

Oracle® Fusion Middleware

WebCenter Sites: Mobility Server Implementation Guide

11g Release 1 (11.1.1.8.0)

E42079-01

July 2013

Oracle Fusion Middleware WebCenter Sites: Mobility Server Implementation Guide, 11g Release 1 (11.1.1.8.0)

Copyright © 2011, 2012, 2013 Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle America, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

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.

This software or hardware and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Table of Contents

Contents

About Oracle WebCenter Sites: Mobility Server	6
Prerequisites to Building a Mobile Site	7
Considerations for Your Mobile Web Site	7
Third Party Services.....	8
Mobile Video and User-Generated Content	8
Access.....	9
Required Skills.....	9
WebCenter Sites Version 7.6 Installation	9
Mobility Server Installer.....	9
Text Editor.....	9
Discovery.....	10
Identifying Content Appropriate for Mobile.....	10
Identifying Sites.....	11
Identifying First Asset Types to Mobilize (most easily consumed).....	11
Creating a mobile sitemap (on paper)	11
Installation	13
Mapping	14
The Mapper module	15
Overview of Mappings	16
Overview of Field Mappings	18
Overview of Display Object Children	19
Mapping Walk-through.....	21
Storing Mapping Changes	25
Tasks in the Mapper Module	25
Attribute inheritance	25
Mapping Lists of Assets.....	26

Design.....	29
MSAdmin.....	29
User templates for site structure.....	29
User templates for look and feel	30
Template Folders and Workflow	31
Create Logos, Headers and Footers.....	31
Creating a custom mapped Display Object template	32
Combining Display Objects	32
Advanced Template Changes.....	34
Publishing.....	35
Appendices.....	36
Appendix 1: Display Objects	36
List of Display Objects	36
Advertisement	37
AppCacheMonitor.....	37
Bookmark.....	37
Carousel	40
Collapsible.....	40
Menu (deprecated).....	48
Detail.....	49
DropDownMenuGroup	49
Index.....	50
Listing.....	50
ListFilter.....	50
GeoDetail and GeoListing	51
ProductDetail and ProductListing	51
SimpleTable.....	51
TabGroup	53
Appendix 2: XML Examples.....	54
Site Navigation XML Example	54

Page XML Example.....	54
Appendix 3: Folder Structure.....	55
Default Mobility Server Folder Structure after Automated Install.....	55
Appendix 4: Offline Storage Guide	56
Introduction	56
Configuration	56
Creating the Cache Manifest	57
Notifying the User of Caching	57
Publishing and Delivery vs. Editorial Environments.....	57
Final Steps.....	58
The mobilityserver:updateAppCache task.....	58
The AppCacheMonitor Display Object.....	61
Appendix 5: RSS, Atom and other XML Feeds	62
Overview	62
Field Mappings.....	62
Sample RSS Feed.....	63
Advanced Mappings.....	64
OTHER REFERENCE DOCUMENTS	64

About Oracle WebCenter Sites: Mobility Server

1

Oracle WebCenter Sites: Mobility Server, the newest deployment solution in Oracle WebCenter Sites' encompassing Web Experience Management (WEM) suite, provides a single environment through which to create and manage dynamic content for thousands of mobile devices.

Mobility Server leverages the powerful CMS capabilities of Oracle's WebCenter Sites beneath a remarkably easy to use interface. All traditional web content can be easily configured for use on myriad types of mobile devices, in real time.

This guide walks you through the process of creating a new mobile site using Mobility Server.

We strongly recommend that you carefully review the information contained in this document as well as the *Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 Installation Guide* prior to installing Mobility Server.

Prerequisites to Building a Mobile Site

2

Considerations for Your Mobile Web Site

Oracle WebCenter Sites: Mobility Server allows customers to quickly bring their existing content, through existing workflows, to mobile sites that match their organization's web style guide while providing excellent usability across thousands of mobile devices. To accomplish this, Mobility Server takes a different approach to site creation and design than that of WebCenter Sites.

Instead of starting from a blank slate, Mobility Server ships with its own templates. These templates leverage **Display Objects**, which render discrete portions of the mobile site. Each Display Object has unique appearance and/or functional capabilities. These Display Objects have properties which are controlled using the **MSAdmin** WEM application (e.g. image size). Display Objects can be extended, and templates can be modified. However, for reasons described later in this document, we recommend that template changes are kept to a minimum.

This document will guide you through:

- Identifying which content you should consider displaying on your mobile site
- How to create your site structure
- Mapping content to Display Objects
- What content Mobility Server should display
- How to display them
- How to publish the mobile site to a delivery environment

This document will not cover styling of your mobile site in depth. The Mobility Server User Guide describes the rich library of properties that can be safely changed through MSAdmin to generate appropriate markup and styling for any device, from WAP 2.0 to the latest HTML5 and CSS3. These properties have been designed and tested to ensure compatibility across thousands of mobile phones and tablets. Additional changes can always be made by adding or overriding styles and markup though it is imperative that those changes are tested rigorously on a wide range of mobile devices and mobile browsers.

Third Party Services

Mobility Server is able to consume content from sources other than WebCenter Sites assets. There are two approaches to this: a) Mobility Server directly consuming RSS/ATOM or proprietary XML feeds, or b) Mobility Server consuming 3rd party content that is already exposed to WebCenter Sites.

Please see your account representative for more details.

Mobile Video and User-Generated Content

Mobility Server does not include support for mobile video transformation and playback or user-generated content. However, these offerings can be added to your mobile website through simple integration with third party hosted services provided by Oracle's partner, netomat, Inc.

Please speak with your account representative for details on contacting a netomat representative.

Access

You must have access to your Mobility Server installation's file-system. Unix command-line knowledge is necessary unless your system administrator has provided you with tools for accessing the file-system (e.g. FTP access).

Required Skills

HTML: Changes to the Mobility Server templates described in this document require only knowledge of HTML, although experience with any server-side language (JSP, PHP, ASP, etc.) or templating system is recommended.

WebCenter Sites Version 7.6 Installation

WebCenter Sites 7.6 or greater, with WEM, is a Mobility Server requirement. Mobility Server installs MSAdmin, a WEM application for configuring your mobile site in management environments.

Mobility Server Installer

This document assumes that you have procured Mobility Server and are ready to install it. We recommend that you wait until you have completed reading through Chapter 4 to begin installation.

Text Editor

During the design phase you will be asked to modify PHP templates, but *no PHP experience is necessary*. The changes, as you will see, should be very simple to anyone with HTML and any server-side programming experience. They can be done with any text editor, e.g. vim, emacs, notepad, etc.

Identifying Content Appropriate for Mobile

When designing your mobile site, it's important to keep in mind that the mobile site is not simply a "shrunk-down" version of your regular web site. Even as mobile devices are released with higher and higher screen resolutions, their screen sizes are limited when compared to desktop browsers. Furthermore, the limitations and unique capabilities of mobile browsers and devices have significant impact on what content can and should be presented and how it should be laid out, and your mobile site should be designed with these idiosyncrasies in mind. For instance, touch screen-based devices introduced new challenges and opportunities for mobile site design. Mobile designers quickly learned that the width between links or buttons on touch devices should be no less than a finger-tip width as links placed too close together would often lead users to the wrong page. Thus, it's an important exercise to determine which and how much content you want on your mobile site, especially your mobile home page.

Another important consideration is that mobile users are most often on-the-go, unlike desktop-browser users who are usually sitting at a desk. They are often time-constrained and visiting your site for a specific purpose rather than a desire to generally browse around. Therefore, ensure that their point of entry is uncluttered, loads quickly and important content is easy to locate. Typically, contact, location and timely event information should be easiest to access or navigate to. In many cases, this may mean that much of the information that resides on the homepage or one link away on a desktop browser is omitted from a mobile homepage, or possibly from the mobile site entirely.

Given the mobile device screen size and the different usage patterns typically seen on mobile, you may also wish to create content that is intended solely for the mobile experience. This may include shorter descriptions or summaries of existing content or new content such as location-based listings that leverage the device's GPS capabilities.

Identifying Sites

Mobility Server supports multiple WebCenter Sites sites per installation. You will need to provide a WebCenter Sites site name when configuring Mobility Server for storing installation-wide options. Mobility Server installation will install an MSAdmin WEM application, and it will also create an asset of the type, ***MobilityServerConfig***. This asset should never be directly modified, but it may be important during troubleshooting as it needs to be published for display property and other mobile site changes to appear in other environments.

Identifying First Asset Types to Mobilize (most easily consumed)

Mobility Server can expose content found in any asset that is exposed through WebCenter Sites' REST API, including images and, depending on the licensing agreement, video and other multimedia. Beginning with Mobility Server 2.0.1, this 'mapping' of WebCenter Sites asset types to Mobility Server can be achieved, and adjusted, without any code changes. A built-in administration utility called the Mapper module is available for this purpose.

In prior versions, Mobility Server did not follow associations between assets (with the exception of associations to image assets). At this time, an asset's associations are now automatically available without the need to map them. Refer to the Version 11.1.1.8.0 Template API Guide for more information on how to access associations.

Creating a mobile sitemap (on paper)

In some cases, it may be necessary to create new attributes in WebCenter Sites for existing asset types to support mobile content. If, for example, your desktop site's product descriptions are usually 1000 words, it may make sense to create a 'mobile_summary' or just 'summary' attribute that allows editors to describe a product in just 200 words. This may be helpful in some cases when content has been modeled using complex associations that should be simplified for mobile devices and mobile users.

Once asset types have been identified, it is helpful to create a sketch of the topology of the site. MSAdmin allows for many options to alter the design of your mobile site on the fly, but changes to the

site structure often require changes to WebCenter Sites templates, or to the Mobility Server templates. There are many approaches to organizing this content. There are four approaches that rely on WebCenter Sites' content organization:

1. As a new sitemap, just for the mobile site, driven by WebCenter Sites content organization
2. As a branch of the existing sitemap
3. Based on the existing sitemap, with omissions
4. Using a programmatic approach or tag to identify content from the sitemap that should appear on mobile.

For more information, please see the [Design](#) section.

Installation

4

Once you have a good idea of what you are building you can install Oracle WebCenter Sites: Mobility Server. You should begin with a development or editorial environment, as the configuration options for delivery environments may need to be tuned for your specific needs. The Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 Installation Guide describes the installation process, and Appendix 1 in that document can be reviewed when you are ready to install staging or delivery environments.

Mapping

5

Mobility Server uses **Display Objects** to represent discrete portions of the mobile site with unique appearance or functional capabilities. Each Display Object includes out-of-the-box templates specifically designed and rigorously tested to support Touch devices, Tablets, older Smartphone/Blackberry devices and Feature phones. Thus, by simply mapping web content to a Display Object, customers are ensured that their content will be displayed appropriately across thousands of different devices.

Each Display Object contains fields that will be mapped to WebCenter Sites asset attributes or XML elements in data sources. Display objects have default fields, but may be extended to contain hundreds of additional fields. Some Display Objects, such as listings, may be easily linked to other Display Objects. The table below lists the Display Objects that ship with Mobility Server and descriptions of each Display Object can be found in [Appendix 1](#). Additional Display Objects may be installed separately.

Display objects are the building blocks for your mobile site. A typical installation begins with only a single mapped Index Display Object; additional Display Objects must be mapped using the mapper module. In this section we will describe how to instantiate Display Objects and map content to them. Some Display Objects do not require a data source (e.g. [Advertisements](#)), some require only a single WebCenter Sites asset (e.g. [Detail](#)), while others accept custom data sources (e.g. the 'assetlist' or 'rssitemlist' source types for a Listing Display Object). It is recommended that you first read through this section, then begin by mapping a Detail Display Object to familiarize yourself with the basic mapping procedure.

List of Display Objects

Name	Fields	Requires Assets	Links to DOs	Contains Child DOs****
Advertisement	<i>None</i>	No	<i>n/a</i>	No

AppCacheMonitor	<i>None</i>	No	<i>n/a</i>	No
Bookmark	<i>None</i>	No	<i>n/a</i>	No
Carousel	headline, thumbnail (<i>image</i>)	Yes	Yes	No
Collapsible	<i>None</i>	No	No	Yes
Detail	author, body (<i>text</i>), headline, image (<i>image</i>)	Yes	<i>n/a</i>	No
DropdownMenuGroup	<i>None</i>	Yes	<i>n/a</i>	Yes
GeoDetail	city, country, description (<i>text</i>), name, state, street, zip	Yes	<i>n/a</i>	No
GeoListing	city, country, description (<i>text</i>), name, state, street, zip	Yes	Yes	No
Index	<i>None</i>	No	<i>n/a</i>	No
ListFilter	headline, description (<i>text</i>)	Yes	Yes	No
Listing	headline, description (<i>text</i>), gridImage***	Yes	Yes	No
Menu	<i>None</i>	Yes	Yes	No
ProductDetail	color, description (<i>text</i>), price, upc, name	Yes	<i>n/a</i>	No
ProductListing	color, description (<i>text</i>), price, upc, name	Yes	Yes	No
SimpleTable	<i>any**</i>	Yes	<i>n/a</i>	No
TabGroup	<i>None</i>	No	<i>n/a</i>	Yes

* non-string data types are shown in parenthesis.

** any and all mapped fields will be displayed automatically

*** tablet family will render listing differently if gridImage is mapped

**** child DOs are specified via the [mapper interface](#)

Note: More information about each Display Object is provided in [Appendix 1 Display Objects](#).

The Mapper module

The Mapper module is an administrative web-frontend tool that allows an administrator to control their site's Display Object Mappings. A "mapping" is defined as the link between an external content source (most likely, a subset of fields from a WebCenter Sites asset) and a Display Object to be used on a template.

Principally, the Mapper module lets an admin add, edit, delete, and view Display Object Mappings without the need to directly manipulate the database. In addition, the Mapper module also offers a view of content-data stored in Mobility Server and allows for the execution of certain administrative jobs (“tasks”) related to Mappings.

The Mapper module is available without the need for special configuration by navigating to:

```
http://<MobilityServer>/mapper
```

Authentication with WebCenter Sites is necessary to access the Mapper module.

The three sections of the Mapper module are: Mapping, Mapping Data, and Tasks. They can be accessed by clicking the links in the heading of any page in the module.

- **Mapping:** This is the main section of the module. In this section, an administrator can view, add, edit, and delete mappings and their fields.
- **Mapping Data:** This section offers a tabular view of the content-data stored in Mobility Server associated with each mapping. As users browse your mobile site, Mobility Server will populate its own database with content-data from the external sources pointed to by your mappings. This section shows you this content-data, which can be useful when troubleshooting.
- **Tasks:** This section allows for the execution of certain tasks related to mappings. These tasks can also be run directly with php from the command-line; they are accessible here for your convenience. Tasks are discussed in more detail in a later section.

Overview of Mappings

Below is a view of the Mapping table with several Display Objects already mapped and an explanation of the data:

ORACLE WebCenter Sites | Mobility

Logged in as: fwadmin | Logout [debug]

Mobility Server Control Panel

Mapping Mapping Data Tasks

[+ Add Mapping](#)

ID	Mapping Label	Display Object Type	Source Type	Source Value	CS Site	CS Type	Link Label	List Item Limit	Action
57	AllArticles	ListFilter	assetlist	/cs/Satellite?c=AVIArticle&pagename=SEUtils/...	avisports	AVIArticle	ASDetail		
54	AppCacheMonitor	AppCacheMonitor							
17	ASBaseball	Index	asset	1327351719525	avisports	Page			
21	ASBaseballAlsoInTheNews	Listing	assetlist	/cs/Satellite?c=AVIArticle&pagename=SEUtils/...	avisports	AVIArticle	ASDetail		
46	ASBaseballArticles	Index	asset	1327351719525	avisports	Page			
45	ASBaseballTabGroup	TabGroup			avisports				
20	ASBaseballTopStories	Listing	assetlist	/cs/Satellite?c=AVIArticle&pagename=SEUtils/...	avisports	AVIArticle	ASDetail		
48	ASBaseballWebDetail	Detail	atomyentry	http://api.flickr.com/services/fee...id=35034357755@N01&lang=en-us&format=atom					
47	ASBaseballWebList	Listing	atomyentrylist	http://api.flickr.com/services/fee...id=35034357755@N01&lang=en-us&format=atom			ASBaseballWebDetail	3	
56	ASBreakingNewsDetail	Detail	atomyentry	http://api.flickr.com/services/fee...set=72157632698401900&nsid=9...us&format=atom					

(1) (2) (3) (4) (5) (6) (7) >

netomat mobility server

ID: The primary key of the display_object row (automatically assigned by the system)

Mapping Label: A friendly name given to this mapping. This label is used in direct links to the mapping and in template-API functions.

Display Object Type: The Display Object used by this Mapping.

Source Type (optional): The type of content-source used by this Mapping. Most of these sources refer to WebCenter Sites, but sources for XML, RSS and Atom exist as well. Note that some Display Objects (e.g. Advertisement) do not require content-sources.

Source Value (optional): The exact content-source used by this Mapping. The type of this value depends on the Source Type selected. For instance: if the source type is "assetlist," the source value must be a URL to an assetlist JSP. If this source type is "rssitemlist," the source value must be a URL to an RSS feed. In the case of the assetlist source type, its source value can take any of the following three forms:

1. A full, valid URL, e.g.
`http://HOST:PORT/cs/Satellite?c=Product&pagename=SEUtils/Mobility/AssetTypeList`

2. A partial URI starting with a forward slash, e.g.
`/cs/Satellite?c=Product&pagename=SEUtils/Mobility/AssetTypeList`

This should be relative to the Content Server host and port that Mobility Server is configured to use.

3. A query string, to be appended to the `cs_satellite_uri` definition, e.g.
`c=Product&pagename=SEUtils/Mobility/AssetTypeList`

CS Site (optional): The WebCenter site with which the targeted asset type is associated, in the case that the **Source Type** of this mapping refers to WebCenter sites.

CS Type (optional): The asset type of the WebCenter sites asset being targeted, in the case that the **Source Type** of this mapping refers to WebCenter sites.

Link Label (optional): This references the **Mapping Label** of another mapping in the case that this Display Object creates a list of items.















List Item Limit (optional): Limits the number of items to be displayed for a Display Object that creates a list of items.

For more information on what data is needed by each Display Object, refer to [Appendix 1 Display Objects](#).

The following are reserved words in Mobility Server that should not be used for field names: *id*, *cid*, *typeId*, *categoryId*, *updateDate*.

Overview of Field Mappings

Each Mapping has a set of Field Mappings associated with it. These field mappings control what fields are available on templates for that mapping. Below is a view of a sample Field Mappings table and an explanation of the data:

ID	CS Field Name	CS Field Type	Field Mapping Label	
12	headline	string	Headline	 
13	subheadline	string	ByLine	 
14	author	string	Author	 
15	postDate	date	PostedDate	 
16	body	text	Body	 
17	relatedImage	string	Image	 
18	body	embeddedassets	BodyEmbeddedAssets	 

CS Field Name: If the currently selected mapping is for a WebCenter sites content-source, this field corresponds to an asset attribute. If the Mapping is for a syndication (RSS or Atom) or other XML source type this field is an XPath expression targeting an XML element in the data of that source. For more information, see [Appendix 5: RSS, Atom and other XML Feeds](#).

CS Field Type: The data type of the field. Can be one of the following: *string, date, double, image, video, integer, text, embeddedassets*.

Field Mapping Label: The label by which this field mapping is referred. These are case sensitive.

Overview of Display Object Children

Certain Display Objects serve as containers for multiple other Display Objects, such as the [Collapsible](#) and [TabGroup](#) Display Objects. These Display Objects do not support fields, but instead contain child Display Objects, as seen below:

Child Mapping ID	Child DO ID	Child DO Type	Sort order
20	25	ASCarousel	1
18	60	ASFrontPageTeasers	5
19	23	StoreList	50

[Edit Children](#)

Child Mapping ID: The database ID of the parent-child relationship. This is generated automatically.

Child DO ID: The database ID of the child display object. This is generated automatically.

Child DO Type: The specific Display Object mapping that is mapped as a child.

Sort Order: Defines the order in which child Display Objects appear. For the Collapsible Display Object, this is the top-to-bottom order of sections, and for the TabGroup Display Object, this is the order in which the tabs will appear.

Every Child Display Object can also have options -- in fact, for the Collapsible and TabGroup Display Objects, the label option is required to display the section title and the tab title, respectively.

Mobility Server Control Panel

Mapping Mapping Data Tasks

Editing Options for Child StoryCarousel of FrontPageTabGroup

Option name

Option value

+ Add Child Option

ID	Option name	Option value	Actions
24	label	Recent Stories	X

Edit Mapping Edit Children

Option Name: As of Mobility Server 11.1.1.8.0, only the "label" and "cid" options are supported option names.

Option Value: The corresponding value for the named option. In the example above, "Recent Stories" is the text that would appear as the 'label' for one of the tabs in the TabGroup Display Object; clicking on it would display the StoryCarousel mapping. When adding a child Display Object that refers to an "asset" source type, the option name can be set to 'cid' and you may also specify the 'cid' of a given asset to be included.

Mapping Walk-through

By now you should be able to tell that managing mappings with the module is as simple as finding the mapping in the main table and taking the desired action. Mappings can be edited and deleted while Field Mappings can be added or removed from a Mapping. The most important part of mapping is understanding what data is required for your Display Object of choice along with what fields are used on that Display Object's default template (refer to [Appendix 1 Display Objects](#)).

As an example, let's create a simple mapping between a WebCenter sites "Articles" asset type and a Mobility Server "Detail" Display Object. **Note:** Your WebCenter Sites installation may not have a "SampleSite" site, or an "Articles" asset type, in which case you can either create one or use another asset type of your choosing).

You can add a mapping by clicking the "Add Mapping" button above the mapping table on the main page of the mapping section and fill out the form as so:

Mobility Server Control Panel

Mapping

Mapping Data

Tasks

Adding a New Mapping

[Mapping](#) / Add Mapping

CS Site

SampleSite

CS Type

Articles

Warning: Changing the CS Type will cause all field mappings to be deleted.

Source Type

asset

Display Object Type

Detail

Mapping Label

Warning: Changing the Mapping Label will necessitate renaming custom templates and references to them.

Source Value

Link Label

List Item Limit



Add Mapping

Click "Add Mapping" to add the new mapping to the system.

You will then be forwarded to the "Edit Fields" page for your newly created mapping. Add each field with the "Add Field" form until the fields table looks like so:

ID	CS Field Name	CS Field Type	Field Mapping Label
193	headline	string	Headline
194	subheadline	string	ByLine
195	author	string	Author
196	postDate	date	PostedDate
197	body	text	Body
198	relatedImage	string	Image
199	body	embeddedassets	BodyEmbeddedAssets

You've now successfully created your first mapping. To test it, first select an asset of the asset type you mapped to and find its CID. Then, navigate to the page for your new mapping using this CID by navigating to this URL:

```
http://<MobilityServer>:<port>/do/MyArticleDetail/<CID>
```

You should see your mapped content rendered in the style used by the Detail default template.

If something went wrong, it may also be helpful to follow the mobility server log file, found by default under *mobilityserver/log/frontend_prod.log*, as this may provide useful information regarding errors in mapping or help identify common mistakes. Viewing your web server's error log may be helpful as well. The most common mistakes are that an attribute you mapped to doesn't exist for the given asset type or a required field was not mapped. Also make sure that the selected data types are correct.

Finally, you can navigate to the "Mapping Data" section of the module to see the WebCenter Sites asset data that Mobility Server stored in its database. It should look something like so:

Mobility Server Control Panel

Mapping		Mapping Data	Tasks			
ID	Mapping Label	CID	Fields			
1		1334880977833	No data available			
34	ASDetail	1328196048535	Field Mapping Label	CS Field Type	Value	
			Headline	string	Simple Ways to Introduce A New Generation to Tennis	
			ByLine	string	Fun Tips for the Whole Family	
			Author	string	TREVOR PATERSON	
			PostedDate	date	2011-03-01 14:08:09	
			Body	text	...	
			Image	string	AVImage:1327351718813	
			BodyEmbeddedAssets	embeddedassets	AVImage:1328196049609;AVImage:1327351719024	

1

Storing Mapping Changes

Before your newly mapped Display Object can be deployed to other environments you must store the mapping back to WebCenter Sites. This can be done either from the "tasks" section in the mapper module or by issuing the following command from the Mobility Server folder:

```
php symfony mobilityserver:storeMapping
```

Mapping changes cannot be published to other Content Server environments until this task is run.

Tasks in the Mapper Module

The "tasks" section of the Mapper Module allows you to run mapping-related tasks rather than having to do so from the command line. The following tasks are available:

mobilityserver:fetchMapping ("Fetch Mapping")

Fetches Display Object Mappings stored in WebCenter Sites and stores them in Mobility Server.

mobilityserver:storeMapping ("Store Mapping")

Stores all of the Display Object Mappings currently in Mobility Server, and stores them in WebCenter Sites.

mobilityserver:dumpMapping ("Dump Mapping")

Serializes all Display Object Mappings to a yml file that can be used as fixture data by doctrine.

mobilityserver:clearDisplayObjects ("Clear Display Objects")

Clears all stored assets out of the database.

Attribute inheritance

When a Display Object Field is mapped to an attribute of a WebCenter Sites asset, Mobility Server simulates inheritance from parent to child assets. If an attribute does not exist on a targeted asset, Mobility Server will attempt to find the attribute on the asset's parent. If found, Mobility Server will

function normally as if the value of the attribute came from the targeted asset. If not found, Mobility Server will not search higher in the asset hierarchy and will function as if the attribute does not exist.

Mapping Lists of Assets

It is sometimes necessary for Mobility Server to retrieve sets of assets, based on multiple criteria, for use in Display Objects. One way to accomplish that is to create new templates in WebCenter Sites that output an XML representation of the desired assets. The following is an example that illustrates how to create a custom template that will generate a list of assets for Mobility Server to consume. The XML generated by this example is suitable for the 'assetlist' source_type.

1. First, load the WebCenter Sites Advanced Interface, and choose the 'design' (sometimes 'Site Design', sometimes 'Developer') menu. Choose 'New' from the top left and select 'CSElement' to begin creation of a new CSElement.
 - a. Set assignees, and continue to the next page.
 - b. Enter memorable values for the 'Rootelement' and 'Element Storage Path/Filename' fields. Ex: Mobility/XMLforMS2 and Mobility/XMLforMS2.jsp
 - c. Enter the JSP from the table below and then continue to next page.

```
<%@ taglib prefix="cs" uri="futuretense_cs/ftcs1_0.tld" %>
<%@ taglib prefix="asset" uri="futuretense_cs/asset.tld" %>
<%@ taglib prefix="ics" uri="futuretense_cs/ics.tld" %>
<%@ taglib prefix="render" uri="futuretense_cs/render.tld" %>
<%@ page
import="COM.FutureTense.Interfaces.* ,COM.FutureTense.Util.ftMessage ,COM.
FutureTense.Util.ftErrors" %>

<cs:ftcs>

<!-- SEUtils/Mobility/asset typeList
INPUT
OUTPUT
--%>

<!-- Record dependencies for the SiteEntry and the CSElement --%>
<ics:if condition='<%=ics.GetVar("seid")!=null%'><ics:then>
    <render:logdep cid='<%=ics.GetVar("seid")%' c="SiteEntry"/>
</ics:then></ics:if>
<ics:if condition='<%=ics.GetVar("eid")!=null%'><ics:then>
    <render:logdep cid='<%=ics.GetVar("eid")%' c="CSElement"/>
</ics:then></ics:if>
```

```

<asset:search type='<%=ics.GetVar("c")%>' prefix="pfx"
what="id,updateddate" list="alist" />
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<page>
<ics:listloop listname="alist">
  <ics:listget fieldname="id" listname="alist" output="astid"/>
  <ics:listget fieldname="updateddate" listname="alist"
output="upd"/>
    <asset cid='<%=ics.GetVar("astid")%>'
type='<%=ics.GetVar("c")%>'
lastUpdated='<%=ics.GetVar("upd")%>' mobileenabled='true' />
</ics:listloop>
</page>
</cs:ftcs>

```

- d. The map fields can be left empty. Click save.
2. Next, choose 'new' and select 'SiteEntry' to begin creation of a new SiteEntry.
 - a. Name, pagename and root element should match the values chosen in step 1.b. Ex: 'Mobility/XMLforMS2' (for each).
 - b. **IMPORTANT:** Choose "Yes" for "Wrapper Page".
 - c. Under 'Pagelet parameters', enter a new parameter named 'cs.contenttype' with a value of 'text/xml'.
 - d. Under cache rules, choose 'uncached' for testing.
 - e. Click save.
3. Test your new page in the browser:

`http://HOST:PORT/CONTEXT/Satellite?c=ASSET_TYPE&pagename=Mobility/XMLforMS2`

Your browser may display warnings if the XML is not perfectly formed; in most cases, these warnings can be ignored, as Mobility Server should be able to overcome issues with white-space and other minor problems.

This new XML can be used to map Mobility Server Listing Display Objects: ProductListing, GeoListing, the more generic Listing or SimpleTable. To map a Listing, create a mapping with the following values:

CS Site: <your site>

CS Type: <your asset type>

Source Type: assetlist

Display Object Type: Listing

Mapping Label: MyListing

Source Value: c=<your asset type>&pagename=<your pagename>

(example: c=Content_C&pagename=Mobility/XMLforMS2)

Link Label: <label for a target Display Object that you've mapped>

Next, we'll need to map each field that Mobility Server needs for this newly mapped Display Object. For this Display Object listing, the only fields that are required are 'headline' and 'description'. If a short description field is available in your asset, we recommend using that; if not, Mobility Server will truncate the field automatically.

You should now be able to view your newly mapped Display Object by visiting:

```
http://<MobilityServer>:<port>/do/<label for your listing do, e.g.  
MyListing>
```

There are two ways to change the look and feel of your site. The first is MSAdmin. MSAdmin is detailed in the Mobility Server User Guide. Your team will need to extend at least some of the templates to create a basic site structure. Additional design changes can also be made through user templates (Please refer to [Appendix 3](#) for complete folder structure).

MSAdmin

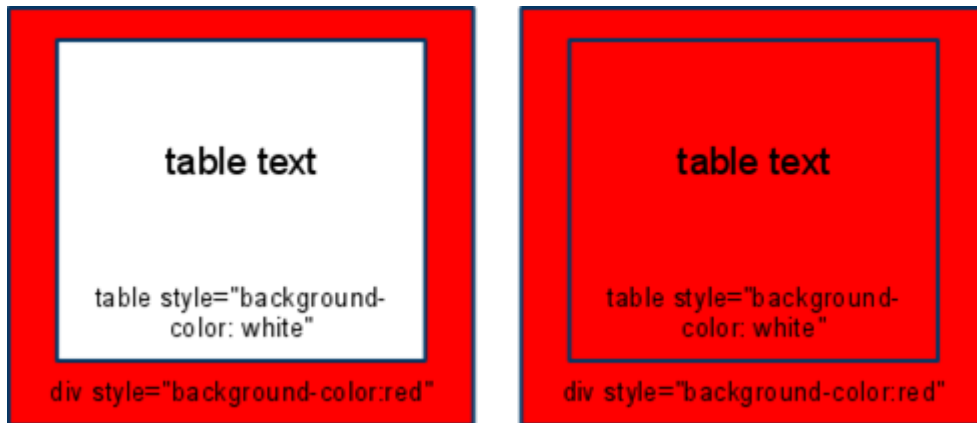
After completing installation, MSAdmin should be available for your desired site upon login. Once inside MSAdmin, your mobile site design can be tweaked to reflect your organization's style guide. As you map new Display Objects, they will appear in MSAdmin. It is strongly recommended that your design team goes through each attribute for each device class prior to requesting any changes to the templates. Please see the Mobility Server User Guide for more information on using MSAdmin.

User templates for site structure

When first setting up your mobile site, you must create and modify existing templates to allow visitors to navigate your site. You may only need to copy existing templates and add links to your main 'index' page, or you may want to create many custom templates. The menu Display Object can also be used for a more dynamic mobile site structure allowing for changes from within WebCenter Sites.

User templates for look and feel

Templates may also be used to tweak the look and feel of your site; however, before you edit a template, it is important to keep in mind that MSAdmin display properties were created to ensure that these templates will work, and even fail, in a predictable fashion, keeping the site legible on thousands of mobile and tablet devices. On the other hand, making even a basic change to a style change to a user template may result in unpredictable behavior or poor rendering on popular devices that you may not have in your organization. If, for example, a designer were to set a 'background-color' CSS property to 'red' on a 'div' element, and include a table within that div element, they may expect it to render like the image below on the left on all devices. It may even appear this way on all devices tested by the designer:



Since background-color is a valid CSS property on table elements, and it appears to work on most phones, a developer may not realize that certain very common handsets lack support for this attribute, resulting in an illegible site such as the one above, on the right. Thus, it is recommended that, whenever feasible, design changes be implemented by changing DO property values via MSAdmin.

Please consider the warnings above, but understand that changes to the templates can be made safely. Any text that is hard-coded should be safe to change. Re-ordering of Display Objects or other elements within the templates is necessary, and is safe and encouraged.

Template Folders and Workflow

The user templates folder exists in the 'templates' folder off of the root of your mobility server installation. Templates reside in two other folders as well: defaults/public and defaults/private. The contents of these folders should never be modified, as any changes to the files in these folders may be overwritten either when Mobility Server is upgraded, or when new Display Objects are installed.

Templates found in the 'defaults/public' folder may be copied to the user templates folder. The templates in the 'defaults/private' folder are not intended to be extended, and changes to those templates or their variants are not supported.

Each of these template folders support the same structure which can be found in the appendix. Each folder contains a folder for each device family, as well as a 'shared' folder that contains fragments of code that may be shared across device families. A Device Family is a set of mobile phones that share similar attributes and features, for example, the touch user interface. The device families are defined as Basic, Smartphone, Touch and Tablet. Aside from changes to the shared folder, most changes to the mobile site should have corresponding changes in each folder.

Create Logos, Headers and Footers

One safe and necessary task to perform to templates is changing the logo file. First, create a 140x30 png logo, similar to the one found in */web/images/logo.png*. Store this file, using your organization's filename in your mobility server's */web/images* folder.

1. First, create copies of the templates files you will need to override. This means creating the following files in your templates folder (assuming the default families):
 - a. `basic/fragments/layout.php`
 - b. `smartphonebb/fragments/layout.php`
 - c. `touch/fragments/layout.php`
 - d. `tablet/fragments/layout.php`
2. For each family, adjust the image path in **layout.php** to reflect your logo image.
3. Restart Mobility Server and preview your changes in MSAdmin, confirming the change for each device family.

Creating a custom mapped Display Object template

Mobility Server allows you to create a new template for each mapped Display Object that you create. First, map a new Display Object. For the sake of simplicity, let's assume that a new 'Index' Display Object has been mapped with a label of 'MyIndex'. Once the mapping exercise is complete, your Display Object will already be visible at `http://<MobilityServer>/do/MyIndex`.

Copy the `defaults/public/_Index.php` template to `templates/basic/MyIndex.php`. Using a text editor, make some basic changes to this file. Repeat this process for `templates/smartphonebb/MyIndex.php`, `templates/touch/MyIndex.php` and `templates/tablet/MyIndex.php`

After restarting mobility server, you should be able to preview your changes at:
`http://<MobilityServer>/do/MyIndex`.

Combining Display Objects

Once you have created a new template for your custom 'MyIndex' Display Object, you can add multiple Display Objects to that template. Refer to the mapping section and map a new Display Object, for example a 'SportsArticleList' Display Object, ensuring that this Display Object can be previewed at `http://<MobilityServer>/do/SportsArticleList`. To add that Display Object to your new MyIndex template, add these lines to each version of your MyIndex.php file:

```
<?php
$h->includeDisplayObject( "SportsArticleList" )
?>
```

Similarly, you may now also map new Display Objects for Business Articles, Top News, etc. and add them to your MyIndex template.

Browsing to `/do/MyIndex` should now display a page that also contains an article listing. More details are available in the Mobility Server Template API Guide.

Additional Fields or Static Content


```
$h->displayObject->getCustomField1()
```

Please see the *Display Object Fields* section of the Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 Template API Guide for more information on the dynamically generated get methods.

This example assumes you are working in a copy of a detail template file in the ***templates/touch*** folder. ***\$h->displayObject*** represents the current article being displayed, and ***getCustomField1*** instructs Mobility Server to look for a field named '***Custom_Field1***'. Additional fields (e.g. Custom_Field2, called by getCustomField2() ...and so on) may be added as well.

Advanced Template Changes

7

The Oracle WebCenter Sites: Mobility Server templates have access to a rich library of features that allow developers to extend or modify the existing templates. Direct access to the underlying device database is also available so that custom functionality can be added. Examples include determining screen size (helpful for making decisions about how much content should be displayed), how to link to a phone number (so that clicking or tapping a phone number initiates the phone's dialer), or accessing thumbnails at a given percentage of screen size. Calls to these functions and many others can be added to the templates where appropriate.

These functions and how they can be used are described in a separate document entitled *Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 Template API Guide*. Please see your account representative for more details.

Publishing

8

Since changes that are made through MSAdmin are stored back to Content Server assets, and all content is stored in WebCenter Sites, changes made by authors or even most design changes will be published. Template changes, however, need to be manually copied to other Mobility Servers. Changes described in Chapter 5, Mapping, should be available on other Mobility Servers in a given environment once mapping changes are stored. Please refer to the Mapping section's "Storing Mapping Changes" subsection.

Please see your account representative for the latest publishing information.

Appendices

Appendix 1: Display Objects

List of Display Objects

Name	Fields	Requires Assets	Links to DOs	Contains Child DOs****
Advertisement	<i>none</i>	No	<i>n/a</i>	No
AppCacheMonitor	<i>none</i>	No	<i>n/a</i>	No
Bookmark	<i>none</i>	No	<i>n/a</i>	No
Carousel	headline, thumbnail (<i>image</i>)	Yes	Yes	No
Collapsible	<i>none</i>	No	No	Yes
Detail	author, body (<i>text</i>), headline, image (<i>image</i>)	Yes	<i>n/a</i>	No
DropDownMenuGroup	<i>none</i>		<i>n/a</i>	Yes
GeoDetail	city, country, description (<i>text</i>), name, state, street, zip	Yes	<i>n/a</i>	No
GeoListing	city, country, description (<i>text</i>), name, state, street, zip	Yes	Yes	No
Index	<i>none</i>	No	<i>n/a</i>	No
ListFilter	headline, description (<i>text</i>)	Yes	Yes	No
Listing	headline, description (<i>text</i>), gridImage***	Yes	Yes	No
Menu	<i>none</i>	Yes	Yes	No
ProductDetail	color, description (<i>text</i>), price, upc, name	Yes	<i>n/a</i>	No
ProductListing	color, description (<i>text</i>), price, upc, name	Yes	Yes	No
SimpleTable	<i>any**</i>	Yes	<i>n/a</i>	No

TabGroup	<i>none</i>	No	<i>n/a</i>	Yes
--------------------------	-------------	----	------------	-----

* non-string data types are shown in parenthesis.

** any and all mapped fields will be displayed automatically

*** tablet family will render listing differently if gridImage is mapped

**** child DOs are specified via the [mapper interface](#)

Advertisement

Advertisement is a Display Object that can be used for displaying advertisements on your mobile pages. This may be used to hold code for calling advertisements from third-party platforms, or it may be used to serve 'house' advertisements. Each mapped advertisement may be enabled or disabled with MSAdmin using an 'ads visible' display property.

It is recommended that a separate mapping be added for each location on a page that an ad is placed, e.g. 'TopAd', 'BottomAd', etc. Having separate mapped advertisements for each placement, as well as for each device class, allows you to customize your ad inventory. Ads that are suitable for the touch family of devices may not be suitable for the basic or smartphone families.

AppCacheMonitor

AppCacheMonitor presents information about the offline status of the mobile site to the end user. Details on its operation can be found in [Appendix 4: Offline Storage Guide](#)

Bookmark

The Bookmark DO can be used to enable end users to place an "app-like" icon on the Home Screen of their mobile device which, when tapped, will open the mobile device's Internet browser and take the user directly to the Customer's mobile website. This DO is currently available for the Touch family of devices. To enable the Bookmark for a mobile site, the following steps need to be followed:

1. Use the Mobility Server Mapping User Interface to enable the Bookmark DO for the mobile site; the only necessary fields are "Display Object Type" and "Mapping Label", both of which should be set to Bookmark. A screenshot is provided below:

Mobility Server Control Panel

Mapping
Mapping Data
Tasks

Adding a New Mapping

Mapping / Add Mapping

CS Site

CS Type Warning: Changing the CS Type will cause all field mappings to be deleted.

Source Type

Display Object Type

Mapping Label Warning: Changing the Mapping Label will necessitate renaming custom templates and references to them.

Source Value

Link Label

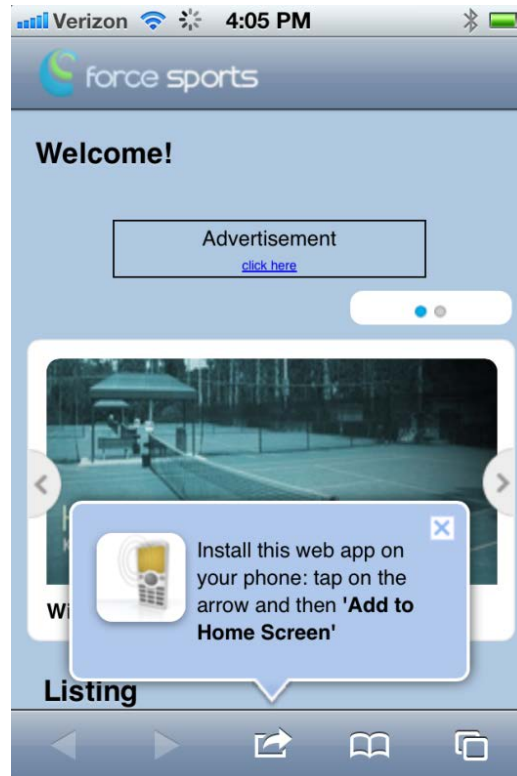
List Item Limit

+ Add Mapping

2. Add the following line to /templates/touch/Index.php:

```
$h->includeDisplayObject( "Bookmark" );
```

Once this step has been completed, a pre-formatted Bookmark "floater" will appear on the mobile website's home page as can be seen in the sample screenshot below:



The default title used by the Bookmark will be the title set at the top of the Home page (/templates/touch/index.php), e.g. `$h->setTitle("Force Sports");`

3. The default home page icon can be modified by creating an icon and saving it in the web/ folder with the following file name: "bookmark-do-icon.png". The recommended HomeScreen icon size is 57x57 pixels.

Carousel

Carousel generates an animated slideshow of photos on supported devices, allowing the visitor to browse through images. It is useful for promotions, featured products, or recent articles. The carousel's animation and control features degrade gracefully for lower-end devices. In the worst case, it will still display at least a static image and/or caption, but for most modern touch devices it will display an animated slideshow that the user can control.

It is recommended that carousel be limited to five images.

MSAdmin contains controls for hiding or displaying captions and other portions of the carousel.

Collapsible

Collapsible generates an accordion-style widget, with collapsible sections which can be expanded and collapsed by clicking on the titles. Collapsible Display Objects do not have fields, but instead have child Display Objects, which are defined in the Mapper, as well as options for each child Display Object such as the title. The most important option for a child Display Object is "Label", which defines the text that will appear above the collapsible section.

For example, let's create a Collapsible Display Object with three locations comprising the sections.

Mobility Server Control Panel

Mapping Mapping Data Tasks

Adding a New Mapping

[Mapping](#) / Add Mapping

CS Site	<input type="text" value="SampleSite"/>
CS Type	<input type="text" value="---"/>
<i>Warning: Changing the CS Type will cause all field mappings to be deleted.</i>	
Source Type	<input type="text" value="---"/>
Display Object Type	<input type="text" value="Collapsible"/>
Mapping Label	<input type="text" value="SomeLocations"/>
<i>Warning: Changing the Mapping Label will necessitate renaming custom templates and references to them.</i>	
Source Value	<input type="text"/>
Link Label	<input type="text"/>
List Item Limit	<input type="text"/>

+ Add Mapping

Note that while creating the new mapping takes you to the Field Mappings editor, no fields need to be created for this Display Object, so we return to the Mapping editor and notice that there is a new section available:

Child Mapping ID	Child DO ID	Child DO Type	Sort order
No data available			
Edit Children			

Clicking on Edit Children allows us to add child Display Objects. Let's assume that we've already mapped a list of locations to a Detail Display Object with the label StoreDetail. We can add three of these, and specify the CID in the Display Object Child Options (which will be covered later in this tutorial.)

[Mapping](#) | [Mapping Data](#) | [Tasks](#)

Editing DO Children for **SomeLocations**

Child display object type:







Child sort order:

[+ Add DO Child](#)

Child Mapping ID	Child DO ID	Child DO Type	Sort order	Actions
22	24	StoreDetail	1	
23	24	StoreDetail	2	

[Edit Mapping](#)

Then, we edit each child Display Object's options by clicking on the edit icon:

Child Mapping ID	Child DO ID	Child DO Type	Sort order	Actions
22	24	StoreDetail	1	 
23	24	StoreDetail	2	 
24	24	StoreDetail	3	 

And add a label for each one:

Editing Options for Child StoreDetail of SomeLocations

Option name

Option value

[+ Add Child Option](#)

ID	Option name	Option value
No data available		

As well as a specific CID for that location. This will ensure that the Collapsible Display Object will include that particular asset.

Editing Options for Child StoreDetail of SomeLocations

Option name

Option value

[+ Add Child Option](#)

ID	Option name	Option value	Actions
26	label	First Location	×

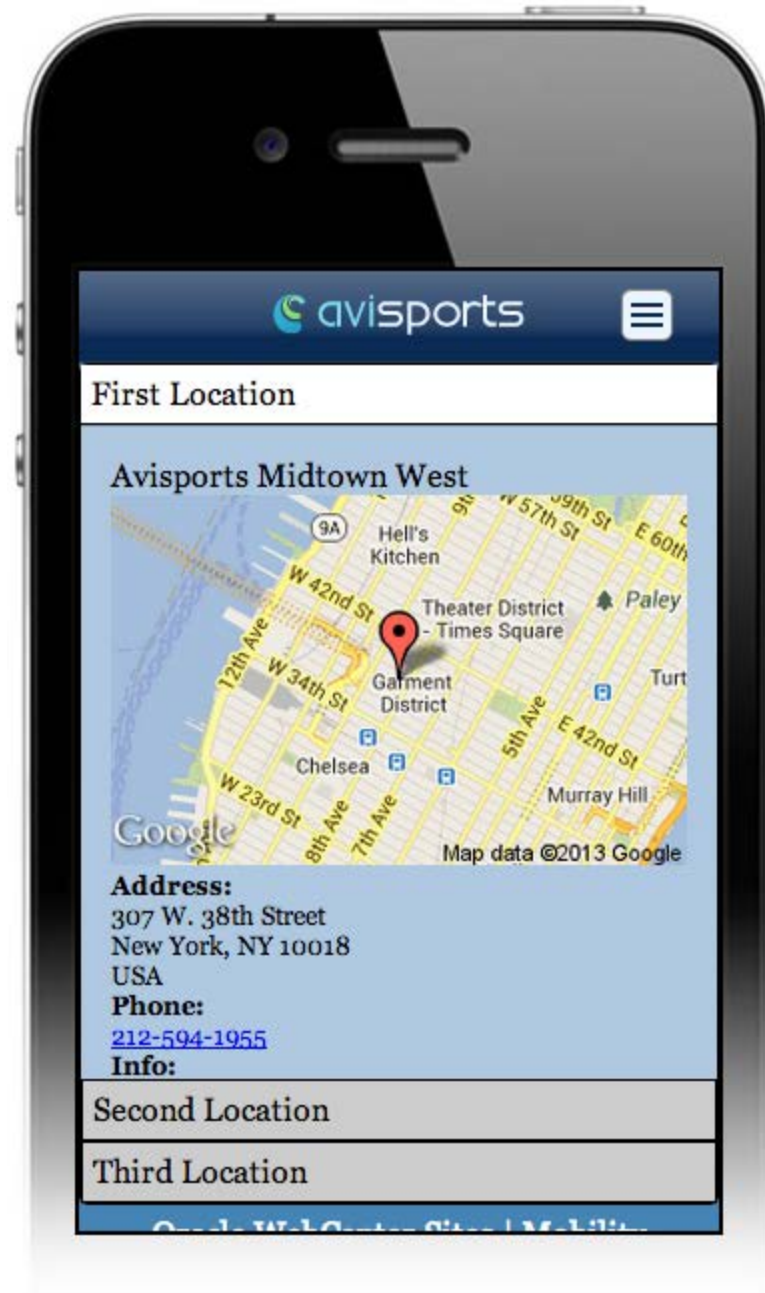
Repeat these steps for the other two children.

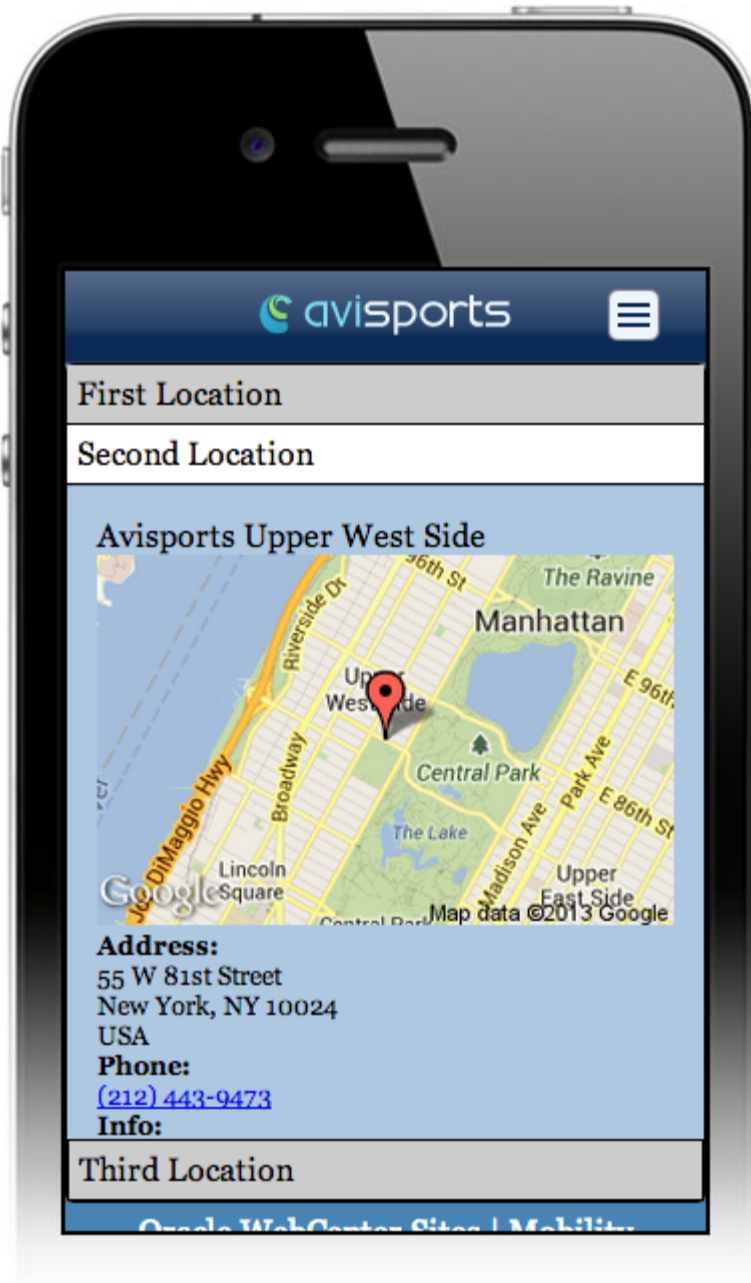
Finally, we include it in the Index template:

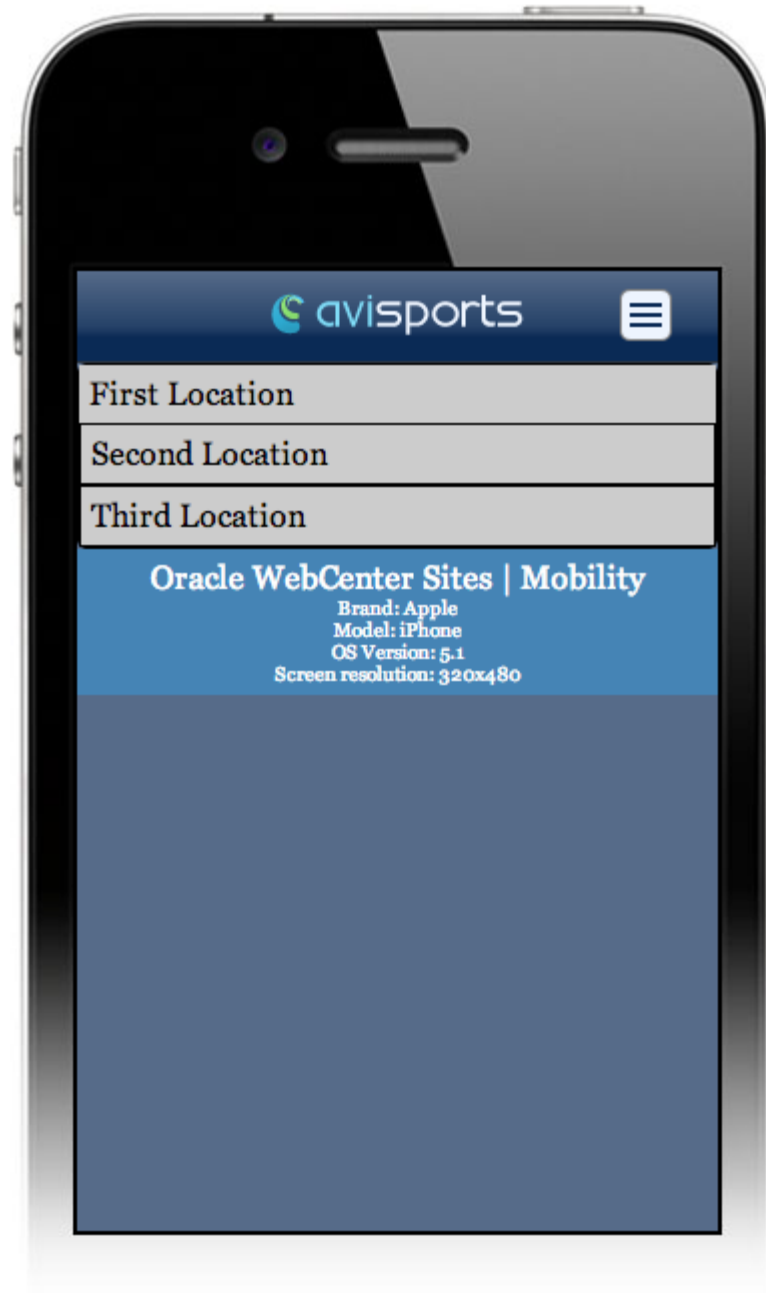
```

<?php
    $h->includeDisplayObject( "SomeLocations" );
?>
```

And see the Collapsible Display Object. Clicking on the titles will expand and collapse the various sections; note, however, that only one section will be displayed at a time.







MSAdmin contains controls for border styling and colors of the Collapsible.

Menu (deprecated)

In the [Discovery](#) section above, the approach that would be taken to organize existing content into a sitemap for the mobile site was determined. The menu DO is one way to implement that sitemap. Two WebCenter Sites templates will need to be created using that approach. These templates will then generate the resulting XML that the Menu Display Object will use. The first WebCenter Sites template to be created should reside by default at `/cs/Satellite?pagename=SEUtils/Mobility/Navigation` and will contain a `site_navigation` element. This URL can be adjusted in:

`mobilityserver/apps/frontend/config/app.yml`. This element contains a single attribute, `lastUpdate`, containing a unix-style timestamp. (Please refer to [Appendix 2: Site Navigation XML Example](#) for full XML code.)

`site_navigation` can contain `navigation` elements. These navigation elements can be nested. These elements look like this:

```
<navigation lastUpdate='1298768235613' cid='1255731768473' name='Home'  
serviceUrl='/cs/Satellite?pagename=SEUtils/Mobility/Home' >
```

- `lastUpdate` is a unix-style timestamp
- `name` is the name of the page or area to navigate to, displayed on the mobile site
- `serviceUrl` should contain a relative URL leading to a `page` XML document, as described in the next section.

The second WebCenter Sites template that should be created will generate `page` XML documents. The `page` xml contains the WebCenter Sites asset ids (`cid`) of the assets to be shown on each mobile page. (Please refer to [Appendix 2: Page XML Example](#) for full XML code.)

The `page` element will contain a `name` attribute with the homepage name to be used in navigation:

```
<page cid='1255731768473' name='Home'>
```

The `page` element contains a list of `asset` elements:

```
<asset cid='1255731772322' type='Article' lastUpdated='2011-02-25 11:49:55.05'/>
```

The `asset` elements contain the following attributes:

- `cid`, the asset cid.
- `type`, the asset type of the asset.
- `lastUpdated`, a timestamp.

Detail

Detail is a Display Object that is recommended for full pages, though multiple detail Display Objects could be used on a single page. The default fields and layout lend themselves well to articles, press releases, or similar asset types. GeoDetail and ProductDetail are variations of this Display Object.

Detail pages by default expect a single image per asset which is mapped as its 'image' field. If additional images are added to the 'description' field, those will appear on your mobile site as well. MSAdmin contains many properties for adjusting how, if at all, these images are displayed.

Detail Display Objects accept a source_type of 'asset', and can therefore be used either to represent a single asset (e.g. 'CompanyContactInformation') or accept an asset's CID as a parameter to render any asset of a given type that has been mapped to it.

DropdownMenuGroup

A DropdownMenuGroup is a Display Object specifically meant to function as a 'drop-down' or 'pop-out' interface. It does not contain any fields, but rather contains Child Display Objects in a manner similar to Collapsible or TabGroup. When it is included, a clickable button will appear - clicking it will pop out a container with the included Child Display Objects; clicking again outside of the container will dismiss it.

Note that when including a DropdownMenuGroup Display Object, you must also place a div with an ID that has the same name as the label with a "-target" attached - this lets Mobility Server know where to place the visible display object.

An example for the Touch interface is below; note that on the tablet interface, the div ID ought to be "ms-tablet-PopupMenu-target", and that if you change the mapping label to something else, the div ID will change as well.

```
<div style="width: 20%; padding-top: 3px;">
  <?php $h->includeDisplayObject("PopupMenu");?>
</div>
```

```
<div id="ms-touch-PopupMenu-target"></div>
```

Index

Index can be thought of as a container for other Display Object, though it is a Display Object itself. Asset types cannot be mapped to it. By default, Mobility Server ships with a single mapped Index Display Object in the user templates folder. Index may also be useful to create pages with content not available via WebCenter Sites.

Listing

Listing accepts the 'assetlist', 'rssitemlist', 'atometrylist' or 'xml' source types, allowing it to display anything from a short, static list of items to a very large list of items, so it is suitable both for short menus, as well as for long directory-style listings. It also uses the 'link_label' field to allow it to create links to other Display Objects. For example, 'MyListing' may be created with a 'link_label' of 'MyDetail'. If so, Mobility Server will attempt to create a link for each row in the listing to a 'MyDetail' Display Object with the same CID as a parameter.

In the tablet family, the listing will be rendered differently depending on how it has been mapped. If a 'gridImage' field is mapped, the listing will appear as a grid of approximately 3x3 images in portrait mode, or 4x2 in landscape mode. The 'gridImage' image will be used as the image for each item in the grid. If 'gridImage' is not mapped, the listing will appear as a single column list similar to the way it is rendered in other families.

ListFilter

ListFilter offers the same functionality as Listing with the added feature of a text input field that, as a user types in it, filters down the list to only contain items with "Headline" fields matching the entered text. This filtering is done completely client-side with javascript, so it happens instantaneously as the user types. It's recommended you use a ListFilter Display Object when you expect that users will be

looking for a specific item (or group of items) in a listing, rather than browsing from item to item. ListFilter is supported on touch and tablet devices.

On devices in the Basic and Smart family which do not support Javascript, ListFilter acts as a normal Listing.

GeoDetail and GeoListing

GeoDetail and GeoListing are used to create location-based services. When all of the default fields are mapped, Mobility Server will automatically geocode addresses, saving a longitude and latitude value for each corresponding WebCenter Sites asset. This data can then be used by either (or both) GeoListing and GeoDetail to display maps optimized for a visitor's handset. Further, GeoListing allows for most devices in the touch family to share their location with Mobility Server, allowing Mobility Server to present a list of locations to the visitor, sorted by proximity. In the tablet family, the GeoListing and GeoDetail are used together to show an interactive map along with the list of locations.

ProductDetail and ProductListing

ProductDetail and ProductListing can be used together to create catalogs of products. In addition to the features already present and described in the listing and detail Display Object, ProductDetail and ProductListing allow for prices, SKUs, and other information to be presented to visitors.

SimpleTable

SimpleTable can be used to display a tabular listing of data. SimpleTable is unique in that it does not have any default fields. Instead, you may map anywhere from two to ten fields of a given asset type to a SimpleTable Display Object. The 'field_name' values, may be created with any string, allowing you to dynamically column labels for each column in a simple table. The data for each column will then be taken from 'source_field_name'.

For example, to display a SimpleTable of articles, we would first map a listing of Articles to the SimpleTable Display Object:

CS Site

CS Type Warning: Changing the CS Type will cause all field mappings to be deleted.

Source Type

Display Object Type





Mapping Label Warning: Changing the Mapping Label will necessitate renaming custom templates and references to them.

Source Value

Link Label

List Item Limit

Then we map three fields as our columns. Note that the order of the columns depends on the order in which they were mapped as fields - the ID in the field mapping indicates this ordering.

ID	CS Field Name	CS Field Type	Field Mapping Label	
206	author	string	Author	 
208	abstract	string	Description	 
207	headline	string	Headline	 

And include the table in our Index template:

```
<?php
$h->includeDisplayObject("AllArticlesTable")
?>
```

TabGroup

The TabGroup Display Object includes multiple Display Objects, allowing the user to access each one by a series of tabs running along the top. A TabGroup may contain an arbitrary number of child Display Objects, and will allow the user to drag the tabs from side to side to access those out of view. Clicking on a tab shows the appropriate Display Object underneath the tabs.

For instructions on how to map child objects to a TabGroup, see the section on the [Collapsible](#) Display Object.

Appendix 2: XML Examples

Site Navigation XML Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<site_navigation lastUpdate="1313514053774">
  <navigation type="Page" lastUpdate="1313514053774" cid="1255731768473" name="Home"
serviceUrl="/cs/Satellite?pagename=SEUtils/Mobility/Home">
  <navigation type="Page" lastUpdate="1313514053774" cid="1255731768476" name="Features"
serviceUrl="/cs/Satellite?pagename=SEUtils/Mobility/Features"/>
  <navigation type="Page" lastUpdate="1313514053774" cid="1255731768484" name="TopNews"
serviceUrl="/cs/Satellite?pagename=SEUtils/Mobility/TopNews"/>
</navigation>
</site_navigation>
```

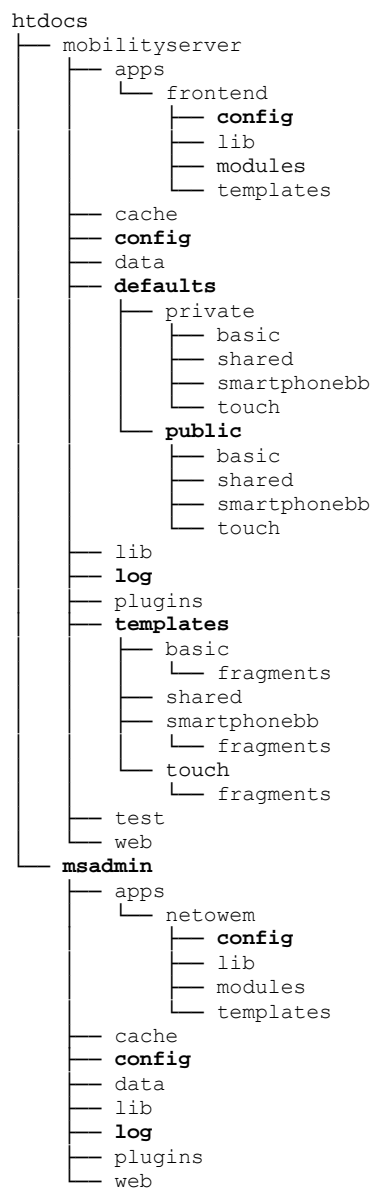
Page XML Example

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<page cid='1255731768476' name='Features'>
  <asset cid='1255731772030' type='Articles' lastUpdated='2010-09-16 16:11:21.0' mobileenabled='true' />
  <asset cid='1255731807156' type='Articles' lastUpdated='2011-08-02 17:35:26.066' mobileenabled='true' />
  <asset cid='1255731807104' type='Articles' lastUpdated='2010-09-22 07:57:10.812' mobileenabled='true' />
</page>
```

Appendix 3: Folder Structure

Default Mobility Server Folder Structure after Automated Install

Folders in **bold** contain files that may be referred to in this document, or you may wish to review.



Appendix 4: Offline Storage Guide

Introduction

HTML5 has introduced a mechanism called the **ApplicationCache**. The ApplicationCache allows a site to specify to a browser which resources of the site should be cached and made available while browsing offline. This is done via a cache manifest file, which contains a list of resources (absolute URLs or relative paths) for the browser to cache. The cache manifest file is referenced by a 'manifest' attribute on the html element; when this attribute is present, HTML5-compatible browsers will download the manifest file and begin the process of caching every resource referenced by it.

The ApplicationCache is especially beneficial on mobile platforms because it offers some relief for slow and/or unstable connections. As such, Mobility Server 11.1.1.8.0 offers built-in support for enabling ApplicationCache usage on your mobile site. By having a user's mobile device cache your site's resources, you ensure an overall smoother browsing experience. The ApplicationCache greatly helps to bridge the gap between native and web-based mobile apps.

Configuration

To enable support for the ApplicationCache in Mobility Server, first set the use_appcache setting in app.yml to "true".

The following Apache directives must also be added to Apache's configuration:

```
AddType text/cache-manifest .appcache
ExpiresByType text/cache-manifest "access plus 0 seconds"
```

Note that these are included in the web/.htaccess file that ships with Mobility Server, so this may not be necessary.

Creating the Cache Manifest

Once ApplicationCache support has been enabled and Apache has been configured, it is now time to decide what to cache and create the manifest file. The new ***mobilityserver:updateAppCache*** command-line task is provided to do this. This task will generate cache manifest(s) to your specifications based on the content displayed via Mobility Server. One manifest will be generated for each device family you specify.

Notifying the User of Caching

After configuration is complete and the manifest has been generated, everything is setup so that HTML5-enabled browsers will now cache your site. Although not necessary, it is recommended that the user is informed that caching is taking place. For this purpose, a new Display Object called ***AppCacheMonitor*** has been provided. When included on a page, this Display Object will hook into the browser's ApplicationCache interface in order to relay information about the caching process to the user.

Publishing and Delivery vs. Editorial Environments

One problem with ApplicationCache is that when new content is published to a page, this content will not appear to users if they have previously cached that page. To overcome this, the cache manifest should be regenerated when Publishing occurs, using the same options to generate it as you specified to ***mobilityserver:updateAppCache***.

As with most caching options, it is recommended that the ApplicationCache only be enabled in Delivery environments.

Final Steps

As a last step, test to ensure that caching works as you expected. Most devices feature an 'airplane mode' feature that disables all networking features, allowing you to test how the device will work in offline mode.

The JavaScript console in the Google Chrome browser is very useful for seeing what the `ApplicationCache` is doing behind the scenes. Understanding how `ApplicationCache` and the cache manifest are intended to work is critical for troubleshooting. The following resources can be helpful:

"WHATWG: HTML Living Standard"

<http://www.whatwg.org/specs/web-apps/current-work/#applicationcache>

"A Beginner's Guide to Using the Application Cache"

<http://www.html5rocks.com/en/tutorials/appcache/beginner/>

The HTML5 standard - Section 5.6: "Offline Web Applications"

<http://www.w3.org/TR/html5/offline.html>

The `mobilityserver:updateAppCache` task

The `updateAppCache` task will generate (or regenerate) a cache manifest file based on content displayed by Mobility Server. This task will always update the revision line of the manifest file, instructing the browser to retrieve new content.

Usage

```
symfony mobilityserver:updateAppCache [--explain] [--crawl="..."] [--family="..."] [--bump] [--nocache] [--noassets] [--noimages] [--images="..."] [--exclude="..."] [--include="..."]
```

<code>--explain</code>	Explain why certain assets are or are not being cached.
------------------------	---

--crawl	Specifies the URL at which Mobility Server is serving content; required.
--family	Specify a list of device widths for which to generate content; by default this task will generate a cache manifest for 320, 360, 384 and 768-pixel wide devices, which cover most of the devices which will be able to access the touch and tablet templates.
--bump	Only update the revision date in the Cache Manifest but do not regenerate its content. This forces clients to re-cache pages without changing which URLs and paths are cached.
--nocache	Do not cache anything at all. This option overrides any others. This is useful for generating an empty manifest for when you want to disable caching on a site that previously had it enabled.
--noassets	Only cache static things like stylesheets and Javascript.
--noimages	Do not cache any images. This overrides the --images option.
--exclude	Specify any labels to prevent them, and their associated content, from being cached. Note that the --include option overrides this. (multiple values allowed).

Examples

Disable caching altogether:

```
php symfony mobilityserver:updateAppCache --nocache
```

Cache only Javascript and CSS files necessary for the site to run, without caching any Display Objects:

```
php symfony mobilityserver:updateAppCache --crawl=http://mobile.example.com/
--noassets
```

Create cache manifest, crawling the site located at http://mobile.example.com/ first:

```
php symfony mobilityserver:updateAppCache --crawl=http://mobile.example.com/
```

Cache only ArticleDetail and ArticleList Display Objects, without their images:

```
php symfony mobilityserver:updateAppCache --crawl=http://mobile.example.com/
--noimages --include=ArticleDetail --include=ArticleList
```

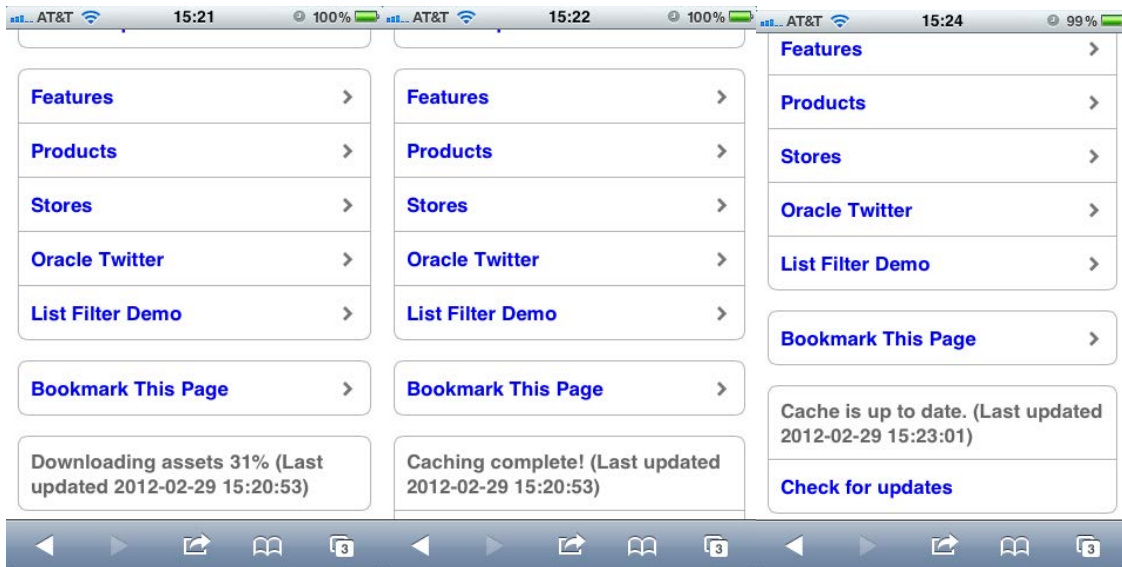
This will generate cache manifest files for the touch families only.

```
php symfony mobilityserver:updateAppCache --crawl=http://mobile.example.com/  
--family=320 --family=360 --family=384
```

The AppCacheMonitor Display Object

When included on a page, this Display Object will hook into the browser's ApplicationCache interface in order to relay information about the caching process to the user.

Note that this Display Object is not required to enable AppCache support on your site. Its use is recommended in order to inform the user of the work their browser is performing behind the scenes.



Appendix 5: RSS, Atom and other XML Feeds

Overview

RSS and Atom feeds can be mapped to Display Objects. To accomplish this, the following source types should be used:

- `rssitemlist`
- `rssitem`
- `atomitemlist`
- `atomitem`

The ***rssitemlist*** and ***atomitemlist*** source types are intended for use with the *Listing* and *ProductListing* Display Objects. The ***rssitem*** and ***atomitem*** source types are intended for use with *Detail* and *ProductDetail* Display Objects. In each of these cases, the source value must be set to a full RSS (for `rssitemlist` and `rssitem`) or Atom feed (for ***atomitemlist*** and ***atomitem***). If the same source value is used, a link label can be used to link a Listing or ProductListing Display Object to a Detail or ProductDetail Display Object.

An XML source type is also available for any other XML sources that you may want to map.

Field Mappings

When mapping a Display Object to an RSS or Atom source type, the field names take on a new meaning. Instead of expecting a CS asset type attribute name, they will expect an XPath statement. The starting point for these XPath expressions is the ***item*** element for RSS, and ***entry*** for Atom. This allows for a simple shorthand to be used, not requiring any XPath knowledge in most cases.

The following example demonstrates mapping standard sub-elements of an RSS item element to a Listing DO:

ID	CS Field Name	CS Field Type	Field Mapping Label	
66	author	string	Author	<input type="checkbox"/> <input type="checkbox"/>
68	pubDate	date	DatePosted	<input type="checkbox"/> <input type="checkbox"/>
65	description	string	Description	<input type="checkbox"/> <input type="checkbox"/>
64	link	string	Link	<input type="checkbox"/> <input type="checkbox"/>
67	title	string	Text	<input type="checkbox"/> <input type="checkbox"/>

Sample RSS Feed

The following is a very basic example of an RSS feed from <http://en.wikipedia.org/wiki/RSS>

```
<?xml version="1.0" encoding="UTF-8" ?>
<rss version="2.0">
<channel>
  <title>RSS Title</title>
  <description>This is an example of an RSS feed</description>
  <link>http://www.someexemplerssdomain.com/main.html</link>
  <lastBuildDate>Mon, 06 Sep 2010 00:01:00 +0000 </lastBuildDate>
  <pubDate>Mon, 06 Sep 2009 16:45:00 +0000 </pubDate>
  <ttl>1800</ttl>

  <item>
    <title>Example entry</title>
    <description>Here is some text containing an interesting
description.</description>
    <link>http://www.wikipedia.org/</link>
    <guid>unique string per item</guid>
    <pubDate>Mon, 06 Sep 2009 16:45:00 +0000 </pubDate>
  </item>

</channel>
</rss>
```

Advanced Mappings

Display Objects mapped using an RSS or Atom source type accept XPath statements as CS Field Name data. As described previously, by default the XPath function will begin within 'item' elements -- a single item element for rssitem or a list of items for rssitemlist -- so a CS Field Name of 'link' is equivalent to '//rss/channel/item/link' or './item/link'.

The XPath functionality can be useful for feeds that have additional XML elements, or for including data from the root element. For example, a Detail Display Object named 'MyRSSItemDetail' could have a custom field mapped named 'feed_title' which takes a CS Field Name value of './title'. Each resulting MyRSSItemDetail instance would populate 'feed_title' with the channel title.

Display Objects mapped to the XML source type behave similarly, but XPath functions always begin processing from the document root instead of taking shortcuts to individual RSS or Atom items as described above.

OTHER REFERENCE DOCUMENTS

[Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 Installation Guide](#)

[Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 User Guide](#)

[Oracle WebCenter Sites: Mobility Server Version 11.1.1.8.0 Template API Guide](#)