Apache Maven PDF Plugin

v. 1.3

User Guide

The Apache Software Foundation **2015-02-11**

Table of Contents

1 Introduction

1.1 Apache Maven PDF Plugin

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

General instructions on how to use the PDF Plugin can be found on the usage page. Some more specific use cases are described in the examples given below. Last but not least, users occasionally contribute additional examples, tips or errata to the plugin's wiki page.

In case you still have questions regarding the plugin's usage, please have a look at the FAQ and feel free to contact the user mailing list. The posts to the mailing list are archived and could already contain the answer to your question as part of an older thread. Hence, it is also worth browsing/ searching the mail archive.

If you feel like the plugin is missing a feature or has a defect, you can fill a feature request or bug report in our issue tracker. When creating a new issue, please provide a comprehensive description of your concern. Especially for fixing bugs it is crucial that the developers can reproduce your problem. For this reason, entire debug logs, POMs or most preferably little demo projects attached to the issue are very much appreciated. Of course, patches are welcome, too. Contributors can check out the project from our source repository and will find supplementary information in the guide to helping with Maven.

1.1.3 Examples

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
- Configuring reports

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.

Notes:

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 command line.
2. By default, the PDF plugin uses the FOP implementation. The plugin also supports the iText implementation, you just need to add `-Dimplementation=itext` in the command 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>
  <executions>
    <execution>
      <id>pdf</id>
      <phase>site</phase>
      <goals>
        <goal>pdf</goal>
      </goals>
      <configuration>
        <outputDirectory>${project.reporting.outputDirectory}</outputDirectory>
        <includeReports>false</includeReports>
      </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.x
  outputName="maven-pdf-plugin">

  <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.vm"/>
    <item name="Filtering Document Descriptor" ref="examples/filtering.apt"/>
    <item name="Configuring Reports" ref="/examples/configuring-reports.xml.vm"/>
    <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.

Notes:

1. Only a few of document metadatas are used by the Fo/iText sinks like author, confidential, date and title.
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>
        <version>1.3</version>
        <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 Document Descriptor

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

4.1 Configuring Reports

Note for Maven 3 users: due to the changes in the reporting API, report inclusion does **not** work with Maven 3. See MPDF-41.

Since version 1.1, all Maven reports will be included by default in the generated PDF. You should configure the <reporting/> section of your POM similar than the site plugin

For instance, you could have the following:

```
<project>
  ...
  <reporting>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>2.1.2</version>
        <reportSets>
          <reportSet>
            <reports>
              <report>project-team</report>
              ...
            </reports>
          </reportSet>
        </reportSets>
      </plugin>
      ...
    </plugins>
  </reporting>
  ...
  <project>
```

Notes:

1. to exclude the reporting generation inside the PDF, you should add `-DincludeReports=false` in the command line.
2. only internal reporting plugins will be added in the PDF, external reporting plugins like Javadoc will be skipped.

4.1.1 Enhancements

Having many reports increases **hugely** the build time, so it is recommended to select only the wanted reports to be included in the PDF. It is recommended to define a `reporting` profile in your pom, similar to the following:

```
<project>
  ...
  <profiles>
    <profile>
      <id>pdf</id>
      <reporting>
        <plugins>
          <plugin>
            <artifactId>maven-project-info-reports-plugin</artifactId>
            <version>2.1.2</version>
            <reportSets>
              <reportSet>
                <reports>
                  <report>cim</report>
                  <!-- take too long time
                  <report>dependencies</report> -->
                  <report>dependency-convergence</report>
                  <report>dependency-management</report>
                  <!-- already present
                  <report>index</report> -->
                  <report>issue-tracking</report>
                  <report>license</report>
                  <report>mailing-list</report>
                  <report>plugin-management</report>
                  <report>plugins</report>
                  <report>project-team</report>
                  <report>scm</report>
                  <report>summary</report>
                </reports>
              </reportSet>
            </reportSets>
          </plugin>
        </plugins>
      </reporting>

      <build>
        <defaultGoal>pdf:pdf</defaultGoal>
        <plugins>
          <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-pdf-plugin</artifactId>
            <version>1.3</version>
          </plugin>
        </plugins>
      </build>
    </profile>
    ...
  </profiles>
  ...
  <project>
```

4.1.2 Maven Reporting Plugins Issues

The Maven Project supports several reporting plugins. Unfortunately, some releases reporting plugins have known issues with the PDF plugin, mainly due to a wrong use of the Sink object in their implementation.

The following tables show reporting plugins which have been tested with the PDF plugin. The failed plugins have been fixed and deployed on https://repository.apache.org/content/repositories/snapshots, so you will be able to use them.

Note: fixing these reporting plugins is a work in progress. If you used a SNAPSHOT plugin which fails with the PDF plugin, you should fill a bug report in our issue tracker.

4. maven-changelog-plugin

Reports	Release (2.1)	Snapshot (2.2-SNAPSHOT)
changelog:changelog		
changelog:dev-activity		
changelog:file-activity		

4. maven-changes-plugin

Reports	Release (2.1)	Snapshot (2.2-SNAPSHOT)
changes:changes-report		
changes:jira-report		
changes:trac-report		

4. maven-checkstyle-plugin

Note: could take a lot of time.

Reports	Release (2.3)	Snapshot (2.4-SNAPSHOT)
checkstyle:checkstyle		

4. maven-dependency-plugin

Reports	Release (2.1)	Snapshot (2.2-SNAPSHOT)
dependency:analyze-report		

4. maven-plugin-plugin

Reports	Release (2.5)	Snapshot (2.5.1-SNAPSHOT)
plugin:report		

4. maven-pmd-plugin

Reports	Release (2.4)	Snapshot (2.5-SNAPSHOT)
pmd:cpd		
pmd:pmd		

4. maven-project-info-reports-plugin

Note: dependencies report could take a lot of time.

Reports	Release (2.1.2)	Snapshot (2.2-SNAPSHOT)
project-info-reports:cim		
project-info-reports:dependencies		
project-info-reports:dependency-convergence		
project-info-reports:dependency-management		
project-info-reports:index		
project-info-reports:issue-tracking		
project-info-reports:license		
project-info-reports:mailing-list		
project-info-reports:plugin-management		
project-info-reports:plugins		
project-info-reports:project-team		
project-info-reports:scm		
project-info-reports:summary		

4. maven-surefire-report-plugin

Reports	Release (2.4.3)	Snapshot (2.5-SNAPSHOT)
surefire-report:report		

4.1.3 Mojo Reporting Plugins Issues

The Mojo Project proposes several reporting plugins. Like the Maven project, some releases have known issues with the PDF plugin due to the Sink object uses.

The following tables show reporting plugins which have been tested with the PDF plugin. Some of them have been fixed and deployed on http://snapshots.repository.codehaus.org.

Note: these test tables are only for your information. If you find issues, please contact directly the mojo team

4. clirr-maven-plugin

Reports	Release (2.2.2)	Snapshot (2.2.3-SNAPSHOT)
clirr:clirr		

4. cobertura-maven-plugin

Reports	Release (2.3)	Snapshot (2.4-SNAPSHOT)
cobertura:cobertura		

4. l10n-maven-plugin

Reports	Release (1.0-alpha-2)	Snapshot (1.0-alpha-3-SNAPSHOT)
l10n:report		

4. javancss-maven-plugin

Reports	Release (2.0)	Snapshot (2.1-SNAPSHOT)
javancss:report		

5 Limitations

5.1 Known Bugs and Limitations

Just a brief selection...

5.1.1 Current Limitations

- Current prerequisite is Maven >= 2.0.6. It's using Doxia-1.1.x via the shade-plugin (see MNG-3402).

5.1.2 Missing Features

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

5.1.3 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

6 FAQ

6.1 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?

6.2 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]

6.3 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 whith 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: <img src="image.jpg"/> inside my text.
</p>
```

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

```
<p align="center">
  <img src="image.jpg"/>
</p>
```

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

```
<div class="figure">
  <img src="image.jpg"/>
</div>
```

[top]