**Spring。主要包括IOC（控制反转）、AOP（面向切面编程）**

使用maven仓库管理依赖jar包。

导入lombok简化Javabean的书写，可以在类中直接定义属性，不用手动书写get、set、tostring方法。但是具体使用还需要在IDEA中install相关的插件。

<!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

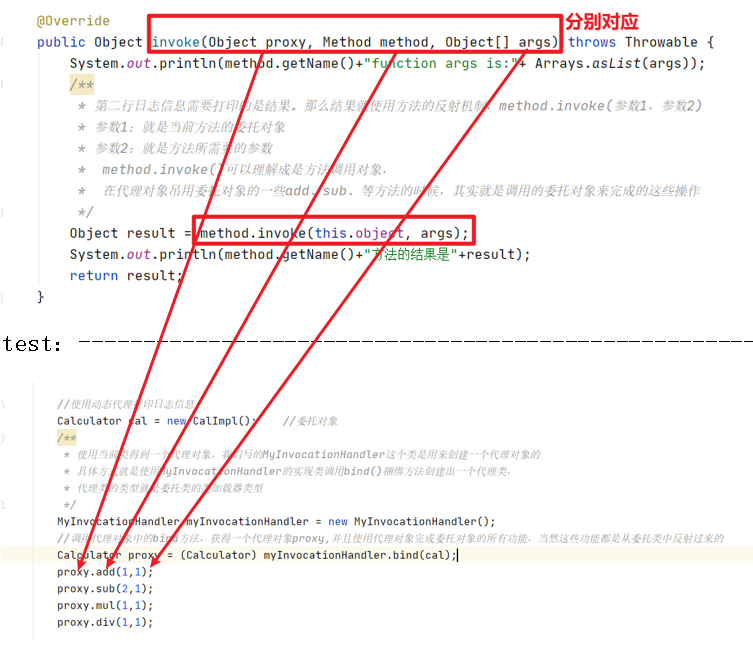
<version>1.18.12</version>

<scope>provided</scope>

</dependency>



**动态代理实现AOP的理解：**

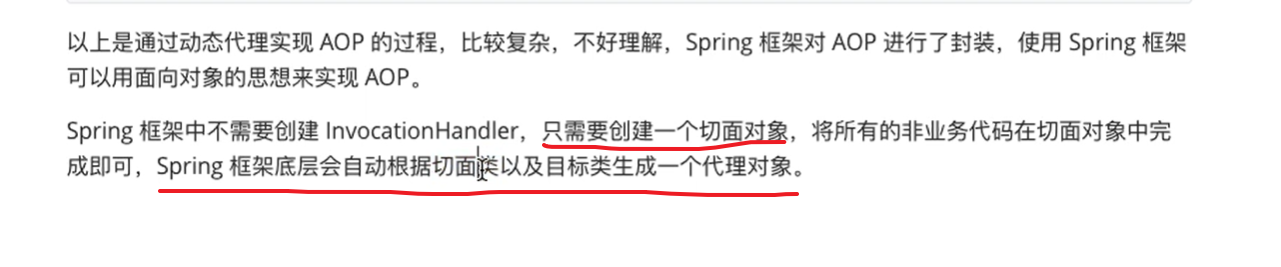


注意：在中间的**method.invoke(this.object,args)**的这个方法其实就是调用的委托类中的业务逻辑处理代码。

this.object **----**🡪当前的MInvocationHandler类中传入的委托类

args----🡪指的是方法所需要的参数

**Sprng体现AOP的书写配置：**



Spring框架中不需要创建InvocationHandler，只需要创建一个切面对象，将所有的非业务代码在切面对象中完成即可，Spring框架会自动根据切面类以及目标类生成一个代理对象

这里的切面类，在spring底层仍需要一个（bean）对象来进行对AOP的实现，所以原本是需要在xml配置文件中去配置当前这个类的具体对象的，但是现在可以使用注解的方式来进行bean对象的一个配置。



配置Spring-aop.xml文件



下面的这条aop:aspectj-autoproxy 代表的意思就是之前底层使用动态代理完成的aop操作。

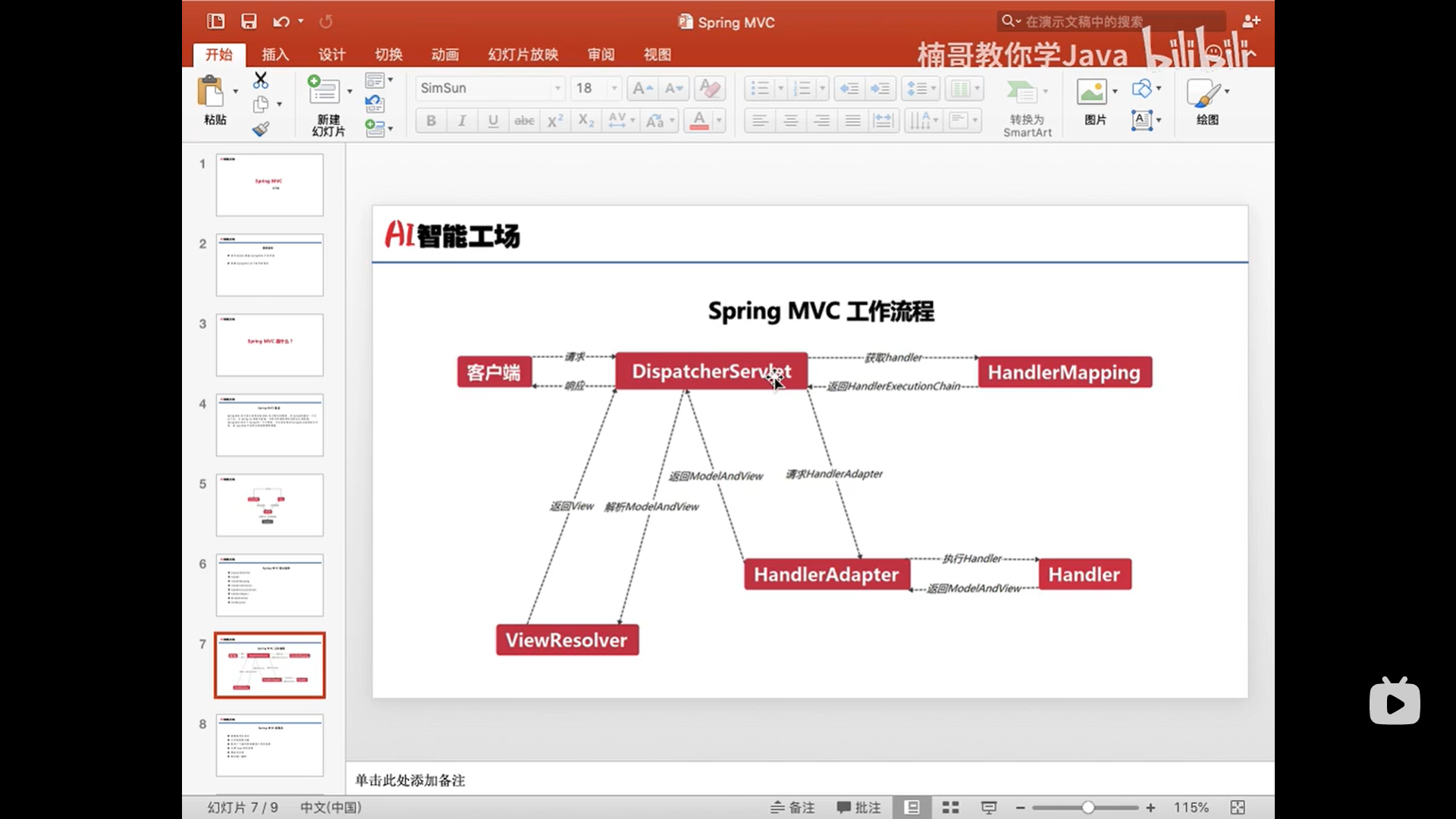
概念性的知识



**SpingMVC部分**



**具体的SpringMVC工作流程：**



工作流程文字描述：



**@RequestMapping()相关参数：**



**Spring MVC数据绑定**



**SpringMVC处理页面中文乱码的问题：**



在进行数据绑定的时候对于json数据的绑定，需要引入jQuery的.js文件，引入之后，因为之前配置的过滤器会自动过滤所有的请求，所以当服务器请求静态资源时，就会报错。

针对于这种情况，服务器访问静态资源的时候被拦截，可以在web.xml中配置：

会过滤掉所有访问以.js结尾的请求，使其使用默认的方法进行访问。即不会被拦截

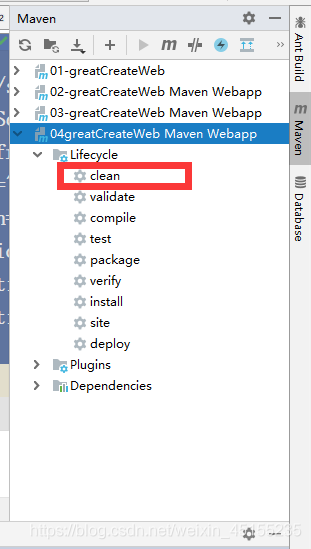
<servlet-mapping>

<servlet-name>default</servlet-name>

<url-partten>\*.js</url-partten>

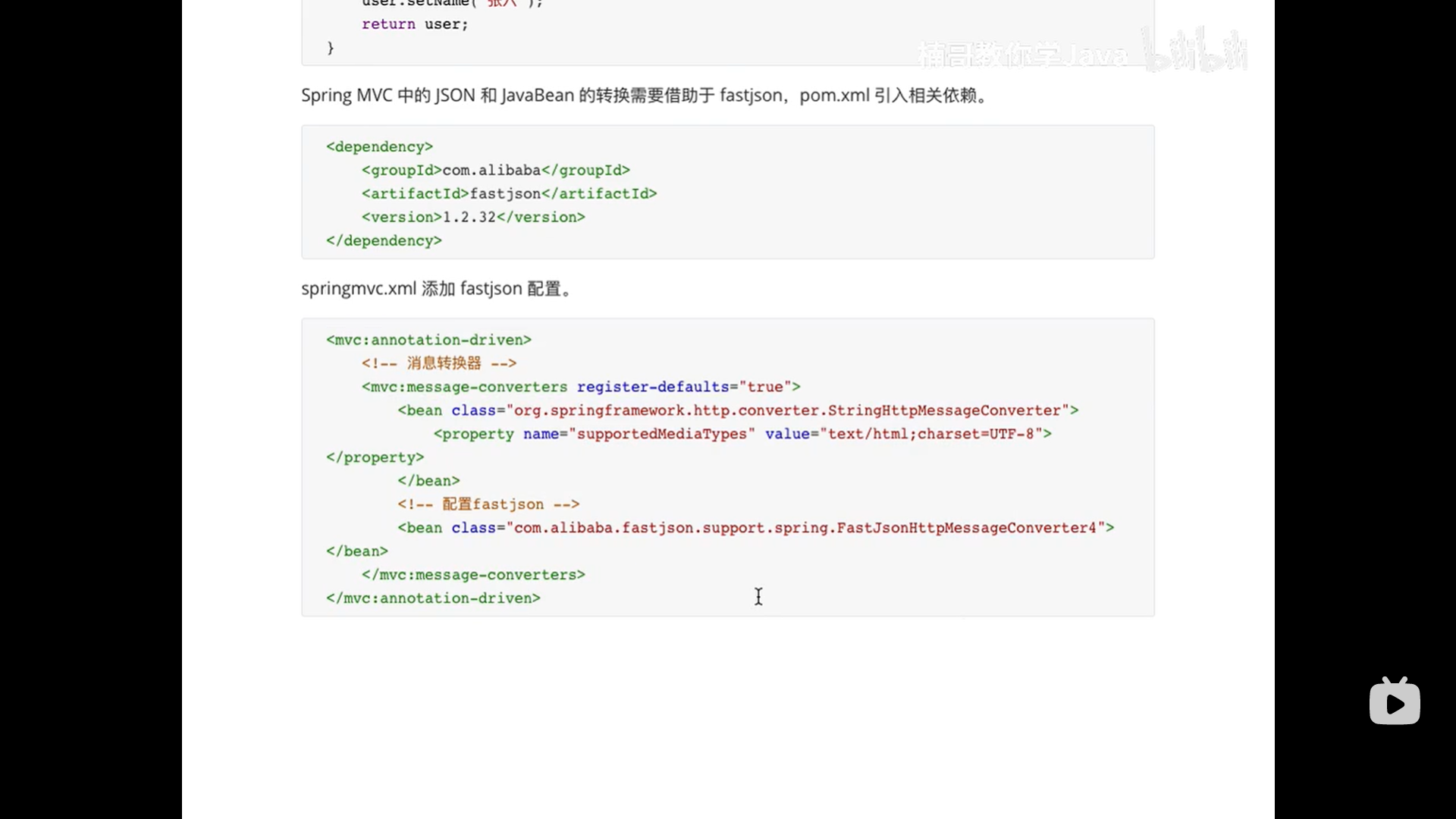
</servlet-mapping>

在以上操作完成之后，可能会遇到资源依旧无法访问的情况，这个时候，就需要进入自己的对应的maven声明周期栏，点击clean，清空target文件夹



然后重启Tomcat服务器，问题就解决了。

针对于json数据类型格式和Javabean类型之间的转换：



SpringMVC模型数据解析：

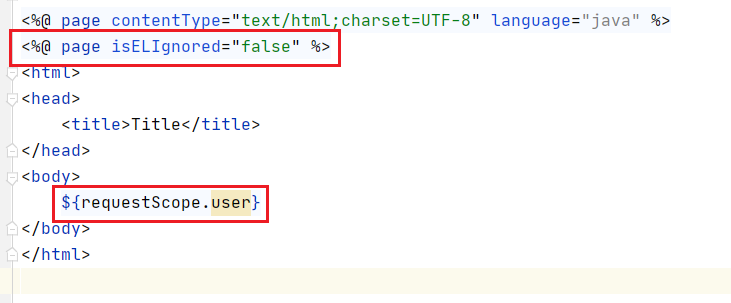


**将模型数据绑定到request对象。**

1、Map数据模型的使用

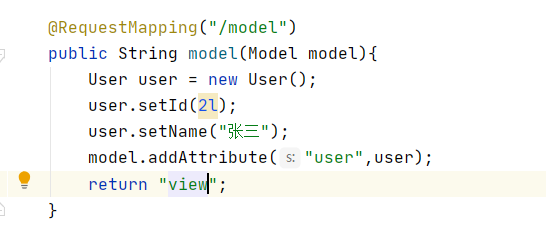


其中，我们先将数据添加到数据模型当中，然后再交给viewResolver进行绑定。上面的代码的参数列表中的map集合，是用来对外传输数据的，即将user对象的部分属性set以后，再添加到map集合中作为模型数据，然后返回的view.jsp即为视图，展示给用户。当中的map集合实际上也是以request的attribute的方式添加进request域中，所以：



在视图页面(view.jsp)中，直接使用EL表达式即可获取到对应的对象及其属性。但是需要在jsp页面上声明好<%@ page isELIgnored=”false”%>，不然EL表达式的读写会被忽略禁用。

2、Modle数据模型的使用：



与map类似，只不过Model是以attribute键值对的形式添加到模型数据，然后发送到request域中。

3、ModelAndView数据模型的使用：



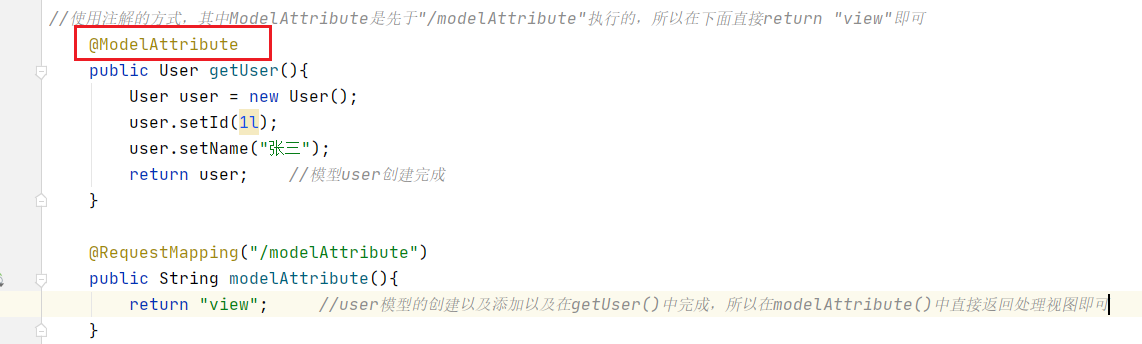
等等有八种不同的方式对模型数据进行添加。

4、原生的方式（直接将数据添加到request域中）：



上面的几种方式都是间接将模型数据添加到request域中，现在直接将user对象添加到request域中。

5、使用注解的方式：



在调用modelAttribute方法的时候，已经先调用了上面的getUser()，所以数据模型user在此之前已经创建完成，即在modelAttribute()中直接处理返回视图就行了。

将模型数据绑定到session对象

1、直接使用原生的Servlet API

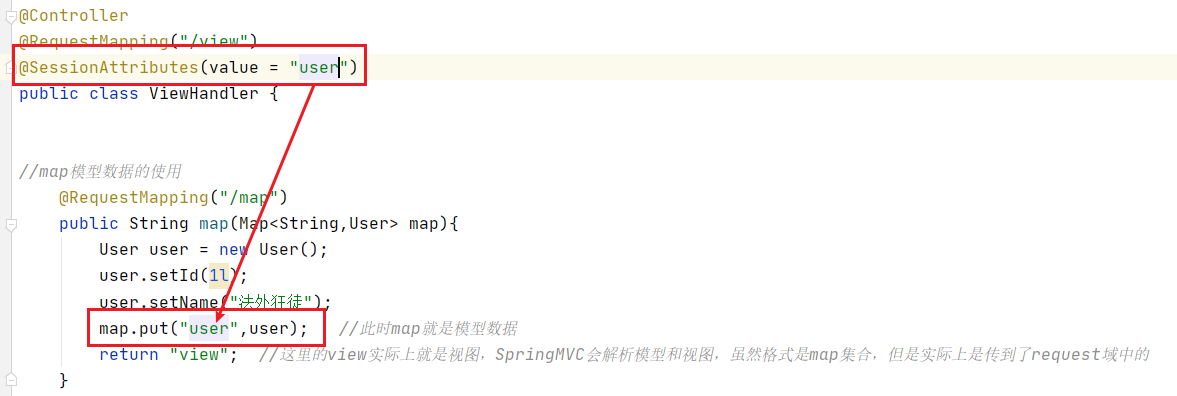


这两种方式都可以

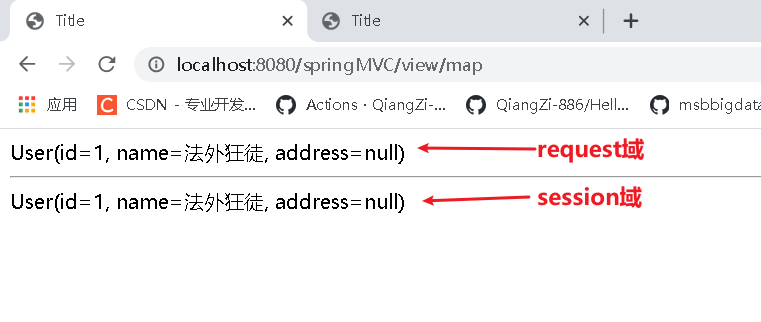
2、@SessionAttributes()使用注解的方式绑定到session对象

注意，注解是SessionAttributes，而且需要将注解添加到Handler类上。注解括号中的添加方式不止一种。

1. 直接使用value，对应下面向request域中添加的对象的键值：



注解中的value属性就是下面例如使用map添加模型数据，其中需要设置键值对，而对应的键，就是value属性所对应的”user”。即当map向request域中添加数据之后，上面也会通过注解在session域中添加相应的user对象。此时对map进行测试，观测session域中是否有user对象：

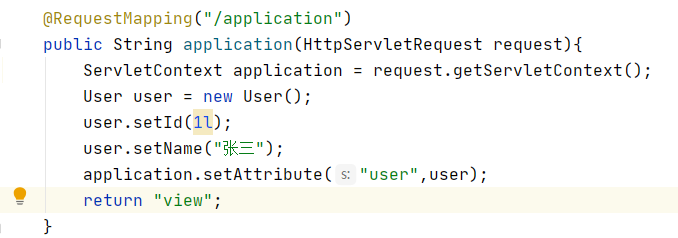


验证成功

也可以通过@SessionAttributes(types = User.class)通过类型比对进行添加。**但是**，如果这样写，那么当前Handler下的所有符合类型或者是键值的对象都会被添加到session域中，但其实我们使用的并没有那么多，所以还是原生的Servlet API使用的更多，而且资源也不会浪费。



**将模型数据绑定到application对象**



Jsp:

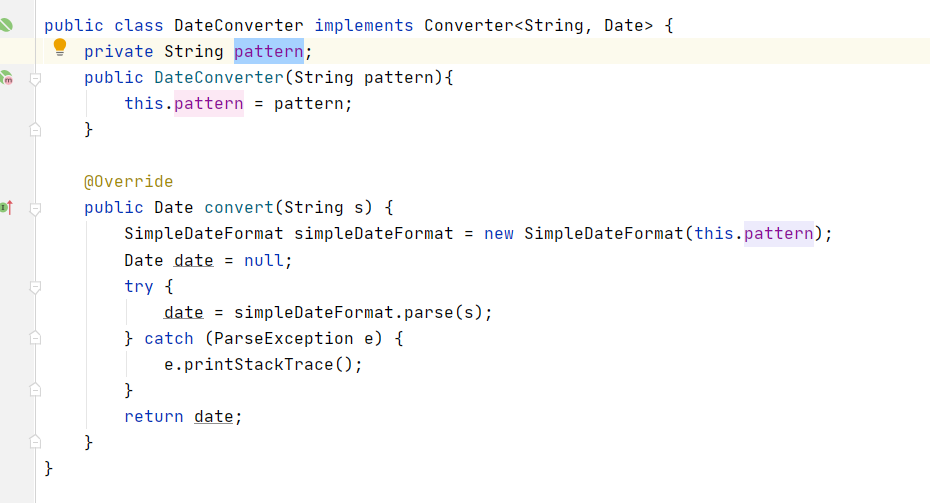


**SpringMVC自定义数据转换器：**

Spring框架虽然提供了很多数据转换器，但是对于一些实际应用的情况的时候，还是需要来自定义数据转换器。

定义String转date数据的转换器：

首先需要定义DateConverter转换器，实现Converter接口，在接口处实现泛型，<String,Date>，定义完成之后实现其中的方法就是传进的String参数，传出的Date数据了。如果不加泛型默认的都是Object类型。



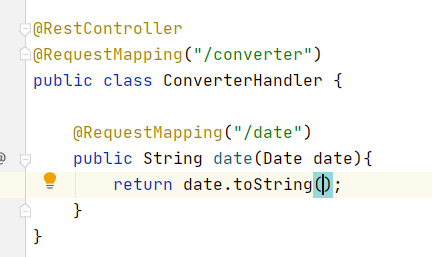
在其中需要先定义一个String类型的pattern，由于需要转换成的类型是date，所以需要规定类型的格式，pattern使用来定义date数据的格式的，比如：yyyy-MM-dd。然后再使用构造函数，在Spring的IOC容器中自动实例化。

做完操作之后，需要在spring-mvc.xml中，配置自定义转换器，主要就是使用构造器来进行对bean的配置，calss指向DateConverter类型type定义成java.lang.String，值value定义成”yyyy-MM-dd”。

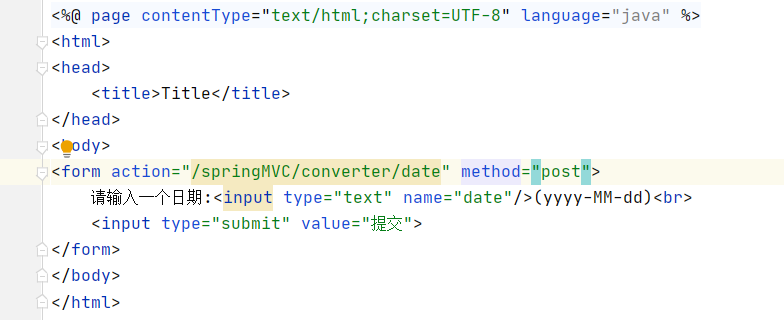


还有一点，在配置自定义转换器的bean之后还需要在mvc的driven中配置conversion-service。

最后在Controller层中新建ConverterHandler，参数接受date类型的数据，然后直接返回出date数据的toString，在页面上输出即可。



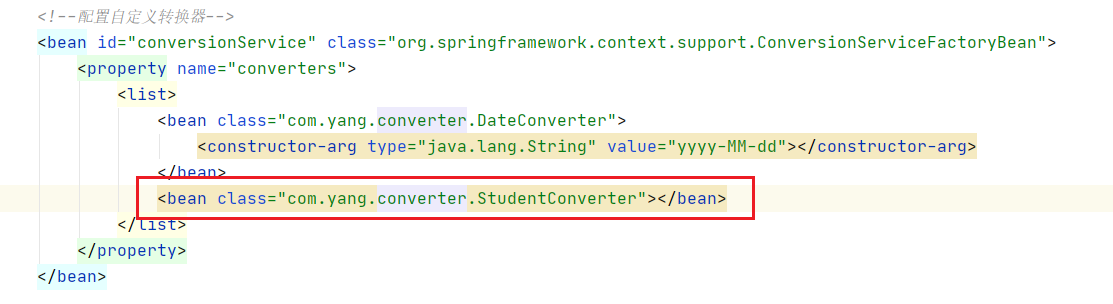
新建addDate.jsp页面，form表单指向上面Handler中的date方法。



整体流程大致为：首先用户访问jsp页面，然后填写String类型的信息，之后在spring-mvc.xml中的自定义转化器捕获信息，调用DateConverter对信息进行转换，返回一个date类型的数据给ConverterHandler中的date方法，再在页面上直接将date类型的数据打印输出。

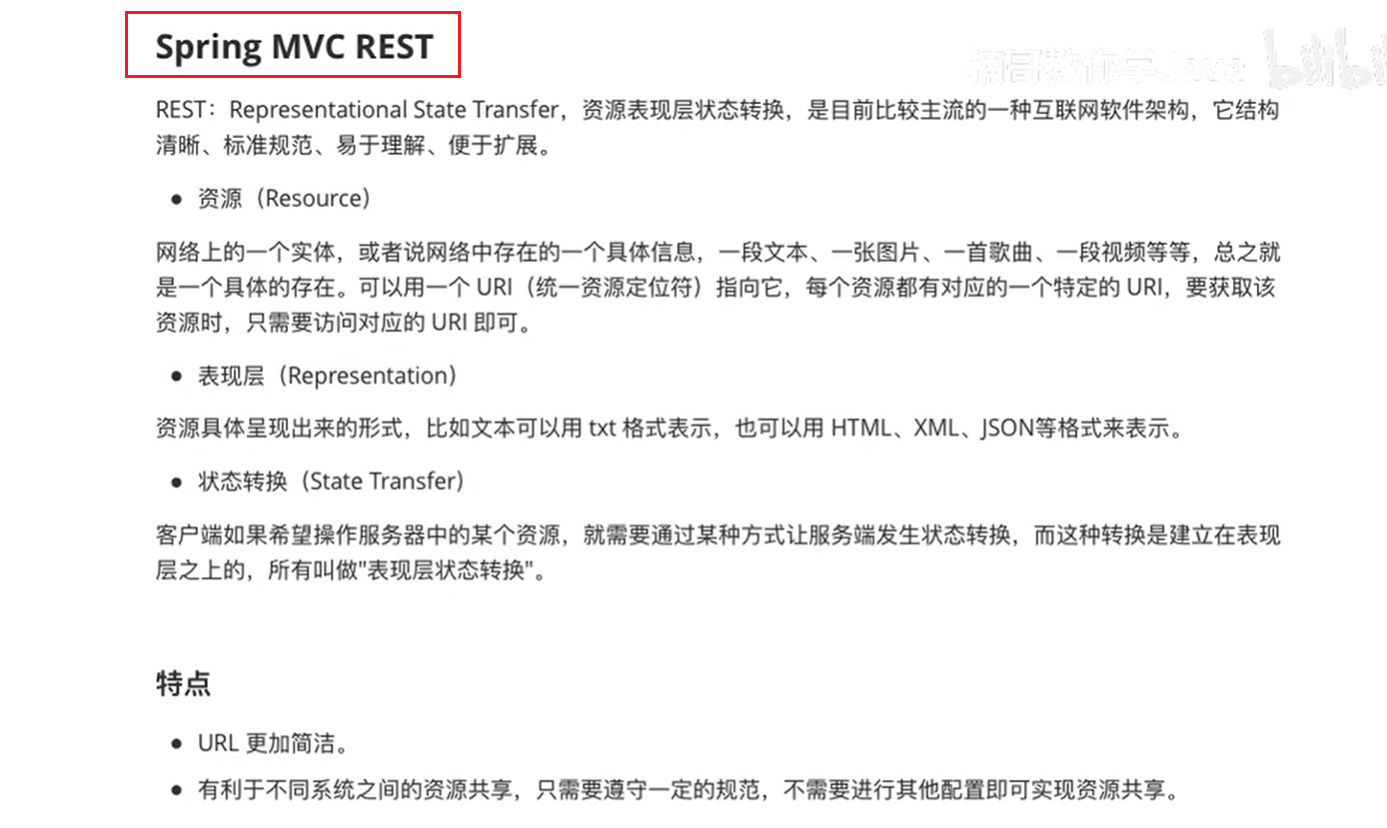
**实操简单自定义数据类型转换：将用户输入的String类型的数据转换成Student类型：（主要记录实现思想，具体代码IDE上有）**

首先创建Student的Javabean类，包括long型的id、String型的name、int型的age。上面使用Date注解，可以省略student中属性的set、get方法的书写。然后定义StudentConverter转换器，同样定义好泛型<String,Student>，Student类型不需要使用构造方法的其他格式，所以只用一个重写的convert方法即可，然后接受到的String用split方法（具体使用”-”隔开的规则对字符串分割），使用String[]数组接收。新建Student类，实例化对象，然后使student对象的set属性方法对属性赋值（注意long和int类型的数据使用Long、Integer包装类进行转换）。最后返回student对象。同样在spring-mvc.xml中配置自定义数据转换器。



书写addStudent.jsp，表单提交到ConverterHandler中的student中，自定义转换器获取到之后调用StudentConverter转换数据，最后提交给student()方法在客户端上打印出最终结果。

**Spring MVC REST风格：**



**SpringMVC 文件的上传和下载**

首先在pom依赖中添加commons-io和commons-fileupload两个依赖，然后在创建一个upload.jsp：



1、input的type属性设置为file，

2、form的method属性设置为post（get请求只能将文件名传给服务器）

3、form的enctype设置为multipart-form-data（如果不设置只能将文件名传给服务器）

如果想要预览展示效果，那么只需要在jsp页面后面跟上图片路径就可以进行效果预览。

文件下载略（详见IDEA或者md笔记）

**SpringMVC的表单标签库**（单选框、复选框、下拉框以及相应使用集合实现）

**SpringMVC数据校验:**

有两种方式：1、基于Validator接口。2、使用Annotation JSR-303标准进行校验

**Validator**：其中基于 Validator 接口的方式需要自定义 Validator 验证器，每一条数据的验证规则需要开发者手动完成，使用 Annotation JSR - 303 标准则不需要自定义验证器，通过注解的方式可以直接在实体类中添加每个属性的验证规则，这种方式更加方便，实际开发中推荐使用。

**Annotation**：使用 Annotation JSR - 303 标准进行验证，需要导入支持这种标准的依赖 jar 文件，这里我们使用 Hibernate Validator。

具体代码实现详见IDEA编译器或者SpringMVC笔记.md

注意！！！！！我所遇见的bug，踩坑！！

pom.xml依赖：



然后在pom依赖不断更改之后，maven项目一定一定一定要clean一下，才不会再报错！！！！！

