# JAVADOC：自动生成你的程序开发文档

当初学习编程序的时候，就从来没有想过要给自己写的那几个程序编写一份完整的文档，所有的注释都仅仅是为了自己当时能够想起这段代码到底是干什么的，没有人想过这些代码的升级、共享问题。但是，开始做商业软件之后，一切都变了，尤其是大型的团队开发项目中。

大家也许注意到了，java 的 API 文档总是紧紧跟随着 JSDK 的版本的提高而保持着最新的状态。试想一下，手工维护这么复杂的文档可能吗？当然不可能，这一切都是 javadoc 这个小程序的功劳（当然也有 java 类库作者们做程序注释的一份功劳）。API 文档就是用它根据最新的源代码自动生成的，一切都是这么容易——只需要你把本来就要写的注释写得更规范一些，再稍微详细一些。然而，大家似乎好像根本就没有意识到它的存在，很少有人会用它来为自己的程序生成文档。不知道，你现在是否对它有了兴趣？好吧，下面我们就开始这个令人轻松的自动文档生成之旅。

【如何插入注释】

javadoc 为你生成代码不是凭空来的，也不是它会自动分析你的代码——每个人都有自己的代码风格，如果要进行分析翻译恐怕还是机器码更容易生成百倍。它的分析机制依赖于你按照规范为你的代码添加应有而足够的注释。只有你老老实实写注释，才有可能把以前需要做双份的工作一次做了。

Javadoc 工具可以从下列对象中提取出信息：

· 包。

· 公共类。

· 公共接口。

· 公共或者受保护的方法。

· 公共或者受保护的变量/常数。

针对每一种特性，你都应该提供一条注释。每一条注释都要以/**打头，以*/结尾。在每条/** …… */文档注释可包括任意格式的文字。它的第一个句子应该是一个总结性的语句，javadoc 会自动把它提出来制作总结页。当然，这里你完全可以使用一些 HTML 的记号，例如<I>表示斜体；...表示等宽的“打印机”字体；<B>表示粗体；甚至用包括一副图象等等。（但是不要使用类似于<title>的标题格式的标记，或者水平分隔线等，它们会和文档自己的格式发生冲突）然后在后面接上一些特殊的“标记”。每个标记以@开头，例如@author 或者@param 等等。

加入在注释中包含了指向其它文件的其它文件的链接的话（例如你的插图和图片），需要将那些文件放置在名为 doc-files 的子目录中。javadoc 会将这些目录以及其中的文件从源目录复制到文档目录。下面我们分类解释一下我们可能会比较常用的一些标记。

■常规注释

下面这些标记可以在所有文档注释中使用。

Ø @since 版本

该标记可以生成一个注释表明这个特性（类、接口、属性或者方法等）是在软件的哪个版本之后开始提供的。

Ø @deprecated 类、属性、方法名等

这个标记可以增加一条注释，指出相应的类、方法或者属性不再应该使用。javadoc 仅将首句复制到概览部分和索引中。后面得句子还可以解释为什么不鼓励使用它。有时候，我们也推荐另外一种解决办法，例如：

@deprecated Use theAnotherMethod

或者使用{@link}标记给一个推荐的连接关于它的使用我们将马上介绍。

Ø @see 链接

这个标记在“See also”（参见）小节增加一个超链接。这里的链接可以是下面几项内容：

· package.class#member label 添加一个指向成员的锚链接，空格前面是超链接的目标，后面是显示的文字。注意分隔类和它的成员的是#而不是点号，你可以省略包名或者连类名也一块省略，缺省指当前的包和类，这样使注释更加简洁，但是#号不能省略。label 是可以省略的。

· label 这个不用解释了吧。

· "Text" 这个直接在“See also”中添加一段没有链接的文字。

Ø {@link 链接目标 显示文字}

其实准确的说，link 的用法和 see 的用法是一样，see 是把链接单列在参见项里，而 link 是在当前的注释中的任意位置直接嵌入一段带超级链接的文字。

■为类和接口添加注释

类或者接口的注释必须在所有 import 语句的后面，同时又要位于 class 定义的前面。除了上面所说的常规标记以外，你还可以在类注释中使用如下标记：

Ø @author 作者名

当使用 author 标记时，用指定的文本文字在生成的文档中添加“Author”（作者）项。文档注释可以包含多个@author 标记。可以对每个@author 指定一个或者多个名字。当然你同样可以使用多个作者标记，但是它们必须放在一块儿。

Ø @version 版本

这个标记建议一个“版本”条目，后面的文字可以是当前版本的任意描述。

下面是类注释的一个例子：

/**

* An implementation of a menu bar. You add JMenu objects to the

* menu bar to construct a menu. When the user selects a JMenu

* object, its associated JPopupMenu is displayed, allowing the

* user to select one of the JMenuItems on it.

*

* For information and examples of using menu bars see

* href="http://java.sun.com/docs/books/tutorial/uiswing/components/menu.html">How to Use Menus,

* a section in The Java Tutorial.

* For the keyboard keys used by this component in the standard Look and

* Feel (L&F) renditions, see the

* JMenuBar key assignments.

*

* Warning:

* Serialized objects of this class will not be compatible with

* future Swing releases. The current serialization support is appropriate

* for short term storage or RMI between applications running the same

* version of Swing. A future release of Swing will provide support for

* long term persistence.

*

* @beaninfo

* attribute: isContainer true

* description: A container for holding and displaying menus.

*

* @version 1.85 04/06/00

* @author Georges Saab

* @author David Karlton

* @author Arnaud Weber

* @see JMenu

* @see JPopupMenu

* @see JMenuItem

*/

■方法注释

紧靠在每条方法注释的前面，必须有一个它所描述的那个方法的签名。同样除了使用常规用途的标记以外，还可以使用如下针对方法的注释：

Ø @param 变量描述

当前方法需要的所有参数，都需要用这个标记进行解释，并且这些标记都应该放在一起。具体的描述（说明）可同时跨越多行，并且可以使用 html 标记。

Ø @return 该方法的返回值

这个标记为当前方法增加一个返回值（“Returns”）小节。同样具体描述支持 html 并可跨多行。

Ø @throws 该方法抛出的异常

这个标记为当前方法在“Throws”小节添加一个条目，并会自动创建一个超级链接。具体的描述可以跨越多行，同样可以包括 html 标记。一个方法的所有 throws 都必须放在一起。

下面是一个方法注释的例子：

/**

* Returns the model object that handles single selections.

*

* @param ui the new MenuBarUI L&F object

* @return the SingleSelectionModel property

* @see SingleSelectionModel

* @see JComponent#getUIClassID

* @see UIDefaults#getUI

*/

■包和综述注释

前面的都是针对某一个类、方法等的注释，可以直接放在 JAVA 源文件中。然而为了生成一个包的注释，必须在每个包的目录下放置一个名为 package.html 的文件来对包进行描述。标签....之间的文字都会被 javadoc 自动提取出来。

也可以为所有源文件提供一个综述注释，写入名为 overview.html 文件中，将其放在所有源文件所在的父目录下面。标签 .... 之间的文字也都会被 javadoc 自动提取出来，做成文档的 Overview

【如何提取程序文档】

首先，我们还是依照惯例来看看 javadoc 的基本用法，你可以通过 javadoc -help 来获得它当前版本的具体设定细节。

javadoc [options] [packagename] [sourcefiles] [@files]

参数可以按造任意顺序排列。

· options 命令行选项。

· packagenames 一系列包的名字，空格分隔，必须分别制定想要为之建立文档的每一个包。Javadoc 不递归作用于每一个包，也不允许使用通配符。

· sourcefiles 一系列源文件名，用空格分隔。源文件名可以包括路径和通配符如“*”。

· @files 以任意次序包含包名和源文件的一个或者多个文件。当在 sourcefiles 中需要指定的文件太多的时候，就可以使用它来简化操作。目标文件是以空格或者回车来进行分隔的源文件名。

其中常用的选项有：

-d 路径

指定 javadoc 保存生成的 HTML 文件的目的目录，缺省为当前目录。

-author

在文档中包含作者信息（默认情况下会被省略）

-version

在文档中包含版本信息（在默认情况下会被省略）

-header header 文本

指定放置在每个输出文件顶部的页眉文件。该页眉文件将放在上部导航栏的右边，header 文本可以包括 HTML 标记和空格，但是如果这样就必须用引号将它括起。在 header 中的任何内部引号都不许使用转义。

-footer footer 文本

指定放置在每个输出文件底部的脚注文本。脚本将放置在下部导航栏的右边，其它同 header 一样。

-bottom text

指定放置在么个输出文件底部的文本。该文本将放置在页底，位于导航栏的下面。其它同 header 参数。

-protected

只显示受保护的和共有的类及成员，这是缺省状态。

-public

只显示公有的类和成员。

-package

只显示包、受保护的和公有的类及成员。

-private

显示所有的类和成员，如果是内部开发使用的程序文档，这个将非常有用。

-sourcepath sourcepathlist

当将包名传递给 javadoc 的时候，可以指定查找源文件（.java）的搜索路径。但必须注意，只有当用 javadoc 命令指定包名时才能使用 sourcepath 选项。如果省略 sourcepath，则 javadoc 使用 classpath 查找源文件。注意：你需要把 sourcepath 设置成目标包的源文件所在的目录，例如：你在从 c:jproject 里有一个包 cn.com.linuxaid，你想为它里面的文件生成文档，那么你就必须写成 c:jprojectcncomlinuxaid。

-clathpath clathpathlist

指定 javadoc 查找“引用类”的路径，“引用类”是值带文档的类加上它们引用的任何类。javadoc 将搜索指定路径的所有子目录。classpathlist 可以包含多个路径，它们用分号分隔。

下面我们来举一个例子：

假设，我们需要在 targetdocdir 放置我们生成的文档，需要对 c:jproject 里的 cn.com.linuxaid 包内的源文件建立程序文档。那么我们需要进入 c:jprojectcncom（也就是包含了 overview.html 的目录——假如你提供了它的话）。然后运行

javadoc -d targetdocdir cn.com.linuxaid

除了 javadoc 提供了丰富的选项参数来让你定制你所需要生成的程序文档以外，还可以借助 doclet 来产生任何形式的输出，具体的情