
Maven PDF Plugin

v. 1.0
User Guide

Table of Contents

1	Introduction	1
2	Usage	2
3	Filtering	5
4	Limitations	7
5	FAQ	8

1 Introduction

1.1 Overview

This plug-in allows you to generate a PDF version of your project's documentation.

1.1.1 Goals Overview

The PDF Plugin only has one goal.

- [pdf:pdf](#) Generates a PDF document containing all project documentation.

1.1.2 Usage

Instructions on the basic usage of the PDF Plugin can be found [here](#).

1.1.3 Example

Have a look at the [pdf](#) version of this web site.

The following examples show how to use the PDF Plugin in more advanced usecases:

- [Filtering Document Descriptor](#)
- [Site Phase Integration](#)

2 Usage

2.1 Using The PDF Plugin

The Maven PDF Plugin allows you generate a PDF document of your documentation.

2.1.1 From The Command-line

The PDF plugin can be called to execute from the command-line without any additional configurations. Like the other plugins, to run the PDF plugin, you use:

```
mvn pdf:pdf
```

where the first `pdf` refers to the plugin's alias, and the second `pdf` refers to a plugin goal.

By default, the pdf will be generated in `target/pdf/` directory.

Note 1: By default, the PDF plugin generates a PDF document which aggregates all your site documents. If you want to generate each site document individually, you need to add `-Daggregate=false` in the comand line.

Note 2: By default, the PDF plugin uses the **FOP** implementation. The plugin also supports the **iText** implementation, you just need to add `-Dimplmentation=itext` in the comand line.

2.1.2 As Part Of Your Build Process

The PDF plugin can be put into a project's `pom.xml` so that it gets executed everytime the project is built. Below is a sample configuration (to put into the list of `<plugins>` in the `<build>` section of your `pom.xml`) for running the PDF plugin in the `site` phase everytime the project is built:

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-pdf-plugin</artifactId>
  <version>1.0-SNAPSHOT</version>
  <executions>
    <execution>
      <id>pdf</id>
      <phase>site</phase>
      <goals>
        <goal>pdf</goal>
      </goals>
      <configuration>
        <outputDirectory>${project.reporting.outputDirectory}</outputDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>
```

Note: In this case, the pdf plugin is coupled with the Maven Site plugin to generate both site documentation and pdfs into the default output site directory, i.e. `target/site`. You just need to call `mvn site`.

2.1.3 Document Descriptor

By default, the pdf plugin processes all source files as specified in the site-plugins's `site.xml`. You can customize which files to include in which order by using a document descriptor (by default `src/site/pdf.xml`). An example is given below:

```
<document xmlns="http://maven.apache.org/DOCUMENT/1.0.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/DOCUMENT/1.0.1 http://maven.apache.org/xsd/document-1.0.1.xsd"
  outputName="maven-pdf-plugin-${project.version}">
  <meta>
    <title>Maven PDF Plugin</title>
    <author>The Apache Maven Project</author>
  </meta>
  <toc name="Table of Contents">
    <item name="Introduction" ref="index.apt"/>
    <item name="Usage" ref="usage.apt"/>
    <item name="Filtering" ref="examples/filtering.apt"/>
    <item name="Limitations" ref="limitations.apt"/>
    <item name="FAQ" ref="faq.fml"/>
  </toc>
  <cover>
    <coverTitle>${project.name}</coverTitle>
    <coverSubTitle>v. ${project.version}</coverSubTitle>
    <coverType>User Guide</coverType>
    <projectName>${project.name}</projectName>
    <projectLogo>http://maven.apache.org/images/maventxt_logo_200.gif</projectLogo>
    <companyName>The Apache Software Foundation</companyName>
    <companyLogo>http://www.apache.org/images/asf_logo_wide.png</companyLogo>
  </cover>
</document>
```

The meta information is only used for the pdf cover page if no cover element is given. The toc generates a Table of Contents and specifies the order of files to include in the pdf. For a complete description of the file format, see the [Document Model Reference](#).

Note 1: Only a few of document metadatas are used by the Fo/iText sinks like author, confidential, date and title.

Note 2: The document descriptor supports filtering as described in the [filtering example](#).

2.1.4 Internationalization

The PDF plugin is able to generate internationalized pdfs, similar to the [site creation](#).

To enable multiple locales, add a configuration similar to the following to your POM:

```
<project>
...
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pdf-plugin</artifactId>
      <configuration>
        <locales>en,fr</locales>
      </configuration>
    </plugin>
  </plugins>
</build>
...
</project>
```

This will generate two pdfs, one English and one French. Like the site plugin, if en is your current locale, then it will be generated at the root of the site, with a copy of the French translation of the site in the fr/ subdirectory.

The following is a complete internationalized directory layout for site and pdf plugins:

```
+ - src/
  + - site/
    + - apt/
      | + - index.apt      (Default version)
      |
    + - fr/
      | + - apt/
      |   + - index.apt   (French version)
      |
    + - site.xml          (Default site descriptor)
    + - site_fr.xml       (French site descriptor)
    + - pdf.xml           (Default pdf descriptor)
    + - pdf_fr.xml        (French pdf descriptor)
```

2.1.5 Specific FOP Configuration Properties

All the layout properties that are used for the pdf conversion using the FOP implementation are read from a default configuration file. The properties in this file may be overridden by using a custom configuration file `pdf-config.xml` located in `src/site/resources/`.

The format of this file has to be exactly the same as the original [default configuration file](#). However, you only need to specify the properties that you want to override, *e.g.* to change the font size of pre-formatted text, you could use:

```
<xsl:stylesheet
  version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format">
  <xsl:attribute-set name="body.pre" use-attribute-sets="base.pre.style">
    <xsl:attribute name="font-size">8pt</xsl:attribute>
  </xsl:attribute-set>
</xsl:stylesheet>
```


3 Filtering

3.1 Filtering Document Descriptor

The document descriptor (aka `src/site/pdf.xml`) could be filtered by System properties, Maven project properties and some date properties.

Expression Samples	Description
<code>\${JAVA_HOME}</code>	The JAVA_HOME environment value.
<code>\${project.name}</code>	The project name defined in <code><name/></code> tag in the <code>pom.xml</code> .
<code>\${project.developers[0].email}</code>	The email of the first developed defined in <code><developers/></code> tag in the <code>pom.xml</code> .
<code>\${date}</code>	The current date displayed in ISO-8601 format (i.e. <code>yyyy-MM-dd</code>), for instance <code>2009-06-22</code> .
<code>\${time}</code>	The current time displayed in ISO-8601 format (i.e. <code>HH:mm:ss 'Z'</code>), for instance <code>12:26:48Z</code> .
<code>\${dateTime}</code>	The current dateTime displayed in ISO-8601 format (i.e. <code>yyyy-MM-dd 'T' HH:mm:ss 'Z'</code>), for instance <code>2009-06-22T12:24:17Z</code> .
<code>\${year} \${month} \${day}</code>	The single date informations.
<code>\${hour} \${minutes} \${second}</code>	The single time informations.

3.2 Example

For instance, if you have defined the following `pom.xml` and `pdf.xml`:

```
<project>
  <modelVersion>4.0.0</modelVersion>
  <version>1.0-SNAPSHOT</version>
  <name>Your project</name>
  ...
  <developers>
    <developer>
      <email>your@email.com</email>
      ...
    </developer>
  </developers>
  ...
</project>
```

```
<document xmlns="http://maven.apache.org/DOCUMENT/1.0.1"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/DOCUMENT/1.0.1 http://maven.apache.org/xsd/document-1.0.1.xsd"
  outputName="maven-pdf-plugin-doc-${project.version}">
  <meta>
    <title>User guide of ${project.name} version ${project.version}</title>
    <author>${project.developers[0].email}</author>
  </meta>
  <toc name="Table of Contents">
    ...
  </toc>
  <cover>
    <coverdate>${date}</coverdate> <!-- current date in ISO 8601 format -->
    <!-- <coverdate>${day}/${month}/${year}</coverdate> current date in French format -->
    ...
  </cover>
</document>
```

The title will be User guide of Your project version 1.0-SNAPSHOT and the author will be your@email.com.

4 Limitations

4.1 Known Bugs and Limitations

Just a brief selection...

4.1.1 Current limitations

- Current prerequisite is Maven $\geq$ 2.0.6. It's using Doxia-1.1 via the shade-plugin (see [MNG-3402](#)).
- Maven generated pages (reports) are ignored.

4.1.2 Missing features

- Menu sub-items are not supported in TOC (every source document starts a new chapter).

4.1.3 Known bugs

None.

4.1.4 Implementation specific issues

These are not limitations of the plugin itself, but are listed here for completeness.

- Apache FOP issues
 - Table widths are always uniformly distributed.
 - Identical id attributes (eg anchors) within one document will lead to a build failure.
- iText issues

5 FAQ

5.1 Frequently Asked Questions

General

- 1 [Is it possible to create a book?](#)
- 2 [What graphics formats are supported?](#)

Specific problems

- 1 [Why does my image not fit on the page?](#)
- 2 [How can I center/in-line my image?](#)

5.2 General

Is it possible to create a book?

The [Doxia Book code](#) currently only supports the iText module for generating a pdf book.

[\[top\]](#)

What graphics formats are supported?

You can use the same graphics formats as are supported by the chosen implementation, eg see [Apache FOP Graphics Formats](#) and [iText Images](#). You should probably take care of image resolution, see bellow.

[\[top\]](#)

5.3 Specific problems

Why does my image not fit on the page?

This is most likely a resolution problem, for instance your image was saved with a 72 dpi resolution. Try to use an image with a higher resolution, like 96 dpi. You could resize your image with this program: [gimp](#). This is the only solution if you include the image from an apt source file (since in apt there is no possibility to specify the size of an image), if you are using xdoc, you may additionally indicate the size of the image using the width/height attributes of the `img` tag.

[\[top\]](#)

How can I center/in-line my image?

If you are using apt then your images will always be block-level elements, ie they will get centered in a separate paragraph. Apt does not support in-line images.

Using xdoc you are more flexible. By default a simple `<img>` tag can be used for an in-line image, eg:

```
<p>
  Here's a little icon:  inside my text.
</p>
```

If you want your image centered you may put it explicitly inside a centered paragraph:

```
<p align="center">
  
</p>
```

or you may use the Doxia-specific class attribute in a surrounding `<div>` block:

```
<div class="figure">
  
</div>
```

[\[top\]](#)