



Volantis Mobility Server[™]

ver. 5.1

Getting Started

Volantis Mobility Server™

Version 5.1

Getting Started

Volantis Mobility Server: Getting Started

Published date 2009-01-29

© Copyright 2001-2009 Volantis Systems Ltd. All rights reserved.

This document, as well as the software described in it, is provided under license and may be used or copied only in accordance with the terms of such license. The content of this manual is provided for information only, is subject to change without notice, and should not be construed as a commitment by Volantis Systems Limited. Volantis Systems Limited assumes no responsibility for errors, inaccuracies or omissions, or for damages resulting from the use of information contained therein. This document includes confidential information which may be kept confidential and used strictly only in accordance with the terms of such license.

Except as permitted by such license, no part of this publication may be reproduced, used, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Volantis Systems Limited.

Volantis, Volantis Systems Limited, the Volantis logo, Intelligent Content Adaptation, Intelligent Content Delivery, Volantis Mobility Server, Volantis Suite of Mobile Content Applications, Volantis Mobile Content Storefront, Volantis Mobile Content Transcoder, Volantis Mobile Content HomeDeck, Volantis Mobile Content Broker, Media Access Proxy, Volantis Mobilizer, XDIME, Multi-Channel Server, Distributed Service Broker, Message Preparation Server, all trademarks and logos that contain Volantis and XDIME and certain other trademarks and logos appearing in this documentation, are trademarks or registered trademarks of Volantis Systems Ltd. in the United Kingdom and other countries.

All other company names or products are trademarks or registered trademarks of their respective owners.

Included software

Apache Software Foundation

Free Software Foundation

Jaxen Project

Sun Microsystems Inc

Table of contents

Getting started	4
Policies	4
XDIME	4
Installation	5
Required software	5
Building Volantis Mobility Server	5
Running the installer	5
Copy the Eclipse plugins	6
Setting up Tomcat	6
Mobile preview	8
Installing Firefox and User Agent Switcher	8
Emulating mobile devices	8
Changing user agents	10
Creating a project	11
Creating a project	11
Adding an image component	12
Adding a component	12
Adding a variant	12
Choosing a device	13
Adding the Smartphone variant	14
Creating a layout	16
Adding a layout	16
Choosing a device	16
Using grids	17
Adding a format	18
Mobile layout	19
Checklist	19
Adding a theme	21
Adding a theme	21
Adding a selector	21
Setting style properties	22
Style Rules view	22
Checklist	23
Welcome page	25
Declaring XML namespaces	25
Binding to the layout and theme	25
Body text	25
Checklist	26
Form content	27
Adding form elements	27
Checklist	27
Complete XDIME	29
Response JSP	31
JSP content	31
Summary	32
Remote projects - PHP and Ruby	33
Render from PHP	33
Configure MCS	33
Render from Ruby	33
Migrating to XDIME	35
XSL preliminaries	36
Body content	36
Checklist	38
XSL transform	39

Getting started

Welcome to Volantis Mobility Server 5.1™.

Getting started is a beginner's guide, which will help you get rolling with your first mobile site. It covers the install and deploy process for a simple development system, and describes a welcome page that you can run 'out of the box' and access from both PC systems and mobile devices. We show how this page is constructed using XDIME 2 markup, and how it works with the device independent policy information to render your content for a range of devices.

When you have finished, we suggest that you follow the tutorial *Next steps with MCS*. It is based on a more complex site, with its own set of resources. It goes into more detail about the creation of projects, components, layouts and themes. It also takes you through the construction of a wider range XDIME 2 pages with more code examples.

Each tutorial module describes one main task. In each module step-by-step sections guide you through the process till the task is completed. All the materials used in this guide and the tutorials are shipped and installed with MCS.

Policies

Volantis Mobility Server adapts page content to multiple device characteristics. The adaptation is driven by policies, and a large database of device attributes.

Policies describe the individual pieces of information that MCS needs to render pages that match the devices used by a range of users. By applying different policies, you can distinguish between 'device-specific' and 'device-independent' information. This policy based separation enables quicker, easier maintenance as well as rapid implementation.

The first part of *Getting started* explains the process of creating a project, an image component, a layout, and theme for the welcome page. You will use the Eclipse IDE to edit policies.

XDIME

The second part of this guide introduces XDIME markup (XHTML Device Independent Markup Extensions). XDIME 2 is a standards based XML vocabulary, using familiar elements from XHTML and XForms. This makes it very easy to create XDIME content if you already know these languages.

We assume that you have a working knowledge of XHTML 1. Where XDIME differs from XHTML, we have highlighted it in the topic *Migrating to XDIME*. Usually this is because XDIME adopts some conventions from the emerging XHTML 2 standard.

XForms is a relatively new W3C recommendation. We touch on forms only briefly in this tutorial, illustrating a way to respond to form input with a JSP. We encourage you to move on to study the forms topics in the XDIME documentation. Refer to the *Working with XForms* tutorial for further information.

We also show how to configure MCS so that you can process content from a remote site that renders XDIME from a PHP or Ruby framework, and demonstrate how to 'print' a sample 'Hello' page in both languages.

Installation

Required software

There are several software components that you will need. Volantis Mobility Server requires a web application server, JDK 1.5 or 1.6, and the Eclipse IDE is needed for policy development. See *MCS Supported platforms* for further information. You will also find an XML editor useful for creating XDIME code.

If you plan to download the source code, you will also need Maven to build the Community edition.

1. Download and install a copy of the Apache **Tomcat** web application server, if you don't already have it. The latest, stable release should be fine. The instructions in this guide are based on Tomcat Version 5, which you can find at <http://tomcat.apache.org/>
2. Download and install the Eclipse IDE, if you don't already have it. Version 3.2.2 is a good choice. Later versions are not compatible with Volantis Mobility Server. You can find downloads of several versions of Eclipse at <http://archive.eclipse.org/eclipse/downloads/index.php>

Building Volantis Mobility Server

To build the Community edition you will need to download five distinct `.tar` or `.zip` files (both are available). Because of dependencies, you should always build them in the following sequence in cases where you wish to build Volantis Mobility Server in its entirety.

This is because the build process will initially retrieve the required `.jar` files from a Volantis server. Once any component is built the JARs will be located in your local Maven repository. If you make changes to the source you may not want the original JARS to be reused in all cases.

- Synergetics - Synergetics-5.1-ce-src.tgz/zip
 - Device Repository - DeviceRepository-5.1-ce-src.tgz/zip
 - Media Access Proxy - MAP-5.1-ce-src.tgz/zip
 - Multi-Channel Server - MCS-5.1-ce-src.tgz/zip
 - Message Preparation Server - MPS-5.1-ce-src.tgz/zip
1. If you have not done so already, download the Community edition sources from the following URL and unzip them to a directory
<http://www.volantis.com/sources-download>
 2. Download Maven and install it following the instructions on the Maven site. Version 2.0.8, which is in use at Volantis, can be obtained at <http://maven.apache.org/download.html>
 3. Ensure that MAVEN_HOME is set to the Maven download location, and JAVA_HOME is set to your JDK installation directory
 4. On Unix
`export MAVEN_OPTS="-Xmx256M -XX:MaxPermSize=96M"`
 5. Or on Windows
`set MAVEN_OPTS=Xmx256M -XX:MaxPermSize=96M`
 6. Change directory to the MCS source code root directory, and unset your CLASSPATH variable if it's set
 7. Ensure that version 2.0.8 of Maven is used. On Unix add \$MAVEN_HOME/bin to the beginning of your PATH variable. On Windows add %MAVEN_HOME%\bin to the PATH variable
 8. If you are not on Linux, build the source with the command
`mvn -DskipTests install`
 9. To generate the Javadoc (MCS only), change directory to `javadoc` and issue the command
`mvn javadoc:javadoc`
 10. Optionally build the installers (MCS, MAP and MPS) by changing directory to `installer/api` (MCS and MPS) or just `installer` (for MAP) and issuing the command
`mvn izpack:izpack`

Running the installer

Your project information and the device database can be deployed either as XML files or in a JDBC repository. We recommend that you configure an XML repository initially, and always use it for policy development. When you move to a production system, you can make any necessary configuration changes for a suitable environment, and import the policies and device data to a JDBC based service. Refer to the administration topics *Setting up the MCS runtime*, *Configuring a JNDI datasource* and *Importing and exporting repositories* for more information.

To follow this guide you need only install MCS. Later, if you want to use media transcoding features, you will also need to install Media Access Proxy (MAP). See *Installing MAP* in Eclipse Help or the *MAP Installation*

and *Administration Guide* for details. When you install MCS, you can enter the MAP host and port settings if you wish.

1. Download the binary installer from the community edition web site (<http://www.volantis.com/downloads>), or use the one you have built. A wizard guides you through the installation process.
2. Start the installer with the command
`java -jar mcs_installer-5.1.jar`
3. Accept the license agreement
4. Accept the default pack selection
5. Select 'XML' as a value in the dropdown for *Initial repository vendor*
6. On the application server page, select Apache Tomcat from the drop down list. Leave the *Host* as localhost, but change the port number if you run Tomcat on a different port.
7. Accept the default Media Access Proxy *URL*
8. Enter the file location to install MCS or accept the default
9. The installer displays a summary of some installation choices. Use the **Previous** button to go back and change any options. When you are satisfied with all your choices, click **Next** to start the installation.
10. Optionally save your installation settings by clicking **Generate an automatic installation script**, and naming the file. To repeat the installation run the following command
`java -jar mcs_installer-5.1.jar [configuration_file]`
11. Click **Done** to complete the installation

Copy the Eclipse plugins

Volantis Mobility Server includes a number of Eclipse plugins for tools and documentation. These need to be copied from the installation to the Eclipse environment as follows.

1. Copy the contents of the `/MCS/eclipse/plugins` directory, including any sub-directories, to the Eclipse `plugins` directory location
2. Also copy the contents of `/MCS/eclipse/features/` to the `features` directory
3. An alternative way to install the plugins to Eclipse is to use Eclipse manager:
 1. Open Eclipse
 2. Go to **Help | Software Updates | Find and Install** to open the wizard window
 3. Select **Search for new features to install**
 4. Press the **New Local Site** button and select the `/MCS/eclipse` directory
 5. Press the **Finish** button
 6. Accept the terms in the license agreement
 7. Press **Next**, and then **Finish** to complete

Setting up Tomcat

Below the `webapps` directory of your installation, you'll find a directory called `mcs`. When you have completed this task, you can check that everything has installed correctly by displaying the welcome page.

1. Copy this directory and its contents to the `webapps` directory in your Tomcat installation.
2. Start Tomcat and point your PC browser at
`http://localhost:8080/mcs/projects/welcome/simple_welcome.xdime`
You should see the following page.



Useful links

[Volantis home page](#)
[Support gateway](#)
[Community](#)

Congratulations!

You have successfully installed and configured Volantis Mobility Server Community Edition™.

Volantis Mobility Server

Volantis Mobility Server provides a comprehensive, standards-based, development and execution environment for delivering multi-channel content, applications and services that are automatically optimized for thousands of different consumer devices.

Built around the principle of device-independent delivery, Volantis Mobility Server uniquely separates content, design, and device-issues into abstract device-independent policies. This enables a 'create once, run anywhere' environment which reduces complexity, cost and time to market for both development and maintenance.

System components

Have a look at the key components in Volantis Mobility Server.

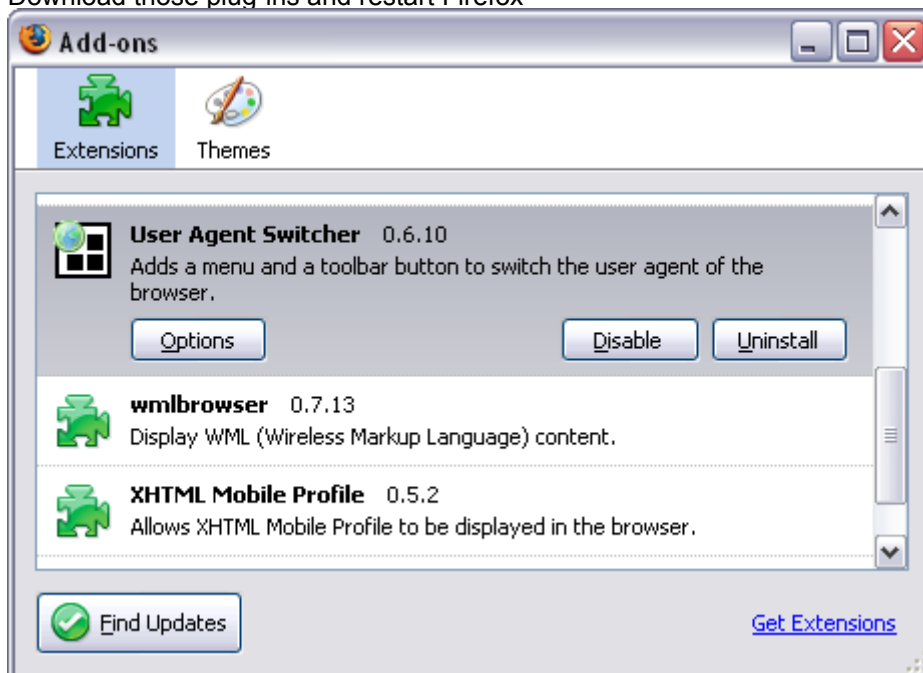
*Copyright © 2000-2008 Volantis Systems Ltd. All Rights Reserved.
Volantis™ is a trademark of Volantis Systems Ltd.*

Mobile preview

The Firefox web browser allows you emulate different mobile devices, which may be helpful while you develop mobile applications. To preview the emulated version of your web application you will also need to install the User Agent plugin.

Installing Firefox and User Agent Switcher

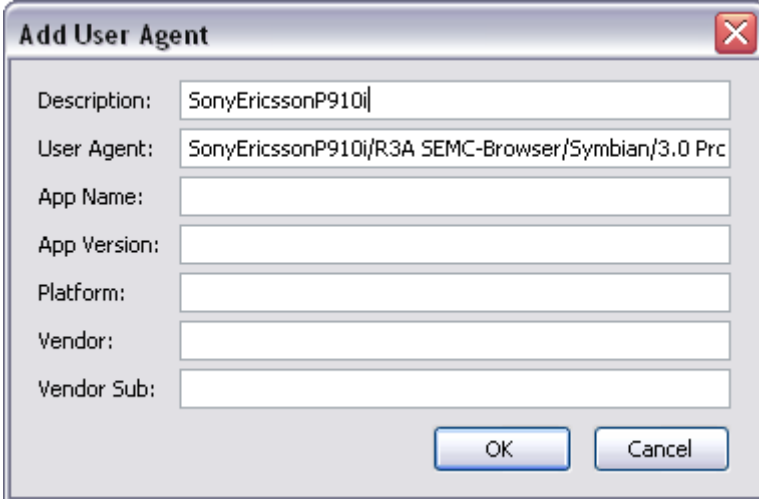
1. Install the Firefox, referring to the Firefox documentation for details
2. Start Firefox
3. Choose **Tools** | **Add-ons** from the Firefox menu to open the Add-ons dialog
4. Click the **Get Extensions** link. Firefox will open the web service that allows you to download and install extensions.
5. On the list of available extensions find the 'User Agent Switcher' plug-in
6. To avoid issues previewing the mobile site, you should install also the 'XHTML Mobile Profile' and 'wmlbrowser' plug-ins
7. Download those plug-ins and restart Firefox



Emulating mobile devices

Now you can emulate Smartphone devices.

1. Choose the **Tools** | **User Agent Switcher** | **Options** | **Options...** from the Firefox menu
2. Choose the *User Agents* option
3. Click the **Add** button to add a new device
4. Type the name of device in the *Description* text box; in our case 'SonyEricssonP910i'
5. Enter the User Agent string for the Smartphone devices; 'SonyEricssonP910i/R3A SEMC-Browser/Symbian/3.0 Profile/MIDP-2.0 Configuration/CLDC-1.0'

A screenshot of a Windows-style dialog box titled "Add User Agent". The dialog has a close button (X) in the top right corner. It contains several text input fields: "Description:" with the text "SonyEricssonP910i", "User Agent:" with the text "SonyEricssonP910i/R3A SEMC-Browser/Symbian/3.0 Prc", "App Name:", "App Version:", "Platform:", "Vendor:", and "Vendor Sub:". At the bottom right, there are two buttons: "OK" and "Cancel".

Add User Agent

Description: SonyEricssonP910i

User Agent: SonyEricssonP910i/R3A SEMC-Browser/Symbian/3.0 Prc

App Name:

App Version:

Platform:

Vendor:

Vendor Sub:

OK Cancel

6. Click the **OK** button
7. The 'SonyEricssonP910i' appears on the list of available devices
8. Switch the user agent to that device by picking it from the **Tools | User Agent Switcher | SonyEricssonP910i** menu
9. Open the MCS web application to preview the emulated version of the page



- To return to the normal preview choose the **Tools | User Agent Switcher | Default** from the Firefox menu

Changing user agents

Switching user agents does *not* invalidate pages that Firefox has already cached. A consequence is that if you visit a page, change the user agent and visit the page again, it's very likely that you will simply see the previously loaded version of the page - even if you force a reload. You need to clear the cache to view the page using a different user agent. Normally, you also need to restart Firefox.

A good way to ensure that you see the correct version of a site is to use the following sequence.

1. Start Firefox
2. Clear the cache by choosing **Tools | Clear Private Data** and ensuring that the *Cache* is checked. Make sure that you are not going to clear other data that you wish to keep, and then click **Clear Private Data Now**.
3. Select the user agent
4. Access the pages you want to view

Creating a project

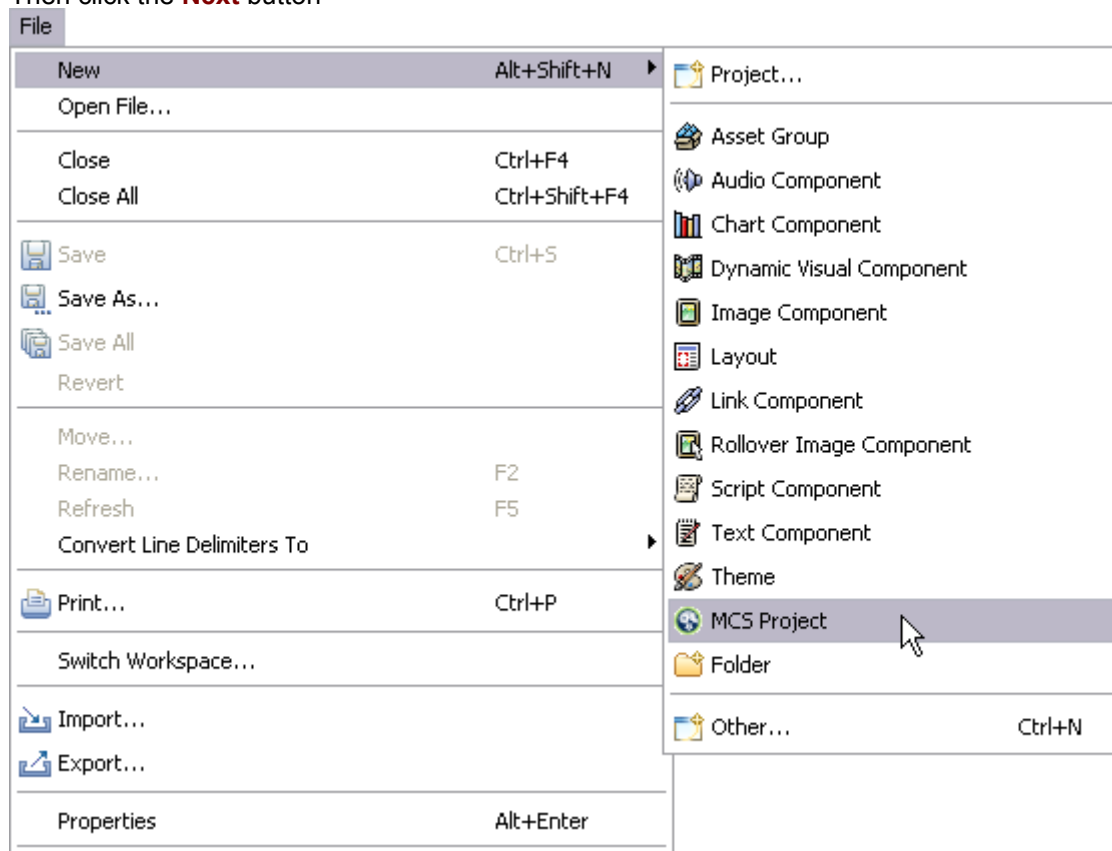
Here are the steps to create the welcome project in Eclipse.

Usually you will create the policy directories in Eclipse. However, this time you will start with the project resources and XDIME 2 files that come with Volantis Mobility Server.

By default the policies are stored in the `WebContent/mcs-policies` directory created by the *MCS Project* wizard. Our layouts and theme policies are placed in that location. An additional folder, stores all image component policies.

Creating a project

1. Choose the **File | New | Project**
2. Pick **MCS Project** from the list of available wizards. If you cannot see this entry in the menu, navigate instead to **Other** and then **MCS** and then **MCS Project**.
3. Then click the **Next** button



4. Name the project 'welcome' and uncheck the *Use default location* option
5. **Browse** to `webapps/mcs/projects/welcome` in your Tomcat installation
6. Click the **Next** button
7. Now you need to provide the location of the device repository. Click the **Browse** button and navigate to the repository file in your installation `MCS/repository/devicerepository/devices.mdpr`.
8. Click the **Finish** button to complete

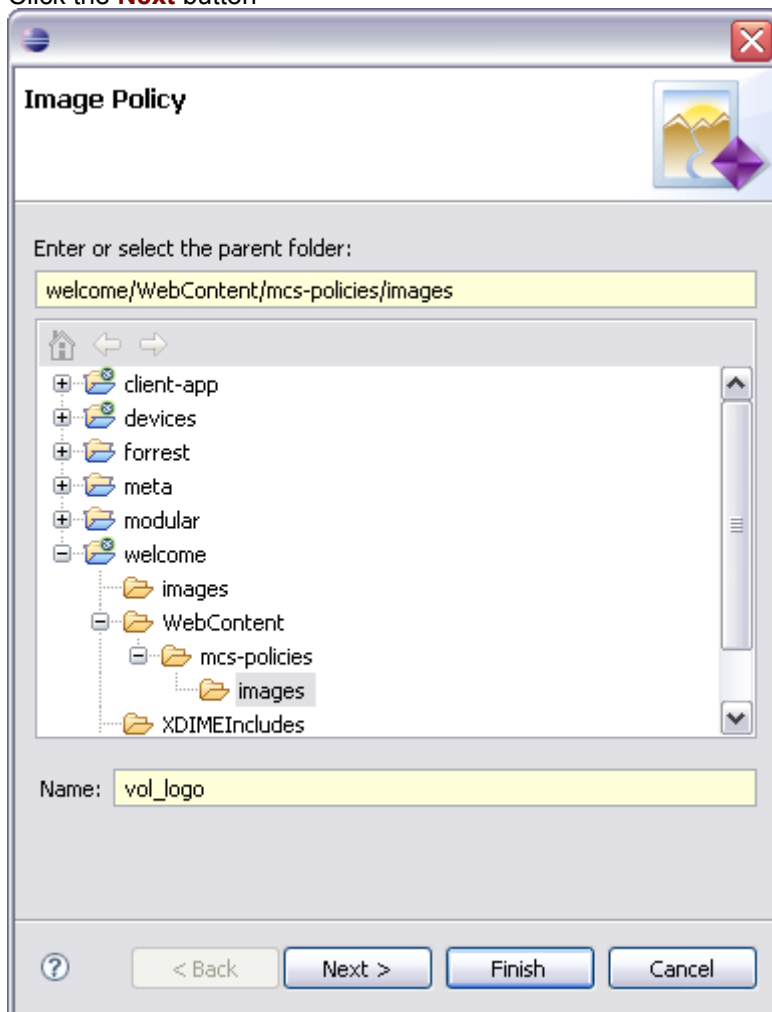
Adding an image component

Now you will see how we added an image component for the Volantis logo to the project, created an image variant, and associated it with a standard device family.

All the image files are located in the `welcome/images` directory. The image component policies are in the `welcome/WebContent/mcs-policies/images` directory.

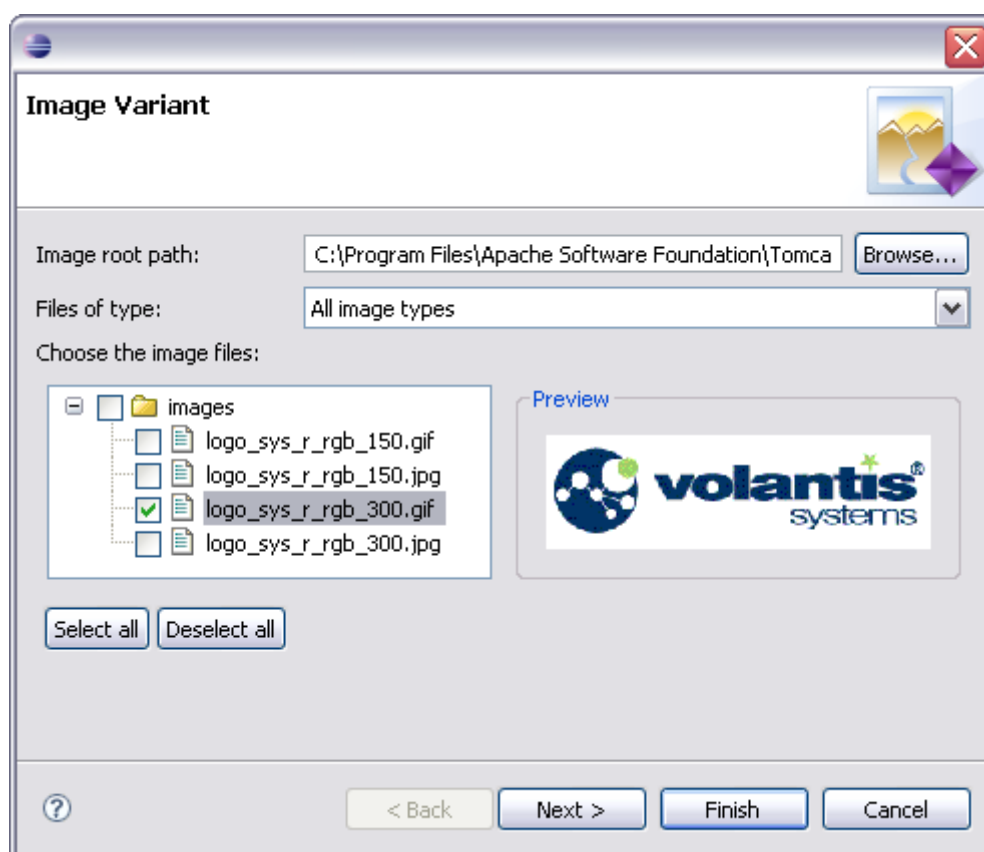
Adding a component

1. Choose **File | New | Image Component**
2. Select the `welcome/WebContent/mcs-policies/images` folder in the Navigator view, and enter 'vol_logo' as the component name. MCS will automatically append the `.mimg` image component policy file extension when the file is created.
3. Click the **Next** button



Adding a variant

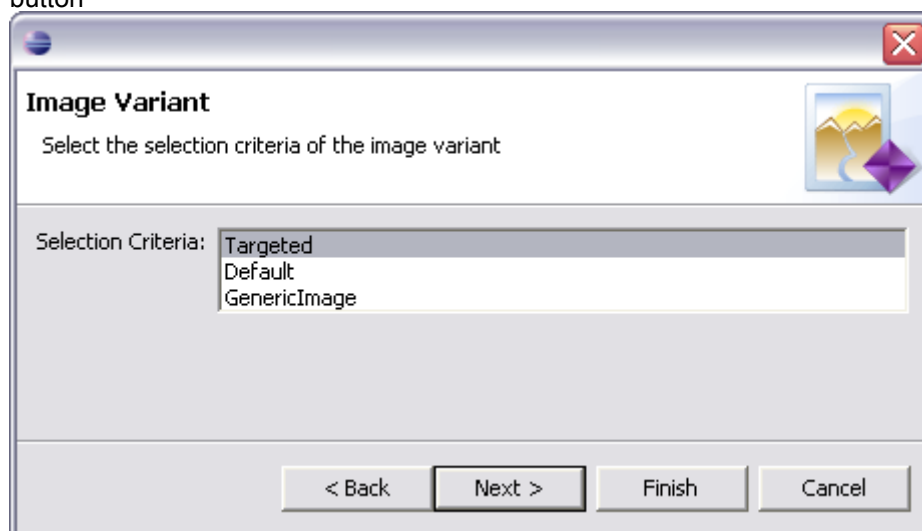
1. Use the **Browse** button to locate `welcome/images` folder as the *Image root path*
2. Select 'GIF(*.gif)' from the *Files of type* list to filter the list, and choose `logo_sys_r_rgb_300.gif` from the `images/PC` directory. To preview the variant, select the filename.
3. Check the box next to the filename to select it, and click the **Next** button. You should choose no more than one variant to enable the **Next** button and target devices at this time.



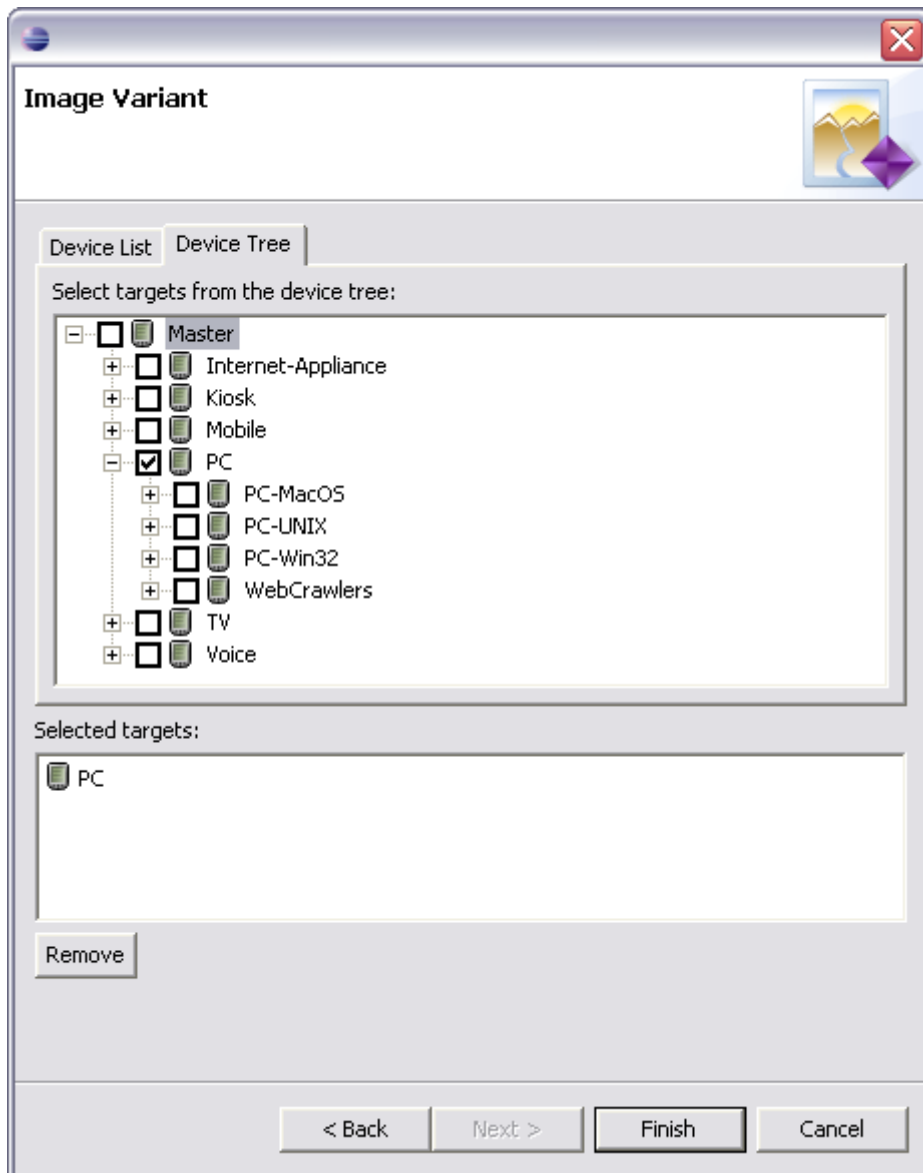
Choosing a device

Most component variants are device specific. They can be targeted at a particular device or device family. MCS searches the repository for a variant which is targeted at the device currently in use. If there is no exact match, MCS searches for variants that match the parent of the current device in the device tree. This search continues up the hierarchy until either a variant is found, or MCS reaches the root of the device tree.

1. On the next page of the wizard choose 'Targeted' in the *Selection criteria* box, and click the **Next** button



2. Go to the *Device Tree* tab and expand the 'Master' tree to find 'PC' devices. Check the box next to the device to select it, and click the **Finish** button.

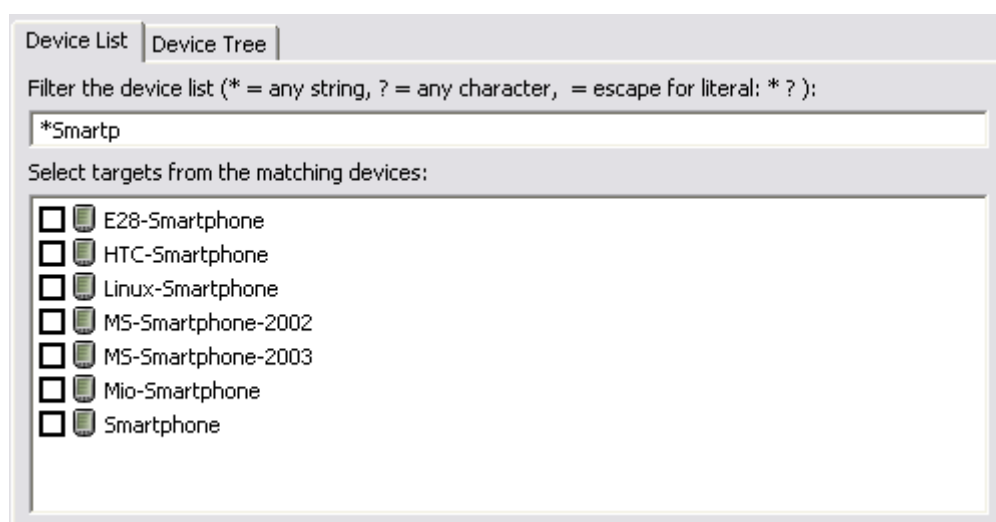


3. MCS displays the Image Component Editor. Select the PC variant in the *Variants* section to display the related attributes.

Adding the Smartphone variant

Now we will add a logo variant for Smartphone devices to the existing policy.

1. In the policy editor click the **New** button located below the Variants section to open the wizard
2. This time choose `logo_sys_r_rgb_150.gif` from the `images` directory
3. Click the **Next** button
4. On the next page of the wizard pick the 'Targeted' and click **Next**
5. On the *Device List* tab type '*Smartphone' in the top text box to filter list of available devices. The list of matched devices is updated as you type.



6. Check the 'Smartphone' box and click **Finish**. The new variant is listed in the Variants section.

Creating a layout

Next we use a layout to specify the physical position of elements on pages. To display content on a range of devices you need to define variants that are suitable for different device characteristics.

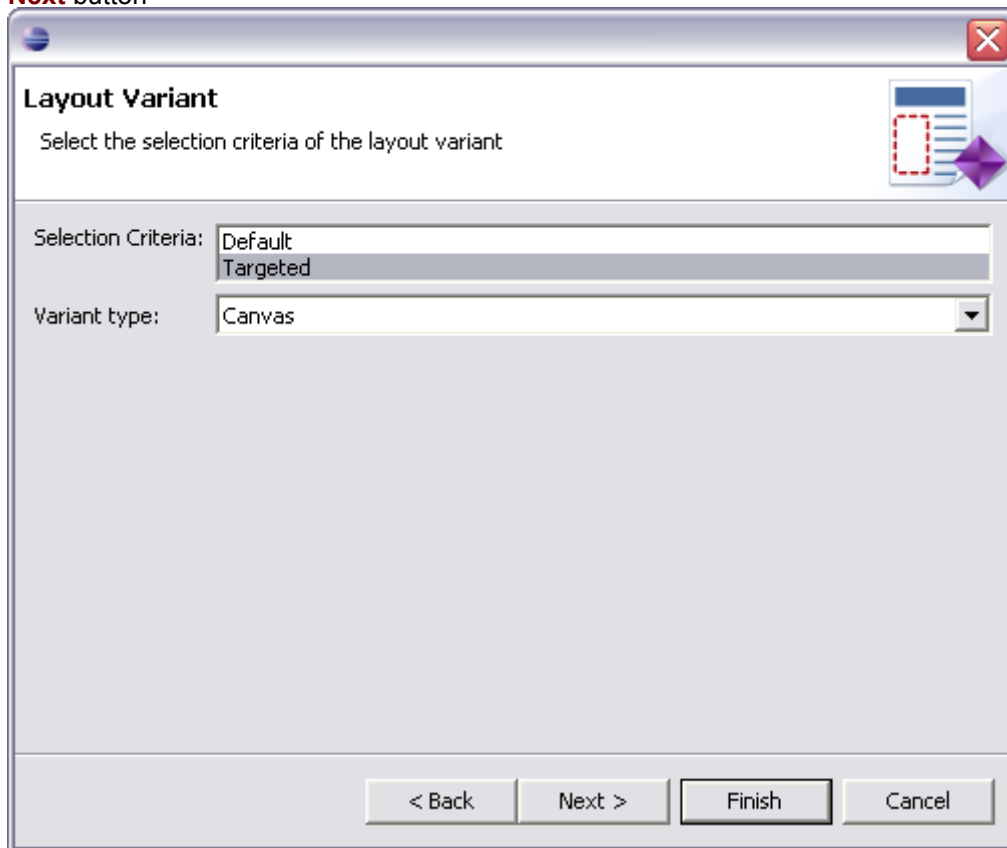
This task is very like creating an image policy. This time you will learn a quick way to choose a wizard.

Adding a layout

1. Right click on the `welcome/WebContent/mcs-policies` folder, and choose **New | Layout** from the context menu
2. Call the new layout 'welcome' and then click the **Next** button. The wizard will automatically add the `.mlyt` layout file extension.

Choosing a device

1. Choose 'Targeted' in the *Selection Criteria* box, select 'Canvas' as the *Variant type* and click the **Next** button



2. Type 'PC' in the top box to filter the list of available devices
3. From the filtered list of devices choose 'PC', by checking the box next to it, and click the **Finish** button. The new layout file called 'welcome.mlyt' is displayed in the `welcome/WebContent/mcs-policies` folder and is opened in the Layout editor.
4. Now click the **New** button under the variants section and add a layout variant for the Smartphone
5. The editor has two pages, an Overview page listing the layout variants and their general attributes, and a Design page where you can develop your layout.

Alerts and Actions

No alerts at this time.

Attributes

The following attribute values are defined for the policy:

Device category policy:

Variants

The following variants are defined for the policy:

Targeted, PC

Variant Selection Criteria

The following fields define the selection criteria for the selected variant:

Criteria: Targeted

PC

Browse... Delete

Targets:

Variant Content

The following fields define the content for the selected variant:

Default fragment:

Default segment:

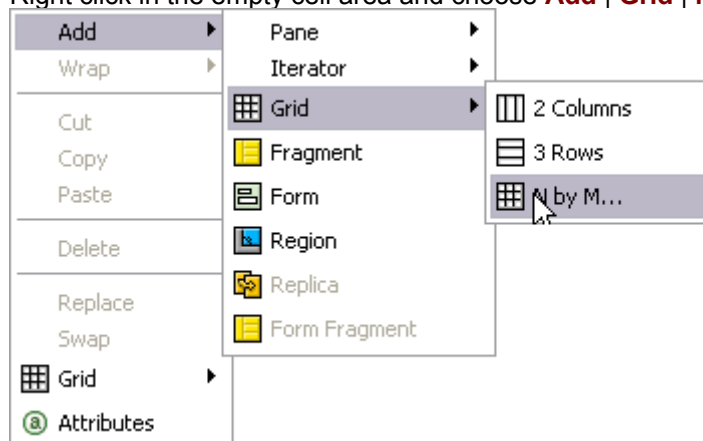
New... Add Delete

Overview Design

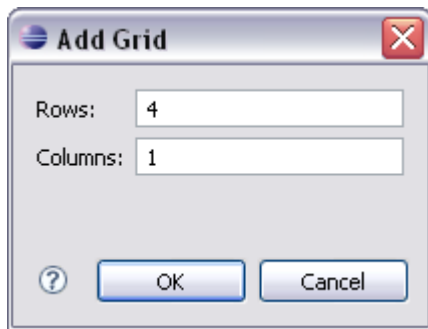
Using grids

When designing a layout you start with a grid structure that defines the skeleton of the page. Next, you add elements that hold targeted content. Grids give a device layout a basic structure for your content. You can nest grids inside one another, and also wrap one grid element in another.

1. Pick the 'Targeted, PC' variant in the list of variants and switch to the Design page by clicking on the tab at the bottom of the window, to display a single empty layout cell
2. Right click in the empty cell area and choose **Add | Grid | N by M** to open the Add Grid dialog

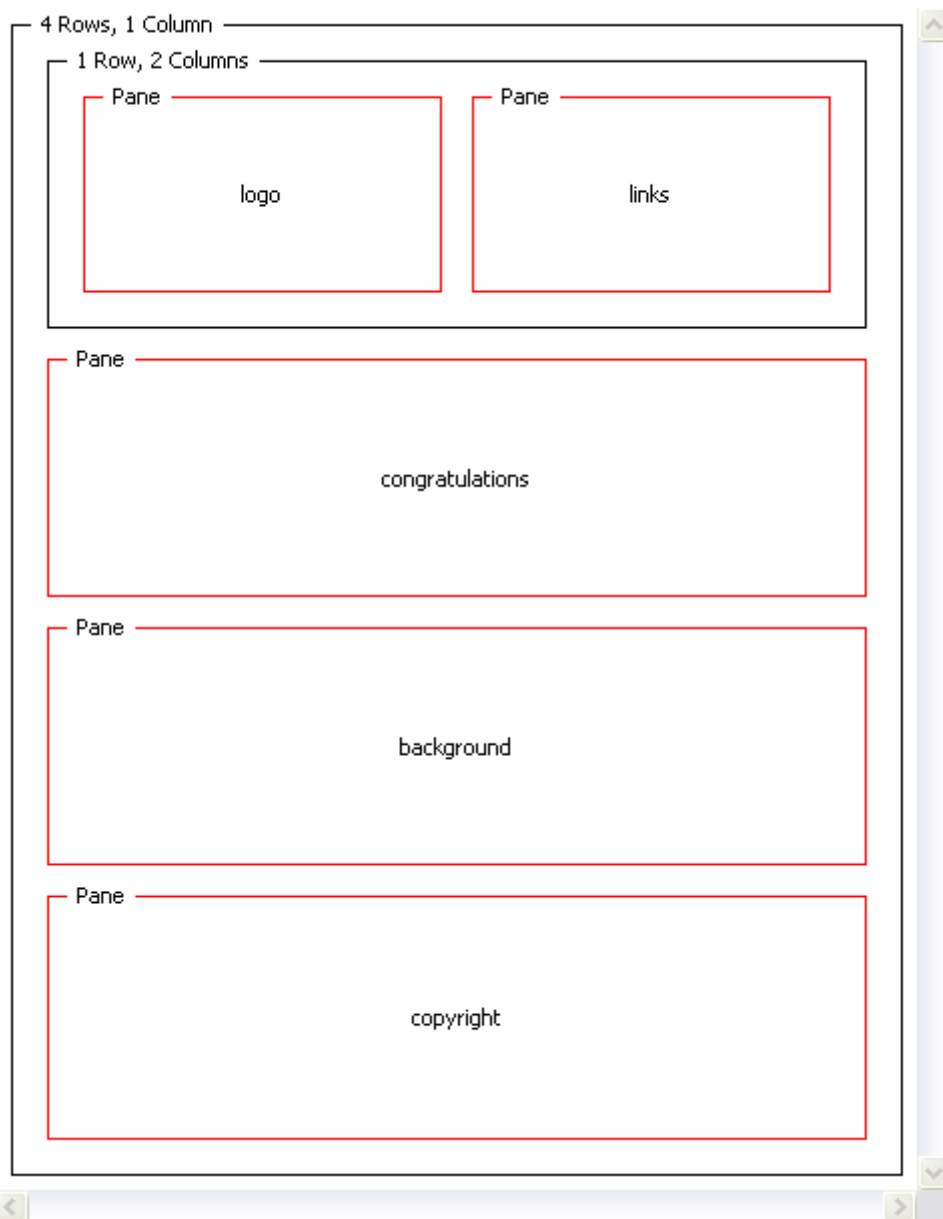


3. Enter values for 4 rows and 1 column and click **OK**. There are now empty cells for each row.



4. In the top row add 2 columns to hold the logo and links
5. Your final grid structure should look like the one below

Layout Design



Adding a format

MCS displays content in individual panes, which need to have a unique names within the layout.

1. In the top row of the grid, right click on the left-hand cell, and choose **Add | Pane | Pane**. The added pane is highlighted in red. If it is not already visible, MCS displays the Format Attributes view

2. With this pane selected in the page, set the *Name* control in the Format Attributes view to 'logo'. On the page the name is displayed.
3. Continue to add panes and attribute values for the remaining cells in the grid. The values are summarized below.

Mobile layout

The Smartphone layout is similar, using a 5x1 grid, with the links section immediately before the copyright notice. Create it as an exercise, but make the layout more flexible by targeting it at all mobiles on the device tree.

Checklist

Set the following format attribute values for the PC layout.

Name	Width	Border Width	Cell Padding	Cell Spacing	Horizontal Alignment	Vertical Alignment	Markup Optimization	Filter Usability Below
logo	100%	0	0	0	Left	Top	Never	0
congratulations								
background								
links					Left	Center		
copyright					Left			

These are the format attribute values for the mobile page layout.

Name	Width	Border Width	Cell Padding	Cell Spacing	Horizontal Alignment	Vertical Alignment	Markup Optimization	Filter Usability Below
logo	100%	0	0	0			Never	0
congratulations								
background						Bottom		
links					Left			
copyright								

Adding a theme

While you use layouts to specify the physical position of elements on pages, you employ style rules, defined in the theme, for values such as, fonts, colors, backgrounds and borders. Each theme can have variants that you define for individual devices or groups of devices. Styles are device-dependent because devices differ greatly in the support available for this kind of information. So MCS has to identify the requesting device before applying a set of rules. Each variant is made up of a set of style rules, and each rule is identified by a selector, and has a set of properties.

You use the Theme wizard and the Theme editor to associate variants with devices, and to define selectors and properties. You can also use the Style Rules view to get an overall picture of the themes for a particular theme variant, and perform some limited editing actions.

As we did for the layouts, we will start by creating a theme policy, and then target a PC variant.

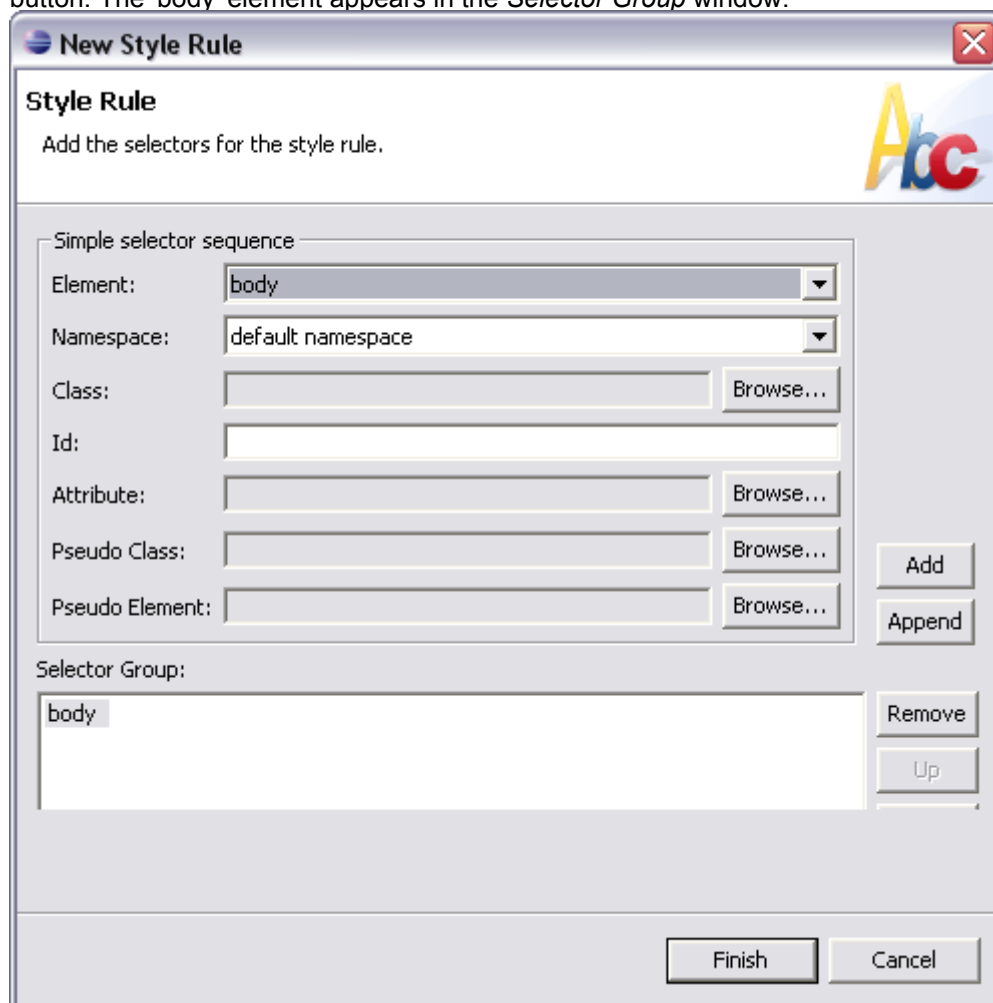
Adding a theme

1. Choose the `welcome/WebContent/mcs-policies` folder, right click and choose **New | Theme**
2. Call the theme 'welcome' and then click the **Next** button. MCS will automatically add the `.mthm` file extension to the theme name.
3. Choose **Targeted** in the *Selection Criteria* dropdown and click **Next**
4. Select the 'PC' device family from either of the tabs and then click the **Finish** button

Adding a selector

The Theme editor has two pages, an Overview page listing the theme variants and their general attributes, and a Design page where you can specify your style selectors and define style properties.

1. Select the PC variant and the **Design** tab
2. Click the **New** button in the *Selectors* section
3. In the *New Style Rule* window choose the 'body' element from the drop down list and click the **Add** button. The 'body' element appears in the *Selector Group* window.

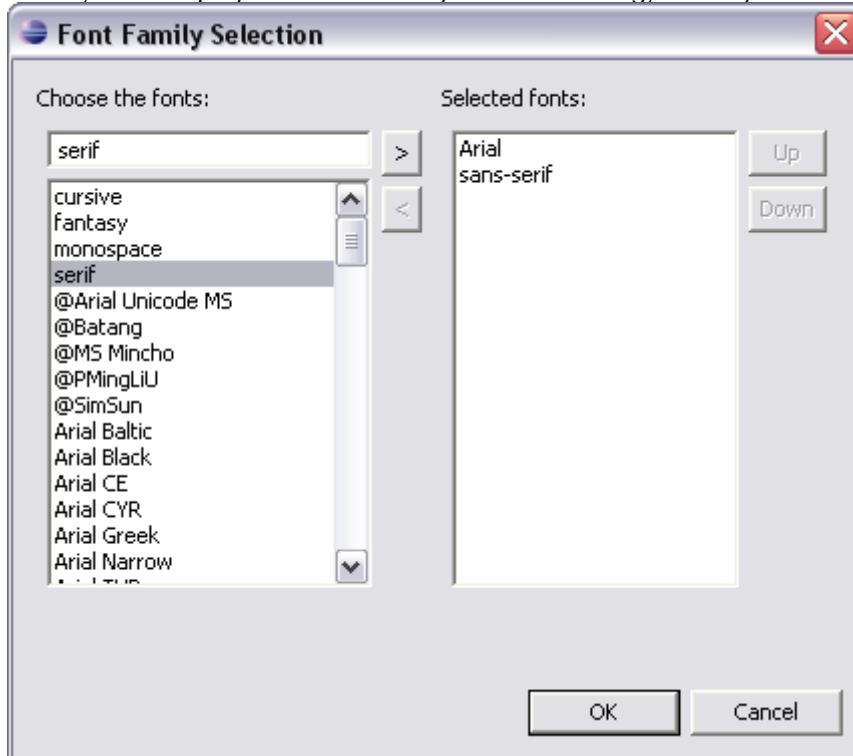


- Click **Finish** to add selector to the theme. The 'body' element appears in the *Selectors* box.

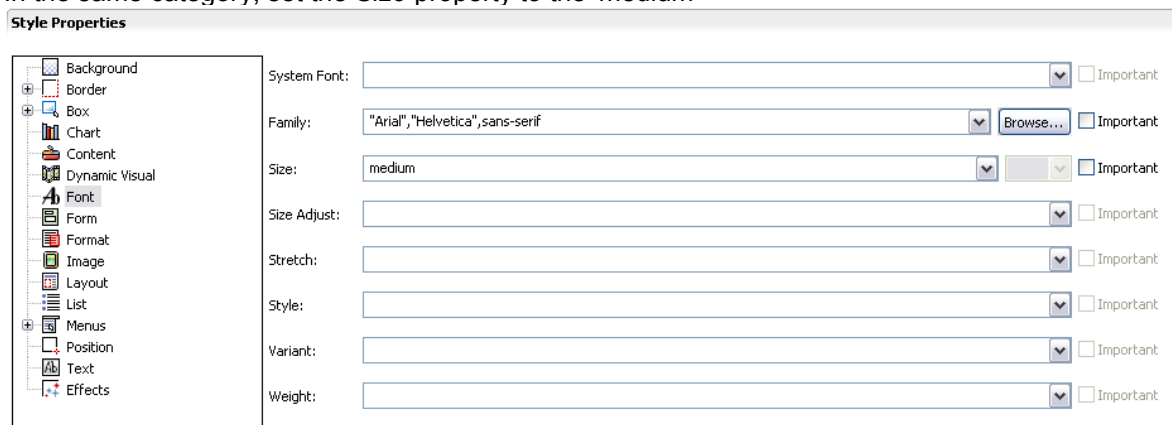
Setting style properties

You set properties in the Style Properties section of the editor. The properties are divided into categories, some of which you can expand if required.

- Choose the 'body' element in the Selectors section
- Select the Font category and set the *Family* property to 'Arial,sans-serif'. If you click the Browse button, MCS displays the Font Family Selection dialog, where you can build a list to use.



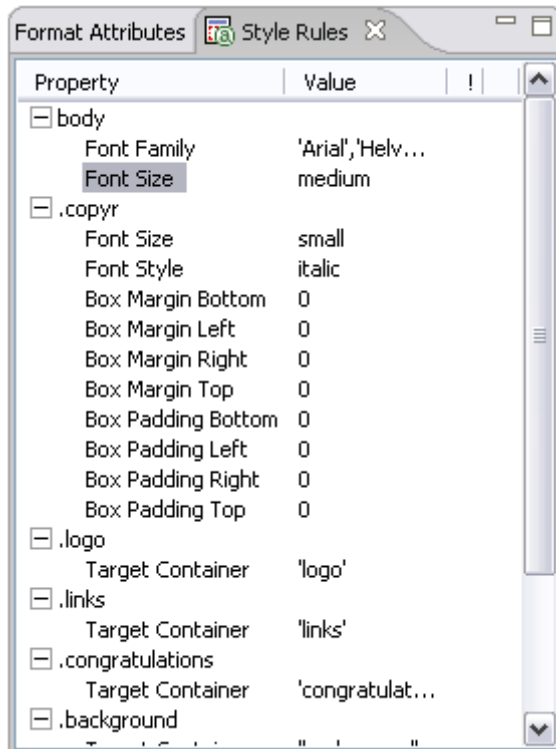
- In the same category, set the *Size* property to the 'medium'



Style Rules view

The Style Rules view provides a convenient way to view all the rules for a particular theme variant. It contains a table that displays the style rules, organized by selector sequence.

- Choose **Window | Show View | Other**, and select the Style Rules view under MCS Views
- Pick a theme variant on the Overview page to populate the view
- Expand a selector and choose a property. MCS displays the property in the Design page.



Checklist

The selectors and classes for both variants are listed in the tables below.

PC variant

Property	Control	Value
body		
Font	Family	'Arial', sans-serif
	Size	medium
.copyr		
Font	Size	small
	Style	italic
Box Margin	All	0
Box Padding	All	0
.logo		
Layout	Target Container	'logo'
.links		
Layout	Target Container	'links'
.congratulations		
Layout	Target Container	'congratulations'
.background		
Layout	Target Container	'background'
.copyright		
Layout	Target Container	'copyright'
a		
Box	Display	list-item
List	Type	none

Mobile variant

Property	Control	Value
body		

Property	Control	Value
Font	Family	'Arial', sans-serif
	Size	8pt
.copyr		
Font	Size	8pt
	Style	italic
Box Margin	All	0
Box Padding	All	0
.logo		
Layout	Target Container	'logo'
.links		
Layout	Target Container	'links'
.congratulations		
Layout	Target Container	'congratulations'
.background		
Layout	Target Container	'background'
.copyright		
Font	Size	8pt
Layout	Target Container	'copyright'
xforms select1		
Font	Size	8pt
xforms submit		
Font	Size	8pt
a		
Box	Display	list-item
List	Type	none

Welcome page

An XDIME 2 page has a well defined structure based on XHTML 2. It must start with a `html` element containing namespace declarations for the vocabularies in use. The majority of XDIME 2 presentation elements do not require a namespace prefix. However, there are some groups of elements, that do require a prefix.

The MCS layout and theme policies bound to an XDIME 2 page are declared in the `head` section using the `link` element. The required `title` element defines the title of the page.

The standard welcome page in MCS uses methods that are a bit advanced for this introduction. So we will create a page with identical content using the same underlying element structure. When you follow the tutorials, you will see how templating and includes can be used to handle repeating content.

Declaring XML namespaces

1. Open your XML editor, create a new file, and save it as `simple_welcome.xdime` in the project folder
2. Enter the following markup, which is a typical namespace declaration.

```
<?xml version="1.0" encoding="UTF-8"?>
<html
  xmlns="http://www.w3.org/2002/06/xhtml12"
  xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs">
</html>
```

Binding to the layout and theme

1. Enter the code shown below to bind the `welcome.mlyt` layout and the `welcome.mthm` theme to this page called 'Welcome to XDIME XML'

```
<head>
  <title>Welcome to XDIME XML</title>
  <link rel="mcs:layout" href="/welcome.mlyt"/>
  <link rel="mcs:theme" href="/welcome.mthm"/>
</head>
```

Body text

1. Next, enter a `body` element, and a `div` to contain a reference to the logo image component. The `class` attribute on the `div` targets the content to this pane. XDIME 2 uses the `object` to refer to media. Notice that the fallback text is contained in the element content, rather than in an attribute value.

```
<body>
  <div class="logo">
    <object src="images/vol_logo.mimg">
      <span>Volantis Systems Ltd</span>
    </object>
  </div>
</body>
```

2.

```
<div class="congratulations">
  <h2>Congratulations!</h2>
  <p>You have successfully installed and configured
    the Volantis Mobility Server&#0x94;.</p>
</div>
```

3. Here is the content for the background pane

```
<div class="background">
  <h3>Volantis Mobility Server</h3>
  <p>Volantis Mobility Server (VMS) provides a comprehensive, standards-based,
    development and execution environment for delivering multi-channel content,
    applications and services that are automatically optimized for thousands
    of different consumer devices.</p>
  <p>Built around the principle of device-independent delivery, VMS uniquely
    separates content, design, and device-issues into abstract device-independent
    policies. This enables a 'create once, run anywhere' environment which reduces
    complexity, cost and time to market for both development and maintenance.</p>
</div>
```

4. Enter the links section

```
<div class="links">
  <h2>Useful links:</h2>
  <p>
```

```
<a href="http://www.volantis.com">Volantis home page</a>
<a href="http://support.volantis.com/">Support gateway</a>
<a href="http://community.volantis.com/">Community</a>
</p>
</div>
```

5. Last of all add the copyright pane content

```
<div class="copyright">
  <p class="copyr">Copyright&#0x93; 2000-2008 Volantis Systems Ltd. All
    Rights Reserved.</p>
  <p class="copyr">Volantis&#0x94; is a trademark of Volantis Systems Ltd.</p>
</div>
```

Checklist

Here are some reference notes on the XDIME elements you have met so far.

Name	Purpose
html	The container for the XDIME 2 document.
head	Information such as the title and other metadata that is not document content. The contained <code>title</code> element is required.
title	The title element is used to identify the document. It can contain only text.
link	Defines a single link. Multiple links and relationships may be used in the <code>head</code> section of a document.
body	Contains the document's content.
div	A section used to add extra structure to documents. Style sheets can be used to control the presentation.
object	A reference to a media resource. <code>Object</code> allows a variety of media resources to be placed on the rendered page. The resource may be specified as an image policy or a device depended mime type. <code>Object</code> also allows the control of media transcoding to ensure the resource will render on the requesting agent. See <i>References to media</i> for details.
h1, h2, h3, h4, h5, h6	Heading elements for sections within a document. The number associated with each element indicates its relative position in the hierarchy of headings, with 1 indicating the top level and 6 the lowest level.
p	Block element that defines a paragraph.
a	Used to create an explicit link to another place in the same document or to another document. The current document is the source of the link and the value of the <code>href</code> attribute, defines the link target

Form content

Now we'll add a very simple form to the welcome page. A single-select list will allow users to pick a component description for Volantis Mobility Server. The variable response will be served by a JSP.

The additions include a namespace declaration for the form elements, an `xf:model` definition inside the `head` element, an `xf:select1` element and an `xf:submit` button.

Adding form elements

1. First, add the namespace declaration to the attributes of the `html` element.
`xmlns:xf="http://www.w3.org/2002/xforms"`
2. Now, include the following code in the `head` section of the page. The `xf:model` element defines the data model for the form. In the `xf:submission` element, the `action` attribute contains the definition URL to be used when the form is submitted. The form data is processed by `component.jsp`, described in the next topic.

```
<xf:model>
  <xf:submission id="part" action="component.jsp"/>
</xf:model>
```
3. In the background section, after the existing content, enter the code for the section header, and the `xf:select1` element. The `ref` attribute contains the name of the parameter passed to the JSP. There is one `xf:item` for each component in the dropdown list. `xf:label` provides the visible listing and `xf:value` contains the parameter value that will be passed to the JSP. The label for the list as a whole is required, but we have left it empty.

```
<h3>System components</h3>
<p>Have a look at the key components in Volantis Mobility Server.</p>
<p>
  <xf:select1 ref="item">
    <xf:label/>
    <xf:item>
      <xf:label>Multi-Channel Server</xf:label>
      <xf:value>mcs</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Media Access Proxy</xf:label>
      <xf:value>map</xf:value>
    </xf:item>
    <xf:item>
      <xf:label>Message Preparation Server</xf:label>
      <xf:value>mps</xf:value>
    </xf:item>
  </xf:select1>
</p>
```
4. Finally, add the `xf:submit` button element within the same paragraph. The `submission` attribute must match the `id` attribute value on the `xf:submission` element in the model.

```
<xf:submit submission="part">
  <xf:label>Go</xf:label>
</xf:submit>
```

Checklist

Take some time to review the structure of the XForms elements in detail by following the links below.

Name	Purpose
<code>xf:model</code>	Represents a form definition, used as a container for elements associated with its submission.
<code>xf:submission</code>	Defines the information to submit, where it should be sent, and how to send it.
<code>xf:select1</code>	A form control that provides for selection of a single choice from an available list of options; for example, a group of radio buttons.
<code>xf:item</code>	Container for the option values in <code>xf:select</code> and <code>xf:select1</code> elements.
<code>xf:label</code>	Provides a descriptive label for forms controls.
<code>xf:value</code>	Provides the option values associated with select controls.

Name	Purpose
xf:submit	A form control that initiates submission of the form data for processing.
Form attributes	Binding expressions provide the mapping between the data associated with a particular control and the data provided for processing when the form is submitted. The expression is the name of the parameter associated with the control's data when the form is submitted. These parameters appear on the submission URL if the form is submitted using the HTTP <code>GET</code> method, and are hidden if the <code>POST</code> method is used. The methods are specified as attributes on the <code>xf:submission</code> element.

Complete XDIME

The complete code for the XDIME welcome page follows. It only remains to work on the response.

```
<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/2002/06/xhtml2"
  xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs"
  xmlns:xf="http://www.w3.org/2002/xforms">
  <head>
    <title>Welcome to XDIME XML</title>
    <link rel="mcs:layout" href="welcome.mlyt"/>
    <link rel="mcs:theme" href="welcome.mthm"/>
    <xf:model>
      <xf:submission id="part" action="component.jsp"/>
    </xf:model>
  </head>
  <body>
    <div class="logo">
      <object src="images/vol_logo.mimg">
        <span>Volantis Systems Ltd.</span>
      </object>
    </div>

    <div class="congratulations">
      <h2>Congratulations!</h2>
      <p>You have successfully installed and configured Volantis Mobility
        Server&#x2122;.</p>
    </div>

    <div class="background">
      <h3>Volantis Mobility Server</h3>
      <p>Volantis Mobility Server provides a comprehensive, standards-based,
        development and execution environment for delivering multi-channel
        content, applications and services that are automatically optimized
        for thousands of different consumer devices.</p>
      <p>Built around the principle of device-independent delivery, Volantis
        Mobility Server uniquely separates content, design, and
        device-issues into abstract device-independent policies. This
        enables a 'create once, run anywhere' environment which reduces
        complexity, cost and time to market for both development and
        maintenance.</p>
      <h3>System components</h3>
      <p>Have a look at the key components in Volantis Mobility Server.</p>
      <p>
        <xf:select1 ref="item">
          <xf:label> </xf:label>
          <xf:item>
            <xf:label>Multi-Channel Server</xf:label>
            <xf:value>mcs</xf:value>
          </xf:item>
          <xf:item>
            <xf:label>Media Access Proxy</xf:label>
            <xf:value>map</xf:value>
          </xf:item>
          <xf:item>
            <xf:label>Message Preparation Server</xf:label>
            <xf:value>mps</xf:value>
          </xf:item>
        </xf:select1>
        <xf:submit submission="part">
          <xf:label>Go</xf:label>
        </xf:submit>
      </p>
    </div>

    <div class="links">
      <h3>Useful links</h3>
      <p>
```

```
        <a href="http://www.volantis.com">Volantis home page</a>
        <a href="http://support.volantis.com/">Support gateway</a>
        <a href="http://community.volantis.com/">Community</a>
    </p>
</div>
<div class="copyright">
    <p/>
    <p class="copyr">Copyright &#x00A9; 2000-2008 Volantis Systems Ltd.
        All Rights Reserved.</p>
    <p class="copyr">Volantis&#x2122; is a trademark of Volantis
        Systems Ltd.</p>
</div>

</body>
</html>
```

Response JSP

This page illustrates how to generate XDIME 2 from a Java Server Page, based on the same theme and layout as the welcome page.

MCS handles JSPs by identifying the `.jsp` file extension in a `web.xml` filter definition, just as it does with `.xdime`. The setting that follows is made by default.

```
<filter-mapping>
  <filter-name>MCSFilter</filter-name>
  <url-pattern>*.xdime</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>MCSFilter</filter-name>
  <url-pattern>*.jsp</url-pattern>
</filter-mapping>
```

JSP content

1. Create an XML file `component.jsp` in the main welcome folder
2. The script takes a single 'item' parameter, passed by the form. Enter the following code after the XML declaration, but before the `html` element. Note that the content type of the response is set to `x-application/vnd.xdime+xml`

```
<?xml version="1.0" encoding="UTF-8"?>
<%@ page import="com.volantis.mcs.servlet.*" %>
<%
  String result = null;
  response.setContentType("x-application/vnd.xdime+xml");
  String item = MarinerServletRequestContext.
    findInstance(request).getParameter("item");
%>
```

3. As you did for the welcome page, add the common sections to the page. This time use the congratulations section for the title, and omit the links section.

```
<html xmlns="http://www.w3.org/2002/06/xhtml12"
  xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs">
  <head>
    <title>Component information</title>
    <!-- Links to the layout and theme for this page -->
    <link rel="mcs:layout" href="/welcome.mlyt"/>
    <link rel="mcs:theme" href="/welcome.mthm"/>
  </head>
  <body>
    <div class="logo">
      <!-- the logo pane -->
      <object src="images/vol_logo.mimg">
        <span>Volantis Systems Ltd.</span>
      </object>
    </div>
    <div class="copyright">
      <p class="copyr">Copyright &#x00A9; 2000-2008 Volantis Systems Ltd.
        All Rights Reserved.</p>
      <p class="copyr">Volantis&#x2122; is a trademark of Volantis
        Systems Ltd.</p>
    </div>
  </body>
</html>
```

4. The variable part of the response is handled in the background pane by an if/else selector, with the rendering depending on the 'item' parameter value. Enter the following code.

```
<div class="background">
<% if (item.equals("mcs")) { %>
  <h3>Multi-Channel Server</h3>
  <p>Multi-Channel Server (MCS) helps you manage the
    complexity of delivering a wide variety of content to PCs,
    PDAs, mobile phones, interactive digital TV, internet
    appliances, games consoles, VoiceXML and interactive kiosks.</p>
  <p>To deliver to multiple channels, you need to present content,
    services and applications in a consistent way across all
    target devices. In MCS you can separate application design
    from device delivery, and build a cost-effective and scalable
```



```

        system, by defining policies.</p>
<%> else if (item.equals("map")) { %>
    <h3>Media Access Proxy</h3>
    <p>The Media Access Proxy (MAP) is a web application that is used to
        perform transcoding and transformation operations on media files
        referenced by XDIME 2. It also handles XDIME content and ICS URLs in a
        backward compatible manner.</p>
<%> } else { %>
    <h3>Message Preparation Server</h3>
    <p>Message Preparation Server (MPS) builds on the core functionality
        of MCS to allow the optimization of message-based or WAP push content. It
        provides the ability to write applications to generate and
        transmit messages to subscribers' devices. This allows
        applications to be created that can support mass distribution
        of messages to provide significant end user function. The
        messages might, for example, contain information that users
        had subscribed to.</p>
<%> } %>
</div>

```


5. Finally add a link back to the welcome page.

```

<p>
    <a href="simple_welcome.xdime">Welcome page</a>
</p>

```

6. Now you can test your form. The response for the Media Access Proxy component should look like this.



Component information

Media Access Proxy

The Media Access Proxy (MAP) is a web application that is used to perform transcoding and transformation operations on media files referenced by XDIME 2. It also handles XDIME content and ICS URLs in a backward compatible manner.

[Welcome](#)

Copyright © 2000-2008 Volantis Systems Ltd. All Rights Reserved.
Volantis™ is a trademark of Volantis Systems Ltd.

Summary

Congratulations! You have completed your first steps on the way to learning how to use Volantis Mobility Server. To follow up, we suggest you move on to *Next steps with MCS*. You will find it reinforces what you have learned here, with more detailed layout and theme arrangements and sample pages for a full web application - the Jive Sports site.

But there is more to follow here if you wish. There is a topic which shows how to set up a remote project using PHP and Ruby. Then, if you are familiar with XSLT, you can have a look at the topic *Migrating to XDIME*, which provides an example XSL stylesheet to transform the welcome page from XHTML to XDIME.

Remote projects - PHP and Ruby

If you want to render XDIME from a portal that is developed using another framework, for example PHP or Ruby, you will need to host your content on a remote server, and direct MCS accordingly.

All that is required to configure MCS is additional filters in the `web.xml` file for your chosen language, and a change to a single properties file.

Then you can modify your server to render XDIME rather than XHTML. We will use some very simple content to illustrate the approach.

Note: Ensure that the content type for the remote server is set to `x-application/vnd.xdime+xml`, otherwise the page will not render correctly.

Render from PHP

Suppose your existing site has been developed with PHP. Create a page with the following content, and save it as `test.php` on your remote server. By now you are familiar with some of the XDIME namespaces. We have included all of those that you might need on your site in this example.

We have used the built in layout `error.mlyt` which MCS uses to render error pages.

```
<?php header("Content-type: x-application/vnd.xdime+xml"); ?>
<?php print"<?xml version='1.0' encoding='UTF-8'?">

<html xmlns="http://www.w3.org/2002/06/xhtml2"
  xmlns:xf="http://www.w3.org/2002/xforms"
  xmlns:si="http://www.volantis.com/xmlns/2006/01/xdime2/si"
  xmlns:sel="http://www.w3.org/2004/06/diselect"
  xmlns:pipeline="http://www.volantis.com/xmlns/marlin-pipeline"
  xmlns:template="http://www.volantis.com/xmlns/marlin-template"
  xmlns:urid="http://www.volantis.com/xmlns/marlin-uri-driver"
  xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs" >
  <head>
    <title>Hello</title>
    <link rel="mcs:layout" href="/error.mlyt"/>
  </head>
  <body>
    <div style="mcs-container: 'error'">Hello MCS, this is PHP.</div>
  </body>
</html>"; ?>
```

Configure MCS

1. You will recall how we added a filter to the `web.xml` file for JSP pages. A similar entry is needed for PHP.


```
<filter-mapping>
  <filter-name>MCSFilter</filter-name>
  <url-pattern>*.php</url-pattern>
</filter-mapping>
```
2. An additional 'remapping' filter entry is needed to handle content from your remote site


```
<filter-mapping>
  <filter-name>RemappingFilter</filter-name>
  <url-pattern>*.php</url-pattern>
</filter-mapping>
```
3. Open the `mcs/projects.properties` file, which may refer to multiple remote servers. By default it contains


```
remote=http://myhost:8080/resource/
```
4. Add a property prefix and the URL of your service. The prefix will be used in requests to the remote server.


```
php_server=http://myphp:80/
```
5. The URL of your file should take the form `http://[mcshost]:[mcsport]/mcs/[property_prefix]/[path_to_file]`. The prefix must immediately follow the 'mcs' part. So to request the page you would use


```
http://mcs_server:8080/mcs/php_server/test.php
```

Render from Ruby

Ruby pages can be set up in a similar fashion.

1.

```
#!/usr/bin/ruby
require 'cgi'
cgi = CGI.new()

cgi.out("x-application/vnd.xdime+xml") {
  "<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n\n" +
  "<html xmlns=\"http://www.w3.org/2002/06/xhtml2\"\n" +
  "  xmlns:xf=\"http://www.w3.org/2002/xforms\"\n" +
  "  xmlns:si=\"http://www.volantis.com/xmlns/2006/01/xdime2/si\"\n" +
  "  xmlns:sel=\"http://www.w3.org/2004/06/diselect\"\n" +
  "  xmlns:pipeline=\"http://www.volantis.com/xmlns/marlin-pipeline\"\n" +
  "  xmlns:template=\"http://www.volantis.com/xmlns/marlin-template\"\n" +
  "  xmlns:urid=\"http://www.volantis.com/xmlns/marlin-uri-driver\"\n" +
  "  xmlns:mcs=\"http://www.volantis.com/xmlns/2006/01/xdime/mcs\" >\n" +
  "    <head>\n" +
  "      <title>Hello</title>\n" +
  "      <link rel=\"mcs:layout\" href=\"/error.mlyt\"/>\n" +
  "    </head>\n" +
  "    <body>\n" +
  "      <div style=\"mcs-container: 'error'\">Hello MCS, this is Ruby.</div>\n" +
  "    </body>\n" +
  "  </html>\n"
}
```
2. Enter the filter content in the `web.xml` file

```
<filter-mapping>
  <filter-name>MCSFilter</filter-name>
  <url-pattern>*.rbx</url-pattern>
</filter-mapping>
<filter-mapping>
  <filter-name>RemappingFilter</filter-name>
  <url-pattern>*.rbx</url-pattern>
</filter-mapping>
```
3. Enter a property like this

```
ruby_server=http://myruby:80/
```
4. Ruby requires CGI processing, so you need to modify the path in your request to

```
http://mcserver:8080/mcs/ruby_server/cgi-bin/test.rbx
```

Migrating to XDIME

This topic suggests some very basic ideas about how to migrate your existing site content to XDIME 2 from XHTML. We illustrate the development a simple XSL transform, which focuses on two notable differences between XHTML and XDIME 2 - handling media assets, and forms.

There are many possible variations in web portal design, and we encourage you to extend these ideas to meet the needs of your mobile site, taking account of handling a wider range of elements and attributes.

In the examples that follow, we have assumed that our welcome page is written in XHTML, and that it uses the same structure and styles as the mobile version we have recently created. We also assume that you are familiar with the essentials of XSLT processing.

You will need to generate the XDIME XML declarations, transform an `img` reference to an `object` reference, and generate XForms markup from the `form` element source. Everything else uses elements that are familiar from XHTML.

The source example is `source_welcome.html`. Note that we have decided on the name of the logo image component ahead of time, and used the `.mimg` file extension.

```
<?xml version="1.0" encoding="UTF-8"?>
<html>
  <head>
    <title>Welcome to XDIME XML</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div class="logo">
      
    </div>
    <div class="congratulations">
      <h2>Congratulations!</h2>
      <p>You have successfully installed and configured Volantis Mobility
        Server&#x2122;</p>
    </div>
    <div class="background">
      <h3>Volantis Mobility Server</h3>
      <p>Volantis Mobility Server provides a comprehensive, standards-based,
        development and execution environment for delivering multi-channel
        content, applications and services that are automatically optimized
        for thousands of different consumer devices.</p>
      <p>Built around the principle of device-independent delivery, Volantis
        Mobility Server uniquely separates content, design, and
        device-issues into abstract device-independent policies. This
        enables a 'create once, run anywhere' environment which reduces
        complexity, cost and time to market for both development and
        maintenance.</p>
      <h3>System components</h3>
      <p>Have a look at the key components in Volantis Mobility Server.</p>
      <form action="component.jsp" method="get">
        <p>
          <select name="item">
            <option value="mcs">Multi-Channel Server</option>
            <option value="map">Media Access Proxy</option>
            <option value="mps">Message Preparation Server</option>
          </select>
        </p>
        <p>
          <input type="submit" value="Go" />
        </p>
      </form>
    </div>
    <div class="links">
      <h3>Useful links</h3>
      <p>
        <a href="http://www.volantis.com">Volantis home page</a>
        <a href="http://support.volantis.com/">Support gateway</a>
        <a href="http://community.volantis.com/">Community</a>
      </p>
    </div>
  </body>
</html>
```

```
</p>
</div>
<div class="copyright">
  <p class="copyr">Copyright &#x00A9; 2000-2008 Volantis Systems Ltd. All
    Rights Reserved.</p>
  <p class="copyr">Volantis&#x2122; is a trademark of Volantis Systems
    Ltd.</p>
</div>
</body>
</html>
```

XSL preliminaries

In the stylesheet we will use three parameter values for the layout and theme names, and the binding between the `xf:model` and `xf:submit` elements, making the transform useful for a range of cases.

1. Enter the XML declaration, the `xsl:stylesheet` element, and other top level elements. The stylesheet must declare the same namespaces as those in the XDIME target.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
  xmlns="http://www.w3.org/2002/06/xhtml12"
  xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs"
  xmlns:xf="http://www.w3.org/2002/xforms">

  <xsl:param name="layout">welcome.mlyt</xsl:param>
  <xsl:param name="theme">welcome.mthm</xsl:param>
  <xsl:param name="bind">part</xsl:param>

  <xsl:output method="xml" encoding="UTF-8"/>

</xsl:stylesheet>
```

2. Next comes the main template, using the parameter values '\$layout' and '\$theme' to populate the link attribute values. Inside a test for the existence of a `form` element in the source, the '\$bind' parameter value sets the identifier of the form's `xf:submission` element, and the `action` attribute is copied.

```
<xsl:template match="/">
  <html xmlns="http://www.w3.org/2002/06/xhtml12"
    xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs"
    xmlns:xf="http://www.w3.org/2002/xforms">
    <head>
      <title><xsl:value-of select="//head/title"/></title>
      <link rel="mcs:layout" href="{ $layout }"/>
      <link rel="mcs:theme" href="{ $theme }"/>
      <xsl:if test="//form">
        <xf:model>
          <xf:submission id="{ $bind }">
            <xsl:attribute name="action">
              <xsl:value-of select="//form/@action"/>
            </xsl:attribute>
          </xf:submission>
        </xf:model>
      </xsl:if>
    </head>
    <body>
      <xsl:apply-templates select="//body//div"/>
    </body>
  </html>
</xsl:template>
```

Body content

1. The processing of the common elements is very straightforward, because of the similarities between XDIME and XHTML.

```
<xsl:template match="h2">
  <h2>
    <xsl:value-of select="."/>
  </h2>
</xsl:template>

<xsl:template match="h3">
  <h3>
```

```

        <xsl:value-of select="."/>
    </h3>
</xsl:template>

<xsl:template match="div">
    <div class="{@class}">
        <xsl:apply-templates/>
    </div>
</xsl:template>

<xsl:template match="p">
    <p>
        <xsl:apply-templates/>
    </p>
</xsl:template>

<xsl:template match="a">
    <a>
        <xsl:copy-of select="@*" />
        <xsl:value-of select="."/>
    </a>
</xsl:template>

```

2. Enter the following code to use the XDIME 2 `object` element instead of `img`. Note that the fallback text is contained in the element content as distinct from an attribute. You need not copy the height and width values. MCS will get these from the file system when you add the original image as a variant.

```

<xsl:template match="img">
    <object src="{@src}">
        <span>
            <xsl:value-of select="@alt" />
        </span>
    </object>
</xsl:template>

```

3. We need to detect whether or not a `select` element is a multiple select. The original `name` attribute is passed as the new `ref` attribute on the `xf:select` element. In this case the required top level `xf:label` is empty. Each option element is turned into `xf:label` and `xf:value` pairs.

```

<xsl:template match="//select">
    <xsl:choose>
        <xsl:when test="@multiple">
            <!-- handle multiple select cases -->
        </xsl:when>
        <xsl:otherwise>
            <xf:select1 ref="{@name}">
                <xf:label/>
                <xsl:for-each select="option">
                    <xf:item>
                        <xf:label>
                            <xsl:value-of select="."/>
                        </xf:label>
                        <xf:value>
                            <xsl:value-of select="@value" />
                        </xf:value>
                    </xf:item>
                </xsl:for-each>
            </xf:select1>
        </xsl:otherwise>
    </xsl:choose>
</xsl:template>

```

4. The XHTML input `type` attribute allows several possible values for different kinds of control. The example shows a test for the 'submit' value only. The '\$bind' parameter is used again to ensure that the `xf:submit` button triggers submission in the XForms model.

```

<xsl:template match="//input">
    <xsl:choose>
        <xsl:when test="@type='submit'">
            <xf:submit submission="{ $bind }">
                <xf:label>
                    <xsl:value-of select="@value" />
                </xf:label>
            </xf:submit>
        </xsl:when>
    </xsl:choose>

```

```
</xsl:when>
<!-- catch additional input types here -->
</xsl:choose>
</xsl:template>
```

5. Run the transform, and save the output in the main welcome folder as `simple_welcome_test.xdime`
6. Go to `http://localhost:8080/mcs/projects/welcome/simple_welcome_test.xdime` in your browser. You should see an identical page to the one you constructed as XDIME source.

Checklist

Now you can extend this XSL transform to apply additional XHTML elements, and experiment with some pages from your own site. The complete code is shown in the next topic.

XSL transform

Here is the source for the XSL transform you prepared in the previous topic. You will find the file `xhtml2xdime.xsl` in the main welcome folder in your project.

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"
  xmlns="http://www.w3.org/2002/06/xhtml12"
  xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs"
  xmlns:xf="http://www.w3.org/2002/xforms" exclude-result-prefixes="#default">
  <!--skeleton xhtml 1.0 to xdime 2-->
  <xsl:param name="layout">welcome.mlyt</xsl:param>
  <xsl:param name="theme">welcome.mthm</xsl:param>
  <xsl:param name="bind">part</xsl:param>
  <xsl:output method="xml" encoding="UTF-8"/>
  <xsl:template match="/">
    <html xmlns="http://www.w3.org/2002/06/xhtml12"
      xmlns:mcs="http://www.volantis.com/xmlns/2006/01/xdime/mcs"
      xmlns:xf="http://www.w3.org/2002/xforms">
      <!--more namespace decs as required-->
      <head>
        <title>
          <xsl:value-of select="//head/title"/>
        </title>
        <link rel="mcs:layout" href="{ $layout }"/>
        <link rel="mcs:theme" href="{ $theme }"/>
        <xsl:if test="//form">
          <xf:model>
            <xf:submission id="{ $bind }">
              <xsl:attribute name="action">
                <xsl:value-of select="//form/@action"/>
              </xsl:attribute>
            </xf:submission>
          </xf:model>
        </xsl:if>
      </head>
      <body>
        <xsl:apply-templates select="//body//div"/>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="h2">
    <h2>
      <xsl:value-of select="."/>
    </h2>
  </xsl:template>

  <xsl:template match="h3">
    <h3>
      <xsl:value-of select="."/>
    </h3>
  </xsl:template>

  <xsl:template match="div">
    <div class="{ @class }">
      <xsl:apply-templates/>
    </div>
  </xsl:template>

  <xsl:template match="p">
    <p>
      <xsl:copy-of select="@*" />
      <xsl:apply-templates/>
    </p>
  </xsl:template>

  <xsl:template match="a">
```



```
<a>
  <xsl:copy-of select="@*" />
  <xsl:value-of select="." />
</a>
</xsl:template>

<xsl:template match="img">
  <object src="{@src}">
    <span>
      <xsl:value-of select="@alt" />
    </span>
  </object>
</xsl:template>

<xsl:template match="//select">
  <xsl:choose>
    <xsl:when test="@multiple">
      <!-- handle multiple select cases with xf:select -->
    </xsl:when>
    <xsl:otherwise>
      <xf:select1 ref="{@name}">
        <xf:label/>
        <xsl:for-each select="option">
          <xf:item>
            <xf:label>
              <xsl:value-of select="." />
            </xf:label>
            <xf:value>
              <xsl:value-of select="@value" />
            </xf:value>
          </xf:item>
        </xsl:for-each>
      </xf:select1>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<xsl:template match="//input">
  <xsl:choose>
    <xsl:when test="@type='submit'">
      <xf:submit submission="{ $bind }">
        <xf:label>
          <xsl:value-of select="@value" />
        </xf:label>
      </xf:submit>
    </xsl:when>
    <!-- catch additional input types here -->
  </xsl:choose>
</xsl:template>
</xsl:stylesheet>
```