

Documentation Generators

Steven J Zeil

March 3, 2013



Outline

- 1 **Source Code (API) Documentation**
 - javadoc
 - doxygen
 - Other Tools

- 2 **Project Reports**

- 3 **Project Websites**



Documentation Generators

... because everyone *loves* writing documentation.



Outline I

1 Source Code (API) Documentation

- javadoc
- doxygen
- Other Tools

2 Project Reports

3 Project Websites



Source Code Documentation

- For as long as people have been writing source code, they've been looking for ways to ease the effort of documenting that code.
 - Often after-the-fact
- Earliest examples were automatic flowchart generators
Generating flowcharts from source code.
 - Raw results were poor quality
 - But still could be claimed to satisfy client requirements
 - As flowcharts declined in popularity, so did the demand for these tools.
 - Still offered in reverse engineering tools (e.g.
 - Flowchart synced to code viewer
 - Human retitles blocks as “understanding” of the code progresses



API Documentation

API documentation tools are now more common

- Reflect modern emphasis on re-usable interfaces
- Combine info from
 - a (limited) language parser
 - extracts info about module/function structure and function parameters
 - and specially formatted blocks of comments embedded in the source code
 - encourages updating comments as code is modified
- Generate linked documents to facilitate browsing of referenced type names and other entities
- Some IDEs understand this markup as well and use it enhance “live” help while editing code.



javadoc

Perhaps the best known tool in this category

- part of the standard Java distribution
- achieved prominence when Sun used it to document the Java “standard library”.
 - E.g., 1.6, 1.7



Javadoc Comments

- Javadoc markup is enclosed in comments delineated by `/**...*/`
 - And therefore processed as normal comments by the Java compiler.
- A comment block precedes the entity that it describes
 - e.g., This page is generated from **SegmentationTransformer.java**.
- In addition to “free-form” text, can contain special markup



Common Javadoc Markup

- `@author` *authorName*
- `@version` *versionNumber*
- `@param` *name description*
- `@return` *description*
- `@throws` *exceptionClassName description*
- `@see` *crossReference*



Running javadoc

- Command line

```
javadoc -d destinationDir -sourcepath sourceCodeDir \  
-link http://docs.oracle.com/javase/7/docs/api/
```

- Can add multiple source paths, links to external libraries
 - Can also specify which packages from source code to document
- Eclipse: Project⇒Generate Javadoc...
 - ant

```
<javadoc packagenames="edu.odu.cs.*"  
        destdir="target/javadoc"  
        classpathref="javadoc.classpath" Author="yes"  
        Version="yes" Use="yes" defaultexcludes="yes">  
  <fileset dir="." defaultexcludes="yes">  
    <include name="extractor/src/main/java/**" />  
    <include name="generatedSource/gen-src/**" />  
    <exclude name="**/*.html" />  
  </fileset >  
  <doctitle ><![CDATA[<h1>ODU CS Extract  
                        Project </h1>]]></doctitle >  
</javadoc>
```



doxygen

- the most popular API generator for C/C++
 - Also works with Objective-C, C#, Java, IDL, Python, PHP, VHDL, and FORTRAN
- Markup is essentially identical to **javadoc**
- Output can be HTML, LaTeX, or RTF
- Can also generate
 - various non-quite-UML diagrams
 - and hyperlinked source code



Running doxygen

- Command line

```
doxygen configFile
```

The config file can contain any of a bewildering set of options in typical property-file style:

```
PROJECT_NAME = C++ Spreadsheet  
INPUT = src/model  
OUTPUT_DIRECTORY = target/doc  
EXTRACT_ALL = YES  
CLASS_DIAGRAMS = YES  
GENERATE_HTML = YES  
GENERATE_LATEX = YES  
USE_PDFLATEX = YES
```

- Eclipse: Eclox plugin
- Ant (3rd-party contributed task)



Other API Documentation Generators

The need to parse module and function structure and function parameters means that a distinct parser is needed for each programming language.

This leads to a variety of tools, e.g.,

- jsDoc for Javascript
- YARD for Ruby
- sandcastle for .Net



Outline I

1 Source Code (API) Documentation

- javadoc
- doxygen
- Other Tools

2 Project Reports

3 Project Websites



Test Reports

We've already looked JUnit, which can be used to generate test reports like this one.

This is generated in ant via the `junitreport` task:

`junitreport.xml.listing`

Other common test reports

- Javadoc of unit test code
- Coverage reports



Static Code Analyzers

Many tools that we will cover later for analyzing code can produce useful (or at least, impressive) documentation as a side effect.

- Example



Configuration Reports

Configuration managers (to be covered later) generate reports about the dependencies among the software components.

Examples:

- Maven
- Ivy



Outline I

1 Source Code (API) Documentation

- javadoc
- doxygen
- Other Tools

2 Project Reports

3 Project Websites



Project Websites

- Traditionally hand-constructed
 - Or “grown” (Wikis)
- Some build managers will generate websites linking together reports
 - Example



Forges

A *software forge* is a collection of web services for the support of collaborative software development:

- Project web sites
- Networked access to version control
 - Release (download) support
- Communications (e.g., messaging, wikis, announcements)
- Bug reporting and tracking
- Project personnel management



Forge Examples

Among the best known forges are

- the original, SourceForge, (1999)
- Google Code, (2006)
- GitHub, (2008)

The CS Dept currently runs its own installation of

- Fusion Forge
 - forked from GForge
 - forked from SourceForge

