**SringMVC**

Spring MVC是一个基于MVC架构的用来简化web应用程序开发的应用开发框架，它是Spring的一个模块，无需中间整合层来整合 ，它和Struts2一样都属于表现层的框架。在web模型中，MVC是一种很流行的框架，通过把Model，View，Controller分离，把较为复杂的web应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合。

**Springmvc的优点**

**它是基于组件技术的。**全部的应用对象,无论控制器和视图,还是业务对象之类的都是 java组件.并且和Spring提供的其他基础结构紧密集成.

**不依赖于Servlet API**(目标虽是如此,但是在实现的时候确实是依赖于Servlet的)

**可以任意使用各种视图技术,而不仅仅局限于JSP**

**支持各种请求资源的映射策略**

**它应是易于扩展的**

**SpringMVC与Struts2的主要区别**

**1、springmvc的入口是一个servlet即前端控制器，而struts2入口是一个filter过滤器。**

**2、springmvc是基于方法开发，**传递参数是通过方法形参，可以设计为单例或多例(建议单例)，**struts2是基于类开发，**传递参数是通过类的属性，只能设计为多例。

**3、struts2采用值栈存储请求和响应的数据**，通过OGNL存取数据，**springmvc通过参数解析器**是将request对象内容进行解析成方法形参，将响应数据和页面封装成ModelAndView对象，最后又将模型数据通过request对象传输到页面。 Jsp视图解析器默认使用jstl。

4、**性能上Struts1>SpringMvc>Struts2 开发速度上SpringMvc和Struts2差不多,比Struts1要高**

**Spring MVC的主要组件**

**前端控制器 DispatcherServlet**（不需要程序员开发）

作用：接收请求、响应结果，相当于转发器，DispatcherServlet是整个流程控制的中心，由它调用其它组件处理用户的请求，有了DispatcherServlet就减少了其它组件之间的耦合度。

**处理器映射器HandlerMapping**（不需要程序员开发）

作用：根据请求的URL来查找Handler。

**处理器适配器HandlerAdapter**

注意：通过HandlerAdapter对处理器进行执行，这是适配器模式的应用，通过扩展适配器可以对更多类型的处理器进行执行。在编写Handler的时候要按照HandlerAdapter要求的规则去编写，这样适配器HandlerAdapter才可以正确的去执行Handler。

**处理器Handler**（需要程序员开发）

作用：Handler是继DispatcherServlet前端控制器的后端控制器，在DispatcherServlet的控制下Handler对具体的用户请求进行处理。

**视图解析器 ViewResolver**（不需要程序员开发）

作用：负责将处理结果生成View视图，View Resolver首先根据逻辑视图名解析成物理视图名即具体的页面地址，再生成View视图对象，最后对View进行渲染将处理结果通过页面展示给用户。

**视图View**（需要程序员开发jsp）

作用：View是一个接口，它的实现类支持不同的视图类型（jsp，freemarker，pdf等等）。

**SpringMVC的控制器是不是单例模式,如果是,有什么问题,怎么解决**

是单例模式，所以在多线程访问的时候有线程安全问题，不要用同步，会影响性能，解决方案是在控制器里面不能写字段

**创建SpringMVC步骤**

**1、导入jar包：spring-webmvc**

**2、在web.xml中注册中央控制器DispatcherServlet**

****

**3、创建SpringMVC配置文件springmvc.xml，**该xml配置文件可以任意命名，需要跟第三步中的init-param中的param-value保持一致

**4、创建一个类去实现org.springframework.web.servlet.mvc.Controller接口，**通常称这样的类为Controller，它的作用有些类似servlet，或者可以认为在Spring MVC里面，就是使用了Controller来代替了servlet，它提供了比servlet更加丰富的功能。

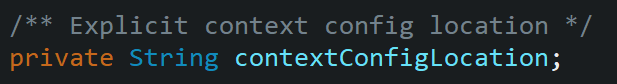


**5、在springmvc.xml配置文件中注册第五步创建的Controller**

****

**param-name为什么要填写contextConfigLocation**

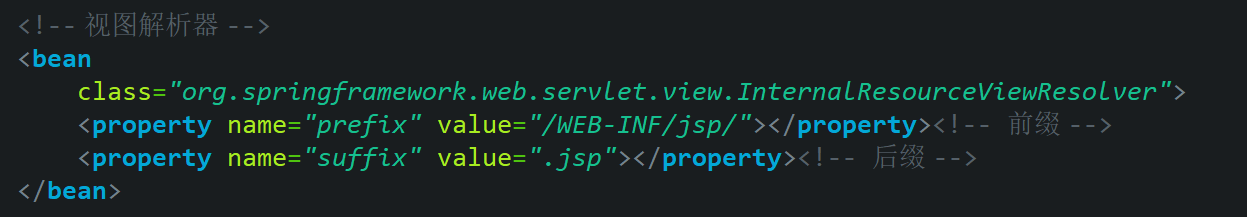
通过源码可以看到DispatcherServlet继承了FrameworkServlet，在这个类中有个contextConfigLocation成员变量，所以在param-name中将其命名为contextConfigLocation。

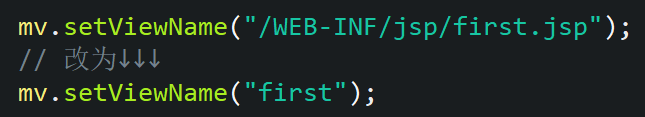
****

**配置视图解析器**

在controller中我们在ModelAndView中的setViewName方法里面传入要跳转的jsp的路径和名字，如果有多个controller的话，需要在每个里面都编写路径和jsp的名字，这样的话比较繁琐，这时可以通过使用Spring MVC给我们提供的一个视图解析器来解决这个问题。

**在配置文件中添加**





**url-pattern的设置**

之前在web.xml文件中配置DispatcherServlet的时候，将url-pattern配置为\*.do的方式，除了这种方式之外还可以配置为其他任意方式：\*.action、\*.abc、\*.123

只要请求的url中包含配置的url-pattern，该url就可以到达DispatcherServlet。工作中通常都将url-pattern配置为\*.do的方式。不过现在这种.do的配置方式不如以前多了，因为一种叫做restful风格的URL越来越流行了。

**设置url-pattern为/\***

如果将url-pattern设置为/\*之后，web项目中的jsp都不能访问了会报出404的错误，这是因为DispatcherServlet会将向JSP页面的跳转请求也当作是一个普通的Controller请求，会对其进行处理，而此时是找不到与其相应的controller。简单来说就是/\*这种配置会被web服务器匹配到.jsp上面。在实际开发中最好不要这样配置url-pattern。

**设置url-pattern为/**

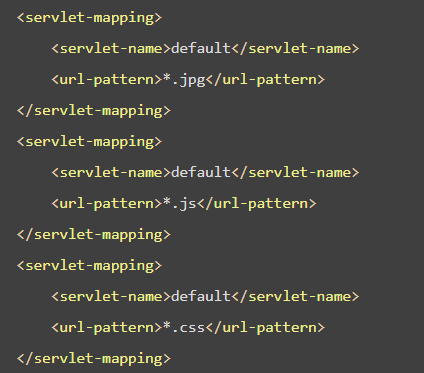
如果将url-pattern设置为/之后，只要是在web.xml文件中找不到匹配的URL，它们的访问请求都将交给DispatcherServlet处理，静态资源：css文件，js文件，图片也会被拦截并交给DispatcherServlet处理。

该配置方式不会拦截jsp文件和jspx文件，因为这个在tomcat中的conf目录里面的web.xml文件中已经添加的相应的处理方式了，他会交给org.apache.jasper.servlet.JspServlet来处理。即我们可以正常访问系统中的jsp文件。现在restful风格的URL越来越流行，这种配置方式使用的也越来越多了。

**三种解决静态资源不能访问的问题**

**使用defaultServlet**

打开tomcat中conf/web.xml,在这个文件中有一个叫做DefaultServlet的配置,当系统找不到处理某次url请求该交由谁处理的时候，就会交给这个servlet处理。可以通过使用这个DefaultServlet来处理静态资源，在该文件中添加下面配置，要添加在DispatcherServlet的前面，这样系统就会将带有下面后缀名的请求交给defaultservlet来处理



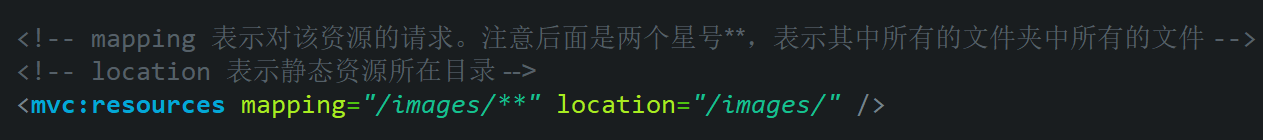
**使用mvc:default-servlet-handler**

在springmvc.xml文件中添加，该方式会对所有的请求进行处理，然后交由相应的servlet，这种方式其实最终也是由DefaultServlet来处理



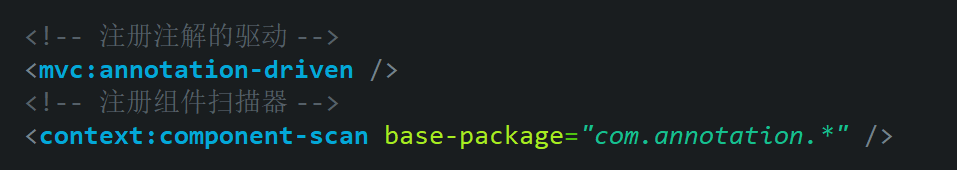
**使用mvc:resources（常用、效率高）**

在SpringMVC中提供了mvc:resources标签用来解决静态资源无法访问的问题，只需要在配置文件中添加即可，这样会交给SpringMVC的ResourceHttpRequestHandler类来处理



**注解（工作常用）**

**注册扫描器**

****

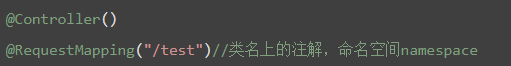
如果没有添加注解驱动，在springmvc.xml文件中配置了静态资源，即使用了mvc:resources标签，访问controller的时候会出现404错误。

**@Controller：**表示当前类为一个Controller，该类可以不继承Controller接口

**@RequestMapping：**表示当前方法为Controller中的方法，该方法要对value属性所指定的URL进行处理与响应，被注解的方法的名称是可以随意命名的。**当有多个请求可以匹配该方法时，可以写上一个String类型的数组**。

**C:\Users\hasee\Desktop\QQ浏览器截图20190218152959_看图王.pngC:\Users\hasee\Desktop\QQ浏览器截图20190218152959_看图王(1).png**

**@RequestMapping注解还可以定义在类上面**



**@RequestMapping中的通配符**

**/test\*.do**

表示请求的url中只要是以test开头就可以被当前方法处理。

**/\*test.do**

表示请求的url中只要是以test结尾就可以被当前方法处理。

**tt/\*/test.do**

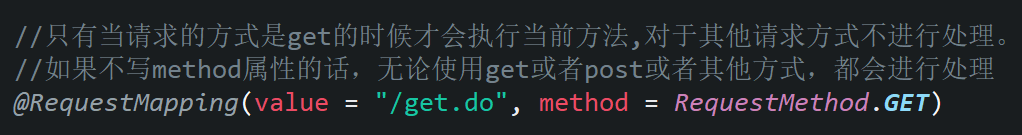
表示在test.do的前面，只能有两级路径，第一级必须是tt，而第二级随意。

**tt/\*\*/test.do**

表示在test.do的资源名称前面，必须以tt路径开头，而其它级的路径包含几级，各级又叫什么名称，均随意。

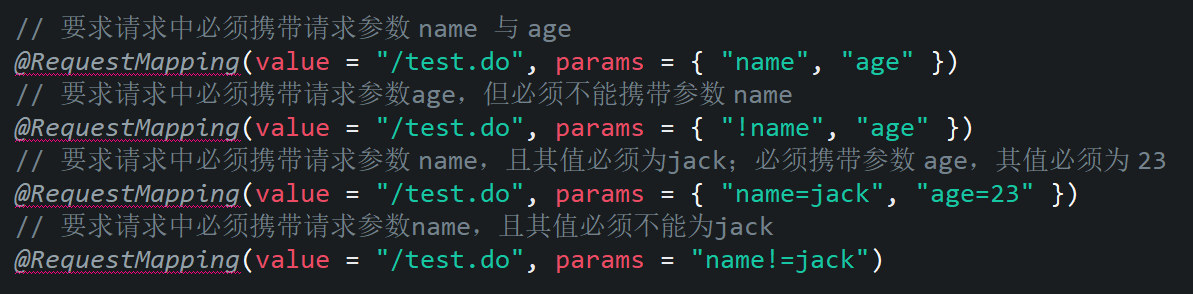
**请求的提交方式**

在@RequestMapping中有一个method属性，改属性可以设置接收请求的提交方式，**RequestMethod是一个枚举类型**，里面包含了大部分的提交方式。



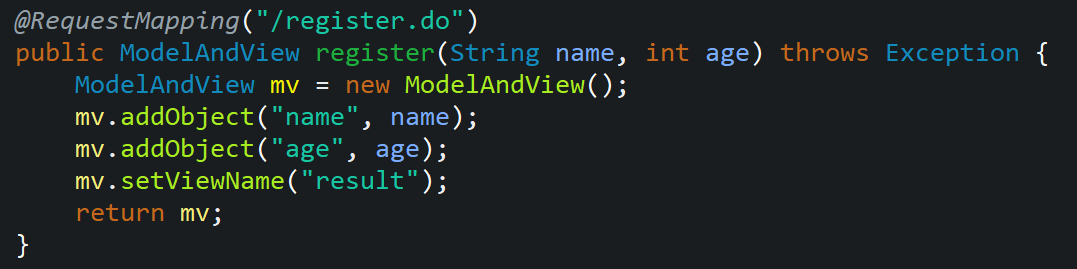
**请求中携带的参数**

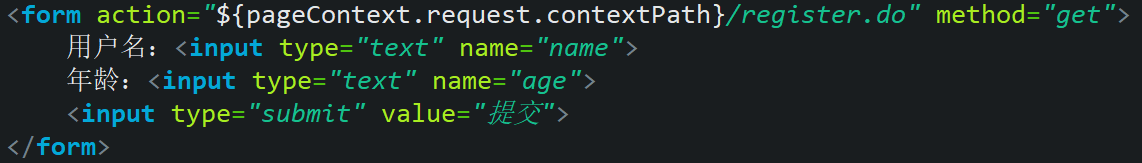
在RequestMapping中还有一个属性是params，通过这个属性我可以指定请求中必须携带的参数。



**接收表单提交的参数**

当方法的参数列表中的参数名和表单中的参数名一致时，spring MVC会自动为其赋值，通过这种方式就可以在controller中获取表单提交的数据了





在controller的方法里面，可以写多个参数，也可以一个都不写，参数列表中如果没有表单中的参数名则不会自动赋值，参数列表中如果出现表单中没有的参数名则为null，不过spring mvc只会自动为以下五个参数进行自动赋值：**HttpServletRequest、HttpServletResponse、HttpSession、请求携带的参数、用于承载数据的Model**

**针对Post方式的乱码问题**

可以使用spring MVC为开发者提供的CharacterEncodingFilter来解决乱码问题，这个其实就是一个过滤器。在web.xml文件中进行配置，最好将其设置在其他过滤器之前，因为过滤器是按照web.xml中的顺序执行的



**针对Get方式的乱码问题**

由于参数是通过URL传递的，所以上面通过request设置的编码格式是不起作用的。

**1、可以在每次发生请求之前对URL进行编码**

例如：Location.href="/encodeURI"("http://localhost/test/s?name=中文&sex=女");

**2、修改tomcat的配置文件server.xml**

<Connector URIEncoding="utf-8" connectionTimeout="20000" port="8080" protocol="HTTP/1.1" redirectPort="8443"/>

**3、对参数进行重新编码**

String userName = new String(request.getParamter("userName").getBytes("ISO8859-1"),"utf-8")

**参数名不一致的情况**

在实际开发中有可能出现表单中参数名和spring mvc方法中的参数名不一致，可以使用**@RequestParam()**注解来解决这个问题

在springMVC方法的参数列表中添加@RequestParam()注解



**value：指定请求参数的名称**，即表单中input的name值。

**required：指定该参数是否是必须传入的，boolean类型。**若为true，则表示请求中所携带的参数中必须包含当前参数，否则会报400错误。若为 false，则表示有没有均可。

**defaultValue：指定当前参数的默认值。**如果请求中没有该参数，则当前方法参数将取该默认值。

**使用对象来接收表单提交的参数**

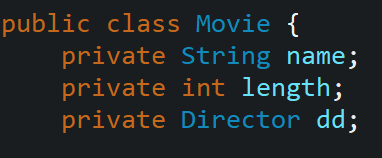
倘若表单中的参数较多的话，可以定义一个javabean，将表单中的参数都写到javabean的属性里面，然后将这个bean作为参数写到controller的方法中。**bean里面的属性名跟表单的name保持一致。**



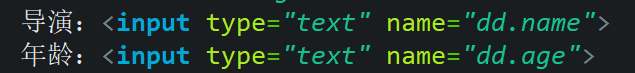
**对象属性参数的接收**

当请求中的参数是一个对象属性中的属性时，我们可以在表单中的name中写上**对象属性.属性**的方式进行处理。

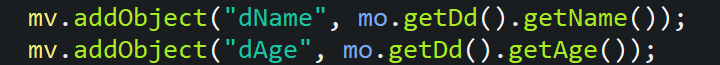
**bean**



**jsp**



**controller**



**PathVariable注解**

Controller除了可以接收表单提交的数据之外，还可以获取url中携带的变量，即路径变量，此时需要使用@PathVariable注解来设置。



**Controller中方法的返回值类型**

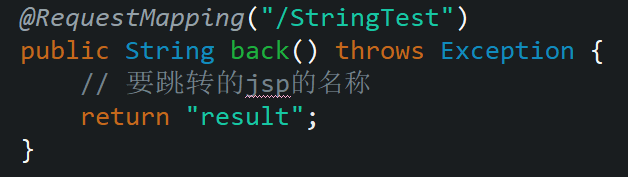
Controller方法的返回类型除了ModelAndView，还可以返回其他类型的对象，在实际应用中需要根据不同的情况来使用不同的返回值：**ModelAndView、String、void、自定义类型，**一般返回String较好。

**1、返回ModelAndView**

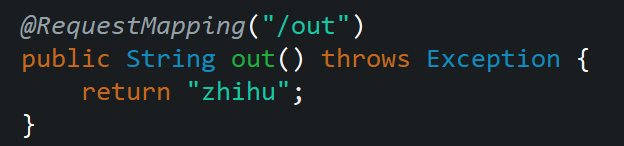
需要跳转到jsp或其他资源，且又要传递数据，此时方法返回ModelAndView比较方便。如果只传递数据，或者只跳转jsp或其他资源的话，使用ModelAndView就显得有些多余了。

**2、返回String类型**

**返回内部资源**



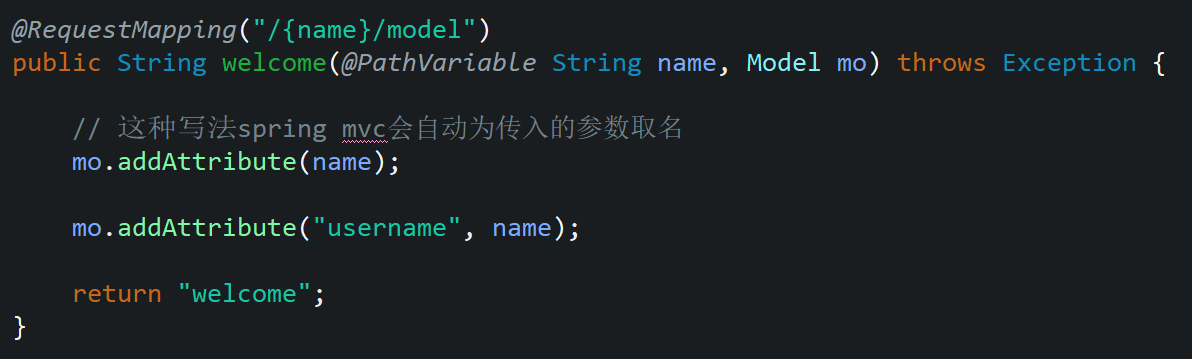
**返回外部资源**





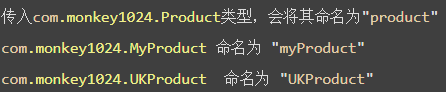
**Model对象**

Model是一个接口，写在controller的方法中的时候，spring mvc会为其进行赋值。可以使用Model对象来传递数据，也就是说我们可以使用Model传递数据并且将方法返回值设置为String类型，通过这种方式实现与方法返回ModelAndView一样的功能。



**取名**

spring mvc会根据传入的参数对其进行取名。name为String类型则取名为string。**不常用**。

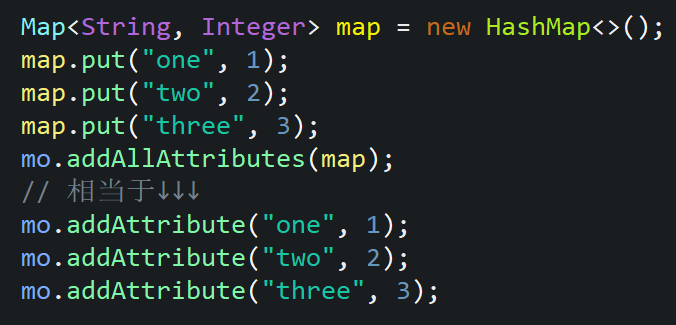


**Model接口中还有两个方法**

**addAllAttributes(Collection<?> attributeValues)：**会将传入的list中的数据对其进行命名



**addAllAttributes(Map attributes)：**会将map中的key作为名字，value作为值放入到model对象中



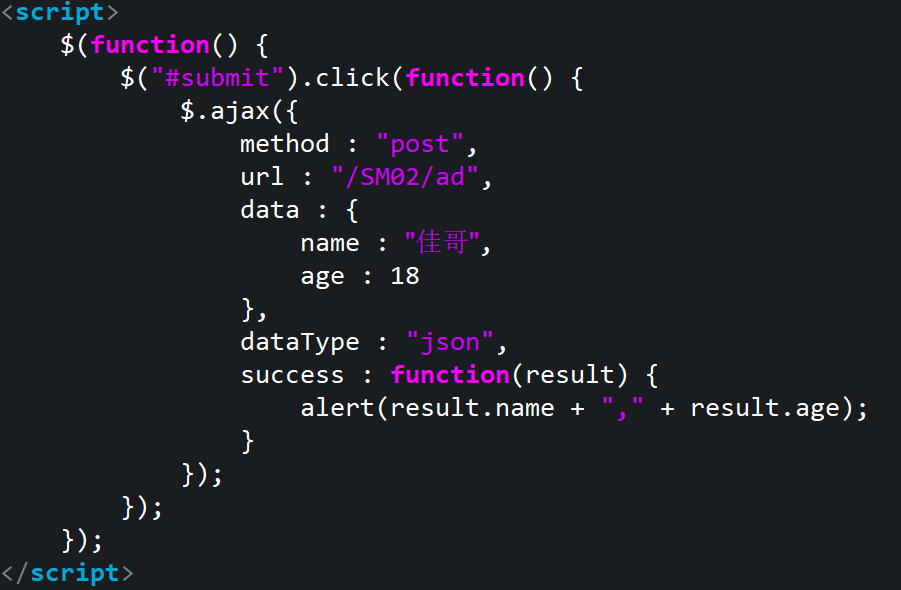
**3、返回void**

如果不用spring mvc完成资源的跳转，可以将controller中的方法返回值设置为void。一般情况下有两个应用场景：**通过原始的servlet来实现跳转，ajax响应**

**原始的servlet来实现跳转**



**ajax响应**

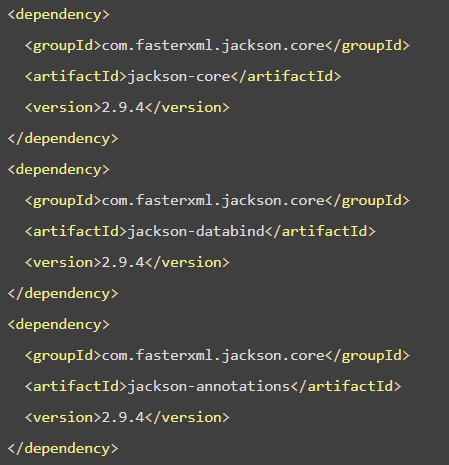




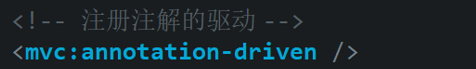
**4、返回Object类型**

倘若需要controller中的方法返回Object类型，需要先配置下面内容

**添加jackson的jar包，在Spring mvc中使用了jackson来进行json数据格式的转换。**

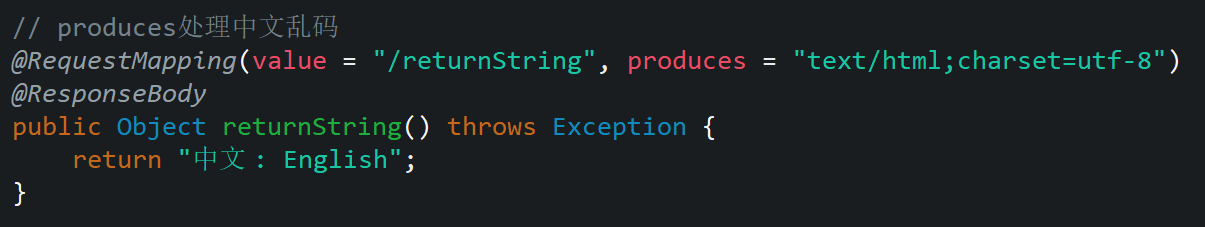


**在springmvc.xml文件中添加注解驱动。**



**例如返回字符串**

在controller方法中返回字符串，spring mvc会根据字符串跳转到相应的jsp中。这里返回的字符串会添加到响应体中传递到jsp页面中，此时需要在方法上添加一个注解**@ResponseBody**即可。

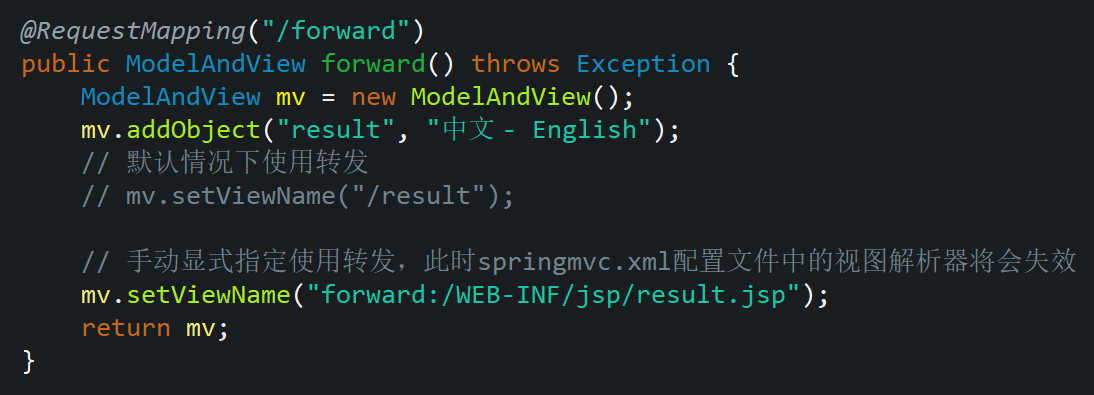


**转发与重定向**

spring mvc底层其实就是一个servlet，因此在spring mvc中也存在转发和重定向的概念。对于转发的页面，可以是在WEB-INF目录下的页面；而重定向的页面，是不能在WEB-INF目录下的。因为重定向相当于用户再次发出一次请求，而用户是不能直接访问WEB-INF目录下的资源的。根据所要跳转的的资源，可以分为跳转到jsp页面和跳转到其他controller。

**返回ModelAndView时的请求转发**

当controller中的方法返回ModelAndView的时候，默认是使用的转发。也可以显式的指出要使用转发，**需在setViewName()指定的视图前添加forward：**，此时springmvc.xml文件中配置的视图解析器将会失效，即需要在setViewName中写上jsp相对于项目根的路径。**如果要跳转到其他controller中，需要显式的写上forward**

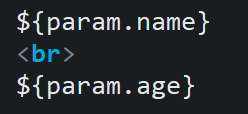


**返回ModelAndView时的重定向**

如果要实现重定向，**需在setViewName()指定的视图前添加redirect：**，此时springmvc.xml文件中配置的视图解析器将会失效，即需要在setViewName中写上jsp相对于项目根的路径。**重定向的方式在request域中的数据会失效，此时可以通过ModelAndView来传递数据，但是只能传递基本数据类型和String类型**，因为spring mvc会将传递的数据以请求参数的方式放到url的后面，**此时任何数据类型都会被转换为普通的字符串**。另外，我们在jsp中取值的时候需要使用 EL 表达式中的请求参数param读取。除此之外，你还可以将数据放到HttpSession域中。在重定向到其他controller中的方法的时候，只要保证两者的参数名一致即可实现数据的传递，可以传递对象。

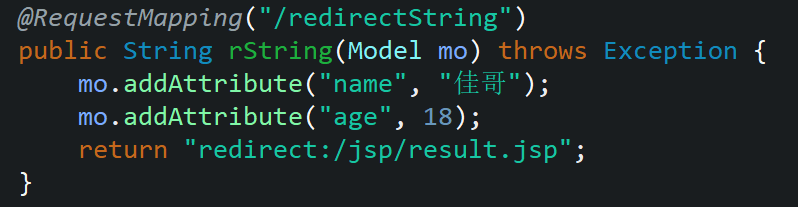


**jsp取值**



**返回String类型的重定向**

这里要想传递数据的话，需要使用Model实现，因为这里的数据会放在url中，所以只能传递基本数据类型和String类型，jsp需用param取值。



**返回void的重定向和转发**

当方法没有返回值即返回void的时候，重定向和转发操作都是使用的servlet的api。

**转发**



**重定向**



**SpringMvc怎么和AJAX相互调用的？**

通过Jackson框架就可以把Java里面的对象直接转化成Js可以识别的Json对象。具体步骤如下 ：

加入Jackson.jar

在配置文件中配置json的映射

在接受Ajax方法里面可以直接返回Object,List等,但方法前面要加上@ResponseBody注解。

**Spring MVC的异常处理**

可以将异常抛给Spring框架，由Spring框架来处理；我们只需要配置简单的异常处理器，在异常处理器中添视图页面即可。

**异常处理器的配置方法（常用三种）**

使用spring mvc中的异常处理器 SimpleMappingExceptionResolver

使用自定义异常处理器

使用异常处理注解

**SimpleMappingExceptionResolver**

当系统出现异常之后，可以让spring mvc跳转到指定的jsp中，这样子对于用户来说比较体验好，对于开发者来说也比较好定位问题。

首先自定义一个异常，再定义一个controller，在里面抛出异常，修改配置文件



**exceptionMappings：**Property类型属性，用于指定具体的不同类型的异常所对应的异常响应页面。Key为异常类的全名，value则为响应页面路径，如果配置了视图解析器的话，那会使用视图解析器中的配置。

**defaultErrorView：**指定默认的异常响应页面。若发生的异常不是exceptionMappings中指定的异常，则使用默认异常响应页面。

**exceptionAttribute：**捕获到的异常对象，一般异常响应页面中使用，在el表达式中可以获取到value中的值。

**自定义异常处理器**

倘若需要在项目中捕获特定的异常，然后再根据捕获的异常做一些操作的时候，按照以前的写法，我们需要在每次捕获异常之后或者抛出之前进行操作，这段代码就会重复的出现在很多类里面，导致代码冗余，此时可以通过自定义异常处理器来解决。

自定义异常处理器，需要实现**org.springframework.web.servlet.HandlerExceptionResolver**接口，并且该类需要在springmvc.xml配置文件中进行注册。

**首先来自定义一个异常实现HandlerExceptionResolver接口，只要有异常发生，都会自动执行接口方法resolveException()。**



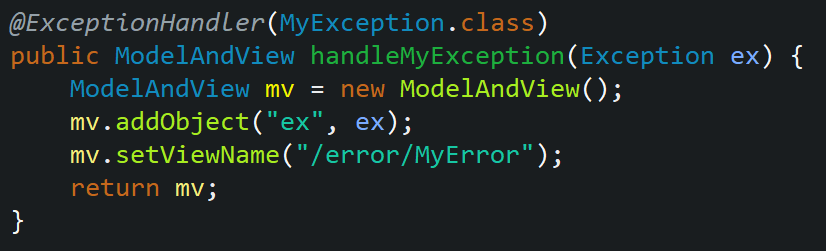
**在springmvc.xml文件中配置异常处理**



**使用注解处理异常**

使用注解**@ExceptionHandler**可以将一个方法指定为异常处理方法，该注解只有一个可选属性value，是一个Class<?>数组，用于指定该注解的方法所要处理的异常类，当controller中抛出的异常在这个Class数组中的时候才会调用该异常处理方法。

而被注解的异常处理方法，其返回值可以是ModelAndView、String，或void，方法名随意，方法参数可以是Exception及其子类对象、HttpServletRequest、HttpServletResponse等。系统会自动为这些方法参数赋值。



上面使用**@ExceptionHandler**注解标注了一个处理MyException的异常，不过只有在当前的controller中抛出MyException之后才会被该方法处理，其他controller的方法中抛出MyException异常时候是不会被处理的，解决这个问题的办法就是**单独定义一个处理异常方法的Controller，然后让其他Controller来继承它，但是这样做的弊端就是继承这个类之后就不能继承其他类了。**

**三种异常处理方式的总结**

**使用spring mvc中的异常处理器 SimpleMappingExceptionResolver**

这种方式会产生大量的冗余代码，不建议使用

**使用自定义异常处理器**

这种方式将异常处理统一编写到一个类中，便于管理和维护，**建议使用**。

**使用异常处理注解**

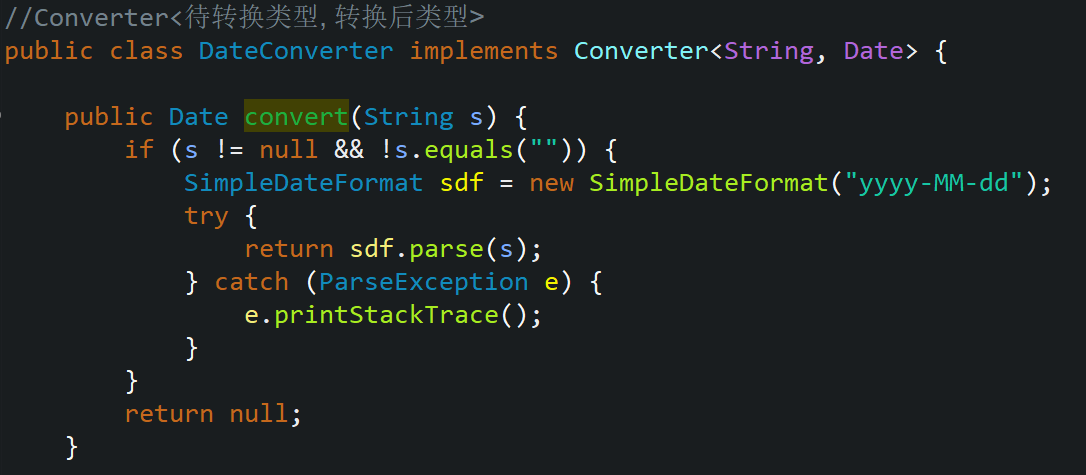
如果将异常处理的方法都放到一个基类中，其他类在继承这个类之后就不能继承其他类了，扩展性较差。

**SpringMVC中的类型转换器**

从前台提交到controller中方法的表单数据会根据在方法参数中定义的类型来自动转换，不用再强制的去转换。这个是通过**spring mvc中的默认类型转换器（converter）**来实现的，不过这些默认转换器不是万能的，有些类型就不能转换，比如日期类型，此时可以自定义一个类型转换器来进行日期类型的转换。

**自定义类型转换器**

创建一个类实现Converter接口，该接口中的泛型，前面的类是待转换的类型，后面的是转换之后的类型。

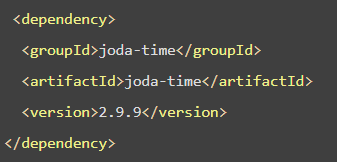


**配置springmvc.xml文件**



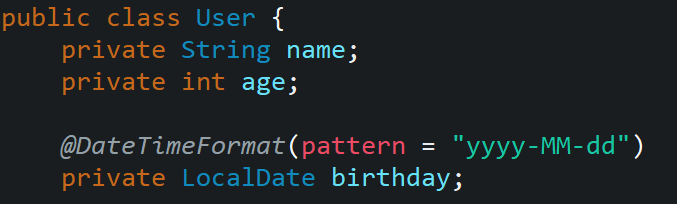
**@DateTimeFormat注解**

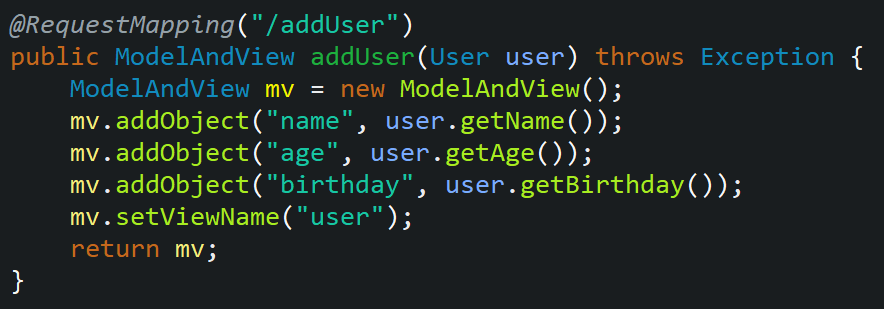
通过该注解可以进行日期类型的转换，这样就不用单独编写日期类型的转换器。@DateTimeFormat注解支持joda time，在jdk7之前的api中，关于日期处理相关的类都不太好用，而joda time提供非常好用的日期处理相关的api，因此joda time深受很多开发者的喜爱，在jdk8中已经将joda time的类纳入。如果使用的是jdk8（不含）之前的版本，并且要使用@DateTimeFormat注解的话，需要引入joda time相关的jar包。





还可以将日期转换为LocalDate类型，需要注意的是，如果要将日期转换为LocalDate类型，那么需要将这个属性写到javabean里面，不能使用controller中的方法参数的方式，此时会报出错误。

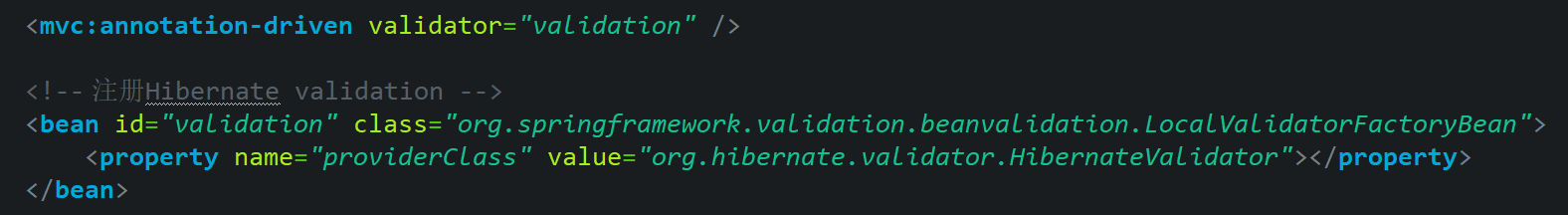


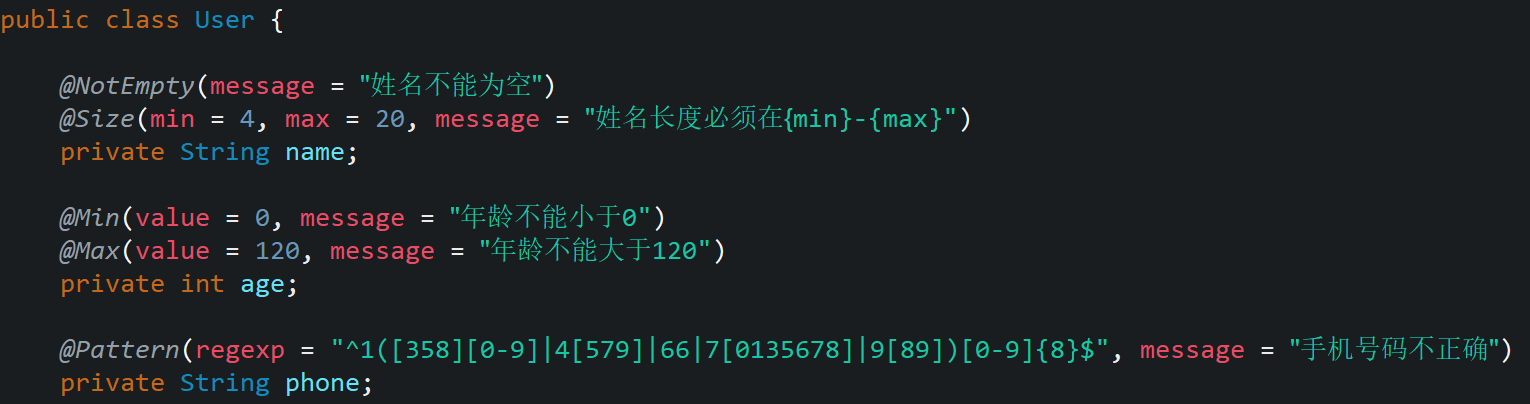


**数据校验**

**永远不要相信用户的输入**，在开发的系统凡是涉及到用户输入的地方，都要进行校验，这里的校验分为前台校验和后台校验，前台校验通常由javascript来完成，后台校验主要由java来负责。可以通过spring mvc+hibernate validator。在java中有一个bean validation的数据验证规范，该规范的实现者有很多，其中hibernate validator使用的较多一些，这里的hibernate validator是hibernate框架下的一款用于数据校验的框架，以前我们统称的hibernate一般特指的是hibernate orm。

**springmvc.xml配置**





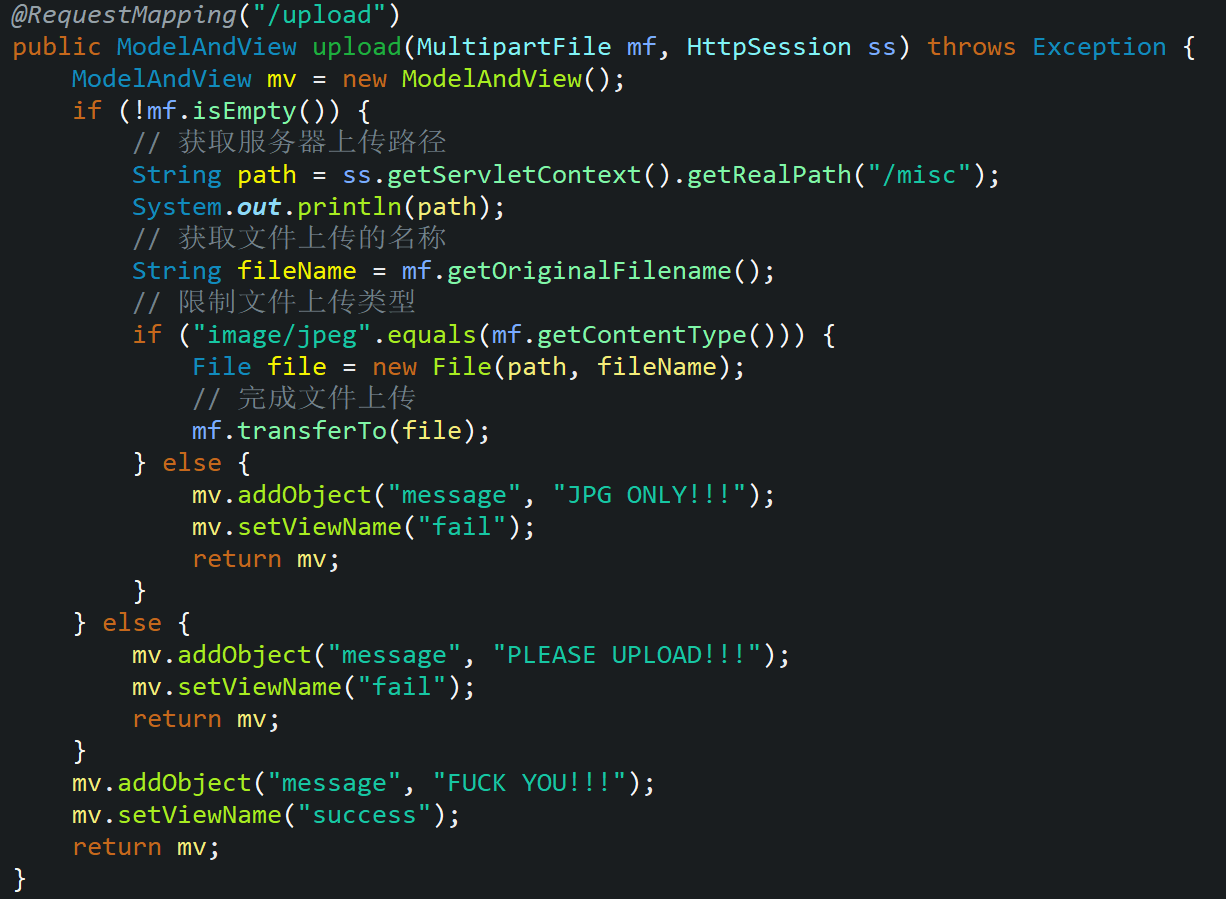




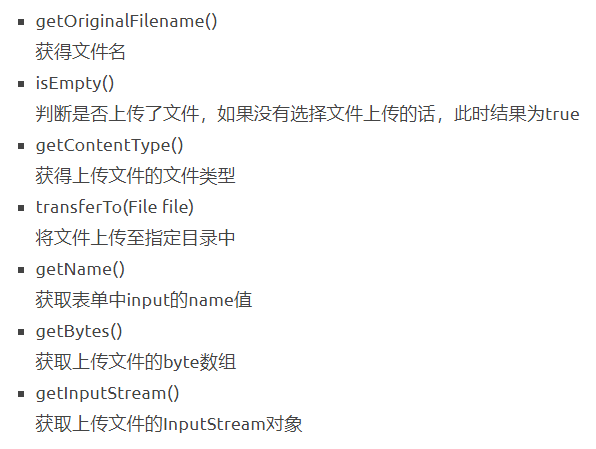
**文件上传**

**编写文件上传的表单和controller**

编写处理文件上传的controller，在处理文件上传的方法中需要添加MultipartFile类型的参数，MultipartFile本身是一个接口。**表单中上传文件的input的type需写file**。方法中的另外一个参数HttpSession的主要作用就是**获取服务器中用来存放上传文件的路径**。



**MultipartFile中的方法**



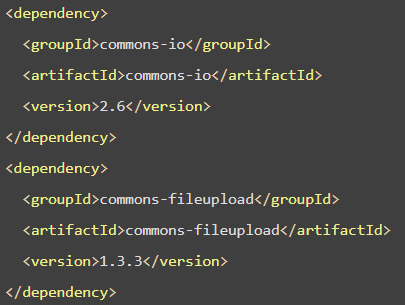
在spring mvc中支持两种上传文件的方式

**使用apache的commons-io和commons-fileupload实现文件上传**

**使用servlet3.0实现文件上传**

**使用commons组件实现文件上传**

导入jar包



在springmvc.xml配置文件中添加multipartResolver，里面的id必须要写成multipartResolver，会有DispatcherServlet来调用，可以在这里设置上传文件大小、字符编码等内容。



**设置临时上传文件目录的作用**

**提高安全性**

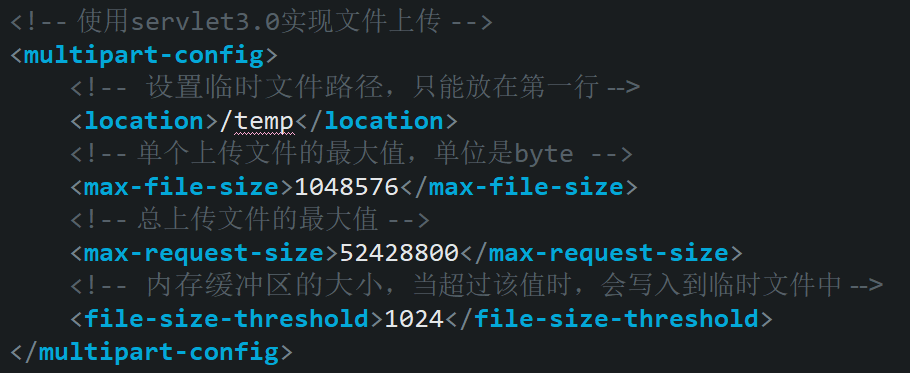
客户端上传的文件直接传到临时目录，这样子对于客户端来说隐藏了真实的文件存放目录

**便于管理**

当用户取消上传或上传失败的话，直接操作临时目录即可，无需再去修改真实目录中的文件。

**使用servlet3.0实现文件上传**

在web.xml中的中央控制器的servlet配置，**location的标签需要写在multipart-config中的第一行。**



**在springmvc.xml文件中配置multipartResolver**

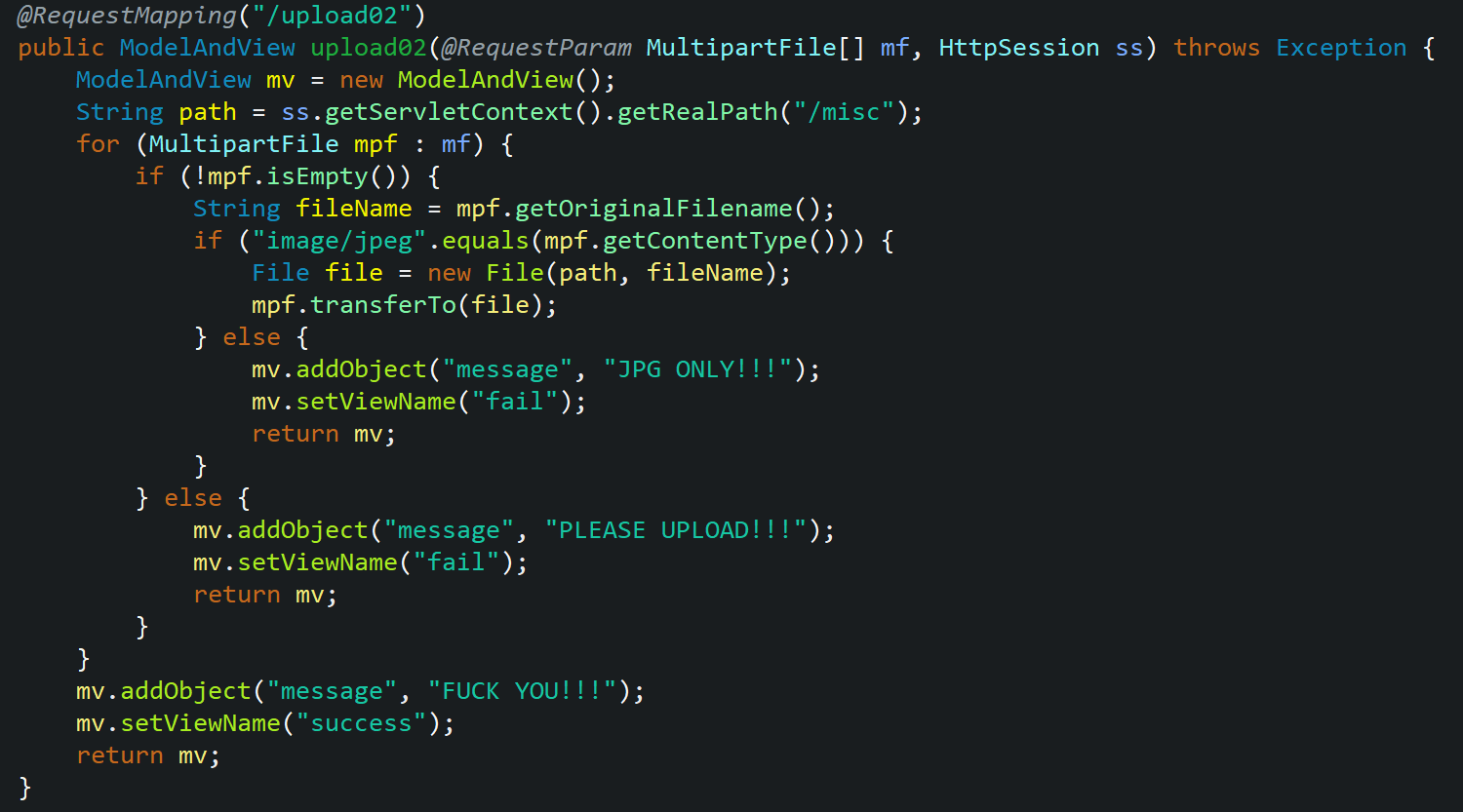


**上传文件超出设定大小的异常处理**

当上传文件超出指定大小时，会抛出MaxUploadSizeExceededException异常。可以通过spring mvc的异常处理方式来进行处理，给用户一个友好的提示信息。

**上传多个文件**

jsp中多个input的name要相同，type为file类型。在controller中添加方法，方法参数中需要添加MultipartFile[]，并且要在其前面加上@RequestParam注解



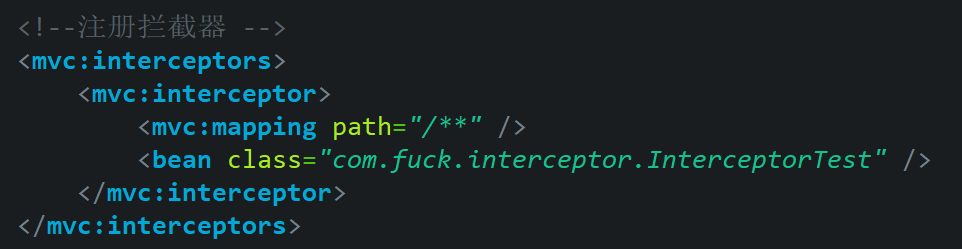
**拦截器**

拦截器（interceptor）是spring mvc中提供的一种类似filter过滤器的技术，只不过拦截器只能拦截controller的请求，而filter可以过滤所有请求，其他内容大体上两者是近似的。

**定义拦截器**

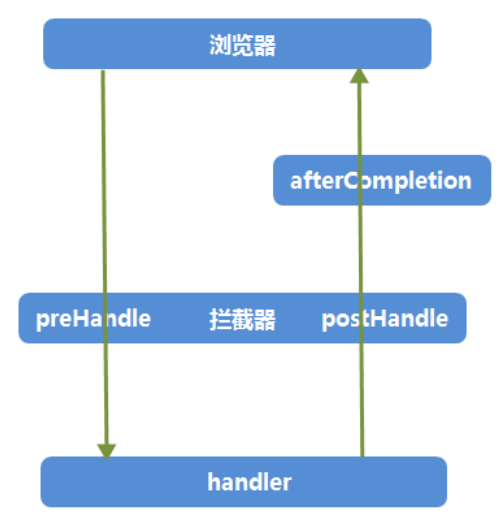


**在springmvc.xml文件中注册拦截器，/\*\* 表示对所有controller拦截**

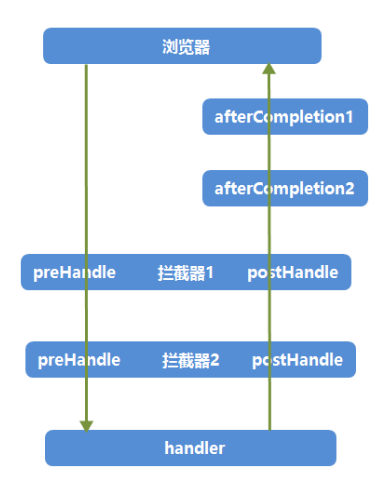


**拦截器的工作流程**

整体的流程如图所示，这里的handler实际上是在DispatcherServlet和Controller之间调用的一个东西，实际上DispatcherServlet是通过handler找到处理当前请求的Controller中的方法，然后再去调用该方法的。



**多个拦截器时，会形成拦截器链。拦截器链的执行顺序，与其注册顺序一致。**



**SpringMVC工作流程**

