

# Documentation Generators

Steven J Zeil

March 3, 2013

## Contents

<b>1</b>	<b>Source Code (API) Documentation</b>	<b>3</b>
1.1	javadoc . . . . .	6

1.2	doxygen . . . . .	16
1.3	Other Tools . . . . .	19
<b>2</b>	<b>Project Reports</b>	<b>20</b>
<b>3</b>	<b>Project Websites</b>	<b>27</b>



### Documentation Generators

...because everyone *loves* writing documentation.

.....

## 1 Source Code (API) Documentation

### 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.

.....



### 1.1 javadoc

#### **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

```
/**  
 *  
 */  
package edu.odu.cs.extract.control;  
  
import org.jdom.Document;  
  
import edu.odu.cs.extract.dataflow.Dataflow;  
import edu.odu.cs.extract.dataflow.QuickTransformer;
```



```
import edu.odu.cs.extract.dataflow.TransformationResult;
import edu.odu.cs.extract.inputprocessing.segmentation.SegmentationTransformer;
import edu.odu.cs.extract.utils.Properties;

/**
 * Transforms a PDF file dataflow into Raw IDM by attempting a di
 * of text PDF, but passing pages thought to be scanned on for OC
 * OCR-to-rawIDM conversion.
 *
 * @author zeil
 *
 */
public class SegmentationTransformer extends QuickTransformer {
```





```
/**
 *
 */
public SegmentationTransformer() {
    super();
}

/* (non-Javadoc)
 * @see edu.odu.cs.extract.dataflow.ThreadedTransformer#doTra
 */
@Override
public TransformationResult doTransform(Dataflow[] in) throws
    String status = "success";
```



```
String message = "OK";

IDMDataflow inputDF = (IDMDataflow) in[0];
Document unsegmentedIDM = inputDF.getDocument();
String mergeFailed = unsegmentedIDM.getRootElement().getAttributeValue("mergeFailed");

if (mergeFailed != null && "failed".equals(mergeFailed)) {
    status = "warning";
    message = "unable to merge pages from OCR";
}

// Segment document
Document segmentedIDM = new Segmentation(unsegmentedIDM).reS
```



```
IDMDataflow outputDF = new IDMDataflow (in[0].getTrace(), se  
  
/*  
    File idmOutput = null;  
    Properties p = Properties.getProperties();  
    File ocrOutDir;  
    if (p.getPropertyAsBoolean(Properties.Names.DEBUG_MODE))  
        ocrOutDir = p.getPropertyAsFile(Properties.Names.DEBUG_DIR);  
    else  
        ocrOutDir = p.getPropertyAsFile(Properties.Names.TEMP_DIR);  
    if (p.getPropertyAsBoolean(Properties.Names.SEGMENTATION_ARC  
    String idmExtension = p.getProperty(Properties.Names.SEGMENT  
    idmOutput = new File (ocrOutDir,  
        inputDF.getTrace().getName() + idmExtension);
```



```
        new IDMProxy(segmentedIDM).saveAs(idmOutput);
    }
    */
    return new TransformationResult(outputDF, status, message, nu
}

@Override
public String getOutputExtension() {
    Properties p = Properties.getProperties();
    return p.getProperty(Properties.Names.SEGMENTATION_OUT_EXT);
}
```



```
}
```

- 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 >&lt;h1&gt;ODU CS Extract</pre></div><div data-bbox="37 891 108 930" data-label="Page-Footer"><p>CS795</p></div><div data-bbox="484 891 514 930" data-label="Page-Footer"><p>15</p></div><div data-bbox="838 875 867 924" data-label="Image"><img alt="Home icon"/>A small blue icon of a house, representing a home or navigation button.</div><div data-bbox="892 882 929 928" data-label="Image"><img alt="Envelope icon"/>A small blue icon of an envelope, representing an email or contact button.</div>
```

```
Project </h1>]]></ doctitle >  
</javadoc>
```

.....

## 1.2 doxygen

### 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)



.....

### 1.3 Other Tools

#### 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

.....



## 2 Project Reports

### 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:

```
<project name="code2html" basedir="." default="build">  
  
  <record name="ant.log" action="start" append="false" />  
  
  <taskdef classpath="JFlex.jar" classname="JFlex.anttask.JFlexTask" name=  
  
  <echo>loading build-${os.name}.paths</echo>  
  <include file="build-${os.name}.paths"/>
```



```
<target name="generateSource">
  <mkdir dir="src/main/java"/>
  <jflex file="src/main/jflex/code2html.flex"
    destdir="src/main/java"/>
  <jflex file="src/main/jflex/code2tex.flex"
    destdir="src/main/java"/>
  <jflex file="src/main/jflex/list2html.flex"
    destdir="src/main/java"/>
  <jflex file="src/main/jflex/list2tex.flex"
    destdir="src/main/java"/>
</target>
```



```
<target name="compile" depends="generateSource">
  <mkdir dir="target/classes"/>
  <javac srcdir="src/main/java" destdir="target/classes"
    source="1.6" includeantruntime="false"/>
</target>

<target name="compile-tests" depends="compile">
  <mkdir dir="target/test-classes"/>
  <javac srcdir="src/test/java" destdir="target/test-classes"
    source="1.6" includeantruntime="false">
    <classpath refid="testCompilationPath"/>
  </javac>
</target>
```



```
<target name="test" depends="compile-tests">
  <property name="mypath" refid="testExecutionPath"/>
  <echo>testExecutioPath is ${mypath}</echo>
  <echoproperties/>
  <mkdir dir="target/test-results/details"/>
  <junit printsummary="yes"
    haltonfailure="yes" fork="no"
  >
    <classpath refid="testExecutionPath"/>
  <formatter type="xml"/>
  <batchtest todir="target/test-results/details">
    <fileset dir="target/test-classes">
      <include name="**/*Test*.class"/>
    </fileset>
  </batchtest>
</target>
```



```
    </fileset>
  </batchtest>
</junit>
<junitreport todir="target/test-results">
  <fileset dir="target/test-results/details">
    <include name="TEST-*.xml"/>
  </fileset>
  <report format="frames" todir="target/test-results/html"/>
</junitreport>
</target>

<target name="build" depends="test">
  <jar destfile="codeAnnotation.jar" basedir="target/classes">
```





```
    <manifest>
      <attribute name="Main-Class"
                value="edu.odu.cs.code2html.Code2HTML"/>
    </manifest>
  </jar>
</target>

<target name="clean">
  <delete dir="target"/>
</target>

</project>
```



### 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

.....

## 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

.....

