Introduction

Designed to help customer administrators and managers quickly make their Web support experience available on the major mobile platforms, this Oracle best practice guide clearly identifies the steps to be taken when using Oracle RightNow Customer Portal Cloud Service to deploy the mobile pages that define your customers’ mobile experience on smartphones or tablet devices. It highlights the key elements for properly branding and adjusting the page content to provide the best-possible experience for mobile users. As with a cookbook, rather than reading it cover to cover, most readers will look for the necessary steps provided for their task at hand. Many of the steps, particularly the expert steps, are situational and depend on the objectives of your mobile project. They guide consultants and developers who want to get up to speed on how mobile pages are structured and how they can be branded.

Please note that this guide does not cover creation of widgets to extend your mobile experience. Please refer to existing documentation\(^1\) on Oracle RightNow Customer Portal Cloud Service for an overview of the widget creation process. Creating widgets for mobile is identical to the process of creating widgets for any other Oracle RightNow Customer Portal Cloud Service page.

\(^1\) community.rightnow.com/developer/fileexchange/Customer_Portal/Default.htm
Getting Mobile

This section outlines the actions you need to take to review, edit, test, and deploy the pages that define your mobile experience. Expert sections—noted with a label below the section header—are specifically for developers.

In each section, you will encounter a “Did You Know?” or “Important to Note!” reference. These sections embody helpful tips, tricks, and information about the mobile pages and how you can use them—in some cases, giving examples of their use.

Adopting the best practices and making the changes defined within this document anticipate a certain level of familiarity with Oracle RightNow Customer Portal Cloud Service and the tools and capabilities that come with that product. For reasons of brevity, this guide does not define all the terms that someone familiar with Oracle RightNow Customer Portal Cloud Service should already have some exposure to and experience with. So terms such as *WebDAV*, covered in detail in the product’s formal documentation, are not defined within this document. If you are new to Oracle RightNow Customer Portal Cloud Service, you should review the product documentation before you start attempting to undertake the tasks and steps presented here.

Finally, even though this document endeavors to identify changes in functionality and the releases the changes are associated with, the steps and some of the illustrative screen shots of the results may be different in your version of the product. Always consult your release notes and the product manuals if you see a mismatch between a task and how your product version supports it. Product manuals should always be considered the single source of authority and current and correct if there is a small difference or mismatch.

Reviewing Mobile Pages

Reviewing mobile pages for the first time is relatively straightforward. To review pages, you need a login to Oracle RightNow Customer Portal Cloud Service, with permission to view and access the Customer Portal Administration pages.

Follow these steps:

1. Log into the Customer Portal Administration pages (at http://<your site url>/ci/admin), as in the example in Figure 1.
2. If you are the first person to view your mobile pages, click on the **Page Set Mappings** button or link. If someone has already enabled one or more mobile page sets, you can skip steps 2 and 3.

3. If all of the page set mappings are disabled (they all have the **Enable** link visible, and the table is covered by hatch line shading), enable one or more of the page sets (three of them—the first three outlined in Figure 2—are provided out of the box). When selecting one, you should see a change like the one in Figure 3, with the Android setting enabled. That particular line will be updated. (The **Enable** link changes to **Disable**, and the row turns gray).
Figure 3. When you’re enabling changes to a setting on the Page Set Mapping page, the row turns gray.

4. With one or more entries enabled, you are able to review the pages in development mode. (Enabling a page set doesn’t make it available to your customers; you would have to deploy it first, as described in a later section). Click the **Back** button and then the **Set Environment** button or link. You will see a screen like the one in Figure 4. Select the **Development** link from the first box, and after deselecting the checkbox on the second question—“Allow my current browser ...,”—select the mobile environment from the second drop-down menu and then select **Set viewing mode and view the site**.

Figure 4. Setting the environment enables you to look at your mobile pages.
5. You are now looking at your mobile pages as they will appear to a consumer on the appropriate
device (in this case, on an Android browser). You may want to resize your experience to the size of a
typical mobile device’s screen, but you can test the full experience.

**Did you know?** The best desktop browser for reviewing your pages will not be Microsoft Internet
Explorer. Google Chrome or later releases of Firefox will provide better experiences, because their
browsers support the styling elements for WebKit and Mozilla that are built into the Oracle RightNow
Customer Portal Cloud Service stylesheets. The example in Figure 5 illustrates the differences between
Internet Explorer (left) and the Chrome experience (right).

**Did you know?** The latest versions of Oracle RightNow Customer Portal Cloud Service provide two
out-of-the-box mappings by default for Apple iPhone and Android. Older versions of the product also
supported WebOS devices and came with page set mapping for WebOS. The mappings for supported
versions are always the top items on the Page Set Mapping page. You can enable or disable each of
these settings and augment the list with your own entries, as you can see in Figures 2 and 3, where the
page set mapper has been modified to add several more entries identifying other mobile platforms.

![Figure 5](image-url)

Figure 5. You’ll see some differences between Microsoft Internet Explorer and Google Chrome mobile page sets on
your desktop browser.
Determining the Devices That Will Be Supported

Now that you are familiar with mobile pages, you are ready to use the functionality in Oracle RightNow Customer Portal Cloud Service that enables you to manage specific devices. To review the pages, you need a login to Oracle RightNow Customer Portal Cloud Service with permissions to view and access the Customer Portal Administration pages.

Follow these steps:

1. Log into the Customer Portal Administration pages (at http://<your site url>/ci/admin), as in the example in Figure 6.

![Dashboard](image)

Figure 6. Log into the Customer Portal Administration pages.

2. Click the **Page Set Mappings** link, which takes you to the Page Set Mapping page, where you can review your pages (see Figure 7).
3. Click the **Add Mapping** button. You will see a new row appear below any existing settings, as in Figure 8.

4. Enter the regular expression for the device you want to add. The example that follows adds a mapping for a game system browser you want to support. The major effort here, particularly if you are directing back to an existing page set, is to work out the characteristics of the game system you need to capture.

That information is made available within something called a user agent, which is an identifier that any Web browser provides to a Web server when it requests a page. To find your user agent, you can do one of several things. You can log into the Customer Portal Administration pages through your target device and navigate to the **Set Environment** button or link. That will tell you the user agent for your device, such as in the example in Figure 9.
For those who can't do on-device testing, there are great Websites that document the various available user agents. One to try is xytrax.com (zytrax.com/tech/web/browser_ids.htm). In this case, the user agent of the product you are looking to support is

Opera/9.30 (Nintendo Wii; U; ; 2047-7; en).

Like most other user agents, it shows the browser (Opera 9.30), the language of the device ("en," for English), and the identity of the device. As browser versions change, RightNow Customer Portal Cloud Service will identify it based on a constant: the name of the device. Enter that device into the row you created (see Figure 10). Precede and follow the name of the device with forward slashes (/) to represent a distinct regular expression instance, because it is entirely possible within Oracle RightNow Customer Portal Cloud Service to have multiple individual regular expressions on a single line. You could check on the browser, device, and language if you wanted to.
5. Click **Save** to save the entry. In the future (after you test and deploy your pages in a later step), any Nintendo Wii system that accesses the support pages, chat, or any other component of the service experience will be directed to the “mobile” page set.

**Important to know.** Oracle RightNow Customer Portal Cloud Service uses Perl-like string matching for regular expressions. If you want to find both the platform and the browser type in the user agent string, it may look like this:

```
/Opera .+Nintendo Wii/i
```

The order is important, as the regular expression is defined in sequence. At the end of the regular expression, `/i` checks for all matches, regardless of case.

Creating a New Mobile Page Set

The next step is to create a new mobile page set. There are several reasons a mobile page set may be required, such as the following:

- **A need to offer a different experience on different devices.** Some of the devices you want to support may have different screen dimensions; different input methods (a pointer instead of touch, for example); or even a limited keyboard, such as with most game system controllers.
• **The use of fundamentally different stylesheets on each device.** Although the themes capability in Oracle RightNow Customer Portal Cloud Service will enable you to apply different stylesheets to the same page set, you may run into a situation in which the page design for specific devices is so specific that a separate page set makes sense. Contrast the dimensions and interaction requirements for an Apple iPad with those of a smaller smartphone, for example. Clearly, this represents an opportunity to offer two very different experiences, each tailored to fit the different dimensions and capabilities of the device.

• **Segmentation.** You may want to offer different answers for a specific device or make targeted marketing offers to iPad users, for example.

Creating a new mobile page set is very straightforward but requires access to your WebDAV-accessible pages on your Oracle RightNow Customer Portal Cloud Service instance and the administration experience shown in previous entries in this document. The following steps move through the process, illustrating a best practice.

1. A best practice is to copy an existing page set and make changes to the copy. Through WebDAV, copy the mobile page set and rename it for the target device—an iPad, in this example (see Figure 11).

![Figure 11. The mobile page set folder is highlighted here.](image-url)

---

2 Ibid.
2. Page setting mappings will map onto folders in /euf/development/views/pages. Copy the folder on your desktop (Ctrl-C), and then type Ctrl-V to save the folder (in this example, renaming it iPad), as in Figure 12.

![Figure 12. Rename the copied folder (highlighted).](image)

3. The next step is to add the mapping you want for this new page set. The mapping under **Page Set** is straight to the folder name, as you can see in Figure 13.

![Figure 13. Add the iPad reference to the page set mapper.](image)

**Did you know?** In following the above steps, you have just copied the pages. You can change widgets and pages within your new folders to your heart’s content and create that distinct experience, but you need to keep in mind that these pages use a template and themes associated with the out-of-the-box page set. If you want these pages to use a different template or page styling, you will ultimately have to
copy and modify those files too. Templates and styling pages will be in the following directories, the mobile versions of which you should also copy and edit.

- Templates: /euf/development/views/templates/
- Themes: /euf/assets/themes/

Adding Your Brand to the Mobile Experience

The next task you may be interested in is adding navigation items and branding to the mobile experience. For the most part, doing this requires making changes to two sets of files: the template files and the stylesheet assets associated with your mobile theme. When you are happy with the changes, you will want to test them and redeploy. Adding a new template will typically require you to edit the header (see Figure 14) of every file in your pages folder.

To follow this series of steps, you will need administrator access to the customer portal pages on WebDAV. You will also want to make sure that you have some level of expertise with Cascading Style Sheets (CSS) and HTML.

The pages you copied in an earlier set of steps are still using the out-of-the-box mobile experience, with separate templates and themes for making the appropriate changes. Before making changes, consider the structure and content of the template and then look at the part of those pages that are critical for branding purposes.
The header component is the one that appears at the top of a mobile page. It defines any navigation (out of the box) and should be used to define any branding components not referenced in the stylesheets. Increasing the vertical size of that component isn’t recommended (see the best practices in the final section of this document). Consumers shouldn’t have to navigate down the page to see the content, so try to keep branding and navigation minimal (at least at the top of the page).

Image logos can be defined in the template or the stylesheet for the site. The best practice is to define them in the stylesheet. In the above example, it’s defined in the mobile template file as the first entry under rn_Navigation.

The following steps describe the process of changing the color palette used in the header—specifically the background color and gradation. This process requires making some changes to the site CSS file associated with the theme.

1. The first step is to find the file to which you will be making changes. Every template should have at least one theme, so first inspect the template header to find out where that template is placed.
2. The theme defined—and the stylesheet associated with it—is in the assets folder.

3. There are three main components in the themes folder (see Figure 15).
   - The site.css file, which governs high-level styling for the experience (including colors and fonts)
   - The display CSS for the various widgets contained in the experience
   - The images associated with the experience

Figure 15: The themes folder has three main components.
For now, focus on the site.css file, restricting your work to the elements covered by the header file. Open that file, and look at the contents for the header, the top-level element governing the content at the top of the page. This example shows you how to change the shading of the color gradient in the header in the mobile project. Look at the gradient elements defined for the header element (highlighted in the following code).

```css
header {
    background: #7F93AD;
    background: -moz-linear-gradient(top, #B0BCCD, #8195AF, #6D84A2);
    background: -webkit-gradient(linear, 0% 0%, 0% 100%, color-stop(0.0, #B0BCCD), color-stop(0.5, #8195AF), color-stop(1.0, #6D84A2));
    color: #FFF;
    min-height: 43px;
    overflow: visible;
    padding: 0 8px;
    text-align: center;
    text-shadow: 0 1px 3px rgba(0, 0, 0, 0.7);
}
```

4. Adjust the styling for these two elements. One of the best ways to test before you edit is by using Firebug (getfirebug.com) and Firefox. Firebug is an add-in for the Firefox browser that significantly improves your ability to review, troubleshoot, and debug Web pages. When you open Firebug on the mobile display pages and inspect the header element, you see the header element definitions (see Figure 16). The following example updates a component of the header style definition to show you how easy it is to experiment in Firebug without having to edit the site.css file first.

Figure 16. Inspect the header styling in Firebug.
5. The example makes a very trivial change—updating the three components of the \texttt{-moz} shader to red, green, and blue shades—to illustrate the process of trying different shading components (see Figure 17). Similar tests can be used to change the URL of the image and switch the menu or Search button in the header on or off. (Because Firebug enables you to make “review only” changes to both the displayed page and its styling, all changes are lost whenever the page is refreshed in the browser). In Firebug you can change each element definition by placing your cursor over the respective attribute and clicking to edit. To make these changes final, update the site.css file with the new settings and upload them to your Oracle RightNow Customer Portal Cloud Service instance through WebDAV. You can use the same process—making changes to the entries in the header component of the template file—to add or remove buttons.

![Figure 17: Modify the background gradients in Firebug.](image)

**Important to note!** The reason the Internet Explorer and BlackBerry browsers render out-of-the-box pages poorly is because both browsers (with the exception of the BlackBerry’s newer WebKit-based BlackBerry browser introduced in OS 6.x) ignore Mozilla (-moz) or WebKit (-webkit) styling definition entries in the stylesheet files (like the one in Figure 17). If you plan to add support for Internet Explorer mobile and older BlackBerry browsers, adding new styling entries these browsers can read can often be the most significant changes you will need to make to your mobile experience.

**Did you know?** You cannot see WebKit definitions in Firebug, because Firefox doesn’t support them. The Firebug inspector displays only the elements that the Firefox browser can interpret. Firefox ignores the WebKit definitions. Keep that in mind when making your final changes to your design, particularly if you are testing primarily on a single platform.

**Did you know?** Firebug Inspect and Chrome Inspect Element are great tools for reviewing the content of pages and understanding page structure. Chrome Inspect Element is an out-of-the-box feature of the Google Chrome browser, and Firebug is a very popular add-on for the Firefox browser.
that accelerates the process of styling a page. Go to **Tools -> Add Ons** from your Firefox browser to find the tool and add it to your browser.

**Segmenting the Experience**

**Expert**

Themes are a great feature within Oracle RightNow Customer Portal Cloud Service for swapping out look-and-feel elements, depending on what you know about the consumers, their navigation to your pages, the information you collect from them as part of the experience, and the behaviors they exhibit as part of their visit. To provide a segmented experience (specifically for mobile), you want to intercept whatever values you want to segment within a component called the `pre_page_render` hook. When you have intercepted the information that determines whether a new look and feel needs to be generated, you need to “swap” themes. Because themes can be used to swap out styling and layout and even hide or display widgets based on the visibility of the div elements they are based on, you can make rather significant changes to the experience you offer.

To complete this step, you will need to have some expertise with PHP and familiarity with the hooks component. You will also need access to your files and folders through WebDAV.

1. In the **hooks.php** file, define a `pre_page_render` hook that can be used to execute the segmented experience. The **hooks.php** file is available through `euf/development/config`. Make a new entry in that file to reference the segmentation hook. In this case, the changes made to that file are as follows:

   ```php
   $rnHooks['pre_page_render'][] = array(  
     'class' => 'Themeswitch',  
     'function' => 'themeswitcher',  
     'filepath' => ''  
   );
   
   Next, build a new model in the `/development/models/custom` folder (a value for `filepath` doesn’t need to be defined, because `filepath` needs to be modified only if the Themeswitch file is in a subdirectory). Let’s take a look at the contents of the **Themeswitch** file:

   ```php
   <?php
   class Themeswitch extends Model  
   {  
     function __construct()  
     {  
       Parent::__construct();  
     }  
     function themeswitcher()  
     {  
       // return if not logged in  
       if (!isLoggedIn()) {  
         Return;  
       }  
     }  
   }  
   
   $CI =& get_instance();  
   $c_id = $CI->session->getProfileData('c_id');  
   $CI->load->model('standard/Contact_model');
   ```
$contact=$CI->contact_model->get($c_id);
//find the gender custom field
$gender = null;
foreach ($contact->custom_fields as $cf) {
    if ($cf->col_name == 'c$gender') {
        $gender = $cf;
        Break;
    }
}
if(is_null ($gender)) {
    exit('ERROR – gender custom field not found');
}
//depending on the value of gender, load the appropriate theme
if($gender->menu_items[$gender->value]=='Female') {
    $CI->themes->setTheme('/euf/assets/themes/mobile_female');
} else {
    $CI->themes->setTheme('/euf/assets/themes/mobile_male');
}

3. If you set a custom field to record gender in the contact record, adjust the theme appropriately (see Figure 18). This example copied the standard theme, named it “mobile_female” and “mobile_male,” and made several changes to incorporate gender-specific images in the template and styling—hiding or showing a guided assistance widget for guidance that is gender-specific.

4. Updating the template is the last part of the equation. Make the following changes to your template file, replacing any current theme definitions with the following:

<rn:theme path="/euf/assets/themes/mobile_male" css="site.css" />
<rn:theme path="/euf/assets/themes/mobile_female" css="site.css" />

Figure 18. Build the segmented mobile experience in the customer portal, here according to gender.
Important to note! One thing to be careful about when building hooks is to make sure that the mobile-specific theme is used only during a mobile-specific experience. Check on the loaded page set in your pre_page_render hook. Otherwise, you will throw errors if you try to change themes that aren’t defined in any other active page set template. To avoid that issue, embed your theme swapper in a conditional that checks for the active page set, as in the example below. The name of the page set maps onto the name defined within the page set mapper tool.

```php
$pageSet = $CI->getPageSetPath();
if($pageSet === 'mobile'){
    //...do themey stuff
}
```

Did you know? The simplest way to do segmentation on mobile pages is to use the rn:condition tag to provide simple conditional sections. This tag can take advantage of a limited set of qualifiers (the number of searches the consumer has performed, for example) to define content to display. One of the more straightforward examples is to use rn:condition to show the chat link only for authenticated customers with a service-level agreement (SLA) to use the chat tool. Here’s an example of how to do that:

```xml
<rn:condition sla="chat ">
Your link to chat on your mobile pages....
</rn:condition>
```

Embedding Single-Question-at-a-Time Guides in Mobile Content or Navigation

Using guides is a great way to enhance the mobile experience. They complement the tactile experience delivered through a mobile device—on which a user is likely to be more willing to tap the interface to navigate quickly to necessary content than to read through a long answer. Within the out-of-the-box experience on the detail page, we reference by default an option to show a cascading answer on a pop-up page. However, this might not be an ideal experience for your mobile customers. The pop-up can confuse customers by interfering with browse navigation on some browsers, particularly older mobile browsers that do not support tabbed browser windows. Longer guides can also be less useful on a mobile device (where the screen real estate is smaller), because they force users to spend too much time scrolling.

To perform this change you should be comfortable with making markup changes and have access to your mobile files through WebDAV.

1. Log into WebDAV, and navigate to the /euf/development/views/pages/mobile/answers folder.
2. Open the details file for editing.

```xml
<n:meta title="#rn:php:SEO::getDynamicTitle('answer', getUrlParm('a_id'))#" template="mobile.php" answer_details="true" clickstream="answer_view"/>
<section id="rn_PageTitle" class="rn_AnswerDetail">
    <h1 id="rn_Summary"><rn:field name="answers.summary" highlight="true"></h1>
    <div id="rn_AnswerInfo">
```
2. To complete your work, just remove the popup_window_url attribute and add
   single_question_display = "true" in its place. Without this attribute, the solution will default to
cascading guides on a pop-up page. Save your changes, apply, test, and deploy!

Important to note! Avoiding the pop-up attribute options for guide widgets on mobile pages is
recommended, even though by default in Oracle RightNow Customer Portal Cloud Service, they are
defined as a standard attribute within mobile pages. Although they do use screen real estate more
effectively, pop-up pages tend to break the browse history within the older generation of mobile
browsers and make the process of navigating back to earlier pages in the experience more difficult.

Delivering the Mobile Experience on a Different Interface to the Desktop Experience

Providing different mobile experiences on the same interface is easy, but directing customers to
different interfaces for mobile experiences takes some work. There are times, however, when you may
want to do this. For example, you may want to separate out answers for desktop and mobile use,
because elements, such as Adobe Flash embeds, don’t translate well to all mobile devices but may
already be used within your answers since they are more accessible on desktop browsers. Although
there are ways to do that on a single interface, the cleanest approach is often to have a different URL.
Let's explore how to do that for your iPhone customers who cannot consume Flash content.

To perform this task, you will need access to the Customer Portal Administration pages and a login
for authentication.
1. Log into the Customer Portal Administration pages (at http://<your site url>/ci/admin), as in the example in Figure 19.

![Figure 19: Log into the Customer Portal Administration pages.]

2. Click the **Page Set Mappings** link to review your pages (see Figure 20).

![Figure 20: Review your pages in the page set mapper.]

3. Leave the existing iPhone entry disabled. Click the **Add Mapping** button. You will see a new row appear below any existing settings, as in Figure 21.
Figure 21: When adding a new mapping, you will see a new row appear.

4. Enter the regular expression for the iPhone. The page set entry, instead of being a relative mapping to the mobile page set, will now use an absolute address for the target interface. This mapping will be to the absolute URL address http://<site URL>/app, as in Figure 22.

Figure 22: Create an absolute address in the page set mapper.

5. Now that the traffic is being redirected to another interface, the next step is to go into the other interface and make sure that the appropriate mapping is set. In this case with the iPhone, make sure an iPhone-specific mapping is active on the target interface; otherwise, users will see the out-of-the-box standard desktop experience. Now all the iPhone traffic after deployment will be forwarded to an interface with exposed content that is exclusively mobile and stringently segregates answer views to the iPhone only.

**Important to note!** Don’t forget that last step. It’s an easy one to forget.

**Single Sign-on from a Mobile App via Pass-through Authentication**

**Expert**

One of the challenges for a “mixed mode” app—mixed mode being where the mobile Web experience is encased within a native app—is when authentication is being handled by the app and the pages have to “know” who the customer is (for incident submission) and what that person’s privileges are (for channel access or answer report filtering).

There are many techniques for addressing these kinds of requirements, from pass-through to external validation through a service such as OpenID. The next section presents a simple example that takes an e-mail address passed across to the embedded browser when the browser control is displayed—with the e-mail encoded and passed across as an argument on the URL. The example decodes that value,
keeping it available for prepopulation when the user navigates to an Ask a Question or Chat Launch page. Taking attributes you are capturing in the native app experience that may have a service or support use and making those available within the browser plug-in for use will always be a best practice. You never want to force the user to reauthenticate to different parts of the mixed-mode experience. Where possible, try to reduce user keyboard entry to the minimum required to provide an optimal experience.

To complete this task, you will need to have a reasonable grasp of PHP, an understanding of how widgets are constructed in Oracle RightNow Customer Portal Cloud Service, and familiarity with the process of creating custom widgets.

1. Start with a widget controller. Within that controller, you need to grab the value of the e-mail URL parameter and add it to a session cookie you can inspect later.

```php
//also grab the email address URL parameter and store within session
(to be used on ask page)
$email = getUrlParm('email');
if($email) {
    //now store within the session
    $this->CI->session->setSessionData(array('email'=>$email));
}
```

2. On the Ask page, another custom widget reads in the value, so you can reference the value in an input widget on your form, as shown in the highlighted lines below.

```php
class EmailAddress extends Widget
{
    function __construct()
    {
        parent::__construct();
    }
    function generateWidgetInformation()
    {
    }
    function getData()
    {
        $email = $this->CI->session->getSessionData('email');
        if($email) {
            $this->CI->load->model('custom/Encrypt_model');
            $email = $this->CI->Encrypt_model->decodeAndDecrypt($email);
            if($this->isEmailValid($email)) {
                $this->data['email'] = $email;
            }
        }
    }
```
private function isEmailValid($email)
{
    $emailPattern = sprintf("^%s$", trim(getConfig(DE_VALID_EMAIL_PATTERN)));
    $emailPattern = ereg_replace('\w', '0-9A-Za-z_', $emailPattern);

    //Check if emails are valid
    if(strlen(getConfig(DE_VALID_EMAIL_PATTERN)))
    {
        if($email && !ereg($emailPattern, $email)) {
            return false;
        }
    }
    return true;
}

3. Grab the e-mail address from the session data, and then decrypt the value. Within the view.php file:

4. The above section updates the standard/input/FormInput widget with the default value being the e-mail address variable populated within the controller.php file. This takes the e-mail address and prefills it in the Ask a Question page, saving the user from unnecessary keyboard entry.

Did you know? The biggest challenge for any mixed-mode solution can be preserving the state of the user interaction for before and after you move the interaction to an embedded browser. Where do you want customers to go to when navigating away from the browser control? Is it to the last app page they viewed, or will a navigation choice take them back to their chosen page? Do you need to preserve and make available identity or another attribute that was defined within the mobile Web interaction when a user is directed back in the native app? If either of these cases applies, you will need to save that state, possibly both before and after the browser is first presented or discarded. To learn more about state saving for mobile apps, see xoriant.com/blog/mobile-application-development/saving-the-state-of-iphone-app.html, http://www.eigo.co.uk/Managing-State-in-an-Android-Activity.aspx, and stackoverflow.com/questions/2566485/webview-and-cookies-on-android for some helpful tips on saving state.
Important to note! The highlighted section in the above code snippet also does some decryption of the provided attribute (the e-mail address isn’t being provided in clear text). The Encrypt_model custom model was removed to save space in this example. If you are interested in seeing how to build a model to decrypt encoded/encrypted content (authentication or state variables), have a look at the entry in the appendix for an example.

Ensuring That the Mobile Experience Is Appropriately Optimized for Custom Pages

Expert

One of the easiest things to forget when designing a mobile experience is the importance of performance. Making sure the page is optimized for the mobile experience is about much more than the assets loaded onto the page; it also concerns the JavaScript libraries loaded to support it.

Loading the default libraries from the desktop experience will load a bigger footprint than the out-of-the-box pages on mobile devices. This option is not recommended. To address this, make a simple change to the page template to load a smaller JavaScript framework module that will not load library components that aren’t typically utilized on mobile pages.

To make this change, you will need to have WebDAV access to your development folders.

1. Within your template or pages, add the following javascript_module entry: mobile_may_10. (On the first line, the mobile_may_10 entry is not specific to the May release—the date refers to the initial availability of javascript_module, although there aren’t JavaScript modules for each specific release).

Note that excluding the javascript_module entry from your rn:meta tag will, for upgradability purposes, load the bigger frameworks designed for desktop browsers—something to look out for if you are experiencing performance challenges with your mobile experience.

```
<rn:meta javascript_module="mobile_may_10"/>

<!DOCTYPE html>
<html lang="#rn:language_code#">
<head>
    <meta name="viewport" content="width=device-width; initial-scale=1.0; minimum-scale=1.0; maximum-scale=1.0; user-scalable=no;"/>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <meta http-equiv="Content-Style-Type" content="text/css"/>
    <meta http-equiv="Content-Script-Type" content="text/javascript"/>
    <title><rn:page_title/></title>
    <rn:theme path="/euf/assets/themes/mobile" css="site.css"/>
    <rn:head_content/>
```
Deploying Your Mobile Pages

The final activity, after making all your mobile-related mappings and changes is to deploy your pages. This last step should be done after you have reviewed the mapping settings you have on your customer portal pages and completing any UAT on the mobile experience. Once you are satisfied that those mappings are sufficient for the devices you intend to support (at least one enabled setting is needed for the mobile pages to be made available), go to the Deployment Manager within the agent desktop or, as of the February 2011 release of RightNow Customer Portal (now known as Oracle RightNow Customer Portal Cloud Service) make use of identical functionality available on the Customer Portal Administration pages from the Deploy menu.

You will need access to the Deployment Manager in either location to complete this task.

The following instructions provide guidance on deploying on November 2010 or earlier releases of RightNow Customer Portal.

1. Navigate to the Deployment Manager at Configuration -> Site Configuration -> Customer Portal (see Figure 23).
2. Once you’ve selected the correct interface, click the Deploy button on the main menu bar. This will deploy your pages. Congratulations! You are mobile!

The following instructions provide guidance on deploying on February 2011 or newer releases of RightNow Customer Portal (now Oracle RightNow Customer Portal Cloud Service), in which more-granular functionality for deploying customer portal sites or individual files was introduced. The major changes in these releases is in the degree of control over what components can be staged and deployed, thanks to a major revision of the Deployment Manager capability. The following steps cover the most common initial deployment scenario:

1. Navigate to the Deployment Manager, available at Configuration -> Site Configuration -> Customer Portal or from the Deploy menu on your Customer Portal Administration pages.
2. Select the interface to be deployed. In the File Select dialog box, accept the defaults (all files) and click Next.
3. Select the page set mapping configurations you want to migrate. Any items you want to move should have a Copy to Staging value selected. Click Next to move to staging.
4. After reviewing the modifications to be made, click the Stage button. This will move the files whose mappings you have been testing in development into staging. The main difference between development and staging is that staging compiles all your files to deliver the performance you would expect to see in the production area. Always do performance testing in staging rather than development.

Assuming that staging is completed successfully, your files and mappings are available in the staging environment. You have one more step to complete to make them available to your target consumers, and that involves moving them via Promote on your toolbar (on the desktop) or from a promotion page link on the Customer Portal Administration pages.

5. Clicking Promote takes you to a page that records the changes that will be made to the production environment when you promote your files. Again, check that any files you have been working on
and the right page set mapping changes are correctly referenced. Then click **Promote** and confirm. Congratulations! Your mobile pages are now live.

![Deployment Manager](image)

**Figure 23: Open the Deployment Manager**

**Did you know?** Have you deployed to mobile in haste and now need to make changes that require the mobile pages to be temporarily unavailable? The easiest way to undeploy mobile pages is to change the page mappings list to make each mapping unavailable. If you disable the individual rows in that form and then redeploy your pages/page-set mappings, following the instructions above, all users—regardless of device—will be directed to the default browser experience for that device, giving you time to make any mobile page tweaks.

**Important to note!** Not sure your mappings are correct? Didn’t test them thoroughly in development? Once you’ve deployed, any current mappings become **no longer editable** (see Figure 24). If you have made a mistake in a mapping, you will need to disable it, add a new entry to correct it, and then redeploy it.
Getting Visibility into the Devices Using Your Pages

A little-known (but incredibly valuable) addition as of the August 2010 release of RightNow Customer Portal—and currently available in Oracle RightNow Customer Portal Cloud Service—is analytics that enable you to see the browsers and operating systems being used by your consumers and the page sets in use on a site. These reports are tremendously valuable for identifying what devices customers are using to access your pages. Schedule these reports for regular review.

To view these reports, you will need access to analytics:

1. Open the Analytics option on the navigation bar, and select Reports Explorer from the Analytics Items list.

2. Navigate to Service -> Site Reports -> Customer Portal. You will see the list of new reports highlighting browser activity on your site.

3. Click Visits by Browser to review the list of browsers accessing your site, as shown in Figure 25.
Best Practices for Mobile

Figure 25: Oracle RightNow Customer Portal Cloud Service’s reporting capabilities enable you to see user visits by browser.

**Important to note!** The browsers will identify themselves by a shortened user agent name. It may not be readily apparent to the user that Safari 5, for example, is a mobile version of Apple’s browser. Use **Visits by Browser and Operating System** to get a fuller picture of the devices in use.

**Did you know?** Your chat agents should know when they are chatting with users who are on a mobile device. Make sure the **End-User Browser** field (see Figure 26) is included on the chat workspace, so agents can avoid sending large blocks of text or large graphic elements to mobile users. The **End-User Agents**, **End-User Browser**, **End-User Operating System**, and **End-User IP Address** fields are available as options on the workspace in Oracle RightNow Chat Cloud Service.
Figure 26: This chat workspace notifies the chat agent when a user is using a mobile device.

Addressing Mobile Through Best Practices

Now that you have started on the road to getting mobile, how do you ensure that the mobile Web experience you deliver is an excellent one? The following best practices will ensure a great mobile experience for your mobile-empowered consumers.

• **Performance is critical.** With the exception of gaming systems, mobile browsers are typically on wireless devices. Wireless connections—away from a wireless local area network (WLAN)—have to deal with high latency and poor connectivity. It’s critical that the page design and your answer content are not crowded with rich media that takes a long time to load. This also applies to chat interactions. Try to avoid sending graphic images as part of chat interactions. Consumer tolerance is key—users will be satisfied when they are not conscious of the time it is taking to load the page.

• **Be aware of platform differences.** Not all mobile browsers are made the same. Avoid Flash, for example, if you anticipate that your user population will be using an iPhone, iPod Touch, or iPad. Styling and stylesheets differ significantly between the various Web platforms. Even if you direct all mobile browsers to a single mobile page set, you may want to consider using themes functionality to offer tailored styling for each device type. BlackBerry and Microsoft browsers require modification of the out-of-the-box page set, for instance.

• **Limit key entry.** Key entry (particularly with onscreen keyboards for smaller devices) can be a painful experience. Make use of the Support home page to list most-used answers, and try to avoid forcing the user to search. Avoid free text entry when possible. Limit your custom fields on mobile
ask-a-question or chat request pages to those that are critical. When chatting with a user on a mobile device, ask questions in a way that minimizes typing by the user. For example, yes-or-no questions will lead to a better experience than open-ended questions that need to be answered in a text box.

- **Avoid attachments.** Don’t assume that mobile devices will have appropriate viewers for attachments. Avoid attachments, if possible, in your answers and other content.

- **Plan for touchscreen interaction.** Given the expectation that touch will become the dominant interaction technique, consider the ramifications for buttons and links in your design and content. Using a touchscreen is an imprecise method of interaction, so buttons should be large and easily selectable. User research shows that consumers with touchscreen devices like to select things and expect higher rates of answer feedback, for example, as a consequence of this preference. Links should be well spaced (horizontally and vertically) to help users avoid errors when making selections.

- **Up-and-down scrolling only.** Most mobile user interfaces (in native and Web apps) are more likely to use up-and-down scrolling than left-to-right. This has implications for content width. Where possible, use an alternative to tabular presentation or landscape type images, which may force horizontal (left-to-right) scrolling.

- **Don’t crowd the top of the page with navigation functions.** Users of a mobile browser are much more task-oriented and less likely to browse than other users. Take them straight to the content. Don’t distract them with clutter. Place any components that take up significant real estate (navigation link buttons, for example) toward the bottom of the page.

- **Make sure the content is suitable for the mobile context.** Make it easy to read, which includes not making your content overly long. Guides are particularly useful for compacting large amounts of text content. Take care in how you display your guide; a cascading guide may not provide the best experience. Font faces should be bigger than for the standard desktop experience. Don’t force your consumer to zoom in. Use clear and simple language. Limit content to what the user has requested, avoiding extraneous material.

- **Avoid pop-up browser windows where possible.** With a few notable exceptions, the tabs concept hasn’t reached the broader mobile Web yet. For most browsers, pop-ups will be displayed on their own page. This can make navigation back to originating pages a little difficult, because the Back button (if available on your browser) may be grayed out.

- **Provide user experience choices.** This is particularly true for the tablet experience. In the case of the iPad, the existing (desktop) Web experience is actually pretty good. The only things it lacks are the high performance of the mobile page set, plus button and links well suited to touch interaction. For tablet users specifically, you may want to provide the option of switching to/from both page sets, which is accomplished through the page set switcher widget (available as of the August 2010 release of RightNow Customer Portal).
Conclusion

By using Oracle RightNow Customer Portal Cloud Service’s mobile capabilities to review, edit, test, and deploy the mobile experience you offer, you can improve customer satisfaction, aligning the branding and content of your Web pages to complement mobile browsers. Institute the best practices outlined in this guide to quickly make customer support and branding opportunities available on major mobile platforms. Oracle RightNow Customer Portal Cloud Service’s mobile capabilities in particular are powerful and were designed to be as easy to use as possible, pointing you well on your way to meeting your mobile objectives.
Appendix: Decrypting URL Arguments Passed Through to an Embedded Mobile Browser

The $key variable represents the shared key. It is used on the app side to encrypt content, and you use it on your side for decryption. It should be a key with an identical value on both sides:

```php
<?php
class Encrypt_model extends Model {
    function __construct()
    {
        parent::__construct();
    }
    function decodeAndDecrypt($encrypted_str)
    {
        $key = "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX";
        $base64_decoded = base64_decode($encrypted_str);
        $str = $base64_decoded;

        if($str) {
            //call ske_buffer_decrypt API function
            //expose via config verb:
            SEC_CP_ADDTL_FCN_USAGE='ske_buffer_decrypt,encrypt_model.php'
            define (RSSL_KEYGEN_NONE, 9);
            define (RSSL_RSA_PAD_PKCS1, 1);
            $iv = NULL;

            //capture any error output of the decrypt function call
            (since it prints it to the screen...)
            ob_start();
            $original_str = ske_buffer_decrypt($str, $key, "aes256",
            RSSL_KEYGEN_NONE, RSSL_RSA_PAD_PKCS1, NULL, $iv);
            $encryptionErrorMessage = ob_get_clean();

            return $original_str;
        }
        else {
            return '';    
        }
}