



News
License
Downloads
Documentation
Mailing Lists
Contributing
Site Map
ASF
Security
Sponsorship
Sponsors



[Home](#)

Documentation

In an effort to make it easier to find desired documentation, we have divided our documentation section into the following six areas:

- [Getting started](#) - This area captures a few links from the areas below that will help you get started with Felix.
- [FAQs](#) - This area captures all FAQ documentation, which typically varies from subproject to subproject.
- [Community](#) - This area captures documentation associated with how the Felix community works and how to become involved in it.
- [Development](#) - This area captures documentation for Felix developers or those interested in becoming Felix developers; this is not intended for Felix users although some information may be useful.
- [Subprojects](#) - This area captures user documentation for the various Felix subprojects.
- [Tutorials, examples, and presentations](#) - This area captures general user documentation that does not necessarily fit into any single subproject.

In addition a [site map](#) is available as a table of contents of the site.

If you are unable to find the documentation you need, please ask on the [mailing lists](#). Also, feedback on improving the documentation and/or organization of this site is welcome.

The Felix web site and documentation are managed with the [Apache CMS](#). For Apache Felix specific extensions see the [Site How-To](#) section below.

Books

The most important "books" on OSGi are of course the specifications themselves. They are quite a good read but are also complete and contain stuff, you might be interested in. So for a starter you might interested to read the following:

- Core Specification (Layer, Services, LiveCycle part)
- Compendium Specification
- Enterprise Specification

Visit the [OSGi Alliance Specifications](#) page to download the specifications for free.

Apart from the specifications a number of books have recently been published. They are listed here in no particular order:

- [OSGi in Action](#); Richard Hall, Karl Pauls, Stuart McCulloch, David Savage; Manning Publications, 2011. Covers most (if not all) of R 4.2 specs and also contains a lot of side information. The authors of this book are committers and/or PMC members of the Apache Felix project.

- [OSGi and Equinox: creating highly modular Java systems](#); Jeff McAffer, Paul VanderLei, Simon Archer; Addison-Wesley, 2010. Despite its dependency on Equinox it might be helpful, because they use Declarative Services intensely for their sample application throughout the book and we use Declarative Services intensely, too.
- [OSGi in Depth](#); Alexandre de Castro Alves; Manning, 2011. I cannot say anything on this book except that he seems to have used Apache Felix for the samples....
- [Enterprise OSGi in Action: With examples using Apache Aries](#); Holly Cummins, Timothy Ward; Manning, 2013 Covers Enterprise OSGi specifications, which are becoming more and more important.
- [Java Application Architecture: Modularity Patterns with Examples Using OSGi \(Agile Software Development Series\)](#); Kirk Knoernschild; Prentice Hall, 2012 Primarily covers Java Application Architecture using OSGi for the framework to implement.
- [OSGi in Practice](#); Neil Bartlett; Free PDF EBook. Many OSGi core concepts are greatly explained, and its free;
- [Instant OSGi Starter](#); Jamie Goodyear, Johan Edstrom; Packt, 2013
- [OSGi and Apache Felix 3.0 Beginner's Guide](#); Walid Joseph Gédéon; Packt, 2010
- [Spring Dynamic Modules in Action](#); Arnaud Cogoluègnes, Thierry Templier, Andy Piper; Manning, 2010
- [Building Modular Cloud Apps with OSGi](#); Paul Bakker, Bert Ertman; O'Reilly, 2013. Uses Apache Felix. For experienced Java developers in the enterprise, this practical, hands-on book shows you how to use OSGi to design, develop, and deploy modular cloud applications.

Site How-To

The site is managed with the [Apache CMS](#) where the source is kept in SVN at <https://svn.apache.org/repos/asf/felix/site/trunk/content>.

To update the documentation using the CMS system

- Install the bookmarklet from the [cms](#) page. You only have to do this once.
- Navigate to the page you wish to edit (on the live site, not in the cms).
- Click the bookmarklet. There will be a short pause while the CMS system is initialised for you.
- Click on Edit (to skip this step hack the bookmarklet to add an 'action=edit' param to the query string)
- The page editor should then be displayed.
- Click Submit to save your edit to the workarea
- Click Commit to save the updated file to SVN and trigger a staged build. (to skip this step click on the "Quick Commit" checkbox in the Edit form).
- The results should appear shortly on the [staging](#) site. (You may have to force the page to refresh in order to see the updated content)
- Once you are happy with the updated page, click on Publish Site to deploy.

There is also a [Reference Manual](#) of the Apache CMS as well as a video tutorial at <http://s.apache.org/cms-tutorial>.

Features of the Apache Felix Site

This section lists some Apache Felix features to help with the maintenance of the site, such as automatic link generation.

Start the file with a Title: line to define the page title and the first H1 tag:

```
Title: Page Title
```

```
Here comes the content separated with a blank line from the header ...
```

The last modification information from SVN (revision, committer, and date/time) is automatically added when the page is rendered

Excerpts can be added to a page using the Excerpt : header:

```
Title: Page Title
```

```
Excerpt: Summary of the page for inclusion in other pages;  
        continuation of the excerpt must be indented
```

```
Here comes the content separated with a blank line from the header ...
```

Metadata from child pages can be referred to in the content with the Django variable reference notation using the child page name (without extension) as its container; e.g. for the child page named childpage:

```
{{ children.childpage.headers.excerpt }}  
{{ children.childpage.headers.title }}
```

Content Pages can contain Django templates of the form `{{ ... }}` and `{% ... %}`. If so, the page content is evaluated as a Django template before running it through the page template.

Any page in the site can be referenced with `refs.pagename` returning properties:

```
.path  
    the absolute path of the page on the site  
.headers  
    page headers (e.g. .title, .excerpt)  
.content  
    the raw page content
```

All pages in the children namespace are also available in the refs namespace

Some usefull hints:

Printing title of another page "handler":

```
:::django  
{{ refs.handler.headers.title }}
```

Printing excerpt of another page "handler":

```
:::django  
{{ refs.handler.headers.excerpt }}
```

Linking to another page "handler":

```
:::django  
({{ refs.handler.path }})
```

Printing title as a link to another page "handler":


```
:::django
[{{ refs.handler.headers.title }}]({{ refs.handler.path }})
```

Printing excerpt as a link to another page "handler":

```
:::django
[{{ refs.handler.headers.excerpt }}]({{ refs.handler.path }})
```

Print a bullet pointed child page list:

```
:::django
{% for label, page in children %}* [{{ page.headers.title }}](
{% endfor %}
```

 It is important to have the first part as a single line, otherwise the Django/Markdown combo will create a list for each entry.

Code Highlighting

Code Highlighting works by indenting code by four blanks. To indicate the type of highlighting precede the code style text with either `:::<lexer>` to get high lighted code using the given `<lexer>` or `#!<lexer>` to get high lighted code with line numbers using the given `<lexer>`. See <http://www.apache.org/dev/cmsref.html#code-highlighter> for main info and <http://pygments.org/docs/lexers/> for supported lexers

HTML and Markdown

Markdown supports embedding HTML. But be aware that contents of HTML elements are not further handled by the Markdown converter. Thus it is not possible to embed Markdown syntax inside of HTML elements to have them converted.

Manual Generation

When committing changes to pages into SVN the pages are automatically generated in [the staging site](#).



To generate the site locally, you must have checked out the complete SVN to access the tools:

```
$ svn https://svn.apache.org/repos/asf/felix/site/ felix-site
```

To manually generate the site or single pages the [site](#) can be checked out from SVN. In addition Perl and Python must be installed for the build tools to work.

To prepare for site build, the Markdown daemon has to be started:

```
$ export MARKDOWN_SOCKET="$PWD/tools/build/./markdown.socket"
$ export PYTHONPATH="$PWD/tools/build"
$ python "$PWD/tools/build/markdown.py"
```

The MARKDOWN_SOCKET environment variables is also required by the `build_site.pl` and `build_file.pl` scripts to connect to the Markdown daemon.

To build the complete site use the `build_site.pl` script:


```
$ tools/build/build_site.pl --source-base $PWD/trunk \
--target-base $PWD/trunk/target
```

To build a single page use the `build_file.pl` script:

```
$ tools/build/build_site.pl --source-base $PWD/trunk \
--target-base $PWD/trunk/target \
--source content/documentation.mdtext
```

The argument to the `--source` parameter is relative to the `--source-base` folder.

Configuring site generation on Mac

 Those instructions were computed on Mountain Lion.

A couple of Python and Perl libraries are required and need to be installed

```
$ sudo easy_install Pygments
$ sudo easy_install Markdown
```

And for the Perl modules:

```
$ sudo cpan install XML::Atom::Feed
$ sudo cpan install XML::RSS::Parser
$ sudo cpan install XML::Parser::Lite
$ sudo cpan install XML::RSS::Parser::Lite
$ sudo cpan install Net::Twitter
$ sudo cpan install SVN::Client
```

Be careful that some of those commands require time... Once done, launch the markdown daemon with the following command from the SVN root:

```
$ export MARKDOWN_SOCKET="$PWD/tools/build/./markdown.socket"
$ export PYTHONPATH="$PWD/tools/build"
$ python "$PWD/tools/build/markdown.py"
```

And finally, generate the web site from the svn root with:

```
tools/build/build_site.pl --source-base $PWD/trunk --target-t
```

Rev. 1700393 by cziegeler on Tue, 1 Sep 2015 06:04:06 +0000

Apache Felix, Felix, Apache, the Apache feather logo, and the Apache Felix project logo are trademarks of The Apache Software Foundation. All other marks mentioned may be trademarks or registered trademarks of their respective owners.