SCSS

Introduction

A SCSS module is provided for magnolia 5 as a community project on magnolia forge.
Since the Vaadin UI framework used by magnolia leverages SCSS and comes with a SCSS compiler, a simple module can expose this functionality for use by your themes.

SCSS, also known as SASS, is an extension to CSS. In a nutshell, SCSS extends CSS with many useful concepts such as *nested rules* or *mixins* and also lets you define variables and use simple programming constructs (like for loops) in your CSS stylesheets. A SCSS compiler turns your SCSS files into plain old CSS for shipping to the browser.

If you have large or complicated CSS files, or if it has ever bothered you that you have to define the same color values or font-families over and over again in the same CSS file, then SCSS might be for you.

More information on SCSS can be found at: http://sass-lang.com

Getting the module

For Magnolia 5.4 you'll want to use the current 1.0.3-SNAPSHOT version, just being tested for release. You can get it at https://git.magnolia-cms.com/projects/FORGE/repos/magnolia-module-resources-scss/

Please also see the technical details section below about how this module changes resource handling in Magnolia 5.4.

When installing the module, take care that it needs a newer version of vaadin-sass-compiler: `vaadin-sass-compiler 0.9.13`

Maven Dependency:

```
<dependency>
  <groupId>net.jaardvark</groupId>
  <artifactId>magnolia-module-resources-scss</artifactId>
  <version>1.0.3-SNAPSHOT</version>
</dependency>
```

Getting the source:

```
git clone https://runger@git.magnolia-cms.com/scm/forge/magnolia-module-resources-scss.git
```

Using SCSS in magnolia

After installing the SCSS module you will be able to create two new kinds of resources in the resources workspace:

- SCSS - for plain SCSS stylesheets.
- Processed SCSS - for SCSS stylesheets which additionally contain freemarker code.

Create resources of the appropriate type and fill them with your SCSS code.

When you request an SCSS resource, the SCSS compiler compiles your SCSS into CSS on the fly. It will process any imports, resolving them against magnolia's resources workspace. Errors will be reported to magnolia's log files on author and public instances and additionally as a comment in the CSS output on author instances.

When using "Processed SCSS" stylesheets, the freemarker processing happens before the SCSS processing.

> ⚠ Be aware that the SCSS resources are subject to the same caching (magnolia's cache, resources "far future" caching and browser cache) as other resources in magnolia, so make sure your caches are flushed if things aren't changing after you modify the resources.

Limitations

The SCSS module does not include an SCSS compiler. Instead, it uses the SCSS compiler provided by the Vaadin UI framework included with magnolia. Therefore the SCSS module is limited to supporting those aspects of SCSS supported by Vaadin's compiler. At the moment the support is rather poor - many things are not implemented. However, the most important and commonly used features of SCSS are working (mixins, nested rules, variables), so you can already benefit a lot from the current level of functionality. Also, the compiler may improve in the future, in which case magnolia's support for SCSS will improve with it.

License

Use at your own risk, your milage may vary.

Provided under a BSD License, with the express invitation to use, fork or otherwise change for your own purposes. Please consider contributing in the form of code, documentation, bug reports or feedback.

License

```
Copyright (c) 2014, Richard Unger
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those
of the authors and should not be interpreted as representing official policies,
either expressed or implied, of Magnolia Inc.
```

Technical Details

SCSS is compiled using Vaadin's Stylesheet compiler for the moment, and is therefore subject to the limitation of this compiler.

Processed resources under magnolia 5.4

Magnolia 5.4 introduced a new resources module with radically re-worked resources architecture under the hood. Resource loading was unified so that resources from class path, file system and JCR can all be handled under a unified API. Nice in theory, in practice it isn't quite ready for action.

For our sassy resources, there is an important feature missing in 5.4: the concept of processed resources, or in fact any notion at all of "rendering" resources before delivering them out to the browser. This was completely dropped in 5.4, and presents a problem for modules like the SCSS module, where the resource output is "rendered" (compiled) from a raw description. The *resource file* just contains the raw description (SCSS code), but the *resourc e delivered to the browser* has to be CSS (compiled SCSS).

In addition, the new resources module now delivers any resource on the filesystem (in the resources folder), the JCR resources workspace or the class path (!!) without checking any permissions or applying any security. This represents (IMHO) a serious regression and security problem.

The SCSS module corrects both these problems. It introduces a replacement for the ResourcesServlet (`net.jaardvark.magnolia.resources.RenderingResourcesServlet`) which uses configurable renderers to output the resources. The renderers can be as simple as just copying the resource contents to the output (BinaryResourceRenderer, TextResourceRenderer), or more complex, like the SCSSResourceRenderer.

The configuration for the resource rendering (to be found under `/modules/resources-scss/config/renderers`) uses voters, **and is by default configured to a white-listing system which renders only resource files having certain extensions**. You can use the classes `net.jaardvark.magnolia.resources.voters.ResourceOriginVoter`, `net.jaardvark.magnolia.resources.voters.ResourcePathVoter` or other voter classes to configure more complex setups if you wish.

The following renderers are available:

- `net.jaardvark.magnolia.resources.Error403ResourceRenderer`
- `net.jaardvark.magnolia.resources.Error404ResourceRenderer`
- `net.jaardvark.magnolia.resources.BinaryResourceRenderer`
- `net.jaardvark.magnolia.resources.TextResourceRenderer`
- `net.jaardvark.magnolia.resources.scss.SCSSResourceRenderer`

In addition, the SCSS module re-introduces the processed resources feature (resources using freemarker code) familiar from earlier versions of magnolia. TextResourceRenderer (or its subclasses like SCSSResourceRenderer) supports a "freemarker" configuration property. If set to true, the renderer will check the resource for freemarker code, and process freemarker if present. To check for freemarker code, the module checks for the presence of a `[#ftl]` (or `<#ftl>`) tag at the beginning of the resource file. If found, freemarker is processed before rendering the resource.