## 一 申明

以下内容 sort by mang

大量参考引用

了http://hi.baidu.com/johnsoncr/blog/item/0d38f18df2080011b31bbadb.html

中的内容，对作者致以最猛列的感谢

## 二 javadoc 简介

我们知道 Java 中有三种注释语句：

1.//用于单行注释。

2./*...*/用于多行注释，从/*开始，到*/结束，不能嵌套。

3./**...*/则是为支持 jdk 工具 javadoc.exe 而特有的注释语句。

javadoc 工具能从 java 源文件中读取第三种注释，并能识别注释中用@标识的一些特殊变量(见表)，制作成 Html 格式的类说明文档。javadoc 不但能对一个 java 源文件生成注释文档，而且能对目录和包生成交叉链接的 html 格式的类说明文档，十分方便。

注释中可以出现的关键字，以@开头：

| | |
|---|---|
| @author | 作者名 |
| @version | 版本标识 |
| @parameter | 参数名及其意义 |

| | |
|---|---|
| @since | 最早出现的 JDK 版本 |
| @return | 返回值 |
| @throws | 异常类及抛出条件 |
| @deprecated | 引起不推荐使用的警告 |
| @see | 交叉参考 |

## 三 例子：

以下是一个关于汽车类的例子，来具体说明运用 javadoc 命令时对注释的规范

汽车类有 4 个属性：maxSpeed averageSpeed waterTemperature Temperature

分别用来表示最大速度，平均速度，水温 室温

有 2 个方法 measureAverageSpeed() measureMaxSpeed() 用来测量汽车的平均速度和最大速度

以下例子基本包括了 javadoc 中常用到的注释，下面的篇幅中会一一详细说明的，读者可以试着用 javadoc 命令生成以下例子的说明文档

```
/**
*汽车类的简介
*<p>汽车类具体阐述第一行<br>
*汽车类具体阐述第二行
*@author man
*@author man2
*@version 1.0
*@see ship
*@see aircraft
```

```
*/
public class Bus{
/**
*用来标识汽车行驶当中最大速度
*@see #averageSpeed
*/
    public int maxSpeed;
/**用来标识汽车行驶当中平均速度*/
    public int averageSpeed;
/**用来标识汽车行驶当中的水温*/
    public int waterTemperature;
/**用来标识天气温度*/
    public int Temperature;
    Bus(){
    }
/**
*该方法用来测量一段时间内的平均速度
*@param start 起始时间
*@param end 截止时间
*@return 返回 int 型变量
*@exception java.lang.exceptionthrowwhenswitchis1
*/
    public int measureAverageSpeed(int start,int end ){
        int aspeed=12;
        return aspeed;
    }
/**
*该方法用来测量最大速度
*/
    public int measureMaxSpeed(){

    }
}
```

## 四 javadoc 命令对注释规范的详细介绍

只有/**……*/这样的诠释才能被写入 javadoc 文档。

## 4.1 java文档和javadoc

在 jdk 的 bin 目录下你可以找到 javadoc，如果是 windows 下的 jdk，它的文件名为 javadoc.exe。运用 javdoc 编译 .java 源文件时，它会读出 java 源文件中的文档诠释，并遵照一定的限定和 java 源程序一起进行编译，生成文档。

引见 javadoc 的编译命令之前，还是先懂得一下文档诠释的款式吧。不过为了能够编译下面提到若干例子，这里先引见一条 javadoc 命令：

javadoc -d 文档寄存目录 -author -version 源文件名.java

这条命令编译 1 个名为“源文件名.java”的 java 源文件，并将生成的文档存放在“文档寄存目录”指定的目录下，生成的文档中 index.html 不外乎是文档的首页。-author 和-version 二上选项可以省略。

## 4.2 文档诠释的款式

**文档诠释可以用于对类、属性、方式等进行阐明**。写文档诠释时除了需要运用/**....*/规定之外，还需要注意诠释内部的有些细节毛病。

### 4.2.1 文档和文档诠释的格式化

生成的文档系 html 款式，而这些 html 款式的标识符并非 javadoc 加的，而是咱们在写诠释的时候写上去的。打个比方，需要换行时，不是敲入 1 个回车符，而是写入＜br＞，要是要分段，就该当在段前写入＜p＞。

因而，格式化文档，不外乎在文档诠释中添加相应的 html 标签。

文档诠释的正文并非直接复制到输出文档(文档的 html 文档)，而是读出每一行后，删掉前导的*号及*号从前的空格，再输入到文档的。如

```
/**
*thisisfirstline.＜br＞
*****thisissecondline.＜br＞
thisisthirdline.
*/
```

编译输出后的 html 源码则是

```
thisisfirstline.＜br＞
thisissecondline.＜br＞
thisisthirdline.
```

前导的*号准许持续运用不止一个，其成果和运用 1 个*号一致，但不止一个*号前不可有其它字符分隔，不然分隔符及背后的

*号都将作为文档的内容。*号在这里系作为左边陲运用，如上例的第一行和第二行；要是没有前导的*号，则边陲从第一个有效字符开端，**而不包括前面的空格**，如上例第四行。

还有一点需要阐明，**文档诠释只阐明紧接其后的类、属性或者方式**。如下例：

```
/**comment for class*/
public class Test{
/**comment for a attribute*/
int number;
/**comment for a method*/
public void mymethod(){......}
......
}
```

上例中的三处诠释不外乎区别对类、属性和方式的文档诠释。它们生成的文档分别是阐明紧接其后的类、属性、方式的。“**紧接**”二字特别首要，要是忽视了这一点，就很可能造成生成的文档故障。如

```
import java.lang.*;
/**comment for class*/
public class Test{......}
//此例为准确的例子
```

这个文档诠释将生成准确的文档。但只需要转变其中两行的

位置，化成下例，就会出错：

```
/**comment for class*/
importjava.lang.*;
public class Test{......}
//此例为故障的例子
```

这个例子只把上例的 import 语句和文档诠释局部交换了位置，效果却不尽相同——生成的文档中基本就找不到上述诠释的内容了。缘故何在？

“/**comment for class*/”系对 Test 类的阐明，把它放在“public class Test{......}”之前时，其后紧接着 class Test，吻合限定，因此生成的文档准确。可是把它和“importjava.lang.*;”改换了位置后，其后紧接的不再是类 Test 了，而是一个 import 语句。因为文档诠释只能阐明类、属性和方式，import 语句不在此列，因此这个文档诠释便被当成故障阐明省略掉了。

### 4.2.2 文档诠释的三局部

依据在文档中卖弄的成果，文档诠释分为三局部。先举例如下，以便阐明。

```
/**
*汽车类的简述.
*<p>汽车类具体阐述第一行<br>
```

```
*汽车类具体阐述第二行
*@author man
*@author man2
*@version 1.0
*@see ship
*@see aircraft
*/
public class Bus{
//类中的语句……
}
```

第一部分系简述，也就是注释中的的第二行。文档中，关于属性和方式全是先有一个列表，然后才在背后一个一个的仔细的阐明。列表中属性名或者方法名背后那段阐明不外乎简述。

简述局部写在一段文档诠释的最前面，第一个点号(.)之前(包含点号)。换句话说，不外乎用第一个点号分隔文档诠释，之前系简述，后来系第二局部和第三局部。如上例中的“*汽车类的简述.”。

有时，即使正确地以 1 个点号作为分隔，javadoc 依然会出错，把点号背后的局部也做为了第一部分。为了解决这个毛病，咱们可以运用一个＜p＞标记将第二部分离开为下一段，如上例的“*＜p＞show 方式的仔细阐明第一行....”。除此之外，咱们也可以运用＜br＞来分隔。

第二局部系仔细阐明局部。该局部对属性或者方式进行仔细的阐明，在格式上没有什么特异的哀求。

第三局部系特异阐明局部。这局部包含版本阐明、参数阐明、返回值阐明等。

#### 4.2.3 运用 javadoc 记号

javadoc 记号系插入文档诠释中的特异记号，它们用于标识编码中的特异引佣。javadoc 记号由“@”及其后所跟的记号类型和专用诠释引佣组成。记住了，三个局部——@、记号类型、专用诠释引佣。不过我甘愿把它分成两部分：@和记号类型、专用诠释引佣。 注意**@和记号类型最好紧挨着一块写**

javadoc 记号有如下这些：

@author 对类的阐明 说明开垦该类模块的笔者

@version 对类的阐明 说明该类模块的版本

@see 对类、属性、方式的阐明 参照转向，也就是相关主题

@param 对方式的阐明 对方式中某参数的阐明

@return 对方式的阐明 对方式返回值的阐明

@exception 对方式的阐明 对方式也许抛出的非常进展阐明

下面仔细阐明各记号。

##### **4.2.3.1 @see的运用**

@see 的句法有三种：

@see 类名

@see #方法名或属性名

@see 类名#方法名或属性名

注意 see 后面有空格

类名，可以依据需要只写出类名(如 string)或者写出类全名(如 java.lang.string)。那么什么时候只需要写出类名，什么时候需要写出类全名呢？

要是 java 源文件中的 import 语句包括了的类,可以只写出类名，要是没有包括,则需要写出类全名。java.lang 也曾经默认被包括了。这和 javac 编译 java 源文件时的规矩一致,因此可以单纯的用 javac 编译来判断，源程序中 javac 能找到的类，javadoc 也一定能找到；javac 找不到的类，javadoc 也招不到，这就需要运用类全名了。

方法名或者属性名，如果是属性名，则只需要写出属性名即可；如果是方法名，则需要写出方法名以及参数类型，没有参数的方式，需要写出一对括号。如

成员类型 成员名称 及参数 @see 句法

属性 number：@see number

属性 count：@see count

方式 count()：@see count()

方式 show(boolean b)：@see show(boolean)

方式 main(string[]args)：@see main(string[])

有时也可以躲懒：假使上例中，没有 count 这 1 属性，那么参照

方式 count()就可以简写成@see count。不过，为了安全起见，还是写全@see count()比较好。

@see的第二个句法和第三个句法全是转向方式或者属性的参照，它们有什么分辨呢？

第二个句法中没有指出类名，则默以为目前类。因此它定义的参照，全转向本类中的属性或者方式。而第三个句法中指出了类名，则不错转向其它类的属性或者方式。

对于@see 记号，咱们举个例阐明。因为@see 在对类阐明、对属性阐明、对方式说明时用法全一致，因此这里只以对类阐明为例。

```
/**
*@see string
*@see java.lang.stringbuffer
*@see #str
*@see #str()
*@see #main(String[])
*@see object#tostring()
*/
public class TestJavadoc{
    public int str;
    public void str(){
    //.....
    }
    public static void main(String args[]){
```

```
    //.....
    }
}
```

string 和 stringbuffer 全是在 java.lang 包中，因为这个包系默认导入了的，因此这两个类可以直接写类名，也可以写类全名。str、str()为同名属性和方式，所以方法名需要用()划分。main 系带参数的方式,因此在()中指明了参数类型,注意 main 函数中的参数 String 是大写，若写错或写成小写则不能生成链接。tostring()固然在本类中也有(从 object 继承的),但咱们系想参照 object 类的 tostring()方式，因此运用了 object#tostring()。

古怪的是，为什么其中只有 str、str()和 main(string[])化成了链接呢？那是因为编译时没有把 java.lang 包或者 stirng、stringbuffer、object 三个类的源文件一起参加编译，因此，生成的文档没有对于那三个类的信息，也就不可以建立链接了。背后讲授 javadoc 编译命令的时候还会仔细阐明。

上例中要是把类中的 str 属性去掉，那么生成的文档又会有什么变迁呢？你会发觉，原先系 str,str()，而如今化成了 str(),str()，由于 str 属性已经没有了，因此 str 也表现方法 str()。

#### 4.2.3.2 运用@author、@version阐明类

这两个记号区别用于指明类的笔者和版本。缺省情况下 javadoc 将其忽视，但命令行开关-author 和-version 可以修正这个功效，使其包括的信息被输出，即在使用 javadoc 命令时加上相关参数

如：javadoc -d [存放路径] -author -version 类名.java

这两个记号的句法如下：

@author笔者名

@version版本号

其中，@author可以不止一次运用，以指明不止一个笔者，生成的文档中每个笔者证明运用逗号(,)隔开。@version也可以运用不止一次，**但只有首次有效**，生成的文档中只会卖弄首次运用@version指明的版本号。如下例

```
/**
*@author fancy
*@author bird
*@version version1.00
*@version version2.00
*/
public class TestJavadoc{
}
```

从生成文档的图示中可以看出，两个@author语句全被编译，在文档中生成了笔者列表。而两个@version语句中只要第一句被编译了，只生成了 1 个版本号。

从图上看，笔者列表是以逗号分隔的，要是偶想分行卖弄怎么办？此外，要是偶想卖弄两个以上的版本号又该怎么办？

——咱们可以将上述两条@author语句合为一句，把两个@version语句也合为一句：

@authorfancy＜br＞bird

@versionversion1.00＜br＞version2.00

咱们这么做即达到了目标，又没有弄坏限定。@author后来的笔者名和@version后来的版本号都可以系用户俺定义的所有html款式，因此咱们可以运用＜br＞记号将其分行卖弄。同时，在 1 个@version中指明两个用＜br＞分隔的版本号，也没有弄坏只卖弄第一个@version内容的限定。

#### 4.2.3.3 运用@param、@return和@exception阐明方式

这三个记号全是只用于方法的。@param描写方式的参数，@return描写方式的返回值，@exception 描写方式也许抛出的异常。它们的句法如下：

@param 参数名参数阐明

@return 返回值阐明

@exception 异常类名阐明

每一个@param 只能描写方式的 1 个参数，因此，要是方式需要不止一个参数，就需要不止一次运用@param 来描写。

1 个方式中只能用 1 个@return，要是文档阐明中列了不止一个@return，则 javadoc 编译时会发出正告，且只要第一个@return 在生成的文档中有效。

方法也许抛出的异常应该用@exception 描写。因为一个方式也许抛出不止一个非常，因此可以有不止一个@exception。每个@exception 背后应有简述的异常类名，阐明中应指出抛出异常的缘故。需要注意的系，异常类名该当依据源文件的 import 语句肯定系写出类名还是类全名。　　示例如下：

```
public class TestJavadoc{
/**
*@param naswitch
*@param bexcrescentparameter
*@return trueorfalse
*@return excrescentreturn
*@exception java.lang.exceptionthrowwhenswitchis1
*@exception nullpointerexceptionthrowwhenparameternisnull
*/
public Boolean fun(integern)throwsexception{
switch(n.intvalue()){
case0:
break;
case1:
thrownewexception("testonly");
default:
returnfalse;
}
returntrue;
}
}
```

可以看到，上例中@param bexcrescentparameter 一句系过剩的，由

于参数只是 1 个 n,并没有一个 b 可是 javadoc 编译时只是没有检讨。因而,**写文档诠释时一定要准确匹配参数表和方式中正式参数表的项目**。要是方式参数表中的参数系 a，文档中却给出对参数 x 的说明，或者再多出 1 个参数 i，就会让人摸不着头脑了。@exceptin 也是一致。

上例顺序中只是没有抛出 1 个 nullpointerexception，可是文档诠释中为什么要写上这样一句呢，莫非又是为了演示？这不是为了演示描写过剩的非常也能经过编译，而是为了阐明写异常说明时应考运行时(runtime)非常的可能性。上例顺序中，要是参数 n 系给的 1 个空值(null)，那么顺序会在运行的时候抛出 1 个 nullpointerexception，因而，在文档诠释中添加了对 nullpointerexception 的阐明。

上例中的@return 语句有两个，可是依据限定，同一个方法中，只要第一个@return 有效，其他的会被 javadoc 忽视。因此生成的文档中没有浮现第二个@return 的描写。

讲到这里，该怎样写文档诠释你该当曾经清晰了，下面就开端讲授 javadoc 的常用命令。

**以下几点还须再次注意**

1) 在对属性，方法进行说明时要加上属性或方法的权限，否则在生成的文档中根本找不到该属性或方法

2) #是针对方法或属性，换句话说要 see 方法名或属性名 时才用到#对#的要求:

- 要 see 的类或方法或属性必须存在，否则就变成了 #类名或方法名或属性名 了
- 在类里面对属性或方法注释时一般还是把#加上，这样生成的 see 可以产生链接，但是对类的注释则没有这种限制了

3) 运用@author、@version 说明类后，在使用命令 javadoc 时一定要注意相关参数的使用：

Javadoc －d [存放路径] －author －version 类名.java

若没有使用 －author －version 参数 ，即使你在文档中用了 @author @version， 生成的文档中也是没有这些信息的

## 五 javadoc命令

运行 javadoc-help 可以看到 javadoc 的用法，这里列举常用参数如下：

用法：

javadoc[options][packagenames][sourcefiles]

选项：

-public 仅卖弄 public 类和成员

-protected 卖弄 protected/public 类和成员(缺省)

-package 卖弄 package/protected/public 类和成员

-private 卖弄一切类和成员

-d＜directory＞输出文档的目的目录

-version 包括@version 段

-author 包括@author 段

-splitindex 将索引分为每个字母对应 1 个文档

-windowtitle＜text＞文档的浏览器窗口题目

javadoc 编译文档时可以给定包列表，也可以给出源程序文档列表。譬如在 classpath 下有两个包若干类如下：

fancy.editor

fancy.test

fancy.editor.ecommand

fancy.editor.edocument

fancy.editor.eview

这里有两个包(fancy 和 fancy.editor)和 5 个类。那么编译时(windows 环境)可以运用如下 javadoc 命令：

javadoc fancy\test.java fancy\editor.java fancy\editor\ecommand.java fancy\editor\edocument.java fancy\editor\eview.java

这是给出 java 源文件作为编译参数的方式，注意命令中指出的

系文档路径，该当依据实际情况转变。也可以系给出包名作为编译参数，如：

javadoc fancyfancy.editor

用浏览器打开生成文档的 index.html 文档即可发觉两种办法编译效果的不同.

用第二条命令生成的文档被框架分成了三局部：包列表、类列表和类阐明。在包列表中选择了某个包后来，类列表中就会列出该包中的一切类；在类列表中选择了某个类后来，类阐明局部就会显示出该类的仔细文档。而用第一条命令生成的文档只要两部分，类列表和类阐明，没有包列表。这不外乎两种办法生成文档的最大分辨了。

两种办法编译还有一点不同，

下面再来细说选项。

-public、-protected、-package、-private 四个选项，只需要任选其一即可。它们指定的卖弄类成员的水平。它们卖弄的成员多少系 1 个包括的关系，如下表：

-private(卖弄一切类和成员)

-package(卖弄 package/protected/public 类和成员)

-protected(卖弄 protected/public 类和成员)

-public(仅卖弄 public 类和成员)

-d 选项准许你定义输出目录。要是不必-d 定义输出目录，生成的文档文档会放在目前目录下。-d 选项的用法系

-d 目录名

目录名为必填项，也就是说，要是你运用了-d 参数，就一定要为它指定一个目录。。

-version 和-author 用于节制生成文档时是否生成@version 和@author 指定的内容。**不加这两个参数的情况下，则生成的文档中不包括版本和笔者信息。**

-splitindex 选项将索引分为每个字母对应 1 个文档。默认情况下，索引文档只有一个，且该文档中包括一切索引内容。那是生成文档内容比较少的时候，这么做十分适合，可是，要是文档内容十分多的时候，这个索引文档将包括十分多的内容，显得过于宏大。运用-splitindex 会把索引文档按各索引项的第一个字母进展分类，每个字母对应 1 个文档。这么，就减少了 1 个索引文档的累赘。

-windowtitle 选项为文档指定 1 个题目，该题目会卖弄在窗口的标题栏上。如果不指定该题目，而默认的文档标题为“生成的文档（无题目）”。该选项的用法系：

-windowtitle 题目

题目系一串没有包括空格的文本，由于空格符系用于分隔各参数的，因此不可包括空格。同-d 相似，要是指定了-windowtitle 选项，则一定指定题目文本。

到此为止，java 文档和 javadoc 就引见完了。