Hilfestellung zum Dokumentieren

Huber Sabrina

September 18, 2013

Dokumentieren dient der Verständlichkeit und sollte während dem Programmieren nicht vernachlässigt werden. Ziel ist die Verständlichkeit des eigenen Quellcodes für sich selbst und für andere durch einfaches durchlesen einiger weniger Zeilen, zu gewährleisten.

1 Was muss dokumentiert werden?

Dokumentiert wird in den Quellcode hinein und abschließend generieren diverse Programme (Javadoc, Doxygen)den Text als (z.B HTML) Darstellung.
Alles was nicht auf den ersten Blick hin selbst erklärend ist, sollte dokumentiert werden.

- Ziel und Aufgabe der Klasse und der Methoden
- Aufwendige und komplexe Algorithmen

Jedoch sollten nicht alle Zeilen eines Quellcodes dokumentiert werden. Auf die Beschreibung von einfachen Variablenzuweisungen, gewöhnlichen Getter- und Setter-Methoden oder Konstruktoren, kann verzichtet werden. Romane sind nur selten Hilfreich in einer Dokumentation.

2 Wie dokumentiere ich richtig?

Dokumentationen von Klassen, Methoden und Variablen werden in der Quelldatei als Kommentare in einem besonderen Format geschrieben.
In Java kann ein Zusatztext mit // oder mit /** Text */ zu einer Dokumentation umgewandelt werden.

2.1 Formen der Kommentierung

Verwende // um einen einzeiligen Kommentar niederzuschreiben. Eignet sich gut bei aufwendigen Algorithmen. Dieser Text erscheint jedoch nicht in der Dokumentation,

kann aber jederzeit in der Klasse nachgelesen werden.

Mit /** Text */ wird jene Erklärung geschrieben, welcher in der HTML-Dokumentation zu sehen ist. Wenn in Eclipse nach Eingabe von /** die Entertaste betätigt wird, komplettiert Eclipse den restlichen Ausdruck automatisch. Außerdem fügt er, wenn der Ausdruck vor einer Klasse oder Methode gemacht wurde, den Kommentarabschnitt bereits um einige Tags. Zum Beispiel: Innerhalb der blauen * kann die Dokumentation verfasst

```
/**
 *
 * @param args
 */
public static void main(String[] args) {
```

Figure 1: Main-Methode bekommt Parameter "args" übergeben.

werden. In Java gibt es einige Tags, welche die Darstellung der Dokumentation erleichtern sollen. Eine genauere Auflistung aller Möglichkeiten finden sich hier. In diesem Dokument werden die meist verwendeten Tags aufgeführt.

2.2 Konventionen

Auch bei Tags gibt es einige Konventionen die eingehalten werden müssen, aus dem einfachen Grund, damit kein Chaos entsteht, dort wo sie eigentlich helfen sollen Ordnung zu schaffen.

2.2.1 Tags anordnen

Tags sollten, wenn sie verwendet werden in dieser Reihenfolge eingebunden werden.

1. author
2. version
3. param
4. return
5. exception oder throws
6. see
7. since
8. deprecated

2.2.2 Mehrere gleiche Tags anordnen

Wird ein Tag mehrmals verwendet, gibt es eine kleine Reihenfolge zu beachten.

@author Tags sollten nach chronologischer Reihenfolge aufgelistet werden, somit steht der Erzeuger der Klasse ganz oben, ihm folgen die weiteren Bearbeiter der Klasse.

@param Tags sollten die Variablen nach dem First Come, First Served Prinzip erläutern. Variablen auf welche man in einer Klasse/Methode/Interface als erstes stößt, werden als erstes erklärt, danach die zweite usw.

@throws und @exceptions werden Alphabetisch angegeben.

@see ist schon ein wenig aufwendiger. Von der ausgehenden Klasse aus, sollten see-Tags so angeordnet werden, dass von den nähesten bis zum am weit entferntesten, und vom am schlechtesten qualifizierten Methode bis hin zur am besten qualifiziertesten Methode, die Tags angeordnet werden. Folgendes Abbild 2 verdeutliche die Erklärung.

```
@see #field
@see #Constructor(Type, Type...)
@see #Constructor(Type id, Type id...)
@see #method(Type, Type,...)
@see #method(Type id, Type, id...)
@see Class
@see Class#field
@see Class#Constructor(Type, Type...)
@see Class#Constructor(Type id, Type id)
@see Class#method(Type, Type,...)
@see Class#method(Type id, Type id,...)
@see package.Class
@see package.Class#field
@see package.Class#Constructor(Type, Type...)
@see package.Class#Constructor(Type id, Type id)
@see package.Class#method(Type, Type,...)
@see package.Class#method(Type id, Type, id)
@see package
```

Figure 2: Quelle: http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html

2.2.3 Vorgeschriebene Tags

Theoretisch können bei der Dokumentation auf alle Tags verzichtet werden und nur blanker Text geschrieben werden. Sollten jedoch **Parameter** oder **Returns** vorkommen, müssen diese mit einem Tag beschrieben werden, selbst wenn diese noch so selbsterklärend oder redundant im gesamten Programm sind.

3 Tags

Nun noch einige wichtige Tags und wie sie anzuwenden sind.

Hinweis: Die folgenden Darstellungen zeigen den zu schreibenden Tag im Code in einem rot eingerahmten Bereich, sowie die Enddarstellung im Dokument durch Doxygen an.

3.1 @author

Dieser Befehl zeigt den Autor an. Ideal um im Nachhinein festzustellen, wer für diese Klasse oder Funktion zuständig war.

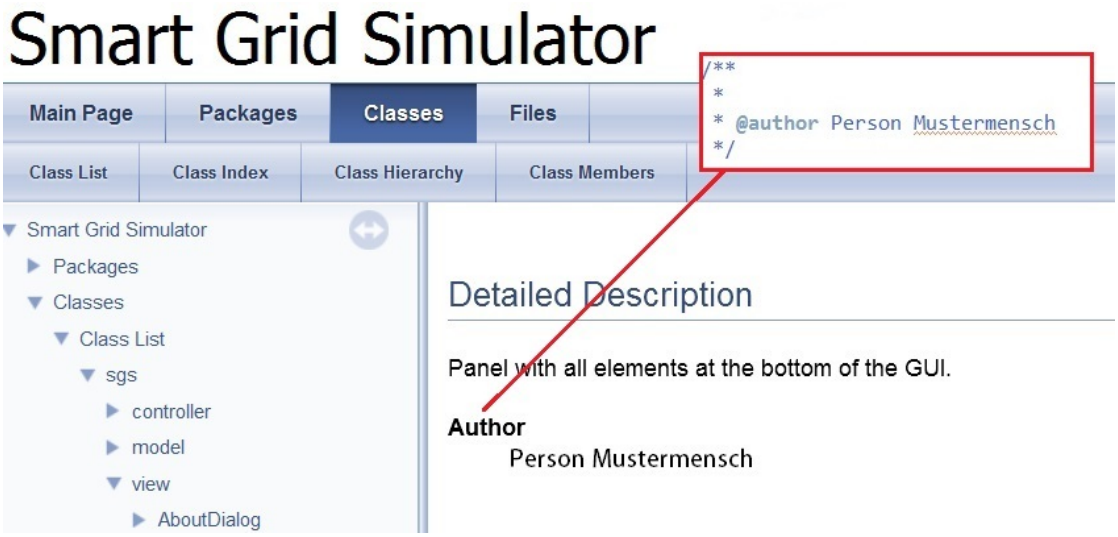

Figure 3: Notwendige Dokumentation im Code und anschließende Ergebnisdarstellung im Dokument.

3.2 @deprecated

Mit Hilfe dieses Tags werden alte Methoden beschrieben, welche nicht mehr in Verwendung sind. Alte unbrauchbare Methoden nicht zu löschen sondern nur auszukommentieren und zu dokumentieren, hat den Sinn, dass im Nachhinein immer nachvollzogen werden kann wie die Funktion sich entwickelt hat und warum alte Optionen verworfen wurden.

3.3 @exception und @throws

Diese zwei Befehle werden zur Beschreibung von der Art der Exception verwenden, welche von einer Methode geworfen werden kann.

3.4 @param

Bei Methoden und Konstruktoren welche Parameter übergeben bekommen, kommentiert ein Generator für Codedokumentationen meist selbst, die übergebenen Paramter in das Dokument (siehe Abbildung: 1).
Wird die Dokumentation wie oben beschrieben mit der Entertaste, nachdem die Methode fertig erstellt wurde, gemacht muss der Anwender meist keine weiteren Angaben mehr machen. Allerdings nur wenn der Parametername **selbsterklärend** ist.

3.5 @return

Nach dem Befehl kann eine Beschreibung hinzugefügt werden, was für ein Rückgabewert zurückgegeben wird und was er nun beinhaltet. Meist lassen sich allein durch diesen Befehl ganze Methoden beschreiben.

3.6 @see und {@link}

Beide erzeugen einen Link auf ein anderes Element der Dokumentation. Hilfreich bei sehr großen langen dokumentieren Klassen. Kann auch verwendet werden um auf andere Klassen hinzuweisen.

Während {@link} eine Verlinkung zu anderen Codeteilen überall wo der Benutzer will erlaubt, kreiert @see im Dokument für sich eine eigene Region um den verlinkten Codeteil zu präsentieren.

```
/**
 *
 * @see SgsGridModel.java
 *
 * Weitere Informationen finden sie unter {@link Sgs.GridModel.java}
 *
```

Figure 4: Verwendung Tag @see und @link im Code

See Also

SgsGridModel.java

Weitere Informationen finden sie unter Sgs.GridModel.java

Figure 5: Ergebnis der Anwendung. @see kreiert einen eigenen Bereich für die Ausgabe. @link integriert sich in den Satz.

Für eine genaue Definition der Verwendung von @link-Tag ist hier der Link.

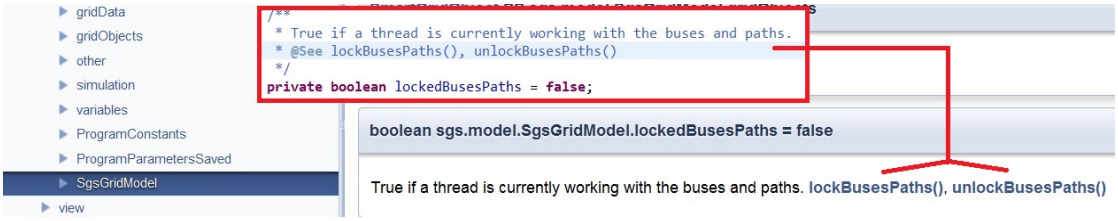

Figure 6: lockBusesPath() und unlockBusesPath() sind hier nun Links.

Eignet sich besonders gut, um einen Pfad von Aufgabenbereichen (welche Methode macht was? Welcher Methode wird nun der Wert übergeben?) darzustellen. Um evtl. schwierige Funktionen welche sich über mehrere Klassen erstreckt besser zu erläutern.

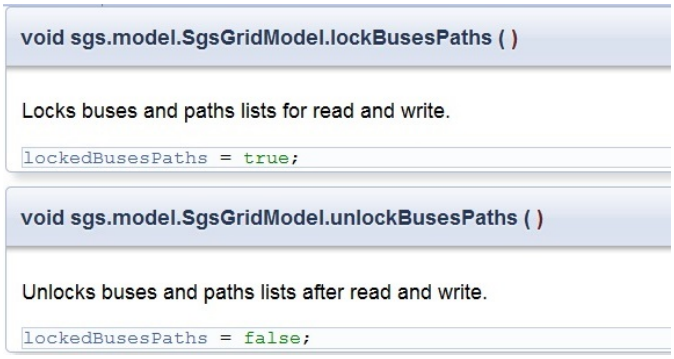

Figure 7: Verwendung der Links führt zum Ergebnis und zu weiteren Erläuterungen

3.7 @since und @version

Diese beiden Tags werden dafür verwendet um anzugeben seit wann etwas besteht. @version lässt sich nur einmal pro Klasse oder Interface angeben. Allerdings kann in jeder Klasse unterschiedliche Features zu unterschiedlichen Zeitpunkten hinzugefügt worden sein. Zur Unterscheidung seit welchem z.B Meilenstein welche Funktion besteht, wird @since verwendet.

3.8 {@code}

Dieser Tag erlaubt es, den in den Klammern vorhandenen Text (siehe Abbildung: 8) explizit als Code darzustellen. Eignet sich besonders gut, wenn komplizierte Algorithmen erklärt werden sollen, um immer die entsprechende Codezeile zu präsentieren.

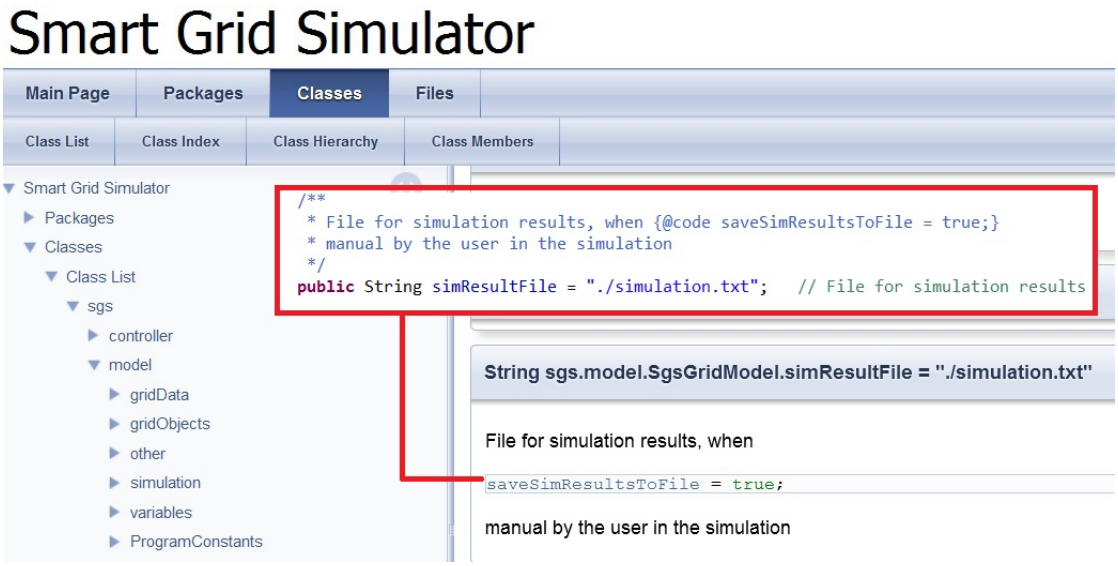

Figure 8: Der Befehl verursacht eine Hervorhebung der Sytnax.

3.9 {@literal}

Der Tag unterdrückt die Interpretierung von beinhalteten HTML oder Javadoc-Tags, sodass eine Textausgabe mit Sonderzeichen möglich wird. {@literal A<B>C} wird in der Ausgabe zu A<B>C .

4 Doxygen

Verwendet wird Doxygen Version 1.8.4, bzw. die GUI frontend Applikation Doxywizard. Sie erlaubt relativ einfach die Dokumentation aus dem Quellcode zu generieren. Um es zu verwenden, muss das Programm in seiner aktuellen Version auf dem eigenen Rechner installiert worden sein.

Es existieren **zwei** Möglichkeiten die Dokumentation automatisch als HTML generieren zu lassen. Bei der ersten Methode wird die Konfigurationsdatei geladen und ausgeführt, bei der zweiten Methode kann selbst bestimmt werden was Doxygen generieren soll und was nicht.

1. erste Möglichkeit
   - Programm "Doxywizard" starten
   - In der Menüleiste auf "File" -> "Open" gehen (siehe 9)
   - Im Workspace unter "SmartGridSimulator/src" die Datei "Doxyfile" auswählen
   - In der Registerkarte "Run" wählen (siehe 10)
   - Startbutton "Run doxygen" betätigen

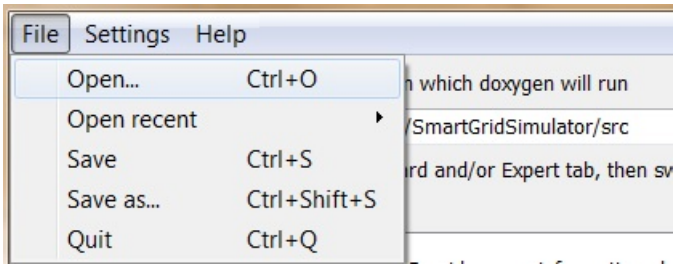

Figure 9: Laden der Konfigurationsdatei

- abwarten bis in der letzten Zeile vom Anzeigefenster "***Doxygen has finished" erscheint
- Mittels "Show HTML output" Button die Dokumentation im Browser anzeigen lassen.

2. zweite Möglichkeit
   - Anweisung von "Step 1" in Doxywizard folgen
   - Registerkarte "Wizard" -> Topic "Project"
   - Projektname "SmartGridSimulator" eingeben
   - Angeben wo sich der Quellcode befindet (hier nur den Sourceordner vom Quellcode auswählen)
   - "scan recursively" ankreuzen
   - Zielordner zum Speichern der Dokumentation angeben
   - In Topics "Mode" auswählen
   - Bei "Select the desired extraction mode:" wird "all entities" ausgewählt
   - Als nächstes muss die Programmiersprache für Java optimiert werden
   - Das Outputformat muss auf HTML mit "navigation panel" eingestellt werden

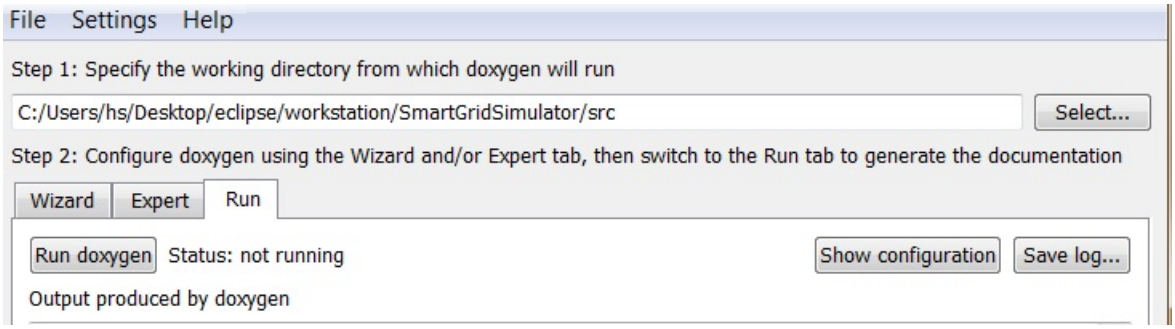

Figure 10: Doxygen starten

- Wechseln auf Registerkarte "Expert"
- Wechseln auf Topic "Build"
- Auswählen EXTRACT_ALL; EXTRACT_PRIVATE; EXTRACT_PACKAGE; EXTRACT_STATIC
- In Registerkarte "run" den Startbutton "run doxygen" wählen (siehe 10)
- Abwarten bis "***Doxygen has finishe" im Anzeigefenster in der letzten Zeile erscheint
- Mittels "Show HTML output" die Dokumentation anzeigen lassen