Proseminar C -
Grundlagen und Konzepte

Codedokumentation

Fabian Behrendt

26. Juni 2014

Gliederung

- Best Practices
  - Code – Konventionen
  - Kommentar - „Regeln“

- astyle

- Codedokumentation
  - Doxygen
    - Funktionsweise
    - Ausführung
    - Vorteile

- Zusammenfassung

- Quellen

Best Practices - Code-Konventionen

- keine Überprüfung durch Compiler

  → dienen der besseren Lesbarkeit, Wartbarkeit, Erweiter-/ Wiederverwendbarkeit, Überprüfbarkeit, ...

- bessere Übersicht:
  - Konsistenter Klammerstil
  - Leerzeilen, wenn nötig
  - Einrückung
  - Zeichen pro Zeile

Best Practices - Code-Konventionen

- bessere Übersicht:
  - Leerzeichen, z.B.
    - zwischen Operatoren und Operanden oder Klammern

      int i = 1; if (x > 2) {…}

    - in for-Schleifen

      for (int x = 2; x < 10; x++) {…}

    - bei der Typumwandlung (type-cast)

      (double) 2

Best Practices - Kommentarstil

- Menge der Kommentare und Wichtigkeit

  int i = 1 //Initialisierung der Variablen i mit 1 ← unsinnig

- selbsterklärender Name einer Variablen spart Kommentare

- Kommentar über „Wie?“, nicht über „Was?“

astyle

- Formatierung von Quelltext nach eigenen Optionen

  (vgl. Java → Eclipse → Strg + Shift + F)

- Installation: *sudo apt-get install astyle*

- Ausführung: *astyle [options] SourceFile1 SourceFile2 …*

- Festlegen der Options-Datei: --options=...

astyle

- einige Optionen:
  - (vorgefertigt) Klammern und Einrückung: -A1 bis -A12 und -A14
  - (manuell) Einrückung (# = 2 - 20):

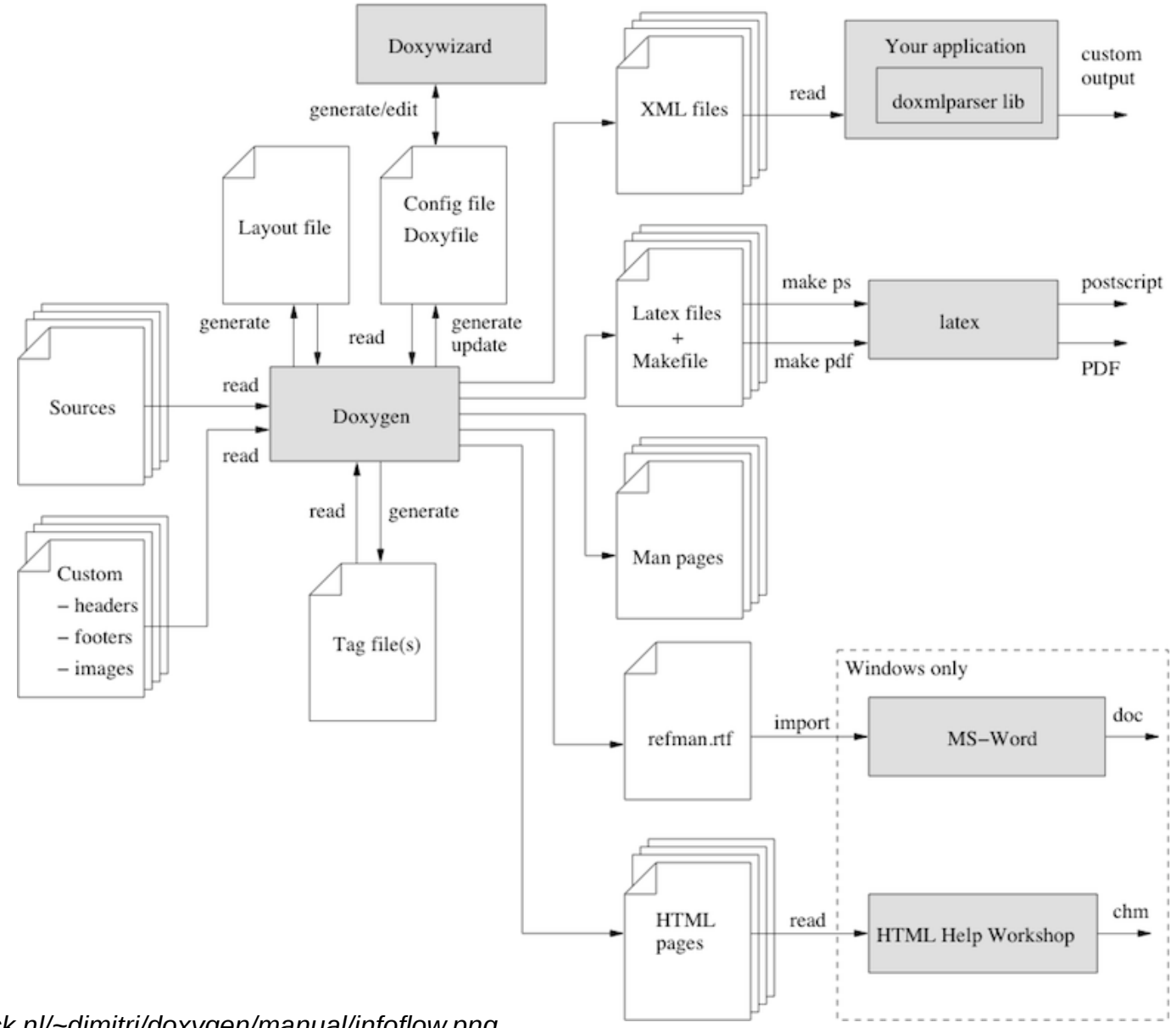
    --indent = spaces [= #] oder -s#

    --indent = tab [= #] oder -t#
  - Leerzeilen löschen:

    --delete-empty-lines oder -xe
  - Zeilenumbruch nach schließenden Klammern

    --break-closing-brackets oder -y

Codedokumentation - Doxygen-Funktionsweise



Quelle: *http://www.stack.nl/~dimitri/doxygen/manual/infoflow.png*

Codedokumentation - Doxygen-Funktionsweise

- ähnlich wie Javadoc

- Kommentare vor Funktionen:

```
/**
 * Kommentar der Funktion
 */
```

oder

```
///
///Kommentar der Funktion
///
```

Codedokumentation - Doxygen-Funktionsweise

- Spezielle Befehle mit @befehl, z.B.:
  - @mainpage Gestaltung der Startseite mit HTML-Befehlen (bspw.
, Link oder <img src = „Pfad“>
  - @brief Kurzbeschreibung der Datei

    Ausführliche Beschreibung
  - @see http://www.google.de/

Codedokumentation - Doxygen-Ausführung

- Erzeugen der Konfigurationsdatei im Terminal:

  doxygen -g

- Wichtige Einstellungen:
  - PROJECT_NAME = Name des Projektes
  - OUTPUT_DIRECTORY = Pfad für die generierte Dokumentation
  - OUTPUT_LANGUAGE = Sprache (z.B. GERMAN)
  - INPUT = Pfad/e für die Quelldateien
  - FILE_PATTERNS = Dateiendungen, z.B. .c, .java
  - GENERATE_TREEVIEW = YES

Codedokumentation - Doxygen-Ausführung

- Für UML-Diagramme:
  - HAVE_DOT = YES
  - UML_LOOK = YES

Zusammenfassung

- Redundanten Code vermeiden und Kommentare geschickt setzen
- Doxygen als Hilfsprogramm für die Dokumentation
- Astyle zur vereinfachten Bearbeitung von Fremdcode

Quellen

- http://www.stack.nl/~dimitri/doxygen/manual/starting.html
- http://www.stack.nl/~dimitri/doxygen/manual/docblocks.html
- http://www.stack.nl/~dimitri/doxygen/manual/index.html
- http://www.cypax.net/tutorials/doxygen/?l=de
- http://www.selflinux.org/selflinux/html/doxygen01.html
- http://astyle.sourceforge.net/astyle.html#_Quick_Start
- http://codebuild.blogspot.de/2012/03/10-best-practices-of-code-commenting.html