用大白话聊聊JavaSE -- 自定义注解入门

注解在JavaSE中算是比较高级的一种用法了，为什么要学习注解，我想大概有以下几个原因：

1. 可以更深层次地学习Java，理解Java的思想.

2. 有了注解的基础，能够方便阅读各种框架的源码，比如hibernate，SpringMVC等等。里面就用到了大量的注解。即便无法阅读源码，以后使用这些框架，会有一种**心理上的安全感**。

3. 方便今后跟别人吹牛。(当然，这也很重要。)

好了，话不多说，我们开始吧。

1. 从注释的角度来理解注解

我想了很久，最终决定以这个小标题作为第一节的标题，我们在编写Java代码的时候，为了让我们的代码看起来通俗易懂，就会加上注释信息。

比如，我们写一个方法，会标注上这个方法的作者，作用，版本等信息。是的，作为一个程序员，编写优雅的注释是一个非常重要的好习惯。

例：

```
/**
 * 用于判断是否是空字符串
 * 方法名：isEmpty
 * 创建人：剽悍一小兔
 * 时间：2016年9月21日-下午6:56:33
 * @param str
 * @return boolean
 */
public static boolean isEmpty(String str) {
    return null == str  || str.equals("")
            || str.matches("\\s*");
}
```

这是一个字符串判空的函数，函数名为isEmpty，虽然看名字大概也能猜到它的作用，可是，一旦加上了注释，瞬间就变得更加清晰了，不是吗？

这种注释，当代码被执行的时候，执行机制会自动忽略掉他们，因为这些文字其实是给程序员看的，而不是给执行机制看的。

写注释是一种美德。

那么，注解又是什么呢？

我个人对它的看法是：所谓的注解，就是写给电脑看的高级注释。

你可能经常会看到代码里面出现@XXX的标志，乍一看感觉挺高深的。反正我当年就是这种感觉，头脑里第一个反应就是这肯定很难！

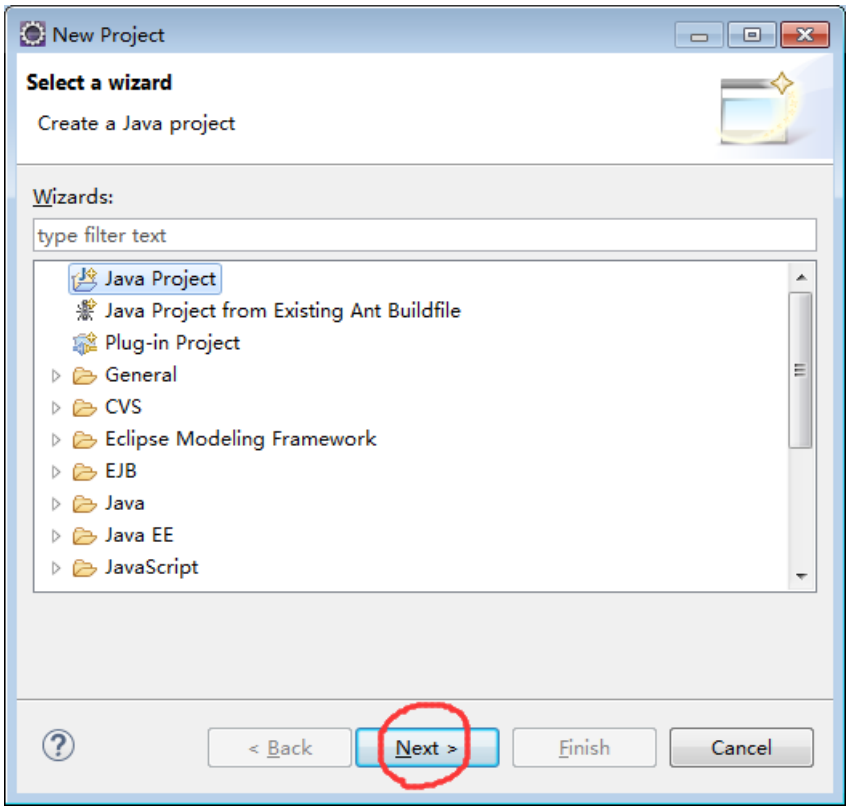
我还是那句话，如果你总想着复杂，那么就永远看不到简单。

我们写注释，是给人看的，而注解就是写给电脑看的。就这么简单。

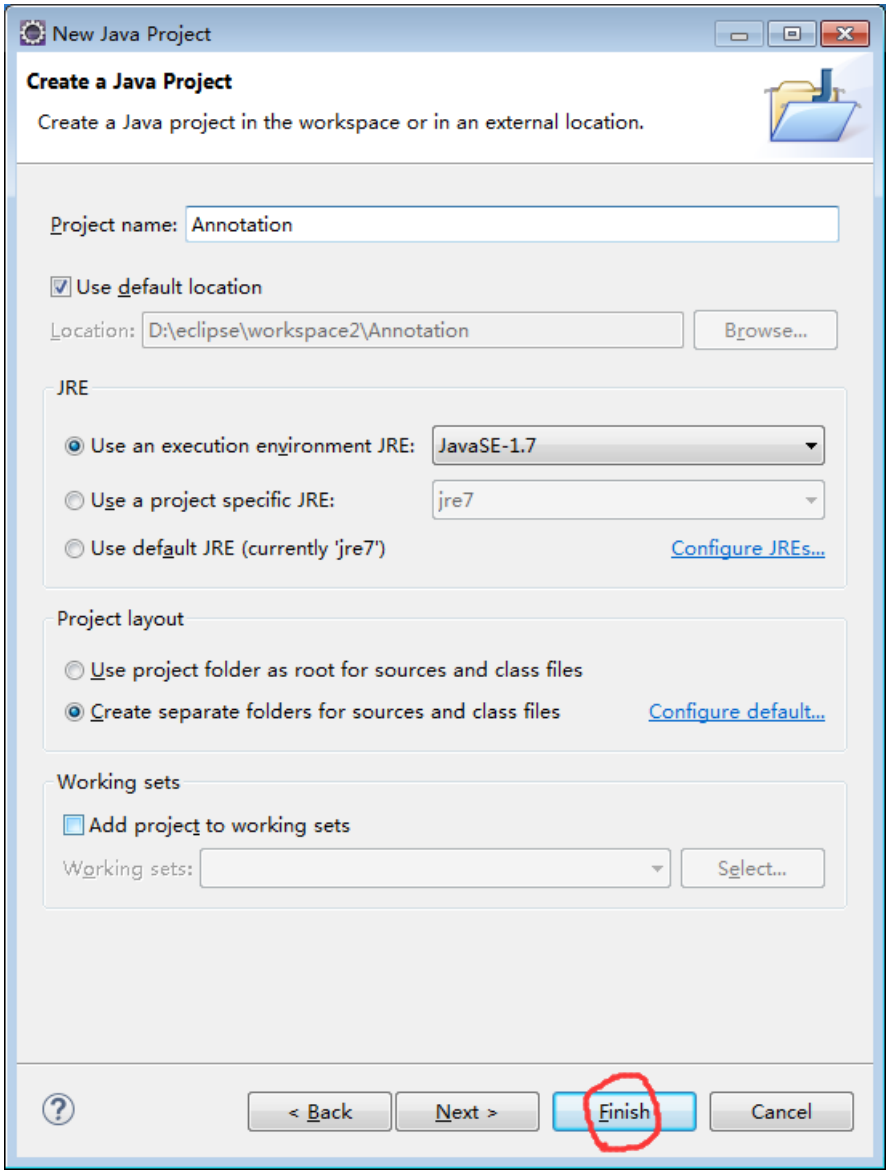
这么说可能有点抽象，没关系，我们来一个快速入门吧。

2.提出问题

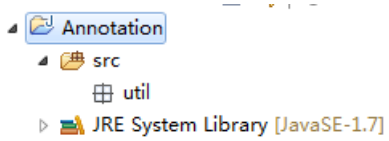
新建一个Java项目



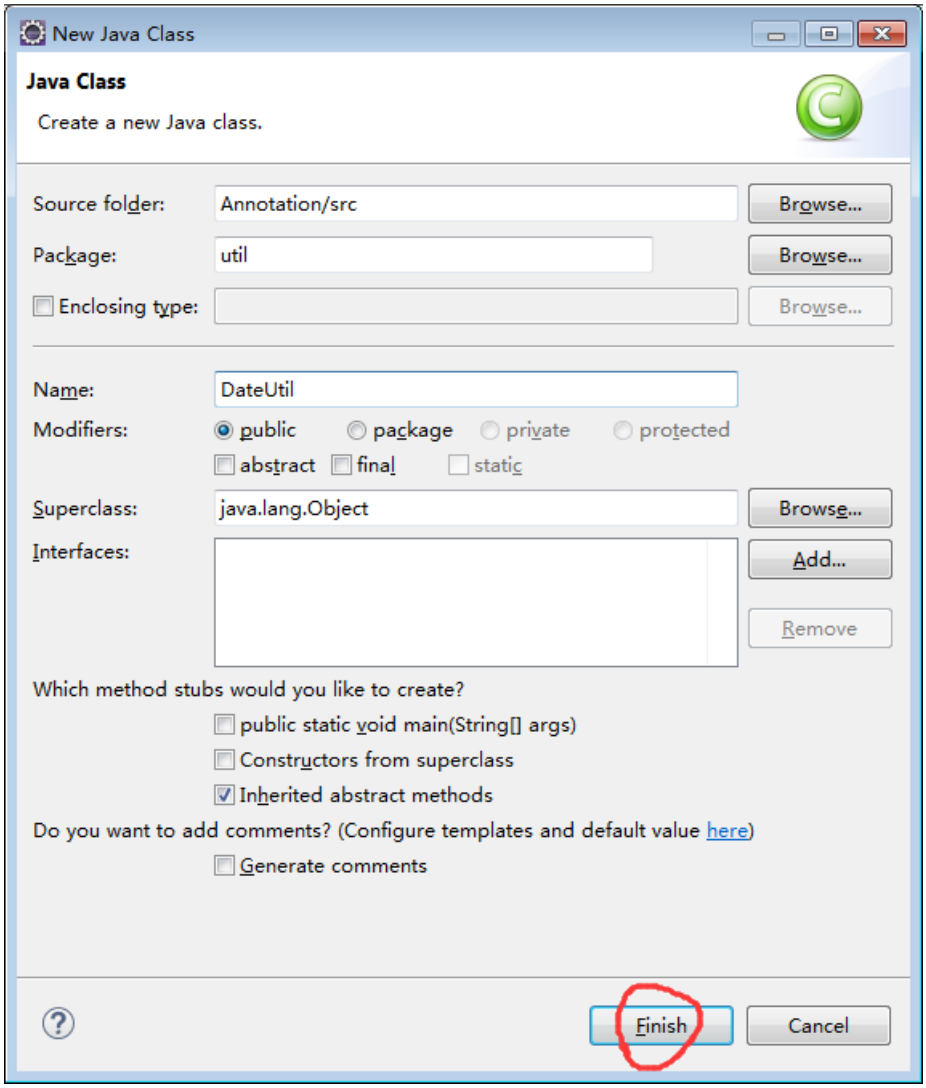
项目名称就叫做Annotation吧



在src旁边右键，新建一个util包，也就是工具包。



弄一个专门处理日期的工具类



随便写一个日期格式化的方法。

```
package util;

import java.util.Date;
import java.text.SimpleDateFormat;

public class DateUtil {

    public static String formatDate( Date date , String formatPattern ){
        return new SimpleDateFormat(formatPattern).format(date);
    }

}
```

注意，导包的时候要是java.util.Date;，而不是java.sql.Date;

测试：

```
Date now = new Date();//获取当前日期
System.out.println(now);
System.out.println(formatDate(now,"yyyy-MM-dd hh:mm:ss"));
```

控制台打印：
Wed Sep 21 19:24:57 CST 2016

2016-09-21 07:24:57

这说明，我们写的方法应该是正确的。

很好，那么接下来要解决一个什么问题呢？就是说，如果我想通过代码来获取关于这个方法的信息，那么该如何做呢？

写注释肯定是不行的，因为注释是写个程序员看的，电脑看不懂，更别提获取注释的内容了，是吧？

于是，注解，这一种高级的注释就出现了。

3.编写注解

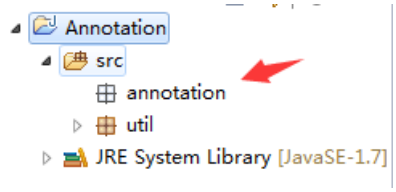
关于注解，要明确三个问题：

1. 要给谁加注解啊？
2. 什么时候注解起作用啊？
3. 要注解那些东西呢？

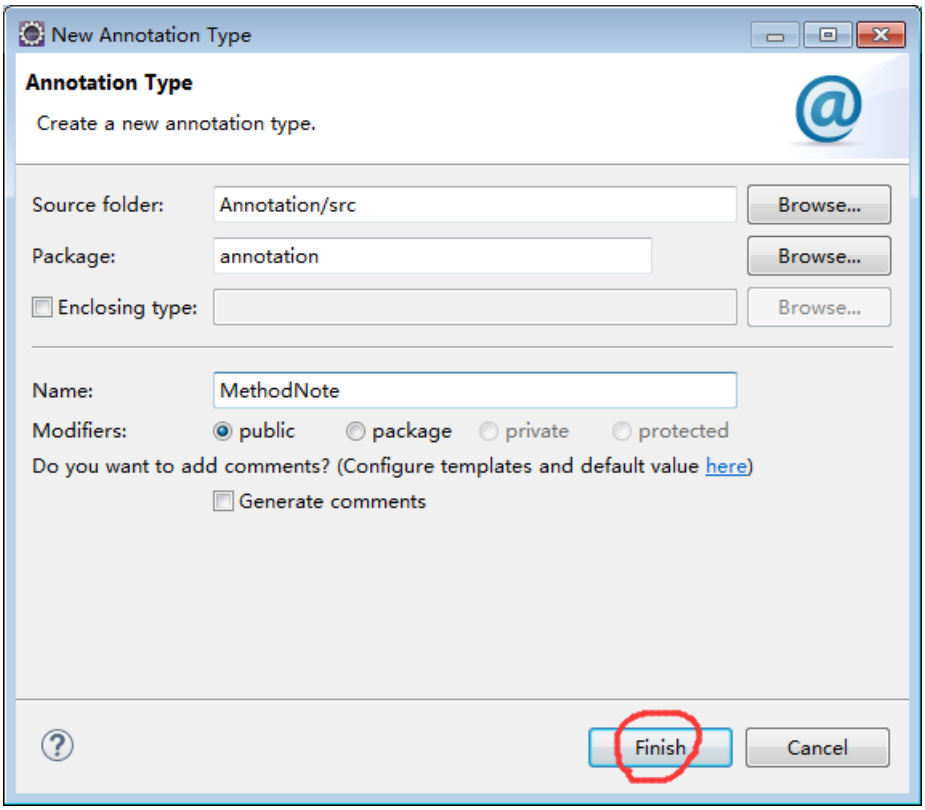
因为是快速入门，所以大概知道这些就足够了。

现在，我们来新建一个注解，毫无疑问，所谓的注解，它还是一个Java类，你不要被它吓到。

新建一个注解包。



new一个Annotation，就叫MethodNote，意思就是说，这个是加在方法上的，为了给方法加一些电脑能看得懂的说明。



第一个问题是要给谁加注解啊？那么，这个注解类是需要加在方法上的，于是就这样写：

```java
package annotation;

import java.lang.annotation.ElementType;
import java.lang.annotation.Target;

@Target(ElementType.METHOD)
public @interface MethodNote {

}
```

这就表示，该注解要加在方法上。

接下来，让我们来明确第二个问题：什么时候注解起作用啊？

我们希望在程序运行的时候，注解发挥作用，就是说，当你的程序跑起来了，电脑才开始阅读这些注解。

```java
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface MethodNote {

}
```

这句话的意思就是说，我这个注解啊，是在程序跑起来的时候，RUNTIME嘛，就是跑起来的时候，才发挥作用的。

非常好，那么最后一个问题：要注解那些东西呢？

一个方法，最重要的信息包括：作用，创建时间，作者，版本，返回值等等。我们随便抽取几个，就作用和创建时间吧！

```java
@Target(ElementType.METHOD)
@Retention(RetentionPolicy.RUNTIME)
public @interface MethodNote {

    //方法作用，有默认值。
    String description() default "作者很懒，没有写本方法的作用。";

    //方法创建时间
    String createTime() ;

}
```

这种写法有点类似于写接口的方法。

好了，我们的第一个注解就编写完成了！写好了就马上用呗，现在我们给日期格式化的方法加上咱自己编写的注解。

```java
@MethodNote(createTime = "2016-9-21")
public static String formatDate(Date date , String formatPattern){
    return new SimpleDateFormat(formatPattern).format(date);
}
```

这就是所谓的注解，其实也很简单的吧。就是这么来的，它归根到底还是一个Java类。

4.通过Java反射获取方法的注解信息

好了，回到正题，我们已经对formatDate方法进行了注解，那么，既然这个注解是写给电脑看的，那么电脑就肯定有办法在其他Java类中获得这些信息，对吧？

如何获得呢，对了，用反射机制。

上代码：

```
public static void main(String[] args) throws NoSuchMethodException, SecurityException {

    Class classOfDateUtil = DateUtil.class;
    Method formatDate = classOfDateUtil.getMethod("formatDate", Date.class,String.class);
    MethodNote methodNote = formatDate.getAnnotation(MethodNote.class);

    System.out.println("方法描述：" + methodNote.description());
    System.out.println("创建日期：" + methodNote.createTime());
}
```

结果：

方法描述：作者很懒，没有写本方法的作用。
创建日期：2016-9-21

本章结束 ...

本章对Java自定义注解做了一个快速入门，希望对你有所帮助。