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[1]

Contents

*Comments, Help, Questions, Hints, Critics to joerman.lieder@gmx.net

[1] www.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 formattet 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.

Option	Table of Content
`toc0`	no level
`toc1`	highest level, default
`toc2`	the two highest levels
`toc3`	all levels
`toc`	like `toc3`
`notoc`	like `toc0`

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.

field	author
method	category
constr	deprecated
	parameter
fullname	see
package	serial
inherits	serialData
implements	serialField
outerclass	since
	return
elementname	throws
inheritOf	version
inheritancetable	

Table 1: Language commands

3 Design-Commands

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

4 Linking in the Output-PDF

For linkings the arguments of `\jdtype` and the first arguments of `\JDpara` and `\jdInhEntry` must contain the link-information. These information are set with `\jdtypesimple{type}` or `\jdtypearray{name}{dimension}` or `\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 `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.

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 `jdclass`. `jdclass` has an argument with the classname. An option can be given with the type `class` or `interface` or `enum`. Default is `class`. Inside of the class environment, the following structure has to be kept.

- `jdclassheader`
- `jdinheritancetable`
- `jdfield`
- `jdconstructor`
- `jdmethod`

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

7.1 The environment `jdinheritancetable`

The table entries can be produced with the `\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 `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.

- `\jdpublic`

- \jdprotected
- \jdprivate
- \jdstatic
- \jdabstract
- \jdfinal
- \jdtransient
- \jdvolatile

7.2.2 Codebased Attributs

The following attributes are not javadoc-based but contains important information

- \jdpackage{packagename} The package containing the class.
- \jdinherits{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 fiels. A method without type gets automatically `void`.

7.2.3 Javadocbased Attributs

Most arguments have an argument containing their description.

- \JDcategory{description}
- \JDdeprecated{description}
- \JDserial{description}
- \JDserialData{description}
- \JDserialField{description}
- \JDsince{description}
- \JDtext{description}
- \JDversion{description}

- \JDreturn{description}

There are three other attributs, that can be named more than once and/or contain more than one argument.

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

7.3 Table with commands and environments

The table 2 sums up, which commands can be named in which environment.

Command	jdclassheader	jdfield	jdconstructor	jdmethod
\jdpublic	X	X	X	X
\jdprotected	X	X	X	X
\jdprivate	X	X	X	X
\jdstatic	X	X	X	X
\jdabstract	X	X	X	X
\jdfinal	X	X	X	X
\jdtransient		X		
\jdvolatile		X		
\jdpackage{packagename}	X			
\jdinherits{classname}	X			
\jdimplements{interface}	X			
\jdouterclass{classname}	X			
\jdtype{type}		X		X
\JDauthor{description}	X	X	X	X
\JDcategory{description}	X	X	X	X
\JDdeprecated{description}	X	X	X	X
\JDsee{description}	X	X	X	X
\JDserial{description}	X	X	X	X
\JDserialData{description}			X	X
\JDserialField{description}		X		
\JDsince{description}	X	X	X	X
\JDtext{description}	X	X	X	X
\JDversion{description}	X			
\JDreturn{description}			X	X
\JDpara{datatype}{name}{description}			X	X
\JDthrows{exceptionname}{description}			X	X

Table 2: Usage of commands in environments