
Table of Contents

Preface	v
1. Introduction	1
Document syntax	1
Wiki macros	1
2. How-to	2
Document structure	2
Document content	2
Rich text	2
Verbatim text	3
Blocks	3
Definition list	8
Modularization	8
3. Project integration	9
Javatm code embedding	9
Producing a code catalog	9
Maven plugins	9
The Maven Wikbook plugin	9
The Maven jdocbook plugin	9

List of Figures

2.1. The controller 6

List of Tables

2.1.	2
2.2. Text emphasis	2
2.3. Document links	3
2.4. External links	3
2.5. Verbatim text	3
2.6. Paragraph generation	3
2.7. List examples	4
2.8. Table examples	5
2.9. A simple table	5
2.10. A table with a row header	5
2.11. A table with a row footer	5
2.12. The table	5
2.13. A list inside a table	5
2.14.	5
2.15. Image example	6
2.16. Admonitions	6
2.17. A simple code example	6
2.18. Simple XML validation	7
2.19. XML pretty printed	7
2.20. A screen example	7
2.21. A simple callout	7
2.22. A callout anchor and its text	8
2.23. Definition lists	8

List of Examples

2.1. Document embedding	8
3.1. The Wikbook dependency for producing a code catalog	9

Preface

Wikbook is a set of tools for efficiently writing docbook documentation. For this purpose several choices have been made that are explained below

- The Docbook system can produce high quality documents with different outputs.
- Documentation written in wiki language is a good trade off between document richness and simplicity

Wikbook aims also to provide exclusive features that makes documentation easier to write. The perfect example we like to give is the inclusion of Java[™] source code at the heart of the documentation as it allows to have always up to date example that stay in sync with the documentation. The goal is to make the documentation easier to write and maintain.

Wikbook is also a lab to experiment new things and we are sure that it will get richer over time. We hope that people will find Wikbook interesting to write their documentation, experiment new ideas and contribute them.

Wikbook aims to be open and does not want to lock users into a proprietary system. The proof is that Wikbook is based on the open Docbook format.

Important

This document is a work in progress and will get richer over time.

Chapter 1. Introduction

Document syntax

Wiki syntaxes are not unique, Wikbook relies on an external framework used at the hear of XWiki that is able to make wiki syntaxes coexist. Any wiki syntax recognized by this framework can be used for documentation writing. The following syntaxes are recognized:

- XWiki 1.0 and 2.0 [<http://platform.xwiki.org/xwiki/bin/view/Main/XWikiSyntax>]
- Creole [<http://www.wikicreole.org/wiki/Creole1.0>]
- JSP Wiki [<http://www.jspwiki.org/wiki/TextFormattingRules>]
- Media Wiki [http://ang.wikipedia.org/wiki/Wikipedia:How_to_edit_a_page#The_wiki_markup]
- Confluence [<http://www.atlassian.com/software/confluence/>]
- TWiki [<http://twiki.org/cgi-bin/view/TWiki/TextFormattingFAQ>]

For the purpose of simplicity, the Wikbook documentation uses the XWiki 2.0 syntax.

Wikbook attempts to fully leverage the wiki syntax, however the horizontal rule wiki syntax is not supported. Indeed there is no such concept in the docbook system and horizontal rule are simply ignored.

Wiki macros

A wiki macro is way to complete the wiki syntax. Wiki syntax is usually not enough to express some ideas and a wiki macro provides a way to plug new behavior into a wiki syntax.

Wikbook makes use of wiki macro to integrate specific docbook features.

Chapter 2. How-to

Wiki concepts are naturally mapped onto docbook concepts.

Document structure

A book is structured into chapters and each chapter is structured into sections. A wiki document provides a syntax for creating nested sections with different levels. The book structure can naturally leverage the wiki document structure with the following rules:

- Top level section correspond to book chapters
- Other wiki sections correspond to book sections

Table 2.1.

<code>= Section 1 =</code> <code>== Section 2 ==</code>	<code><chapter></code> <code><title>Section 1</title></code> <code><section></code> <code><title>Section 2</title></code> <code></section></code> <code></chapter></code>
--	--

Note

All wiki sections don't have to be explicitly declared, it is possible to *omit* the declaration of a section. When it occurs, implicit docbook section will be created with no title.

Document content

The content of a book can be seen as a collection of content, each content can be categorized into blocks or rich text. Rich text can either be simple text or it can be made richer, like a word in bold. A block contains usually rich text and gives a meaning to the text, the most natural block is the paragraph.

Rich text

Emphasis

The following syntax can be used to put the emphasis for creating richtext

Table 2.2. Text emphasis

<code>**bold**</code>	bold
<code>//italic//</code>	<i>italic</i>
<code>__underline__</code>	<u>underline</u>
<code>.,subscript.,</code>	_{subscript}
<code>^^superscript^^</code>	^{superscript}
<code>--striketrough--</code>	striketrough
<code>##monospaced##</code>	monospaced

<code>{{code}}inlineCode(){{/code}}</code>	<code>inlineCode()</code>
--	---------------------------

Links

We can distinguish two kinds of links. A link can reference a target inside the document such as an anchor or a section or it can reference an URL.

The anchor macro plays an important role in internal links as it specifies the potential target of a link. Any internal link should reference a valid anchor inside the document. An anchor can be placed anywhere in text but it can also be present in a section title.

Table 2.3. Document links

<code>= Chapter 1 {{anchor id=chapter_1 /}}chapter></code>	<code><chapter></code>
<code>A [[link>>#chapter_1]] to the chapter</code>	<code><title>Chapter 1</title></code>
<code>The [[#chapter_1]] can be linked</code>	<code><para>A <link linkend="chapter_1">link</link></code>
<code>A [[link>>#foo]] to an anchor {{anchor id=foo /}}The {{ref linkend="chapter_1"/> can be</code>	<code><para>A <link linkend="chapter_1">link</link> to an</code>
	<code><para>A <link linkend="foo">link</link> to an</code>
	<code></chapter></code>

Table 2.4. External links

The http://www.foobar.com site
The FooBar [http://www.foobar.com] site
Send a mail to <foo@bar.com>

Verbatim text

Verbatim escapes the wiki syntax and is useful for citing wiki in a wiki document. It is useful for creating document such as this documentation.

Table 2.5. Verbatim text

<code>{{[foo>>#bar]}}</code>	<code><para>[foo&gt ;&gt ;#bar]</para></code>
------------------------------------	---

Blocks

Paragraphs

Unlike docbook, wiki paragraphs are implicitly defined, the general rule is that any text content that does not contain wiki instruction is one paragraph. The wikbook system creates docbook paragraphs when it is required. The simplest example is that any content in a section is a paragraph.

Table 2.6. Paragraph generation

<code>= Chapter 1 =</code>	<code><chapter></code>
<code>The paragraph of the chapter 1.</code>	<code><title>Chapter 1</title></code>
<code>== Section 1 ==</code>	<code><para>The paragraph of the chapter 1.</para></code>
<code>The paragraph of the section 1.</code>	<code><section></code>
<code>== Section 2 ==</code>	<code><title>Section 1</title></code>
<code>The first paragraph of the section 2.</code>	<code><para>The paragraph of the section 1.</para></code>
	<code></section></code>
<code>The second paragraph of the section 2</code>	<code><section></code>


```

<title>Section 2</title>
<para>The first paragraph of the section 2.
<para>The second paragraph of the section 2
</section>
</chapter>

```

Lists

It is easy to create lists in wiki syntax, whereas the docbook XML is very tedious. Several types of lists are possible such as bullet or ordered list.

Table 2.7. List examples

<pre> * item 1 ** item 1.1 ** item 1.2 * item 2 </pre>	<pre> • item 1 • • item 1.1 • item 1.2 • item 2 </pre>
<pre> 1. item 1 11. item 1.1 11. item 1.2 1. item 2 </pre>	<pre> 1. item 1 2. a. item 1.1 b. item 1.2 3. item 2 </pre>
<pre> (% style="upperroman" %) 1. item 1 11. item 1.1 11. item 1.2 1. item 2 </pre>	<pre> I. item 1 II. 1. item 1.1 2. item 1.2 III.item 2 </pre>

It is possible to configure also the list style according to the docbook capabilities.

- Bullet style
 - • disc
 - circle
 - square
- Numbering style
 - • arabic
 - loweralpha
 - lowerroman
 - upperalpha
 - upperroman

- arabicindic

Tables

Tables are mapped to the docbook tables, here are a few examples

Table 2.8. Table examples

<div><div> 1 2 3</div><div> 4 5 6</div></div>	<div>Table 2.9. A simple table</div> <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr></table>	1	2	3	4	5	6
1	2	3					
4	5	6					
<div><div> =1 =2 =3</div><div> 4 5 6</div></div>	<div>Table 2.10. A table with a row header</div> <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr></table>	1	2	3	4	5	6
1	2	3					
4	5	6					
<div><div> 1 2 3</div><div> =4 =5 =6</div></div>	<div>Table 2.11. A table with a row footer</div> <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr></table>	1	2	3	4	5	6
1	2	3					
4	5	6					
<div><div>(% title="The table" %)</div><div> 1 2 3</div><div> 4 5 6</div></div>	<div>Table 2.12. The table</div> <table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr></table>	1	2	3	4	5	6
1	2	3					
4	5	6					

By default a table expects inline content, that means that any content inside the table will not be interpreted as wiki text but as normal text. It is possible to change that behavior with the usage of the `((...))` block allowing the usage of nested tables or lists.

Table 2.13. A list inside a table

<pre> (((* item 1 * item 2)))</pre>	<table><tr><td>Table 2.14.</td></tr><tr><td><ul style="list-style-type: none">• item 1• item 2</td></tr></table>	Table 2.14.	<ul style="list-style-type: none">• item 1• item 2
Table 2.14.			
<ul style="list-style-type: none">• item 1• item 2			

Note

This document makes an extensive usage of this feature.

Warning

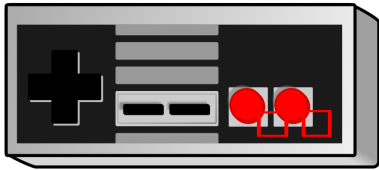
Inside a complex content block, the usage of structural elements such as section is not allowed.

Images

Images can be displayed a block elements. Image display can be parameterized for all output but it is possible to target a specific output with a prefix. The *fo* prefix affects the PDF output and the *html* targets

the HTML content. More details about docbook images parameterization can be found in the docbook imagedata [<http://www.docbook.org/tdg/en/html/imagedata.html>] reference.

Table 2.15. Image example

[[image:images/controller.png title="The controller" align="center" html:scale="5
<p>Figure 2.1. The controller</p> 

Admonitions

Table 2.16. Admonitions

Note	{{note}}Some noticeable text{{/note}}	Note some noticeable text
Warning	{{warning}}you should not do that{{/warning}}	Warning you should not do that
Tip	{{tip}}a usefull tip{{/tip}}	Tip a usefull tip
Caution	{{caution}}beware!!!{{/caution}}	Caution beware!!!
Important	{{important}}something important{{/important}}	Important something important

Special blocks

A set of special blocks are available, they allow to give a special representation to the emboddied text.

The *code* macro creates a docbook programlisting XML element to display anything related to code. Special features are available that makes it very powerful

- XML code can be validated and pretty printed with a configuration indentation
- External Java code can be embedded inside the document

Table 2.17. A simple code example

{{code}}x = x + 1{{/code}}
x = x + 1

Table 2.18. Simple XML validation

<code>{{code language=xml}}<valid/>{{/code}}</code>
<code><valid/></code>

Table 2.19. XML pretty printed

<code>{{code language=xml indent=2}}<valid><formatted/></valid>{{/code}}</code>
<pre> <valid> <formatted/> </valid> </pre>

The *screen* macro creates a docbook screen XML element to display anything related to the computer screen:

Table 2.20. A screen example

<pre> {{screen}} julien:core julien\$ ls -l total 24 -rw-r--r-- 1 julien staff 1878 Apr 26 15:08 pom.xml drwxr-xr-x 5 julien staff 170 Apr 26 15:08 src drwxr-xr-x 5 julien staff 170 Apr 26 18:46 target -rw-r--r-- 1 julien staff 4090 Apr 26 15:09 wikbook.core.iml {{/screen}} </pre>
<pre> julien:core julien\$ ls -l total 24 -rw-r--r-- 1 julien staff 1878 Apr 26 15:08 pom.xml drwxr-xr-x 5 julien staff 170 Apr 26 15:08 src drwxr-xr-x 5 julien staff 170 Apr 26 18:46 target -rw-r--r-- 1 julien staff 4090 Apr 26 15:09 wikbook.core.iml </pre>

Callouts

Callouts are useful for creating a set of references that refers to a block of code. At the moment they are only available for Java™ language. A callout is declared inside a code block using comments in special format.

Table 2.21. A simple callout

<pre> {{code language=java}}public void foo() { System.out.println("This is going to the output"); // <1> A callout } {{/code}} </pre>
<pre> public void foo() { System.out.println("This is going to the output"); 1 } </pre>
<p>1 A callout</p>

Callout anchor don't have to be mixed with the related text, a simple declaration without any text will just create an anchor. Somewhere else in the code base, the callout text can be declared by adding an equals

sign between the right angle bracket and the text. As a consequence, several lines can be referenced with the same callout.

Table 2.22. A callout anchor and its text

<pre> {{code language=java}}public void foo() { int a = 0; // <1> int b = 0; // <1> } // <1> = An assignment {{/code}}</pre>	<pre> public void foo() { int a = 0; 1 int b = 0; 2 } 12 An assignment</pre>
--	--

Definition list

Wiki definition lists are managed as docbook variable lists.

Table 2.23. Definition lists

<pre> ; term : definition</pre>	<pre> <variablelist> <varlistentry> <term>term</term> <listitem>definition</listitem> </varlistentry> </variablelist></pre>
<pre> (% title="the title" %) ; term : definition</pre>	<pre> <variablelist> <title>the title</title> <varlistentry> <term>term</term> <listitem>definition</listitem> </varlistentry> </variablelist></pre>

Modularization

The *include* macro is a good way to provide modularization of a complex document.

Example 2.1. Document embedding

```
{{include document="embedded.wiki" /}}
```

Important

Embedding does a special treatment to sections: when a document is embedded, its section are reconsidered with respect to the most inner section embedding the document.

Chapter 3. Project integration

This chapter focuses on the integration of the documentation with the project it documents.

Java[™] code embedding

Wikbook allows to embed Java[™] source code inside the documentation. It is very powerful when it comes to explain a tutorial or to explain an API.

Producing a code catalog

A code catalog is produced by a Java 6 compiler plugin that create source code fragments at the compilation. It is triggered automatically with the Java 6 compiler when the Wikbook APT jar is on the classpath and produce a code catalog within the jar file itself.

Example 3.1. The Wikbook dependency for producing a code catalog

```
<dependency>
  <groupId>org.wikbook</groupId>
  <artifactId>wikbook.apt</artifactId>
  <version>{{property.wikbook.version}}</version>
</dependency>
```

A code catalog is created from annotated Java[™] code when it is annotated with the `@org.wikbook.apt.annotation.Documented` annotation. The annotation can target:

- A class
- A method
- A constructor
- A field

todo

Maven plugins

The Maven Wikbook plugin converts wiki document to a Docbook document. In order to produce final consumable documents (PDF, HTML), the Docbook document must be converted to new documents, this process is usually done via an XSL stylesheet that takes the Docbook document and transforms it. The Maven jdocbook plugin is a nice integration of the XSL stylesheet transformation with Maven. We recommend to use the Maven Wikbook and jdocbook plugin altogether.

The Maven Wikbook plugin

The Maven Wikbook plugin.

The Maven jdocbook plugin

The Maven jdocbook plugin.