### JAVA第三阶段—DAY07-JAVA案例

1. SpringMVC入门

讲解SpringMVC的配置和控制器的编写

1）导入SpringMVC依赖

<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->  
<dependency>  
<groupId>org.springframework</groupId>  
<artifactId>spring-webmvc</artifactId>  
<version>5.2.8.RELEASE</version>  
</dependency>

2）Spring配置文件

<?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"  
      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   
      https://www.springframework.org/schema/context/spring-context.xsd   
      http://www.springframework.org/schema/mvc   
      https://www.springframework.org/schema/mvc/spring-mvc.xsd">  
      <!--扫描包-->  
      <context:component-scan base-package="com.blb.mvc"></context:component-scan>  
      <!--视图解析器 帮助查找视图文件的位置 index 转换为 /WEB-INF/pages/index.jsp -->  
      <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
          <!--配置视图的前缀-->  
          <property name="prefix" value="/WEB-INF/pages/"></property>  
          <!--配置视图的后缀-->  
          <property name="suffix" value=".jsp"></property>  
      </bean>  
      <!--处理静态资源-->  
      <mvc:default-servlet-handler/>  
      <!--启动注解配置-->  
      <mvc:annotation-driven/>  
</beans>

3） web.xml配置文件

<?xml version="1.0" encoding="UTF-8"?>  
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
        xmlns="http://xmlns.jcp.org/xml/ns/javaee"  
        xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"  
        id="WebApp\_ID" version="3.1">  
<display-name>Archetype Created Web Application</display-name>  
<!--配置核心控制器-->  
<servlet>  
  <servlet-name>dispatcherServlet</servlet-name>  
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
  <init-param>  
    <!--springmvc的配置文件路径-->  
    <param-name>contextConfigLocation</param-name>  
    <param-value>classpath:spring-mvc.xml</param-value>  
  </init-param>  
    <!--启动顺序-->  
  <load-on-startup>1</load-on-startup>  
</servlet>  
<servlet-mapping>  
  <servlet-name>dispatcherServlet</servlet-name>  
  <!--处理所有资源-->  
  <url-pattern>/</url-pattern>  
</servlet-mapping>  
</web-app>

4） 控制器

    /\*\*  
    \* 自定义控制器  
    \*/  
@Controller  
    public class MyController {  
     
        //配置请求的URL路径 http://localhost:8080/index  
        @RequestMapping(value = "/index",method = RequestMethod.GET)  
        public String toIndex(){  
            //视图解析器 ： index --> /WEB-INF/page/index.jsp  
            return "index";  
        }  
     
    }

1. 登录案例，参数绑定

分别演示参数绑定的几种方式：

1. 直接定义和表单元素name相同的参数
2. 通过@RequestParam注解配置参数
3. 通过对象进行参数绑定

<html>  
<head>  
  <title>登录</title>  
</head>  
<body>  
  <span style="color:red">${msg}</span>  
  <form action="/user/login" method="post">  
      <input name="username" placeholder="请输入用户名"><br>  
      <input name="password" placeholder="请输入密码"><br>  
      <input type="submit" value="登录">  
  </form>  
</body>  
</html>

实现参数绑定的几种实现：

1）直接定义和表单元素name相同的参数

@Controller  
@RequestMapping("user")  
public class UserController {  
​  
  @PostMapping("/login")  
  public void login(String username, String password){...}  
}

2）如果名称不一样，可以通过@RequestParam注解配置参数

public void login(@RequestParam("username")String uname,   
@RequestParam("password")String pwd){...}

3）通过对象也可以进行参数绑定

public class User {  
​  
  private String username;  
  private String password;  
  ...  
}  
​  
public void login(User user){...}

4）通过传统的HttpServletRequest实现

public void login(HttpServletRequest req){  
      String username = req.getParameter("username");  
      String password = req.getParameter("password");  
}

1. 登录案例，页面跳转

演示forward和redirect两种页面跳转方式

@PostMapping("/login")

public String login(User user, Model model){

if("admin".equals(user.getUsername()) && "123".equals(user.getPassword())){

//保存用户数据

model.addAttribute("user",user);

//重定向

return "redirect:/index";

}

//保存错误提示数据

model.addAttribute("msg","账号或密码错误");

//请求转发

return "forward:/login";

}

1. Ajax交互

演示如何接收Ajax的JSON参数，以及如何返回JSON给Ajax

1） 导入依赖

SpringMVC需要JSON解析的相关依赖

<dependency>  
  <groupId>com.fasterxml.jackson.core</groupId>  
  <artifactId>jackson-databind</artifactId>  
  <version>2.9.5</version>  
</dependency>

2）用户对象

public class User {  
​  
  private String username;  
  private String password;  
​  
  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;  
  }  
​  
  @Override  
  public String toString() {  
      return "User{" +  
              "username='" + username + '\'' +  
              ", password='" + password + '\'' +  
              '}';  
  }  
​  
  public User(String username, String password) {  
      this.username = username;  
      this.password = password;  
  }  
​  
  public User() {  
  }  
}

3） 控制器，在参数中添加了@RequestBody，方法前添加@ResponseBody

将会接收一个User对象参数，然后返回一个User集合给Ajax

/\*\*  
\* 自定义控制器  
\*/  
@Controller  
public class MyController {  
​  
  @ResponseBody  
  @RequestMapping("/ajax")  
  public List<User> ajaxTest(@RequestBody User user) {  
      System.out.println("收到了JSON对象："+user);  
      //返回JSON集合给ajax  
      List<User> users = Arrays.asList(new User("user-1","123"),  
              new User("user-2","123"),  
              new User("user-3","123"));  
      return users;  
  }  
}  
​

4） user.jsp

<%@ page contentType="text/html;charset=UTF-8" language="java" isELIgnored="false" %>  
<html>  
<head>  
  <title>首页</title>  
</head>  
<body>  
  <input id="username" placeholder="请输入用户名"><br>  
  <input id="password" placeholder="请输入密码"><br>  
  <button>测试</button>  
  <script type="text/javascript" src="/js/jquery-3.3.1.js"></script>  
  <script>  
      //点击测试  
      $("button").click(function () {  
          var json = {username:$("#username").val(),password:$("#password").val()};  
          $.ajax({  
              type: "post",  
              dataType: "json",//数据为JSON格式  
              url: "/ajax",  
              contentType: "application/json;charset=UTF-8",//指定消息请求类型  
              data: JSON.stringify(json),//将json对象转成json对象  
              success: function (data) {  
                  console.log(data);  
              }  
          });  
      });  
  </script>  
</body>  
</html>