目录

[概述 1](#_Toc1548)

[RequestMapping 4](#_Toc15747)

[请求参数的绑定 5](#_Toc5242)

[自定义类型转换 8](#_Toc19955)

[获取url路径的值 10](#_Toc793)

[响应数据和结果视图 10](#_Toc10467)

[响应json数据、异步请求 ajax 11](#_Toc27141)

[文件的上传与下载 11](#_Toc18577)

[异常处理 13](#_Toc23240)

[SSM整合 15](#_Toc8061)

[Spring整合Spiringmvc 16](#_Toc4222)

# 概述

springMVC是属于web层面的框架 负责接受请求和相应

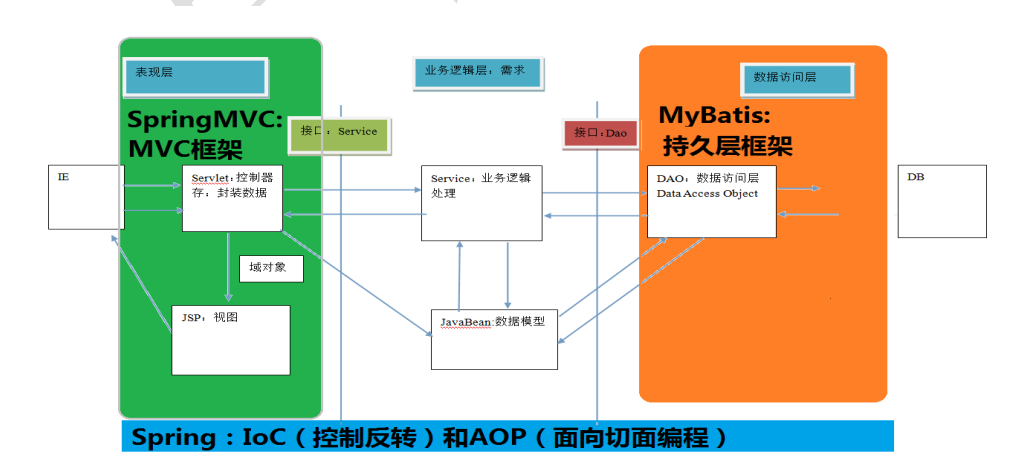
Spring是属于服务层的框架

Mybatis属于dao层框架负责crud

Springmvc其中一大核心是配置控制器servlet



SSM框架



Springmvc相比Struts2处理ajax请求更加方便，因为他实现功能是创建方法，而Struts2实现功能是创建新的对象

在 SpringMVC 的各个组件中，处理器映射器、处理器适配器、视图解析器称为 SpringMVC 的三大组件。

使 用 <mvc:annotation-driven> 自动加载 RequestMappingHandlerMapping （处理映射器） 和

RequestMappingHandlerAdapter （ 处 理 适 配 器 ） ， 可 用 在 SpringMVC.xml 配 置 文 件 中 使 用

<mvc:annotation-driven>替代注解处理器和适配器的配置。

它就相当于在 xml 中配置了：

<!-- 上面的标签相当于 如下配置-->

<!-- Begin -->

<!-- HandlerMapping -->

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping"></bean>

<bean

class="org.springframework.web.servlet.handler.BeanNameUrlHandlerMapping"></bean>

<!-- HandlerAdapter -->

<bean

class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerA

dapter"></bean>

<bean

class="org.springframework.web.servlet.mvc.HttpRequestHandlerAdapter"></bean>

<bean

class="org.springframework.web.servlet.mvc.SimpleControllerHandlerAdapter"></bean>

<!-- HadnlerExceptionResolvers -->

<bean

class="org.springframework.web.servlet.mvc.method.annotation.ExceptionHandlerExcept

ionResolver"></bean>

<bean

class="org.springframework.web.servlet.mvc.annotation.ResponseStatusExceptionResolv

er"></bean>

<bean

class="org.springframework.web.servlet.mvc.support.DefaultHandlerExceptionResolver"

></bean>

<!-- End -->

**注意：**

**一般开发中，我们都需要写上此标签（虽然从入门案例中看，我们不写也行，随着课程的深入，该标签还**

**有具体的使用场景）。**

**明确：**

我们只需要编写处理具体业务的控制器以及视图

## RequestMapping

**出现位置：**

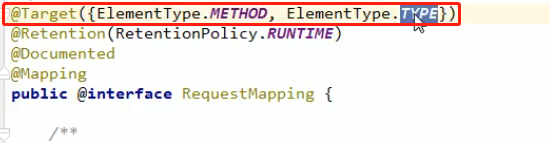
类上：

请求 URL 的第一级访问目录。此处不写的话，就相当于应用的根目录。写的话需要以/开头。

它出现的目的是为了使我们的 URL 可以按照模块化管理:

阅读@RequestMapping

@Target中表面该注解可以作用在方法上、也可以作用在类上



例如：

账户模块：

/account/add

/account/update

/account/delete

...

订单模块：

/order/add

/order/update

/order/delete

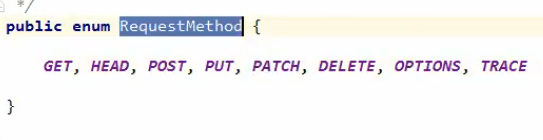
红色的部分就是把 RequsetMappding 写在类上，使我们的 URL 更加精细。

方法上：

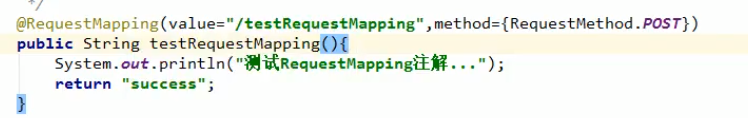
请求 URL 的第二级访问目录

超链接都是GET请求

枚举类



在方法中选择了post请求后就没办法再接受GET请求方式的超链接方式



就会出现405的错误

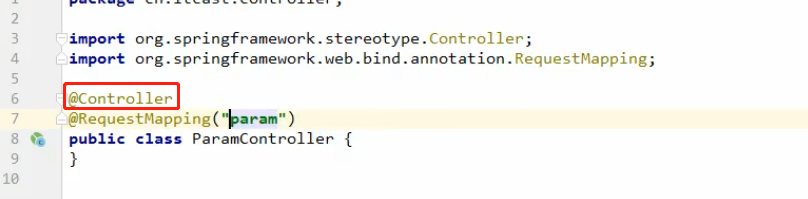


Params属性也是用来限制的

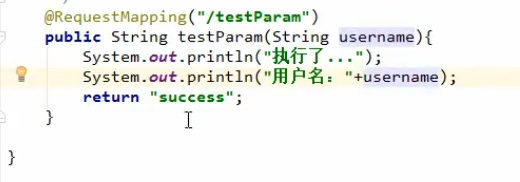


## 请求参数的绑定

使用@Controller作用是把类交给容器管理

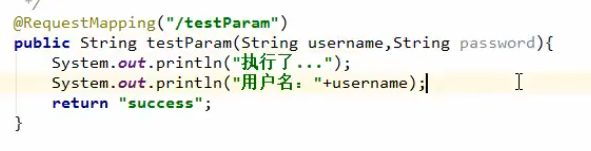


使用@controller过后会自动识别jsp或者html穿入的参数username，切记一定要同名才能自动识别，写错了封装不上

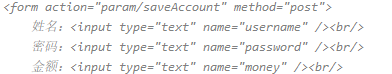


这是jsp写的超链接参数<herf = “”>超链接属性

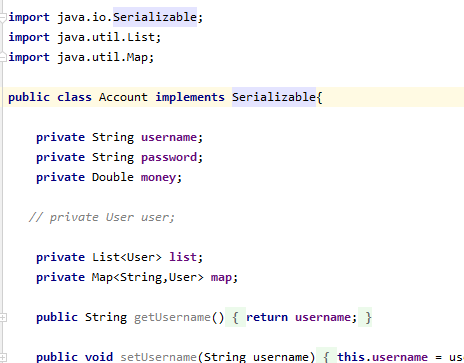




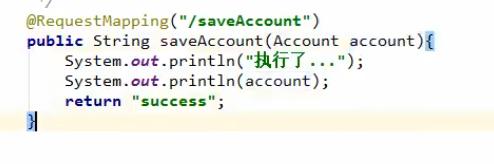
当属性多了的时候使用了JavaBean的方式来封装属性，jsp中使用表单名称也要一一对应



JavaBean要实现序列化接口

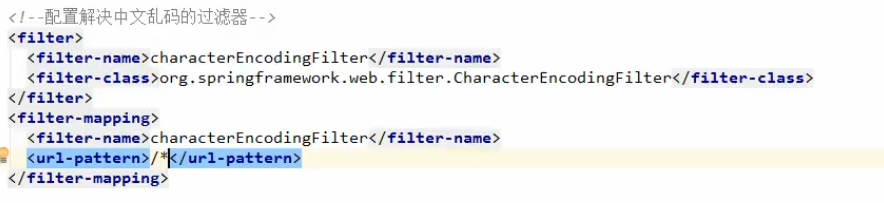


Account对象中

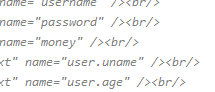


表单直接输入中文会乱码于是设计了控制器

/\*表示所有请求都要过滤一下



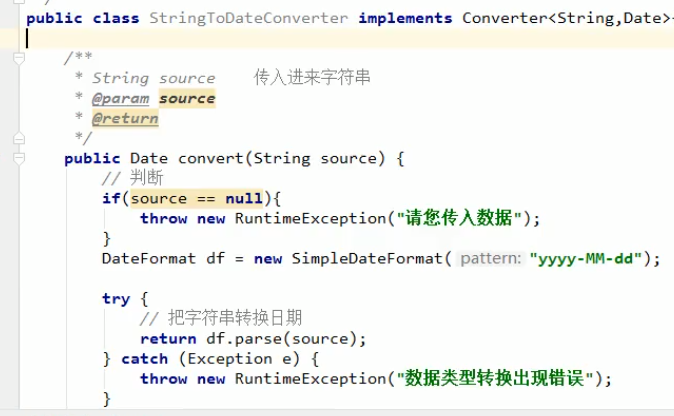
对象里面有应用的就写表达式user.uname user.age 等等



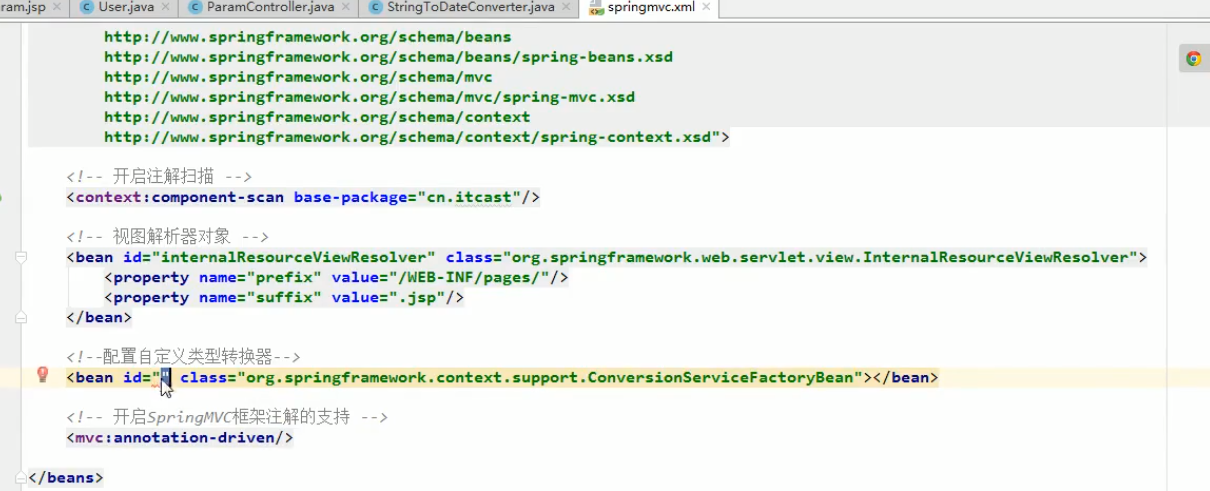
Controller识别jsp的前置是在controller.java中配置了@controller注解，和@RequestMapping

## 自定义类型转换

自定义类型转换器



Springmvc里面配置各种解析器，包括路径什么

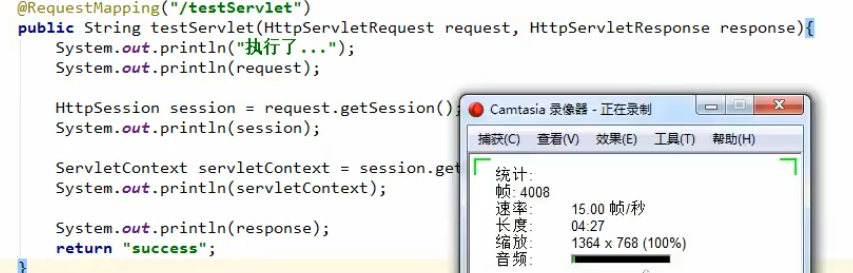


在mvc中默认有使其处理器映射器、处理器适配器、视图解析器生效

在springmvc配置转换类，把转换类扔给容器



获取请求对象的地址

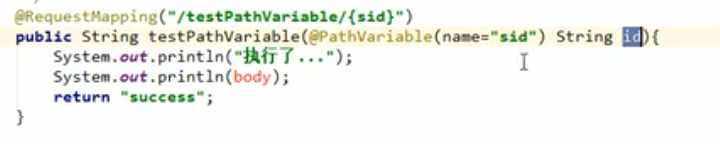


@RequestParam和@RequestBody都是帮忙获取请求体的值用于赋值给方法体

请求地址一样，根据请求方式的不同获得的类也不同，restfule方法请求路径都是一个，类似于方法重构



## 获取url路径的值



## 响应数据和结果视图



使用关键字的方式



前端控制器 mvc配置location和mapping



* 必须有，否则会使requestMapping失效。

location:指location指定的目录不要拦截，直接请求，这里指在根目录下的resources文件下的所有文件

mapping:值在resources文件下的所有文件(\*\*代表所有文件)

所有意思就是在根目录下resources的所有文件不会被DispatcherServlet拦截，直接访问，当做静态资源交个Servlet处理

## 响应json数据、异步请求 ajax

实现jsp按钮点击就会触发function函数，实现ajax，



Controller代码，@ResponseBody用于获取请求对象的值

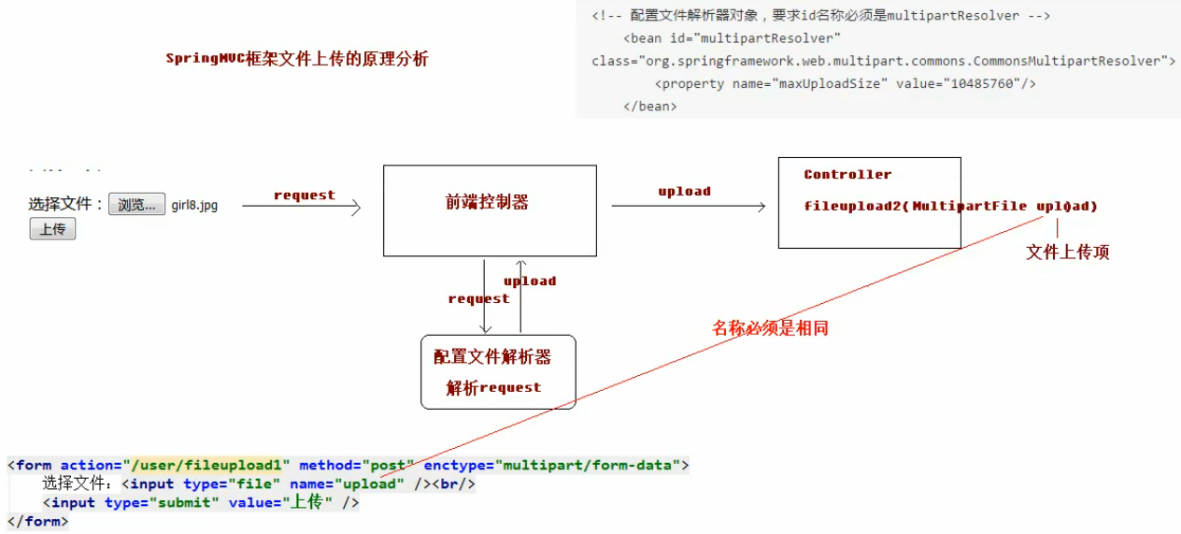


## 文件的上传与下载

Id值固定格式



配置

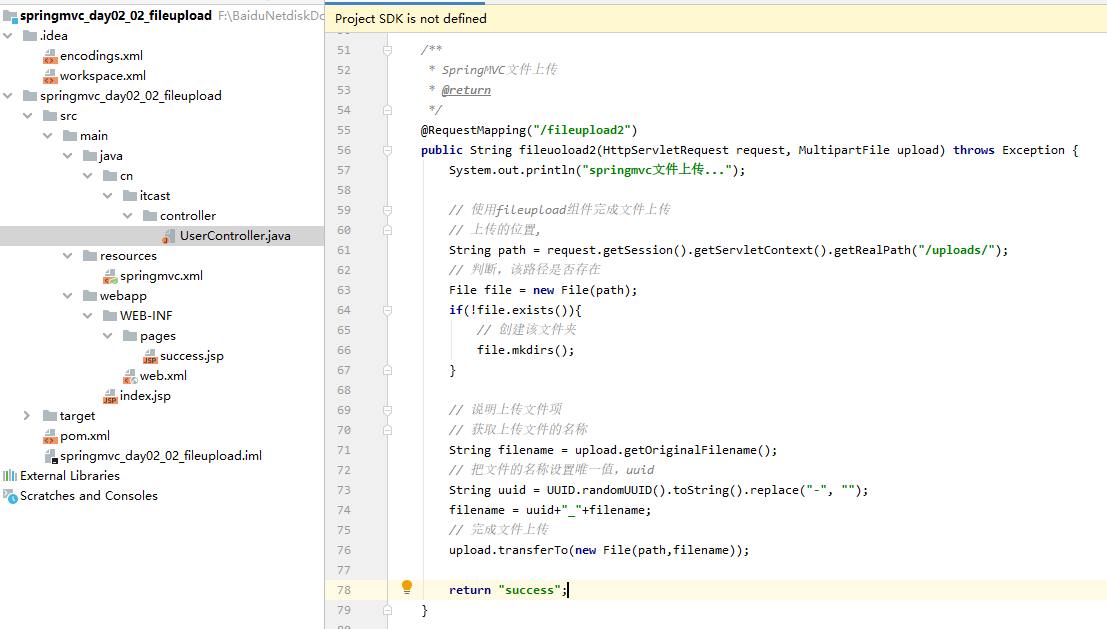


在controller中实现这个功能要在对应方法下的入参部分接受request请求和MultipartFile文件

上传路径是本地文件夹下的相对路径

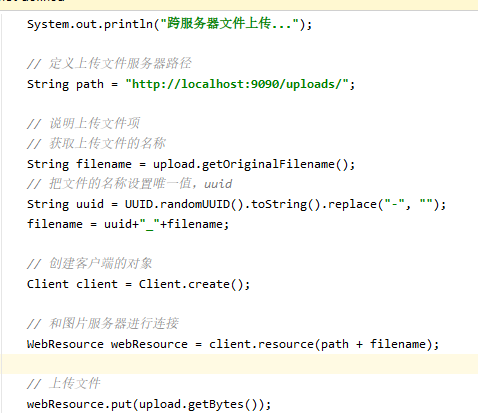


Return的值不是字符串，而是同名的jsp文件

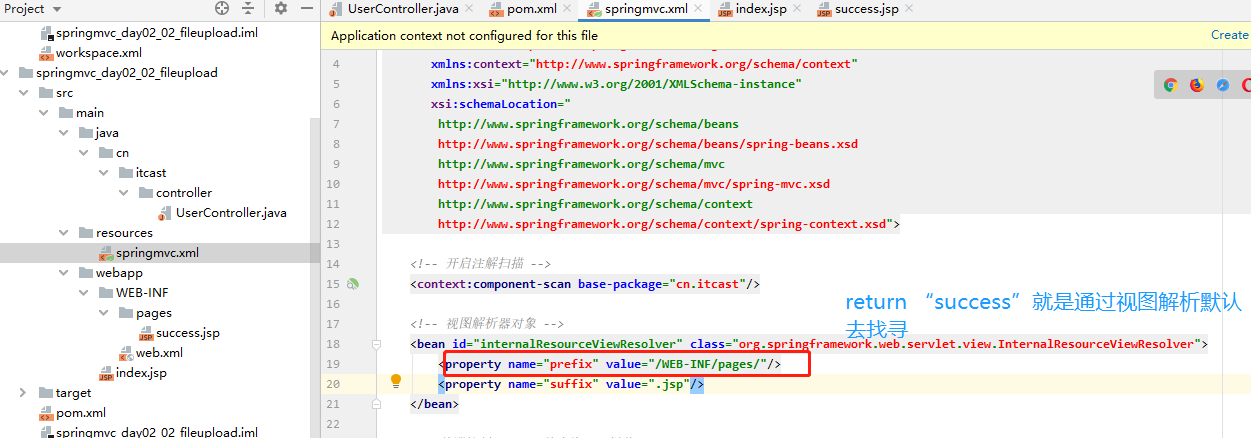


·

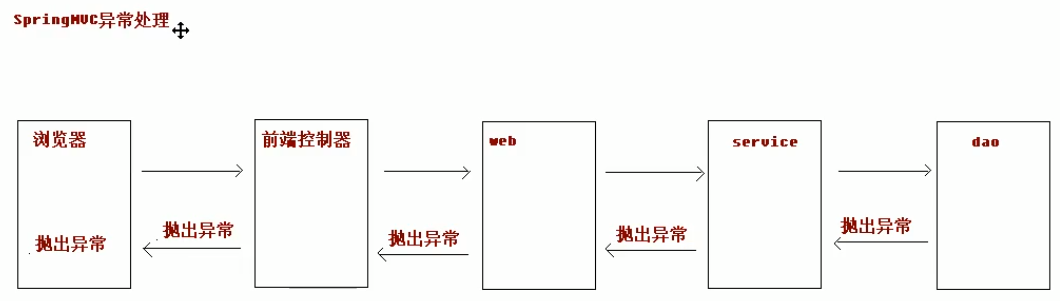
上传到其他服务器



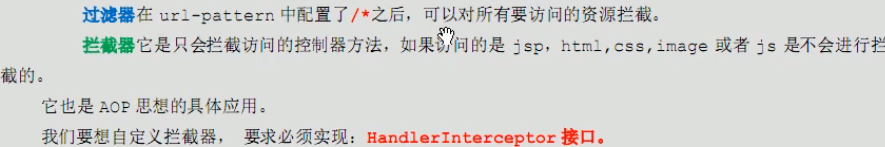
视图解析



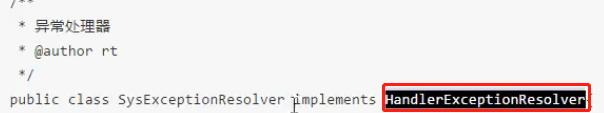
## 异常处理



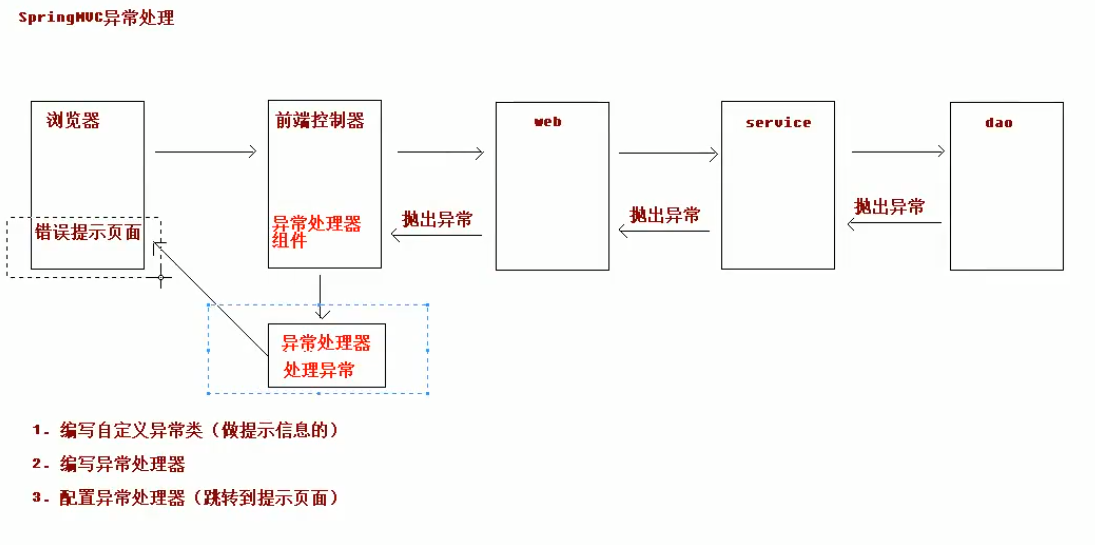
异常处理直接返回给浏览器十分不友好，于是引入拦截器和过滤器区别



自定义异常处理器必须实现这个接口



处理方式



拦截器配置



# SSM整合

整合的方向是使用spring去整合其他框架

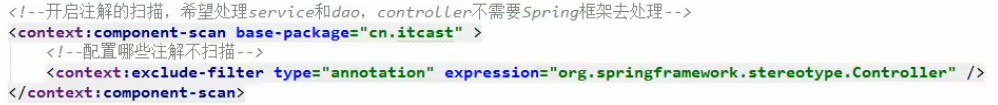
Javabean为什么非得实现序列化接口

实现 [serializable接口](https://www.baidu.com/s?wd=serializable%E6%8E%A5%E5%8F%A3&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHRknHu9uHm4PWTznWNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmknjfkn1cd)的作用是就是可以把对象存到字节流，然后可以恢复。所以你想如果你的对象没实现序列化怎么才能进行 [网络传输](https://www.baidu.com/s?wd=%E7%BD%91%E7%BB%9C%E4%BC%A0%E8%BE%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHRknHu9uHm4PWTznWNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmknjfkn1cd)呢，要 [网络传输](https://www.baidu.com/s?wd=%E7%BD%91%E7%BB%9C%E4%BC%A0%E8%BE%93&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHRknHu9uHm4PWTznWNb0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHmknjfkn1cd)就得转为字节流，所以在分布式应用中，你就得实现序列化，如果你不需要分布式应用，那就没那个必要实现序列化。

想使用注解，一定要在配置文件中开启注解扫描才可以，扫描配置



配置只扫描某些文件夹



把service这个类交给了IOC这个容器进行管理



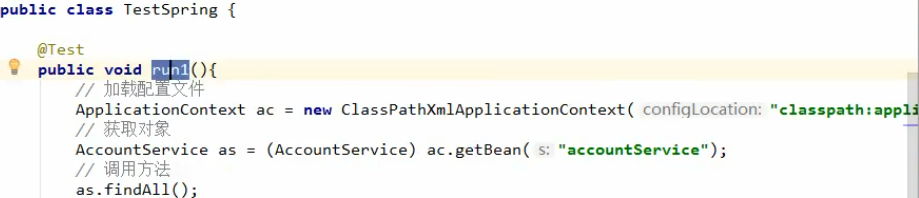
Junit中的@test

@Test注解的方法就是我们实际执行的测试方法。

实际上，测试类中默认的测试方法执行顺序：按照测试方法方法名字母排序顺序执行。

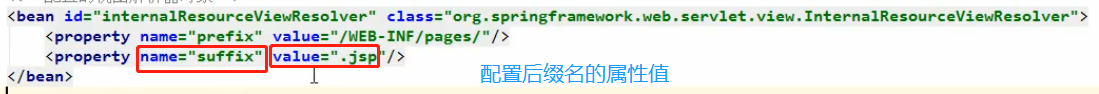
比如我们把测试方法testMethod2()方法名前加一个字母a，再执行下确认执行的顺序。

执行测试方法（此步骤是测试spring环境配置的效果）



测试类通过加载.xml配置文件，从容器中获取service类，再调用类的方法来实现功能

## Spring整合Spiringmvc

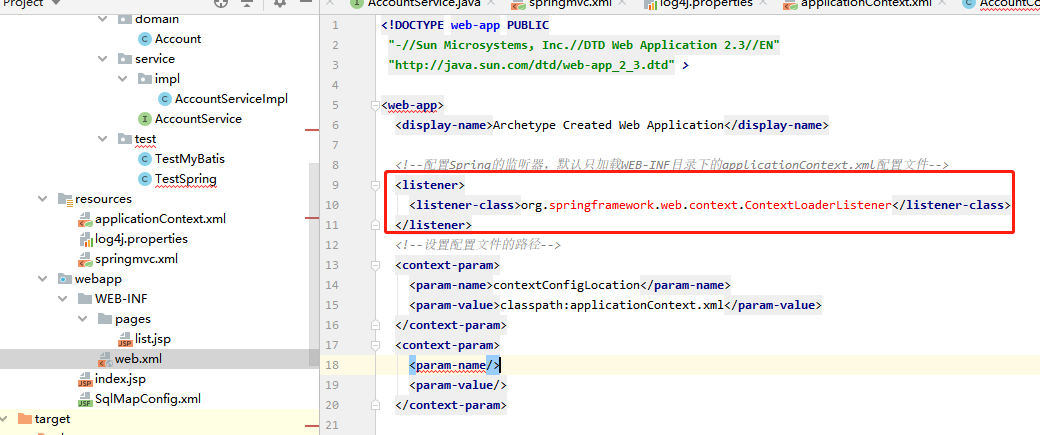


spring层有server

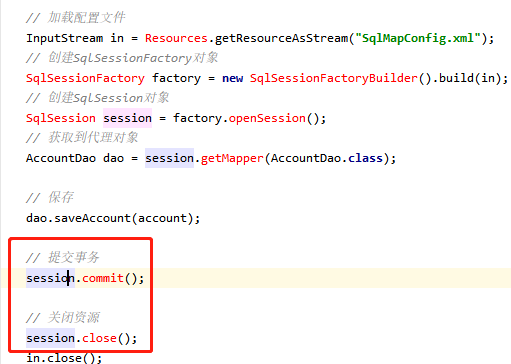
Mybatis层有dao

Springmvc层有Controller

整合的主要方式是在web.xml配置spring的监听器，让springmvc去监听spring

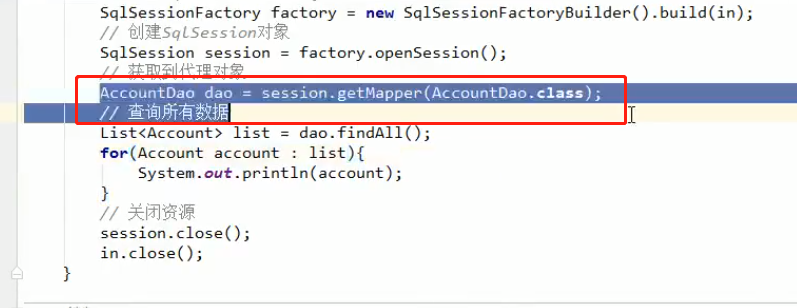


工厂创建的SqlSession对象里面有commit() 实现事务的提交和close()关闭资源



整合的核心点

就是如何把生成的代理对象如何保存到容器当中去



细化配置的三大核心步骤

**在spring的核心配置文件applicationContext.xml配置整和mybatis**



