



Maven and docbook

1.0.0

Florent Ramière (Jaxio)

Copyright © 2005-2010

Copies of this document may be made for your own use and for distribution to others, provided that you do not charge any fee for such copies and further provided that each copy contains this Copyright Notice, whether distributed in print or electronically.

1. Introduction	1
1.1. Docbook and maven	1
1.2. Celerio and Springfuse	1
1.3. XSL syntax highlighting	1
1.4. Maven plugins	1
1.5. Looking around	1
1.6. Enter docbkx	2
1.7. Cocoon is my friend	2
1.8. There you have it	2
1.9. Examples	2
2. Syntax highlighting examples	4
2.1. Java example	4
2.2. Java with callouts example	4
2.3. XML	4
3. Docbook authoring for a java developer	5
3.1. Resources	5
3.2. command	5
3.3. screen	5
3.4. filename	5
3.5. classname	6
3.6. literals	6
3.7. itemizedlist	6
3.8. imageobject	6
3.9. Maven	7
4. Bugs	8
4.1. Regression	8
4.2. PDF XSL and image	8
5. Conclusion	9
5.1. Docbkx works great	9
5.2. Download maven project	9
5.3. Missing artifacts ?	9
5.3.1. Manual installation	9
5.3.2. Maven profile installation	9
5.4. Download PDF	10

Chapter 1. Introduction

1.1. Docbook and maven

I was looking for a good [docbook](#) support with [maven](#) .

It was hell.

1.2. Celerio and Springfuse

We write our documentation for [Springfuse](#) and [Celerio](#) using [Docbook](#) . We used to have an [ant](#) build to produce html, html-chunked and pdf out of it. We had an old XSL and had to do many xsl updates (and that was a reaaaaaaal pain)

We wanted to upgrade the xsl and use maven ... and we wanted to have syntax highlighting.

1.3. XSL syntax highlighting

Here at what I looked at for highlighting via xsl :

- <http://www.sagehill.net/docbookxsl/SyntaxHighlighting.html>
- <http://xmlguru.cz/2006/07/docbook-syntax-highlighting>
- <http://sourceforge.net/projects/xslth/>

1.4. Maven plugins

I looked at these maven plugins :

- <http://wiki.docbook.org/topic/Maven>
- <http://mojo.codehaus.org/docbook-maven-plugin/introduction.html>
- <http://maven-plugins.sourceforge.net/maven-sdocbook-plugin/index.html>

I could not find something was was suitable.

1.5. Looking around

As I could not find my way, I went to do something I do very frequently : look for what the best teams did. Look in spring code, in hibernate, in drools.

- The Spring folks does it the ant way : <https://src.springframework.org/svn/spring-build/trunk/project-build/docbook/>



Argh

No more ant for me :)

- The Jboss guys does it with maven using a specific plugin : <http://www.jboss.org/maven-jdocbook-plugin/>
- None handled syntax highlighting

1.6. Enter docbkx

Then I stumbled upon <http://code.google.com/p/docbkx-tools/> It looked like very much to what I needed.

I was up to speed very quickly with their documentation <http://docbkx-tools.sourceforge.net/docbkx-maven-plugin/plugin-info.html> and their [examples here](#)

... Until I tried to handle correctly [PDF](#) . I am not sure what I did wrong. But many stuff were missing.

By example when I imported my old xsl, and I had this kind of errors :

```
Embedded error: org.apache.fop.apps.FOException: null:1:28951: Error(1/28951):
No element mapping definition found for (Namespace URI:"http://xml.apache.org/fop/extensions", Local Name: "dest
```

I tried many alternatives, tried different FOP version, looked at bugs reports, source code.

I was lost, the html was okay, but pdf would not produce correct results (or no result at all)

1.7. Cocoon is my friend

While looking for many kind of errors on google, I stumbled on a commit from <http://cocoon.apache.org/> . I looked into their source at <http://svn.apache.org/repos/asf/cocoon/cocoon3/trunk/> I downloaded it, built it (thank you maven)

Woah : everything I needed was working smoothly !

1.8. There you have it

I created this blog entry, so do you not loose 2 or 3 days of your life having to deal with xsl, lib, fop version, weird errors, and frustration !

1.9. Examples

You can checkout this article in

- in multi page html <http://www.springfuse.com/blog/docbook/html/maven-docbook-syntax-highlighting.html>
- in single page html <http://www.springfuse.com/blog/docbook/html-single/maven-docbook-syntax-highlighting.html>

- in pdf <http://www.springfuse.com/blog/docbook/maven-docbook-syntax-highlighting.pdf>

Chapter 2. Syntax highlighting examples

2.1. Java example

Here is a java example with syntax highlighting

```
@Entity
@Table(name = "ATTACHMENT")
@Cache(usage=CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Attachment implements Comparable<Attachment>, Serializable {
    static final private long serialVersionUID = 1L;
    static final private Log logger = LoggerFactory.getLog(Attachment.class);

    // Raw Properties
    private String key;
    private String contentType;
    private byte[] binary;

    ...
}
```

2.2. Java with callouts example

This is a java example with syntax highlighting and callouts

```
@Entity
@Table(name = "ATTACHMENT")
@Cache(usage=CacheConcurrencyStrategy.NONSTRICT_READ_WRITE)
public class Attachment implements Comparable<Attachment>, Serializable {
    static final private long serialVersionUID = 1L;
    static final private Log logger = LoggerFactory.getLog(Attachment.class);

    // Raw Properties
    private String key;
    private String contentType;
    private byte[] binary;
```

- ❶ The @javax.persistence.Table annotation is JPA blabla
- ❷ This is the last line: binary field

2.3. XML

This is XML with syntax highlighting

```
<security:anonymous />
<security:http-basic />
<security:logout
    logout-url="/logout.action"
    logout-success-url="/index.action"/>
<security:remember-me user-service-ref="accountDetailsServiceImpl"/>
```

Chapter 3. Docbook authoring for a java developer

3.1. Resources

I added the most used docbook commands, check out these sites to learn some more:

- <http://www.sagehill.net/docbookxsl/>
- <http://docs.huihoo.com/xml/using-docbook/>
- <http://www.ogsadai.org.uk/documentation/ogsadai3.0/ogsadai3.0-axis/DocbookExamples.html>

3.2. command

You can specify commands like this : `<command>javac -version</command>`

The output: **javac -version**

3.3. screen

You can specify output like this : `<screen>screen output</screen>`

The output:

```
c:\tutorial\springfuse-example>mvn initialize
[INFO] Scanning for projects...
[INFO] -----
[INFO] Building springfuse-example
[INFO]   task-segment: [initialize]
[INFO] -----
[INFO] [sql:execute {execution: Create and initialize the database}]
[INFO] Executing file: c:\tutorial\springfuse-example\src\main\sql\h2\drop.sql
[INFO] Executing file: c:\tutorial\springfuse-example\src\main\sql\h2\create.sql
[INFO] Executing file: c:\tutorial\springfuse-example\src\main\sql\h2\comment.sql
[INFO] Executing file: c:\tutorial\springfuse-example\src\main\sql\h2\init.sql
[INFO] 45 of 45 SQL statements executed successfully
[INFO] [springfuse:extract {execution: Extract database Meta Data}]
[INFO] Ready to extract the database schema
[INFO] Connecting to database jdbcUrl=jdbc:h2:~/h2/quickstartdb
[INFO] Connected OK
[INFO] database Product Name: H2
[INFO] database schema extracted
[INFO] File data-model.springfuse passed reverse conversion OK
[INFO] File data-model.springfuse created successfully
[INFO] You are now ready to upload data-model.springfuse to http://www.springfuse.com/
and generate your project!
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 2 seconds
[INFO] Finished at: Tue Jan 27 16:10:07 CET 2009
[INFO] Final Memory: 4M/8M
[INFO] -----
```

3.4. filename

You can specify filenames like this :

```
<filename>src/main/generated-java/com/springfuse/example/model/Account.java</filename>
```

The output: src/main/generated-java/com/springfuse/example/model/Account.java

3.5. classname

You can specify classname like this : `<classname>GenericDaoService</classname>`

The output: GenericDaoService

3.6. literals

You can specify literals like this : `<literal>jdbc.driver</literal>`

The output: jdbc.driver

3.7. itemizedlist

You can set lists like this

```
<itemizedlist>
  <listitem><para>Spring Core</para></listitem>
  <listitem><para>Spring MVC</para></listitem>
  <listitem><para>Spring Security</para></listitem>
  <listitem><para>Spring Integration</para></listitem>
  <listitem><para>Spring Web Flow</para></listitem>
</itemizedlist>
```

The output

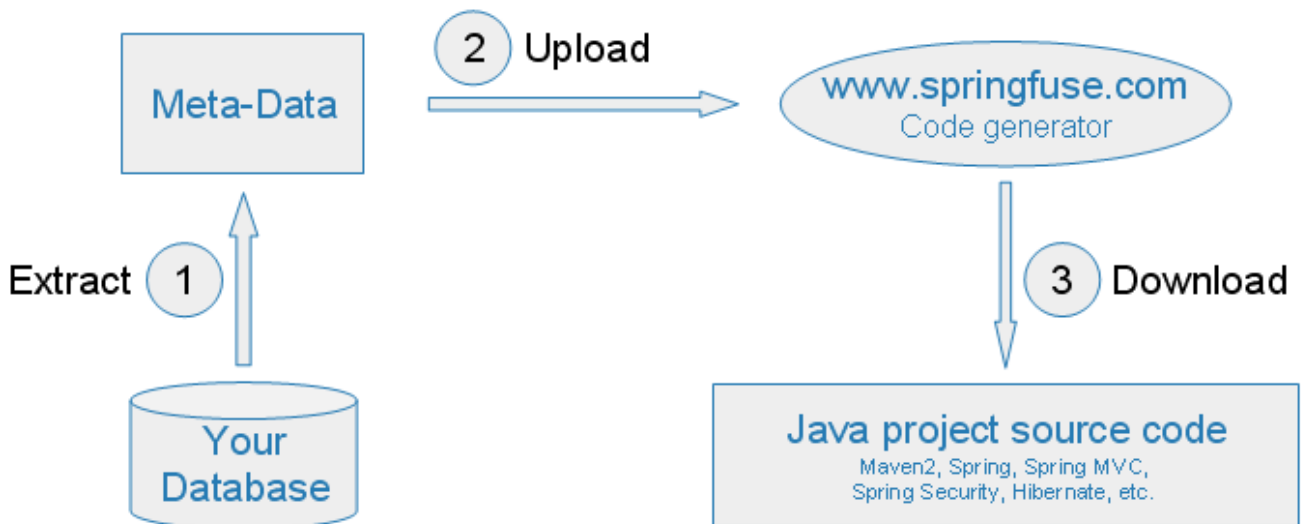
- Spring Core
- Spring MVC
- Spring Security
- Spring Integration
- Spring Web Flow

3.8. imageobject

You can add images like this

```
<mediaobject>
  <imageobject>
    <imagedata fileref="images/how-springfuse-works.png" align="center" />
  </imageobject>
  <caption>How springfuse works</caption>
</mediaobject>
```

The output:



How springfuse works

3.9. Maven

You can use the properties from your pom

Table 3.1. Maven properties in docbook

Property	XML	Value
pom.groupId	<?eval \${project.groupId}?>	com.springfuse.blog.docbook
pom.artifactId	<?eval \${project.artifactId}?>	docbook
pom.name	<?eval \${project.name}?>	docbook
pom.version	<?eval \${project.version}?>	1.0.0

Chapter 4. Bugs

4.1. Regression

For some reason the latest version (2.0.9) of the plugin produces :

```
Embedded error: org.apache.fop.apps.FOException: null:1:28951: Error(1/28951):  
No element mapping definition found for (Namespace URI:"http://xml.apache.org/fop/extensions", Local Name: "dest
```

I had to rely on version 2.0.8 this sole behavior stole hours from me.

4.2. PDF XSL and image

I also could not add an image in the `fopdf.xsl` on the cover page.

```
<fo:block>  
  <fo:external-graphic src="file:images/logo.png" />  
</fo:block>
```

❶ Not working

I had to use this notation instead:

```
<fo:block>  
  <fo:external-graphic src="file:src/docbkx/resources/images/logo.png" />  
</fo:block>
```

❶ Working

Even if this kind of bug seems obvious, it was not that easy to find when mixed between version problems, dependency hell, syntactic errors etc.

Anyway, hopefully this will help you to build great looking documentation.

Chapter 5. Conclusion

5.1. Docbkx works great

[Docbkx](#) is the right tool for the job. When correctly setup, it works like a charm.

5.2. Download maven project

You can download this maven project and build it by yourself here: <http://www.springfuse.com/blog/docbook/docbook-1.0.0-src.zip>

5.3. Missing artifacts ?

To produce image in PDF I found the solution reading [Open JPA](#) : FOP needs the JAI jars, unfortunately they are not in the central maven repository.

5.3.1. Manual installation

- You need to install manually the [Java Advanced Imaging framework](#) .
- Download the release at http://download.java.net/media/jai/builds/release/1_1_3/jai-1_1_3-lib.zip
- Unzip
- Install both jar locally in your repository by executing

```
mvn install:install-file -Dfile=jai-1_1_3\lib\jai_core.jar -DgroupId=javax.media -DartifactId=jai-core -Dversion=1.1.3
```

and

```
mvn install:install-file -Dfile=jai-1_1_3\lib\jai_codec.jar -DgroupId=javax.media -DartifactId=jai-codec -Dversion=1.1.3
```

5.3.2. Maven profile installation

I created the `install` maven profile: it will download the file for you and unzip it in `target/` .

- Just execute

```
mvn -Pinstall initialize
```

- Then in `target` execute

```
install.(bat|sh)
```

After that you have the jars in your local repository, you can re-execute the maven command:

```
mvn
```

to produce the documentation

5.4. Download PDF

You can download this article as a PDF at <http://www.springfuse.com/blog/docbook/maven-docbook-syntax-highlighting.pdf>