Auto-Generating Documentation & Source Code

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems
D3S

Pavel Parízek

parizek@d3s.mff.cuni.cz

CHARLES UNIVERSITY IN PRAGUE

faculty of mathematics and physics

Documentation

Types

- Developer
  - System architecture (design)
  - Code documentation
    - **API (methods, interfaces)**
    - Internally used algorithms
- User
  - Tutorials, guides, and examples
  - System administration manual

Documentation generators

- Main features
  - Extracting annotations from source code comments
  - Various output formats (HTML, PDF, LaTeX, man)
  - Generating navigation (links, references, indexes)

- Tools
  - Input: documentation written by the user as source code annotations (comments)
  - **Doxygen**, JavaDoc, NDoc, Sandcastle

Doxygen

- Supported platforms: Unix/Linux, Windows
- Languages: C/C++, Java, C#, PHP, Python, etc
- Annotations format: JavaDoc style, Qt-style
- Output formats: HTML, PDF, LaTeX, man pages

- Released under GPL (open source)

- Home page
  - http://www.doxygen.nl/

Department of Distributed and Dependable Systems

How to run Doxygen

- Creating the default configuration file

```
doxygen -g Doxyfile
```

- Generating documentation from annotations

```
doxygen Doxyfile
```

Configuration

```
PROJECT_NAME =
PROJECT_NUMBER =
PROJECT_BRIEF =

OUTPUT_DIRECTORY = <dir>

EXTRACT_ALL = YES
EXTRACT_PRIVATE = YES
EXTRACT_STATIC = YES
```

```
INPUT = <dir>
RECURSIVE = YES
FILE_PATTERNS =

GENERATE_LATEX = NO
GENERATE_TREEVIEW = YES

JAVADOC_AUTOBRIEF = YES
QT_AUTOBRIEF = YES
```

Department of Distributed and Dependable Systems

Source code annotations: JavaDoc style

```
/**
 * Returns the index of the first occurrence of the
   specified substring, starting at the given index.
 * If the specified substring is not found, then -1
   is returned. The method can also throw exception.


 * @param  str the substring for which to search
 * @param  fromIndex the index from which to start
           the search
 * @return the index of the first occurrence of
           given substring, or -1
 * @throws NullPointerException if the given
           substring is null
 */
public int indexOf(String str, int fromIndex) {
  ...
}
```

Based on the official API documentation for the `java.lang.String` class

Source code annotations: other styles

- Qt style

```
/*!
 * Returns the index of the first occurrence of the
   specified substring, starting at the given index.
 * If the specified substring is not found, then -1
   is returned.
 * \param  str the substring for which to search
 * \param  fromIndex the index from which to start
 * \return the index of the first occurrence of given
          substring, or -1
 * \throws NullPointerException if the string is null
 */
```

- C++ style

```
/// ...
/// ...
/// ...
```

Department of Distributed and Dependable Systems

Annotations

- Classes

```
/**
 * Brief description (first sentence).
 * Full details (the rest).
 */
public class MyData {
```

- Fields

```
/** description */
private int someNumber;
```

Department of Distributed and Dependable Systems

Annotations

- Methods

```
/**
 * Brief description of the method (first sentence).
 * Full details (all text up to the first command).
 * @param id description
 * @param [out] data my output argument
 * @tparam T template parameter
 * @return error code
 * @throws NullPointerException if some arg is null.
 */
public int compute(int id, char* data, T typ) {
  ...
  return 0;
}
```

Department of Distributed and Dependable Systems

Task 1

- Try out basic usage of Doxygen
  - Look into the configuration file (`Doxyfile`) to see additional options
  - Try different settings of configuration variables
  - Write documenting annotations for some program
    - Some example program from the previous lectures or your own program (any supported language)
  - Check the generated output (HTML)

References

- Links to other classes
- Links to functions
  - `function_name()`
  - `function_name(<argument list>)`
  - `class_name#function_name`
  - Example

```
/** Use the method createInput() to prepare data. */
public void myProc1(Data arg) {
```

- *See also* links

```
/**
 * This procedure evaluates the input expression.
 * @sa createInputExpr
 */
void process(Expr e) {
```

Where to put annotations

- Right before the corresponding declaration

```
/**
 * ...
 */
class MyData {
```

- Almost anywhere if you specify the name
  - file MyData.java

```
class MyData { ... }
```

  - some other file

```
/**
 * @class MyData
 * ...
 */
```

Department of Distributed and Dependable Systems

Annotating other entities

- Source code files

```
/**
 * @file mydefs.h
 * ...
 */
```

- Packages (Java)

```
/**
 * @package cz.cuni.mff
 */
```

- Namespaces (C++, C#)

```
/**
 * @namespace gui
 */
```

Formatting

- HTML commands
  - Structure: `<h1>`, `<h2>`, `<br>`, `<p>`
  - Lists: `<ul>`, `<ol>`, `<li>`
  - Font: `<b>`, `<i>`, `<code>`, `<small>`
  - Tables: `<table>`, `<td>`, `<tr>`, `<th>`
- Custom stylesheet (CSS)

Index page

```
/**
 * @mainpage Program
 * @section intro Introduction
 * some text and HTML
 * @section impl Implementation
 */
```

Doxygen: advanced topics

- Grouping annotations (modules)
- Markdown syntax (formatting)
- Mathematical formulas (LaTeX)
- Visualizing relations between code elements
  - Example: inheritance diagrams, call graphs
  - Rendering: Graphviz (the “`dot`” tool)
- Customizable output
  - layout, colors, navigation
- Linking external documents

Task 2

- Try advanced features of Doxygen
  - Links and references
  - Annotating files
  - Formatting output
  - Main page (index)

Department of Distributed and Dependable Systems

JavaDoc

- Part of the standard Java platform
- Input for the generator
  - Java source code files with annotations
  - Comment files (package, overview)
- Output formats: HTML
- Annotation must precede the code element

JavaDoc: features

- Good support for inheritance (method overriding)
  - Copying parts of annotations from superclasses
  - Linking to superclasses and interfaces

- Documenting packages
  - Option 1: `package-info.java`

    ```
    /**
     * ...
     */
    package cz.cuni.mff;
    ```

  - Option 2: `package.html`
    - File saved into the same directory as the .java source files

Running JavaDoc

- Command line

  -
    ```
    javadoc -d myapp/doc -private
       -sourcepath ./projects/myapp/src
        cz.cuni.myapp cz.cuni.myapp.util
       -subpackages cz.cuni.myapp.core
    ```

- Ant task

```
<javadoc destdir="./doc">
  <packageset dir="${src.dir}">
    <include name="cz/cuni/myapp"/>
    <include name="cz/cuni/myapp/util"/>
    <include name="cz/cuni/myapp/core/**"/>
  </packageset>
</javadoc>
```

Department of Distributed and Dependable Systems

Customizing JavaDoc output

- Doclet
  - Extract some information about input Java classes
  - Print all the information in a custom format (style)

- Taglet
  - Define custom tag that can be used in annotations
  - Generates the output of a custom tag (formatting)

Code indexing

- Purpose
  - easy navigation, code browsing and searching
- Tools
  - Ctags
    - Generates large index of names in the source code
    - Integration with many editors (Vim, Emacs, jEdit)
    - Backend for many other tools (mostly Unix/Linux)
    - Supports many languages: C/C++, Java, C#, PHP, TeX
    - http://ctags.sourceforge.net/
  - OpenGrok
    - http://opengrok.github.io/OpenGrok/

Department of Distributed and Dependable Systems

OpenGrok

- Toolset for indexing and presenting large source code repositories (Linux, NetBSD)
  - Based on Ctags
- Output
  - Set of inter-linked HTML files derived from sources
- Example
  - https://nxr.netbsd.org/

Code Generation

Code Generation

- Writing code manually
  - Hard, tedious, and time-consuming work
  - Very error prone (copy & paste mistakes)

- Automated generating (partially)
  - From simple and high-level description
  - Input: template, database, model, UML

Options

- Wizards (Eclipse, NetBeans, Visual Studio)
- Code skeletons from design models (UML)
- Parser generators (ANTLR, JavaCC, Bison)

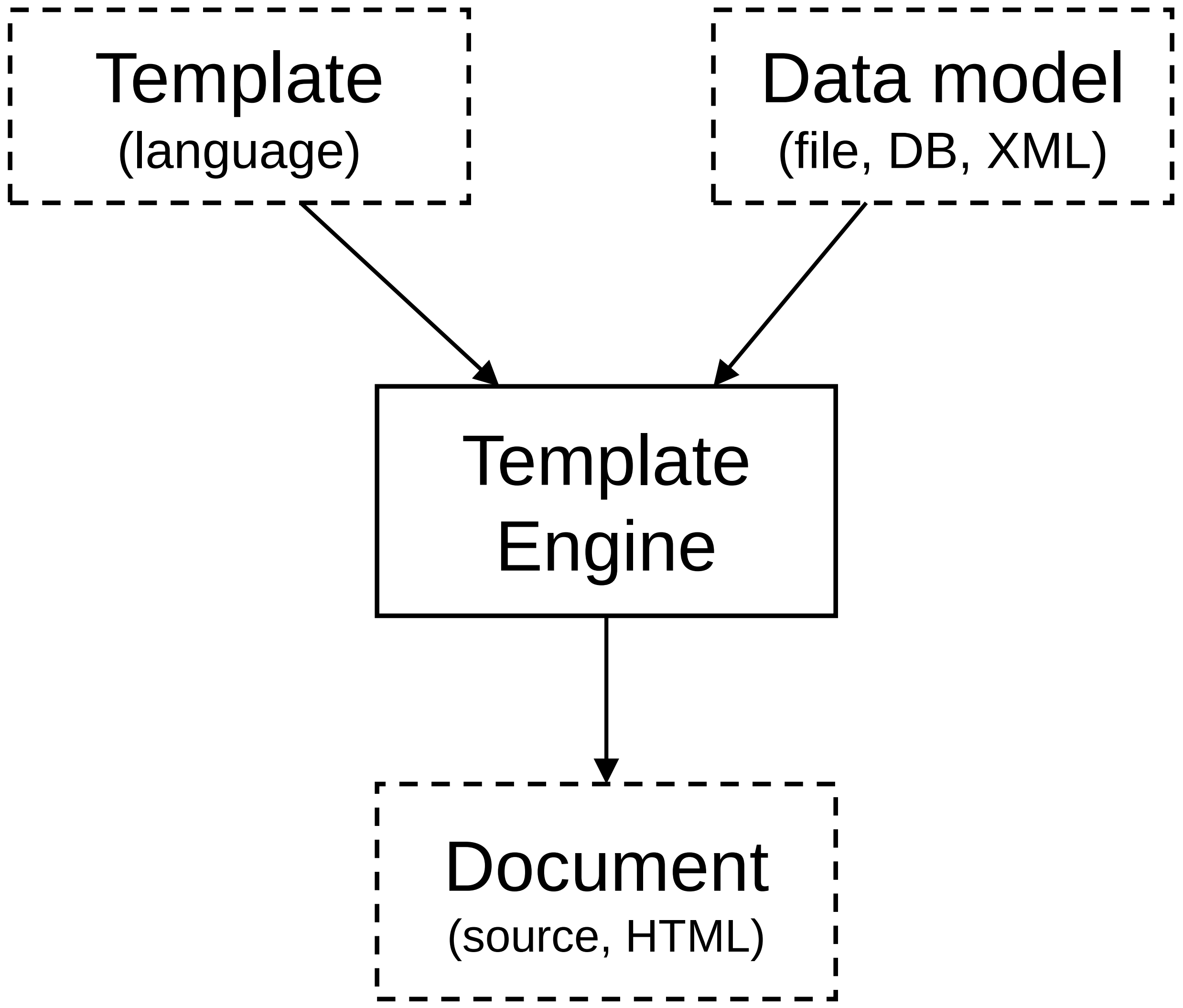
- Generating code with **template engines**

Template engines

- General programming
- Domain specific (Web)

- Tools (frameworks)
  - FreeMarker, T4, StringTemplate, AutoGen

Using template engines



FreeMarker

- General-purpose template engine
  - Open source (BSD license)
  - http://freemarker.org/
- Target platform: Java
  - Easily embeddable in Java programs
    - generic programs, Servlet and JSP containers
  - Special support for web development
    - Generating HTML pages from your templates

How to use FreeMarker

- Input
  - Template
    - Defined in the FreeMarker template language (FTL)
  - Data model
    - Prepared in the Java program

- Running
  - Template processor executed also in Java

FTL: example

```
<table>
<tr><th>Name</th><th>Salary</th></tr>
<#list employees as emp>
  <tr>
  <td>${emp.name}</td>
  <td>
  <#-- print top salaries in bold -->
  <#if (emp.salary > 2000)><b>${emp.salary * 2}</b>
  <#else>${emp.salary + 500}
  </#if>
  </td>
  </tr>
</#list>
</table>
```

FTL: other features

- Direct access to sequence elements
  - `${employees[2].salary}`
- Custom procedures and functions
  - First-class language constructs (assignable)
  - Invoking custom function: `${add(2,3)}`
- Including other files
  - `<#include “header.html”>`
- Custom directives
  - `<@mytag> ... </@mytag>`

Data model: example

```
(root)
   |
   |-- employees
   |       |-- [1]
   |       |    |-- name = “Joe Doe”
   |       |    |-- salary = 1800
   |       |
   |       |-- [2]
   |            |-- name = “John Smith”
   |            |-- salary = 2500
   |
   |-- products
  ...
```

Preparing the data model

```
Map data = new HashMap();
List employees = new LinkedList();
Map emp = new HashMap();
emp.put("name", "Joe Doe");
emp.put("salary", new Integer(1800));
employees.add(emp);
... // more employees
data.put("employees", employees);
```

Executing template processor

- Initialization of FreeMarker
- Loading template from file
- Preparing the data model
- Applying template on data

Initialization

```
Configuration cfg = new Configuration();

cfg.setDirectoryForTemplateLoading(
  new File("resources/templates")
);

cfg.setObjectWrapper(new DefaultObjectWrapper());

cfg.setDefaultEncoding("UTF-8");
cfg.setTemplateExceptionHandler(
  TemplateExceptionHandler.RETHROW_HANDLER
);
cfg.setIncompatibleImprovements(
  new Version(2,3,20)
);
```

Department of Distributed and Dependable Systems

Processing template

- Loading
  - `Template tl =`
    `cfg.getTemplate(“test.ftl”);`
- Applying
  - `FileWriter out =`
    `new FileWriter(“index.html”);`
  - `tl.process(data, out);`
  - `out.flush();`

Department of Distributed and Dependable Systems

How to define custom functions

```
public class AddMethod implements TemplateMethodModel {
  public TemplateModel exec(List args) {
    Integer op1 = new Integer((String) args.get(0));
    Integer op2 = new Integer((String) args.get(1));
    return new SimpleNumber(new Integer(op1 + op2));
  }
}
```

```
data.put(“add”, new AddMethod());
```

Task 3

- Download FreeMarker
  - http://sourceforge.net/projects/freemarker/files/freemarker/2.3.20/
  - https://freemarker.apache.org/freemarkerdownload.html

- Write template, data model, and processing code
  - Option 1: Generating classes from a list of field names and types (declarations, getters and setters, equals)
  - Option 2: Generating code that will create GUI just from a simple textual description (widgets, labels, positions)
  - Option 3: your own idea (e.g., something that you need)
- Specify data model in the Java program
- Use arbitrary output language (C#, C, Java, ...)

Department of Distributed and Dependable Systems

T4

- Text Template Transformation Toolkit
- Target platform: C#, VB (.NET)
- Support: Visual Studio, MonoDevelop
- Web: https://docs.microsoft.com/en-us/visualstudio/modeling/code-generation-and-t4-text-templates?view=vs-2015

Links

- Doxygen
  - http://www.doxygen.nl/
- JavaDoc
  - http://docs.oracle.com/javase/8/docs/technotes/tools/unix/javadoc.html
  - http://docs.oracle.com/javase/8/docs/technotes/guides/javadoc/index.html
- NDoc
  - Platform C#/.NET, documentation comments written in XML
  - http://ndoc.sourceforge.net/
  - http://ndoc3.sourceforge.net/
- Sandcastle
  - Help file builder for Windows/.NET
  - https://github.com/EWSoftware/SHFB
  - http://sandcastle.codeplex.com/
- Further information (recommended)
  - http://www.literateprogramming.com/documentation.pdf

Links

- StringTemplate
  - http://www.stringtemplate.org/
- Project Lombok
  - http://projectlombok.org/
- GNU AutoGen
  - http://www.gnu.org/software/autogen/

Homework

- Assignment
  - http://d3s.mff.cuni.cz/~parizek/teaching/sdt/
- Deadline
  - 17.12.2018 / 18.12.2018