## 使用@Controller配置控制器

1. Springweb-servlet.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"*  *http://www.springframework.org/schema/beans*  *http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/context*  *http://www.springframework.org/schema/context/spring-context.xsd*  *http://www.springframework.org/schema/mvc*  *http://www.springframework.org/schema/mvc/spring-mvc.xsd"*>  <!-- <bean name="/hello" class="controller.Welcome"></bean> -->  <!-- 定义在那个包中去扫描annotation -->  <context:component-scan base-package=*"controller"*></context:component-scan>  <!-- 开启spring的annotaion开关 -->  <mvc:annotation-driven></mvc:annotation-driven>  <bean id=*"viewResolver"*  class=*"org.springframework.web.servlet.view.UrlBasedViewResolver"*>  <property name=*"viewClass"* value=*"org.springframework.web.servlet.view.JstlView"*/>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"*/>  <property name=*"suffix"* value=*".jsp"*/>  </bean>  </beans> |

## 创建支持JSON的Controller

## 在Controller类使用@RestController

1. 在Controller里的方法中，参数前使用@RequestBody

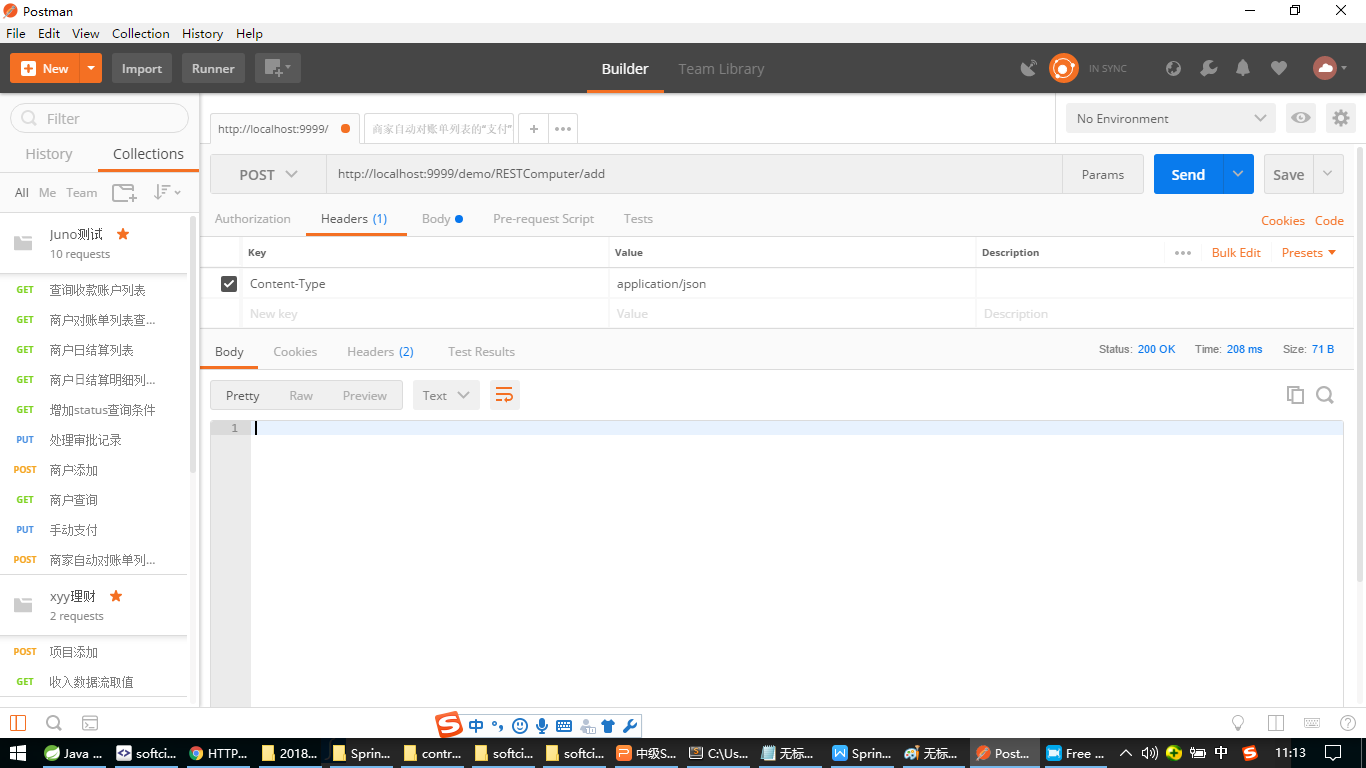
|  |
| --- |
| **package** controller;  **import** org.springframework.web.bind.annotation.RequestBody;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **package** controller;  **import** org.springframework.web.bind.annotation.RequestBody;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.bind.annotation.RestController;  **import** model.Computer;  @RestController  @RequestMapping(path="/RESTComputer")  **public** **class** RestFullComputerController {  @RequestMapping(path="/add", method=RequestMethod.***POST***)  **public** String addComputer(@RequestBody Computer computer){    System.***out***.println(computer);    **return** **null**;  }  } |

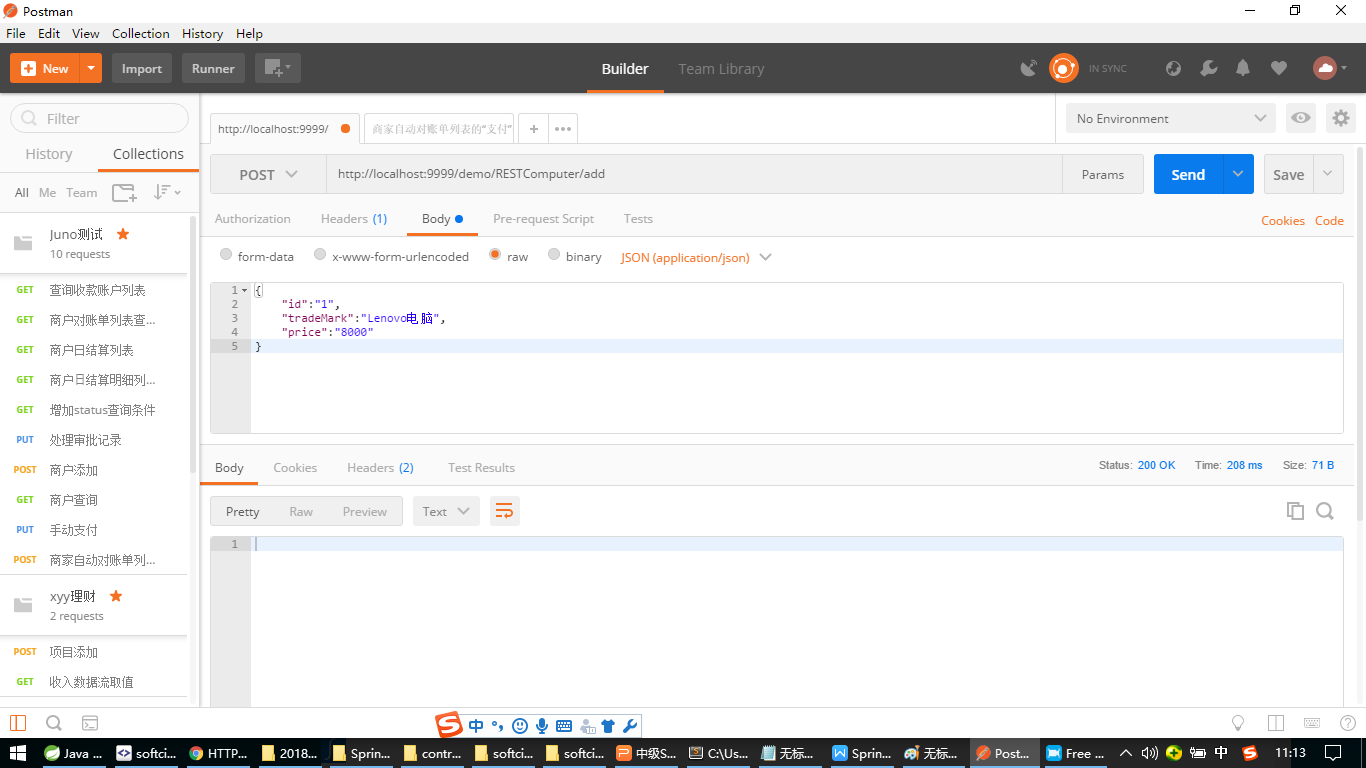
1. 引入jackson-databind的依赖包

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/com.fasterxml.jackson.core/jackson-databind -->  <dependency>  <groupId>com.fasterxml.jackson.core</groupId>  <artifactId>jackson-databind</artifactId>  <version>2.9.4</version>  </dependency> |

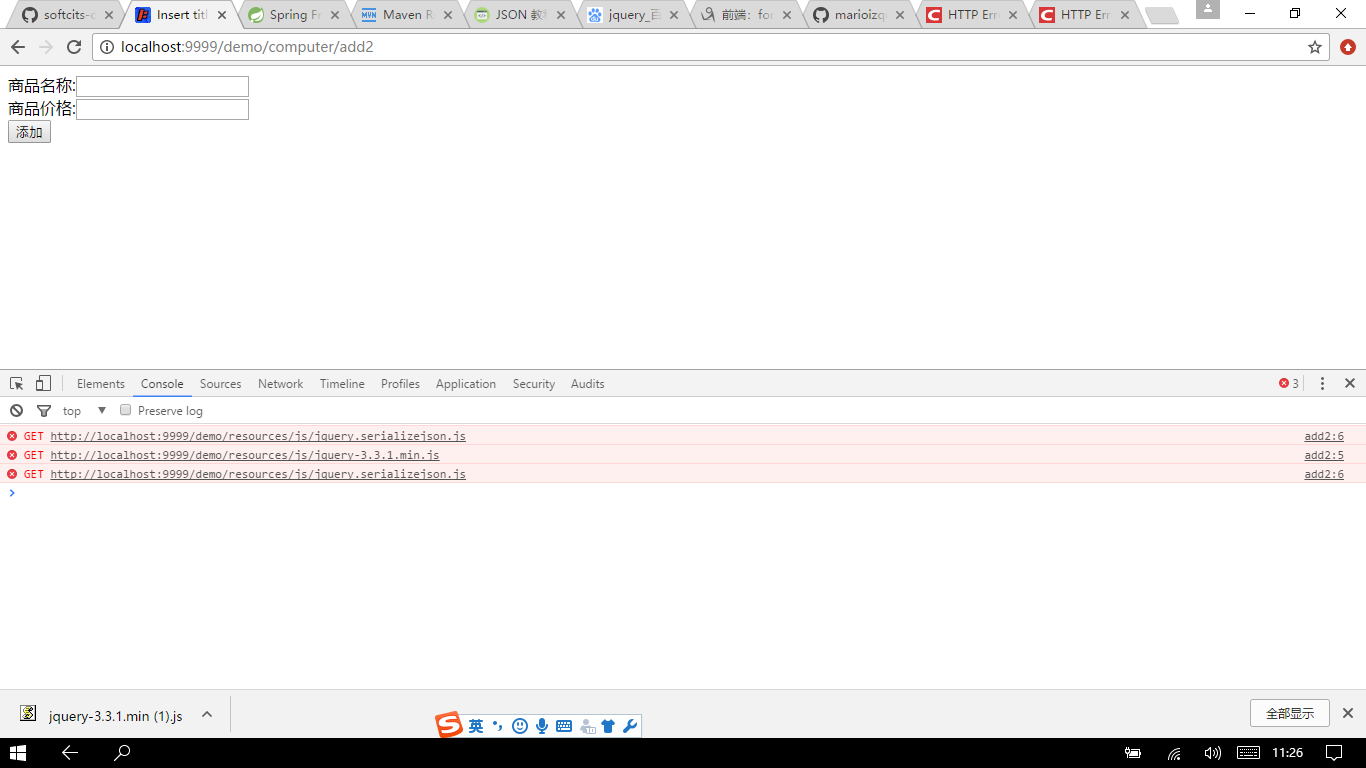
1. 测试

可以使用POSTMAN





如果遇到页面中引入的静态资源无法找到



这是因为http的请求被servelt拦截了，解决方案,在springmvc中添加静态资源映射配置

|  |
| --- |
| <!-- 不让SpringDispatcher拦截/resources下的静态资源 -->  <mvc:resources mapping=*"/resources/\*\*"* location=*"/resources/"*></mvc:resources> |

## 后台向前端返回JSON数据

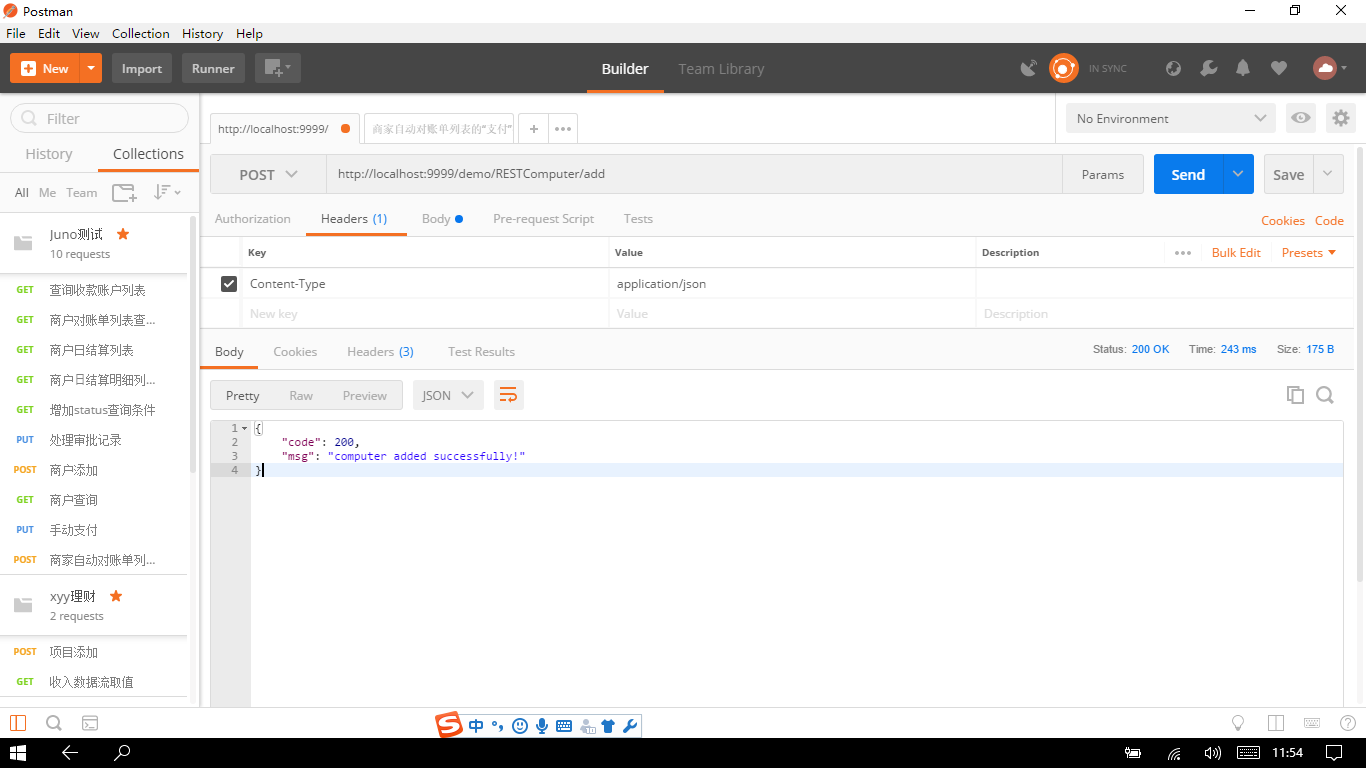
1. 可以创建一个ajax pojo如下

|  |
| --- |
| **package** model;  **public** **class** AjaxModel {    **private** Integer code;  **private** String msg;  **public** AjaxModel() {  **super**();  }  **public** AjaxModel(Integer code, String msg) {  **super**();  **this**.code = code;  **this**.msg = msg;  }  **public** Integer getCode() {  **return** code;  }  **public** **void** setCode(Integer code) {  **this**.code = code;  }  **public** String getMsg() {  **return** msg;  }  **public** **void** setMsg(String msg) {  **this**.msg = msg;  }    } |

1. 在Controller的方法中，要使用@ResponseBody

|  |
| --- |
| **package** controller;  **import** org.springframework.web.bind.annotation.RequestBody;  **import** org.springframework.web.bind.annotation.RequestMapping;  **import** org.springframework.web.bind.annotation.RequestMethod;  **import** org.springframework.web.bind.annotation.ResponseBody;  **import** org.springframework.web.bind.annotation.RestController;  **import** model.AjaxModel;  **import** model.Computer;  @RestController  @RequestMapping(path="/RESTComputer")  **public** **class** RestFullComputerController {  @RequestMapping(path="/add", method=RequestMethod.***POST***)  @ResponseBody  **public** AjaxModel addComputer(@RequestBody Computer computer){    System.***out***.println(computer);    /\*  \* 进行add computer操作  \* 。。。。。。。。。。。  \* \*/    AjaxModel am = **new** AjaxModel(200, "computer added successfully!");    **return** am;  }  } |

使用POSTMAN进行测试



# SpringMVC对文件上传的支持

1. 在SpringMVC(SpringWeb-servlet.xml)的配置中,添加如下,注册CommonsMultipartResolver

|  |
| --- |
| <bean id="multipartResolver"  class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  *<!-- one of the properties available; the maximum file size in bytes -->*  <property name="maxUploadSize" value="100000"/>  </bean> |

1. 创建文件上传的前台页面

|  |
| --- |
| <form method=*"post"* enctype=*"multipart/form-data"*>  商品名称:<input type=*"text"* name=*"tradeMark"* ><br/>  商品价格:<input type=*"text"* name=*"price"*><br/>  商品图片:<input type=*"file"* name=*"pic"*><br/>  <input type=*"submit"* value=*"提交"*>  </form> |

1. 添加fileUpload的依赖

|  |
| --- |
| <!-- https://mvnrepository.com/artifact/commons-io/commons-io -->  <dependency>  <groupId>commons-io</groupId>  <artifactId>commons-io</artifactId>  <version>2.6</version>  </dependency>  <!-- https://mvnrepository.com/artifact/commons-fileupload/commons-fileupload -->  <dependency>  <groupId>commons-fileupload</groupId>  <artifactId>commons-fileupload</artifactId>  <version>1.3.3</version>  </dependency> |

1. 编写Controller

|  |
| --- |
| @RequestMapping(path="/add", method = RequestMethod.***POST***)  //注意!!!如果是文件上传,一定不能使用@RequestBody来转换成JSON数据,而要使用@RequestParam  **public** String compUpload(@RequestParam String tradeMark, @RequestParam String price,  @RequestParam("pic") MultipartFile fileAttach, HttpServletRequest req) **throws** IOException{    String fileName = fileAttach.getOriginalFilename();    /\* System.out.println("tradeMark: " + tradeMark);  System.out.println("price: " + price);  System.out.println("attachment: " + fileName);\*/  //得到磁盘的物理路径  String realPath = req.getSession().getServletContext().getRealPath("/resources/pic");  //创建本地实体文件  //F:\git-repo\SpringWeb\src\main\webapp\resources\pic + "\" + "Logo.png"  File file = **new** File(realPath + "\\" + fileName);    //实现文件的上传拷贝  FileUtils.*copyInputStreamToFile*(fileAttach.getInputStream(), file);    **return** "redirect:/computer/list";  }  } |

## SpringMVC的表单验证

1.引入所需的依赖jar

|  |
| --- |
| <dependency>  <groupId>org.hibernate</groupId>  <artifactId>hibernate-validator</artifactId>  <version>5.1.0.Final</version>  </dependency> |

1. 在与前台表单进行绑定的javabean前添加@Validated

|  |
| --- |
| @ResponseBody  @RequestMapping(path="/add", method=RequestMethod.***POST***)  **public** AjaxModel addComputer(@RequestBody @Validated Computer computer){    System.***out***.println(computer.getTradeMark());  System.***out***.println(computer.getPrice());    /\*  \* 进行add computer操作  \* 。。。。。。。。。。。  \* \*/    AjaxModel am = **new** AjaxModel(200, "computer added successfully!");    **return** am;  } |

1. 具体的Annotation使用方法可以参考 JSR-303规范

<https://www.ibm.com/developerworks/cn/java/j-lo-jsr303/>

|  |
| --- |
| **package** model;  **import** javax.validation.constraints.Max;  **import** javax.validation.constraints.Min;  **import** org.hibernate.validator.constraints.NotBlank;  **public** **class** Computer {  **private** **int** id;  @NotBlank(message="商品名称不能为空")  **private** String tradeMark;  @Min(value=100,message="商品价格不能低于100元")  @Max(value=10000,message="商品价格不能高于10000元")  **private** **int** price;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getTradeMark() {  **return** tradeMark;  }  **public** **void** setTradeMark(String tradeMark) {  **this**.tradeMark = tradeMark;  }  **public** **int** getPrice() {  **return** price;  }  **public** **void** setPrice(**int** price) {  **this**.price = price;  }  **public** Computer() {  **super**();  }  **public** Computer(**int** id, String tradeMark, **int** price) {  **super**();  **this**.id = id;  **this**.tradeMark = tradeMark;  **this**.price = price;  }  @Override  **public** String toString() {  **return** "Computer [id=" + id + ", tradeMark=" + tradeMark + ", price=" + price + "]";  }    } |

1. 一定记得要让spring容器扫描bean里的annotation,在SpringWeb-servlet.xml中,包与包之间用空格分割

|  |
| --- |
| <!-- 定义在那个包中去扫描annotation -->  <context:component-scan base-package=*"controller model"*></context:component-scan> |

## SpringMVC的异常处理

1. 创建一个自定义异常

|  |
| --- |
| **package** MyException;  **public** **class** PCFormException **extends** RuntimeException {  /\*\*  \*  \*/  **private** **static** **final** **long** ***serialVersionUID*** = -4911898188563467119L;  **public** PCFormException() {  **super**();  // **TODO** Auto-generated constructor stub  }  **public** PCFormException(String message, Throwable cause, **boolean** enableSuppression, **boolean** writableStackTrace) {  **super**(message, cause, enableSuppression, writableStackTrace);  // **TODO** Auto-generated constructor stub  }  **public** PCFormException(String message, Throwable cause) {  **super**(message, cause);  // **TODO** Auto-generated constructor stub  }  **public** PCFormException(String message) {  **super**(message);  // **TODO** Auto-generated constructor stub  }  **public** PCFormException(Throwable cause) {  **super**(cause);  // **TODO** Auto-generated constructor stub  }  } |

1. 创建异常处理类

|  |
| --- |
| package MyException;  import org.springframework.web.bind.annotation.ControllerAdvice;  import org.springframework.web.bind.annotation.ExceptionHandler;  import org.springframework.web.bind.annotation.ResponseBody;  import model.AjaxModel;  @ControllerAdvice  public class MyFormExceptionHander {  @ExceptionHandler(PCFormException.class)  @ResponseBody  public AjaxModel pcFormExpHandler(PCFormException pcFormException){  System.out.println("捕获接到了PCFormException异常");  System.out.println("异常信息为:" + pcFormException.getMessage());    return new AjaxModel(500, pcFormException.getMessage());    }  } |

1. 在SpringWeb-servlet.xml配置中添加annotation扫描

|  |
| --- |
| <!-- 定义在那个包中去扫描annotation -->  <context:component-scan base-package=*"controller model MyException service"*></context:component-scan> |