# HelloWorld

|  |
| --- |
|  |

## web.xml

|  |
| --- |
| <!-- 配置 DispatcherServlet --> <servlet>  <servlet-name>dispatcherServlet</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- 配置 DispatcherServlet 的一个初始化参数: 配置 SpringMVC 配置文件的位置和名称  实际上也可以不通过 contextConfigLocation 来配置 SpringMVC 的配置文件, 而使用默认的.  默认的配置文件为: /WEB-INF/<servlet-name>-servlet.xml-->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc.xml</param-value>  </init-param>   <!-- spring在web应用在加载时就创建，而不是第一次请求时创建 -->  <load-on-startup>1</load-on-startup> </servlet>  <servlet-mapping>  <servlet-name>dispatcherServlet</servlet-name>  <url-pattern>/</url-pattern> </servlet-mapping> |

## springmvc.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:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  https://www.springframework.org/schema/context/spring-context.xsd">    <!-- 配置自定扫描的包 -->  <context:component-scan base-package="com.atguigu.springmvc"></context:component-scan>   <!-- 配置视图解析器: 如何把 handler 方法返回值解析为实际的物理视图 -->  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix" value="/WEB-INF/views/"></property>  <property name="suffix" value=".jsp"></property>  </bean> </beans> |

## index.jsp

|  |
| --- |
| <a href="helloworld">Test helloworld</a> |

## success.jsp

|  |
| --- |
| <h4>Sucess Page</h4> |

## HelloWorld

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.RequestMapping;  // 标志为控制器 @Controller public class HelloWorld {  /\*\*  \* 1. 使用 @RequestMapping 注解来映射请求的 URL  \* 2. 返回值会通过视图解析器解析为实际的物理视图, 对于 InternalResourceViewResolver 视图解析器, 会做如下的解析:  \* 通过 prefix + returnVal + 后缀 这样的方式得到实际的物理视图, 然会做转发操作  \*  \* /WEB-INF/views/success.jsp  \*  \* @return  \*/  @RequestMapping("/helloworld")  public String hello(){  System.***out***.println("hello world");  return "success";  } } |

## 或

## web.xml

|  |
| --- |
| *<?*xml version="1.0" encoding="UTF-8"*?>* <web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_4\_0.xsd"  version="4.0">    <!-- 配置 DispatcherServlet -->  <servlet>  <servlet-name>dispatcherServlet</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- 配置 DispatcherServlet 的一个初始化参数: 配置 SpringMVC 配置文件的位置和名称  实际上也可以不通过 contextConfigLocation 来配置 SpringMVC 的配置文件, 而使用默认的.  默认的配置文件为: /WEB-INF/<servlet-name>-servlet.xml  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:springmvc.xml</param-value>  </init-param>-->   <!-- spring在web应用在加载时就创建，而不是第一次请求时创建 -->  <load-on-startup>1</load-on-startup>  </servlet>   <servlet-mapping>  <servlet-name>dispatcherServlet</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

## dispatcherServlet-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:context="http://www.springframework.org/schema/context"  xsi:schemaLocation="http://www.springframework.org/schema/beans  http://www.springframework.org/schema/beans/spring-beans.xsd  http://www.springframework.org/schema/context  https://www.springframework.org/schema/context/spring-context.xsd">   <!-- 配置自定扫描的包 -->  <context:component-scan base-package="com.atguigu.springmvc"></context:component-scan>   <!-- 配置视图解析器: 如何把 handler 方法返回值解析为实际的物理视图 -->  <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  <property name="prefix" value="/WEB-INF/views/"></property>  <property name="suffix" value=".jsp"></property>  </bean> </beans> |

# RequestMapping\_修饰类

## index.jsp

|  |
| --- |
| <a href="springmvc/testRequestMapping">Test RequestMapping</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";   /\*\*  \* 1. @RequestMapping 除了修饰方法, 还可来修饰类  \* 2.  \* 1). 类定义处: 提供初步的请求映射信息。相对于 WEB 应用的根目录  \* 2). 方法处: 提供进一步的细分映射信息。 相对于类定义处的 URL。  \* 若类定义处未标注 @RequestMapping，则方法处标记的 URL相对于 WEB 应用的根目录  \*/  @RequestMapping("/testRequestMapping")  public String testRequestMapping() {  System.***out***.println("testRequestMapping");  return ***SUCCESS***;  } } |

# RequestMapping\_请求方式

## index.jsp

|  |
| --- |
| <form action="springmvc/testMethod" method="POST">  <input type="submit" value="submit"/> </form>  <br><br> <a href="springmvc/testMethod">Test Method</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 常用: 使用 method 属性来指定请求方式  \*/  @RequestMapping(value = "/testMethod", method = RequestMethod.***POST***)  public String testMethod() {  System.***out***.println("testMethod");  return ***SUCCESS***;  } } |

# RequestMapping\_请求参数&请求头

## index.jsp

|  |
| --- |
| <a href="springmvc/testParamsAndHeaders?username=atguigu&age=11">Test ParamsAndHeaders</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 了解: 可以使用 params 和 headers 来更加精确的映射请求.  \* params 和 headers 支持简单的表达式.  \*  \* @return  \* headers = { "Accept-Language=en-US,zh;q=0.8" }  \*/  @RequestMapping(value = "testParamsAndHeaders", params = { "username",  "age!=10" }, headers = { "Accept-Language=zh-CN,zh;q=0.8" })  public String testParamsAndHeaders() {  System.***out***.println("testParamsAndHeaders");  return ***SUCCESS***;  } } |

# RequestMapping\_Ant 路径

## index.jsp

|  |
| --- |
| <a href="springmvc/testAntPath/mnxyz/abc">Test AntPath</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  */\* Ant 格  资源地址支持 源地址支持 3 种 种 匹  ? 匹文件名中一个字 –  \* 匹文件名中任意字 –  \*\* \*\* 匹多 层径 –  @RequestMapping 还 支持 Ant 格 URL  /user/createUser: 匹 –  /user/aaa/createUser、/user/bbb/createUser URL  /user/\*createUser: 匹 –  /user/createUser、/user/aaa/bbb/createUser URL  /user/createUser??: 匹 –  /user/createUseraa、/user/createUserbb URL\*/* @RequestMapping("/testAntPath/\*/abc")  public String testAntPath() {  System.***out***.println("testAntPath");  return ***SUCCESS***;  } } |

# RequestMapping\_PathVariable注解

## index.jsp

|  |
| --- |
| <a href="springmvc/testPathVariable/1">Test PathVariable</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* @PathVariable 可以来映射 URL 中的占位符到目标方法的参数中.  \* @param id  \* @return  \* @PathVariable(value="id") value是默认值，可以不写  \*/  @RequestMapping("/testPathVariable/{id}")  public String testPathVariable(@PathVariable("id") Integer id) {  System.***out***.println("testPathVariable: " + id);  return ***SUCCESS***;  } } |

# RequestMapping\_HiddenHttpMethodFilter 过滤器

## web.xml

|  |
| --- |
| <!--  配置 org.springframework.web.filter.HiddenHttpMethodFilter:  可以把 POST 请求转为 DELETE 或 POST 请求-->  <filter>  <filter-name>HiddenHttpMethodFilter</filter-name>  <filter-class>  org.springframework.web.filter.HiddenHttpMethodFilter  </filter-class>  </filter>   <filter-mapping>  <filter-name>HiddenHttpMethodFilter</filter-name>  <!-- 过滤所有请求 -->  <url-pattern>/\*</url-pattern>  </filter-mapping> |

## index.jsp

|  |
| --- |
| <form action="springmvc/testRest/1" method="post">  <input type="hidden" name="\_method" value="PUT"/>  <input type="submit" value="TestRest PUT"/> </form> <br><br>  <form action="springmvc/testRest/1" method="post">  <input type="hidden" name="\_method" value="DELETE"/>  <input type="submit" value="TestRest DELETE"/> </form> <br><br>  <form action="springmvc/testRest" method="post">  <input type="submit" value="TestRest POST"/> </form> <br><br>  <a href="springmvc/testRest/1">Test Rest Get</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";    /\*\*  \* Rest 风格的 URL. 以 CRUD 为例:  \* 新增: /order POST  \* 修改: /order/1 PUT update?id=1  \* 获取: /order/1 GET get?id=1  \* 删除: /order/1 DELETE delete?id=1  \*  \* 如何发送 PUT 请求和 DELETE 请求呢 ?  \* 1. 需要配置 HiddenHttpMethodFilter  \* 2. 需要发送 POST 请求  \* 3. 需要在发送 POST 请求时携带一个 name="\_method" 的隐藏域, 值为 DELETE 或 PUT  \*  \* 在 SpringMVC 的目标方法中如何得到 id 呢? 使用 @PathVariable 注解  \*  \*/  @RequestMapping(value = "/testRest/{id}", method = RequestMethod.***PUT***)  public String testRestPut(@PathVariable Integer id) {  System.***out***.println("testRest Put: " + id);  return ***SUCCESS***;  }   @RequestMapping(value = "/testRest/{id}", method = RequestMethod.***DELETE***)  public String testRestDelete(@PathVariable Integer id) {  System.***out***.println("testRest Delete: " + id);  return ***SUCCESS***;  }   @RequestMapping(value = "/testRest", method = RequestMethod.***POST***)  public String testRest() {  System.***out***.println("testRest POST");  return ***SUCCESS***;  }   @RequestMapping(value = "/testRest/{id}", method = RequestMethod.***GET***)  public String testRest(@PathVariable Integer id) {  System.***out***.println("testRest GET: " + id);  return ***SUCCESS***;  } } |

# RequestParam 注解

## index.jsp

|  |
| --- |
| <a href="springmvc/testRequestParam?username=atguigu&age=11">Test RequestParam</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* @RequestParam 来映射请求参数.  \* value 值即请求参数的参数名  \* required 该参数是否必须. 默认为 true  \* defaultValue 请求参数的默认值  \*/  @RequestMapping(value = "/testRequestParam")  public String testRequestParam(  @RequestParam(value = "username") String un,  @RequestParam(value = "age", required = false, defaultValue = "0") int age) {  System.***out***.println("testRequestParam, username: " + un + ", age: "  + age);  return ***SUCCESS***;  }  } |

# RequestHeader 注解

## index.jsp

|  |
| --- |
| <a href="springmvc/testRequestHeader">Test RequestHeader</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 了解: 映射请求头信息  \* 用法同 @RequestParam 获取请求头信息  \*/  @RequestMapping("/testRequestHeader")  public String testRequestHeader(  @RequestHeader(value = "Accept-Language") String al) {  System.***out***.println("testRequestHeader, Accept-Language: " + al);  return ***SUCCESS***;  }  } |

# CookieValue 注解

## index.jsp

|  |
| --- |
| <a href="springmvc/testCookieValue">Test CookieValue</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 了解:  \* @CookieValue: 映射一个 Cookie 值.  \* 属性同 @RequestParammodelAttribute  \*/  @RequestMapping("/testCookieValue")  public String testCookieValue(@CookieValue("JSESSIONID") String sessionId) {  System.***out***.println("testCookieValue: sessionId: " + sessionId);  return ***SUCCESS***;  } } |

# 使用POJO作为参数

## User

|  |
| --- |
| package com.atguigu.springmvc.entities;  public class User {  private Integer id;   private String username;  private String password;   private String email;  private int age;   private Address address;   public Integer getId() {  return id;  }   public void setId(Integer id) {  this.id = id;  }   public String getUsername() {  return username;  }   public void setUsername(String username) {  this.username = username;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  }   public String getEmail() {  return email;  }   public void setEmail(String email) {  this.email = email;  }   public int getAge() {  return age;  }   public void setAge(int age) {  this.age = age;  }   public Address getAddress() {  return address;  }   public void setAddress(Address address) {  this.address = address;  }   @Override  public String toString() {  return "User{" +  "username='" + username + '\'' +  ", password='" + password + '\'' +  ", email='" + email + '\'' +  ", age=" + age +  ", address=" + address +  '}';  } } |

## Address

|  |
| --- |
| package com.atguigu.springmvc.entities;  public class Address {   private String province;  private String city;   public String getProvince() {  return province;  }   public void setProvince(String province) {  this.province = province;  }   public String getCity() {  return city;  }   public void setCity(String city) {  this.city = city;  }   @Override  public String toString() {  return "Address [province=" + province + ", city=" + city + "]";  } } |

## index.jsp

|  |
| --- |
| <form action="springmvc/testPojo" method="post">  username: <input type="text" name="username"/>  <br>  password: <input type="password" name="password"/>  <br>  email: <input type="text" name="email"/>  <br>  age: <input type="text" name="age"/>  <br>  city: <input type="text" name="address.city"/>  <br>  province: <input type="text" name="address.province"/>  <br>  <input type="submit" value="Submit"/> </form> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* Spring MVC 会按请求参数名和 POJO 属性名进行自动匹配，  \* 自动为该对象填充属性值。支持级联属性。  \* 如：dept.deptId、dept.address.tel 等  \*/  @RequestMapping("/testPojo")  public String testPojo(User user) {  System.***out***.println("testPojo: " + user);  return ***SUCCESS***;  } } |

# 使用Servlet原生API作为参数

## index.jsp

|  |
| --- |
| <a href="springmvc/testServletAPI">Test ServletAPI</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";    /\*\*  \* 可以使用 Serlvet 原生的 API 作为目标方法的参数 具体支持以下类型  \*  \* HttpServletRequest  \* HttpServletResponse  \* HttpSession  \* java.security.Principal  \* Locale InputStream  \* OutputStream  \* Reader  \* Writer  \* @throws IOException  \*/  @RequestMapping("/testServletAPI")  public void testServletAPI(HttpServletRequest request,  HttpServletResponse response, Writer out) throws IOException {  System.***out***.println("testServletAPI, " + request + ", " + response);  out.write("hello springmvc"); // return SUCCESS;  } } |

# 处理模型数据之ModelAndView

## index.jsp

|  |
| --- |
| <a href="springmvc/testModelAndView">Test ModelAndView</a> |

## success.jsp

|  |
| --- |
| time: ${requestScope.time} |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 目标方法的返回值可以是 ModelAndView 类型。  \* 其中可以包含视图和模型信息  \* SpringMVC 会把 ModelAndView 的 model 中数据放入到 request 域对象中.  \* @return  \*/  @RequestMapping("/testModelAndView")  public ModelAndView testModelAndView(){  String viewName = ***SUCCESS***;  ModelAndView modelAndView = new ModelAndView(viewName);   //添加模型数据到 ModelAndView 中.  modelAndView.addObject("time", new Date());   return modelAndView;  } } |

# 处理模型数据之Map

## index.jsp

|  |
| --- |
| <a href="springmvc/testMap">Test Map</a> |

## success.jsp

|  |
| --- |
| names: ${requestScope.names} |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 目标方法可以添加 Map 类型(实际上也可以是 Model 类型或 ModelMap 类型)的参数.  \* @param map  \* @return  \*/  @RequestMapping("/testMap")  public String testMap(Map<String, Object> map){  System.***out***.println(map.getClass().getName());  map.put("names", Arrays.asList("Tom", "Jerry", "Mike"));  return ***SUCCESS***;  } } |

# 处理模型数据之 SessionAttributes 注解

## User

|  |
| --- |
| package com.atguigu.springmvc.entities;   public class User {  private Integer id;   private String username;  private String password;   private String email;  private int age;   public Integer getId() {  return id;  }   public void setId(Integer id) {  this.id = id;  }   public String getUsername() {  return username;  }   public void setUsername(String username) {  this.username = username;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  }   public String getEmail() {  return email;  }   public void setEmail(String email) {  this.email = email;  }   public int getAge() {  return age;  }   public void setAge(int age) {  this.age = age;  }   @Override  public String toString() {  return "User{" +  "username='" + username + '\'' +  ", password='" + password + '\'' +  ", email='" + email + '\'' +  ", age=" + age +  ", address=" + address +  '}';  }   public User(String username, String password, String email, int age) {  this.username = username;  this.password = password;  this.email = email;  this.age = age;  }  public User() { }  } |

## index.jsp

|  |
| --- |
| <a href="springmvc/testSessionAttributes">Test SessionAttributes</a> |

## success.jsp

|  |
| --- |
| request user: ${requestScope.user } <br><br>  session user: ${sessionScope.user } <br><br>  request school: ${requestScope.school } <br><br>  session school: ${sessionScope.school } <br><br> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @SessionAttributes(value={"user"}, types={String.class}) @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* @SessionAttributes 除了可以通过属性名指定需要放到会话中的属性外(实际上使用的是 value 属性值),  \* 还可以通过模型属性的对象类型指定哪些模型属性需要放到会话中(实际上使用的是 types 属性值)  \*  \* 注意: 该注解只能放在类的上面. 而不能修饰放方法.  \*/  @RequestMapping("/testSessionAttributes")  public String testSessionAttributes(Map<String, Object> map){  User user = new User("Tom", "123456", "tom@atguigu.com", 15);  map.put("user", user);  map.put("school", "atguigu");  return ***SUCCESS***;  } } |

# ModelAttribute注解之使用场景

## User

|  |
| --- |
| package com.atguigu.springmvc.entities;  public class User {  private Integer id;   private String username;  private String password;   private String email;  private int age;   public Integer getId() {  return id;  }   public void setId(Integer id) {  this.id = id;  }   public String getUsername() {  return username;  }   public void setUsername(String username) {  this.username = username;  }   public String getPassword() {  return password;  }   public void setPassword(String password) {  this.password = password;  }   public String getEmail() {  return email;  }   public void setEmail(String email) {  this.email = email;  }   public int getAge() {  return age;  }   public void setAge(int age) {  this.age = age;  }   @Override  public String toString() {  return "User [id=" + id + ", username=" + username + ", password="  + password + ", email=" + email + ", age=" + age + "]";  }  public User(Integer id, String username, String password, String email, int age) {  this.id = id;  this.username = username;  this.password = password;  this.email = email;  this.age = age;  }   public User() {  } } |

## index.jsp

|  |
| --- |
| <!--  模拟修改操作  1. 原始数据为: 1, Tom, 123456,tom@atguigu.com,12  2. 密码不能被修改.  3. 表单回显, 模拟操作直接在表单填写对应的属性值 --> <form action="springmvc/testModelAttribute" method="Post">  <input type="hidden" name="id" value="1"/>  username: <input type="text" name="username" value="Tom"/>  <br>  email: <input type="text" name="email" value="tom@atguigu.com"/>  <br>  age: <input type="text" name="age" value="12"/>  <br>  <input type="submit" value="Submit"/> </form> |

## success.jsp

|  |
| --- |
| abc user: ${requestScope.abc } <br><br>  mnxyz user: ${requestScope.mnxyz } <br><br> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @SessionAttributes(value={"user"}, types={String.class}) @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  /\*\*  \* 1. 有 @ModelAttribute 标记的方法, 会在每个目标方法执行之前被 SpringMVC 调用!  \* 2. @ModelAttribute 注解也可以来修饰目标方法 POJO 类型的入参, 其 value 属性值有如下的作用:  \* 1). SpringMVC 会使用 value 属性值在 implicitModel 中查找对应的对象, 若存在则会直接传入到目标方法的入参中.  \* 2). SpringMVC 会以 value 为 key, POJO 类型的对象为 value, 存入到 request 中.  \*/  @ModelAttribute  public void getUser(@RequestParam(value="id",required=false) Integer id,  Map<String, Object> map) {  System.***out***.println("modelAttribute method");  if(id != null){  // 模拟从数据库中获取对象  User user = new User(1, "Tom", "123456", "tom@atguigu.com", 12);  System.***out***.println("从数据库中获取一个对象: " + user);  map.put("user", user);  }  }   /\*\*  \* 运行流程:  \* 1. 执行 @ModelAttribute 注解修饰的方法: 从数据库中取出对象, 把对象放入到了 Map 中. 键为: user  \* 2. SpringMVC 从 Map 中取出 User 对象, 并把表单的请求参数赋给该 User 对象的对应属性.  \* 3. SpringMVC 把上述对象传入目标方法的参数.  \*  \* 注意: 在 @ModelAttribute 修饰的方法中, 放入到 Map 时的键需要和目标方法入参类型的第一个字母小写的字符串一致!  \*  \* SpringMVC 确定目标方法 POJO 类型入参的过程  \* 1. 确定一个 key:  \* 1). 若目标方法的 POJO 类型的参数木有使用 @ModelAttribute 作为修饰, 则 key 为 POJO 类名第一个字母的小写  \* 2). 若使用了 @ModelAttribute 来修饰, 则 key 为 @ModelAttribute 注解的 value 属性值.  \* 2. 在 implicitModel 中查找 key 对应的对象, 若存在, 则作为入参传入  \* 1). 若在 @ModelAttribute 标记的方法中在 Map 中保存过, 且 key 和 1 确定的 key 一致, 则会获取到.  \* 3. 若 implicitModel 中不存在 key 对应的对象, 则检查当前的 Handler 是否使用 @SessionAttributes 注解修饰,  \* 若使用了该注解, 且 @SessionAttributes 注解的 value 属性值中包含了 key, 则会从 HttpSession 中来获取 key 所  \* 对应的 value 值, 若存在则直接传入到目标方法的入参中. 若不存在则将抛出异常.  \* 4. 若 Handler 没有标识 @SessionAttributes 注解或 @SessionAttributes 注解的 value 值中不包含 key, 则  \* 会通过反射来创建 POJO 类型的参数, 传入为目标方法的参数  \* 5. SpringMVC 会把 key 和 POJO 类型的对象保存到 implicitModel 中, 进而会保存到 request 中.  \*  \* 源代码分析的流程  \* 1. 调用 @ModelAttribute 注解修饰的方法. 实际上把 @ModelAttribute 方法中 Map 中的数据放在了 implicitModel 中.  \* 2. 解析请求处理器的目标参数, 实际上该目标参数来自于 WebDataBinder 对象的 target 属性  \* 1). 创建 WebDataBinder 对象:  \* ①. 确定 objectName 属性: 若传入的 attrName 属性值为 "", 则 objectName 为类名第一个字母小写.  \* \*注意: attrName. 若目标方法的 POJO 属性使用了 @ModelAttribute 来修饰, 则 attrName 值即为 @ModelAttribute  \* 的 value 属性值  \*  \* ②. 确定 target 属性:  \* > 在 implicitModel 中查找 attrName 对应的属性值. 若存在, ok  \* > \*若不存在: 则验证当前 Handler 是否使用了 @SessionAttributes 进行修饰, 若使用了, 则尝试从 Session 中  \* 获取 attrName 所对应的属性值. 若 session 中没有对应的属性值, 则抛出了异常.  \* > 若 Handler 没有使用 @SessionAttributes 进行修饰, 或 @SessionAttributes 中没有使用 value 值指定的 key  \* 和 attrName 相匹配, 则通过反射创建了 POJO 对象  \*  \* 2). SpringMVC 把表单的请求参数赋给了 WebDataBinder 的 target 对应的属性.  \* 3). \*SpringMVC 会把 WebDataBinder 的 attrName 和 target 给到 implicitModel.  \* 近而传到 request 域对象中.  \* 4). 把 WebDataBinder 的 target 作为参数传递给目标方法的入参.  \*/  @RequestMapping("/testModelAttribute")  public String testModelAttribute(User user){  System.***out***.println("修改: " + user);  return ***SUCCESS***;  } } |

# ModelAttribute注解之示例代码

|  |
| --- |
|  |

# ModelAttribute注解之运行原理

|  |
| --- |
|  |

# ModelAttribute注解之源码分析

|  |
| --- |
|  |

# 如何确定目标方法POJO类型参数

|  |
| --- |
|  |

# ModelAttribute注解修饰POJO类型的入参

|  |
| --- |
|  |

# ModelAttribute注解修饰POJO类型的入参

|  |
| --- |
|  |

# SessionAttributes注解引发的异常

|  |
| --- |
|  |

# 视图解析流程分析

## index.jsp

|  |
| --- |
| <a href="springmvc/testViewAndViewResolver">Test ViewAndViewResolver</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  @RequestMapping("/testViewAndViewResolver")  public String testViewAndViewResolver(){  System.***out***.println("testViewAndViewResolver");  return ***SUCCESS***;  } } |

# JstlView

|  |
| --- |
|  |

## i18n.properties

|  |
| --- |
| i18n.username=Username i18n.password=Password |

## i18n\_en\_US.properties

|  |
| --- |
| i18n.username=Username i18n.password=Password |

## i18n\_zh\_CN.properties

|  |
| --- |
| i18n.username=用户名 i18n.password=密码 |

## index.jsp

|  |
| --- |
| <a href="springmvc/testViewAndViewResolver">Test ViewAndViewResolver</a> |

## success.jsp

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%> <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>  <fmt:message key="i18n.username"></fmt:message> <br><br>  <fmt:message key="i18n.password"></fmt:message> |

## dispatcherServlet-servlet.xml

|  |
| --- |
| <!-- 配置国际化资源文件 --> <bean id="messageSource"  class="org.springframework.context.support.ResourceBundleMessageSource">  <property name="basename" value="i18n"></property> </bean> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  @RequestMapping("/testViewAndViewResolver")  public String testViewAndViewResolver(){  System.***out***.println("testViewAndViewResolver");  return ***SUCCESS***;  } } |

# mvc\_view-controller标签

## dispatcherServlet-servlet.xml

|  |
| --- |
| <!-- 配置直接转发的页面 --> <!--  可以直接相应转发的页面, 而无需再经过 Handler 的方法.  但点击跳转就不能用了，需要配置mvc:annotation-driven  --> <mvc:view-controller path="/success" view-name="success"/>  直接输入    <!-- 在实际开发中通常都需配置 mvc:annotation-driven 标签 --> <mvc:annotation-driven></mvc:annotation-driven> |

# 自定义视图

## dispatcherServlet-servlet.xml

|  |
| --- |
| InternalResourceViewResolver 也是视图解析器  <!-- 配置视图 BeanNameViewResolver 解析器: 使用视图的名字来解析视图 --> <!-- 通过 order 属性来定义视图解析器的优先级, order 值越小优先级越高 --> <bean class="org.springframework.web.servlet.view.BeanNameViewResolver">  <property name="order" value="100"></property> </bean> |

## index.jsp

|  |
| --- |
| <a href="springmvc/testView">Test View</a> |

## HelloView

|  |
| --- |
| package com.atguigu.springmvc.views;  import org.springframework.stereotype.Component; import org.springframework.web.servlet.View;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import java.util.Date; import java.util.Map;  @Component public class HelloView implements View {   @Override  public String getContentType() {  return "text/html";  }   @Override  public void render(Map<String, ?> model, HttpServletRequest request,  HttpServletResponse response) throws Exception {  response.getWriter().print("hello view, time: " + new Date());  } } |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  @RequestMapping("/testView")  public String testView() {  System.***out***.println("testView");  return "helloView";  } } |

# 重定向

## index.jsp

|  |
| --- |
| <a href="springmvc/testRedirect">Test Redirect</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.handlers;  import java.io.IOException; import java.io.Writer; import java.util.Arrays; import java.util.Date; import java.util.Map;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.CookieValue; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.RequestHeader; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod; import org.springframework.web.bind.annotation.RequestParam; import org.springframework.web.bind.annotation.SessionAttributes; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.entities.User;  @RequestMapping("/springmvc") @Controller public class SpringMVCTest {   private static final String ***SUCCESS*** = "success";  @RequestMapping("/testRedirect")  public String testRedirect(){  System.***out***.println("testRedirect");  return "redirect:/index.jsp";  } } |

# RESTRUL\_CRUD\_需求

|  |
| --- |
|  |

# RESTRUL\_CRUD\_显示所有员工信息

|  |
| --- |
|  |

# RESTRUL\_CRUD\_添加操作&表单标签

|  |
| --- |
|  |

# RESTRUL\_CRUD\_删除操作&处理静态资源

|  |
| --- |
|  |

# RESTRUL\_CRUD\_修改操作

|  |
| --- |
|  |

# 数据绑定流程分析

|  |
| --- |
| <!--   1. 数据类型转换  2. 数据类型格式化  3. 数据校验.   1). 如何校验 ? 注解 ?  ①. 使用 JSR 303 验证标准  ②. 加入 hibernate validator 验证框架的 jar 包  ③. 在 SpringMVC 配置文件中添加 <mvc:annotation-driven />  ④. 需要在 bean 的属性上添加对应的注解  ⑤. 在目标方法 bean 类型的前面添加 @Valid 注解  2). 验证出错转向到哪一个页面 ?  注意: 需校验的 Bean 对象和其绑定结果对象或错误对象时成对出现的，它们之间不允许声明其他的入参  3). 错误消息 ? 如何显示, 如何把错误消息进行国际化 --> |

# 自定义类型转换器

## springmvc.xml

|  |
| --- |
|  |

## input.jsp

|  |
| --- |
| <form action="testConversionServiceConverer" method="POST">  <!-- lastname-email-gender-department.id 例如: GG-gg@atguigu.com-0-105 -->  Employee: <input type="text" name="employee"/>  <input type="submit" value="Submit"/> </form> <br><br> |

## EmployeeConverter

|  |
| --- |
| package com.atguigu.springmvc.converters;  import org.springframework.core.convert.converter.Converter; import org.springframework.stereotype.Component;  import com.atguigu.springmvc.crud.entities.Department; import com.atguigu.springmvc.crud.entities.Employee;  @Component public class EmployeeConverter implements Converter<String, Employee> {   @Override  public Employee convert(String source) {  if(source != null){  String [] vals = source.split("-");  //GG-gg@atguigu.com-0-105  if(vals != null && vals.length == 4){  String lastName = vals[0];  String email = vals[1];  Integer gender = Integer.parseInt(vals[2]);  Department department = new Department();  department.setId(Integer.parseInt(vals[3]));   Employee employee = new Employee(null, lastName, email, gender, department);  System.***out***.println(source + "--convert--" + employee);  return employee;  }  }  return null;  }  } |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;   @RequestMapping("/testConversionServiceConverer")  public String testConverter(@RequestParam("employee") Employee employee){  System.***out***.println("save: " + employee);  employeeDao.save(employee);  return "redirect:/emps";  } } |

# annotation-driven配置

|  |
| --- |
| ------------ |

# InitBinder注解

|  |
| --- |
|  |

## EmployeeHandler

|  |
| --- |
| package com.atguigu.springmvc.crud.handlers;  import com.atguigu.springmvc.crud.dao.DepartmentDao; import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import org.springframework.validation.Errors; import org.springframework.validation.FieldError; import org.springframework.web.bind.WebDataBinder; import org.springframework.web.bind.annotation.\*;  import javax.validation.Valid; import java.util.Map;   @Controller public class EmployeeHandler {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private DepartmentDao departmentDao;    @InitBinder  public void initBinder(WebDataBinder binder) {  binder.setDisallowedFields("lastName");  } } |

# 数据的格式化

|  |
| --- |
|  |

## 数据转换 & 数据格式化 & 数据校验

|  |
| --- |
| springmvc.xml  <mvc:annotation-driven></mvc:annotation-driven>  或 可以自定义类型转换器也可以使用自带的类型转换器  <mvc:annotation-driven conversion-service="conversionService"></mvc:annotation-driven>  <!-- 配置 ConversionService --> <bean id="conversionService"  class="org.springframework.format.support.FormattingConversionServiceFactoryBean">  <property name="converters">  <set>  <ref bean="employeeConverter"/>  </set>  </property> </bean>  public class Employee {  private Integer id;  @NotEmpty  private String lastName;   @Email  private String email;  //1 male, 0 female  private Integer gender;   private Department department;  --------------------------------------  @Past  @DateTimeFormat(pattern="yyyy-MM-dd")  private Date birth;   @NumberFormat(pattern="#,###,###.#")  private Float salary;  input.jsp  <!--   1. 数据类型转换  2. 数据类型格式化  3. 数据校验.   1). 如何校验 ? 注解 ?  ①. 使用 JSR 303 验证标准  ②. 加入 hibernate validator 验证框架的 jar 包  ③. 在 SpringMVC 配置文件中添加 <mvc:annotation-driven />  ④. 需要在 bean 的属性上添加对应的注解  ⑤. 在目标方法 bean 类型的前面添加 @Valid 注解  2). 验证出错转向到哪一个页面 ?  注意: 需校验的 Bean 对象和其绑定结果对象或错误对象时成对出现的，它们之间不允许声明其他的入参  3). 错误消息 ? 如何显示, 如何把错误消息进行国际化  -->  Birth: <form:input path="birth"/>  <form:errors path="birth"></form:errors>  <br>  Salary: <form:input path="salary"/>  <br>  <input type="submit" value="Submit"/> </form:form> |

## 类型转换失败

|  |
| --- |
|  |

# JSR303数据校验

|  |
| --- |
|  |

|  |
| --- |
| <!--   1. 数据类型转换  2. 数据类型格式化  3. 数据校验.   1). 如何校验 ? 注解 ?  ①. 使用 JSR 303 验证标准  ②. 加入 hibernate validator 验证框架的 jar 包  ③. 在 SpringMVC 配置文件中添加 <mvc:annotation-driven />  ④. 需要在 bean 的属性上添加对应的注解  ⑤. 在目标方法 bean 类型的前面添加 @Valid 注解  2). 验证出错转向到哪一个页面 ?  注意: 需校验的 Bean 对象和其绑定结果对象或错误对象时成对出现的，它们之间不允许声明其他的入参  3). 错误消息 ? 如何显示, 如何把错误消息进行国际化 -->  **引入jar包**    public class Employee {   private Integer id;  @NotEmpty  private String lastName;   @Email  private String email;  @RequestMapping(value = "/emp", method = RequestMethod.***POST***) public String save(**@Valid** Employee employee, Errors result,  Map<String, Object> map) {  System.***out***.println("save:" + employee);  if (result.getErrorCount() > 0) {  System.***out***.println("出错了!");   for(FieldError error: result.getFieldErrors()) {  System.***out***.println(error.getField() + ":" + error.getDefaultMessage());  }  //若验证出错, 则转向定制的页面  map.put("departments", departmentDao.getDepartments());  return "input";  }   employeeDao.save(employee);  return "redirect:/emps"; } |

# 错误消息的显示及国际化

|  |
| --- |
| 如何显示错误消息和如何定制错误消息  input.jsp  <form:errors path="\*"></form:errors> <form:errors path="email"></form:errors> |

# 返回JSON

|  |
| --- |
|  |

## index.jsp

|  |
| --- |
| <script type="text/javascript" src="scripts/jquery-1.9.1.min.js"></script> <script type="text/javascript">  $(function(){  $("#testJson").click(function(){  var url = this.href;  var args = {};  $.post(url, args, function(data){  for(var i = 0; i < data.length; i++){  var id = data[i].id;  var lastName = data[i].lastName;   *alert*(id + ": " + lastName);  }  });  return false;  });  }) </script>  <a href="testJson" id="testJson">Test Json</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;   @ResponseBody  @RequestMapping("/testJson")  public Collection<Employee> testJson(){  return employeeDao.getAll();  } } |

# HttpMessageConverter原理

|  |
| --- |
| 1584081185(1) |

# 使用HttpMessageConverter

|  |
| --- |
|  |

## index.jsp

|  |
| --- |
| <form action="testHttpMessageConverter" method="POST" enctype="multipart/form-data">  File: <input type="file" name="file"/>  Desc: <input type="text" name="desc"/>  <input type="submit" value="Submit"/> </form>  <br><br>  <a href="testResponseEntity">Test ResponseEntity</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @RequestMapping("/testResponseEntity")  public ResponseEntity<byte[]> testResponseEntity(HttpSession session)  throws IOException{  byte [] body = null;  ServletContext servletContext = session.getServletContext();  InputStream in = servletContext.getResourceAsStream("/files/abc.txt");  body = new byte[in.available()];  in.read(body);   HttpHeaders headers = new HttpHeaders();  headers.add("Content-Disposition", "attachment;filename=abc.txt");   HttpStatus statusCode = HttpStatus.***OK***;   ResponseEntity<byte[]> response = new ResponseEntity<byte[]>(body, headers, statusCode);  return response;  }  @ResponseBody  @RequestMapping("/testHttpMessageConverter")  public String testHttpMessageConverter(@RequestBody String body){  System.***out***.println(body);  return "helloworld! " + new Date();  } } |

# 国际化\_概述

## i18n.properties

|  |
| --- |
| NotEmpty.employee.lastName=^^LastName不能为空. Email.employee.email=Email地址不合法 Past.employee.birth=Birth不能是一个将来的时间.   typeMismatch.employee.birth=Birth不是一个日期.   i18n.user=User i18n.password=Password |

## i18n\_en\_US.properties

|  |
| --- |
| NotEmpty.employee.lastName=^^LastName不能为空. Email.employee.email=Email地址不合法 Past.employee.birth=Birth不能是一个将来的时间.   typeMismatch.employee.birth=Birth不是一个日期.   i18n.user=User i18n.password=Password |

## i18n\_zh\_CN.properties

|  |
| --- |
| NotEmpty.employee.lastName=^^LastName不能为空. Email.employee.email=Email地址不合法 Past.employee.birth=Birth不能是一个将来的时间.   typeMismatch.employee.birth=Birth不是一个日期.   i18n.user=用户名 i18n.password=密码 |

## springmvc.xml

|  |
| --- |
| <!-- 配置国际化资源文件 --> <bean id="messageSource"  class="org.springframework.context.support.ResourceBundleMessageSource">  <property name="basename" value="i18n"></property> </bean>  <mvc:view-controller path="/i18n" view-name="i18n"/> <mvc:view-controller path="/i18n2" view-name="i18n2"/> |

## index.jsp

|  |
| --- |
| <!--  关于国际化:  1. 在页面上能够根据浏览器语言设置的情况对文本(不是内容), 时间, 数值进行本地化处理  2. 可以在 bean 中获取国际化资源文件 Locale 对应的消息  3. 可以通过超链接切换 Locale, 而不再依赖于浏览器的语言设置情况   解决:  1. 使用 JSTL 的 fmt 标签  2. 在 bean 中注入 ResourceBundleMessageSource 的示例, 使用其对应的 getMessage 方法即可  3. 配置 LocalResolver 和 LocaleChangeInterceptor --> <br><br> <a href="i18n">I18N PAGE</a> |

## i18n.jsp

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%> <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> <html> <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>Insert title here</title> </head> <body>    <fmt:message key="i18n.user"></fmt:message>    <br><br>  <a href="i18n2">I18N2 PAGE</a>    <br><br>  <a href="i18n?locale=zh\_CH">中文</a>    <br><br>  <a href="i18n?locale=en\_US">英文</a>   </body> </html> |

## i18n2.jsp

|  |
| --- |
| <%@ page language="java" contentType="text/html; charset=UTF-8"  pageEncoding="UTF-8"%> <%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>   <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd"> <html> <head> <meta http-equiv="Content-Type" content="text/html; charset=UTF-8"> <title>Insert title here</title> </head> <body>    <fmt:message key="i18n.password"></fmt:message>    <br><br>  <a href="i18n">I18N PAGE</a>   </body> </html> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @RequestMapping("/i18n")  public String testI18n(Locale locale){  String val = messageSource.getMessage("i18n.user", null, locale);  System.***out***.println(val);  return "i18n";  } } |

# 国际化\_前两个问题

# 国际化\_通过超链接切换Locale

|  |
| --- |
|  |

## springmvc.xml

|  |
| --- |
| <!-- 配置 SessionLocalResolver --> <bean id="localeResolver"  class="org.springframework.web.servlet.i18n.SessionLocaleResolver"></bean>  <!--配置LocaleChangeInterceptor--> <mcv:interceptors>  <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"></bean> </mvc:interceptors> |

## i18n.jsp

|  |
| --- |
| <a href="i18n?locale=zh\_CH">中文</a>  <br><br> <a href="i18n?locale=en\_US">英文</a> |

# 文件上传

|  |
| --- |
|  |

## springmvc.xml

|  |
| --- |
| <!-- 配置 MultipartResolver --> <bean id="multipartResolver"  class="org.springframework.web.multipart.commons.CommonsMultipartResolver">  <property name="defaultEncoding" value="UTF-8"></property>  <property name="maxUploadSize" value="1024000"></property> </bean> |

## index.jsp

|  |
| --- |
| <form action="testFileUpload" method="POST" enctype="multipart/form-data">  File: <input type="file" name="file"/>  Desc: <input type="text" name="desc"/>  <input type="submit" value="Submit"/> </form> |

## success.jsp

|  |
| --- |
| <h4>Success Page</h4> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @RequestMapping("/testFileUpload")  public String testFileUpload(@RequestParam("desc") String desc,  @RequestParam("file") MultipartFile file) throws IOException {  System.***out***.println("desc: " + desc);  System.***out***.println("OriginalFilename: " + file.getOriginalFilename());  System.***out***.println("InputStream: " + file.getInputStream());  return "success";  } } |

# 第一个自定义的拦截器

|  |
| --- |
|  |

## springmvc.xml

|  |
| --- |
| <mvc:interceptors>  <!-- 配置自定义的拦截器 -->  <bean class="com.atguigu.springmvc.interceptors.FirstInterceptor"></bean>   <!-- 配置 LocaleChanceInterceptor -->  <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"></bean> </mvc:interceptors> |

## FirstInterceptor

|  |
| --- |
| package com.atguigu.springmvc.interceptors;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.web.servlet.HandlerInterceptor; import org.springframework.web.servlet.ModelAndView;  public class FirstInterceptor implements HandlerInterceptor{   /\*\*  \* 该方法在目标方法之前被调用.  \* 若返回值为 true, 则继续调用后续的拦截器和目标方法.  \* 若返回值为 false, 则不会再调用后续的拦截器和目标方法.  \*  \* 可以考虑做权限. 日志, 事务等.  \*/  @Override  public boolean preHandle(HttpServletRequest request,  HttpServletResponse response, Object handler) throws Exception {  System.***out***.println("[FirstInterceptor] preHandle");  return true;  }   /\*\*  \* 调用目标方法之后, 但渲染视图之前.  \* 可以对请求域中的属性或视图做出修改.  \*/  @Override  public void postHandle(HttpServletRequest request,  HttpServletResponse response, Object handler,  ModelAndView modelAndView) throws Exception {  System.***out***.println("[FirstInterceptor] postHandle");  }   /\*\*  \* 渲染视图之后被调用. 释放资源  \*/  @Override  public void afterCompletion(HttpServletRequest request,  HttpServletResponse response, Object handler, Exception ex)  throws Exception {  System.***out***.println("[FirstInterceptor] afterCompletion");  }  } |

# 拦截器的配置

## springmvc.xml

|  |
| --- |
| <mvc:interceptors>  <!-- 配置自定义的拦截器 -->  <bean class="com.atguigu.springmvc.interceptors.FirstInterceptor"></bean>   <!-- 配置拦截器(不)作用的路径 -->  <mvc:interceptor>  <mvc:mapping path="/emps"/>  <bean class="com.atguigu.springmvc.interceptors.SecondInterceptor"></bean>  </mvc:interceptor>   <!-- 配置 LocaleChanceInterceptor -->  <bean class="org.springframework.web.servlet.i18n.LocaleChangeInterceptor"></bean> </mvc:interceptors> |

## SecondInterceptor

|  |
| --- |
| package com.atguigu.springmvc.interceptors;  import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse;  import org.springframework.web.servlet.HandlerInterceptor; import org.springframework.web.servlet.ModelAndView;  public class SecondInterceptor implements HandlerInterceptor{   /\*\*  \* 该方法在目标方法之前被调用.  \* 若返回值为 true, 则继续调用后续的拦截器和目标方法.  \* 若返回值为 false, 则不会再调用后续的拦截器和目标方法.  \*  \* 可以考虑做权限. 日志, 事务等.  \*/  @Override  public boolean preHandle(HttpServletRequest request,  HttpServletResponse response, Object handler) throws Exception {  System.***out***.println("[SecondInterceptor] preHandle");  return false;  }   /\*\*  \* 调用目标方法之后, 但渲染视图之前.  \* 可以对请求域中的属性或视图做出修改.  \*/  @Override  public void postHandle(HttpServletRequest request,  HttpServletResponse response, Object handler,  ModelAndView modelAndView) throws Exception {  System.***out***.println("[SecondInterceptor] postHandle");  }   /\*\*  \* 渲染视图之后被调用. 释放资源  \*/  @Override  public void afterCompletion(HttpServletRequest request,  HttpServletResponse response, Object handler, Exception ex)  throws Exception {  System.***out***.println("[SecondInterceptor] afterCompletion");  } } |

# 多个拦截方法的执行顺序

|  |
| --- |
|  |

# 异常处理\_ExceptionHandler注解

|  |
| --- |
|  |

## index.jsp

|  |
| --- |
| <a href="testExceptionHandlerExceptionResolver?i=10">Test ExceptionHandlerExceptionResolver</a> |

## error.jsp

|  |
| --- |
| <h4>Error Page</h4>  ${requestScope.exception} |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @ExceptionHandler({RuntimeException.class})  public ModelAndView handleArithmeticException2(Exception ex){  System.***out***.println("[出异常了]: " + ex);  ModelAndView mv = new ModelAndView("error");  mv.addObject("exception", ex);  return mv;  }  /\*\*  \* 1. 在 @ExceptionHandler 方法的入参中可以加入 Exception 类型的参数, 该参数即对应发生的异常对象  \* 2. @ExceptionHandler 方法的入参中不能传入 Map. 若希望把异常信息传导页面上, 需要使用 ModelAndView 作为返回值  \* 3. @ExceptionHandler 方法标记的异常有优先级的问题.  \* 4. @ControllerAdvice: 如果在当前 Handler 中找不到 @ExceptionHandler 方法来出来当前方法出现的异常,  \* 则将去 @ControllerAdvice 标记的类中查找 @ExceptionHandler 标记的方法来处理异常.  \*/  @ExceptionHandler({ArithmeticException.class})  public ModelAndView handleArithmeticException(Exception ex){  System.***out***.println("出异常了: " + ex);  ModelAndView mv = new ModelAndView("error");  mv.addObject("exception", ex);  return mv;  }  @RequestMapping("/testExceptionHandlerExceptionResolver")  public String testExceptionHandlerExceptionResolver(@RequestParam("i") int i){  System.***out***.println("result: " + (10 / i));  return "success";  } } |

## SpringMVCTestExceptionHandler

|  |
| --- |
| package com.atguigu.springmvc.crud.test;  import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; import org.springframework.web.servlet.ModelAndView;  @ControllerAdvice public class SpringMVCTestExceptionHandler {  @ExceptionHandler(ArithmeticException.class)  public ModelAndView handleArithmeticException(Exception ex) {  System.***out***.println("----> 出异常了: " + ex);  ModelAndView mv = new ModelAndView("error");  mv.addObject("exception", ex);  return mv;  } } |

# 异常处理\_ResponseStatusExceptionResolver

|  |
| --- |
|  |

## index.jsp

|  |
| --- |
| <a href="testResponseStatusExceptionResolver?i=10">Test ResponseStatusExceptionResolver</a> |

## SpringMVCTest

|  |
| --- |
| package com.atguigu.springmvc.test;  import java.io.IOException; import java.io.InputStream; import java.util.Collection; import java.util.Date; import java.util.Locale; import java.util.Map;  import javax.servlet.ServletContext; import javax.servlet.http.HttpSession;  import com.atguigu.springmvc.crud.test.UserNameNotMatchPasswordException; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.http.HttpHeaders; import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.stereotype.Controller; import org.springframework.web.bind.annotation.ControllerAdvice; import org.springframework.web.bind.annotation.ExceptionHandler; 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.RequestParam; import org.springframework.web.bind.annotation.ResponseBody; import org.springframework.web.bind.annotation.ResponseStatus; import org.springframework.web.multipart.MultipartFile; import org.springframework.web.servlet.ModelAndView;  import com.atguigu.springmvc.crud.dao.EmployeeDao; import com.atguigu.springmvc.crud.entities.Employee;  @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @ResponseStatus(reason="测试",value=HttpStatus.***NOT\_FOUND***)  @RequestMapping("/testResponseStatusExceptionResolver")  public String testResponseStatusExceptionResolver(@RequestParam("i") int i){  if(i == 13){  throw new UserNameNotMatchPasswordException();  }  System.***out***.println("testResponseStatusExceptionResolver...");   return "success";  } } |

## UserNameNotMatchPasswordException

|  |
| --- |
| package com.atguigu.springmvc.crud.test;  import org.springframework.http.HttpStatus; import org.springframework.web.bind.annotation.ResponseStatus;  @ResponseStatus(value = HttpStatus.***FORBIDDEN***, reason = "用户名和密码不匹配!") public class UserNameNotMatchPasswordException extends RuntimeException{  private static final long ***serialVersionUID*** = 1L; } |

# 异常处理\_DefaultHandlerExceptionResolver

|  |
| --- |
|  |

## index.jsp

|  |
| --- |
| <a href="testDefaultHandlerExceptionResolver">Test DefaultHandlerExceptionResolver</a> |

## SpringMVCTest

|  |
| --- |
| @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @RequestMapping(value="/testDefaultHandlerExceptionResolver",method=RequestMethod.***POST***)  public String testDefaultHandlerExceptionResolver(){  System.***out***.println("testDefaultHandlerExceptionResolver...");  return "success";  } } |

# 异常处理\_SimpleMappingExceptionResolver

|  |
| --- |
|  |

## springmvc.xml

|  |
| --- |
| <!-- 配置使用 SimpleMappingExceptionResolver 来映射异常 --> <bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">  <property name="exceptionAttribute" value="ex"></property>  <property name="exceptionMappings">  <props>  <prop key="java.lang.ArrayIndexOutOfBoundsException">error</prop>  </props>  </property> </bean> |

## index.jsp

|  |
| --- |
| <a href="testSimpleMappingExceptionResolver?i=2">Test SimpleMappingExceptionResolver</a> |

## SpringMVCTest

|  |
| --- |
| @Controller public class SpringMVCTest {   @Autowired  private EmployeeDao employeeDao;   @Autowired  private ResourceBundleMessageSource messageSource;  @RequestMapping("/testSimpleMappingExceptionResolver")  public String testSimpleMappingExceptionResolver(@RequestParam("i") int i){  String [] vals = new String[10];  System.***out***.println(vals[i]);  return "success";  } } |

# 运行流程图解

|  |
| --- |
|  |

# 源码解析

# Spring整合SpringMVC\_提出问题

|  |
| --- |
|  |

# Spring整合SpringMVC\_解决方案

|  |
| --- |
|  |

# SpringIOC 容器和 SpringMVC IOC 容器的关系

|  |
| --- |
|  |

# SpringMVC对比Struts2

|  |
| --- |
|  |