实验 4 商品类别查询系统实验程序

一、商品类别查询系统需求描述

需求: 现有一个商品表t_product和一个商品类别表t_category, 其中, 商品类别表t_category和商品表t_product是一对多的关系。

商品表 product 的详情如下表所示。

•	Produce 1111 1124 1 - 100//1/1/10				
	商品编号	商品名称	商品详情	商品类别	
	(id)	(goodsname)	(price)	(typeid)	
	1	电视机	5000	1	
	2	冰箱	4000	2	
	3	空调	3000	2	
	4	洗衣机	2000	2	

商品类别表 category 详情如下表所示。

商品类别编号(id)	商品类别名称 (typename)	
1	黑色家电	
2	白色家电	

具体实现要求:

根据表1和表2在数据库分别创建一个商品表product和一个商品类别表category。

通过MyBatis查询商品类别为白色家电的商品所有信息。

二、开发步骤

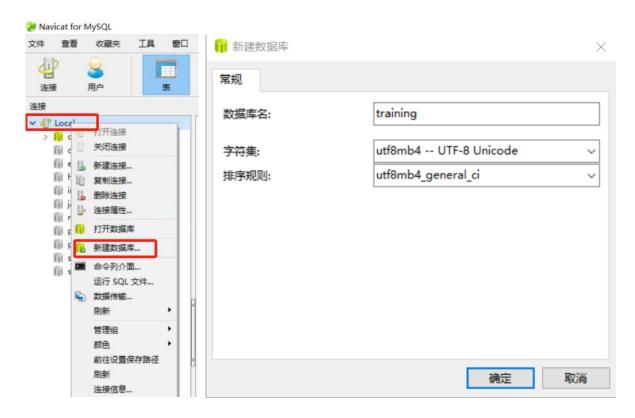
第1步:数据库开发

先使用 MySQL 数据库创建一个 training 数据库,然后创建一个商品表 t_product 和商品类别表 t category, 方法如下:

通过使用 Navicat 工具,**连接上你安装的 MySQL 数据库**,输入用户名和密码进行**登录**。假设建立的 MySQL **连接名为 Local**,数据库登录用**户名和密码为: root/123456。**

登录后,手动创建数据库,需要创建的数据库名为: training,如图: 选中 Local 连接名,

右键选择【新建数据库】



数据库创建好后,创建**商品表 t_product** 和**商品类别表 t_category**,表结构如下。并按要求加入测试数据。

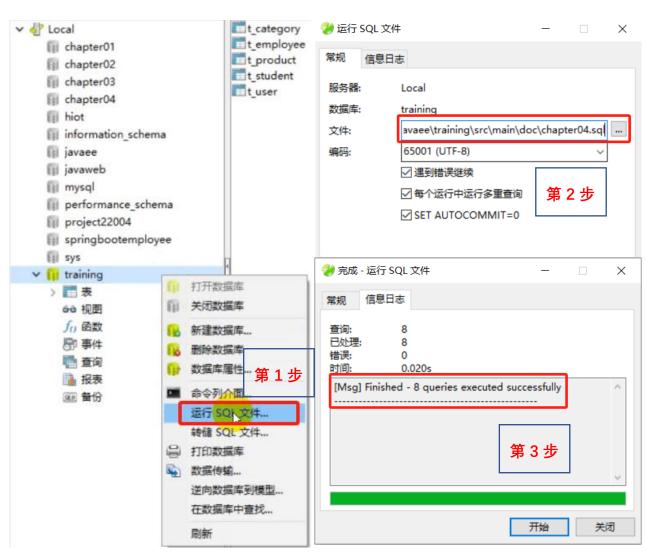
```
USE training;
     CREATE TABLE category (
           id int (32) PRIMARY KEY AUTO INCREMENT,
 4
           typename varchar (40)
      (a);
 6
       INSERT INTO category VALUES (1, '黑色家电');
       INSERT INTO category VALUES (2, '白色家电');
 9
      CREATE TABLE product (
10
           id int(32) PRIMARY KEY AUTO INCREMENT,
           goodsname varchar (40),
           price DOUBLE,
13
14
           category_id int(32) NOT NULL,
           FOREIGN KEY (category id) REFERENCES category (id)
15
       );
16
       INSERT INTO product VALUES (1, '电视机', 5000,1);
17
       INSERT INTO product VALUES (2, '冰箱', 4000,2);
18
       INSERT INTO product VALUES (3, '空调', 5000,2);
19
       INSERT INTO product VALUES (4, '洗衣机', 2000,2);
20
```

至此,数据库部分开发完毕。

注: 以上建表数据库脚本已放置工程的 doc 目录, 文件路径如下图所示:

```
Vigarianing D:\Workspace\Trainging\07.javaee\traini No data sources are configured to run this SQL and provide advanced code assistance.
  > idea
                                            SQL dialect is not configured. MariaDB, MySQL match best.
  > mvn
                                                  USE training;
  ∨ ■ src
    ∨ I main
                                                  CREATE TABLE t_category (
       id int(32) PRIMARY KEY AUTO_INCREMENT,
             d chapter01.sql
                                                      typename varchar(40)
             achapter02.sql
             d chapter 03.sql
                                                  INSERT INTO t_category VALUES (1, '黑色家电');
                                                  INSERT INTO t_category VALUES (2, '白色家电');
            sou chapter04.sql
       > iava
                                                  CREATE TABLE t_product (
       > resources
                                                      id int(32) PRIMARY KEY AUTO_INCREMENT,
     > test
                                                       goodsname varchar (40),
                                                   price DOUBLE,
 > 🖿 target
                                                       category_id int(32) NOT NULL,
     \rm gitignore
                                                      FOREIGN KEY(category id) REFERENCES t category(id)
     mvnw
     mvnw.cmd
                                                  INSERT INTO t_product VALUES (1, '电视机', 5000,1);
                                                  INSERT INTO t_product VALUES (2, '冰箱', 4000,2);
INSERT INTO t_product VALUES (3, '空调', 5000,2);
     m pom.xml
     🚛 training.iml
                                                  INSERT INTO t_product VALUES (4, '洗衣机', 2000,2);
> III External Libraries
```

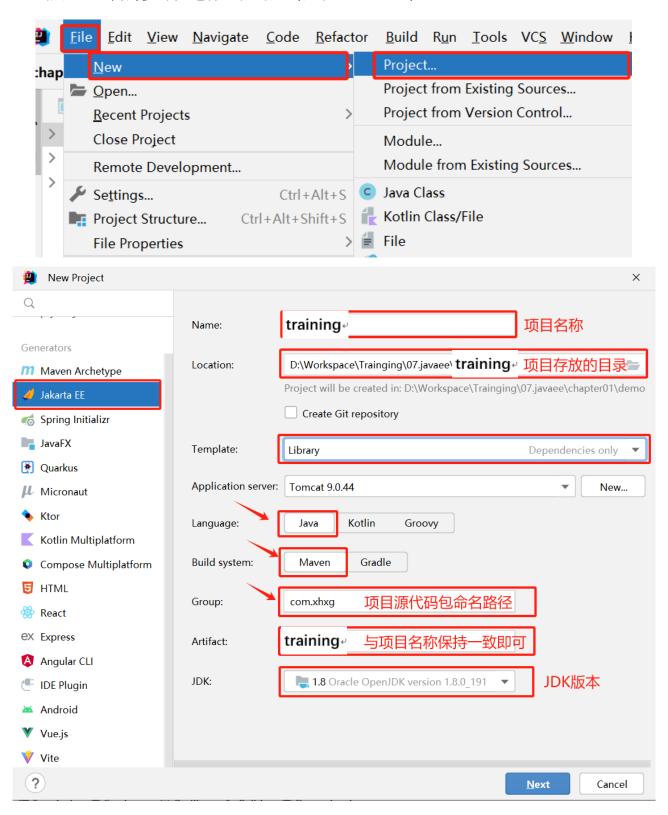
可以通过 Navicat 工具直接**读取此脚本文件**,以完成数据表的创建。

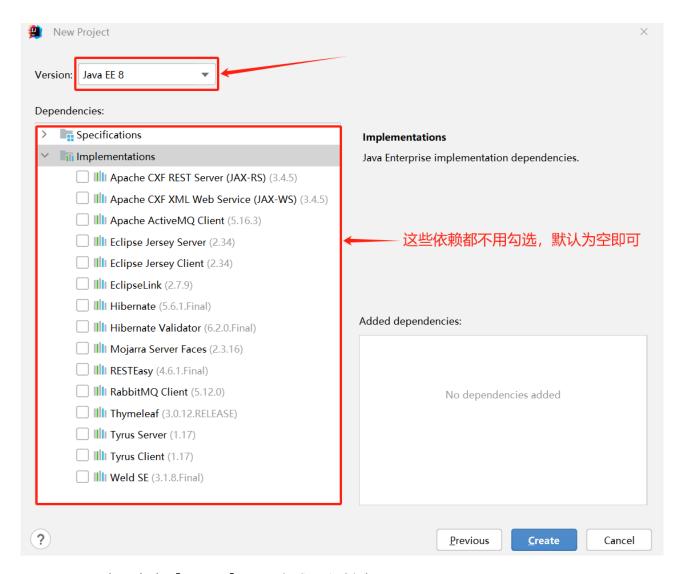


第2步: 创建 Maven 工程

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)

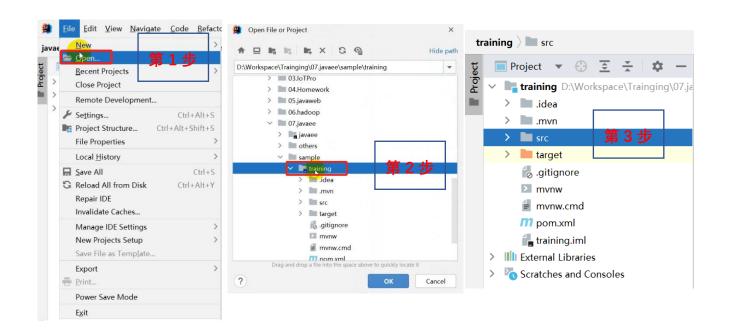




最后一步,点击【Create】即可完成工程创建。

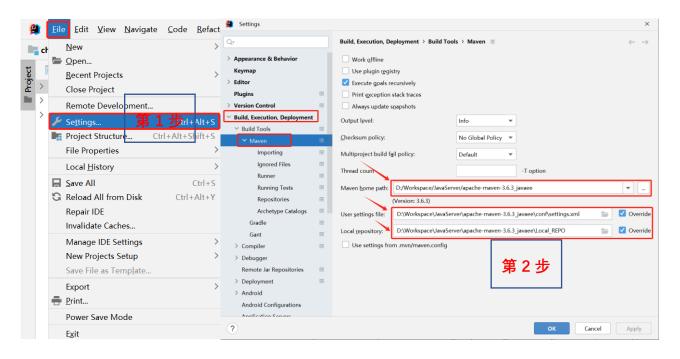
方式二: 导入已存在的 Maven 工程(推荐)

- Step1, 下载并解压 training 工程。从【学习通】→【章节】→【第四章···】→【上机实验 4···】,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,确定即可导入。
- **注意,**导入工程后,对于 IDEA 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。



第3步:配置本地 Maven 路径

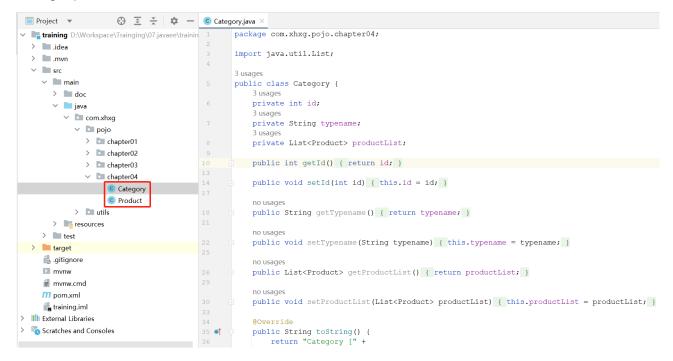
选择【File】→【Setting···】然后找到菜单位【Build, Execution, Deployment】→【Build Tools】→【Maven】,按如下图所示,将你的本地 Maven 路径和配置文件设置好即可。



第 4 步:程序代码开发

1、创建 POJO 实体

打开 training 项目工程。创建代码包为: com. xhxg. pojo. chapter04, 并且创建Category 和 Product 类



2、创建映射文件 CategoryMapper.xml

在 src/main/resources 目录下创建目录 mapper/chapter04, 然后创建一个CategoryMapper. xml 文件,内容如下。

```
⊕ ₹ ★ −  dategoryMapper.xml
                                          <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"</pre>
training D:\Workspace\Trainging\07.
                                                                                                                                             A4 3
· 🗎 .idea
                                                  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
· III .mvn
✓ src
                                          <mapper namespace="com.xhxq.pojo.chapter04.CategoryMapper">
 ∨ I main
    > doc
                                              <select id="findCategoryWithProduct" parameterType="Integer" resultMap="CategoryWithProductResult"</pre>
    > 📄 java
                                                  SELECT c.*,p.id as category_id,p.goodsname,p.price

√ I resources

                                                  from t category c,t product p

∨ Imapper
mapper

      > chapter01
                                                  and c.id=#{id}
         > lim chapter02
                                              </select>
         > chapter03
         ∨ <u>chapter04</u>
                                             Category Mapper.xml
              ProductMapper.xml
                                                  <!-- collection:表示多对多关联映射,
ofType:表示属性集合中元素的类型,
         db.properties
         🚮 log4i.properties
                                                      property:表示属性名,
以上两个在实体类中表示:List<Product> productList
         amybatis-config.xml
 > test
                                                  target
  륂 .gitignore
                                                  </collection>
  mvnw.cmd
                                              </resultMap>
```

```
← CategoryMapper.xml ×

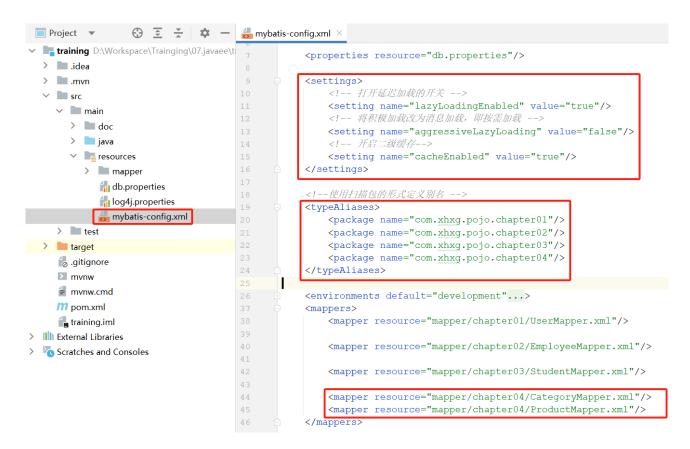
                                                                                                      A4 X!
           <select id="findCategoryWithPorduct2" parameterType="Integer" resultMap="CategoryWithProductResult2";</pre>
              select * from t_category WHERE id=#{id}
           <!-- 嵌套查询, 一对多映射类型 -->
           <resultMap type="Category" id="CategoryWithProductResult2">
              <id property="id" column="id"/>
41
              <result property="typename" column="typename"/>
              <!-- collection: 表示一对多关联映射,
                   ofType:表示属性集合中元素的类型,
                   property:表示实体集合对象属性名,
                   以上两个在实体类中表示: List<Product> productList
                   column:表示Category表关联字段
                   select:表示通过查询获得这个属性的结果集 -->
              <collection property="productList" column="id" ofType="Product"</pre>
                          select="com.xhxg.pojo.chapter04.ProductMapper.findProductById">
              </collection>
           </resultMap>
      </mapper>
```

再创建一个 ProductMapper.xml 文件,内容如下。



3、创建 mybatis-config.xml 配置文件

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



里面的配置项,除了数据库的连接信息外,还需要将前面的 CategoryMapper.xml 和 ProductMapper.xml 映射文件引入,另外还需要创建另一个配置文件 db.properties

4、创建 db.properties 配置文件

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



5、创建一个 Mybatis 连接工具类 JDBCUtils

由于执行测试时,每个方法执行都需要读取配置文件,并根据配置文件的信息构建 SqlSessionFactory 对象、创建 SqlSession 对象、释放资源,这导致了大量的重复代码。为了 简化开发,我们可以将读取配置文件和释放资源的代码封装到一个工具类 JDBCUtils 中(需要 创建一个 utils 的 package 包, 类路径: com.xhxg.utils.JDBCUtils,), 然后通过工具类 (JDBCUtils)

创建 SqlSession 对象。代码如下:

```
package com.xhxg.utils;
  > 🗎 .idea
                                  import org.apache.ibatis.io.Resources;
  > mvn
                                  import org.apache.ibatis.session.SqlSession;
  ∨ src
                                  import org.apache.ibatis.session.SqlSessionFactory;
    ∨ III main
                                  import org.apache.ibatis.session.SqlSessionFactoryBuilder;
      > doc
                                  import java.io.Reader;
      ∨ 📄 java

✓ □ com.xhxg

           > 🛅 pojo
                                  public class JDBCUtils {

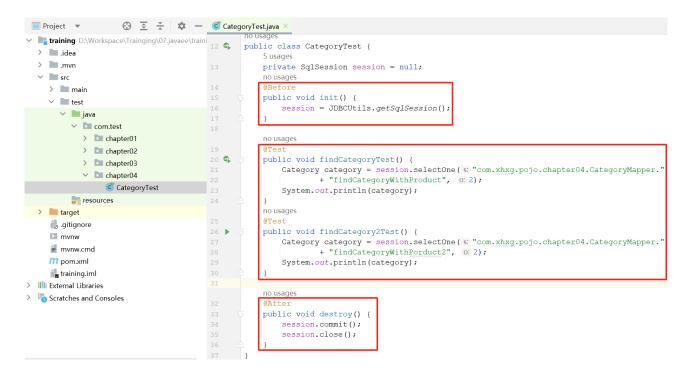
✓ utils

                                     2 usages
               JDBCUtils
                                     private static SqlSessionFactory sqlSessionFactory = null;
      > resources
    > limitest
                                         try {
> 🖿 target
                                             Reader reader = Resources.getResourceAsReader("mybatis-config.xml");
    \rm gitignore
                                             sqlSessionFactory = new SqlSessionFactoryBuilder().build(reader);
    } catch (Exception e) {
                                             e.printStackTrace();
    mvnw.cmd
    m pom.xml
    fraining.iml
> III External Libraries
> Cratches and Consoles
                                      public static SqlSession getSqlSession()
                                         return sqlSessionFactory.openSession();
```

6、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. chapter04, 然后创建测类 CategoryTest,测试的方法如下:



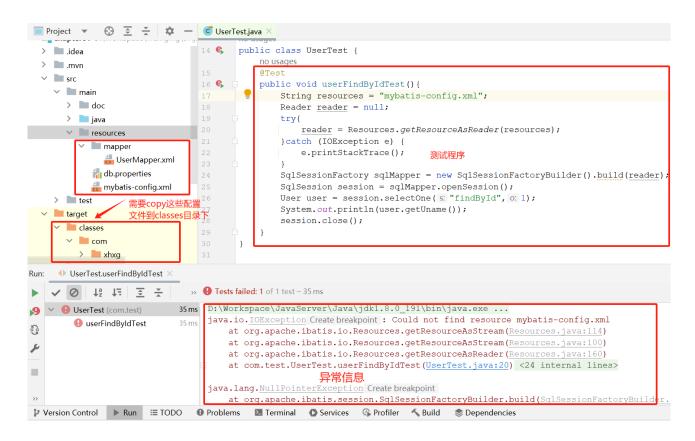
在不同的单元测试方法中,执行不同的单元测试,并输出相应的结果。

至此,本实验完满成功,建立了 Java 代码与数据库的关联,验证了通过 Java 调用,正确获取数据库中的记录。

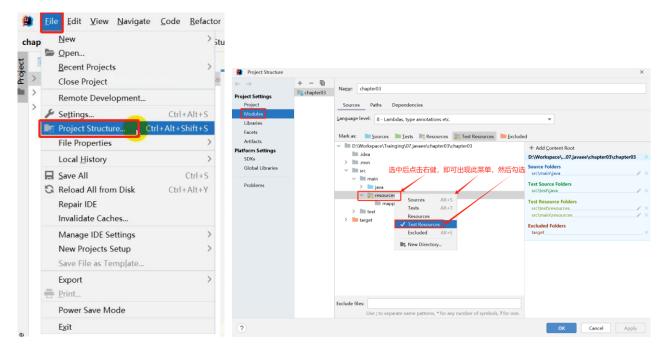
三、常见问题

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