

JavaDoc

-

GUIDLINE FÜR DIE BENUTZUNG VON JAVADOC

Version 0.1

**Inhalt**

[**0.1** **Zielsetzung des Dokuments** 2](#_Toc528062345)

[0.2 Changelog 2](#_Toc528062346)

[1.0 Einführung JavaDoc 3](#_Toc528062347)

[2.0 Glossar 3](#_Toc528062348)

[3.0 Vorhandene und verwendete Tags 3](#_Toc528062349)

[4.0 ÜBERSCHRIFT KAPITEL 3](#_Toc528062350)

[5.0 ÜBERSCHRIFT KAPITEL 3](#_Toc528062351)

[6.0 ÜBERSCHRIFT KAPITEL 3](#_Toc528062352)

# **Zielsetzung des Dokuments**

Das Dokument „JavaDoc“ dient der Aufklärung über das Tool JavaDoc des JDK’s (Java Development Kit) und der Festlegung von zu verpflichtenden Tags und Inhalten in Kommentaren. Hinzu kommt außerdem die Richtlinien unserer Code Formatierung.

* 1. Changelog

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Datum** | **Änderung** | **Geändert von** |
| 0.1 | 31.10.2018 | Initial | Tim Dahm |

1. Einführung JavaDoc

JavaDoc ist ein Dokumentationstool, das Informationen aus dem Quellcode extrahiert und als HTML-Dokumentationsdatei speichert. Es ist seit Version 2 ein Bestandteil des Java Development Kits und bietet neben den standardisierten Tags auch die alternative Tags durch die Verwendung von Doclets. Dabei „parst“ dieses Tool die zu dokumentierenden Java-Quelltexte und sucht nach Kommentaren beginnend mit „/\*\*“ und Tags in diesen Kommentaren beginnend mit „@“ oder „{@“.

1. Glossar

* Java Development Kit: SDK des Unternehmens Oracle
* Tag: Wort beginnend mit „@“ oder „{@“ im Java Quelltext in einem Kommentar
* Taglet: Erweiterungsmöglickeit des Standard-Wortschatzes
* Doclet: Erzeugt Ausgabe in Datei (Standardweise in HTML)

1. Vorhandene und verwendete Tags

Rot: Wird nicht verwendet

Grün: Verwendung gefordert

Gelb: Optionale anwendung, verschönert Endergebnis

|  |  |  |
| --- | --- | --- |
| Tag und Parameter | Ausgabe | Verwendung in |
| @author name | Beschreibt den Autor. | Klasse, Interface |
| @version version | Erzeugt einen Versionseintrag. Maximal einmal pro Klasse. | Klasse, Interface |
| @since jdk-version | Seit wann die Funktionalität existiert. | Klasse, Interface, Instanzvariable, Methode |
| @see reference | Erzeugt einen Link auf ein anderes Element der Dokumentation. | Klasse, Interface, Instanzvariable, Methode |
| @serial | Beschreibt die serialisierten Daten eines Serializable-Objekts. | Klasse |
| @serialField | Beschreibt ein Feld eines Serializable-Objekts. | Klasse, Methode |
| @param name description | Parameterbeschreibung einer Methode. | Methode |
| @return description | Beschreibung des Rückgabewerts einer Methode. | Methode |
| @exception classname description  @throws classname description | Beschreibung einer Exception, die von einer Methode geworfen werden kann. | Methode |
| @deprecated description | Beschreibt eine alte Methode, die nicht mehr verwendet werden sollte. Sollte ab Java 5.0 immer mit der @Deprecated-Annotation verwendet werden. | Methode |
| {@inheritDoc} | Kopiert die Beschreibung aus der überschriebenen Methode. | Überschriebene Methode |
| {@link reference} | Link zu einem anderen Symbol. | Klasse, Interface, Instanzvariable, Methode |
| {@linkPlain reference} | Der Link wird in Standardtext statt in Quelltextzeichen angezeigt. | Klasse, Interface, Instanzvariable, Methode |
| {@value} | Gibt den Wert eines konstanten Feldes zurück. | Statisches Feld |
| {@docRoot} | Gibt den Pfad zum Hauptverzeichnis wieder. | Package, Klassen, Felder, Methoden |
| {@code} | Formatiert Text buchstabengetreu mit dem Quelltextsatz (entsprechend <code>) und blockiert die Interpretierung von beinhalteten HTML oder Javadoc-Tags. | Klasse, Interface, Instanzvariable, Methode |
| {@literal} | Kennzeichnet buchstabengetreuen Text und unterdückt die Interpretierung von beinhalteten HTML oder Javadoc-Tags. | Klasse, Interface, Instanzvariable, Methode |

1. Anwendungsbeispiele

.. TODO

1. ÜBERSCHRIFT KAPITEL

@Lukas TODO