

实验 7 基于注解测试 Bean 的生命周期实验程序

一、 基于注解测试 Bean 的生命周期需求描述

编程：使用注解的方式装配 Bean，要求建立包名：controller 层、service 层、dao 层。

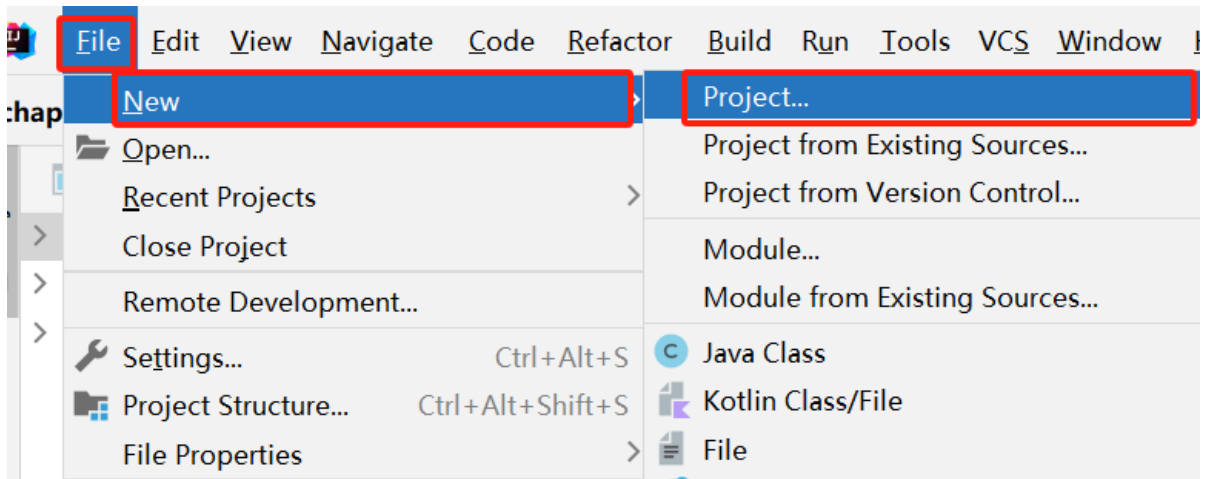
功能：实现保存 save() 功能，要求保存用户 (User) 信息。**调用链：**controller 层是调用入口，然后调用 service 层的保存方法，最后调用 dao 层保存方法，在控制台输出：“用户保存成功!”

二、 开发步骤

第 1 步：创建 Maven 工程

方式一：直接创建新的 Maven 工程

按如下示例的步骤，进行工程创建（基于 IDEA 2022）



New Project

×

Generators

Maven Archetype

Jakarta EE

Spring Initializr

JavaFX

Quarkus

Micronaut

Ktor

Kotlin Multiplatform

Compose Multiplatform

HTML

React

Express

Angular CLI

IDE Plugin

Android

Vue.js

Vite

Name:

training

项目名称

Location:

D:\Workspace\Trainging\07.javaee\training

项目存放的目录

Project will be created in: D:\Workspace\Trainging\07.javaee\chapter01\demo

☐ Create Git repository

Template:

Library

Dependencies only

Application server:

Tomcat 9.0.44

New...

Language:

Java

Kotlin

Groovy

Build system:

Maven

Gradle

Group:

com.xhxc

项目源代码包命名路径

Artifact:

training

与项目名称保持一致即可

JDK:

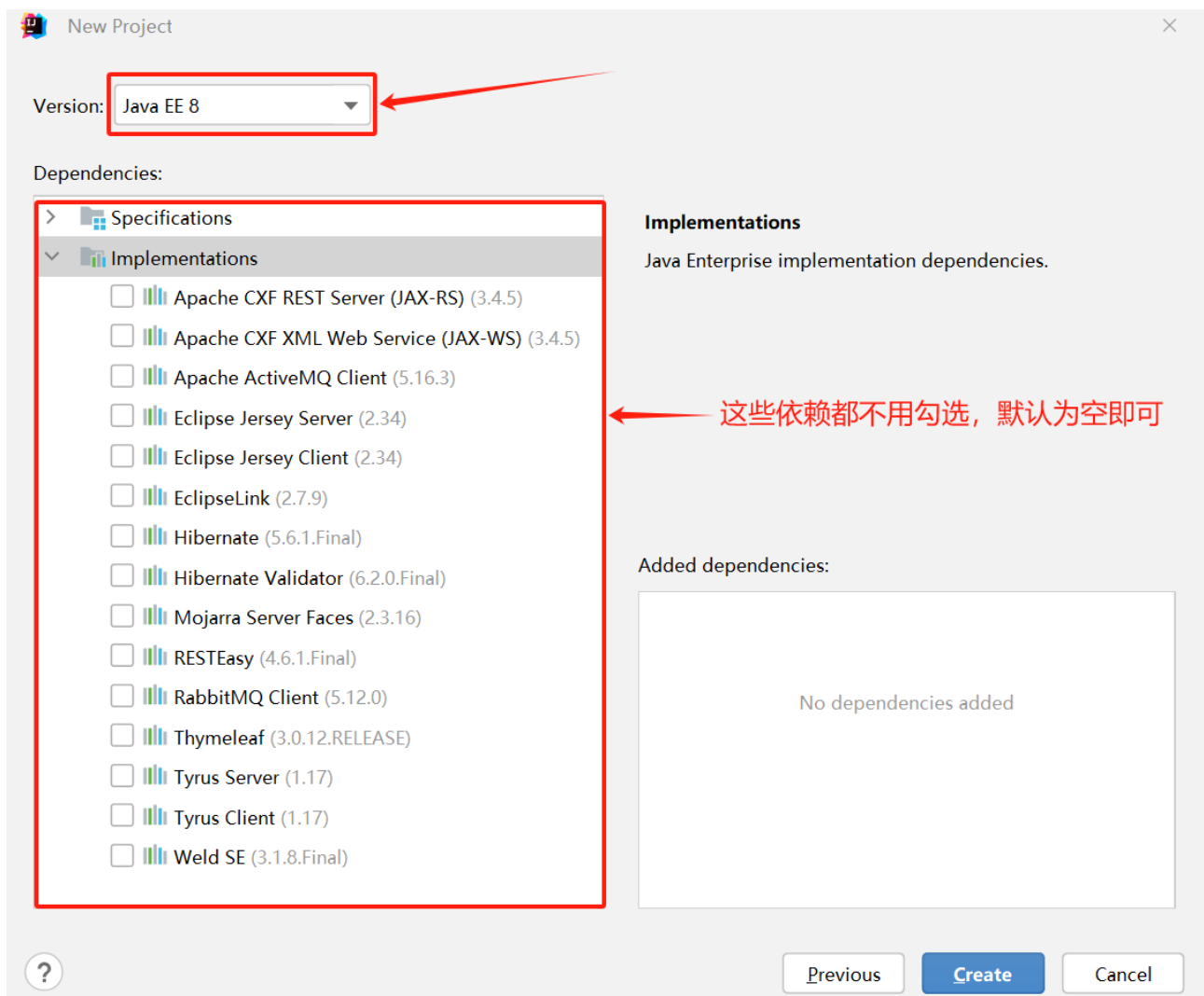
1.8 Oracle OpenJDK version 1.8.0_191

JDK版本

?

Next

Cancel




最后一步，点击【Create】即可完成工程创建。

方式二：导入已存在的 Maven 工程（推荐）

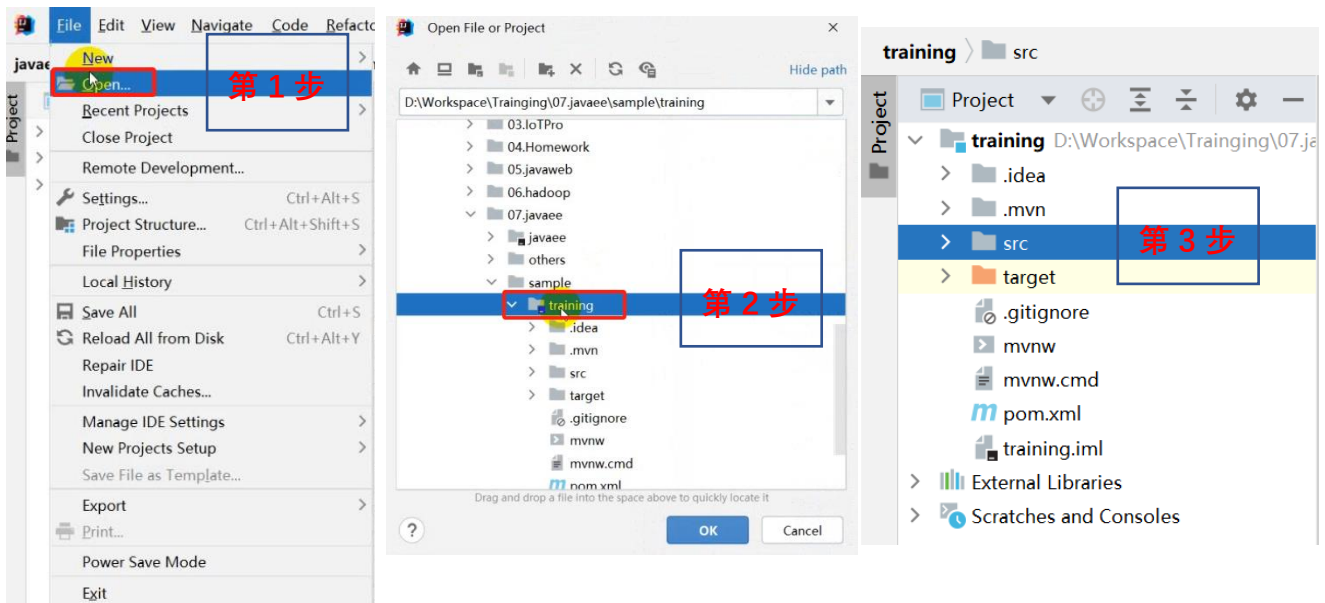
Step1，下载并解压 training 工程。从【学习通】→【章节】→【第七章…】→【上机实验 7…】，找到 training.zip 压缩包，下载到本地，并解压成 training 目录。

Step2，导入 training 工程。打开 IDEA 工具，点击【File】→【Open…】→选中上一步解压的 training 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。

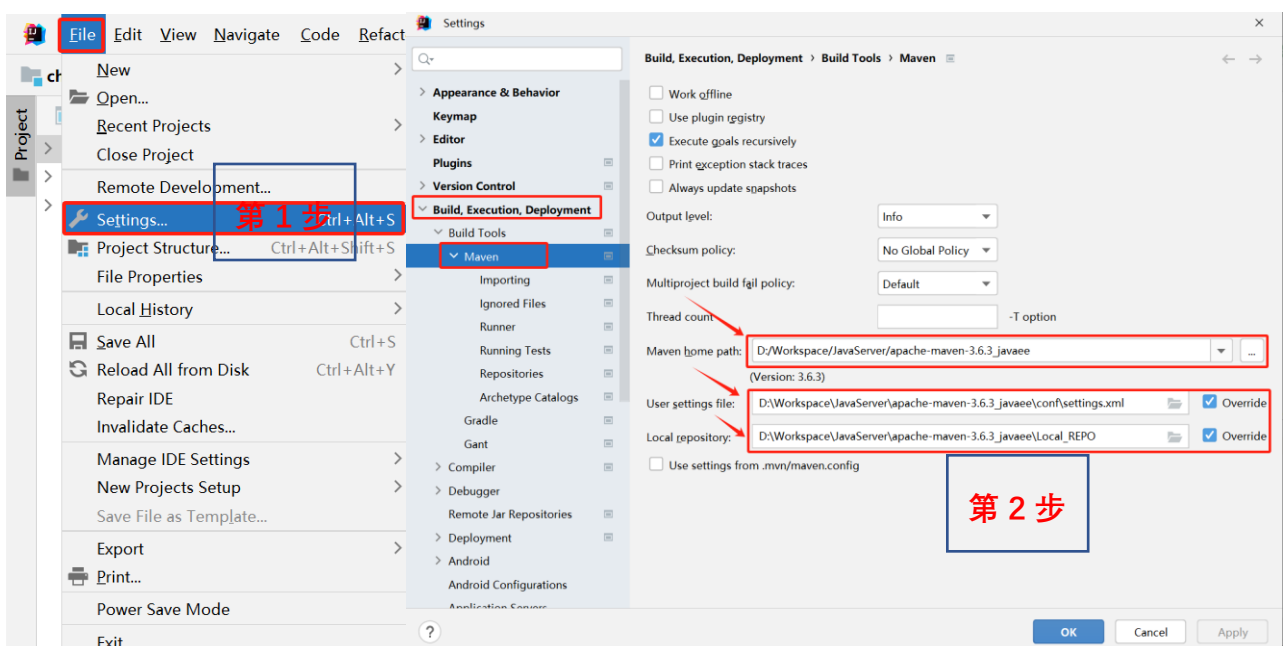
Tips：如果前几个章节已经下载并导入了此 training 工程，那么下载本章节最新工程后，

可以将新工程 **training** 文件夹内的所有文件全部复制，然后覆盖更新替换掉原有已经存在的 **training** 工程文件夹中的文件。这样，保证工程的内容是最新的，跟课程同步的。这样处理后，也就不需要再通过 IDEA 导入新工程了，直接使用旧 **training** 工程即可。



第 2 步：配置本地 Maven 路径

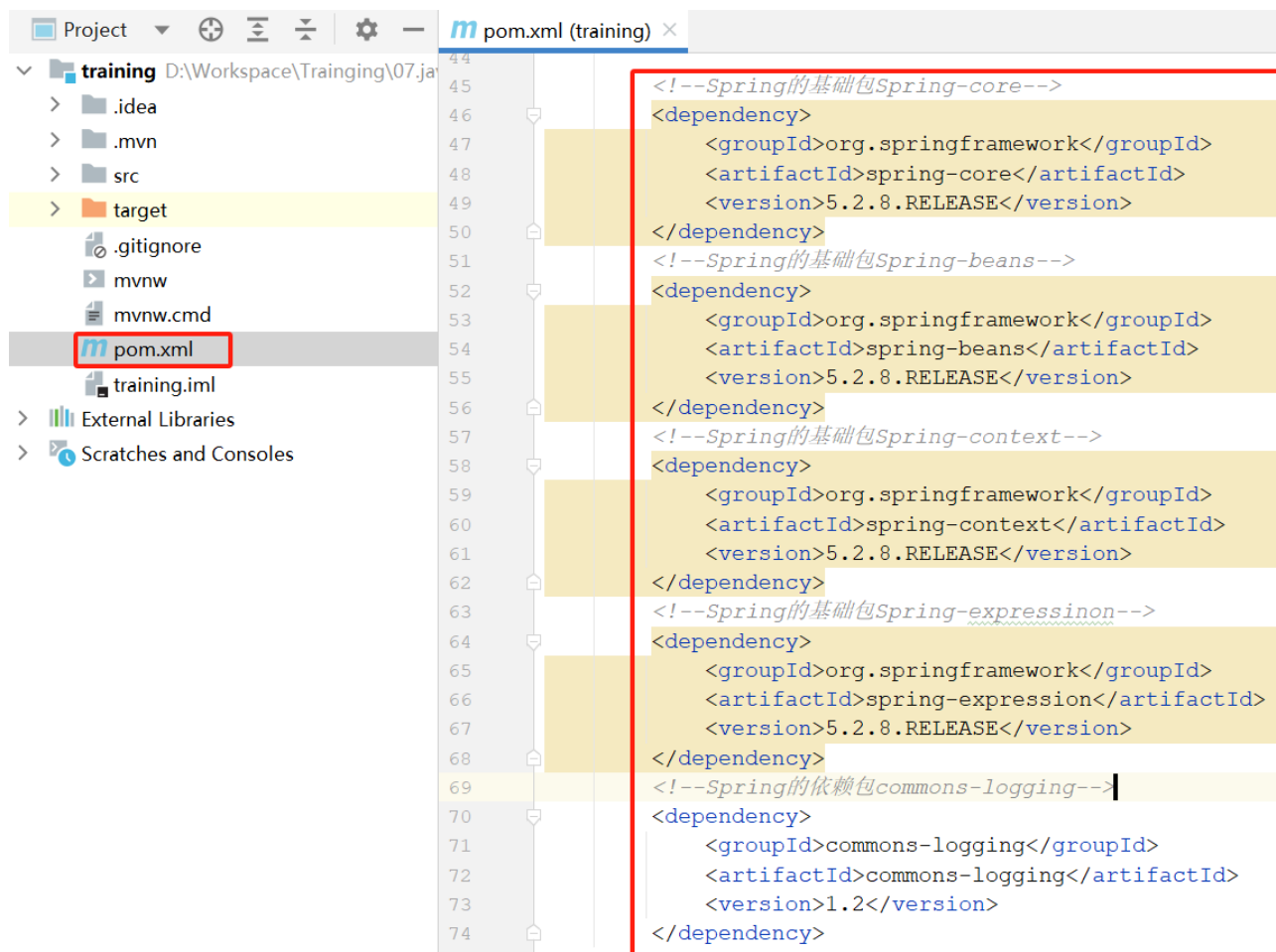
选择【File】→【Setting...】然后找到菜单项【Build, Execution, Deployment】→【Build Tools】→【Maven】，按如下图所示，将你的本地 Maven 路径和配置文件设置好即可。



第 3 步：程序代码开发

1、配置 pom.xml

由于本章是学习 Spring 框架，因此，需要先下载 Spring 的相关依赖，才能正常运行实验代码。pom.xml 文件路径和需要添加的依赖如下截图：



需要添加的依赖原配置为：

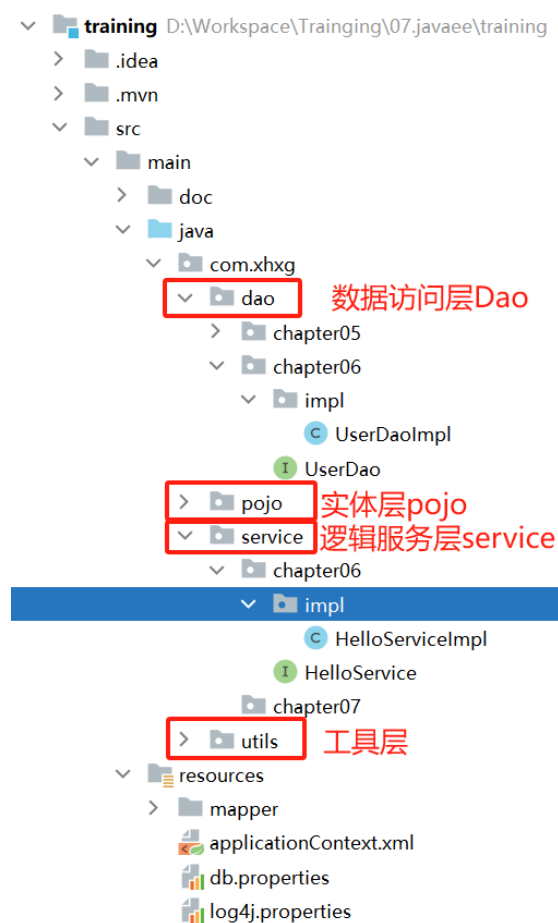
```
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-core</artifactId>
  <version>5.2.8.RELEASE</version>
</dependency>
<dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-beans</artifactId>
  <version>5.2.8.RELEASE</version>
</dependency>
<dependency>
```

```

    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.8.RELEASE</version>
</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-expression</artifactId>
    <version>5.2.8.RELEASE</version>
</dependency>
<dependency>
    <groupId>commons-logging</groupId>
    <artifactId>commons-logging</artifactId>
    <version>1.2</version>
</dependency>

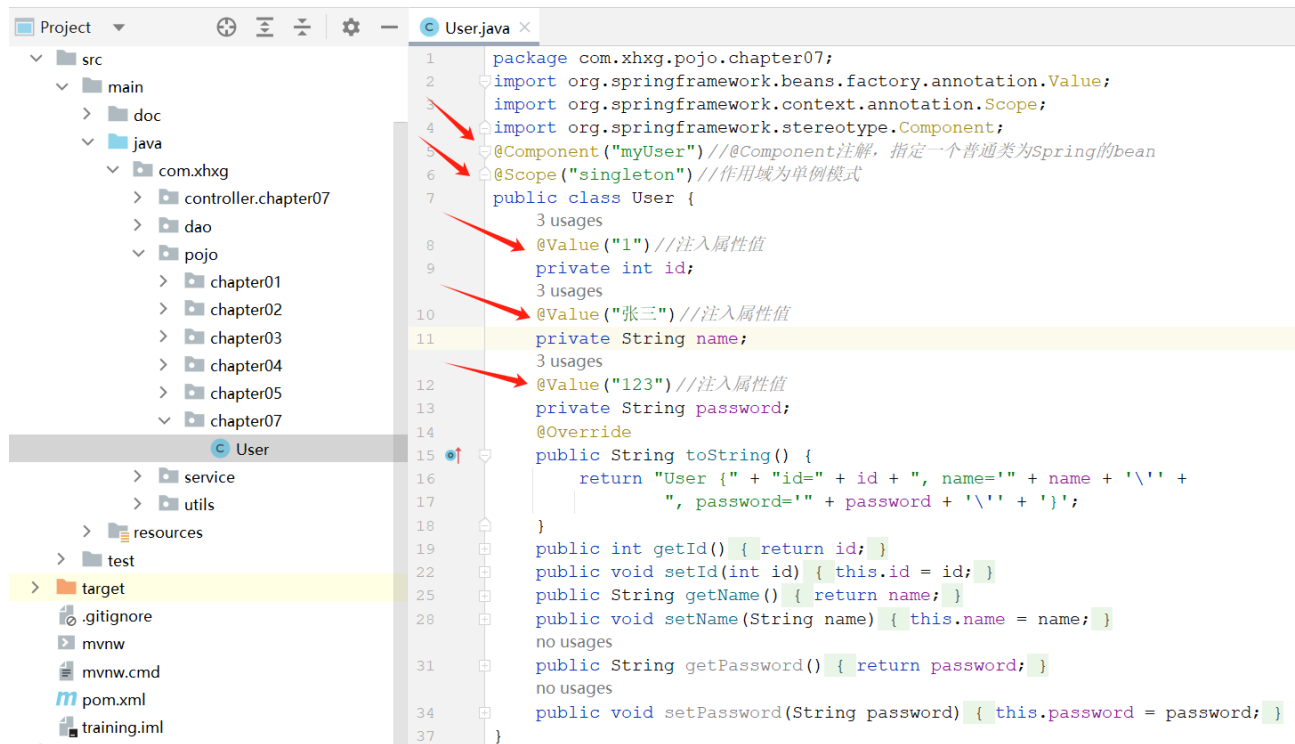
```

知识补充：工程项目代码分层管理规则，一般地项目源码代码包采用分层结构管理，包括 dao 层表示数据访问对象，存放有关对数据库及数据处理的代码；service 层表示逻辑服务层，存放有关业务处理，核心算法的代码。Service 层会调用 dao 层的接口访问数据；pojo 实体对象层，存放有关项目实体代码；utils 项目工具层，存放有关项目的一些公共工具和公共方法等。还有其他根据项目的不同再自行定义。在本实验中需要加入以上的代码包，以完成开发。代码结构如下截图：



2、创建实体类 User

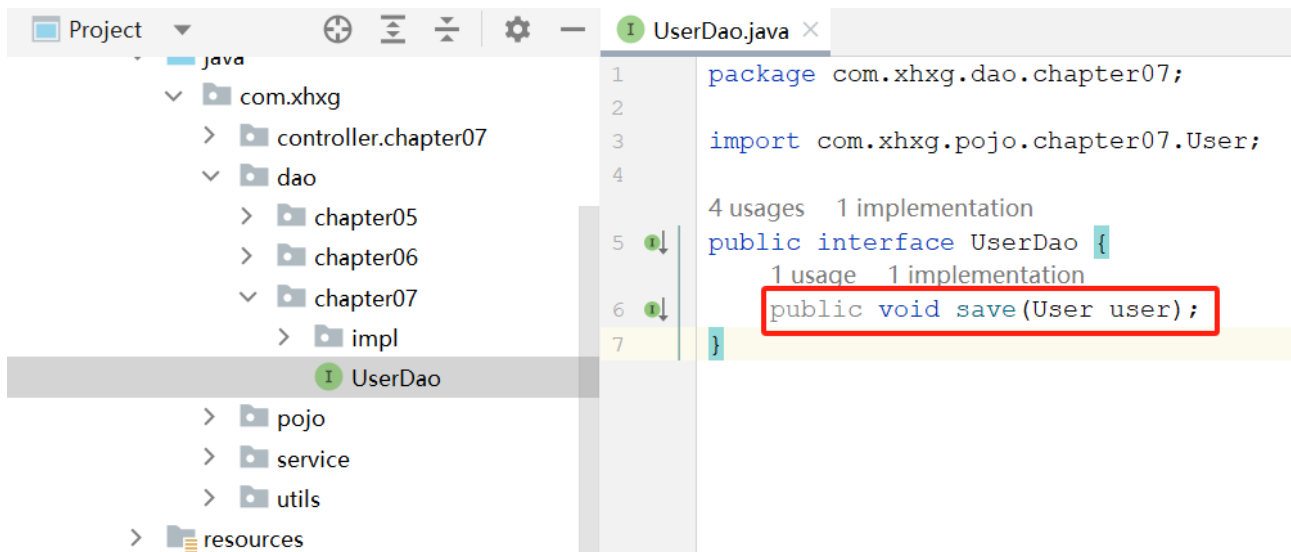
创建代码包为：`com.xhxc.pojo.chapter07`，创建 `User` 实体类。需要使用注解 `@Component("myUser")` 声明为 Spring Bean。另外可以使用注解 `@Scope("singleton")` 声明 Bean 的作用范围，同时使用注解 `@Value("")` 为每个属性注入初始值。代码如下截图所示：



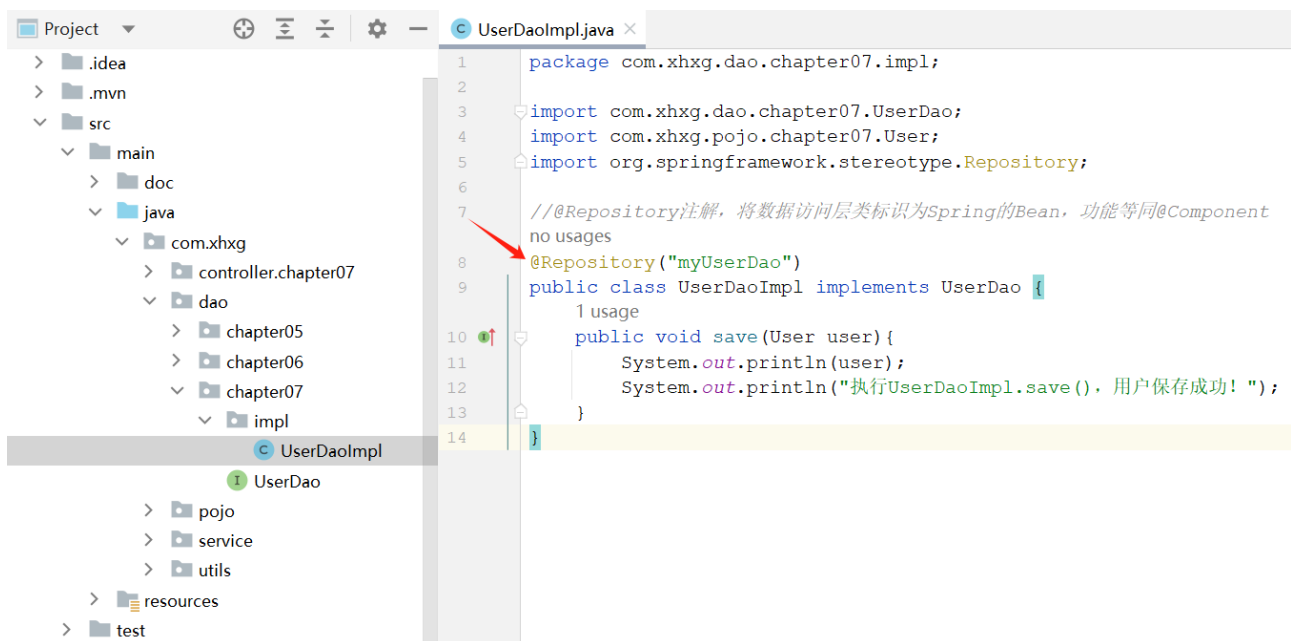
注意：`@Component("myUser")`注解，括号中是自定义的 Spring Bean 名称，这个名称可以省略，当省略时，默认以类的名称（首字母小写）作为 Spring Bean 名称。本实验中，定义了 `User` 实体类，为了区分有其他与此有雷同的实体 Bean，所以自定义 Bean 名称为 **`myUser`**

3、创建 Dao 接口及其实现类

创建代码包为：`com.xhxc.dao.chapter07`，创建 `UserDao` 接口类，定义 `save()` 接口方法，代码如下截图所示：



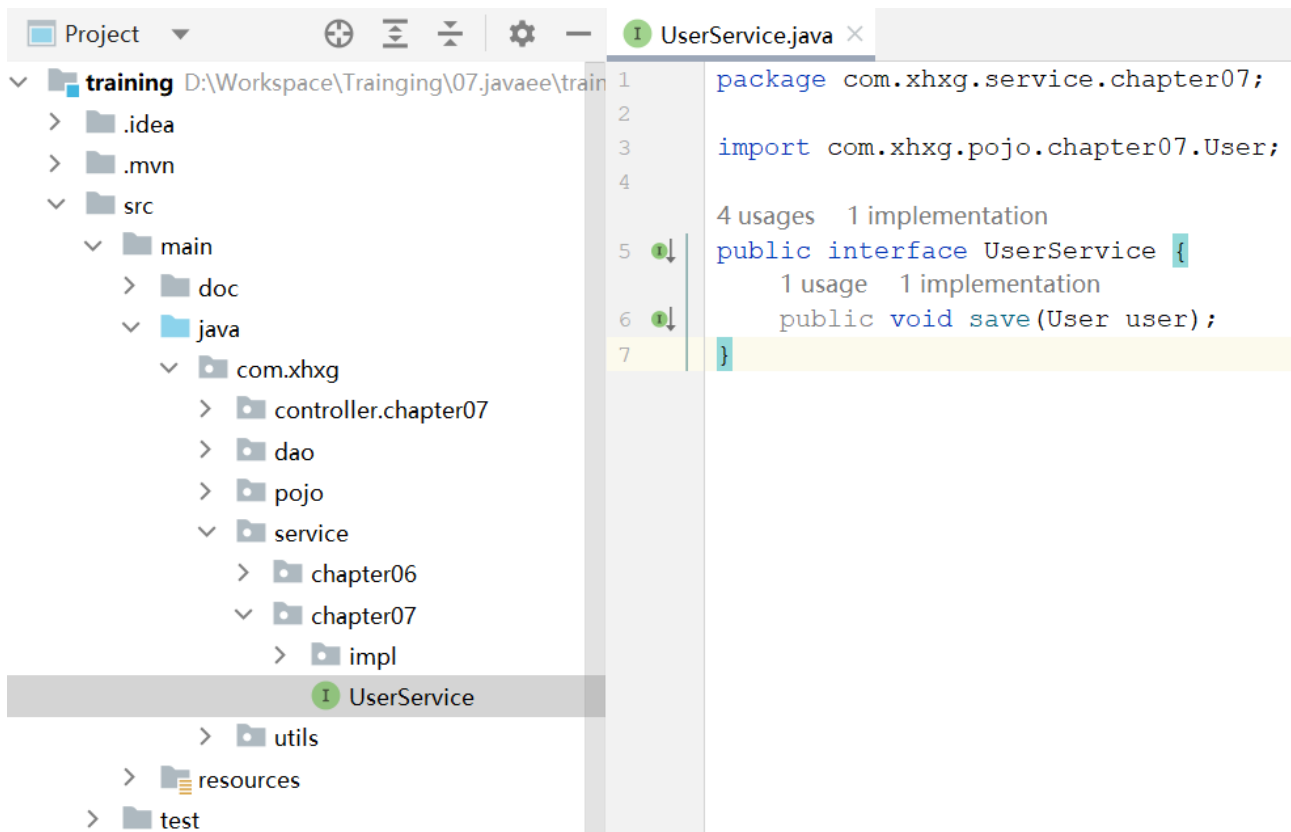
继续创建 UserDao 接口类的实现类 UserDaoImpl，需要创建代码包为：
com.xhcg.dao.chapter07.impl，以存放接口实现类代码。代码如下截图所示：



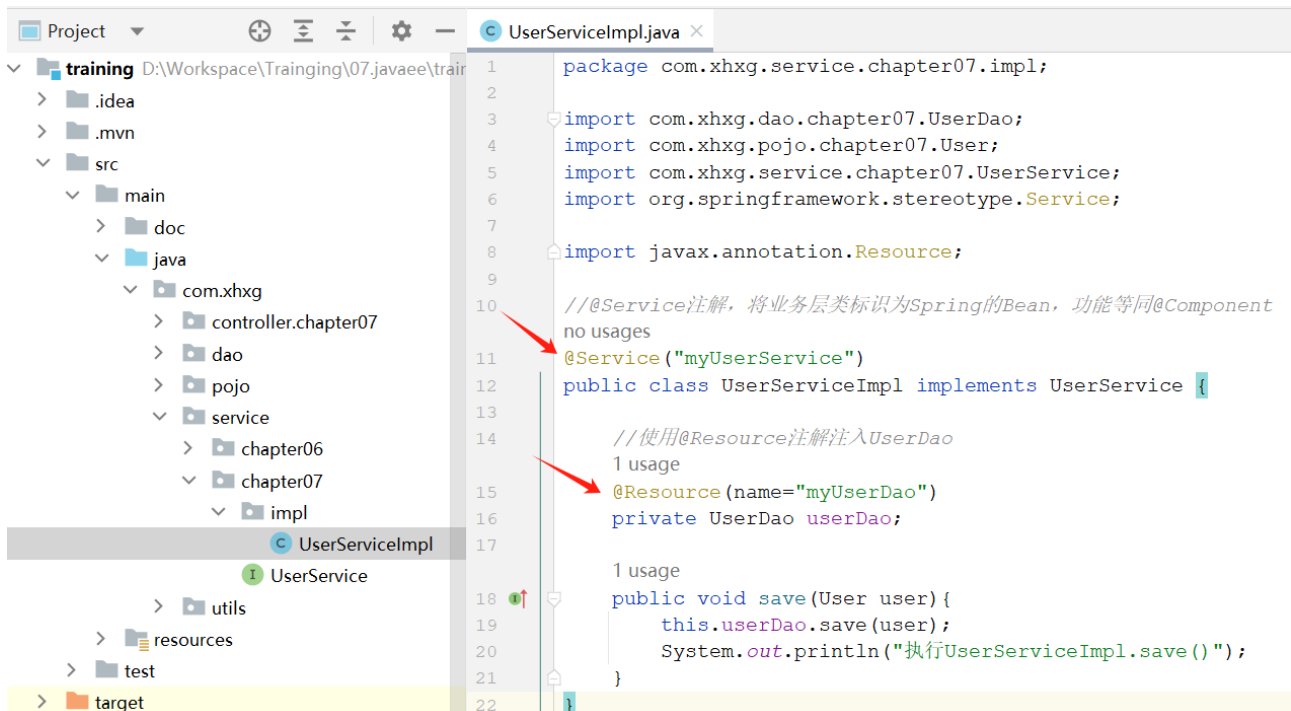
注意：数据接口使用了注解@Repository("myUserDao")，其中括号中的 myUserDao 名称为自定义 Bean 名称，如不自定义，默认以类的名称（首字母小写）作为 Spring Bean 名称

4、创建 Service 接口及其实现类

创建代码包为：com.xhcg.service.chapter07，创建 UserService 接口类，定义 save() 接口方法，代码如下截图所示：



继续创建 UserService 接口类的实现类 UserServiceImpl，需要创建代码包为：com.xhcg.service.chapter07.impl，以存放接口实现类代码。代码如下截图所示：

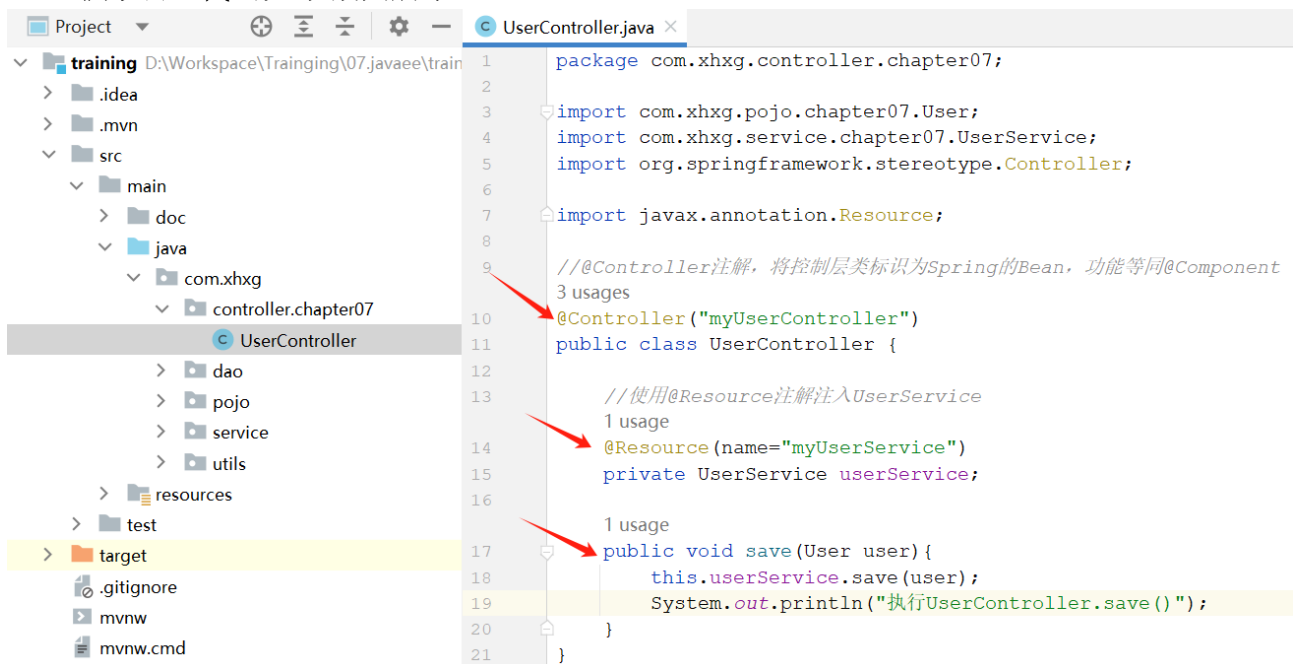


注意：业务层接口使用了注解 `@Service("myUserService")`，其中括号中的 `myUserService` 名称为自定义 Bean 名称，如不自定义，默认以类的名称（首字母小写）作为 Spring Bean 名称。另

外，在注入数据层接口时使用注解@Resource(name="myUserDao")，指向 Dao 数据层的接口 Bean，名称就是数据层中声明的 Bean 名称：myUserDao

5、创建 controller 层实现类

创建代码包为：com.xhcg.controller.chapter07，创建 UserController 类，定义 save() 方法，代码如下截图所示：



注意：controller 层实现类使用了注解@Controller("myUserController")，其中括号中的 myUserController 名称为自定义 Bean 名称，如不自定义，默认以类的名称（首字母小写）作为 Spring Bean 名称。另外，在注入业务层接口时使用注解@Resource(name="myUserService")，指向 service 业务层的接口 Bean，名称就是 Service 业务层中声明的 Bean 名称：myUserService

6、创建 Spring 配置文件 applicationContext.xml

在 src/main/resources 目录下创建一个 applicationContext.xml 文件，内容如下：

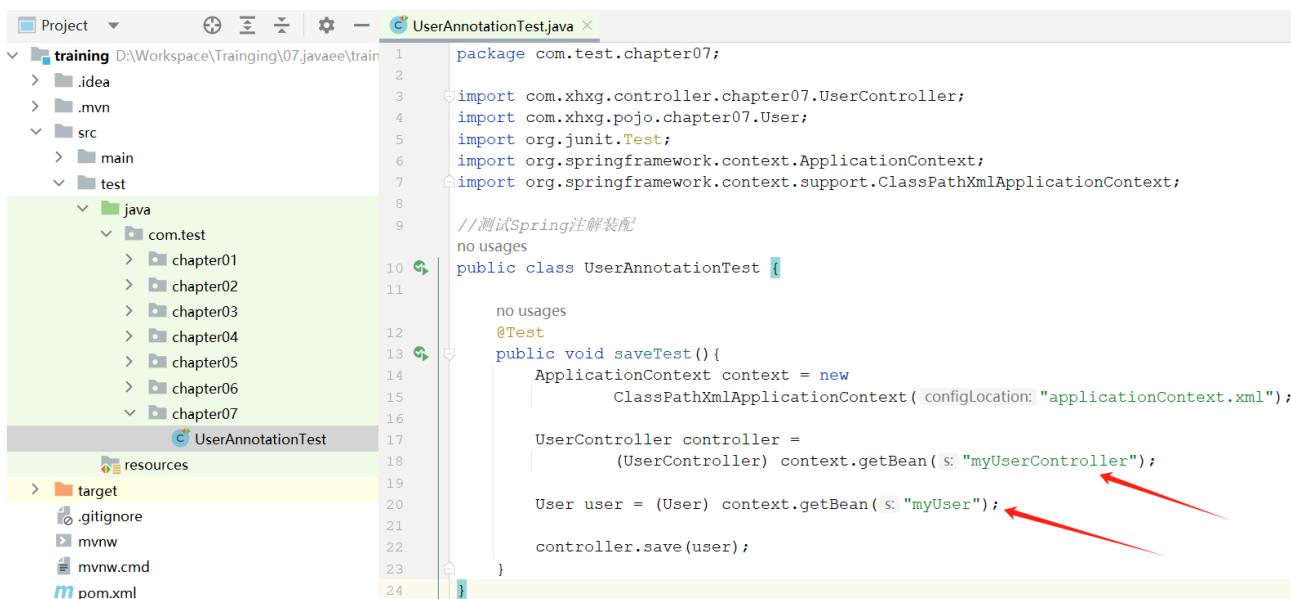


在该 Spring 配置文件中，需要在 beans 节点定义中加入 Context 的约束声明，如上图。另外，开启自动扫描功能`<context:component-scan base-package="com.xhxxg" />`，开启自动扫描后，就可以配合注解的声明，实例化 Bean 对象。

7、编写测试类

以上代码和配置创建好后，就可以编写测试类进行验证测试。

我们在 src/test/java 目录下，建议测试包，路径为：com.test.chapter07，然后创建测试类 UserTest，定义测试方法 saveTest()，代码如下：



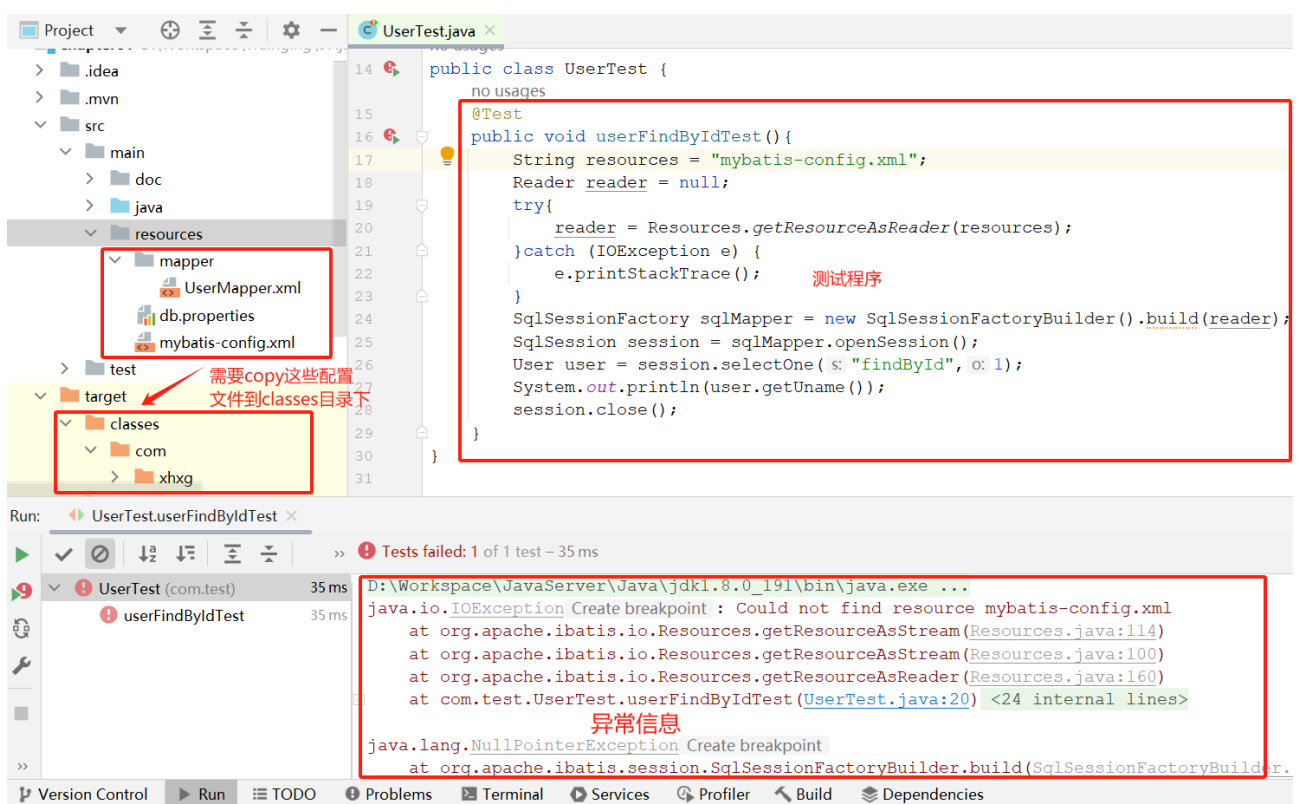
在本单元测试用例中，通过 Spring API 的 **ClassPathXmlApplicationContext** 接口 API 读取 Spring 配置文件 applicationContext.xml，获得 Spring 应用上下文，通过应用上下文就可以获取已被 IoC 容器实例化的 Bean。本实验中，需要获得两个实例化的 Bean，分别是 myUserController 和 myUser 这两个 Bean 名称。通过前者获得调用方法的入口，而后者则是获得已经初始化了的 User 实体的属性值。

至此，本实验圆满成功。

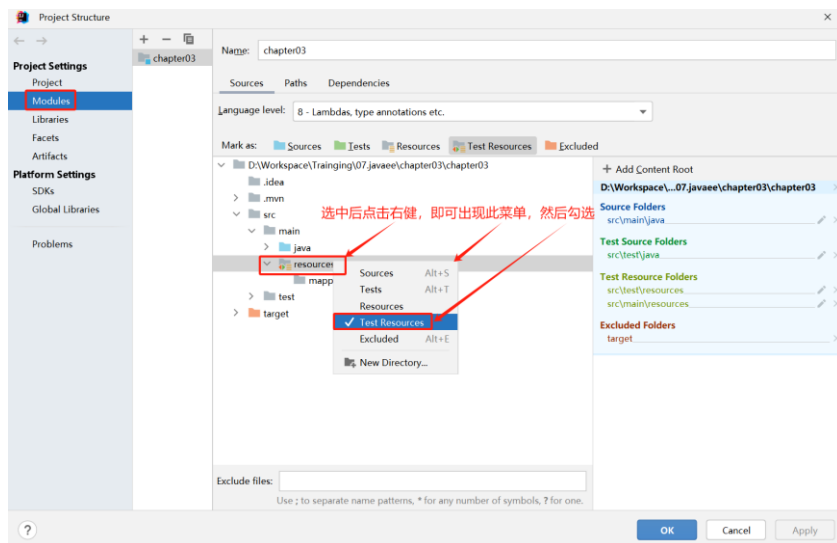
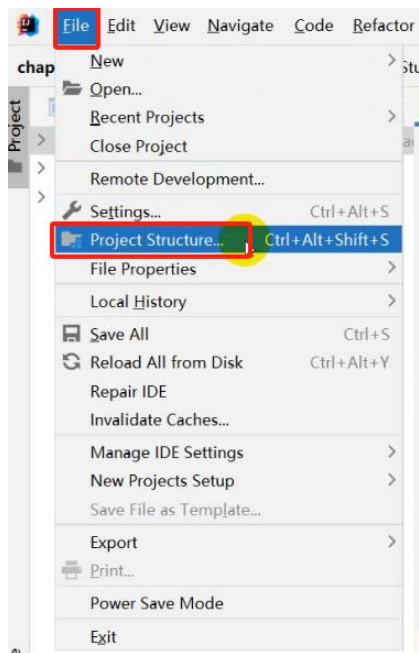
三、常见问题

有时候 Maven 不能正确的复制配置文件到 target 目录中，此时运行测试程序就会报如下异常：Could not find resource mybatis-config.xml

解决办法 1，就是手动将 src/main/resources/目录下的所有配置文件（包含目录）复制到 target/classes/目录下即可解决。（初学者建议采用此方法，增加你对项目工程的熟悉度）



解决办法 2，让 IDEA 自动编译复制，步骤如下截图。配置好后，IDEA 编译时会自动复制 resource 目录中配置文件到编译后的目录，就不需要再手工复制了。



注：若 Test Resources 不生效，可以换成勾选 Resources