The javadoc Package

Jolle

May 24, 2008

Abstract

The javadoc package provides an easy way to document source code. It is related to the javadoc system for java source code and tries to provide the same descriptions. But, of course, source code of other languages can be documented using this package. The package is under GNU GENERAL PUBLIC LICENSE

Contents

1 Introduction 2
2 Options and Required Packages 2
3 Design-Commands 3
4 Linking in the Output-PDF 3
5 Known Issues 3
6 Structure of a class 4
7 Description of a class 4
7.1 The environment jdinheritancetable 4
7.2 Commands for all environments except jdinheritancetable 4
7.2.1 Modifier 4
7.2.2 Codebased Attributes 5
7.2.3 Javadocbased Attributes 5
7.3 Table with commands and environments 6

*Comments, Help, Questions, Hints, Critics to joerman.lieder@gmx.net
1www.gnu.org
1 Introduction

Javadoc is a powerful tool for java developer to document their source code. It produces a comprehensive collection of HTML-pages out of special form-
tet comments in sourc code. The package javadoc uses the same attributes
to describe the source code with \LaTeX. In combination with the TexGen
doclet the \TeX-documentation can be generated with the javadoc out of the
source code.

2 Options and Required Packages

The javadoc-package requires one other package. The longtable is used to
display the tables of inherited fields and methods.

The package provides options to customize the layout and structure of
the document. The package occupies 3 levels of hierarchy, with the options
chapter, section, subsection the highest level can be set, the others will
be adapt automatically. The default is section. The second possibility to
customize the behaviour is to set the table of content. The two forms of
chapter and chapter* and so on can be used. The following table lists the
possible options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Table of Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>toc0</td>
<td>no level</td>
</tr>
<tr>
<td>toc1</td>
<td>highest level, default</td>
</tr>
<tr>
<td>toc2</td>
<td>the two highest levels</td>
</tr>
<tr>
<td>toc3</td>
<td>all levels</td>
</tr>
<tr>
<td>toc</td>
<td>like toc3</td>
</tr>
<tr>
<td>notoc</td>
<td>like toc0</td>
</tr>
</tbody>
</table>

The entries of the table of contents might be changed by other settings
independent of this package options.

The hyperref-option produces links inside of the document. This refers
to the datatypes of parameters, classes, methodreturns, etc. It also produces
many warnings during compilation process due to the missing targets. Using
this option, the package hyperref is loaded. Options can be set with the
\hypersetup command.

The package provides the possibility to use different languages. It belongs
to the headinds and words, no options or packages are loaded. Codespecific
words are not translated. An implemented option is deutsch, default is
english. Other languages can be easily integrated by translating the following
commands. All language-commands start with \jd@lang@, the endings are
listed in table 2.
3 Design-Commands

2 additional commands are helpful. \texttt{\textbackslash jdinh} draws an arrow for inheritance from right to left. \texttt{\textbackslash jdcode} has one argument and changes the font to True-Type.

4 Linking in the Output-PDF

For linkings the arguments of \texttt{\textbackslash jdtype} and the first arguments of \texttt{\textbackslash JDpara} and \texttt{\textbackslash jdInhEntry} must contain the link-information. These information are set with \texttt{\textbackslash jdtypesimple\{type\}} or \texttt{\textbackslash jdtypearray\{name\}\{dimension\}} or \texttt{\textbackslash jdtypegeneric\{name\}\{generic\}}. For generic types the single classes must be signed with the named commands. The targets are set automatically. You can use all these commands without worrying about the use of the \texttt{hyperref} option. Without the option, the links are ignored and produces no errors or warnings.

5 Known Issues

- Problems comes about the linkings to not described classes. There are warnings during compilation process and missing links in the output.
- Only class- and interfacenames are linked, not methods or else.
- The label for linking contains the classname. Two equal named classes produces errors.
- Method- und Fieldnames often produces overfull boxes in the headings.

Table 1: Language commands
6 Structure of a class

The hierarchy levels are already mentioned, here comes the description. Describing a class starts with a classname. This name will be the highest hierarchy level. Then the headings for Fields, Methods, Constructors follow, the lowest level is for the elements of the class (methodnames, fieldnames...)

7 Description of a class

The outer environment for a class is \texttt{jdclass}. \texttt{jdclass} has an argument with the classname. An option can be given with the type \texttt{class} or \texttt{interface} or \texttt{enum}. Default is \texttt{class}. Inside of the class environment, the following structure has to be kept.

- \texttt{jdclassheader}
- \texttt{jdinheritancetable}
- \texttt{jdfield}
- \texttt{jdconstructor}
- \texttt{jdmethod}

The environment \texttt{jdclassheader} can be written once per class, the environments \texttt{jdconstructor} and \texttt{jdinheritancetable} need no argument with the name.

7.1 The environment \texttt{jdinheritancetable}

The table entries can be produced with the \texttt{\textbackslash jdInhEntry} command. It has two arguments, the first one is the element the second the class, that inherited the element.

7.2 Commands for all environments except \texttt{jdinheritancetable}

For all environments the same elements are valid in general. But not all elements are used everywhere. The table 2 lists the usage of commands in environments. The usage of commands not belonging to an environment doesn’t produces a failure, but it has no effect. The javadoc-package has no java-syntax-check, you can call contradictory modifier if you feel to.

7.2.1 Modifier

The following modifier can be named. They have no arguments.

- \texttt{\textbackslash jdpublic}
7.2.2 Codebased Attributes

The following attributes are not javadoc-based but contain important information:

- \jdpackage{packagename} The package containing the class.
- \jدينherits{classname} Inherited class. For a hierarchy, the arrow \jdinh can be used.
- \jdimplements{interface} A Interface, that is implemented. Can be named more than once.
- \jdouterclass{classname} Defines an outer class for an inner one.
- \jdtype{type} Data type, especially for return values, and fields. A method without type gets automatically void.

7.2.3 Javadocbased Attributes

Most arguments have an argument containing their description.

- \JDcategory{description}
- \JDdeprecated{description}
- \JDserial{description}
- \JDserialData{description}
- \JDserialField{description}
- \JDsince{description}
- \JDtext{description}
- \JDversion{description}
There are three other attributes that can be named more than once and/or contain more than one argument.

- \texttt{\textbackslash JDreturn}\{description\}
- \texttt{\textbackslash JDsee}\{description\}
- \texttt{\textbackslash JDauthor}\{authorname\}
- \texttt{\textbackslash JDpara}\{datatype\}\{name\}\{description\}
- \texttt{\textbackslash JDthrows}\{exceptionname\}\{description\}

7.3 Table with commands and environments

The table sums up, which commands can be named in which environment.
<table>
<thead>
<tr>
<th>Command</th>
<th>jclassheader</th>
<th>jfield</th>
<th>jconstructor</th>
<th>jmethod</th>
</tr>
</thead>
<tbody>
<tr>
<td>\jpublic</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jprotected</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jprivate</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jstatic</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jabstract</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jfinal</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jtransient</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jvolatile</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jpackage{packagename}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jinherits{classname}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jimplements{interface}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jouterclass{classname}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jtype{type}</td>
<td>X</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jAuthor{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jCategory{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jDeprecated{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jSee{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jSerial{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jSerialData{description}</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>\jSerialField{description}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jSince{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jText{description}</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>\jVersion{description}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jReturn{description}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jPara{datatype}{name}{description}</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>\jThrows{exceptionname}{description}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Usage of commands in environments