WIKIPEDIA

Javadoc

Javadoc (originally cased **JavaDoc**)[1] is a documentation generator created by Sun Microsystems for the Java language (now owned by Oracle Corporation) for generating API documentation in HTML format from Java source code. The HTML format is used for adding the convenience of being able to hyperlink related documents together.[2]

The "doc comments" format[3] used by Javadoc is the de facto industry standard for documenting Java classes. Some IDEs,[4] like IntelliJ IDEA, NetBeans and Eclipse, automatically generate Javadoc HTML. Many file editors assist the user in producing Javadoc source and use the Javadoc info as internal references for the programmer.

Javadoc also provides an API for creating doclets and taglets, which allows users to analyze the structure of a Java application. This is how JDiff can generate reports of what changed between two versions of an API.

Javadoc does not affect performance in Java as all comments are removed at compilation time. Writing comments and Javadoc is for better understanding the code and thus better maintaining it.

Contents

History

Javadoc was an early Java language documentation generator.[5] Prior to the use of documentation generators it was customary to use technical writers who would typically write only standalone documentation for the software,[6] but it was much harder to keep this documentation in sync with the software itself.

Javadoc has been used by Java since the first release, and is usually updated upon every new release of the Java Development Kit.

The `@field` syntax of Javadoc has been emulated by documentation systems for other languages, including the cross-language Doxygen, the JSDoc system for JavaScript, and Apple's HeaderDoc.

Technical architecture

Structure of a Javadoc comment

A Javadoc comment is set off from code by standard multi-line comment tags `/*` and `*/`. The opening tag (called begin-comment delimiter), has an extra asterisk, as in `/**`.

1. The first paragraph is a description of the method documented.
2. Following the description are a varying number of descriptive tags, signifying:
    1. The parameters of the method (`@param`)
    2. What the method returns (`@return`)
    3. Any exceptions the method may throw (`@throws`)
    4. Other less-common tags such as `@see` (a "see also" tag)

Overview of Javadoc

The basic structure of writing document comments is to embed them inside `/** ... */`. The Javadoc comment block is positioned immediately above the items without any separating newline. Note that any import statements must precede the class declaration. The class declaration usually contains:

```
// import statements

/**
 * @author      Firstname Lastname <address @ example.com>
 * @version     1.6                 (current version number of program)
 * @since       1.2          (the version of the package this class was first added to)
 */
public class Test {
    // class body
}
```

For methods there is (1) a short, concise, one line description to explain what the item does. This is followed by (2) a longer description that may span multiple paragraphs. The details can be explained in full here. This section is optional. Lastly, there is (3) a tag section to list the accepted input arguments and return values of the method. Note that all of the Javadoc is treated as HTML so the multiple paragraph sections are separated by a "<p>" paragraph break tag.

```
/**
 * Short one line description.                          (1)
 * <p>
 * Longer description. If there were any, it would be   (2)
 * here.
 * <p>
 * And even more explanations to follow in consecutive
 * paragraphs separated by HTML paragraph breaks.
 *
 * @param  variable Description text text text.          (3)
 * @return Description text text text.
 */
public int methodName (...) {
    // method body with a return statement
}
```

Variables are documented similarly to methods with the exception that part (3) is omitted. Here the variable contains only the short description:

```
/**
 * Description of the variable here.
 */
private int debug = 0;
```

Note that it is not recommended[7] to define multiple variables in a single documentation comment. This is because Javadoc reads each variable and places them separately to the generated HTML page with the same documentation comment that is copied for all fields.

```
/**
 * The horizontal and vertical distances of point (x,y)
 */
public int x, y;      // AVOID
```

Instead, it is recommended to write and document each variable separately:

```
/**
 * The horizontal distance of point.
 */
public int x;

/**
 * The vertical distance of point.
 */
public int y;
```

Table of Javadoc tags

Some of the available Javadoc tags[8] are listed in the table below:

Tag & Parameter	Usage	Applies to	Since
@author *John Smith*	Describes an author.	Class, Interface, Enum	
{**@docRoot**}	Represents the relative path to the generated document's root directory from any generated page.	Class, Interface, Enum, Field, Method	
@version *version*	Provides software version entry. Max one per Class or Interface.	Class, Interface, Enum	
@since *since-text*	Describes when this functionality has first existed.	Class, Interface, Enum, Field, Method	
@see *reference*	Provides a link to other element of documentation.	Class, Interface, Enum, Field, Method	
@param *name description*	Describes a method parameter.	Method	
@return *description*	Describes the return value.	Method	
@exception *classname description* **@throws** *classname description*	Describes an exception that may be thrown from this method.	Method	
@deprecated *description*	Describes an outdated method.	Class, Interface, Enum, Field, Method	
{**@inheritDoc**}	Copies the description from the overridden method.	Overriding Method	1.4.0
{**@link** *reference*}	Link to other symbol.	Class, Interface, Enum, Field, Method	
{**@linkplain** *reference*}	Identical to {@link}, except the link's label is displayed in plain text than code font.	Class, Interface, Enum, Field, Method	
{**@value** *#STATIC_FIELD*}	Return the value of a static field.	Static Field	1.4.0
{**@code** *literal*}	Formats literal text in the code font. It is equivalent to <code>{@literal}</code>.	Class, Interface, Enum, Field, Method	1.5.0
{**@literal** *literal*}	Denotes literal text. The enclosed text is interpreted as not containing HTML markup or nested javadoc tags.	Class, Interface, Enum, Field, Method	1.5.0
{**@serial** *literal*}	Used in the doc comment for a default serializable field.	Field	
{**@serialData** *literal*}	Documents the data written by the writeObject( ) or writeExternal( ) methods.	Field, Method	
{**@serialField** *literal*}	Documents an ObjectStreamField component.	Field	

Examples

An example of Javadoc to document a method follows. Notice that spacing and number of characters in this example are as conventions state.

```
/**
 * Validates a chess move.
 *
 * <p>Use {@link #doMove(int fromFile, int fromRank, int toFile, int toRank)} to move a piece.
 *
 * @param fromFile file from which a piece is being moved
 * @param fromRank rank from which a piece is being moved
 * @param toFile   file to which a piece is being moved
```

```
 * @param toRank   rank to which a piece is being moved
 * @return            true if the move is valid, otherwise false
 * @since             1.0
 */
boolean isValidMove(int fromFile, int fromRank, int toFile, int toRank) {
    // ...body
}

/**
 * Moves a chess piece.
 *
 * @see java.math.RoundingMode
 */
void doMove(int fromFile, int fromRank, int toFile, int toRank)  {
    // ...body
}
```

See also

- Comparison of documentation generators
- .NET XML documentation comments

References

1. Now cased as 'Javadoc'. See [1] (http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html). Originally cased as 'JavaDoc'. See [2] (http://www.artima.com/intv/jackpot3.html)
2. "Archived copy" (https://web.archive.org/web/20170613233020/http://agile.csc.ncsu.edu/SEMaterials/tutorials/javadoc/). *agile.csc.ncsu.edu*. Archived from the original (http://agile.csc.ncsu.edu/SEMaterials/tutorials/javadoc/) on 13 June 2017. Retrieved 12 January 2022.
3. "javadoc - The Java API Documentation Generator" (http://docs.oracle.com/javase/1.5.0/docs/tooldocs/solaris/javadoc.html). Sun Microsystems. Retrieved 2011-09-30..
4. IntelliJ IDEA (https://www.jetbrains.com/idea/), NetBeans (http://www.netbeans-blog.org/netbeans-ide/generating-javadoc-for-a-project-in-netbeans-ide.html) Archived (https://web.archive.org/web/20170405230224/http://www.netbeans-blog.org/netbeans-ide/generating-javadoc-for-a-project-in-netbeans-ide.html) 2017-04-05 at the Wayback Machine and Eclipse (http://www.eclipse.org/)
5. "How to Write Doc Comments for the Javadoc Tool" (http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html). Sun Microsystems. Retrieved 2011-09-30..
6. Venners, Bill; Gosling, James; et al. (2003-07-08). "Visualizing with JavaDoc" (http://www.artima.com/intv/jackpot3.html). artima.com. Retrieved 2013-01-19. "When I did the original JavaDoc in the original compiler, even the people close around me pretty soundly criticized it. And it was interesting, because the usual criticism was: a good tech writer could do a lot better job than the JavaDoc does. And the answer is, well, yeah, but how many APIs are actually documented by good tech writers? And how many of them actually update their documentation often enough to be useful?"
7. "Java Platform, Standard Edition Tools Reference for Oracle JDK on Solaris, Linux, and OS X, Release 8. Section "Multiple-Field Declarations" " (https://docs.oracle.com/javase/8/docs/technotes/tools/unix/javadoc.html#JSSOR650). Retrieved 20 Dec 2017.
8. JavaSE 13 Documentation Comment Specification (https://docs.oracle.com/en/java/javase/13/docs/specs/javadoc/doc-comment-spec.html)

External links

- Java Platform, Standard Edition Javadoc Guide (https://docs.oracle.com/en/java/javase/13/javadoc/javadoc.html)

- JSR 260 (https://www.jcp.org/en/jsr/detail?id=260) Javadoc Tag Technology Update Java Specification Request (defines new Javadoc tags)
- Improve on Javadoc with ashkelon (https://web.archive.org/web/20130927133806/https://today.java.net/pub/a/today/2004/08/26/ashkelon.html)
- Globaldocs: A viewer to browse multiple Javadocs simultaneously. (http://globaldocs.info/)
- Various Java documentations converted to Windows Help format (https://javadoc.allimant.org/)
