Study Annotation -terry

11/10/2014

<<<<<Annotation in java>>>>>

<The Format of an Annotation>

形式最简单的注释就像这样 @Entity 在@这个符号指示下跟的就是一个注释

注释可以包括一些元素

@Author(name = “Benjamin Franklin”,

data = “3/27/2003”)

@SuppressWarnings(value = “unchecked”)

有时候如果只有一个元素被命名了值，那么名字就可以被省略

@SuppressWarnings(“unchecked”)

也有可能有多个注释在同一个声明上，如果说这多个注释是同一个类型的话，那么这就叫做重复注释（repeating annotation）

@Author(name = “lijun”)

@Author(name = “john”)

Java SE 8支持重复注释

注释的种类定义在了jse API的 java.lang 和 java.lang.annotition里面

<Where Annotation Can Be Used>

注释可以被应用到类，域，方法，和其他程序元素的声明当中

<Declaring an Annotation Type>

许多注解（annotation）使用来替代代码中的注释的（comments）

假如说下面这个注释很重要每一个类里面都要用

public class Terry(){

//Author: junli

//Date: 11/10/2014

//Vision:1.0.0

}

那么就可以把这种相同样子定义成一个注解类型

@interface ClassStart{

String Author();

String Date();

String Vision();

}

注解类型是一种形式的接口，定义好了注解类型之后，你就能这么用

@ClassStart（

anthor = “junli”,

date = “11/10/2014”,

Vision = “1.0.0”）

public class XXX(){}

如果说你要让你注解类型里面的信息显示在javadoc生成文档中的话，你就必须将你的@ClassStart注解定义成是@Documented注解，就是在定义注解类型的时候在上面一行加上@Documented.

@Documented

@interface ClassStart{

}

<Predefined Annotation Types>

一系列的注解类型都是在java se api中提前定义好的，有些注解类型是被java编译器使用的，有些是被其他注解使用的

在 java.lang 中定义好的注解有 @Deprecated @Override @SuppressWarnings

@Deprecated 的意思就是标记的元素是不推荐的,不应该再被使用

// Javadoc comment follows

/\*\*

\* *@deprecated*

\* *explanation of why it was deprecated*

\*/

**@Deprecated**

static void deprecatedMethod() { }

}

@Override注解就是告诉编译器，这个元素是重写的父类中的元素。假如说你注解的元素不是重写的父类的元素的话，就会报错

@SuppressWarnings 就是告诉编译器不报告本来应该要报告的警告

每个编译器的警告都属于一个门类（category），java规范中列出了两个门类，deprecation 和 unchecked（执行了未检查的转换时的警告，例如当使用集合时没有用泛型来指定集合保存的类型） 这个注解可以抑制多个门类的警告

@SuppressWarnings({“unchecked”,”deprecation”})

@SafeVarargs 当应用这个注解注解的方法或者构造器的时候，断言在他用的可变参数的时候是安全的。用了这个注解，unchecked这个类别的警告中有可变参数的部分就会被抑制

<Annotations That Apply to Other Annotations>

适用于其他注解的注解叫做元注解（meta-annotations），有一些元注解定义在java.lang.annotation包中

@Retention指定被标记的注解被保存的形式

RetentionPolicy.SOURCE – The marked annotation is retained only in the source level and is ignored by the compiler.

RetentionPolicy.CLASS – The marked annotation is retained by the compiler at compile time, but is ignored by the Java Virtual Machine (JVM).

RetentionPolicy.RUNTIME – The marked annotation is retained by the JVM so it can be used by the runtime environment

@Documented 每当被这个元注解指定的元素被使用的时候，就会被Javadoc工具记录下来，默认情况下注解是不包括在Javadoc 里面的

@Target 可以限制其他的注解适用于什么样的java元素

ElementType.ANNOTATION\_TYPE can be applied to an annotation type.

ElementType.CONSTRUCTOR can be applied to a constructor.

ElementType.FIELD can be applied to a field or property.

ElementType.LOCAL\_VARIABLE can be applied to a local variable.

ElementType.METHOD can be applied to a method-level annotation.

ElementType.PACKAGE can be applied to a package declaration.

ElementType.PARAMETER can be applied to the parameters of a method.

ElementType.TYPE can be applied to any element of a class.

@Inherited 说明这个注解类型是被父类继承下来的，只能适用于类声明

<<<<Annotations in eclipse>>>>

<<Javadoc>>

javadoc是一个工具，从代码中的Javadoc注释生成html文档

<Javadoc Comments>

Javadoc识别特殊的注释/\*\* … \*/ 在eclipse中默认高亮成蓝色（/\*...\*/默认高亮成绿色）

注释中间可以附上说明

<Javadoc Tags>

Tags是Javadoc认可的用来定义信息的种类的关键字

javadoc里面有一些定义好的Tags

@author[author name] 表示一个类或者一个接口的作者

@version[version] 一个类或者接口的版本信息

@param[argument name][argument description] 描述一个方法或者构造其的参数

@return[description of return][exception description] 描述方法的返回值

@exception[exception thrown][exception description] 描述方法抛出的异常

@throws[exception thrown][exception description] 和@exception一样

{@code [text]} 用代码的字体在表现文本，没有HTML的标记和javadoc tags的嵌套

｛@docRoot｝ 代表任何生成文档的相对路径的根目录

｛@inheritDoc｝ 从最近一次继承或者实现的接口继承注释

｛@link｝ 插入一个在线的链接，用可见的文本的标签，指向指定的包，类或引用类的一个成员名的文档