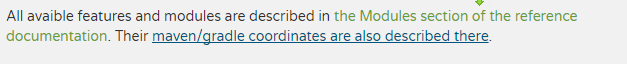
## 1.从官网下载Spring包

1.进入Spring官网

2.点击导航栏的PROJECT

3.点击

4.点击



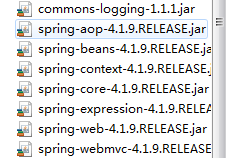
5.搜索ZIP FILES

6.点击

7.选择相应版本Spring包进行下载

## 2.搭建一个SpringMVC项目

1.创建了一个WEB项目

2.导入

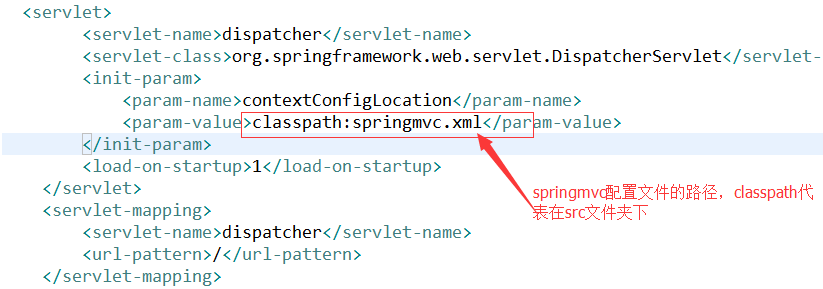
3.修改项目的web.xml配置文件

4.打开SPRING官方文档，搜索the web

5.点击

6.复制到web.xml中



7. 

8.在src下创建一个文件FILE，叫springmvc.xml

9.继续下翻官网文档复制



到springmvc配置文件中

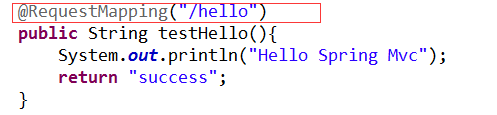
10.将自动扫描配置的路径修改为自己项目的包路径



11.新建包，并创建CLASS

12.将CLASS声明为控制器

13.创建处理方法并映射访问路径



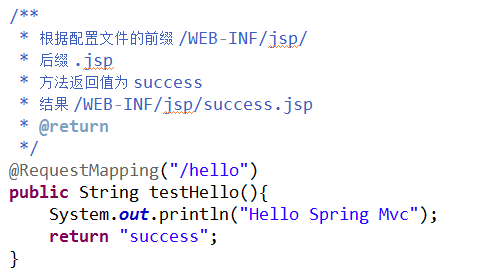
14.配置视图解析器，在官方文档中个搜索 *ViewResolver*



15.根据视图解析器的配置，在WEB-INF文件夹下创建名为JSP的文件夹，用来存放JSP文件

16.创建一个测试的JSP页面，success.jsp

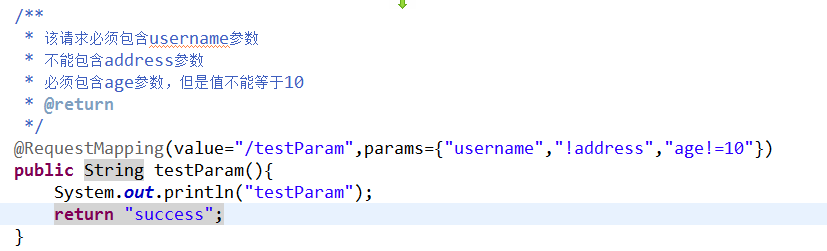
17.改造处理方法



## 3.请求方式设置



## 4.RequestMapping的params属性



## 5.RequestMapping-Ant风格映射

1. \* 星号 代表可以匹配所有字符（不限制长度）

@RequestMapping("/testAnt/\*")

在浏览器输入

/testAnt/12312312 /testAnt/asdasasd 都可以

2. \*\* 两个星号，代表可以匹配多层访问路径（不限制层数）

@RequestMapping("/testAnt/\*\*")

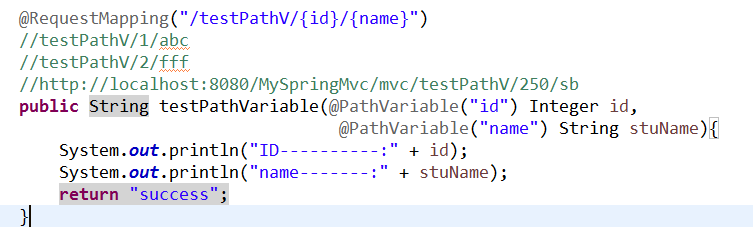
/testAnt/1/2/3/4/5/6/7/8/9/0/ 都可以

3. ? 一个问号代表匹配一个字符（可以有多个）

@RequestMapping("/testAnt/test??")

/testAnt/test12 /testAnt/testbb 都可以

## 6.PathVariable



## 7.RequestParam

获取请求参数

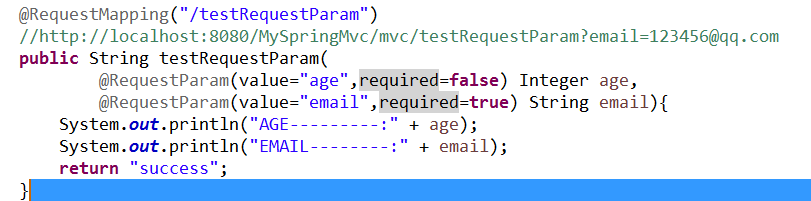
要写在方法参数括号内

两个属性

1.value 要获取的参数名字

2.required 是否必须传递

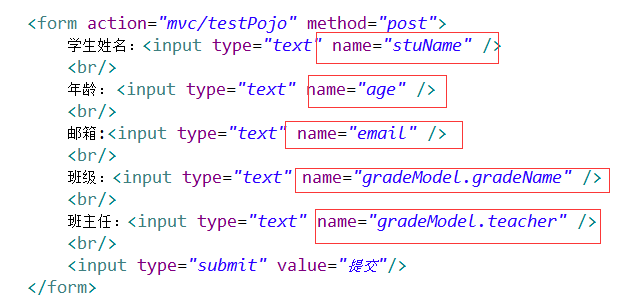
如果将其中的一个参数设置为了必须传递，那么在浏览器上如果不传递该参数，则会报400错误

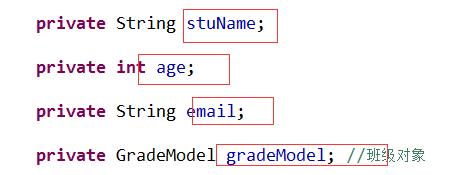


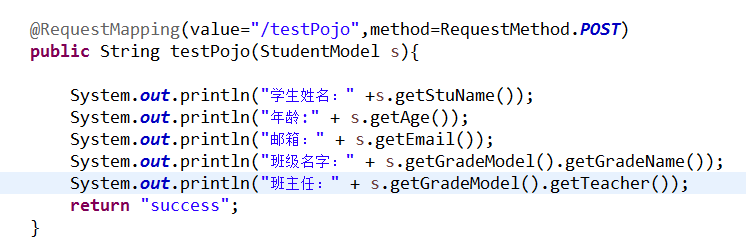
## 8.绑定POJO类型参数（实体类）

需要将页面元素的name属性与实体类对象属性保持一致

区分大小写，支持级联属性

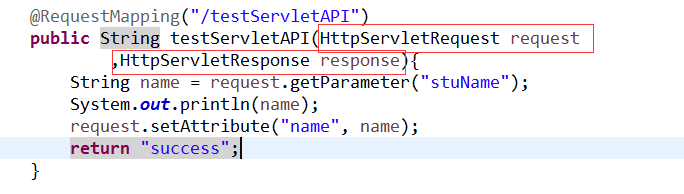






## 9.可接受Servlet API的参数

request,response,session,locale,reader,wirte,inputstream,outputstream, java.security.Principal



## 10.ModelAndView

如果方法的返回值为ModelAndView

那么可返回页面或数据模型

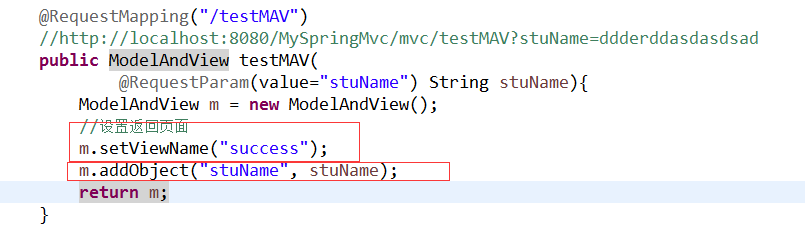
返回页面

setViewName(name); 与返回String类型的用法一致

返回数据模型

addObject(key,value); 与request.setAttribute(key,vlaue); 一致

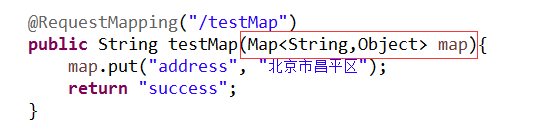
保存在了REQUEST域中



## 11.SpringMvc 隐藏的Map参数

处理方法中，可传递一个Map类型参数。

SpringMvc会自动将此Map参数中的值保存到Request中





## 12.SessionAttributes 注解

在类处定义

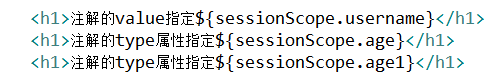


value 属性为 ，将key为username的对象保存到session中



types 属性为，将所有类型为Integer的对象保存到session中





## 13.HiddenHttpMethodFilter

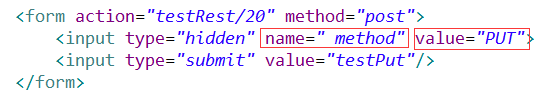
Spring过滤器，可将POST请求，转化为PUT或DELETE请求

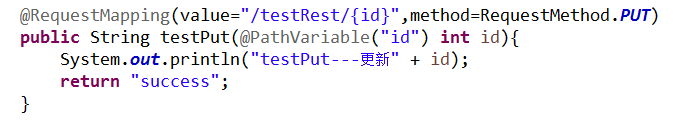
web.xml文件中 配置过滤器



将POST请求转换为PUT请求

在FORM表单中，添加name属性为\_method的隐藏域，value为转换的请求方式



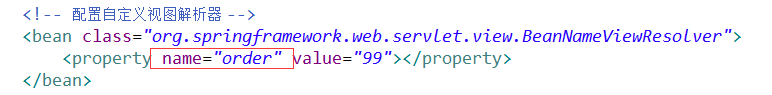


## 14.自定义视图

1.官方文档，搜索the web 点击 17

2.搜索beannameviewresolver

3.复制配置到springmvc.xml文件中



属性order为优先级顺序，value值越大，越优先使用该视图解析器

4.编写自定义视图

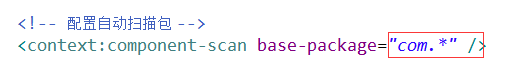


在类定义处添加注解 @Component 将其装入SpringIOC容器中

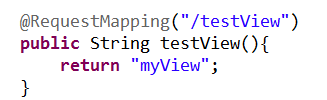
getContentType方法为视图类型

render方法为视图内容

5.修改springmvc配置文件，保证包扫描时可以扫描到自定义视图



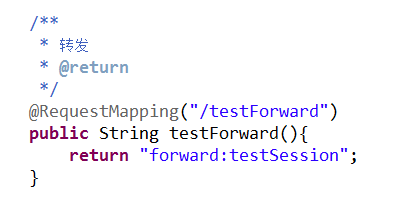
6.编写测试方法



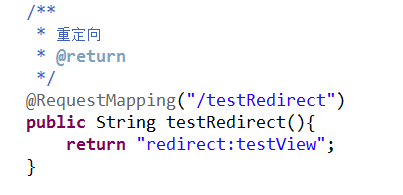
## 15.重定向和转发

在方法的返回值处加前缀

转发 forward:

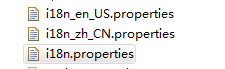


重定向 redirect:



## 16.国际化配置

在SRC下创建资源文件



i18n.properties 默认资源文件



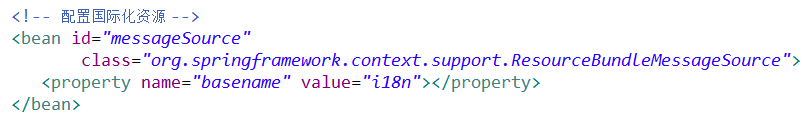
i18n\_zh\_CN.properties 中文资源文件



i18n\_en\_US.properties 英文资源文件



在springmvc.xml文件中配置



创建测试JSP页面

导入标签库



使用标签显示国际化信息

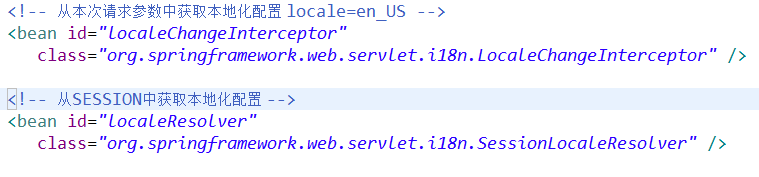


通过设置Internet选项—语言，切换语言 来实现国际化

## 17.通过请求切换国际化

首先在springmvc.xml配置文件中配置

SessionLocaleResolver和LocaleChangeInterceptor



配置国际化资源拦截器



编写测试JSP页面

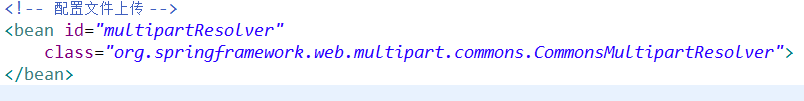


参数locale 为固定 不能写错

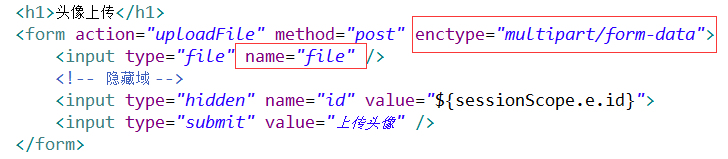
## 18.SpringMvc文件上传

导入JAR包

在sprngmvc.xml 中配置文件上传



创建上传页面



form表单的enctype=”multipart/form-data” 不能写错

要为file控件指定name属性

编写处理代码

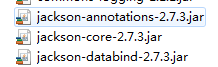


参数file为JSP页面指定的file控件name属性

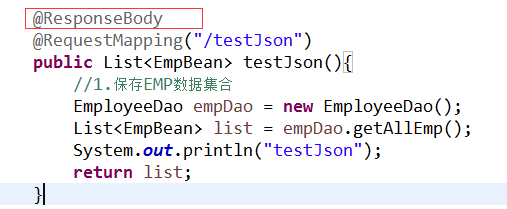
## 19.SpringMvc返回JSON数据类型

springmvc.xml文件中增加配置



导入JAR包

编写测试代码



返回JSON需要将处理器方法注解为@ResponseBody

比如:

需要将List格式化为JSON返回，那么就将List作为方法的返回类型

需要将一个JavaBean格式化为JSON返回，那么就将JavaBean作为方法的返回类型

## 20.SpringMvc拦截器

拦截器中有三个执行方法

preHandle(): 这个方法在业务处理器处理请求之前被调用，在该方法中对用户请求request进行处理。如果决定该拦截器对请求进行拦截处理后还需要调用其他拦截器，或者是业务处理器去进行处理，则返回true。如果不需要则返回false。

postHandle():这个方法在业务处理器处理完请求后，但是向客户端返回响应前被调用。

afterCompletion():这个方法在完全处理完请求后被调用(渲染视图之前)，可以在该方法中进行一些资源清理的操作。

多个拦截器根据配置文件的配置顺序进行执行

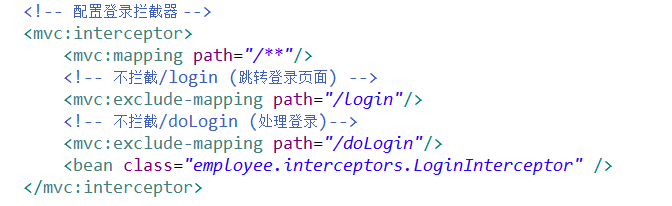
首先顺序执行所有拦截器的preHandle方法

然后倒序执行所有拦截器的postHandle方法和afterCompletion方法

如果第一个拦截器的preHandle方法返回false，那么后续方法都不执行，只执行第一个拦截器的afterCompletion方法

创建拦截器需要实现Spring的HandlerInterceptor 接口





如图，配置了拦截所有请求，但是不拦截/login请求，那么不拦截请求生效。

拦截器方法的内容：



## 21.SpringMvc运行流程

1.用户向服务器发送请求，被DispatcherServlet捕获

2.根据用户发送的请求,调用HandlerMapping获取相应的配置，

最后以Handler　execution　Chain（这三个单词是连在一起的）对象的形式返回

3.DispatcherServlet根据获得的Handler,选择一个合适的HandlerAdapter

4.提取Request中的数据模型，填充Handler入参，开始执行Handler

5.Handler执行完成后，DispatcherServlet返回一个ModelAndView对象

6.根据返回的ModelAndView，选择一个合适的ViewResolver

7.ViewResolver结合ModelAndView，来渲染视图

8.将渲染结果返回给客户端