# Das `javadoc` Package

Jolle*

24. Mai 2008

**Zusammenfassung**

Das `javadoc` Package bietet eine einfache Möglichkeit, Dokumentationen von Quellcode zu erstellen. Es leitet sich vom Dokumentationssystem javadoc für Java ab und bietet dieselben Attribute an. Aber natürlich kann auch Quellcode anderer Sprachen damit dokumentiert werden. Das Paket steht unter GNU GENERAL PUBLIC LICENSE[1]

## Inhaltsverzeichnis



*Verbesserungsvorschläge, Kommentare, Korrekturen, Anregungen und Danksagungen an joerman.lieder@gmx.net

[1] www.gnu.org

# 1 Einführung

Mit dem Programm `javadoc` wird den Programmierern in der Java-Welt ein mächtiges Werkzeug gegeben, ihren Quellcode zu dokumentieren. Javadoc arbeitet mit besonders formatierten Kommentaren im Quelltext und generiert daraus eine umfassende Sammlung an HTML-Seiten. Das Package `javadoc` bietet die Möglichkeit, mit denselben Attributen eine Dokumentation mit LaTeX zu erstellen. Mächtig wird das Package vor allem im Zusammenspiel mit einem Doclet, das die Kommentare aus dem Quellcode in TeX-Dateien mit den Befehlen dieses `javadoc`-Packages umwandelt.

# 2 Optionen und benötigte Packages

Derzeit benötigt `javadoc` nur ein Paket, nämlich `longtable`. Ohne dieses kann die z.T. sehr umfangreiche Tabelle mit geerbten Feldern und Methoden nicht dargestellt werden.

Das Paket bietet Optionen, um das Layout und die Struktur der Dokumentation anzupassen. Das Paket selbst belegt 3 Hierarchiestufen, durch die Optionen `chapter`, `section`, `subsection` lässt sich die oberste Stufe festlegen, die hinteren werden automatisch nach unten geschoben. Wird keine Option angegeben, wird `section` als oberste Stufe verwendet. Die zweite Möglichkeit über Optionen das Verhalten zu beeinflussen, liegt im Festlegen, welche Hierarchiestufe ins Inhaltsverzeichnis mit aufgenommen wird und welche nicht. Dies wird realisiert durch die *-Befehle von `chapter` usw. Die folgende Tabelle listet die möglichen Optionen auf.

| Option | Aufnahme ins Inhaltsverzeichnis |
|---|---|
| `toc0` | keine Stufe |
| `toc1` | oberste Stufe `Default`-Wert |
| `toc2` | obersten zwei Stufen |
| `toc3` | alle Stufen |
| `toc` | entspricht `toc3` |
| `notoc` | entspricht `toc0` |

Unabhängig davon, welche Option angegeben ist, können andere Einstellungen die Darstellung im Inhaltsverzeichnis verändern.

Die `hyperref`-Option kann angegeben werden, wenn erwünscht ist, dass innerhalb der Ausgabedatei Verlinkungen erzeugt werden. So lassen sich von Parametertypen oder Rückgabewerten oder Vererberklassennamen mit Mausklick die entsprechenden Klassen aufrufen, wenn sie denn in derselben Datei liegen. Ansonsten springt Acrobat an den Anfang des Dokuments. Diese Option empfiehlt sich erst am Ende des Schreibvorgangs zu verwenden, da sie sehr viele Warnungen aufgrund nicht verfolgbarer Referenzen erzeugt.

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

Tabelle 1: Sprachbefehle

Wird diese Option gewählt wird das Package `hyperref` ohne eine Option eingebunden. Diese können mit `\hypersetup` nachgeladen werden.

Desweiteren bietet das `javadoc`-Package Möglichkeiten, die Ausgaben zwischen verschiedenen Sprachen umzustellen. Das betrifft aber nur Überschriften und Beschreibungen, natürlich nicht codespezifische Wörter. Außerdem werden keine sprachspezifischen Einstellungen durchgeführt, sondern nur Übersetzungen getätigt. Als derzeit einzige Option lässt sich `deutsch` auswählen, Standard ist Englisch. Um andere Sprachen zu implementieren brauchen nur die Befehle überschrieben werden. Alle Sprachbefehle beginnen mit `\jd@lang@`, die jeweiligen Endungen sind in Tabelle 2 gelistet.

# 3 Befehle zur Darstellung

Bevor wir auf die Umgebungen und die Dokumentationsmöglichkeiten zu Sprechen kommen, sollten noch 2 Befehle genannt werden, die der Gestaltung dienen. `\jdinh` steht für "Inherits" und malt einen Vererbungspfeil von rechts nach links. `\jdcode` benötigt einen Parameter und stellt diesen Text in TrueType-Schrift dar.

# 4 Verlinkung im PDF

Um Verlinkungen innerhalb des PDFs zu erzeugen, muss beim Argument von `\jdtype` und beim jeweils 1. Argument von `\JDpara` und `\jdInhEntry` die entsprechenden Wörter über die Befehle `\jdtypesimple{type}` oder `\jdtypearray{name}{dimension}` oder `\jdtypegeneric{name}{generisch}` als Klassen gekennzeichnet werden. Im generischen Typ müssen die einzelnen

Klassen wieder mit den oben genannten Befehlen gekennzeichnet werden. Die Ziele dieser Verlinkungen werden automatisch gesetzt. Wird die `hyperref`-Option nicht genutzt, können die Befehle trotzdem angegeben werden. Sie erzeugen dann keinen Fehler.

# 5 Known Issues

- Probleme bereiten derzeit die nicht existierenden Verlinkungsziele von Datentypen, deren Klassen nicht beschrieben werden. Diese tauchen dann als Warnungen beim Kompilieren und als Verweise ins Nichts im PDF auf.
- Auch werden außer Datentypen keine anderen Sachen (wie Methoden) verlinkt.
- Beim Verlinken wird nur der Klassenname verwendet, nicht das Package, sodass zwei gleiche Klassen in unterschiedlichen Paketen das gleiche Ziel darstellen. Gelöst werden kann es dadurch, das der volle Name der Klasse angegeben wird. Das Doclet unterstützt derzeit keine generischen Typen und "doppelte" Klassennamen. Weitere Informationen zum Doclet finden sich in der dortigen Dokumentation.
- Methoden- und Feldnamen erzeugen häufig zu volle Boxen in den Überschriften.

# 6 Gliederung einer Klasse

Die Hierarchie-Stufen wurden schon mehrfach erwähnt, nun soll erläutert werden, was sie bedeuten. Um eine Klasse zu beschreiben wird der Klassenname als Hauptüberschrift verwendet (oberste Stufe). Danach folgen Unterteilungen für Felder, Konstruktoren oder Methoden. Und auf der untersten Ebene stehen die Namen der einzelnen Elemente einer Klasse.

# 7 Beschreiben einer Klasse

Als äußere Umgebung für eine Klasse wird `jdclass` verwendet. `jdclass` hat ein Pflichtargument, dass den Namen der Klasse beschreibt. Als optionales Argument lässt sich `class` oder `interface` angeben, wobei standardmäßig ersteres verwendet wird. Auch `enum` wäre ein denkbarer Wert. Innerhalb der Klassendeklaration lassen sich die unterschiedlichen Elemente beschreiben. Dabei ist die folgende Reihenfolge notwendig.

- `jdclassheader`

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

Die Umgebung `jdclassheader` darf nur einmal pro Klasse vorkommen, die Umgebungen `jdconstructor` und `jdinheritancetable` benötigen kein Pflichtargument mit dem Namen.

## 7.1 Die Umgebung `jdinheritancetable`

In dieser Tabelle werden mit `\jdInhEntry` die Einträge für die Tabelle erzeugt. Der Befehl braucht 2 Argumente, das 1. für ein Element, das 2. für die Klasse, von der das Element geerbt wird.

## 7.2 Befehle für alle Umgebungen außer `jdinheritancetable`

Für jede dieser Umgebungen sind im Allgemeinen die gleichen Elemente gültig. Allerdings werden nicht alle Elemente überall verwendet. In Tabelle 2 ist aufgeführt, welche Befehle in welcher Umgebung verwendet werden. Die Verwendung von Befehlen, die nicht zur Umgebung gehören, erzeugt keinen Fehler, hat aber keine Auswirkung. Darüberhinaus unternimmt das `javadoc`-Package auch keine Java-Syntaxprüfung. So ist es durchaus möglich, sich widersprechende Modifier anzugeben, aber nicht sinnvoll.

### 7.2.1 Modifier

Folgende Modifier lassen sich zu einem Element angeben, sie verlangen keine Argumente.

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

### 7.2.2 Codebasierte Attribute

Folgende Befehle sind keine typischen Javadoc-Elemente, enthalten aber wertvolle Informationen zu einer Klasse.

- `\jdpackage{packagename}` Das Package, in dem die Klasse enthalten ist.
- `\jdinherits{classname}` Vererbte Klasse. Um eine Vererbungshierarchie darzustellen, kann der Pfeil `\jdinh` verwendet werden.
- `\jdimplements{interface}` Ein Interface, dass implementiert wird. Kann mehrfach vorkommen.
- `\jdouterclass{classname}` Definiert für innere Klassen den Namen der äußeren.
- `\jdtype{type}` Datentyp, vor allem bei Rückgabewerten oder Feldnamen. Wird bei einer Methode kein Rückgabewert angegeben, gilt automatisch `void`.

### 7.2.3 Javadocbasierte Attribute

Die meisten Javadoc-Attribute haben ein Pflichtargument, dass ihre Beschreibung enthält. Dazu zählen folgende Attribute:

- `\JDcategory{beschreibung}`
- `\JDdeprecated{beschreibung}`
- `\JDserial{beschreibung}`
- `\JDserialData{beschreibung}`
- `\JDserialField{beschreibung}`
- `\JDsince{beschreibung}`
- `\JDtext{beschreibung}`
- `\JDversion{beschreibung}`
- `\JDreturn{beschreibung}`

Darüberhinaus gibt es drei weitere Javadoc-Attribute, die mehrfach auftreten dürfen und z.T. mehrere Argumente annehmen.

- `\JDsee{beschreibung}`
- `\JDauthor{autorname}`
- `\JDpara{datentyp}{name}{beschreibung}`
- `\JDthrows{exceptionname}{beschreibung}`

### 7.3 Zuordnungstabelle Umgebung zu Befehl

Die Tabelle 2 fasst zusammen, welche beschreibenden Befehle in welcher Umgebung auftauchen dürfen.

| Befehl | 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{beschreibung} | X | X | X | X |
| \JDcategory{beschreibung} | X | X | X | X |
| \JDdeprecated{beschreibung} | X | X | X | X |
| \JDsee{beschreibung} | X | X | X | X |
| \JDserial{beschreibung} | X | X | X | X |
| \JDserialData{beschreibung} | | | X | X |
| \JDserialField{beschreibung} | | X | | |
| \JDsince{beschreibung} | X | X | X | X |
| \JDtext{beschreibung} | X | X | X | X |
| \JDversion{beschreibung} | X | | | |
| \JDreturn{beschreibung} | | | X | X |
| \JDpara{datentyp}{name}{beschreibung} | | | X | X |
| \JDthrows{exceptionname}{beschreibung} | | | X | X |

Tabelle 2: Verwendung von Befehlen in Umgebungen