Contents

Preface

Audience iv
Documentation Accessibility iv
Diversity and Inclusion iv
Related Documents iv
Conventions v

1 JavaDoc Tool

JavaDoc Features 1-1
Preface

This guide provides information about using the JavaDoc tool.

Audience

This document is intended for JavaDoc tool users. Users who are developing JavaDoc content should also see the Documentation Comment Specification for the Standard Doclet for detailed information required to create JavaDoc content.

Documentation Accessibility

For information about Oracle’s commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers’ existing technologies and the need to ensure continuity of service as Oracle’s offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

- JDK 18 Documentation
- The javadoc Command for users running the tool to generate API documentation
- Documentation Comment Specification for the Standard Doclet for authors writing content for API documentation
- Javadoc Search Specification for authors writing content for API documentation
Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td><strong>monospace</strong></td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
JavaDoc Tool

The JavaDoc tool is a program that reads Java source files and class files into a form that can be analyzed by a pluggable back end, called a doclet.

To use the JavaDoc tool, you must:

- Use source code that contains Java documentation comments.
- Run the `javadoc` tool with a doclet to analyze the documentation comments and any other special tags. If you don’t specify a doclet in the command, the Standard Doclet is used by default.

The content of any generated files is specific to the doclet. The Standard Doclet generates HTML output, but a different doclet could generate other output, such as a report of misspelled words or grammatical errors.

If you specify a doclet other than the Standard Doclet, then the descriptions in this guide might not apply to the operation of that doclet or the files (if any) that are generated.

In addition to the descriptions in this guide, JavaDoc tool users and content developers should use the following documentation:

- For authors writing content API documentation: Documentation Comment Tag Specification for the Standard Doclet
- For users running the tool to generate API documentation: The javadoc Command
- For end-user readers of API documentation: The Help page, in any generated documentation. The content of the Help page will be customized for the content of the API and the command used to generate the documentation. For example, see the Help page for the Java SE and JDK API specification.

JavaDoc Features

JavaDoc features include enhanced support for code examples, search, summary pages, module system, Doclet API, HTML support, and DocLint.

Enhanced Support for Code Examples

Search

The Standard Doclet provides improved support for code examples, as described in JEP 413: Code Snippets in Java API Documentation. See Programmer's Guide to Snippets for detailed information.

When the JavaDoc tool runs the Standard Doclet, it generates output that enables users to search the generated documentation for elements and additional key phrases defined in the generated API documentation. Search results include matching characters from any position in the search string. The Search facility can also provide page redirection based on user selection.
Summary Page

The Standard Doclet may generate various additional summary pages based on detailed descriptions of individual declarations contained in the API. These pages include information about new API, deprecated API, constant values, and serialized forms. Find links to these pages in the main navigation bar at the top of each page or in the A-Z Index.

Module System

The javadoc tool supports documentation comments in module declarations. Some JavaDoc command-line options enable you to specify the set of modules to document and generate a summary page for any modules being documented. See The javadoc Command for detailed information.

Doclet API

The Doclet API uses current APIs that represent all of the latest language features. See the module jdk.javadoc for detailed information.

HTML Support

The Standard Doclet uses current web standards to generate documentation.

Note:

The Standard Doclet doesn't repair or fix any HTML errors in documentation comments. HTML errors may cause the generated API documentation to fail validation by a conformance checker.

DocLint

DocLint is a feature provided by the JavaDoc tool, as well as the JDK Java compiler, javac, to detect and report issues in documentation comments that may cause the output to be not as the author intended. The issues include missing comments, references to undeclared items (perhaps because of a spelling error), accessibility errors, malformed HTML, and syntax errors. Depending on the severity of each issue, it may be reported as either a warning or an error. See The javadoc Command for more information about DocLint.
Note:

While features like DocLint may be helpful in detecting issues, it is strongly recommended that authors always check and proofread the generated API documentation, to make sure that it is as intended.