Running

RUNNING JAVADOC

Version Numbers - The version number of javadoc can be determined using **javadoc -J-version**. The version number of the standard doclet appears in its output stream. It can be turned off with `-quiet`.

Public programmatic interface - To invoke the Javadoc tool from within programs written in the Java language. This interface is in `com.sun.tools.javadoc.Main` (and javadoc is re-entrant). For more details, see Standard Doclet.

Running Doclets - The instructions given below are for invoking the standard HTML doclet. To invoke a custom doclet, use the -doclet and -docletpath options. See Doclet Overview for more information.

SIMPLE EXAMPLES

You can run javadoc on entire packages or individual source files. Each package name has a corresponding directory name. In the following examples, the source files are located at `C:\home\src\java\awt\*java`. The destination directory is `C:\home\html`.

Documenting One or More Packages

To document a package, the source files (`*.java`) for that package must be located in a directory having the same name as the package. If a package name is made up of several identifiers (separated by dots, such as `java.awt.color`), each subsequent identifier must correspond to a deeper subdirectory (such as `java/awt/color`). You may split the source files for a single package among two such directory trees located at different places, as long as `-sourcepath` points to them both – for example `src1\java\awt\color` and `src2\java\awt\color`.

You can run javadoc either by changing directories (with `cd`) or by using `-sourcepath` option. The examples below illustrate both alternatives.

- **Case 1 - Run recursively starting from one or more packages** - This example uses -sourcepath so javadoc can be run from any directory and -subpackages (a new 1.4 option) for recursion. It traverses the subpackages of the `java` directory excluding packages

rooted at `java.net` and `java.lang`. Notice this excludes `java.lang.ref`, a subpackage of `java.lang`).

- `% javadoc -d \home\html -sourcepath \home\src -subpackages java -exclude java.net:java.lang`

  To also traverse down other package trees, append their names to the `-subpackages` argument, such as `java:javax:org.xml.sax`.

- **Case 2 - Run on explicit packages after changing to the "root" source directory** - Change to the parent directory of the fully-qualified package. Then run javadoc, supplying names of one or more packages you want to document:
- `C:> cd C:\home\src\`
- `C:> javadoc -d C:\home\html java.awt java.awt.event`
- **Case 3 - Run from any directory on explicit packages in a single directory tree** - In this case, it doesn't matter what the current directory is. Run javadoc supplying `-sourcepath` with the parent directory of the top-level package, and supplying names of one or more packages you want to document:
- `C:> javadoc -d C:\home\html -sourcepath C:\home\src java.awt java.awt.event`
- **Case 4 - Run from any directory on explicit packages in multiple directory trees** - This is the same as case 3, but for packages in separate directory trees. Run javadoc supplying `-sourcepath` with the path to each tree's root (colon-separated) and supply names of one or more packages you want to document. All source files for a given package do not need to be located under a single root directory – they just need to be found somewhere along the sourcepath.
- `C:> javadoc -d C:\home\html -sourcepath C:\home\src1;C:\home\src2 java.awt java.awt.event`

Result: All cases generate HTML-formatted documentation for the public and protected classes and interfaces in packages `java.awt` and `java.awt.event` and save the HTML files in the specified destination directory (`C:\home\html`). Because two or more packages are being generated, the document has three HTML frames – for the list of packages, the list of classes, and the main class pages.

Documenting One or More Classes

The second way to run the Javadoc tool is by passing in one or more source files (`.java`). You can run javadoc either of the following two ways – by changing directories (with `cd`) or by fully-specifying the path to the `.java` files. Relative paths are relative to the current directory. The `-sourcepath` option is ignored when passing in source files. You can use command line wildcards, such as asterisk (*), to specify groups of classes.

- **Case 1 - Changing to the source directory** - Change to the directory holding the `.java` files. Then run javadoc, supplying names of one or more source files you want to document.
- `C:> cd C:\home\src\java\awt`
- `C:> javadoc -d C:\home\html Button.java Canvas.java Graphics*.java`

This example generates HTML-formatted documentation for the classes `Button`, `Canvas` and classes beginning with `Graphics`. Because source files rather than package names were passed in as arguments to javadoc, the document has two frames – for the list of classes and the main page.

- **Case 2 - Changing to the package root directory** - This is useful for documenting individual source files from different subpackages off the same root. Change to the package root directory, and supply the source files with paths from the root.
- `C:>` **`cd C:\home\src`**
- `C:>` **`javadoc -d \home\html java\awt\Button.java java\applet\Applet.java`**

  This example generates HTML-formatted documentation for the classes `Button` and `Applet`.

- **Case 3 - From any directory** - In this case, it doesn't matter what the current directory is. Run javadoc supplying the absolute path (or path relative to the current directory) to the `.java` files you want to document.
- `C:>` **`javadoc -d C:\home\html C:\home\src\java\awt\Button.java C:\home\src\java\awt\Graphics*.java`**

  This example generates HTML-formatted documentation for the class `Button` and classes beginning with `Graphics`.

Documenting Both Packages and Classes

You can document entire packages and individual classes at the same time. Here's an example that mixes two of the previous examples. You can use `-sourcepath` for the path to the packages but not for the path to the individual classes.

`C:>` **`javadoc -d C:\home\html -sourcepath C:\home\src java.awt C:\home\src\java\applet\Applet.java`**

This example generates HTML-formatted documentation for the package `java.awt` and class `Applet`. (The Javadoc tool determines the package name for `Applet` from the package declaration, if any, in the `Applet.java` source file.)

REAL WORLD EXAMPLE

The Javadoc tool has many useful options, some of which are more commonly used than others. Here is effectively the command we use to run the Javadoc tool on the Java platform API. We use 180MB of memory to generate the documentation for the 1500 (approx.) public and protected classes in the Java SE Platform, Standard Edition, v1.2.

The same example is shown twice – first as executed on the command line, then as executed from a makefile. It uses absolute paths in the option arguments, which enables the same `javadoc` command to be run from any directory.

Command Line Example

The following command line may be too long for some shells such as DOS. You can use a command line argument file (or write a shell script) to workaround this limitation.

```
C:\> javadoc -sourcepath \java\jdk\src\share\classes ^
    -overview \java\jdk\src\share\classes\overview.html ^
    -d \java\jdk\build\api ^
    -use ^
    -splitIndex ^
    -windowtitle 'Java Platform, Standard Edition 7 API Specification' ^
    -doctitle 'Java Platform, Standard Edition 7 API Specification' ^
    -header '<b>Java™ SE 7</b>' ^
    -bottom '<font size="-1">
      <a href="http://bugreport.sun.com/bugreport/">Submit a bug or
feature</a><br/>
      Copyright &copy; 1993, 2011, Oracle and/or its affiliates. All rights
reserved.<br/>
      Oracle is a registered trademark of Oracle Corporation and/or its
affiliates.
      Other names may be trademarks of their respective owners.</font>'
    -group "Core Packages" "java.*:com.sun.java.*:org.omg.*" ^
    -group "Extension Packages" "javax.*" ^
    -J-Xmx180m ^
    @packages
```

where `packages` is the name of a file containing the packages to process, such as `java.applet java.lang`. None of the options should contain any newline characters between the single quotes. (For example, if you copy and paste this example, delete the newline characters from the `-bottom` option.) See the other notes listed below.

Makefile Example

This is an example of a GNU makefile. For an example of a Windows makefile, see creating a makefile for Windows.

```
javadoc -sourcepath $(SRCDIR)               ^   /* Sets path for source files
*/
        -overview $(SRCDIR)\overview.html   ^   /* Sets file for overview text
*/
        -d \java\jdk\build\api              ^   /* Sets destination directory
*/
        -use                                ^   /* Adds "Use" files
*/
        -splitIndex                         ^   /* Splits index A-Z
*/
        -windowtitle $(WINDOWTITLE)         ^   /* Adds a window title
*/
        -doctitle $(DOCTITLE)               ^   /* Adds a doc title
*/
        -header $(HEADER)                   ^   /* Adds running header text
*/
        -bottom $(BOTTOM)                   ^   /* Adds text at bottom
*/
```

```
        -group $(GROUPCORE)                 ^   /* 1st subhead on overview
page   */
        -group $(GROUPEXT)                  ^   /* 2nd subhead on overview
page   */
        -J-Xmx180m                          ^   /* Sets memory to 180MB
*/
        java.lang java.lang.reflect         ^   /* Sets packages to document
*/
        java.util java.io java.net          ^
        java.applet

WINDOWTITLE = 'Java™ Platform, Standard Edition 6 API Specification'
DOCTITLE = 'Java™ Platform, Standard Edition 6 API Specification'
HEADER = '<b>Java™ Platform, Standard Edition 6'
BOTTOM = '<font size="-1">
      <a href="http://bugreport.sun.com/bugreport/">Submit a bug or
feature</a><br/>
      Copyright &copy; 1993, 2011, Oracle and/or its affiliates. All rights
reserved.<br/>
      Oracle is a registered trademark of Oracle Corporation and/or its
affiliates.
      Other names may be trademarks of their respective owners.</font>'
GROUPCORE = '"Core Packages" "java.*:com.sun.java.*:org.omg.*"'
GROUPEXT  = '"Extension Packages" "javax.*"'
SRCDIR = '/java/jdk/1.7.0/src/share/classes'
```

Single quotes are used to surround makefile arguments.

NOTES

- If you omit the `-windowtitle` option, the Javadoc tool copies the doc title to the window title. The `-windowtitle` text is basically the same as the `-doctitle` but without HTML tags, to prevent those tags from appearing as raw text in the window title.
- If you omit the `-footer` option, as done here, the Javadoc tool copies the header text to the footer.
- Other important options you might want to use but not needed in this example are -`classpath` and -`link`.

TROUBLESHOOTING

General Troubleshooting

- **Javadoc FAQ** - Commonly-encountered bugs and troubleshooting tips can be found on the Javadoc FAQ
- **Version number** - See version numbers.

- **Documents only legal classes** - When documenting a package, javadoc only reads files whose names are composed of legal class names. You can prevent javadoc from parsing a file by including, for example, a hyphen "-" in its filename.

Errors and Warnings

Error and warning messages contain the filename and line number to the declaration line rather than to the particular line in the doc comment.

- `"error: cannot read: Class1.java"` the Javadoc tool is trying to load the class Class1.java in the current directory. The class name is shown with its path (absolute or relative), which in this case is the same as `./Class1.java`.

ENVIRONMENT

`CLASSPATH`

Environment variable that provides the path which javadoc uses to find user class files. This environment variable is overridden by the `-classpath` option. Separate directories with a semicolon, for example:
.;C:\classes;C:\home\java\classes

Src: https://docs.oracle.com/javase/7/docs/technotes/tools/windows/javadoc.html#runningjavadoc