

SelfLinux-0.12.2



Doxygen Howto



Autor: Kim Kulling (sir_kimmi@web.de)
Formatierung: Florian Frank (florian.frank@pingos.org)
Lizenz: LGPL

Inhaltsverzeichnis

1 Einleitung

2 Doxygen

3 Vorbereitungen für den Einsatz

4 Doxygen-Tags

5 Abschliessende Bemerkungen

1 Einleitung

Um als Neueinsteiger in ein komplexes Projekt einsteigen zu können, ist eine Dokumentation beziehungsweise ein Design-Dokument unabdingbar. Vor allem bei OO-Projekten ist das zugrunde gelegte Datenmodell nur schwer aus dem Quellcode zu extrahieren. Die Klassen-Hierarchie und das zugrunde gelegte Design lässt sich nur mit viel Mühe aus dem Spagetti-Code herauslesen. Ausserdem steht man vor dem Problem, dass neue Entwickler auch neue Erweiterungen beziehungsweise Umorganisationen in der Klassenhierarchie vornehmen, diese im Design-Dokument aber vergessen werden, zu dokumentieren. Um diese Probleme zu umschiffen, wäre eine automatisch erstellte Dokumentation praktisch. An diesem Punkt setzt [doxygen](#) an.

2 Doxygen

[doxygen](#) ist ein Tool, mit dem man Dokumentationen für **C++**, **C**, **Java**, **IDL** und **C#** automatisch erstellen kann. Dabei steht [doxygen](#) unter der [GPL](#), was dem Anwender die Möglichkeit anbietet, direkt den Quellcode bearbeiten zu können. [doxygen](#) wird unter Linux entwickelt, man kann es auch unter [Windows-Systemen](#) einsetzen. Dieses Howto beschäftigt sich zunächst einmal mit der [Linux](#)-Variante.

3 Vorbereitungen für den Einsatz

Im Quellcode hinterlassen Entwickler hoffentlich Kommentare, die den Code lesbarer machen sollen. Was läge näher als diese Kommentare so aufzubereiten beziehungsweise anzulegen, dass man aus diesen eine Dokumentation erzeugen kann. Genau das ist das Prinzip von [doxygen](#). Die Kommentare im Quellcode werden nach einer gewissen Konvention bearbeitet und später von [doxygen](#) ausgelesen. Aus diesen erstellt [doxygen](#) dann seine Dokumentation. Das Format der Dokumentation wird mittels einer Konfigurationsdatei erzeugt, welche mittels

```
user@linux / $ doxygen -g <config-file>
```

erzeugt wird. Nun wird eine Konfigurationsdatei erzeugt, welche eine Menge an Konfigurationsflags setzt, die momentan allerdings zum grössten Teil uninteressant sind. Wichtig sind vor allem die folgenden Tags:

```
PROJECT_NAME      = <Name_des_Projektes>
OUTPUT_DIRECTORY = <Pfad_fuer_generierte_Doku>
OUTPUT_LANGUAGE  = <Sprache>
INPUT            = <Pfade_zu_den_Sourcen>
FILE_PATTERNS    = <Endung> zum Beispiel .h, .cpp bei einem c++ Projekt
```

Hat man die notwendigen Tags in der Konfigurationsdatei gesetzt, kann man die Dokumentation mittels des Kommandos

```
user@linux / $ doxygen <Config-File>
```

erzeugen. Unter [Windows](#) nimmt man dazu die entsprechende grafische Oberfläche.

4 Doxygen-Tags

Um die Kommentare von `doxygen` bearbeiten zu lassen, werden spezielle Dokumentations-Blöcke im Quellcode gesetzt. Diese können folgende Formen haben:

```
/**
 * ...Text...
 */

/*!
 * ...Text...
 */

/*!
 * ...Text...
 */
```

Diese Art und Weise der Kommentare ist vom C-Stil abgeleitet. Wer die C++ Kommentare bevorzugt, kann die folgenden speziellen Blöcke benutzen:

```
///
/// ...Text
///

//!
//! ...text...
//!

/////////////////////////////////////////////////////////////////
/// ...Text...
/////////////////////////////////////////////////////////////////
```

Zusätzlich ist es möglich, Kurzbeschreibungen (brief) zu setzen:

```
/*! \brief Beschreibung
 *      geht hier weiter
 *
 * Detaillierte Beschreibungen folgen hier
 */
```

Eine andere Möglichkeit stellt das Folgende dar:

```
/// Brief Beschreibung.
/** Detaillierte beschreibung folgt nach dem Punkt. */
```

Parameterbeschreibungen werden mit dem Tag `\param` eingeleitet:


```
/// Eine Methode.
/*!
 *      Detaillierte beschreibung der Funktionalität
 *      \sa Vater()
 *      \param pa1 Parameter 1
 *      \param pa2 Parameter 2
 */
```

```
int sohn(int pa1, int pa2); /// Detaillierte Beschreibung
```

Einige weitere Tags:

<code>\struct</code>	Dokumentation zu einem C-Strukt
<code>\union</code>	Dokumentation zu einer Union
<code>\enum</code>	Dokumentation zu einem Enumeration Typ
<code>\fn</code>	Dokumentation einer Funktion
<code>\var</code>	Dokumentation einer Variable
<code>\def</code>	Dokumentation eines #defines
<code>\file</code>	Dokumentation eines Files
<code>\namespace</code>	Dokumentation eines Namespaces

5 Abschliessende Bemerkungen

Wer Fragen oder Anmerkungen zu diesem Mini-Howto hat, kann diese direkt an  [Kim Kulling](mailto:Kim.Kulling) senden.