1. 几个基础的注解的使用。

@Required：在set方法的前面使用，表示必须的，否则会报错。

@Autowired可以用来修饰set方法，构造方法以及属性。

这些注解的使用是为了赋值时的方便，spring届时会根据注解将数据赋给这些属性或者对象。(bean的装配)

**@Qualifier可以用来做bean的筛选**

可以使用 **@PostConstruct** 注释作为初始化回调函数的一个替代，**@PreDestroy** 注释作为销毁回调函数的一个替代

在字段中或者 setter 方法中使用 **@Resource** 注释，相当于是一个by\_name的自动连接。

以上参考博文：<http://wiki.jikexueyuan.com/project/spring/annotation-based-configuration/spring-jsr250-annotation.html>

1. 一个简单的注解类的使用

@Retention(RetentionPolicy.***RUNTIME***)

@Target(ElementType.***METHOD***)

**public** **@interface** NeedTest{

**boolean** value() **default** **false**;

}

这边有一系列的属性可以配置

**public** **class** ForumService {

@NeedTest(value=**true**)

**public** **void** deleteForum(**int** forumId){

System.***out***.println("删除论坛模块："+forumId);

}

/\*\*

\*

\* **@param** topicId

\*/

@NeedTest(value=**false**)

**public** **void** deleteTopic(**int** topicId){

System.***out***.println("删除论坛主题："+topicId);

}

}

**public** **static** **void** main(String[] args) {

Class clazz = ForumService.**class**;

Method[] methods = clazz.getDeclaredMethods();

**for** (Method method : methods) {

NeedTest nt = method.getAnnotation(NeedTest.**class**);

**if** (nt != **null**) {

**if** (nt.value()) {

System.***out***.println(method.getName() + "()需要测试");

} **else** {

System.***out***.println(method.getName() + "()不需要测试");

}

}

}

}

//这边没有涉及到xml文件的配置

3.这边是在我们项目中关于注解的部分应用

//这边就涉及到jpa的注解问题(周一的时候，有些问题需要问一下晓进)

有一篇博文: <http://www.ibm.com/developerworks/cn/opensource/os-cn-spring-jpa/>

//这边有几个问题:

//1.在jpa中的建立的那些注解，都标记在了pojo中了，但是，之后呢，他们会起什么样的作用呢，以及在那边控制。

//2.用的是**Hibernate,** 项目中的pojo实体实际上也是自动生成的，通过什么技术。

//3.在项目中为什么mybaties和hibernate都要使用呢。

//下面是回答：

以上。