* **Spring概述**

开源的，用来简化企业级应用开发的应用开发框架。

**(1)简化开发**

Spring对常用的api（比如jdbc）做了封装，这样，可以大大简化 这些api的使用。（比如使用springjdbc访问数据库，就不用考虑如何获取连接和关闭连接了）。

**(2)解耦**

Spring帮我们建立对象之间的依赖关系，这样，对象之间的耦合度会大大降低，代码的维护性会大大提高。

**(3)集成其它框架**

Spring可以将其它的一些框架集成进来(比如用于定时任务处理的Quartz 等),方便这些框架的使用。

* **Spring容器**

**(1)什么是Spring容器?**

Spring框架中的一个核心模块，用于管理对象。

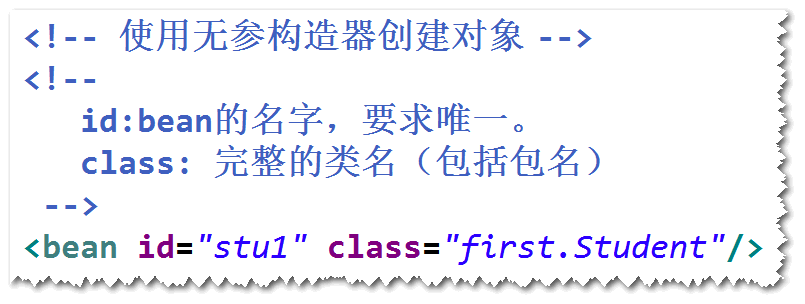
**(2)启动Spring容器**

step1.导包。 spring-webmvc step2. 添加spring配置文件。 step3. 启动容器。

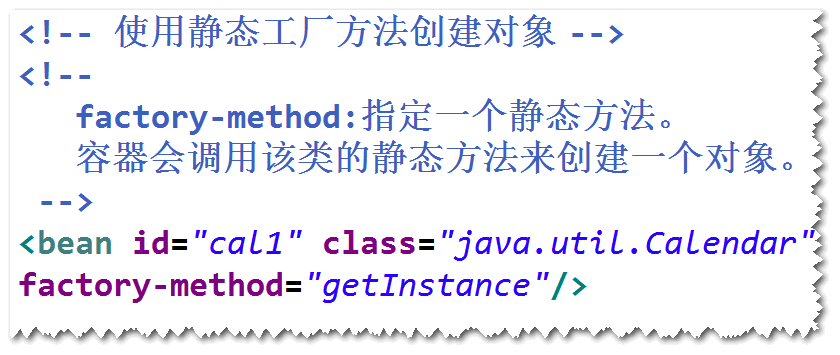


**(3)创建对象**

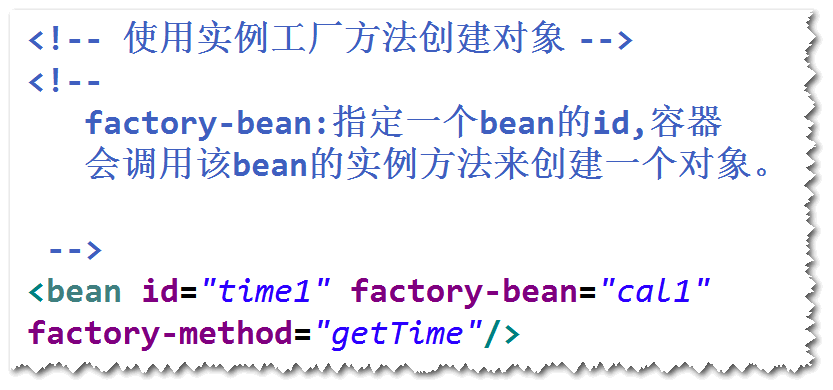
1)方式一 无参构造器 step1. 为类添加无参构造器（或者缺省构造器） step2. 在配置文件里面，添加bean元素。 注：bean就是容器所创建的对象。 step3. 启动容器，调用容器的getBean方法。



2）方式二 静态工厂方法 (了解)



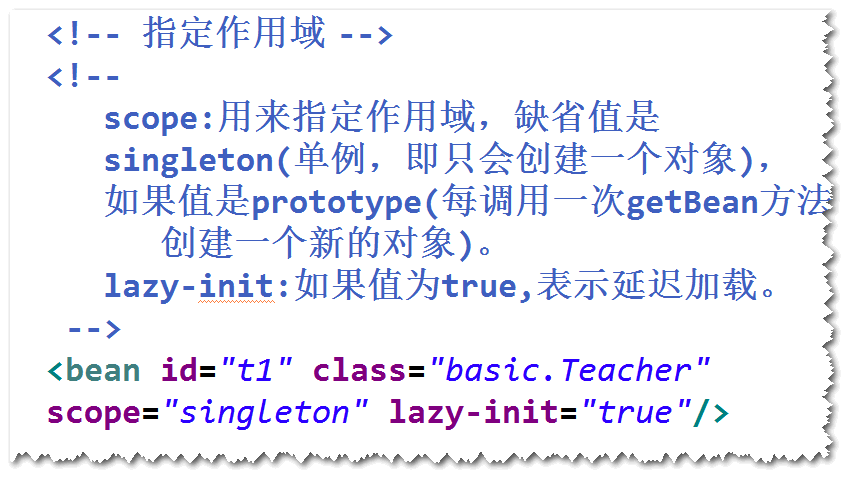
3) 方式三 实例工厂方法（了解）



**(4)作用域**

1）默认情况下，容器对于每个bean元素，只会创建一个实例。

2) 如果将作用域设置为"prototype"，则每调用一次getBean方法， 就会创建一个新的实例。



**(5)延迟加载 (了解)**

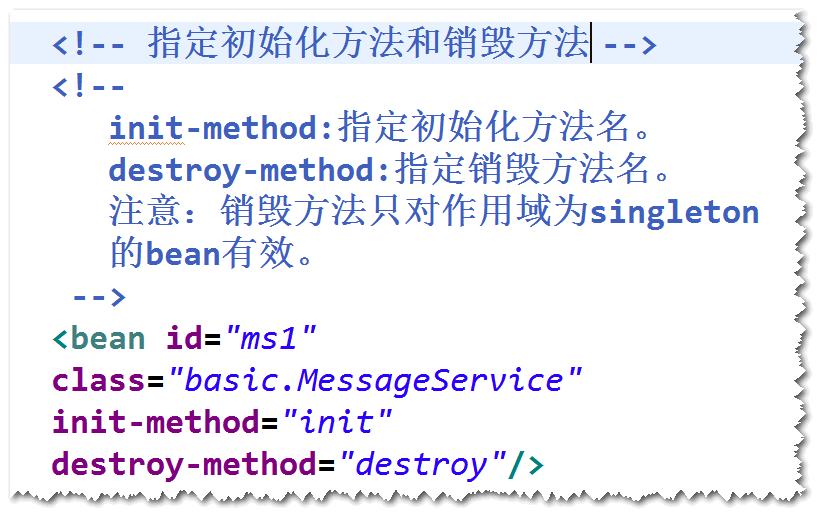
1)默认情况下，容器启动之后，会将作用域为"singleton"的bean 创建好。

2)延迟加载指的是，容器启动之后，对作用域为"singleton"的bean 不再创建，直到调用了getBean方法才会创建。

**(6)生命周期**

1)初始化方法 容器创建好bean的实例之后，会立即调用初始化方法。

2)销毁方法 容器关闭之前，会调用销毁方法。



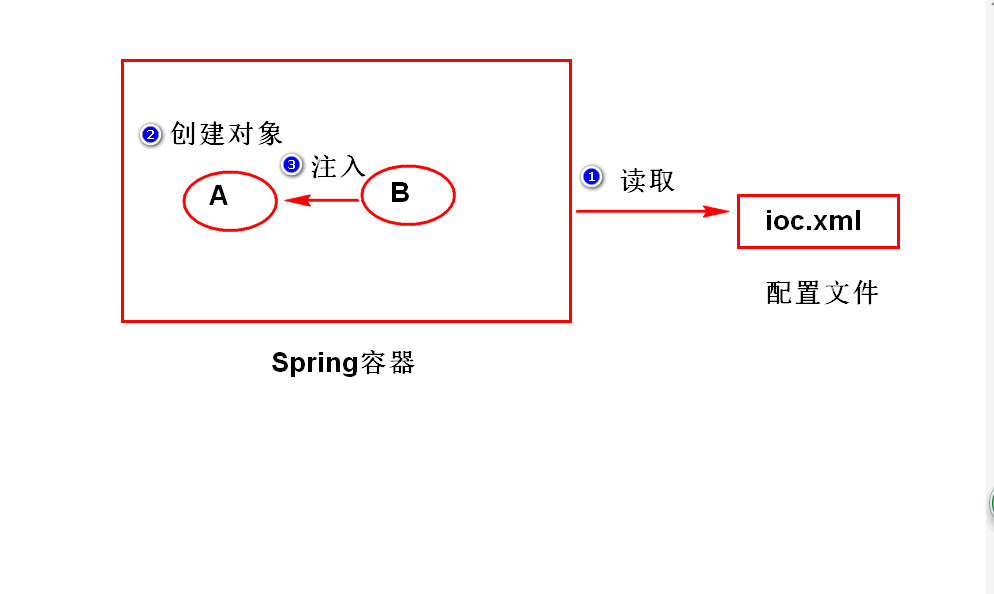
* **IOC/DI**

**(1)什么是IOC(Inversion Of Controll 控制反转)**

对象之间的依赖关系由容器来建立。

**(2)什么是DI(Dependency Injection 依赖注入**

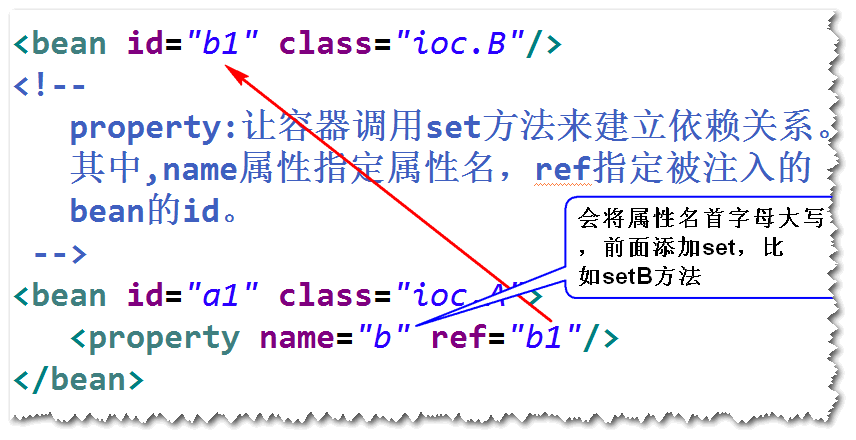
容器可以通过调用set方法或者构造器来建立对象之间的依赖关系。 注： IOC是目标，DI是手段。



**(3)依赖注入的两种方式**

1)方式一 set方法注入 step1. 添加set方法

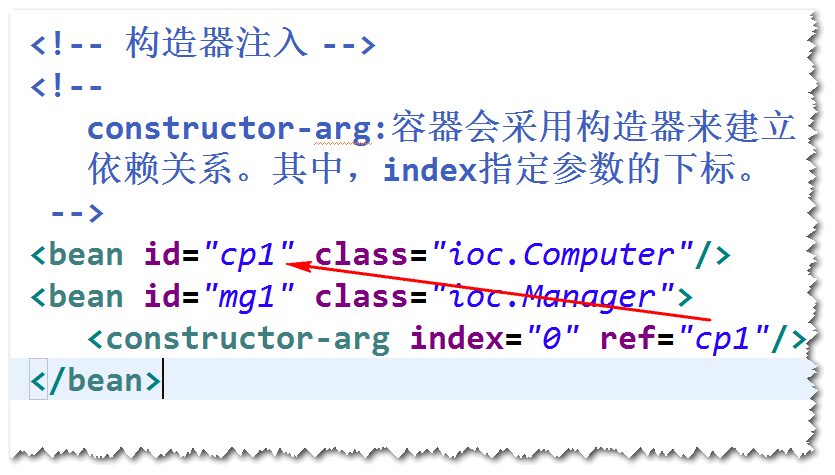
step2. 在配置文件中，使用<property>元素来配置。



2)方式二 构造器注入

step1. 添加构造器

step2. 在配置文件当中，使用<constructor-arg>元素来配置。

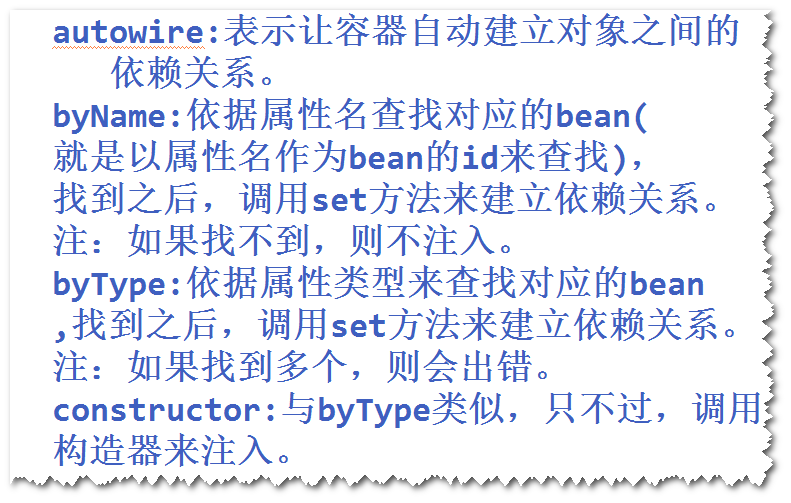


**(4)自动装配 (了解)**

注：自动装配，指的是容器依据某些规则，自动建立对象之间的依赖关系。

1)默认情况下，容器不会自动装配。

2)设置autowire属性

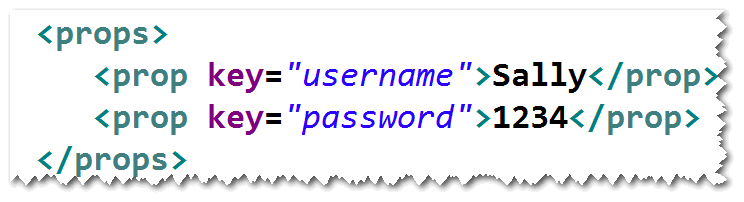
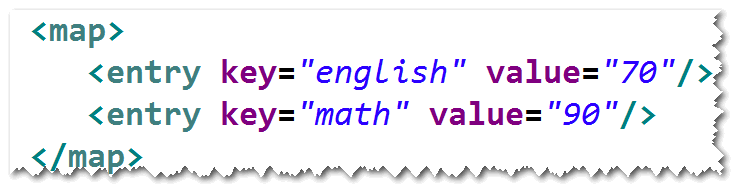
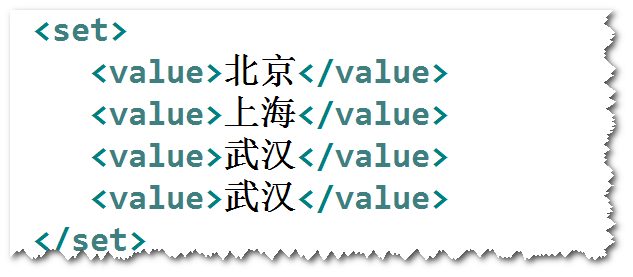
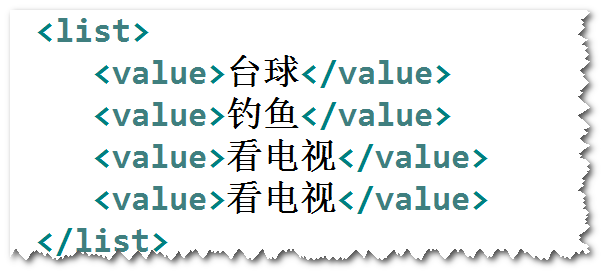


**(5) 注入基本类型的值**

使用value属性来注入，spring容器会帮我们做一些类型的转换工作，比如将字符串转换成数字。

**(6) 注入集合类型的值 (List,Set,Map,Properties)**

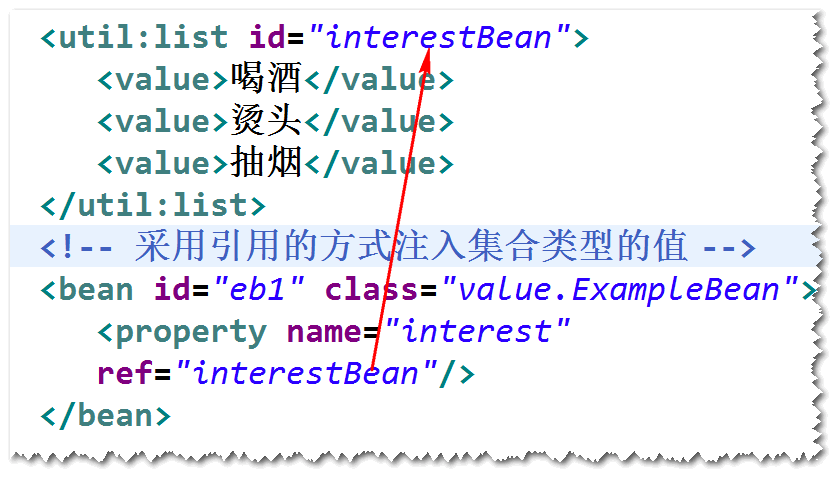
方式一 直接注入



方式二 引用的方式注入

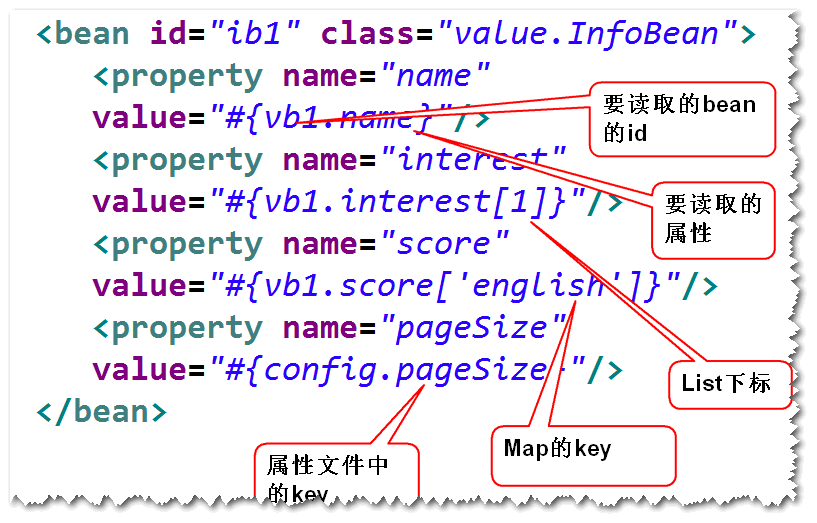
step1. 将集合类型的值先配置成一个bean。

step2. 再将这个bean注入到对应的bean里面。



**(7)spring表达式**

注：读取bean的属性



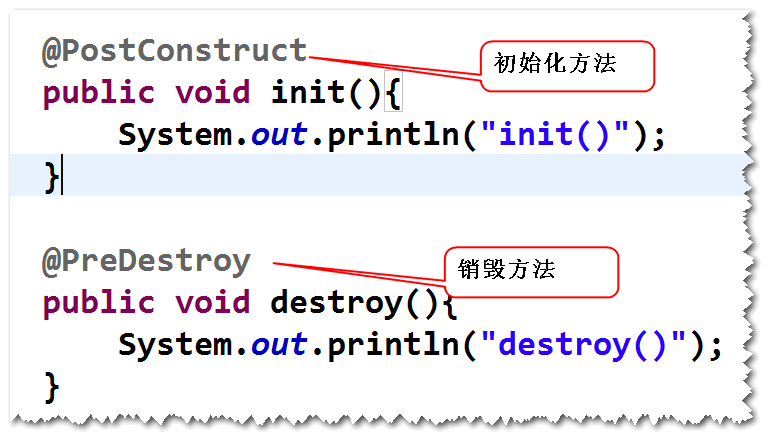
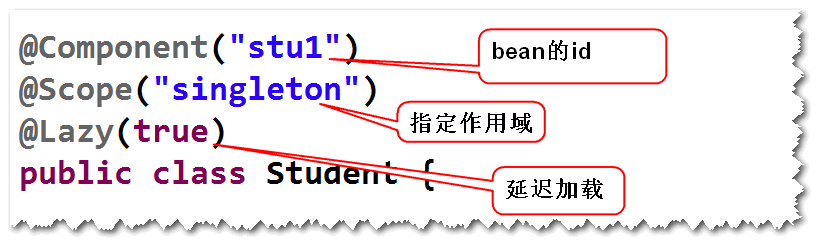
* **使用注解来简化配置文件**

**(1)什么是组件扫描?**

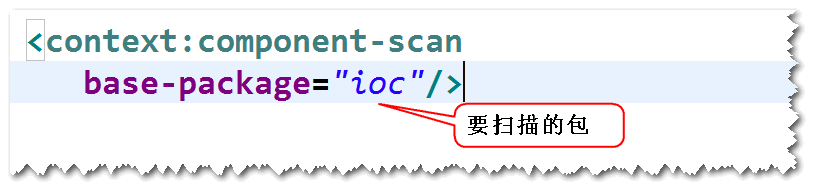
容器会扫描指定的包及其子包下面的所有的类，如果该类前面有特定的注解比如@Component(）, 则容器会将其纳入容器进行管理（相当于在配置文件里面 有一个bean元素）。

**(2)如何进行组件扫描?**

step1. 在类前面添加特定的注解，比如 @Component 注：默认的id是首字母小写之后的类名。

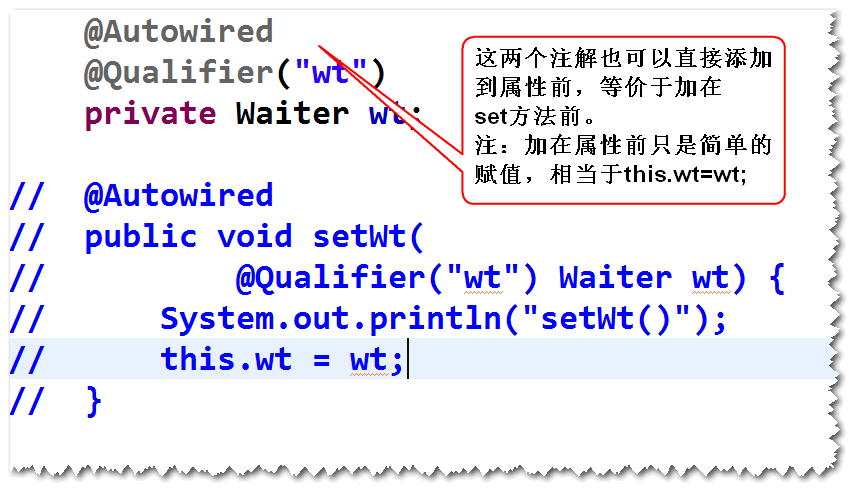
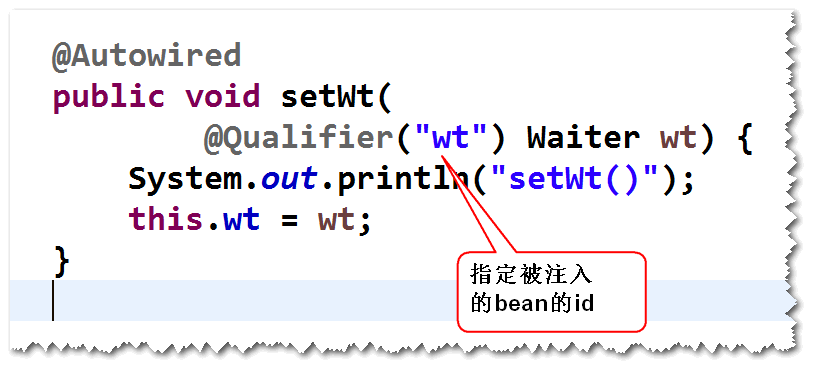


step2. 在配置文件当中，配置组件扫描。

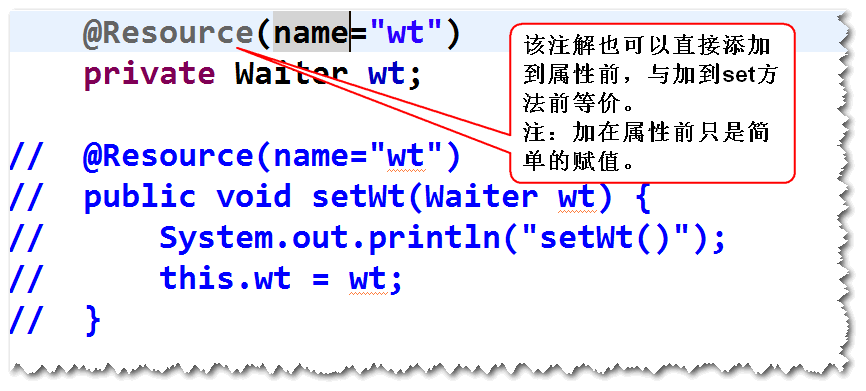
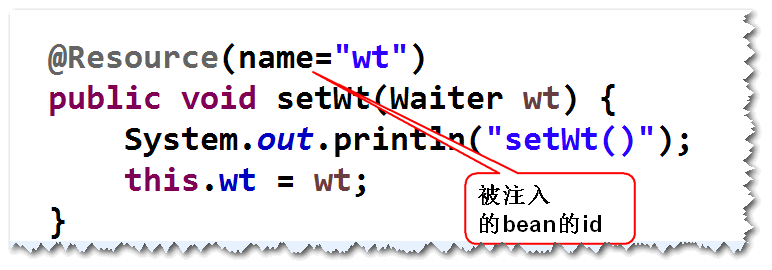


**(3)依赖注入相关的注解**

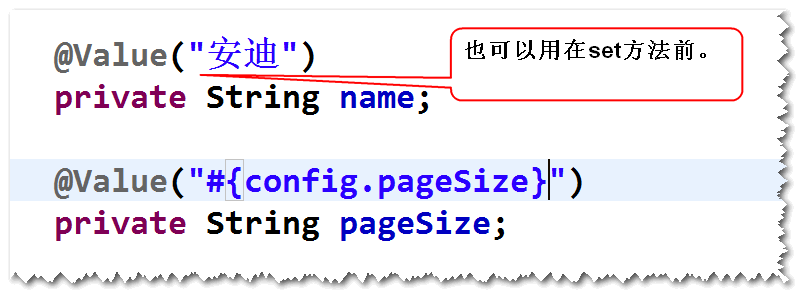
1)@Autowired @Qualifier 注：该注解支持set方法和构造器注入。



2)@Resource 注：该注解只支持set方法注入。



**(4)注入基本类型的值和spring表达式的值**



* **SpringMVC**

**(1)什么是SpringMVC?**

用来简化基于MVC架构的**web应用程序开发**的框架。

注：SpringMVC是spring中的一个模块。

**(2)五大组件**

1)有哪五大组件?

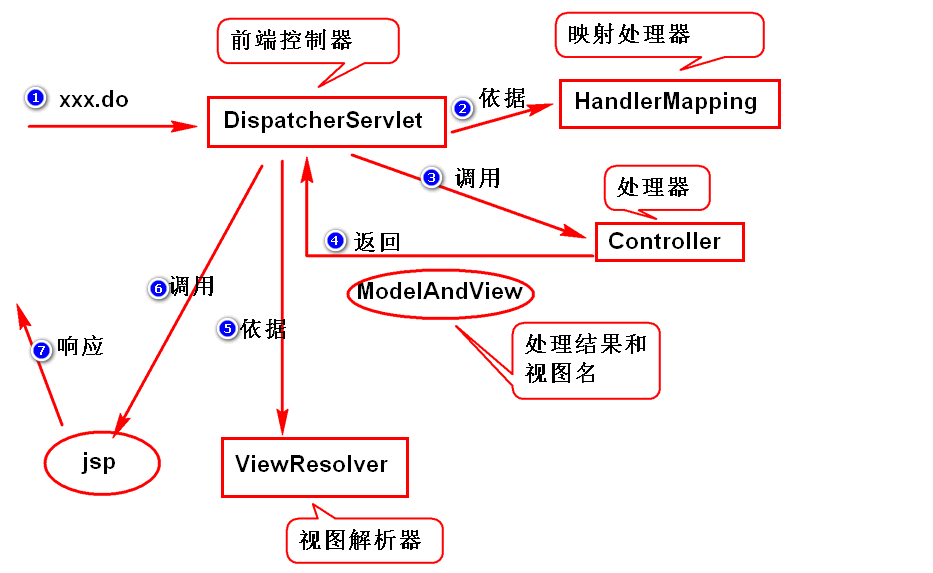
* DispatcherServlet 前端控制器
* HandlerMapping 映射处理器
* Controller 处理器
* ModelAndView
* ViewResolver 视图解析器

2)它们之间的关系

a.请求发送给DispatcherServlet来处理，DispatcherServlet 会依据HandlerMapping的配置调用对应的Controller来处理。 b.Controller将处理结果封装成ModelAndView，然后返回给 DispatcherServlet。

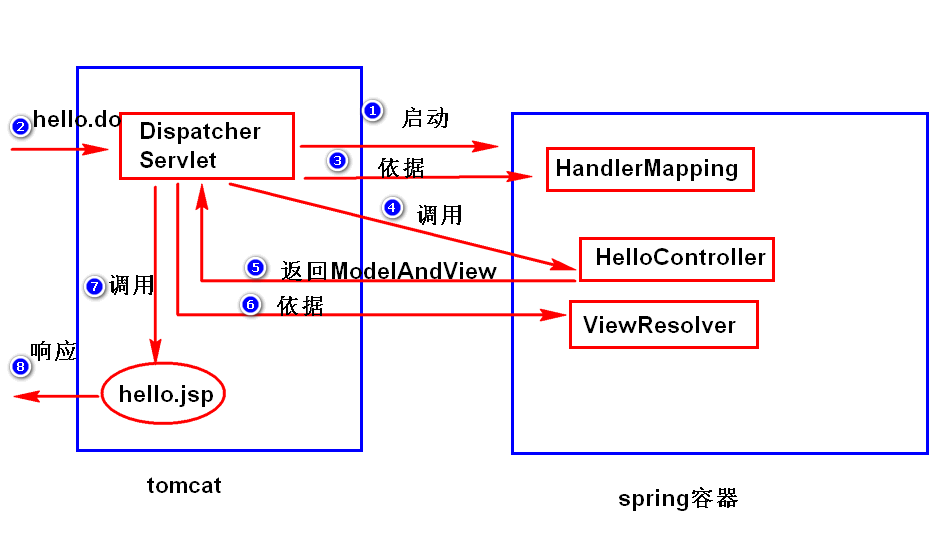
c.DispatcherServlet会依据ViewResolver的解析调用对应的视图对象（比如jsp）来生成相应的页面。

注：视图部分可以使用jsp,也可以使用其它的视图技术，比如 freemarker,velocity等等。



**（3）编程步骤**

* step1. 导包 spring-webmvc
* step2. 添加配置文件。
* step3. 配置DispatcherServlet。
* step4. 写Controller。
* step5. 写jsp。
* step6.在配置文件当中，添加HandlerMapping, ViewResolver的配置。



* **基于注解的SpringMVC应用**

**(1)编程步骤**

step1.导包。 spring-webmvc

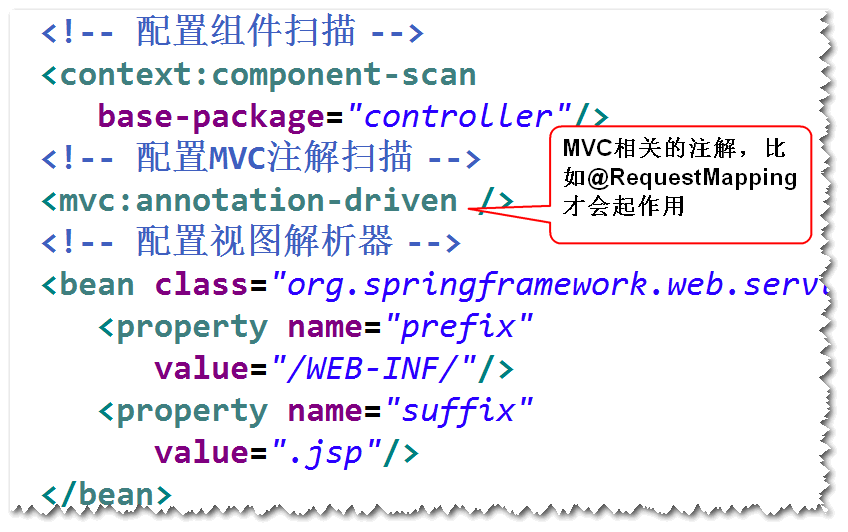
step2.添加spring配置文件。

step3.配置DispatcherServlet。

step4.写Controller

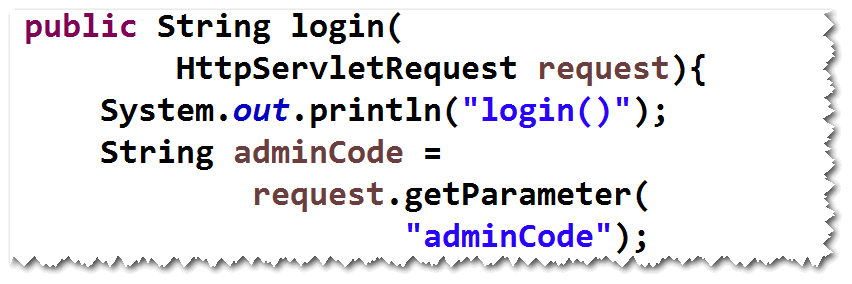


step5.写jsp step6.在配置文件当中，添加ViewResolver配置， 添加组件扫描，添加MVC注解扫描。

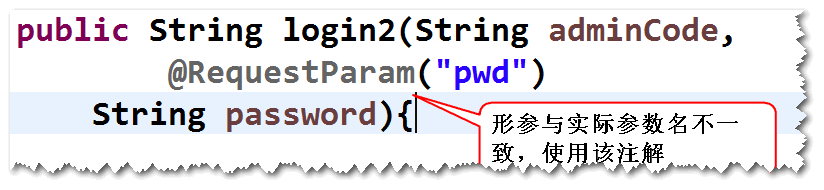


**(2).读取请求参数值**

**1）方式一 通过request提供的方法**

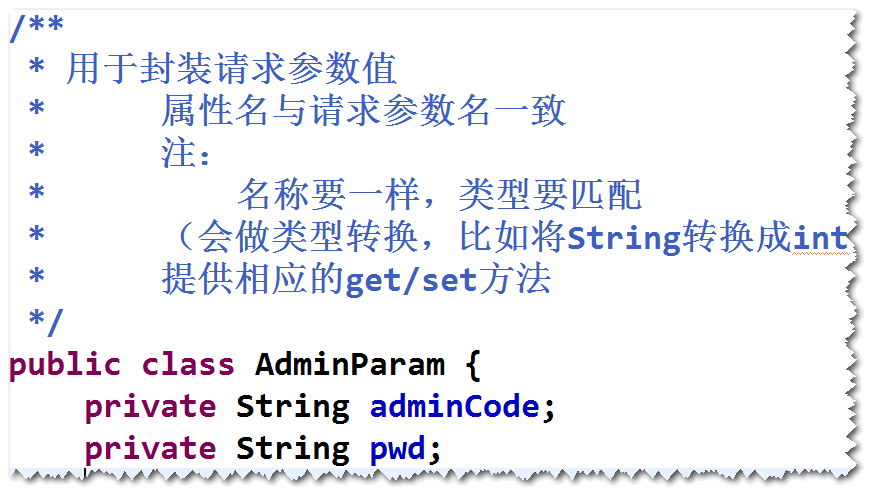


**2) 方式二 通过@RequestParam注解**

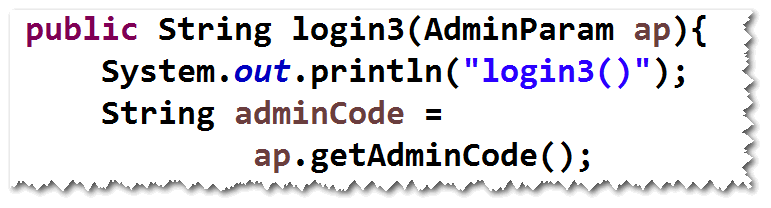


**3) 方式三 通过javabean**

step1. 先写一个java类

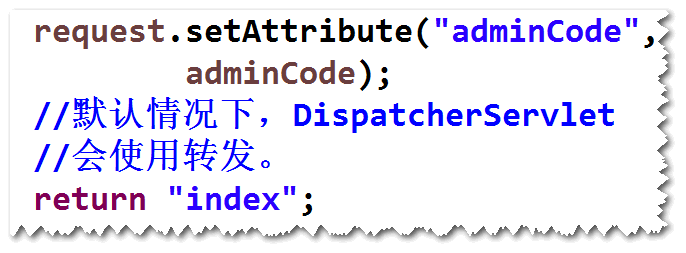


step2. 将该类作为方法参数

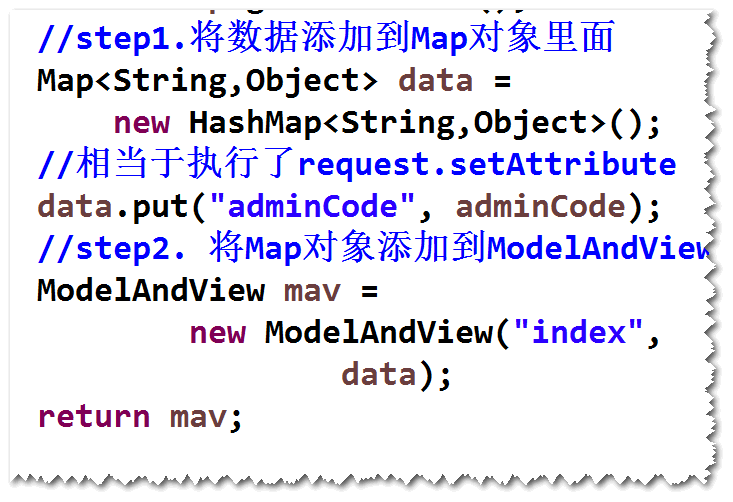


**(3)向页面传值**

**1)方式一 将数据绑订到request**



**2)方式二 返回ModelAndView**



**3)方式三 将数据添加到ModelMap**



**4)方式四 将数据绑订到session**

**(4)重定向**

**1)方法的返回值是String**

比如 return "redirect:toIndex.do";

**2)方法的返回值是ModelAndView**

比如 RedirectView rv = new RedirectView("toIndex.do"); ModelAndView mav = new ModelAndView(rv);

# 系统分层 （扩展）

## (1)为什么要分层?（了解)

为了系统好维护，系统的设计应该要做到“高内聚，低耦合”。

“高内聚”：指的是类的职责要单一，这样，一个类就可以会拆分 成多个类（比如AccountService拆分成了AccountService和 AccountDAO）,这样，就形成了分层的概念。

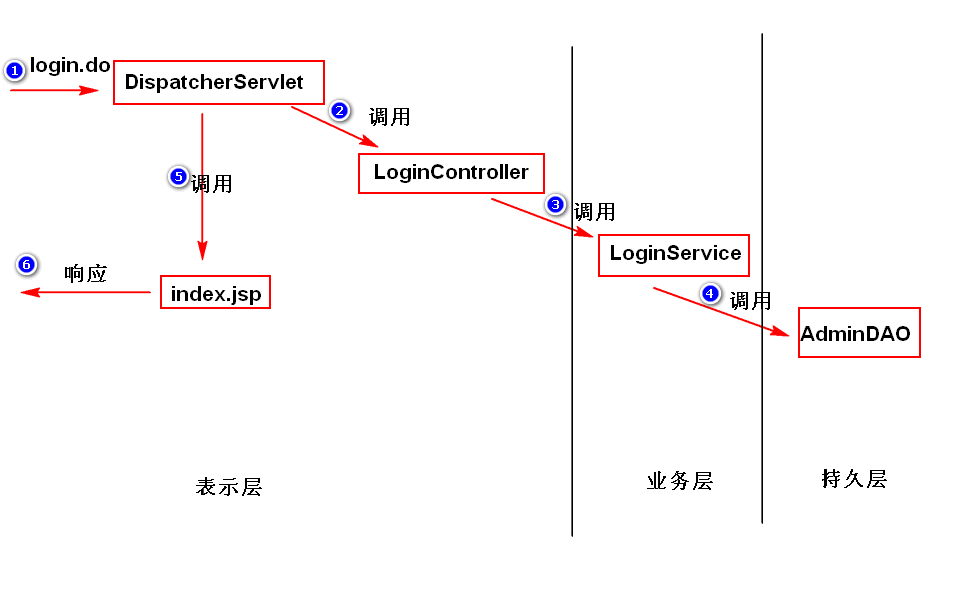
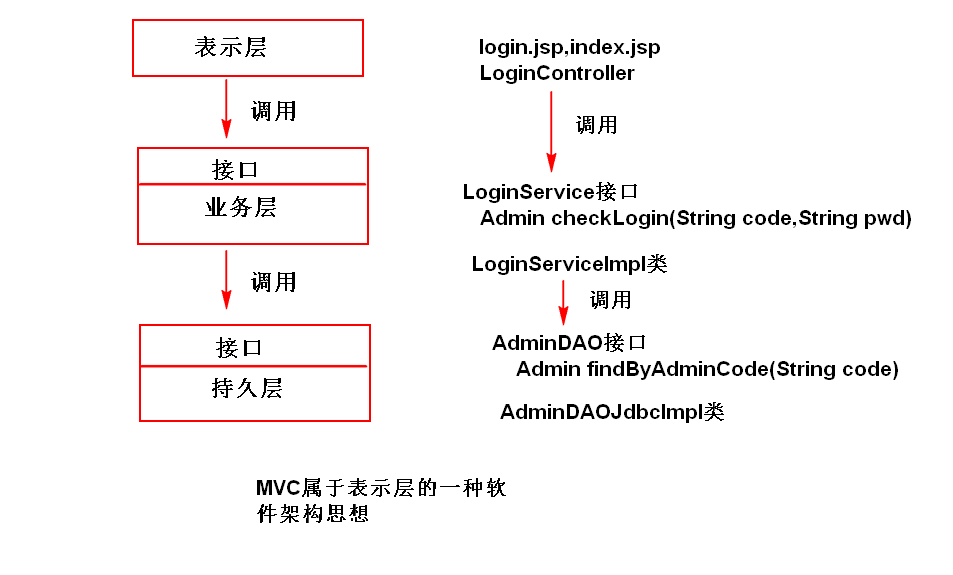
“低耦合”：指的是类与类之间不要直接依赖。（AccountService 要调用AccountDAO，应该使用依赖注入）。

## (2)如何分层?

表示层：数据展现和操作界面，以及请求分发。

业务层：封装了业务逻辑。 持久层: 封装了数据访问逻辑。

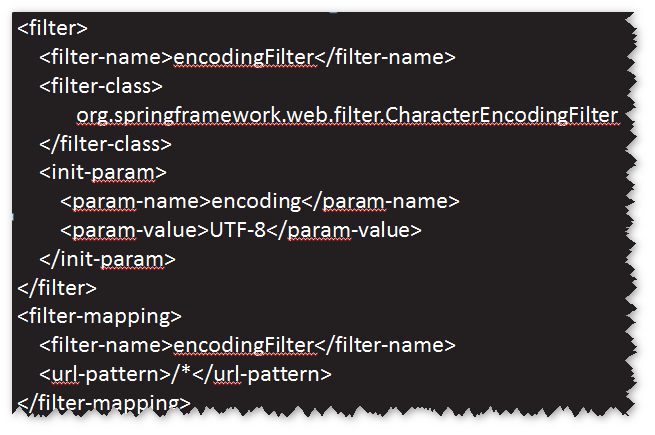
注： 表示层调用业务层，业务层调用持久层。 上一层通过接口来调用下一层提供的服务（这样，下一层的 实现发生了改变，不影响上一层）。



# 1. 表单包含有中文参数值，如何读取?

springmvc提供了一个过滤器(CharacterEncodingFilter), 只需要配置该过滤器即可。

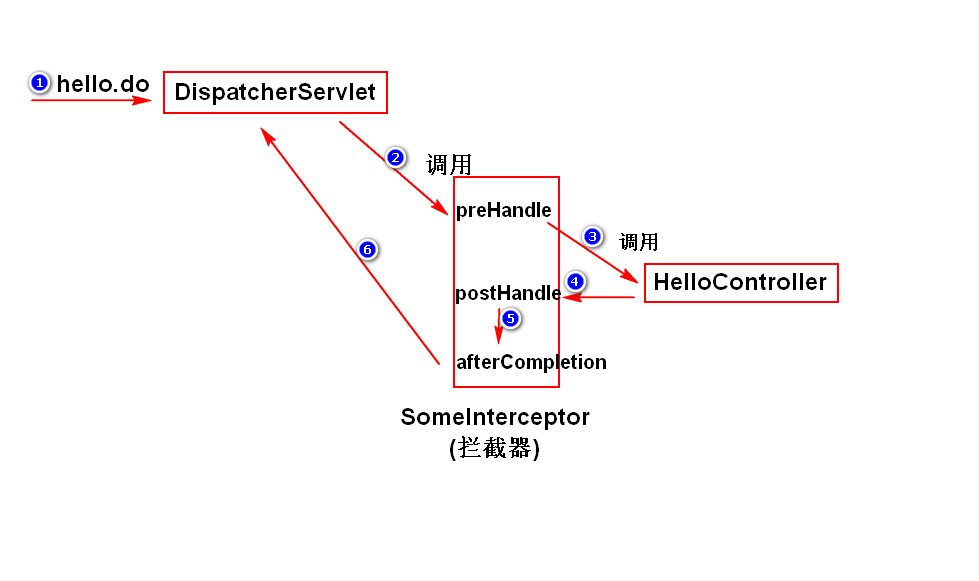
注意： a. 表单的提交方式必须是"post"。 b. 编码与浏览器端的一致。



# 2. 拦截器

## (1)什么是拦截器?

DispatcherServlet收到请求之后，如果有拦截器，会先调用 拦截器，然后再调用Controller。 注： 过滤器属于servlet规范，而拦截器属于spring框架。

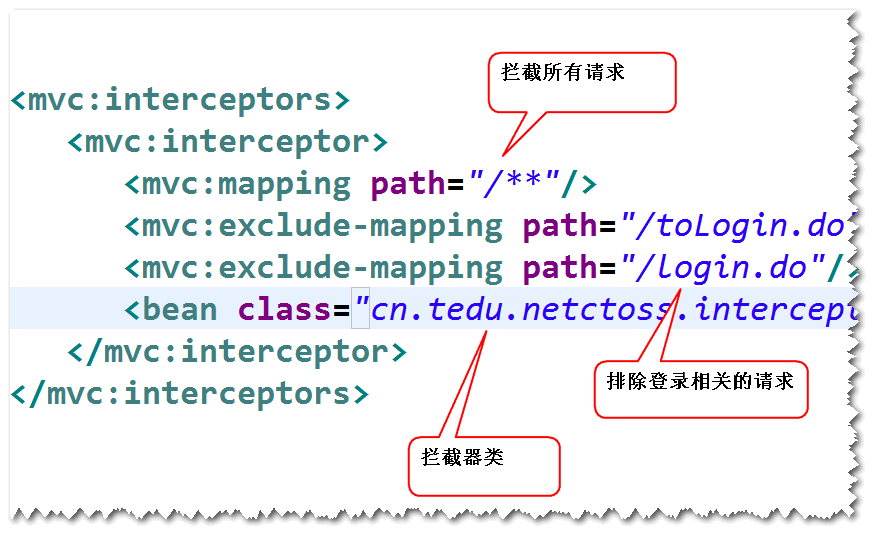


## (2)如何写一个拦截器?

step1. 写一个java类，实现HandlerInterceptor接口。

step2. 在接口方法里面，实现拦截处理逻辑。

step3. 配置拦截器。

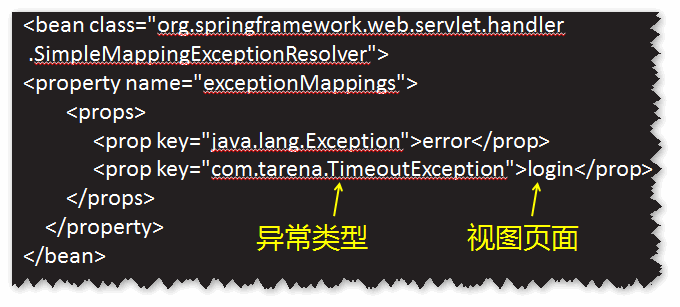


# 3. 让spring框架帮我们处理异常

注：即将异常抛给spring框架，让spring框架来处理。 默认情况下，spring框架会将异常直接抛给最终的用户。

## (1)方式一 配置简单异常处理器。

step1.在spring配置文件当中，配置简单异常处理器。

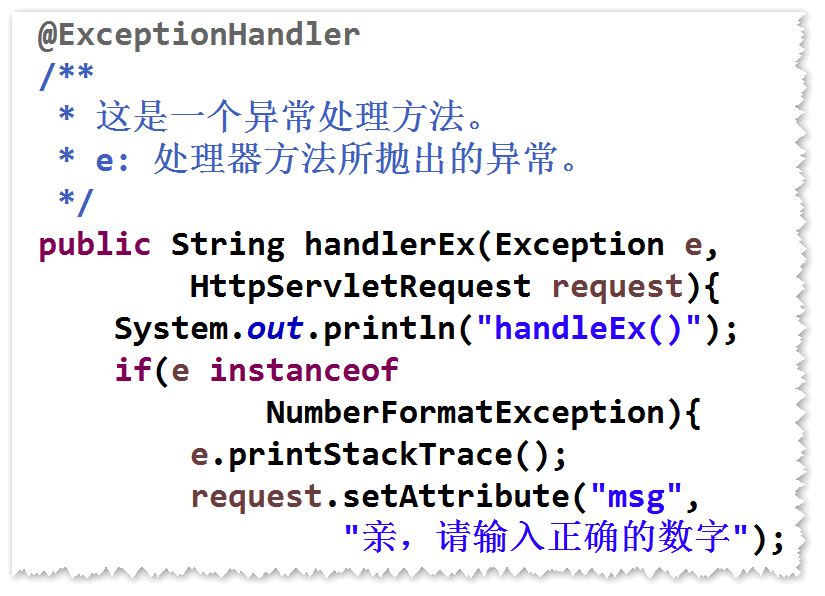


step2.添加相应的异常处理页面。 注：简单异常处理器，不能够对异常做复杂的处理。

## (2)方式二 @ExceptionHandler。

step1. 在处理器当中添加一个异常处理方法。该方法 前面需要添加@ExceptionHandler注解。

step2. 在异常处理方法里面，依据异常类型做不同的处理。



* **SpringJdbc**

## (1)SpringJdbc是什么?

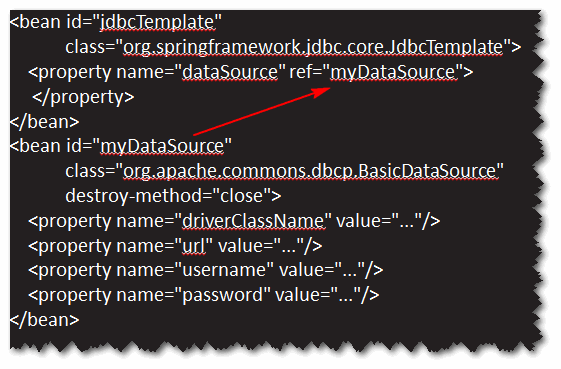
Spring框架对jdbc的封装。

## (2)编程步骤

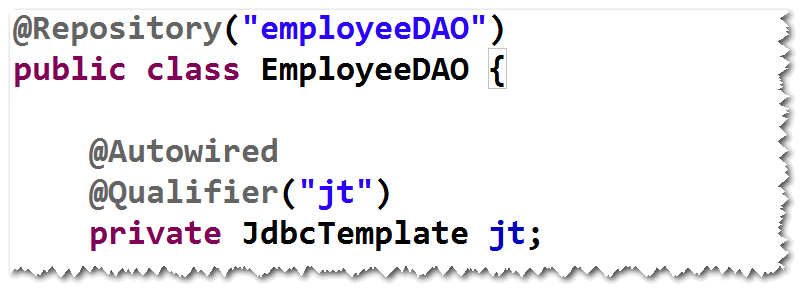
step1.导包 spring-webmvc,spring-jdbc, ojdbc,dbcp,junit。

step2.添加spring配置文件。

step3.配置JdbcTemplate。 注：JdbcTemplate把一些重复性的代码（比如获取连接，关闭） 连接，异常处理等等都写好了），我们只需要调用该对象的 方法就可以很方便的访问数据库。



step4.调用JdbcTemplate的方法来访问数据库。 注：通常将JdbcTemplate注入到DAO。



create table t\_emp( id number(8) primary key, name varchar2(20), age number(3) ); create sequence t\_emp\_seq;