Generisk programmering - opgave 2 - Doxygen

Morten Wendelboe

Datalogisk Institut, Københavns Universitet
Universitetsparken 1, 2100 København Ø
morty@diku.dk

Resumé. Denne opgave ser på mulighederne for at anvende dokumentationssystemet Doxygen som dokumentation for kildekode skrevet i C++.

1. Introduktion

Når man udvikler programbiblioteker og i det hele taget større programmer, kan dokumentationssystemer hjælpe med at holde styr på kildekoden, og ved anvendelsen af et dokumentationssystem kan søgning efter bestemte funktionaliteter i et bibliotek gøres nemmere, da det ikke er nødvendigt at læse selve kildekoden. I denne opgave forsøger jeg at give et overblik over dokumentationssystemet Doxygen.

2. Anvendelse af Doxygen

Doxygen kan bruges til en række forskellige programmeringssprog, men her vil vi nøjes med at anvende C++. Når man skriver en ny klasse, kan man skrive kommentarer ind i tilknytning til selve klassen og dens medlemmer. Hvis man overholder en række konventioner, vil de kommentarer som man skriver blive anvendt af Doxygen til generering af bla. html-sider. Herunder kan man se et eksempel på anvendelsen af kommentarer ved erklæringen af en klasse:

```
//!  Dette er en Testklasse.
/*!
  Her staar der lidt mere forklaring til denne Testklasse.
  Den kan bruges til lidt af hvert.
*/
class Test
{
}
```

Den første linie anvendes til en kort forklaring af hvad klassen bruges til, mens de efterfølgende linier indeholder en nærmere forklaring, hvis en sådan er nødvendig.

Written 3. maj 2007.

Ved erklæringen af en funktion er det også muligt at skrive kommentarer til bestemte argumenter og at kommentere hvad funktionen returnerer. Et eksempel herpå findes i nedenstående kode:

```
//! En plus-funktion
/*! Denne funktion beregner additionen af 2 integers.*/
/*! \param a den ene integer.
    \param b den anden integer.
    \return resultatet af a+b.
*/
int plus(int a, int b);
```

Jeg har lavet en header-fil indeholdende en klasse og har indsat et par kommentarer, som overholder konvensionen i Doxygen. Det er muligt at indsætte kommentarer på flere måder, men de metoder jeg har anvendt i header-filen er de mest udbredte. Herunder kan man se header-filen:

```
#ifndef MYTESTTEST_H
#define MYTESTTEST_H

#include <string>
#include <iostream>
using namespace std;

//!  Dette er en Testklasse.
/*!
  Her staar der lidt mere forklaring til denne Testklasse.
  Den kan bruges til lidt af hvert.
*/
class Test
{
  public:

    //! en konstruktor.
    /*!
     Denne konstruktor boer ikke goere noget.
    */
    Test();

    //! En printerfunktion
    /*! Denne printerfunktion printer teksten i argumentet til konsollen. */
    /*! \param s en std::string, som bliver printet paa konsollen. */
    void printer(std::string s);

    //! En plus-funktion
    /*! Denne funktion beregner additionen af 2 integers.*/
```

```
    /*! \param a den ene integer.
        \param b den anden integer.
        \return resultatet af a+b.
    */
    int plus(int a, int b);

};
#endif
```

Når man anvender Doxygen på header-filen, kan man få genereret html-sider, et latex-dokument samt andre dokument-typer. Latex-dokumentet kan være nyttigt som en reference-manual, mens html-siderne er velegnede til søgning efter bestemte funktionaliteter eller som online dokumentation. Hvis man anvender html-dokumentationen er det muligt at klikke på hyperlinket, hvor der står "Classes". Herved får man et skærmbillede lig det, der er vist på figur 1.

Ud for hver klasse er der en kort beskrivelse af klassen. Denne beskrivelse stammer fra den første kommentarlinie i headerfilen. Hvis man klikker på Test-klassen fremkommer en side, som svarer til den, der er vist på figur 2 og figur 3.

På de nævnte figurer kan man se, at Doxygen har oplistet funktionerne på en overskuelig måde, så det fx er nemt at finde ud af, om en klassen indeholder en funktion med en bestemt opførsel.

3. Anvendelsesovervejelser

Der er en række anvendelsesmuligheder, som jeg ikke har fået undersøgt nærmere. Det drejer sig bla om at Doxygen automatisk kan generere afhængighedsgrafer, hvilket kan være anvendeligt, specielt hvis man skal sætte sig ind i klasse-designet.

Man kan enten bruge Doxygen for sig selv, eller i forbindelse med et udviklingsmiljø. Hvis man anvender Doxygen for sig selv vil Doxygen primært fremstå som et program, der kan frembringe en bedre visualisering af de kommentarer, man skriver ind i kildekoden. Anvendes Doxygen derimod i forbindelse med et udviklingsmiljø er der mulighed for, at udviklingsmiljøet kan drage fordel af Doxygens visualisering. Man kan fx forestille sig, at en bestemt funktion er kommenteret, og så er det muligt for udviklingsmiljøet at fremvise kommentarerne når programmøren skriver funktionens navn. Således kan det fx lade sig gøre at give programmøren besked om, i hvilken rækkefølge argumenterne til en funktion skal gives. Desværre kender jeg endnu ikke til nogle sådanne udviklingsmiljøer.

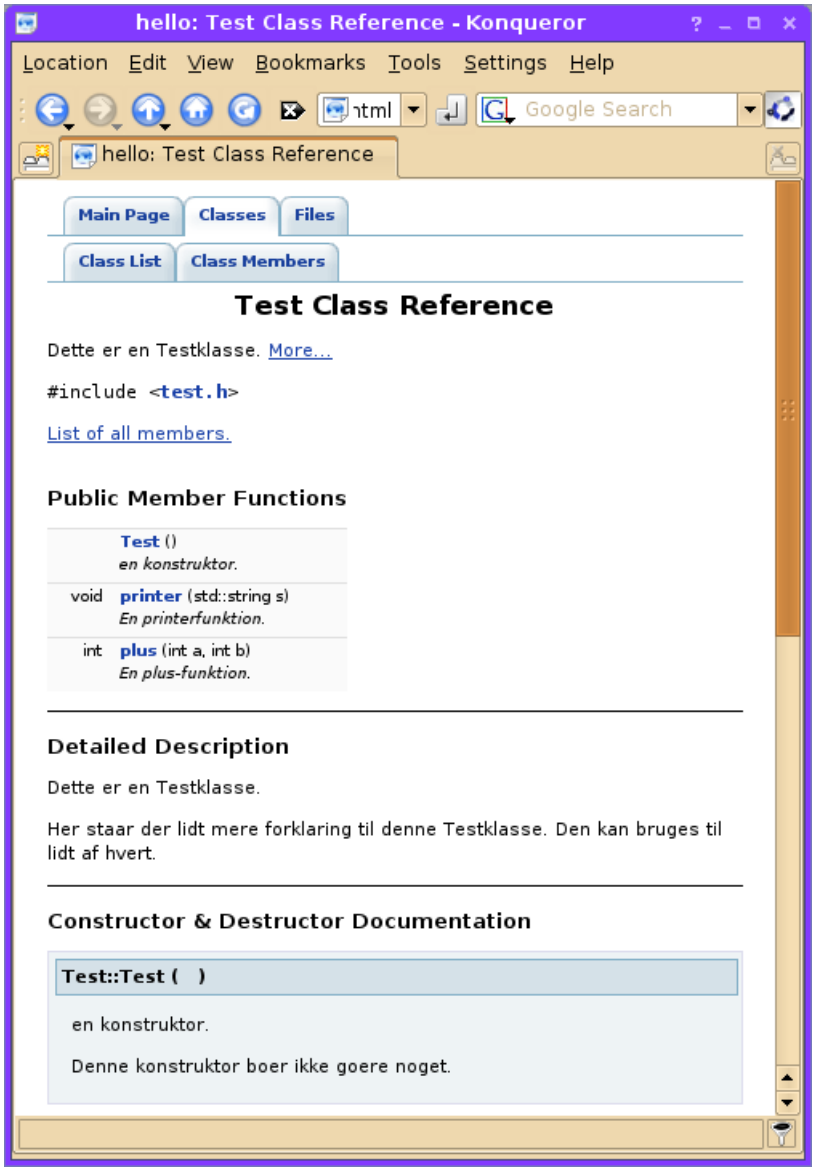
Hvis Doxygen anvendes i et udviklingsmiljø, kan man forestille sig at udviklingsmiljøet gør brugeren opmærksom på manglende kommentarer bla. i forbindelse med erklæringer af nye funktioner. Udviklingsmiljøet kunne fx undersøge, om der findes en kommentar for hver parameter og returværdien

til en funktion.

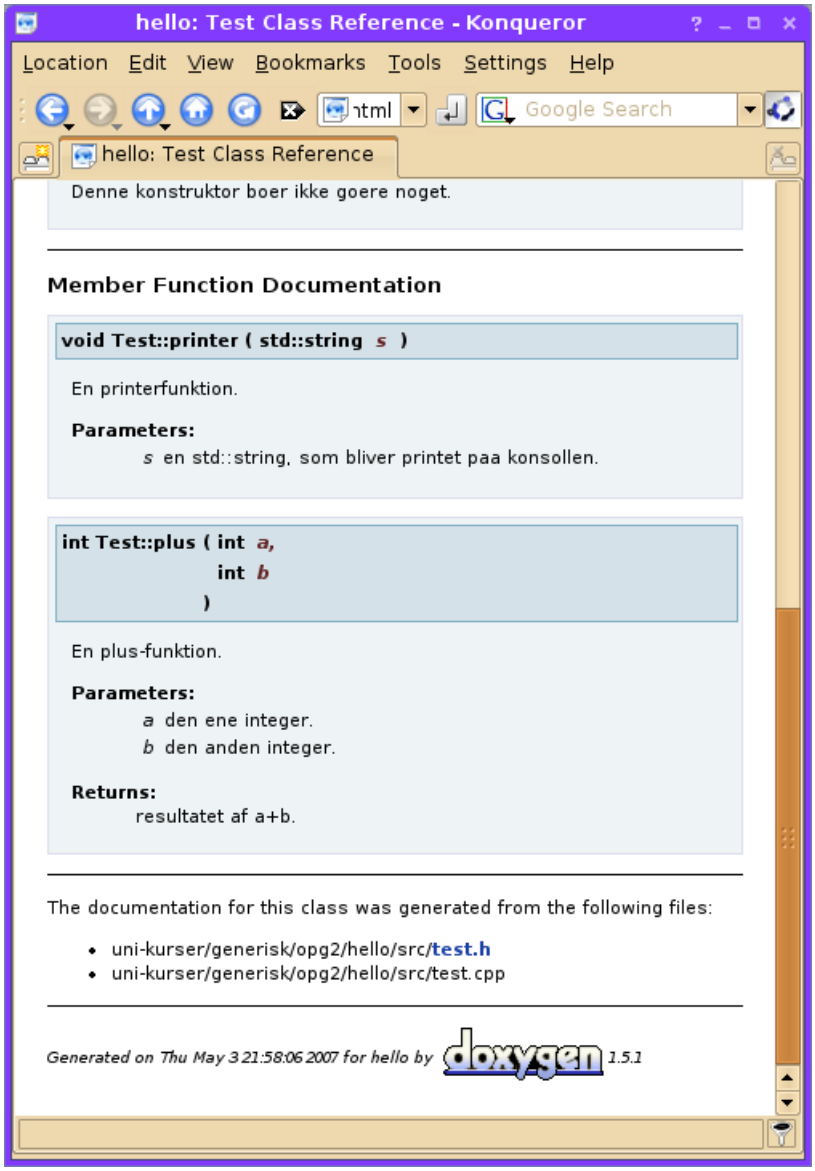
4. Konklusion

Doxygen er et dokumentationsværktøj, som gør det let at skabe et overblik over klasser og funktioner i et bibliotek, således at man ikke behøver at læse kildekoden for at finde ud af om bestemt funktion er tiltænkt en ønsket opførsel. Jeg er desværre ikke bekendt med nogen udviklingsmiljøer, som anvender Doxygen til at give brugeren kommenterende information, mens brugeren skriver kildekode.

Figur 1. Oversigt over hvilke klasser der er.



Figur 2. Første del af oversigt over Test-klassen.



Figur 3. Anden del af oversigt over Test-klassen.