

课程目标

1. Springmvc处理请求乱码解决
2. (重点)实现springmvc注解控制器
3. (重点)掌握控制器中请求方法定义
4. (重点)Springmvc参数绑定
5. 在web.xml文件中添加以下配置解决springmvc取值乱码问题

|  |
| --- |
| *<!--配置解决乱的过滤器-->* <**filter**>  <**filter-name**>encodingFilter</**filter-name**> <**filter-class**>org.springframework.web.filter.CharacterEncodingFilter</**filter-class**>  <**init-param**>  <**param-name**>encoding</**param-name**>  <**param-value**>utf-8</**param-value**>  </**init-param**> </**filter**> <**filter-mapping**>  <**filter-name**>encodingFilter</**filter-name**>  <**url-pattern**>/\*</**url-pattern**> </**filter-mapping**> |

1. （重点）注解控制器的步骤:

1. 在springmvc配置文件中:使用component-scan标签扫描springmvc注解配置

|  |
| --- |
| <**context:component-scan base-package="控制器包所在包名:com.team.controller"**></**context:component-scan**> |

注意:在springmvc配置文件中导入context标签

**xmlns:context="http://www.springframework.org/schema/context"**

**xsi:schemaLocation=”[http://www.springframework.org/schema/context  
http://www.springframework.org/schema/context/spring-context-4.3.xsd](http://www.springframework.org/schema/contexthttp://www.springframework.org/schema/context/spring-context-4.3.xsd)”**

[2. 使用@Controller](mailto:2.使用@Controller)注解定义控制器

|  |
| --- |
| @Controller //指定当前类为控制器 **public class** LoginController2 {  *//定义一系列处理请求的方法: 一个请求一个方法* } |

3. 使用@RequestMapping注解定义处理请求的方法

|  |
| --- |
| *方式一:定义控制器方法返回ModelAndView //@RequestMapping(value = "/请求路径") //配置请求路径* @RequestMapping(value = **"/login2"**) **public** ModelAndView handlerLogin(){  System.***out***.println(**"我可以处理请求啦..."**);  **return null**; }  *定义控制方法二(推荐):返回String 表示视图页面* @RequestMapping(**"/login6"**) **public** String login6(HttpSession session, String username, String password, Model model){  处理请求代码….  **return "index.jsp"**; } |

# 三、springmvc的参数绑定(控制器方法的参数)

## 1.利用ServletApi获取请求中的数据

|  |
| --- |
| @RequestMapping(**"/login2"**) **public ModelAndView login2(HttpServletRequest request, HttpSession session){  //1.利用servletapi获取请求参数  String uname=request.getParameter("uname");  String password=request.getParameter("password");**  ModelAndView mav=**null**;  **if**(uname.equals(**"admin"**)&&password.equals(**"123"**)){  *//使用session保存登入的人* session.setAttribute(**"loginname"**,uname);  *//登入成功* mav=**new** ModelAndView(**"success.jsp"**);  }**else** {  *//登入失败* mav=**new** ModelAndView(**"index.jsp"**);  }  **return** mav; } |

2.利用参数绑定获取请求中的数据

一个表单对象(请求中的参数)对应一个方法的参数,要求表单对象名称与参数名称相同

|  |
| --- |
| @RequestMapping(**"/login3"**) **public** ModelAndView login3(@RequestParam(value = **"username"**,required = **false**) String username, String password,HttpSession session){  //1.利用参数绑定获取请求中的数据  //一个表单对象(请求中的参数)对应一个请求方法参数,请求中的参数名称与方法参数名称相同即可.  //2.利用@RequestParam注解指定请求参数与方法参数对应  //value指定请求参数名称 required表示参数 false可有可无 true一定传  //@RequestParam(value = "请求参数名称",required = false)ModelAndView mav=**null**;  **if**(username.equals(**"admin"**)&password.equals(**"123"**)){  mav=**new** ModelAndView(**"success.jsp"**);  *//使用session对象保存登入的人* session.setAttribute(**"loginname"**,username);  }  **else** {  mav=**new** ModelAndView(**"index.jsp"**);  }  **return** mav; *//跳转* } |

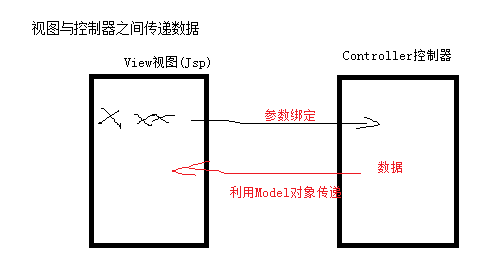
3. 将多个表单对象(请求中的参数)对应到实体参数中,要求表单对象名称与实体类的属性名称名称相同

|  |
| --- |
| @RequestMapping(value = **"/login4"**) **public** ModelAndView login4(HttpSession session,User user){  *//@RequestParam 用在方法参数上，用于指定请求参数  //利用参数绑定请求的数据:  //将多个表单对象(请求中的参数)对应到实体参数中,要求表单对象名称与实体类的属性名称相同* ModelAndView mav=**null**;  **if**(user.getUsername().equals(**"admin"**)&&user.getPassword().equals(**"123"**)){  *//使用session保存登入的人* session.setAttribute(**"info"**,user.getUsername());  mav=**new** ModelAndView(**"success.jsp"**);  }  **else** mav=**new** ModelAndView(**"index.jsp"**);  **return** mav; } |

1. 控制器将数据传递给页面
2. 利用ModelAndView
3. 利用参数的Model对象传递数据

|  |
| --- |
| *//定义控制器请求的方法返回String,String表示返回的网页路径* @RequestMapping(**"/login5"**) **public** String login5(User user, HttpSession session, Model model){ *//表示返回网页路径* **if**(user.getUname().equals(**"admin"**)&&user.getPassword().equals(**"123"**))  {  session.setAttribute(**"loginname"**,user.getUname());  //利用Model对象参数将控制器的数据，传递页面展示  //model.addAttribute("键",值);  model.addAttribute(**"user"**,user);  **return "success.jsp"**;*//返回网页 默认就是请求转发跳转* }**else**{  **return "index.jsp"**;  } } |

1. 视图与控制器之间传递数据



1. 视图解析器
2. 配置springmvc的视图解析器
3. 修改springmvc配置文件定义视图解析器

|  |
| --- |
| *<!--配置视图解析器-->* <**bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  *<!--前缀-->* <**property name="prefix" value="/"**></**property**>  *<!--后缀-->*  <**property name="suffix" value=".jsp"**></**property**> </**bean**> |

1. 在控制器方法中返回视图名称即可(针对请求转发跳转)

**return "success"**; *//返回视图名称(文件名)*

*结果：*

*return “success.jsp”*

扩展:以重定向的方式返回页面

（控制器就不能使用model给页面返回数据、视图解析也不起作用啦）

return “redirect:返回视图页面的路径”;