How to use the JBoss DocBook system

A guide for content developers

ISBN:
Publication date:

How to use the JBoss DocBook system: A guide for content developers

Target Audience

All JBoss.org documentation content developers and style developers should read this document. You should also read this if you want to build JBoss.org DocBooks from the *public SVN repositories* [http://anonsvn.jboss.org].

Preface

This document introduces the unified DocBook system for JBoss.org documentation based on Maven2. Using this system provides the following benefits:

- The DocBook libraries and all other dependencies are stored in Maven plugins located in the *JBoss public maven repository* [http://repository.jboss.org/maven2]. This allows the entire build environment to be automatically downloaded and configured from just a single `pom.xml` file stored in a project's `docs` directory.
- Styles are also stored in Maven plugins so that they can be managed from a central location. This allows updates to be automatically downloaded when new designs are created or bugs are fixed.
- It provides a default style based on the JBoss.org community-driven theme that looks clean, fresh and modern.
- Custom styles can easily be created for individual projects, or groups of related projects, to cater for different requirements. For example some projects may have their own logos or wish to have collapsable menus at the start of each chapter.
- The build process is simplified and standardized. Just follow the instructions in this guide to setup your `docs` directory and copy a very simple `pom.xml` file.

If you have any questions, please feel free to contact *Mark Newton* [mailto:mark.newton@jboss.org] (Content Lead) for more information.

Introduction to DocBook processing

DocBook is an XML format for writing documents. It allows the author to focus on the content itself during the writing process instead of worrying about the presentation.

Using standard DocBook tags, we can tag the content according to its syntatic structure. The DocBook document is then processed using XSL stylesheets so that each tagged DocBook element is transformed to a corresponding element in the target output format. For example each <para></para> element in DocBook could be transformed into a element in XHTML.

Using different XSL stylesheets, we can generate different output formats. For example, we can generate both XHTML and PDF outputs from a single DocBook source. We can also generate multiple versions of XHTML (or PDF) files each with a different style if necessary.

In the JBoss DocBook system, we provide XSL stylesheets to build XHTML, PDF and Eclipse Help output formats from the DocBook source. The build process is illustrated in *Figure 1.1, “The DocBook build process ”*.

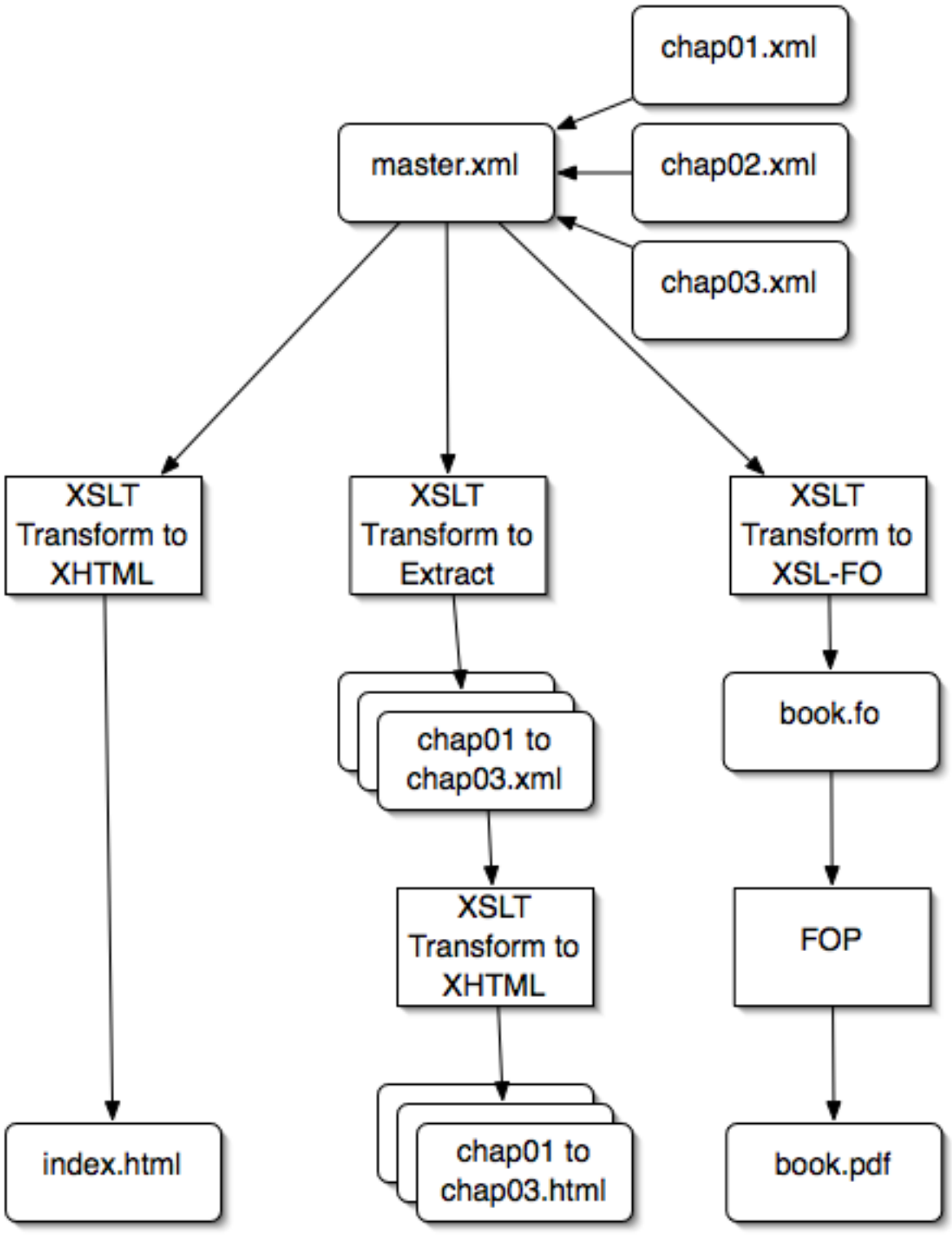

Figure 1.1. The DocBook build process

XHTML is used instead of HTML as it ensures that the content is completely separated from its style using Cascading Style Sheets (CSS) and image files.

How to develop content for JBoss DocBook

1. Install Maven2

Before you can start to build DocBook documents you first need to install and configure *Maven2* [http://maven.apache.org/]. At the time of writing Maven 2.0.8 is the latest released version that has been tested and found to work correctly. System requirements and download instructions can be found *here* [http://maven.apache.org/download.html].

In order for Maven to reference the JBoss public maven repository you will need to create a `~/.m2/settings.xml` file with the following contents:

```
<settings>
  <profiles>
    <profile>
      <id>jboss.repository</id>
      <activation>
        <property>
          <name>!jboss.repository.off</name>
        </property>
      </activation>
      <repositories>
        <repository>
          <id>snapshots.jboss.org</id>
          <url>http://snapshots.jboss.org/maven2</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
        <repository>
          <id>repository.jboss.org</id>
          <url>http://repository.jboss.org/maven2</url>
          <snapshots>
            <enabled>false</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>repository.jboss.org</id>
          <url>http://repository.jboss.org/maven2</url>
          <snapshots>
            <enabled>false</enabled>
```

```
          </snapshots>
        </pluginRepository>
        <pluginRepository>
          <id>snapshots.jboss.org</id>
          <url>http://snapshots.jboss.org/maven2</url>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
</settings>
```

If you already have a `~/.m2/settings.xml` file then you simply need to add the above `<profile>` element to your `<profiles>`.

2. Setup the directories

Each JBoss.org project stores its documentation in a `docs` directory, typically located at `<projectName>/trunk/docs`. By convention the source for each DocBook should be placed into subdirectories directly beneath this. For example, the DocBook source for a project's User Guide should be placed in `<projectName>/trunk/docs/userguide` whilst the DocBook source for a Reference Guide should be placed in `<projectName>/trunk/docs/reference`. Alongside these DocBook subdirectories any number of other subdirectories containing API references, code examples, and articles may also be created.

For example the docs directory for the JBoss Messaging project is shown in *Figure 2.1, “The JBoss Messaging docs directory”*.

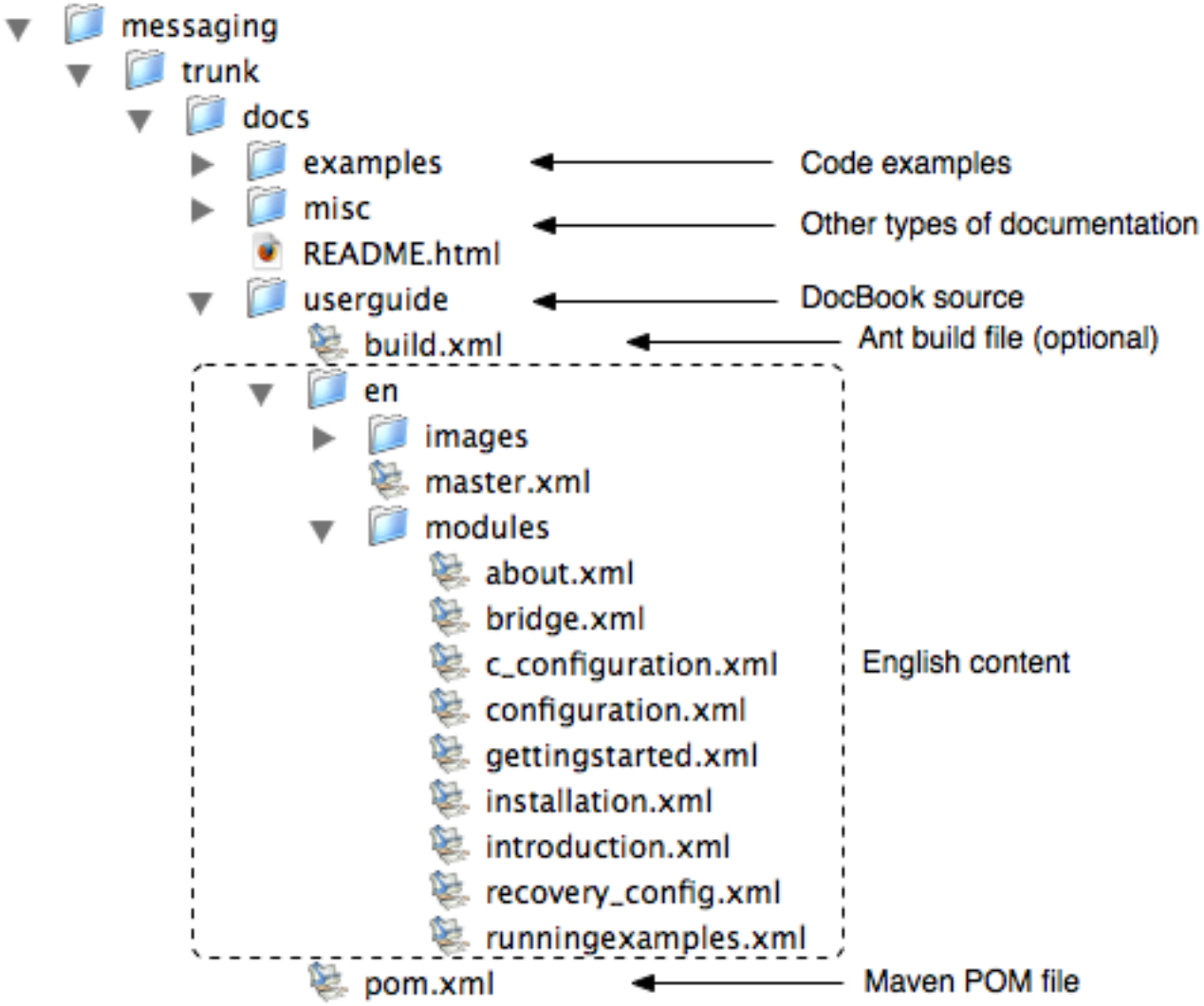

Figure 2.1. The JBoss Messaging docs directory

To facilitate translation into other languages the original source of a DocBook is written in English and located in a directory named `en`. Within this there are typically two subdirectories for the content:

- `images` - stores images
- `modules` - stores DocBook text modules, usually one per chapter

The `master.xml` file contains the top-level `<book>` element and is used to group all of the other text modules together. If you choose to use Ant to perform the build then a `build.xml` file is used to call the relevant tasks. Otherwise a `pom.xml` file can be called using Maven2.

The easiest way to setup a DocBook subdirectory within your project is to copy the *`docbook-support/docs/guide`*

[http://anonsvn.jboss.org/repos/jbossas/trunk/docbook-support/docs/guide/] directory (containing the DocBook source for this guide) and use it as a template.

3. Author the content

Now you can write your content in DocBook format. Make sure that the master file of your DocBook (i.e. the file that contains the <book> element) is called master.xml and lives directly under the en directory (see *Figure 2.1, "The JBoss Messaging docs directory"*). You can either put the entire content in master.xml or divide up the chapters and place them in the modules/ directory. If you do the latter, you should reference the chapter files from the master.xml file via entity references. Please see the *docbook-support/docs/guide/en/master.xml* [http://anonsvn.jboss.org/repos/jbossas/trunk/docbook-support/docs/guide/en/master.xml] file to see how this is done.

4. Build the documents

To perform a build using Maven you simply need to enter the following commands from the directory containing the pom.xml file:

- mvn compile - processes the DocBook source using the XSL stylesheets
- mvn package - creates WAR archives containing the various output formats
- mvn clean - removes the target directory containing the build output

It is also possible to chain commands together if required. e.g. mvn clean compile

Once the compile or package goals have been reached then a target/ directory will be created containing the build output.

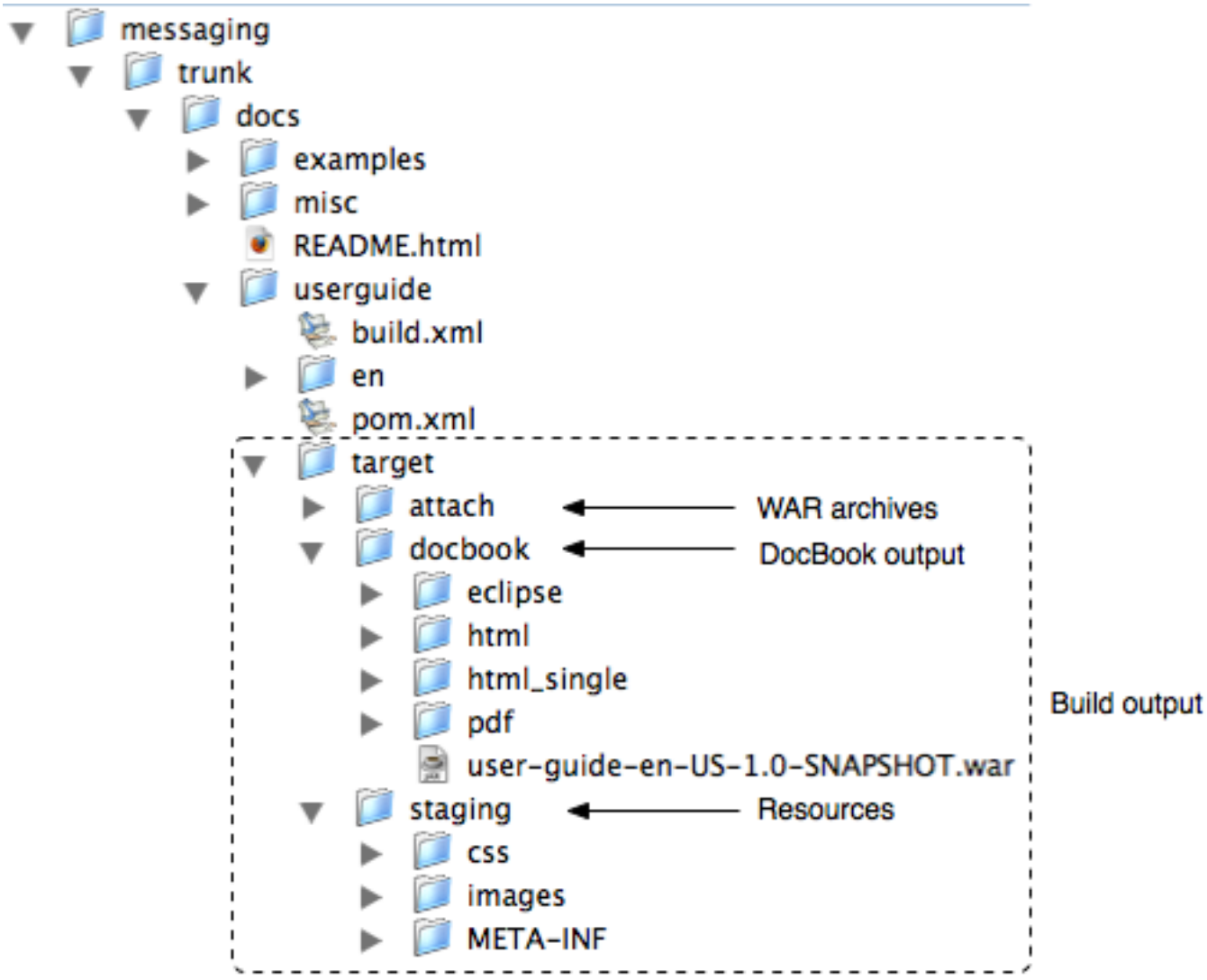

Figure 2.2. The JBoss Messaging target directory

The `attach` directory contains WAR archives that package together all of the files for the various output formats. The intention is that these can be easily deployed into a web server such as Tomcat or JBoss AS.

The `docbook` directory contains a number of subdirectories; one per output format. This is where you will find the XHTML, PDF and Eclipse Help files.

The `staging` directory contains resources such as CSS files, images and fonts that are needed for the final output. They need to be placed into this directory during the build as the DocBook XSL stylesheets can only reference a single location for images. Similarly the FOP libraries can only reference a single location for fonts.

5. Migrating from Ant

Previously the JBoss DocBook system used Ant to perform builds. This meant that each DocBook subdirectory contained a `build.xml` file as follows:

```
<project name="Documentation" default="all.doc" basedir=".">

    <property name="pdf.name" value="jboss-mybook.pdf" />
    <import file="../../../docbook-support/support.xml" />

    <target name="all.doc" depends="clean">
        <antcall target="lang.all">
            <param name="lang" value="en"/>
        </antcall>
    </target>

</project>
```

The idea was for this file to delegate most of the work to a `support.xml` file maintained by the *`docbook-support`* `[http://anonsvn.jboss.org/repos/jbossas/trunk/docbook-support/]` project. Developers were supposed to check out the `docbook-support` project alongside their own project in order to reference this file together with the associated libraries and XSL stylesheets. However what usually happened was that the contents of the `docbook-support` project were copied into each project's SVN tree. This meant that projects ended up having to maintain their own DocBook build systems and any changes made to the `docbook-support` project would have to be manually synchronised with their own copy.

With the new Maven2 based system this is no longer an issue as Maven takes care of downloading dependencies automatically. As a result all of the build environment can now be centrally maintained by the Content Team as a set of Maven plugins. You are free to continue using Ant if you prefer but since you won't be using the Maven plugins you will have to maintain any DocBook libraries, XSL stylesheets and CSS files copied or referenced from the `docbook-support` project yourself.

To perform the build using Ant simply enter the `ant` command from the directory containing the `build.xml` file. After it is finished, you should have three output documents in the following places:

- The `build/en/html` directory contains the HTML version of the document. Each chapter is broken into a separate HTML file and they are linked together by an `index.html` file.

- The `build/en/html_single` directory contains a single `index.html` file which holds the entire document.
- The `build/en/pdf` directory contains the PDF version of the document.

Note

The HTML output created by Ant is not XHTML. If you wish to gaurantee the separation of content and style then Maven2 should be used instead to produce strict XHTML.

If you wish to migrate from Ant to Maven then all you need to do is copy the *`docbook-support/docs/guide/pom.xml`* `[http://anonsvn.jboss.org/repos/jbossas/trunk/docbook-support/docs/guide/pom.xml]` file to your own DocBook subdirectory (alongside the `build.xml` file) and then follow the instructions in the previous section. You can then delete the `build.xml` file as it's not longer needed.

Maintaining the JBoss DocBook system

The structure of the *docbook-support* [http://anonsvn.jboss.org/repos/jbossas/trunk/docbook-support/] project is illustrated in *Figure 3.1, "The docbook-support project "*. The contents are as follows:

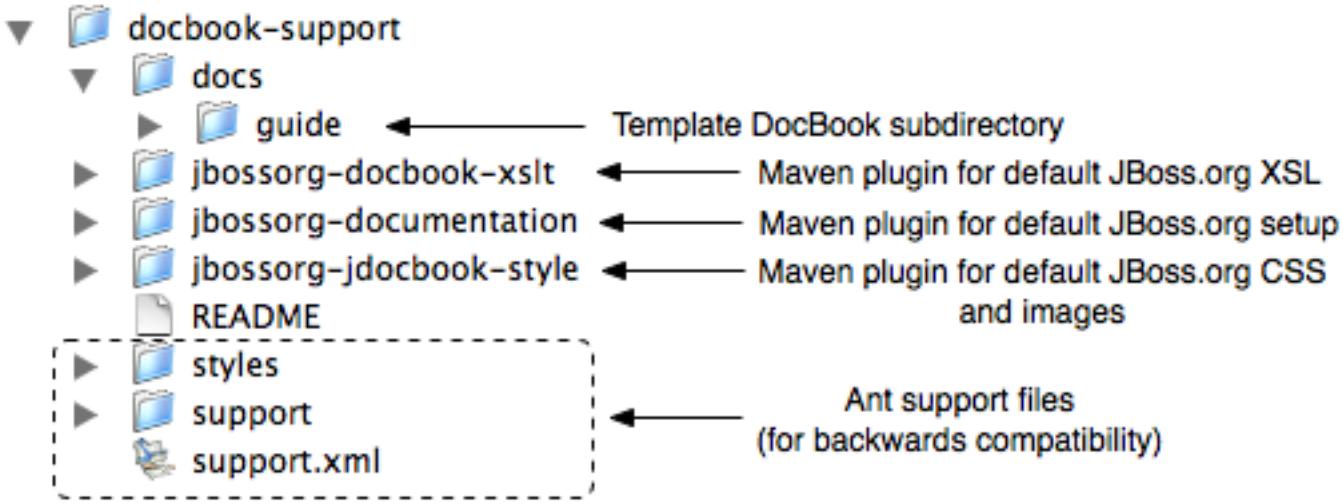

Figure 3.1. The docbook-support project

- The `docs` directory contains the DocBook source for this guide to serve as a template for other projects.
- The `jbossorg-documentation` directory contains the maven plugin source for the parent POM that is referenced by each project's `pom.xml` file. The parent POM contains common configuration information so that it does not have to be duplicated across multiple JBoss.org projects.
- The `jbossorg-docbook-xslt` directory contains the maven plugin source for the default JBoss.org XSL stylesheets. These stylesheets produce XHTML, PDF and Eclipse Help output.
- The `jbossorg-jdocbook-style` directory contains the maven plugin source for the default JBoss.org CSS and image files. These provide the JBoss.org community-driven look & feel.
- The `styles` and `support` directories together with the `support.xml` file are provided for backwards compatibility with projects that wish to continue using Ant instead of Maven. The `styles` directory contains XSL stylesheets together with

CSS files to produce HTML and PDF outputs. The `support` directory contains the DocBook libraries and DTDs along with the standard XSL stylesheets. The `support.xml` file contains common Ant tasks to perform the build.