LatexDoclet Information

Robert McDermid

Mar 2, 1999

Contents

1 What is it?

When programming with Java, Sun has provided a tool which makes it possible to document source code in such a way that the documentation can be formatted into html, in an extremely readable way. All of Sun's API documentation is provided in this format, and developers writing Java code almost always document their code in this way. The tool is known as javadoc.

Starting in JDK 1.2, Sun has changed the format of the html generated by this tool. They have also provided a simple means to change or extend the documentation that is generated. This is by means of the so-called "Doclets". A Doclet is a java class that extends the class `com.sun.javadoc.Doclet`. When running javadoc normally, a standard doclet is automatically used. However, an alternative doclet may be specified on the command line.

In fact, Sun has made this mechanism so powerful, that it is possible to generate documentation in a completely different format from html. That is what this doclet does.

The format generated by this doclet is known as Latex. Latex is a macro package for Donald Knuth's extremely powerful typesetting system known as Tex. It basically allows very precise typesetting of documents in most any form desired. The file format is ascii text, and Latex is then used to compile this to a device independent dvi file. This file can then be printed or viewed online.

I was interested in doing this because sometimes I prefer to have printed documentation for my own code. In particular, when someone new joins your team, and needs to come up to speed, it is often easier for them to read through a printed document at their leisure, than wade through a mass of html. I would have liked to generate a standard word-processor format, something like RTF, because most people are more familiar with word processors these days. However, after studying the RTF spec for a while, I decided that there was no way I was going to fiddle around with that horrible mess (it actually looks like a very debased form of Tex - at least the command format is very similar). It never ceases to amaze me that Microsoft is completely unable to produce anything that is cleanly designed and easy to use.

Anyway, in order to do anything with the output of this doclet, you will need Latex. If you have access to a Unix machine, there's a good chance somebody will have already installed it. If not, it shouldn't be hard to get a copy. For OS/2, try the following URL:

`http://hobbes.nmsu.edu/pub/os2/apps/wp/tex/emtex/`

For Windows, try this URL:

`http://www.ssc.wisc.edu/~dvanness/howto.htm`

One thing to note–the output generated by this tool tends to be quite large. For example, the documentation for just the `java.util` package for JDK 1.2 is over 200 pages. So use it with care. Note the options available for reducing the size of the output.

2 Installation

Installation is very simple. Simply place the file `LatexDoclet.class` on your classpath. Also, there is a .sty file included, `fancyhdr.sty` that must be available to Latex. Use the usual techniques for adding this if you don't have it already, or simply place it in the directory where the generated .tex file goes, and it will be found automatically.

3 Usage

There are three steps to using this doclet. The first is to run javadoc on your source files to generate the Latex source code. The second is to compile the Latex source code to a dvi file. In order to do the second step, you will have to have Latex installed on your machine. The third step is to print or view the .dvi file. Again, you will need Latex for this. Consult your latex documentation for how to do this.

First of all, I will assume you are familiar with the use of javadoc to generate documentation from your source code. You normally run it with a command line like:

```
javadoc package1 package2 package3
```

or:

```
javadoc @packages
```

where *packages* is a file containing a list of all the packages you want to document. You may also specify individual class names, but I don't recommend this, as it doesn't work out quite as nicely.

In order to use the LatexDoclet, you must add the `-doclet` parameter to the javadoc command line, for example:

```
javadoc -doclet LatexDoclet @packages
```

For this to work, the LatexDoclet.class file must be on the classpath.

The LatexDoclet adds several command line options to javadoc. They are documented here:

Option	**Description**
-f file	Specifies the output file name. For example, when compiling the JDK 1.2 base documentation, I specify `-f jdk12_.tex`. It's best to use the extension tex for the output file. The default is `javadoc.tex`.
-title "title"	Specifies the title for the Latex document. The title is displayed on the title page.

Option	Description
-docauthor "author"	Specifies the author for the document. This will also be displayed on the title page.
-nodetails	Specifies that the field, constructor, and method details should not be generated. This will result in a nice summary document that makes a good quick reference.
-nosummary	Specifies that the field, constructor, and method summary tables should not be generated. You may wish to omit these to reduce document length a bit. May be used in conjunction with `-nodetails` in which case you will get a *very* concise summary document.
-twoside	Specifies that the output will be formatted for double-sided printing. This basically just means that the odd page numbers will be shifted on the page slightly towards the outer edge of the page.

Once you have the Latex file, you have to compile it. This is somewhat dependent on your Latex installation, but usually it's just a matter of typing something like:

```
latex myfile.tex
```

Generally, Latex will not be able to resolve all the cross-references on the first pass. It will output a message to this effect. In this case, you should re-run it again. Frequently, you will have to re-run it three times to get all the references sorted out. You should then generate the index. LatexDoclet includes index information. This will be in a file with the same name as your .tex file, but with the extension idx. Typically you would generate the index with a command like:

```
makeindex myfile.idx
```

which will generate a file called myfile.ind. Now, run Latex one more time to include the index. If you do not update the index, and there is an old one around, that one will be used. If you delete the myfile.ind file, then no index will be written to the DVI file.

At this point, you now have a DVI file, which can be printed or viewed online. There are also tools which can convert it to a PDF format, if you wish. For instance, this readme was originally written with Latex and then converted to PDF using such a tool. The details of all this are dependent on your Latex installation, so consult your documentation.

4 Enhancing IPFDoclet

I have tried my best to make the output of LatexDoclet fairly complete, and also to make it look nice. For example, Sun uses a lot of html tagging in it's javadoc

source–obviously this won't work in Latex. However, the doclet attempts to translate most of the common tags into Latex equivalents so, for example, lists, <code> tags, etc. should all come out ok.

However, I'm sure I've missed a few. Also, the output generated does not include all the information that the standard doclet provides. For example, there is no section in the summary listing methods inherited from base classes, and there is no deprecated section. This was a choice on my part - I don't find I use those sections that much, so I didn't bother to put them in. However, if you want to, or want to change the format of the output, I've included the complete source to LatexDoclet. You are free to modify it as you wish.

The source isn't exactly the greatest example of object-oriented programming that I've ever written. In fact, it's not object-oriented at all, all the methods are static, and there's only one class, LatexDoclet. However, I've provided javadoc comments for all the methods, and there are liberal comments through the text. It shouldn't be too hard to figure out.

The javadoc for the class is available here in dvi and pdf formats (courtesy of LatexDoclet!) and also in standard html format.