实验 12　Spring MVC 数据绑定和响应

一、 Spring MVC 数据绑定和响应

需求 1：简单数据绑定

1.默认数据绑定(Spring MVC 默认内置对象)

2.简单数据类型绑定（基本数据类型）

3.RESTful 风格数据绑定

需求 2：简单 Pojo 实体对象数据绑定

1.Pojo 实体对象数据绑定

需求 3：复杂对象数据绑定

1.集合 List 对象数据绑定

2.Json 数据绑定(列表数据)

需求 4：Pojo 复杂对象数据绑定

1.Pojo 复杂对象 List 数据绑定

2.Pojo 复杂对象 Map 数据绑定

需求 5：数据回写

1.携带数据 ModelAndView 回写页面

2.Json 数据回写页面

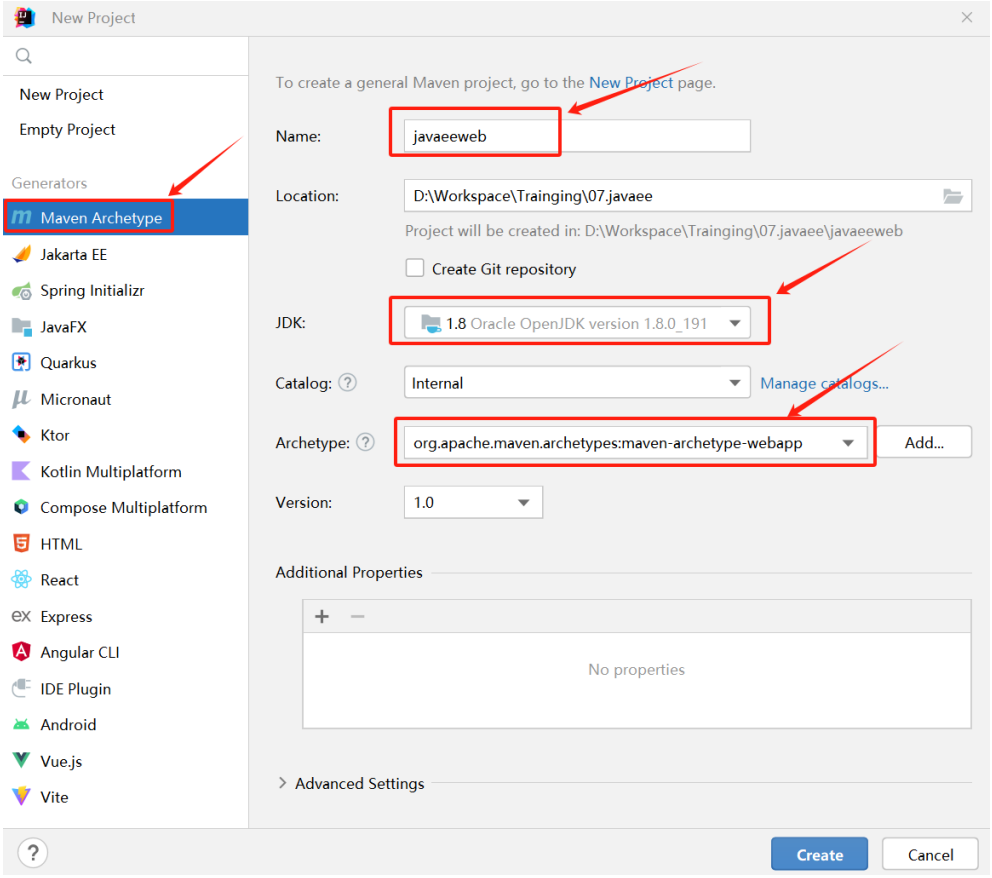
二、 开发准备

1、创建 Maven Web 项目

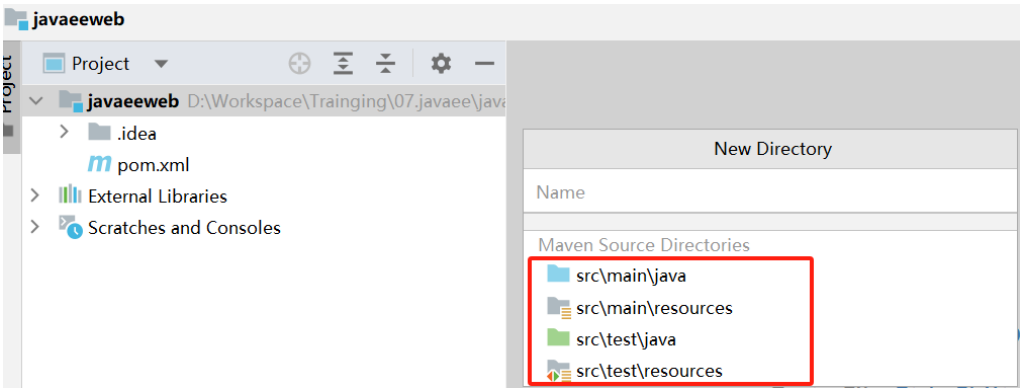
方式一：直接创建 trainingweb 项目（具体过程省略，只列出关键步骤）

IDEA 2022 创建新项目时（**其他版本根据实际**），选择创建 **Maven Archetype** 项目类型

即可，如下



注意，上一步创建后，**需要等待项目的自动构建。**最终会自动创建 webapp 目录以及相应的 web 相关的配置文件。**其他目录需要手工创建目录，如下：**

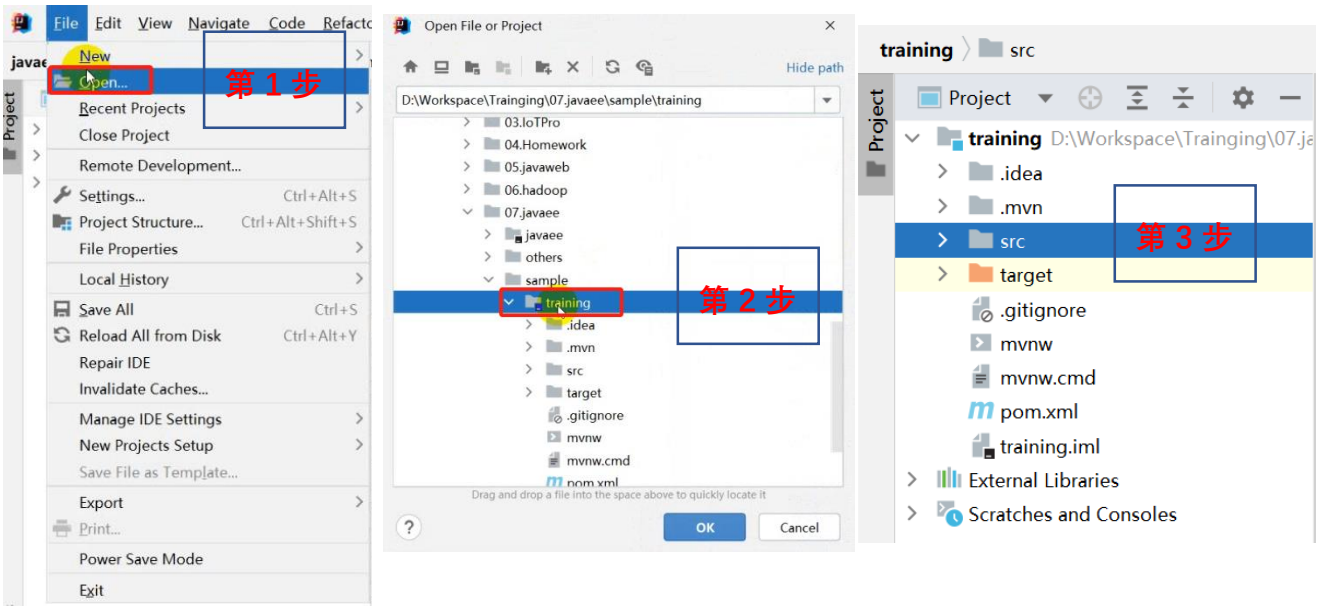


方式二：导入 Maven Web 项目（推荐）

Step1，下载并解压 trainingweb 工程。从【学习通】→【章节】→本章上机实验中找到 **trainingweb.zip** 压缩包，下载到本地，并解压成 trainingweb 目录。

Step2，导入 trainingweb 工程。打开 IDEA 工具，点击【File】→【Open…】→选中上一

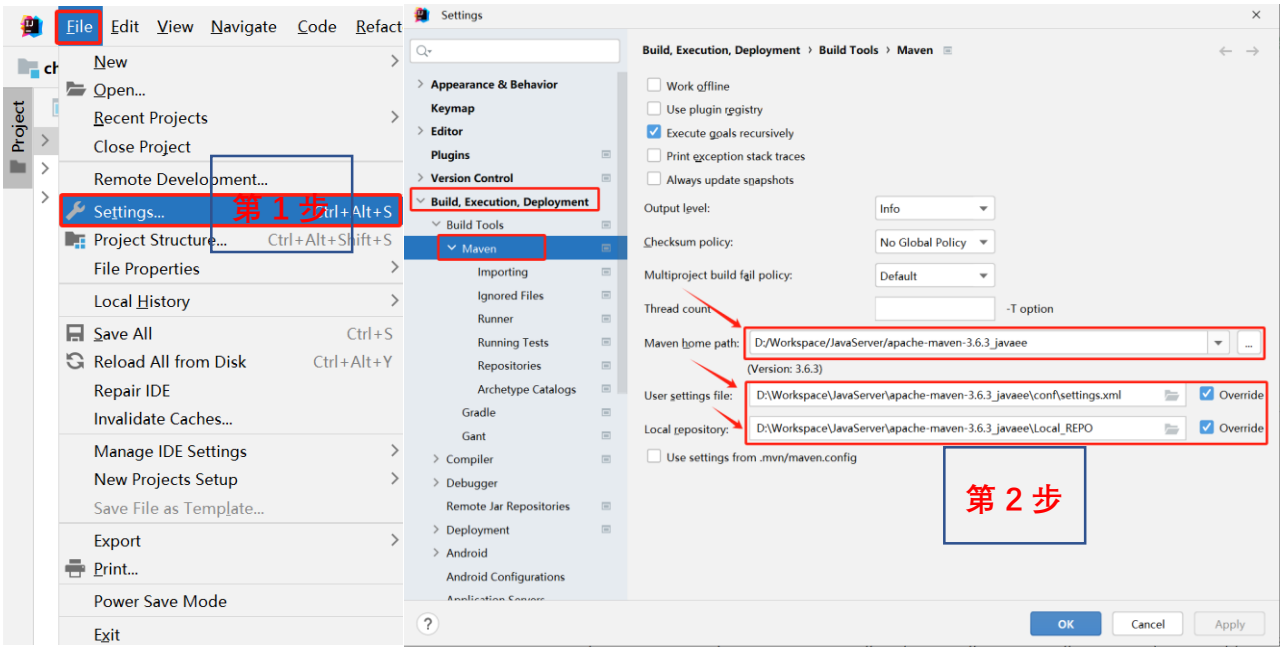
步解压的 trainingweb 工程，如下图所示。。。确定即可导入。



注意，导入工程后，对于 **IDEA 是 2023** 的版本，要点击左上角边的 文件夹小图标，即可查看到工程中的文件。

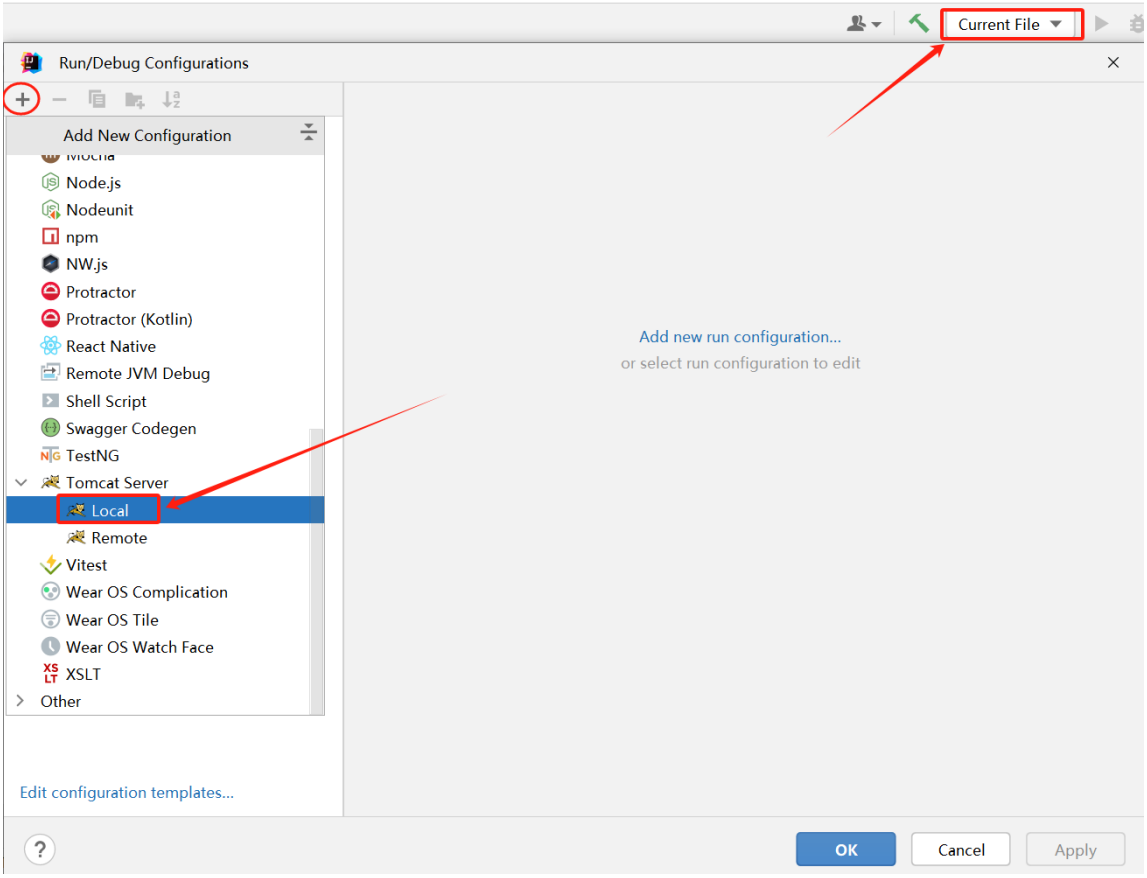
2、配置 Maven 构建工具

选择【File】→【Setting…】然后找到菜单位【Build, Execution, Deployment】→【Build Tools】→【Maven】，按如下图所示，将你的**本地 Maven 路径和配置文件**设置好即可。

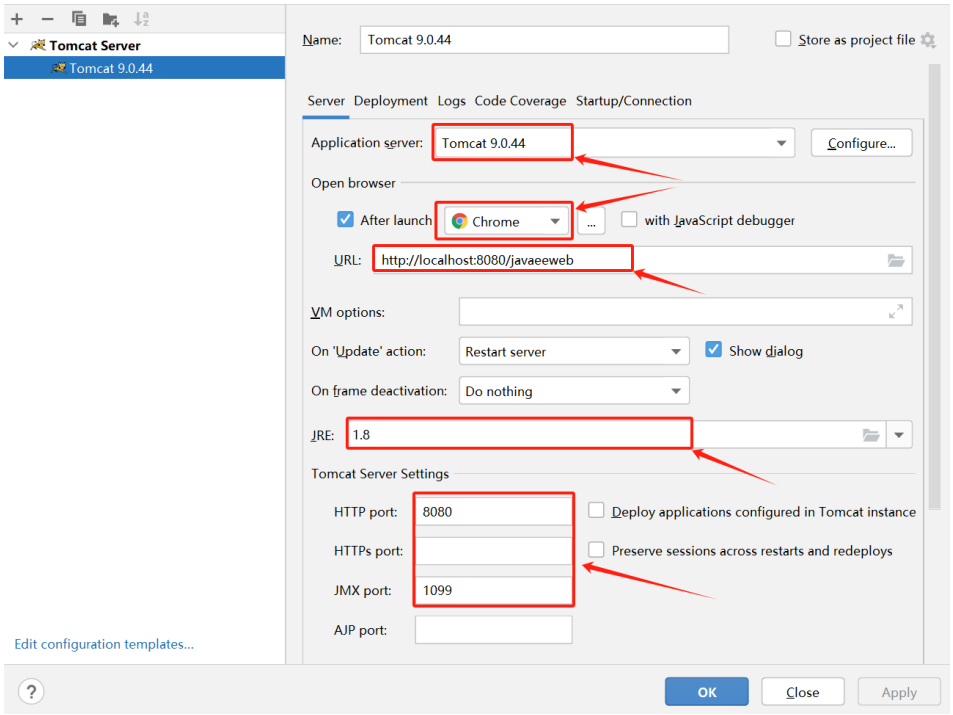


3、配置 Tomcat 服务器

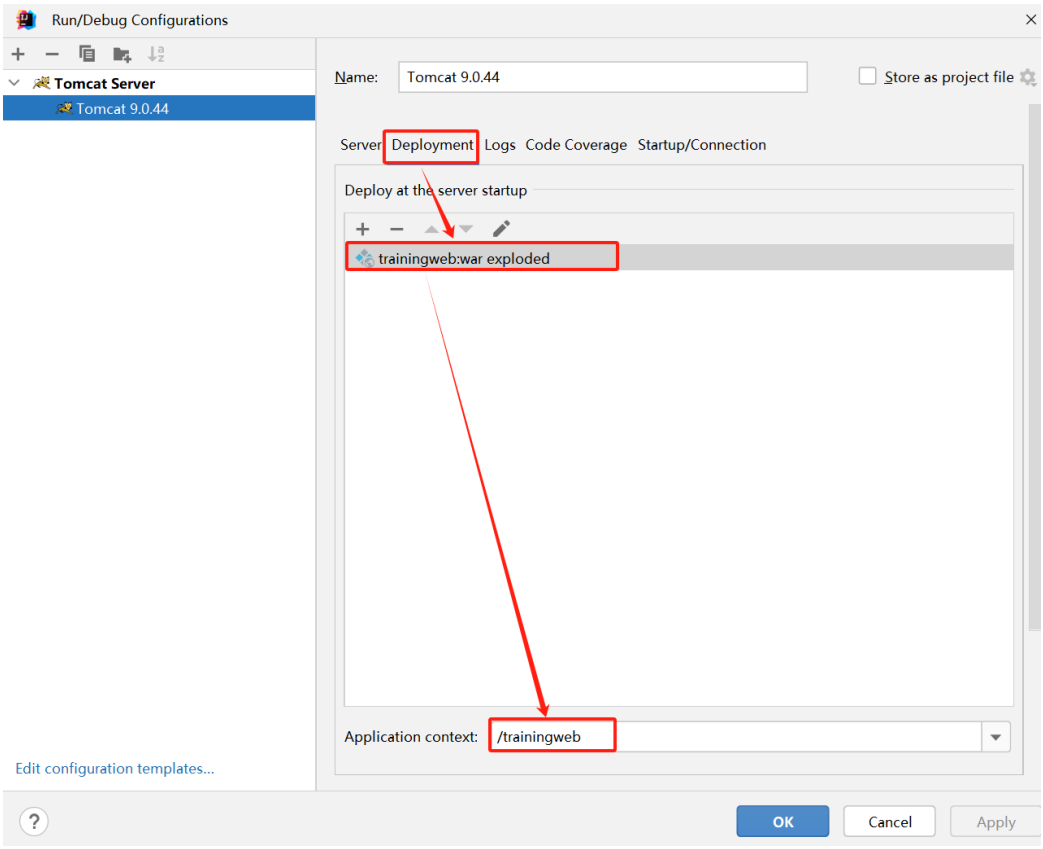
选择【**Edit Configuration**】→点击【**+**】按钮，拉到 Tomcat Server，选择→【**Local**】，如下图所示。



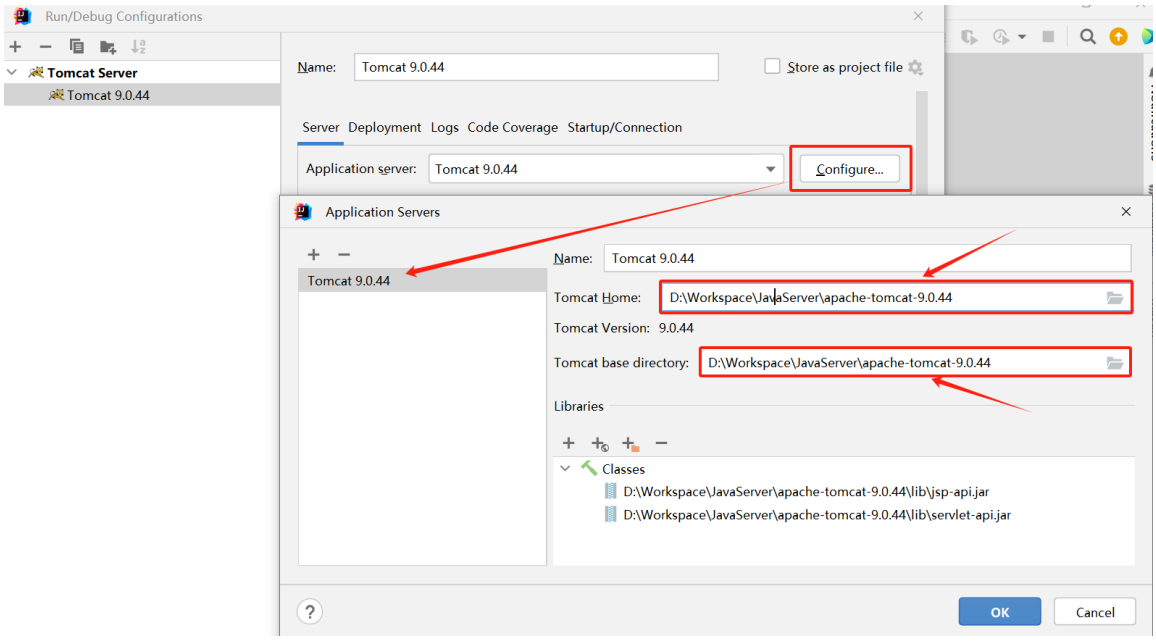
Step1：在 Server 页签配置 Tomcat 服务器，配置如下



Step2：切换到 Deployment 页签，设置部署包为 trainingweb:war exploded，配置如下



Step3：其中 Tomcat 服务器需要指向真实的 Tomcat 目录（指向你本地的 Tomcat），配置如下。



三、 程序开发

第 1 步：配置 pom.xml

在第 10 章、第 11 章配置的基础上，增加新的依赖。

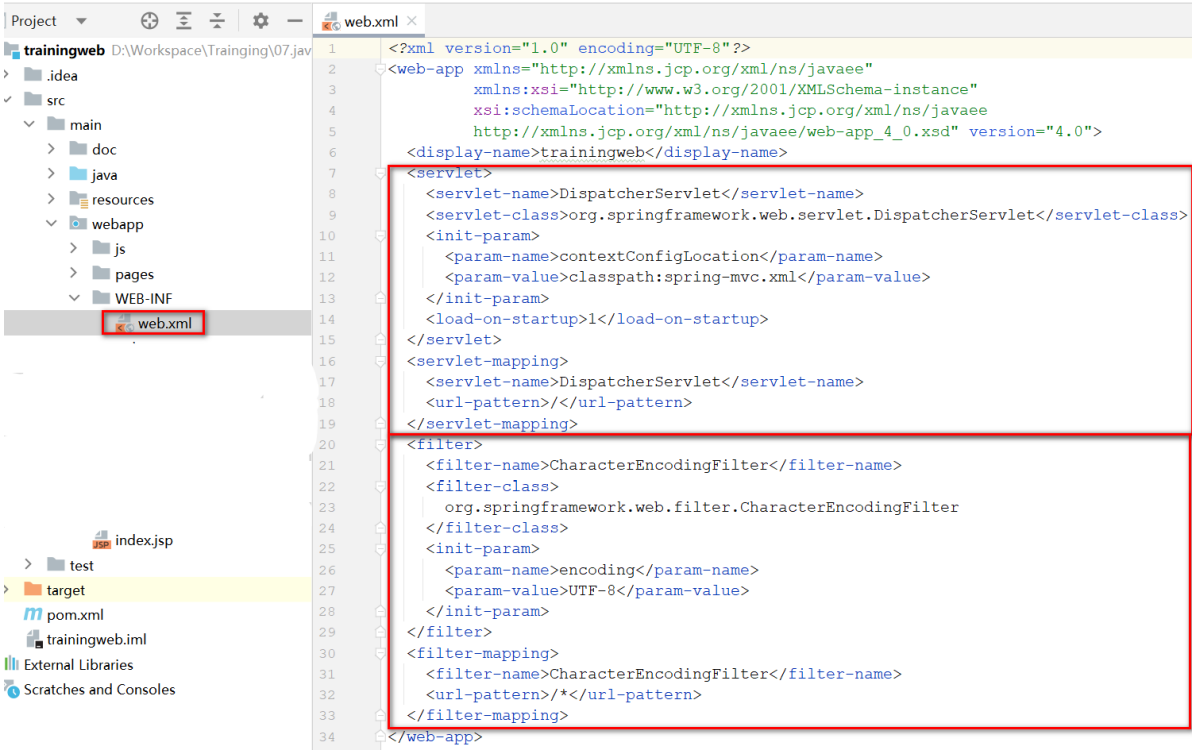
pom.xml (trainingweb)

```
<!--Jackson转换核心包依赖-->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-core</artifactId>
  <version>2.9.2</version>
</dependency>
<!--Jackson转换的数据绑定包依赖-->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.9.2</version>
</dependency>
<!--Jackson JSON转换注解包-->
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-annotations</artifactId>
  <version>2.9.0</version>
</dependency>
```

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-core</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.9.2</version>
</dependency>
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-annotations</artifactId>
  <version>2.9.0</version>
</dependency>
```

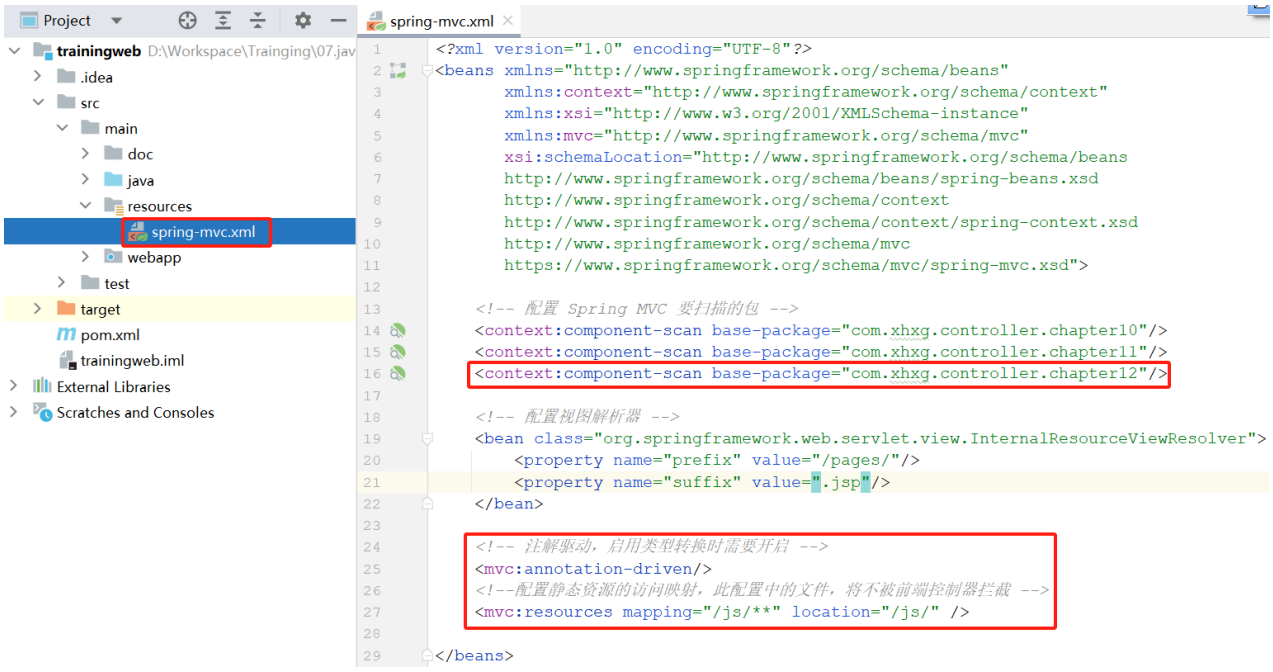
第 2 步：配置前端控制器 web.xml

需要在 web.xml 中配置前端控制器 DispatcherServlet 和配置过滤器，过滤器 Filter 的作用是保证请求传递的中文参数显示正常，不出现乱码。



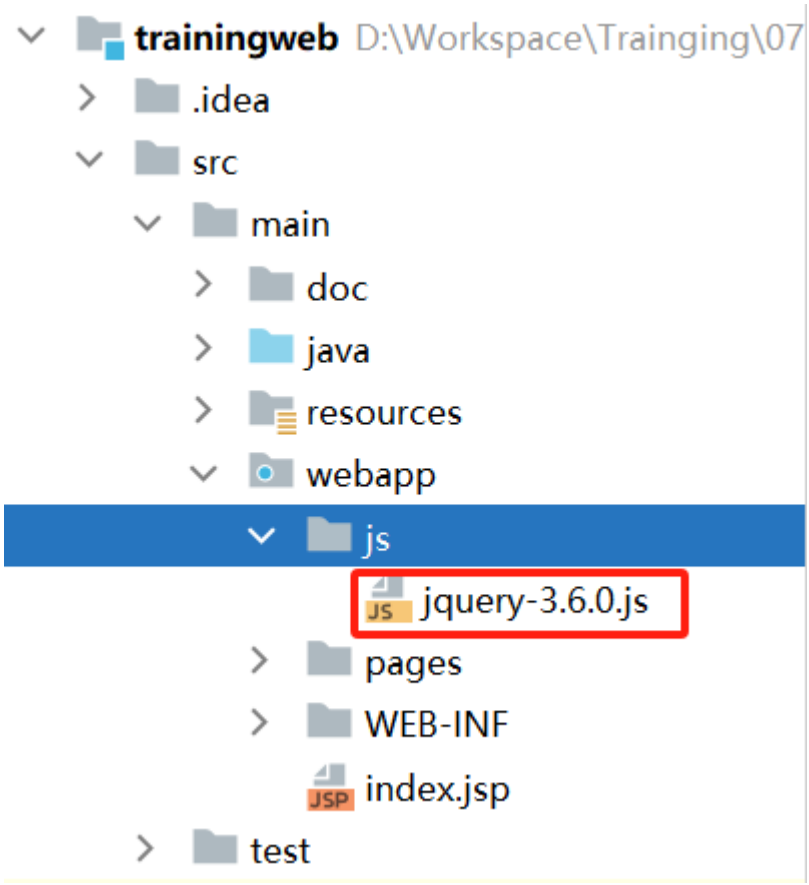
第 3 步：配置处理器映射信息和视图解析器 Spring-mvc.xml 文件

在 src/main/resources 目录下创建 spring-mvc.xml 文件，用于配置处理器映射信息和视图解析器。在实验 10 的配置基础上，加上本章的配置。具体配置如下所示。



第 4 步:、配置公共资源

a.在项目 src/main/webapp 目录下创建目录 js, 加入 jquery 资源库, 需要添加的版本为 **jquery-3.6.0.js**

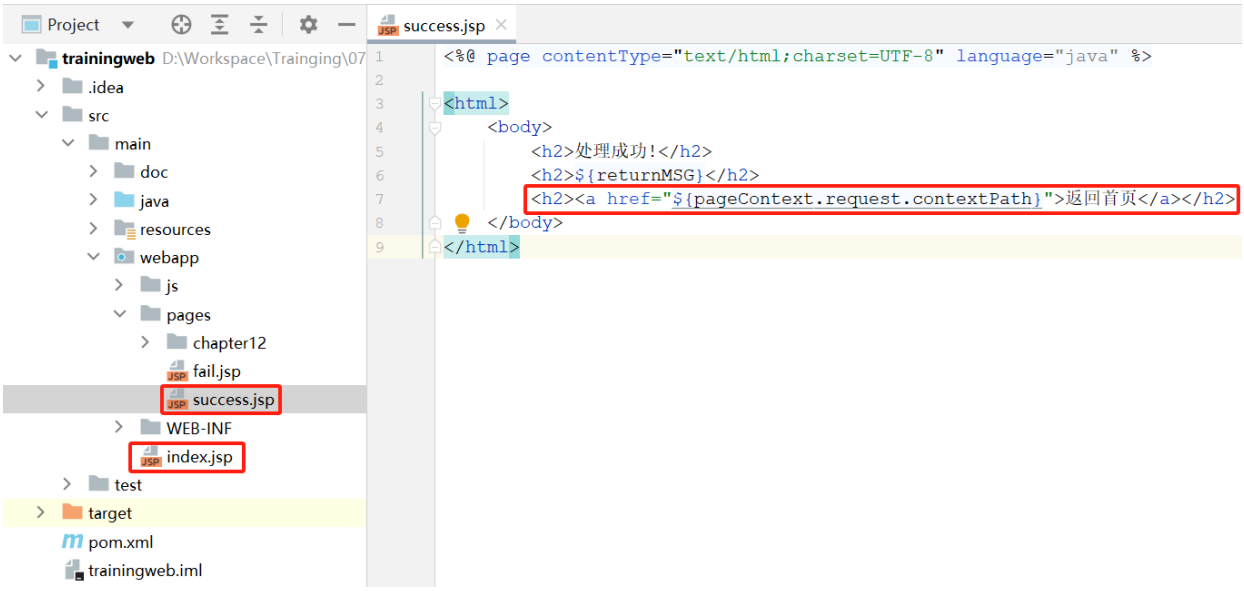


b.配置首页 index.jsp，添加链接地址，链接信息如下：

index.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
    <body>
        <h2>Hello World首页</h2>
        <br/>
        <h4>BoundSimpleController 简单数据绑定</h4>
        <a href="${pageContext.request.contextPath}/chapter12/getUserId?userid=123456">1.默认数据绑定(Spring MVC默认内置对象)</a>
        <br/>
        <a href="${pageContext.request.contextPath}/chapter12/getUserNameAndId?username=Emma&id=123">2.简单数据类型绑定(基本数据类型)</a>
        <br/>
        <a href="${pageContext.request.contextPath}/chapter12/user/jack">3.RESTful风格数据绑定</a>

        <h4>BoundSimpleController 简单Pojo实体对象数据绑定</h4>
        <a href="${pageContext.request.contextPath}/chapter12/register">1.跳转Pojo实体对象数据绑定页面</a>

        <h4>BoundComplexController 复杂对象数据绑定</h4>
        <a href="${pageContext.request.contextPath}/chapter12/gotoProductList">1.跳转集合List对象数据绑定页面</a>
        <br/>
        <a href="${pageContext.request.contextPath}/chapter12/gotoProductJson">2.跳转Json数据(列表)绑定页面</a>

        <h4>BoundComplexController Pojo复杂对象数据绑定</h4>
        <a href="${pageContext.request.contextPath}/chapter12/gotoOrderList">1.跳转Pojo复杂对象List数据绑定页面</a>
        <br/>
        <a href="${pageContext.request.contextPath}/chapter12/gotoOrderInfo">2.跳转Pojo复杂对象Map数据绑定页面</a>

        <h4>BoundSimpleController 数据回写</h4>
        <a href="${pageContext.request.contextPath}/chapter12/showModelAndView">1.携带数据ModelAndView页面</a>
        <br/>
        <a href="${pageContext.request.contextPath}/chapter12/gotoDataShowJson">2.Json数据回写页面</a>
    </body>
</html>
```

c.配置成功页 success.jsp，添加返回首页链接。



第 5 步：需求 1：简单数据绑定开发

a. 创建处理器 Controller 类（简单绑定类）

在项目 src/main/java 目录下创建代码包：**com.xhxg.controller.chapter12**，创建 **BoundSimpleController** 类，使用**@Controller 注解**注册此类为处理器类。另外，还需要使用**@RequestMapping("/chapter12")注解**声明访问空间，需要实现的功能包括：**1.默认数据绑定(Spring MVC 默认内置对象)；2.简单数据类型绑定（基本数据类型）；3.RESTful 风格数据绑定**。代码如下所示：

BoundSimpleController.java

```java
package com.xhxg.controller.chapter12;
import com.xhxg.pojo.chapter12.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.ModelAndView;
import javax.servlet.http.HttpServletRequest;

no usages
@Controller
@RequestMapping("/chapter12")
public class BoundSimpleController {

    //一、简单数据绑定
    //1.默认参数绑定，Spring MVC 4个默认对象
    no usages
    @RequestMapping("/getUserId")
    public String getUserId(HttpServletRequest request, Model model) {
        String userid = request.getParameter( s: "userid");
        model.addAttribute( s: "returnMSG", o: "你好，我是Model封装的信息！");
        System.out.println("userid=" + userid);
        return "success";
    }

    //2.简单数据类型参数绑定，即形参是基本数据类型
    no usages
    @RequestMapping("/getUserNameAndId")
    public String getUserNameAndId(String username, Integer id) {
        System.out.println("username=" + username + ",id=" + id);
        return "success";
    }
```

BoundSimpleController.java

```
30          //3.RESTFul风格参数绑定
            no usages
31          @RequestMapping("/user/{name}")
32          public String getPathVariable(@PathVariable(value = "name")
33                                        String username) {
34              System.out.println("username=" + username);
35              return "success";
36          }
```

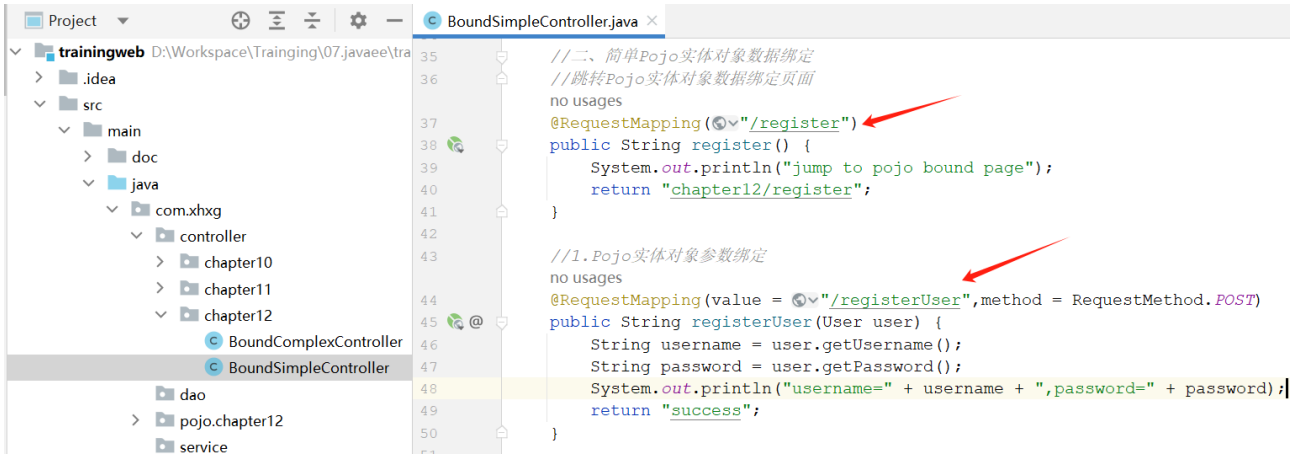
b. 以上三个方法的请求，返回结果均是 success. jsp 页面。而方法的入口路径均是在首页 index. jsp 中配置，因此，需要检查这两个文件的配置即可。即检查如下所示的文件配置

测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

第 6 步：需求 2：简单 Pojo 实体对象数据绑定开发

a. 创建处理器 Controller 类（简单 Pojo 实体对象数据绑定）

在项目 src/main/java 目录下创建代码包：com. xhxg. controller. chapter12，创建 BoundSimpleController 类（第 5 步已经创建，往里面增加方法即可），需要实现的功能包括：1. Pojo 实体对象数据绑定。代码如下所示：



b. 在目录 webapp/pages/下，新创建 chapter12 目录，然后创建 **register.jsp** 页面，内容如下所示：**<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>**

测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

第 7 步：需求 3：复杂对象数据绑定开发

a. 创建处理器 Controller 类（简单 Pojo 实体对象数据绑定）

在项目 src/main/java 目录下创建代码包：**com.xhxg.controller.chapter12**，创建 BoundComplexController 类，需要实现的功能包括：**1.集合 List 对象数据绑定、2.Json 数据绑定(列表数据)**。代码如下所示：

BoundComplexController.java

```java
package com.xhxg.controller.chapter12;
import com.xhxg.pojo.chapter12.Order;
import com.xhxg.pojo.chapter12.Product;
import com.xhxg.pojo.chapter12.User;
import org.springframework.stereotype.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.RequestParam;
import java.util.HashMap;
import java.util.List;
import java.util.Set;
no usages
@Controller
@RequestMapping("/chapter12")
public class BoundComplexController {
    //三、复杂对象数据绑定
    no usages
    @RequestMapping("/gotoProductList")//跳转集合List对象数据绑定页面
    public String gotoProductList() {
        System.out.println("jump to list bound page");
        return "chapter12/productList";
    }
    //1.集合List对象数据绑定，必须要在形参上使用@RequestParam注解上才能使用List作为参数
    no usages
    @RequestMapping("/getProductList")
    public String getProductList(@RequestParam("proIds") List<String> proIds) {
        for (String proId : proIds) {
            System.out.println("获取到了Id为" + proId + "的商品");
        }
        return "success";
    }
```

```java
//跳转Json对象数据绑定页面
no usages
@RequestMapping("/gotoProductJson")
public String gotoProductJson() {
    System.out.println("jump to pojo json bound page");
    return "chapter12/productJson";
}
//2. Json数据绑定(列表数据)
//@RequestBody注解结合Jackson的JSON格式转换器，实现将JSON格式数据绑定到方法形参中。
no usages
@RequestMapping("/getProductJsonList")
public String getProductJsonList(@RequestBody List<Product> products) {
    for (Product product : products) {
        String proId = product.getProId();
        String proName = product.getProName();
        System.out.println("获取到了Id为" + proId + "名称为" + proName + "的商品");
    }
    return "success";
}
```

b. 在目录 webapp/pages/下，新创建 chapter12 目录，然后创建 productList.jsp 页面，内容如下所示：<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

c. 在目录 webapp/pages/下，新创建 chapter12 目录，然后创建 productJson.jsp 页面，内容如下所示：<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

第 8 步：需求 4：Pojo 复杂对象数据绑定开发

a. 创建处理器 Controller 类（简单 Pojo 实体对象数据绑定）

在项目 src/main/java 目录下创建代码包：com.xhxg.controller.chapter12，创建 BoundComplexController 类（第 7 步已经创建，往里面增加方法即可），需要实现的功能包括：1.Pojo 复杂对象 List 数据绑定、2.Pojo 复杂对象 Map 数据绑定。代码如下所示：

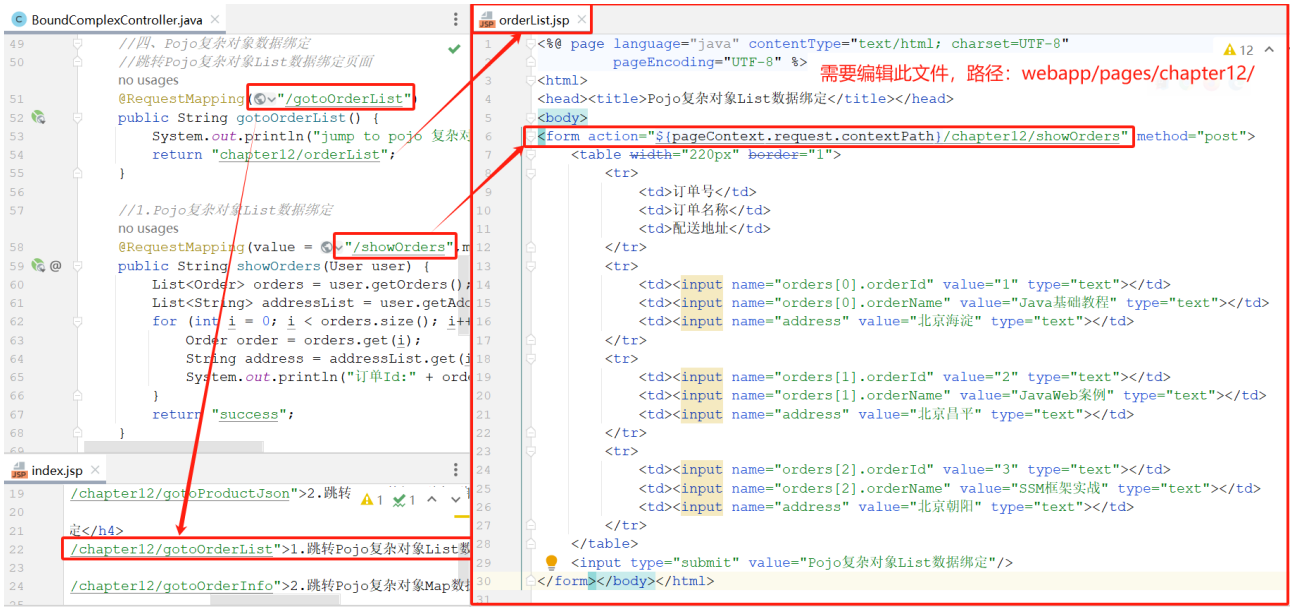
BoundComplexController.java

```
49      //四、Pojo复杂对象数据绑定
50      //跳转Pojo复杂对象List数据绑定页面
        no usages
51      @RequestMapping("/gotoOrderList")
52      public String gotoOrderList() {
53          System.out.println("jump to pojo 复杂对象List page");
54          return "chapter12/orderList";
55      }
56
57      //1.Pojo复杂对象List数据绑定
        no usages
58      @RequestMapping(value = "/showOrders",method = RequestMethod.POST)
59      public String showOrders(User user) {
60          List<Order> orders = user.getOrders();
61          List<String> addressList = user.getAddress();
62          for (int i = 0; i < orders.size(); i++) {
63              Order order = orders.get(i);
64              String address = addressList.get(i);
65              System.out.println("订单Id:" + order.getOrderId()+ ", 配送地址: " + address);
66          }
67          return "success";
68      }
```

BoundComplexController.java

```
69
70      //跳转Pojo复杂对象Map数据绑定页面
        no usages
71      @RequestMapping("/gotoOrderInfo")
72      public String gotoOrderInfo() {
73          System.out.println("jump to pojo 复杂对象Map page");
74          return "chapter12/orderInfo";
75      }
76
77      //2.Pojo复杂对象Map数据绑定
        no usages
78      @RequestMapping("/getOrderInfo")
79      public String getOrderInfo(Order order) {
80          String orderId = order.getOrderId();
81          //获取商品信息
82          HashMap<String, Product> orderInfo = order.getProductInfo();
83          Set<String> keys = orderInfo.keySet();
84          System.out.println("订单id:" + orderId +", 订单商品信息:");
85          for (String key : keys) {
86              Product product = orderInfo.get(key);
87              String proId = product.getProId();
88              String proName = product.getProName();
89              System.out.println(key + "类~" + "商品id:" + proId + ",商品名称: " + proName);
90          }
91          return "success";
92      }
93  }
```

b. 在目录 webapp/pages/下，新创建 chapter12 目录，然后创建 orderList. jsp 页面，内容如下所示：



测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

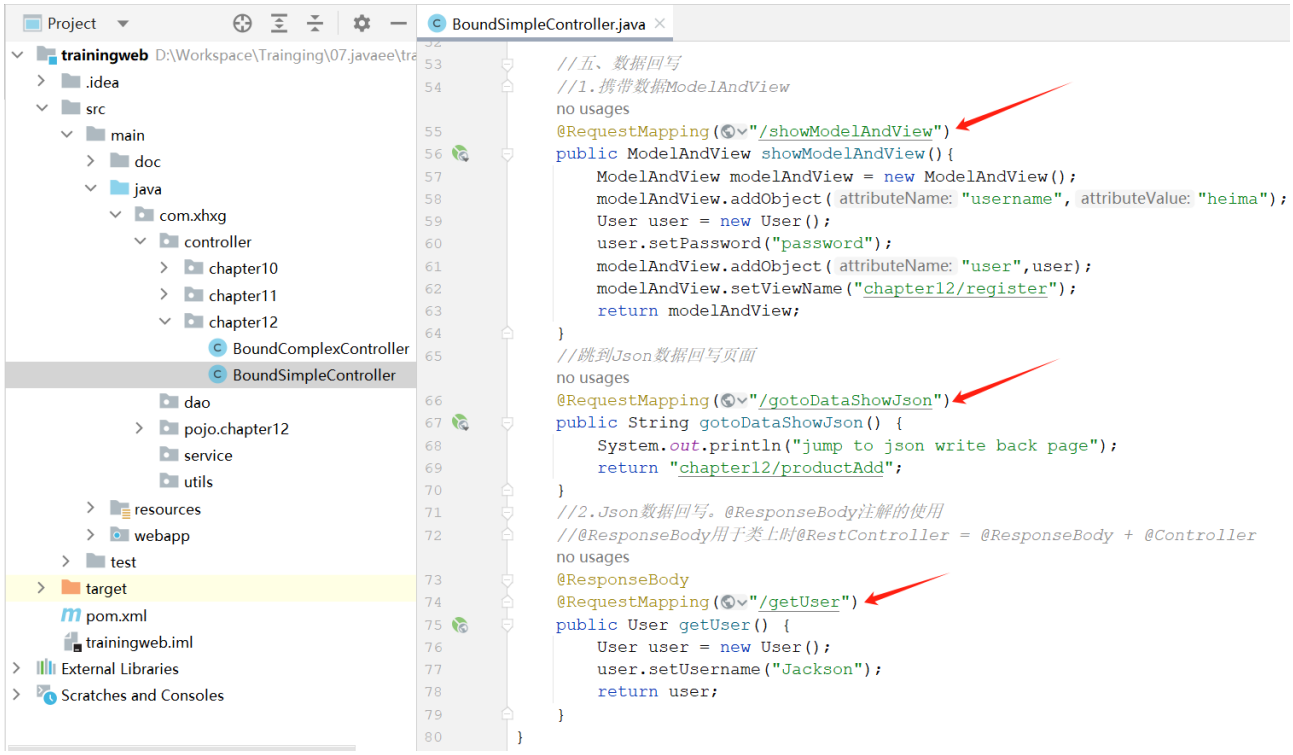
c. 在目录 webapp/pages/下，新创建 chapter12 目录，然后创建 orderInfo. jsp 页面，内容如下所示：

测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

第 9 步：需求 5：数据回写开发

a. 创建处理器 Controller 类（简单 Pojo 实体对象数据绑定）

在项目 src/main/java 目录下创建代码包：com. xhxg. controller. chapter12，创建 BoundSimpleController 类（第 5 步已经创建，往里面增加方法即可），需要实现的功能包括：1. 携带数据 ModelAndView 回写页面、2. Json 数据回写页面。代码如下所示：



b. 在目录 webapp/pages/下，新创建 chapter12 目录，然后检查 register. jsp 页面（此页面第 6 步已经创建过了），内容如下所示：

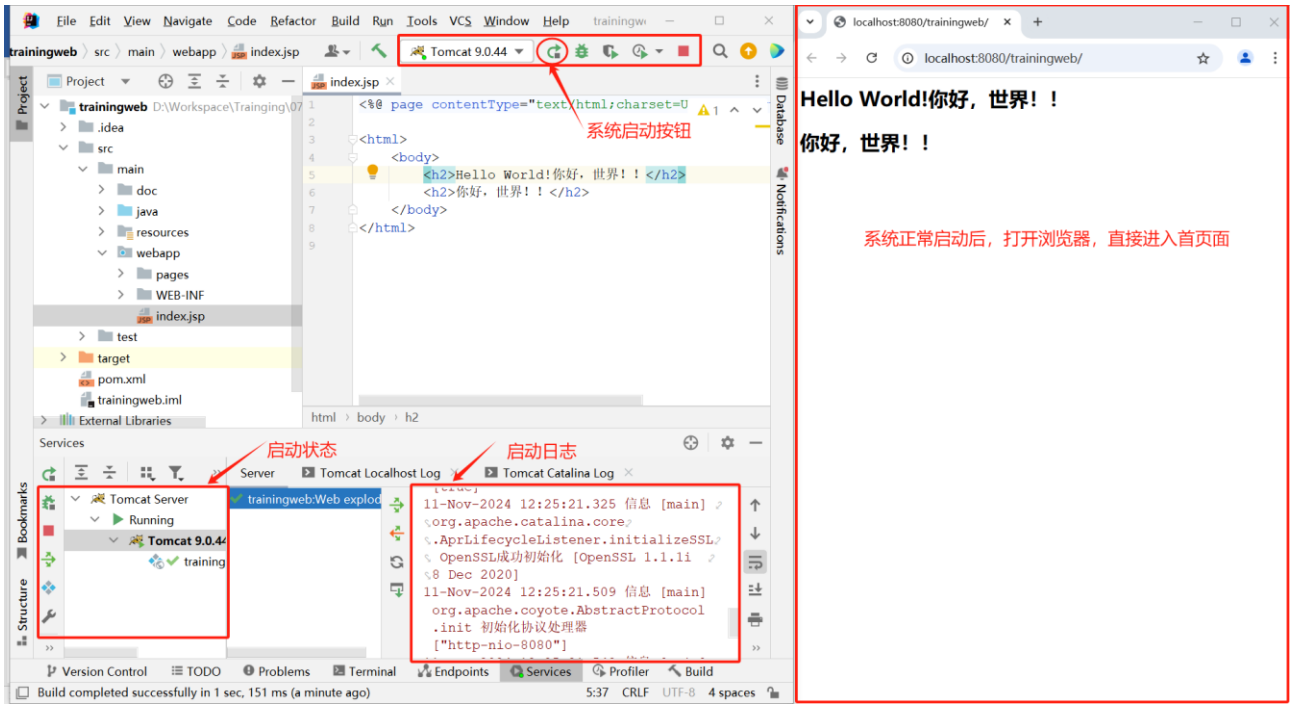
测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

c. 在目录 webapp/pages/下，新创建 chapter12 目录，然后检查 productAdd. jsp 页面，内容如下所示：

测试时，注意查看每个方法的控制台输出，检查是否能正常输出请求的参数。

四、 系统运行

【第三章节】第 3 小节中已经配置好了 Tomcat 服务器（如没有配置好，请继续按第 3 小节的步骤配置好 Tomcat 服务器），所以我们直接启动此服务器即完成系统运行。



五、 系统测试

1、**运行系统后，打开首页 index.jsp，然后逐个点击首页中配置的链接即可测试。要注意页面的变化和 Tomcat 控制台的输入日志。**


localhost:8080/trainingweb/
Hello World首页
BoundSimpleController简单数据绑定
1.默认数据绑定(Spring MVC默认内置对象)
2.简单数据类型绑定(基本数据类型)
3.RESTful风格数据绑定
BoundSimpleController简单Pojo实体对象数据绑定
1.跳转Pojo实体对象数据绑定页面
BoundComplexController复杂对象数据绑定
1.跳转集合List对象数据绑定页面
2.跳转Json数据(列表)绑定页面
BoundComplexController Pojo复杂对象数据绑定
1.跳转Pojo复杂对象List数据绑定页面
2.跳转Pojo复杂对象Map数据绑定页面
BoundSimpleController数据回写
1.携带数据ModelAndView页面
2.Json数据回写页面

六、 常见问题

待定。。。