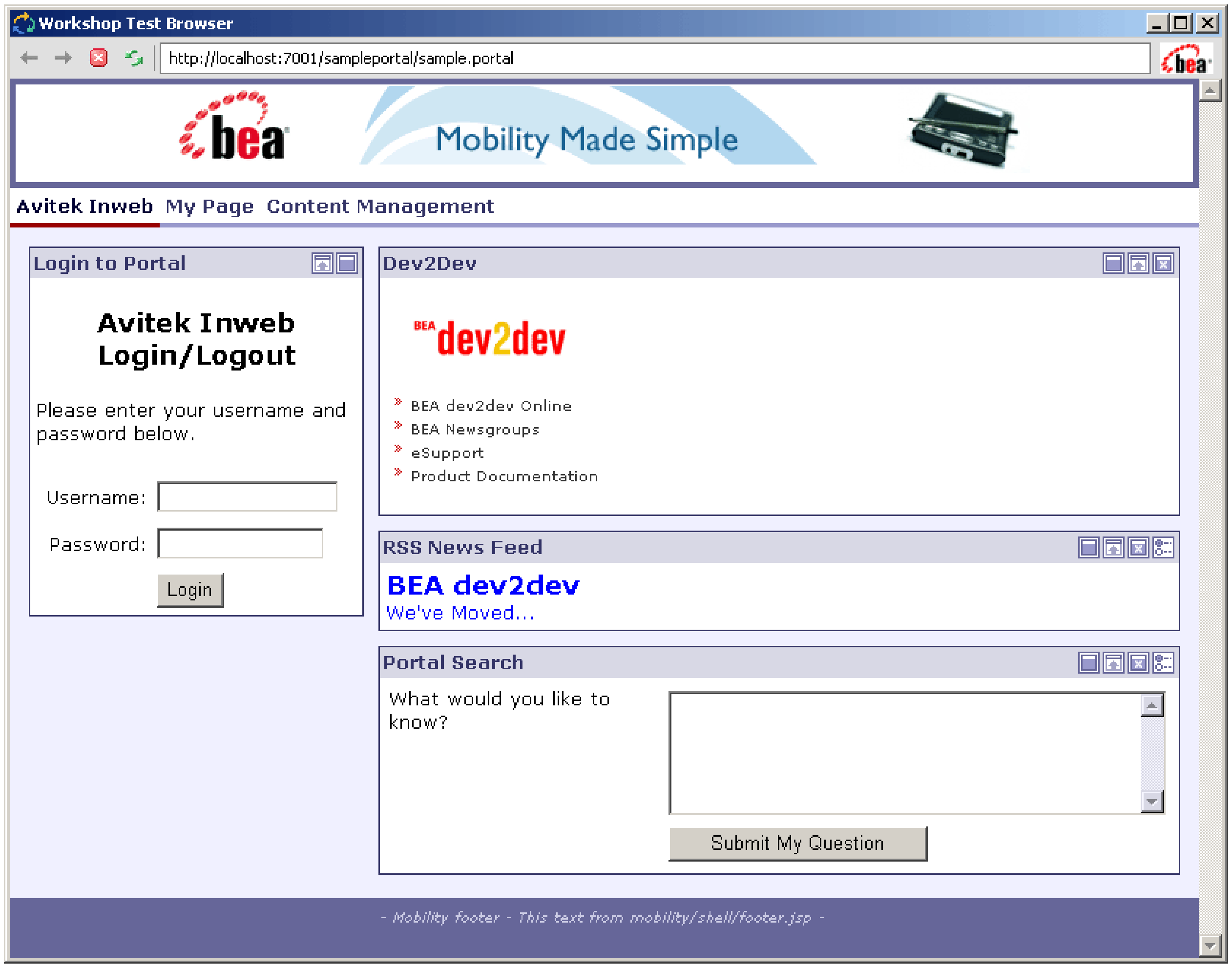
17. Save the changes that you made to the portal and press the Start button again.

18. An updated test browser will display, in which you should now see a sample header and footer, which indicates that the portal has been mobilized.

    Note: If necessary, you can change this header and footer to suit your portal at a later stage—see section “Customize Mobile Images”

    Note: This header and footer will not display if you did not select the Mobility values for the “Look and Feel” and “Shell” desktop attributes, as described on the previous page.

    Header and footer



Workshop Test Browser

http://localhost:7001/sampleportal/sample.portal

Mobility Made Simple

Avitek Inweb My Page Content Management

Login to Portal

Avitek Inweb Login/Logout

Please enter your username and password below.

Username:

Password:

Login

Dev2Dev

BEA dev2dev

- BEA dev2dev Online
- BEA Newsgroups
- eSupport
- Product Documentation

RSS News Feed

BEA dev2dev

We've Moved...

Portal Search

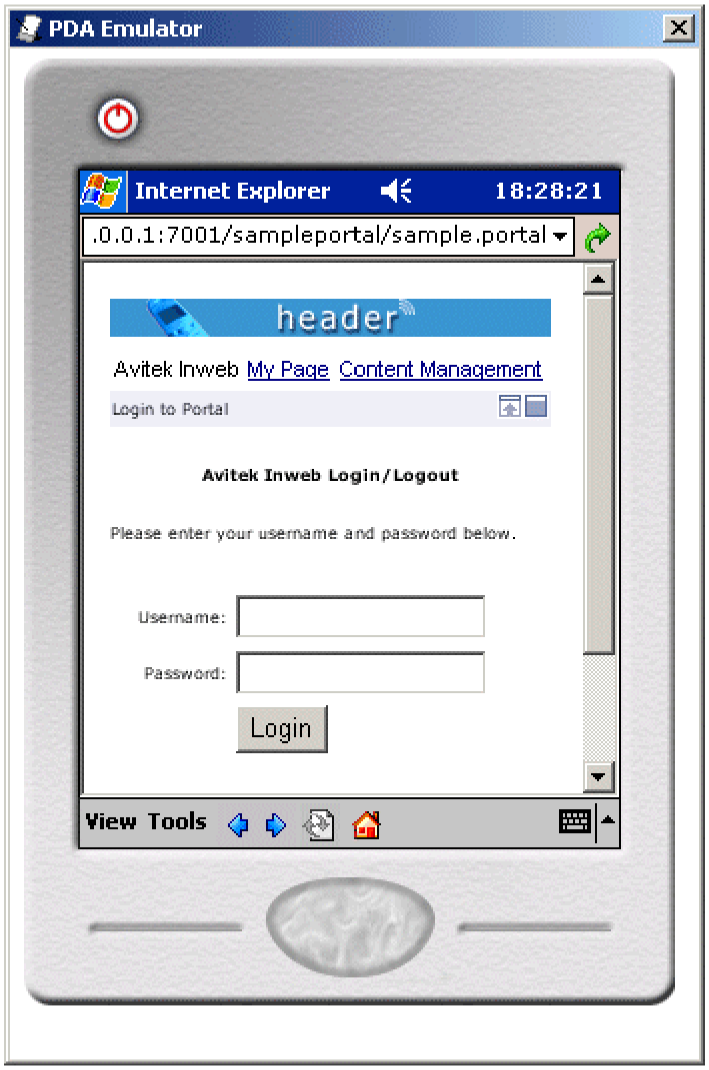
What would you like to know?

Submit My Question

- Mobility footer - This text from mobility/shell/footer.jsp -

19. Now, click the Launch PDA emulator icon again. The PDA emulator is displayed.

PDA emulator


PDA Emulator
Internet Explorer
18:28:21
.0.0.1:7001/sampleportal/sample.portal
header
Avitek Inweb My Page Content Management
Login to Portal
Avitek Inweb Login/Logout
Please enter your username and password below.
Username:
Password:
Login
View Tools

The sample header and footer will display, indicating that the portal is mobilized. The portal functionality, images and navigation links have been successfully delivered to the PDA device and rendered in a manner that is user-friendly and easy to navigate.

20. Now, click the Launch WAP 2.x emulator icon again. The Openwave emulator is displayed.

Openwave emulator

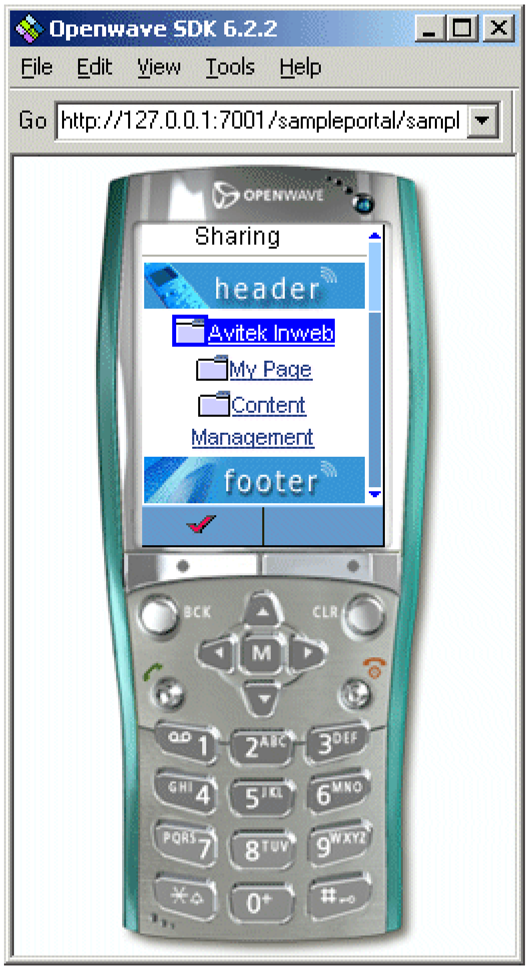


Image Courtesy of Openwave Systems, Inc.

The sample header and footer will display, indicating that the portal is mobilized. The portal functionality, images and navigation links have been successfully delivered to the mobile device and rendered in a manner that is user-friendly and easy to navigate.

Note: Please see the “The Mobility Framework” chapter of *BEA Sample Mobility Portal Guide* for more information on the sample portal, on which the Mobilize Portal template is based.

Specific Mobilization Scenarios

In the majority of cases, the steps described in the previous section will be sufficient to mobilize an existing portal and deliver it to mobile devices. In some cases however, you may need to undertake further steps before the portal can be viewed on these devices. The following scenarios may cause issues that you will need to resolve:

- The presence of custom images in skins
- The presence of badly-formed HTML content
- The presence of portlets and/or content outside of the main desktop book (for example, in a shell's header)
- The presence of functionality that relies on scripting that is not supported on mobile devices
- The presence of styling that relies on cascading style sheet (CSS) features that are not supported on mobile devices

The following sections explain the issues that may arise as a result of these scenarios, and outline the steps that you will need to undertake to address them.

Scenario 1—The Presence of Custom Images in Skins

Issue

The BEA WebLogic Portal "skin mechanism" allows your portal to use different images (and CSS styling and scripts) according to the skin referenced by the Look & Feel selected in the portal.

Each skin specifies a search path for images, CSS and scripts.

Sample JSP code from a portlet selecting an image from a skin

```
<img src='<render:getSkinPath imageName="portlet-bullet.gif"/>'/>
```

At runtime, the portal searches the selected skin's path for the named image (in the preceding example, *portlet-bullet.gif*). If the image is not found, an error occurs.

The Mobilize Portal template installs a new skin at the location **framework/skins/mobility**. This skin's search path is configured (in **framework/skins/mobility/***skin.properties*) with the following search path for images:

```
images.search.path: images,../default/images
```

When you select the Mobility Look & Feel, the Mobility skin is applied. Therefore, if your portal contains other skins, and requires images that are not present in either the Mobility skin or at **framework/skins/default/images**, errors will occur in your portal.

Solution

To ensure that each image can be found in the Mobility skin, you can either:

- Copy all the relevant images into the Mobility skin's image directory (**framework/skins/mobility/images**)

OR

- Modify the Mobility skin's search path to include an existing skin's images. There are typically three properties files that you will need to update. The *skin.properties* file is located at **framework/skins/mobility**. However, because this skin has been mobilized, there are two subdirectories, **pda** and **menudriven**, each with its own *skin.properties* file. Therefore, you will also need to modify f**ramework/skins/ mobility/pda/***skin.properties* and **framework/skins/ mobility/menudriven/***skin.properties*

Scenario 2—The Presence of Badly-Formed HTML Content

Issue

Browsers such as MS Internet Explorer are very tolerant of badly-formed HTML portal content. For example, if you fail to match an opening tag such as `<td>` with its closing tag (`</td>`), Internet Explorer will make a "best guess" at the intention of the content and render it appropriately. As a result of this, a situation may easily arise whereby incorrectly-written HTML content is never identified because the resulting output actually looks correct.

The process of translating HTML content for delivery to other devices can be hindered by such badly-formed content. Therefore, content which previously appeared to work correctly can begin to generate errors once you mobilize the portal.

Solution

If your portal or portlets contain such markup, content warnings or errors will display in the WebLogic Server Application Server console, which will guide you towards correcting the problems.

Note: To prevent these warnings from displaying, change the `operation.mode` setting in the *mis.properties* file from "development" to "production"—for more information see the "Configuration Mode Properties Setting" section of *BEA WebLogic Mobility Server Administration Guide*.

Scenario 3—Portlets and/or Content Located Outside of the Main Desktop Book

Issue

The mobile skeleton installed by the Mobilize Portal template delivers the desktop book, and its pages and portlets, to mobile devices. Content that exists outside the desktop book however, for example, content included in portal shells, will not be automatically mobilized.

Solution

It is usually safe to exclude this content (for example, large banner images for headers) for mobile devices and thus ensure that it is delivered only to full browsers—see the section "Mobilizing Your Shells" in chapter 3 for information on how to do so. However, if important content that you want to deliver to mobile devices (for example, a login portlet) is located in the shell, you will need to incorporate it into the main body of the portal by adding the relevant portlet to your main desktop book.

Scenario 4—The Presence of Functionality That Relies on Non-Supported Scripting

Issue

Portal content that relies on scripting (for example, javascript) will not function correctly on most mobile devices (or on several types of browsers). It is good general practice to include alternative non-script based content for use by mobile devices and other browsers that do not support the scripting language employed.

Solution

If your portal uses scripts for its basic functionality, you will therefore need to add equivalent non-script content in order for the portal to function correctly on mobile devices.

Scenario 5—The Presence of Styling That Relies on Non-Supported CSS Features

Issue

Many mobile devices do not support full CSS styling. Therefore, portal content that relies on CSS styling in order to be rendered correctly will not always display properly on these devices.

WebLogic Mobility Server *can* transform certain CSS styling into markup appropriate for delivery to mobile devices, but it will only do so if the mobile device supports an equivalent to the CSS styling properties in question.

Solution

Therefore, as a general rule, content should be legible and usable without any requirements for CSS styling. If content is authored in this way, it should render correctly on a mobile device.

3—Mobilize Your Own “Look & Feels” and “Shells”

The previous chapter explained how to add a pre-mobilized Look & Feel and shell to your existing portal. However, it is likely that your ultimate goal will be to use your existing Look & Feel and shells when enabling your portal to be accessed from mobile devices. This section shows you how to mobilize your own framework to make this a reality.

There are three main aspects of a portal framework that need to be mobilized:

- Skeletons
- Skins
- Shells

Follow the instructions in the “Mobilize Your Skeletons”, “Mobilize Your Skins” and “Mobilize Your Shells” sections, respectively, to mobilize your own framework.

Note: For more information on Look & Feels, skins, skeletons and shells, see the BEA WebLogic Workshop Online Help.

Mobilize Your Skeletons

Mobilize desktop.jsp

1. Navigate to **framework/skeletons/<yourskeleton>**/*desktop.jsp* and open the *desktop.jsp* file. Its contents will look similar to this:

   ```
   <%@ taglib uri="render.tld" prefix="render" %>
   <%@ taglib uri="mobility.tld" prefix="mm"%>

   <render:beginRender>
   <mm:page/>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
   <!-- Begin Desktop -->
   </render:beginRender>
   <render:endRender>
   <!-- End Desktop -->
   </render:endRender>
   ```

2. Insert the `<mm:page/>` tag as indicated (in bold text above). This tag instructs the WebLogic Mobility Server servlet filter to initialize processing when the *desktop.jsp* file is executed.

   Note: Ensure that you insert the `<mm:page/>` tag directly after the opening `<render:beginRender>` tag.

Add Menu-Driven and PDA Skeletons

The main functionality of the mobile framework is contained in the two mobile sub-skeletons within the skeleton directory. The details of these skeletons are explained in a later section.

For now, it is sufficient to simply copy the **menudriven** and **pda** subdirectories from **framework/skeletons/mobility** into your own skeleton directory.

Modify Search Paths in skeleton.properties

You will recall that when you applied the Mobilize Portal template to your portal web application, you were asked to select a “base skeleton”. The template used this information to generate JSP search paths for the mobile skeletons.

For example, if you chose a skeleton called “avitek”, the search path in the PDA sub-skeleton might be:

```
jsp.search.path: ., .., ../../avitek, ../../default
```

When rendering a portal using this PDA sub-skeleton, BEA Weblogic Portal (WL Portal) will firstly look for each JSP in the **pda** directory (“.”). If the **pda** directory does not contain the named JSP, WL Portal will subsequently look in the parent skeleton’s directory (“..”), followed by the avitek skeleton’s directory (“../../avitek”) and finally the default skeleton’s directory (“../../default”).

If this search order is correct for the skeleton that you are mobilizing, it is not necessary to make any changes. If you require a different search path however, you need to edit the following files accordingly:

framework/skeletons/<your skeleton>/menudriven/*skeleton.properties*

framework/skeletons/<your skeleton>/pda/*skeleton.properties*

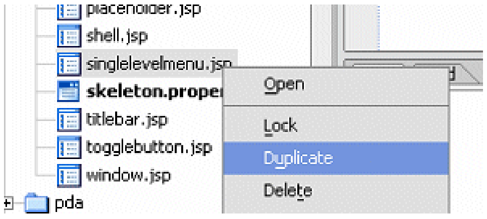
Create Required Mobile Navigation-Menu JSP Files

If your portal has custom book navigation-menus that use skeleton JSP files other than *singlelevelmenu.jsp* and *multilevelmenu.jsp*, you may need to create mobile versions of these additional JSP files. When you apply the Mobilize Portal template to your portal web application, the template searches for any such custom *.menu* files and automatically creates mobile JSP files for the JSP files referenced in these *.menu* files.

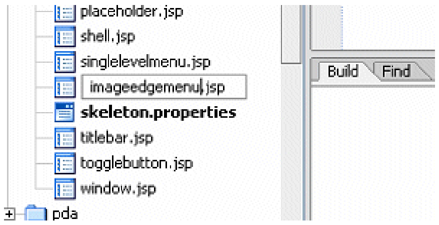
However, if the template did not create a mobile JSP file for one of your custom menus (which may happen for example, if the template failed to find a *.menu* file as it was not located in the standard **framework/markup** directory), or if you have since added additional menus, you will need to manually create mobile versions of these JSP files.

For example, if your skeleton contains a JSP file named *imageedgemenu.jsp*, you will need to create a **menudriven/***imageedgemenu.jsp* file and a **pda/***imageedgemenu.js*p file. To do so, make a copy of the *singlelevelmenu.jsp* in each of these directories (to do so in BEA WL Workshop, right-click on the file and choose **Duplicate**) and rename the copy with the name of the required mobile JSP file, in this example, *imageedgemenu.jsp*.

Duplicate singlelevelmenu.jsp for the required JSP files



Rename singlelevelmenu.jsp



Mobilize Your Skins

To use your own skins with the mobile portal framework, complete the steps in the following sections:

- “Create pda and menudriven Sub-Skins”
- “Configure the Sub-Skins’ Search-Paths”
- “Customize Mobile Images”
- “Add Portal- and Portlet-Specific Images to Your Skin”

Create pda and menudriven Sub-Skins

1. Create two folders named **pda** and **menudriven** in your skin directory.
2. These folders will, at a later stage, each contain a *skin.properties* file and potentially some image and CSS files. For the moment, copy the *skin.properties* files from the folders **framework/skins/mobility/pda** and **framework/skins/mobility/menudriven** into the corresponding new folders in your skin directory.

   Note: The mobility skin and sub-skins installed by the Mobilize Portal template do not place any images or CSS files directly into the skin folders. The skins use another folder to store these files, in order to keep the URLs for these resources as short as possible. Many mobile devices have limitations on the amount of content that they can receive, therefore keeping URLs as short as possible avoids using up valuable page-space that could otherwise be used by different content.

   The folder that the skins use is named **msk** (an acronym for “Mobility SKin”), and is installed by the template at the root of the webapp. In this folder you will find all of the images and CSS files that the mobile portal framework will need to use.

There are several methods of mobilizing your own skin, of which the most commonly-used are:

- Creating your own folders for storing your skin resources (images and CSS files). This involves keeping the folder names as short as possible and placing the folder in the root of the portal web application
- Using the existing **msk** folder for your own resources. If you do not need to maintain multiple skins in parallel, this may be the simplest option
- Placing the skin resources into the skin folder itself (for example, **framework/skins/<yourskin>/pda/images)—**this option is preferable if you have previously created portal skins and if you are not targeting devices with small content size limitations

It is also possible to combine these approaches and place resources in multiple locations, using the search path mechanism described in the next section.

Configure the Sub-Skins’ Search-Paths

Each skin and sub-skin specifies a search-path that is used to locate its associated images. This path is configured in the *skin.properties* file. The PDA sub-skin installed by the Mobilize Portal template at **framework/skins/mobility/pda/***skin.properties* uses the following image search path:

```
images.search.path: images,../../default/images,../../../../msk
```

This configuration directs WL Portal to look firstly in the **framework/skins/mobility/pda/images** location when searching for a skin image from the PDA sub-skin. If the image does not exist at this location, WL Portal will subsequently look in the default skin’s **images** folder, followed by the **msk** directory. Note that the paths are relative to the folder containing the *skin.properties* file (in this scenario, **framework/skins/mobility/pda**).

Similarly, the menudriven sub-skin installed by the Mobilize Portal template at **framework/skins/mobility/menudriven/***skin.properties* uses the following image search path:

```
images.search.path: ../../../../msk,images,../../default/images
```

A similar mechanism is used to configure search paths for CSS files.

Once you have decided where you want to place your sub-skin’s resources, modify the search paths in the *skin.properties* files (that you copied into your sub-skin folders) to point to the chosen location(s).

Customize Mobile Images

In the **msk** folder, the following image files exist for use by the mobile portal framework:

- Images for “back to page” and “back to home” icons: *back.gif, back.wbmp, back2.gif* and *back2.wbmp*
- An images for the footer on monochrome devices: *footer.wbmp*
- Images for color footers for devices of differing widths: *footer84.gif, footer96.gif, footer110.gif, footer125.gif, footer165.gif, footer200.gif* and *footer220.gif*
- An image for the footer for PDA devices: *footerpda.gif*
- An image for the header on monochrome devices: *header.wbmp*
- Images for color headers for devices of differing widths: *header84.gif, header96.gif,* header110.gif, *header125.gif, header165.gif, header200.gif* and *header220.gif*
- An image for the header for PDA devices: *headerpda.gif*
- Default icons for page-links on menu-driven devices (mono and color): *page.wbmp* and *page.gif*
- Horizontal separators used in various places in the menu-driven portal: *separator.gif, seppage.wbmp, sepportlet.wbmp* and *sepwide.gif*

In your own skins, you should replace these images with your own images, while maintaining each image file’s dimensions. Note that omitting any of these image files may prevent the mobile portal framework from rendering the portal correctly.

When creating these images, you may use your preferred image editing tool, or alternatively, use the Image Converter tool—see *Using The Image Converter Guide* for more information.

Add Portal- and Portlet-Specific Images to Your Skin

Your portal and portlets may use the skin mechanism (described in “Scenario 1”) to request images that are not found at any of the search-path locations (these search-paths are described in the section “Configure the Sub-Skins’ Search-Paths”). To address this issue, you can either:

- Create mobile versions of the images, for example by resizing the images and placing them in your sub-skins image folders. See *Using The Image Converter Guide* for more information.
- Alternatively, if these images are already suitable for delivery to mobile devices, you can configure the search paths in your *skin.properties* files so that WL Portal can locate the images in the existing skin folders.

Once again, these techniques can be combined, configuring the search path and “overriding” individual images by placing mobile versions into the sub-skins’ folders.

Mobilize Your Shells

As previously mentioned, portal content that exists outside the desktop book, for example, content included in shells, will not be automatically mobilized. In the majority of cases, it is safe to exclude this content for mobile devices (for example, large banner images for headers) and ensure that it is delivered only to full browsers. To do this, follow the instructions in the “Ensure Content Delivery Only to Full Browsers”section.

Ensure Content Delivery Only to Full Browsers

To ensure that content is delivered only to full browsers you must add Mobility markup to the relevant JSP file. You will typically add `<mm:include>` tags to the JSP file, as demonstrated in the following graphic; however, to determine the correct markup to use for your particular portal, it may be necessary to see the *WebLogic Mobility Server User Guide*.

Navigate to the file **mobility/shell/***header.jsp* in your mobilized portal to view an example of this markup.

Example: Sample markup

Navigate to the file **mobility/shell/***header.jsp* in your mobilized portal to view an example of added Mobility markup (denoted in bold text).

```
<%@taglib prefix="render" uri="render.tld"%>
<%@ taglib uri="mmJSPtaglib" prefix="mm"%>
<mm:include where="IsFullBrowser">
    <table border="0" cellpadding="0" cellspacing="1" width="100%" bgcolor="white">
        <tr>
            <td align="center">
                <img src='<render:getSkinPath imageName="header.gif"/>'>
            </td>
        </tr>
    </table>
</mm:include>
```

Used in the manner described here, the `<mm:include>` tags specify that the content contained within them will be delivered only to full browsers.

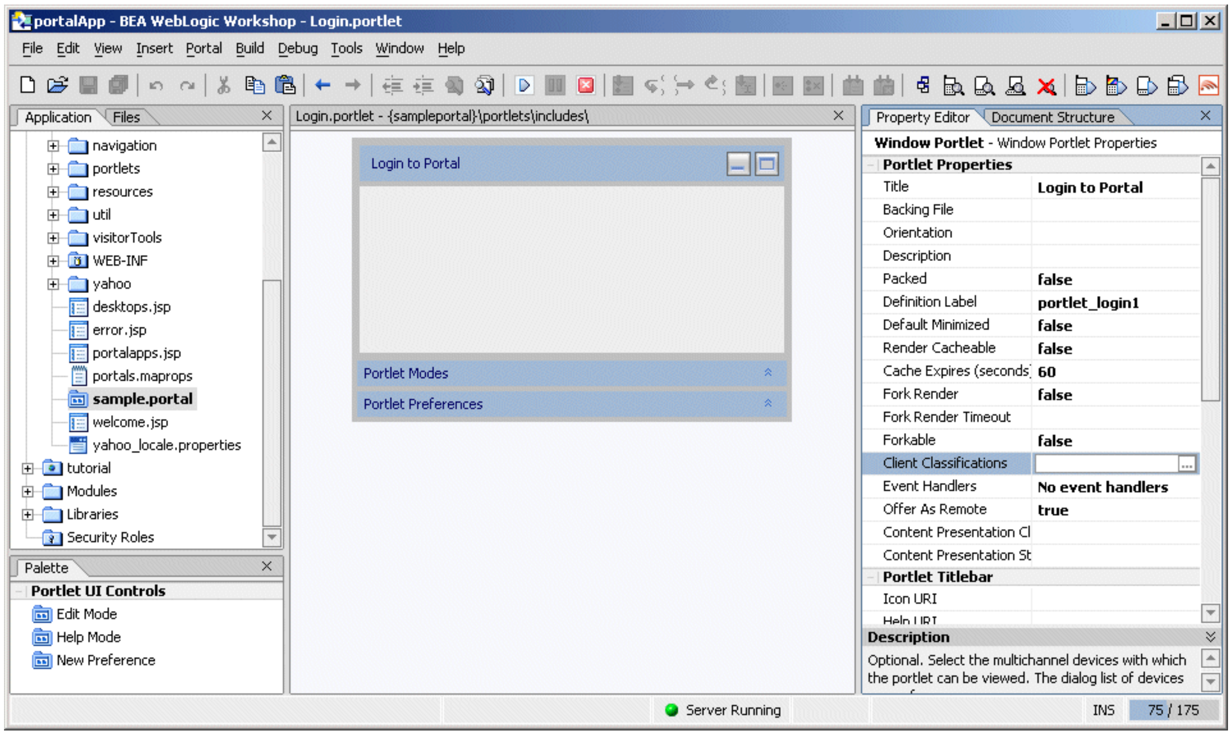
Note: As previously mentioned, it may be the case that important content that you want to deliver to mobile devices (for example, a login portlet) is located in the shell. To ensure that this content is successfully delivered to mobile devices, you will need to incorporate it into the main body of the portal, by adding the relevant portlet to your main desktop book.

Ensure Individual Portlet Delivery Only to Mobile Devices

It is also possible to ensure that an individual portlet is delivered only to mobile devices and not to full browsers. To achieve this, follow these steps:

1. Open the Portlet Designer for the portlet in question.

Portlet Designer



Window Portlet - Window Portlet Properties

Portlet Properties	
Title	**Login to Portal**
Backing File	
Orientation	
Description	
Packed	**false**
Definition Label	**portlet_login1**
Default Minimized	**false**
Render Cacheable	**false**
Cache Expires (seconds)	**60**
Fork Render	**false**
Fork Render Timeout	
Forkable	**false**
Client Classifications	
Event Handlers	**No event handlers**
Offer As Remote	**true**
Content Presentation Cl	
Content Presentation St	
Portlet Titlebar	
Icon URI	
Help URI	

Description

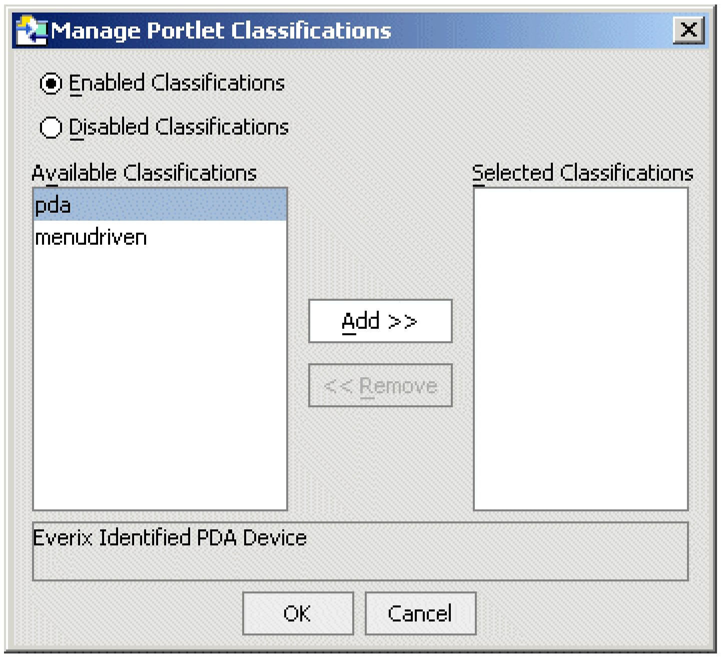
Optional. Select the multichannel devices with which the portlet can be viewed. The dialog list of devices

3. Click the drop-down list in the **Client Classifications** field in the portlet Property Editor.

Note: You may need to click in the blank area around the Portlet Designer to display the properties in the Property Editor.

4. The Manage Portlet Classifications dialog is displayed.

   Manage Portlet Classifications



(●) Enabled Classifications [✓]
( ) Disabled Classifications

Everix Identified PDA Device

5. Click **Enabled Classifications**.
6. Then add the “pda” and “menudriven” classifications to the **Selected Classifications** section by selecting them and clicking **Add**.
7. Click **OK**. The content will now be delivered only to PDA and menu-driven mobile devices and not to full browsers.

Additional Information

Documentation

Please see:

- Mobility Extension for BEA WebLogic Workshop 8.1: Mobilising the Sample Project (1.2 MB, requires Adobe Acrobat Reader)
- Dev2dev Articles
- *Using The Image Converter Guide*

Further Assistance

For any further assistance, contact BEA Systems, Inc.