文件编号： D000-

Mybatis框架技术

版本：3.6.0

第1章 Mybatis框架入门

本章教学目标

- 了解什么是Mybatis框架简介；
- 理解Mybatis框架功能架构；
- 掌握Mybatis框架项目创建；
- 掌握Mybatis框架的配置文件；
- 掌握Mybatis框架的sql映射和测试代码

本章教学内容

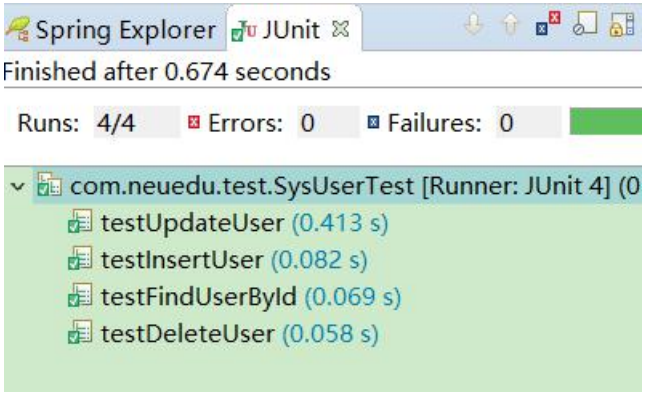
Neuedu

节	知识点	掌握程度	难易程度	教学形式	对应在线微课
MyBatis框架概述	MyBatis框架简介	了解		线下	MyBatis框架简介
	MyBatis框架功能架构	理解	难	线下	MyBatis框架功能架构
入门程序	MyBatis框架项目创建流程	理解		线下	MyBatis框架项目创建流程
	基础配置文件	掌握		线下	基础配置文件
	查询功能实现	掌握		线下	查询功能实现
	插入功能实现	掌握	难	线下	插入功能实现
	删除&修改功能实现	掌握		线下	删除&修改功能实现
本章实战项目任务实现	实战项目任务实现	掌握		线下	实战项目任务实现

本章实战项目任务

- 通过本章内容的学习，完成《跨境电商系统》用户信息的增删改查功能。
  - 创建Maven工程ch01-mybatis-project
  - 编写基础配置文件SqlMapConfig.xml
  - 在mapper文件中编写增删改查的映射配置
  - 编写测试代码

- Junit测试输出效果如下图：



CONTENTS 目录

Mybatis框架简介

Neuedu

- 什么是MyBaits
  - Mybatis 本是apache的一个开源项目iBatis, 2010年这个项目由apache software foundation 迁移到了google code，并且改名为Mybatis 。2013年11月迁移到Github。
  - iBATIS一词来源于“internet”和“abatis”的组合，是一个基于Java的持久层框架。iBATIS提供的持久层框架包括SQL Maps和Data Access Objects（DAO）
  - Mybatis是一个数据持久层(ORM)框架。把实体类和SQL语句之间建立了映射关系，是一种半自动化的ORM实现。

  - 下载地址：https://github.com/mybatis

Mybatis框架简介

Neuedu

- Mybatis的优点：
  - 1.基于SQL语法，简单易学。
  - 2.能了解底层组装过程。
  - 3.SQL语句封装在配置文件中，便于统一管理与维护，降低了程序的耦合度。
  - 4.程序调试方便。
  - 所有sql语句，全部定义在xml（建议）中。也可以通过注解的方式在接口上实现。这些映射文件称之为mapper。

Mybatis框架简介

Neuedu

- 通过编写JDBC代码，例如：根据用户名查询用户信息
  - 全部代码参见：ch01-mybatis01工程

```
Class.forName("oracle.jdbc.driver.OracleDriver");
connection =  DriverManager.getConnection("jdbc:oracle:thin:@10.10.21.3:1521:orcl", "scott", "tiger");
String sql = "select * from userinfo where username = ?";
preparedStatement = connection.prepareStatement(sql);
preparedStatement.setString(1, "王五");
resultSet =  preparedStatement.executeQuery();
while(resultSet.next()){
    System.out.println(resultSet.getString("id")+"  "+resultSet.getString("username"));
}
```

- 发现JDBC操作非常繁琐，
  - 1. 定义数据库连接参数
  - 2. 打开数据库连接
  - 3. 声明SQL语句
  - 4. 预编译并执行SQL语句
  - 5. 遍历查询结果（如果需要的话），对每一记录行进行处理
  - 6. 处理事务
  - 7. 关闭数据库连接
  - 以上步骤每次除了3和5步骤，其他全部是重复工作。

Mybatis框架简介

Neuedu

- 下面是Mybatis的配置文件
  - 根据用户名查询用户信息

```
<mapper namespace="test">
    <select id="findUserByName" parameterType="java.lang.String" resultType="com.neuedu.pojo.User">
        SELECT * FROM USERINFO WHERE username LIKE '%${value}%'
    </select>
</mapper>
```

- 与传统JDBC的比较
  - 减少了61%的代码量
  - 最简单的持久化框架
  - 架构级性能增强
  - SQL代码从程序代码中彻底分离，可重用
  - 增强了项目中的分工
  - 增强了移植性

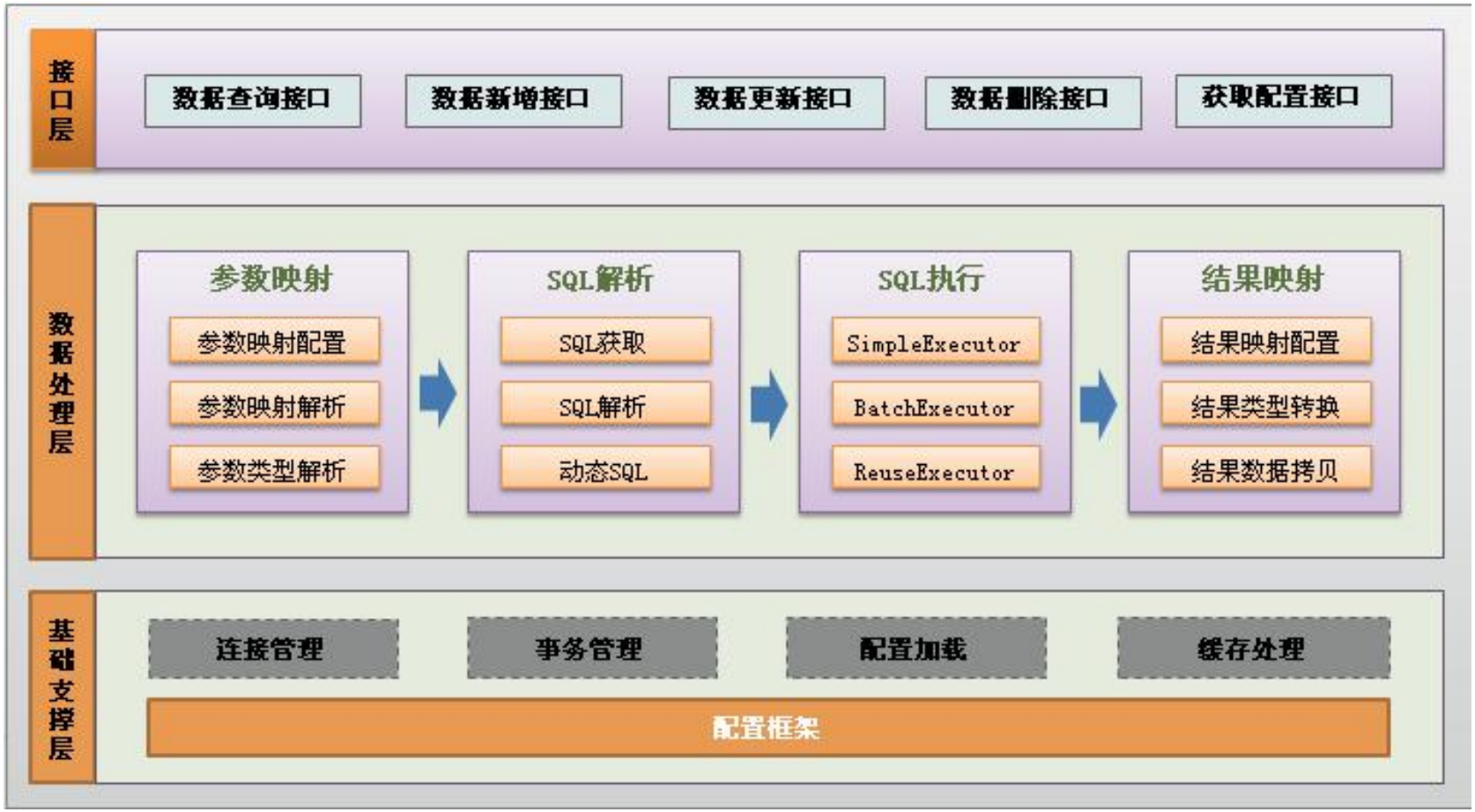
  - 全部代码参见：ch01-mybatis01工程

Mybatis框架功能架构

- Mybatis框架功能架构分为三层
  - API接口层：提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理。
  - 数据处理层：负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。它主要的目的是根据调用的请求完成一次数据库操作。
  - 基础支撑层：负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

Mybatis框架功能架构

- Mybatis架构图

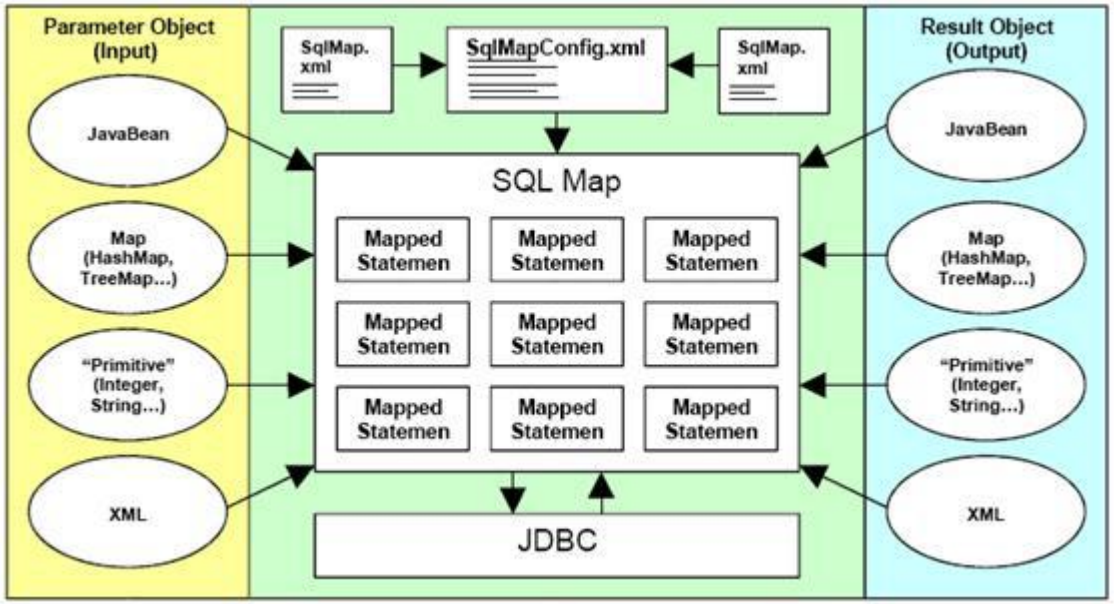


Mybatis框架功能架构

- Mybatis架构图
  - 我们把Mybatis的功能架构分为三层：
  - (1)API接口层：提供给外部使用的接口API，开发人员通过这些本地API来操纵数据库。接口层一接收到调用请求就会调用数据处理层来完成具体的数据处理。
  - (2)数据处理层：负责具体的SQL查找、SQL解析、SQL执行和执行结果映射处理等。它主要的目的是根据调用的请求完成一次数据库操作。
  - (3)基础支撑层：负责最基础的功能支撑，包括连接管理、事务管理、配置加载和缓存处理，这些都是共用的东西，将他们抽取出来作为最基础的组件。为上层的数据处理层提供最基础的支撑。

Mybatis框架功能架构

- Mybatis工作流程



Mybatis框架功能架构

- Mybatis工作流程
  - (1)加载配置并初始化
    - 将SQL的配置信息加载成为一个个MappedStatement对象（包括了传入参数映射配置、执行的SQL语句、结果映射配置），存储在内存中。
  - (2)接收调用请求
    - 传入参数：为SQL的ID和传入参数对象
  - (3)处理操作请求
    - 处理过程：
    - (A)根据SQL的ID查找对应的MappedStatement对象。
    - (B)根据传入参数对象解析MappedStatement对象，得到最终要执行的SQL和执行传入参数。
    - (C)获取数据库连接，根据得到的最终SQL语句和执行传入参数到数据库执行，并得到执行结果。
    - (D)根据MappedStatement对象中的结果映射配置对得到的执行结果进行转换处理，并得到最终的处理结果。
    - (E)释放连接资源。
  - (4)返回处理结果
    - 将最终的处理结果返回。

课堂练习（5分钟）

- 1、简述Mybatis框架及其优点
- 2、与传统的jdbc相比，Mybatis有什么不同
- 3、简述Mybatis架构
- 4、详述Mybatis工作流程

CONTENTS 目录

Mybatis框架项目开发流程

Neuedu

- Mybatis基本要素
  - SqlMapConfig.xml 全局配置文件
  - Mapper.xml 核心映射文件
  - SqlSessionFactory、SqlSession接口

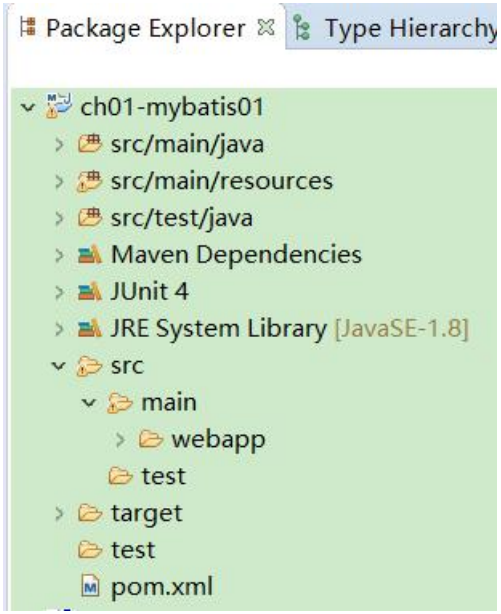
Mybatis框架项目开发流程

Neuedu

- Mybatis项目创建流程
  - 1、新建Maven项目
  - 2、pom.xml配置，导入Mybatis的依赖jar包
  - 3、创建SqlMapConfig.xml 全局配置文件，配置数据源、事务等Mybatis运行环境
  - 4、创建mapper.xml映射文件，配置增、删、改、查的SQL语句。
  - 5、创建SqlSessionFactory，根据全局配置文件创建工厂
  - 6、创建SqlSession，是一个接口，执行数据库操作
  - 7、释放资源

新建maven项目

- File→New→Maven Project→选择maven-archetype-webapp，输入Group id 和Artifact Id →Finish
- 完整的maven web工程有以下几个目录：
  - src/main/java：存放项目代码文件
  - src/main/resources：存放项目配置文件
  - src/main/webapp：存放web页面文件
  - src/test：测试文件
  - 如新建后有缺失，新建补全



基础配置文件—环境配置

Neuedu

- 配置环境
  - 新建SqlMapConfig.xml文件

- 全部代码参见：ch01-mybatis01工程

```
<configuration>
    <environments default="development">
        <environment id="development">
            <transactionManager type="JDBC"/>
            <dataSource type="POOLED">
            <property name="driver" value="${driver}"/>
            <property name="url" value="${url}"/>
            <property name="username" value="${username}"/>
            <property name="password" value="${password}"/>
            </dataSource>
        </environment>
    </environments>
</configuration>
```

基础配置文件—事务管理

Neuedu

- Mybatis 有两种事务管理类型：
  - JDBC – 这个类型直接全部使用 JDBC 的提交和回滚功能。它依靠使用连接的数据源来管理事务的作用域。
  - MANAGED – 这个类型什么不做 ， 它从不提交 、 回滚和关闭连接 。而是让窗口来管理事务的全部生命周期 。（比如说 Spring 或者 JAVAEE 服务器）

基础配置文件—数据源

Neuedu

- 数据源类型有三种： UNPOOLED ， POOLED ， JNDI 。
- UNPOOLED – 这个数据源实现只是在每次请求的时候简单的打开和关闭一个连接。虽然这有点慢，但作为一些不需要性能和立即响应的简单应用来说 ， 不失为一种好选择 。
- POOLED – 这个数据源缓存 JDBC 连接对象用于避免每次都要连接和生成连接实例而需要的验证时间 。对于并发 WEB 应用，这种方式非常流行因为它有最快的响应时间。
- JNDI – 这个数据源实现是为了准备和 Spring 或应用服务一起使用，可以在外部也可以在内部配置这个数据源，然后在 JNDI 上下文中引用它。这个数据源配置只需要两上属性：

查询功能实现--映射配置

Neuedu

```
<mapper namespace="test">
    <!-- 需求：通过id查询用户表的记录 -->
    <!-- 通过 select执行数据库查询
    id：标识 映射文件中的 sql   将sql语句封装到mappedStatement对象中，所以将id称为statement的id
    parameterType：指定输入 参数的类型，这里指定int型
    #{}表示一个占位符号
    #{id}：其中的id表示接收输入 的参数，参数名称就是id，如果输入 参数是简单类型，#{}中的参数名可以任意，可以value或其它名称
    resultType：指定sql输出结果 的所映射的java对象类型，select指定resultType表示将单条记录映射成的java对象。
     -->
    <select id="findUserById" parameterType="int" resultType="com.neuedu.pojo.User">
        SELECT * FROM T_USER WHERE ID = #{value}
    </select>
</mapper>
```

全部代码参见：ch01-mybatis01工程

查询功能—代码实现

Neuedu

```
public void findUserById() throws IOException{
    InputStream inputStream = Resources.getResourceAsStream("config/SqlMapConfig.xml");
    SqlSessionFactory sqlSessionFactory =
            new SqlSessionFactoryBuilder().build(inputStream);
    SqlSession sqlSession = sqlSessionFactory.openSession();
    User user = sqlSession.selectOne("test.findUserById",2);
    System.out.println(user.getUserName());
    sqlSession.close();
}
```

全部代码参见：ch01-mybatis01工程

插入功能实现--映射配置

Neuedu

```
<!-- 添加用户 parameterType：指定输入 参数类型是pojo（包括 用户信息）
    #{}中指定pojo的属性名，接收到pojo对象的属性值，mybatis通过OGNL获取对象的属性值-->
<insert id="insertUser" parameterType="com.neuedu.pojo.User">
    insert into t_user(id,username,birthday,address) values (seq_user.nextval,#{userName},#{birthDay},#{address})
    <!-- 插入数据后，返回自动增长列的ID值，将sql语句的返回值赋给parameterType绑定对象的ID属性
          MySQL中使用:  select LAST_INSERT_ID() -->
    <selectKey keyProperty="id" order="AFTER" resultType="int">
        select seq_user.currval from dual
    </selectKey>
</insert>
```

全部代码参见：ch01-mybatis01工程

插入功能—代码实现

Neuedu

```
public void insertUser() throws IOException{
    InputStream inputStream = Resources.getResourceAsStream("config/SqlMapConfig.xml");
    SqlSessionFactory factory =  new SqlSessionFactoryBuilder().build(inputStream);
    SqlSession session = factory.openSession();
    User u = new User();
    u.setUserName("abc");
    u.setBirthDay(new Date());
    u.setAddress("tianjin");
    session.insert("test.insertUser",u);
    System.out.println(u.getId());
    session.commit();
    session.close();
}
```

全部代码参见：ch01-mybatis01工程

插入功能—代码实现

Neuedu

- 返回插入主键ID
  - selectKey标签实现
  - 在使用Mybatis做持久层时，insert语句默认是不返回记录的主键值，而是返回插入的记录条数；
  - 如果业务层需要得到记录的主键时，可以通过配置的方式来完成这个功能

```
<insert id="insertUser" parameterType="User">
  <!--返回插入用户的ID-->
  <selectKey keyProperty="id" order="AFTER" resultType="int">
      select seq_user.currval from dual
  </selectKey>
  insert into t_user values(seq_user.nextval,#{username},#{birthday},#{address})
</insert>
```

全部代码参见：ch01-mybatis01工程

删除&修改功能实现--映射配置

Neuedu

```
<delete id="deleteUser" parameterType="int">
    delete from t_user where id = #{id}
</delete>
<update id="updateUser" parameterType="com.neuedu.pojo.User" >
    update t_user set username=#{userName},birthday=#{birthDay},address=#{address} where id = #{id}
</update>
```

全部代码参见：ch01-mybatis01工程

删除&修改功能实现—代码实现

Neuedu

```
public void deleteUser() throws IOException{
    InputStream inputStream = Resources.getResourceAsStream("config/SqlMapConfig.xml");
    SqlSessionFactory factory =  new SqlSessionFactoryBuilder().build(inputStream);
    SqlSession session = factory.openSession();
    session.delete("test.deleteUser",24);
    session.commit();
    session.close();
}
@Test
public void updateUser() throws IOException{
    InputStream inputStream = Resources.getResourceAsStream("config/SqlMapConfig.xml");
    SqlSessionFactory factory =  new SqlSessionFactoryBuilder().build(inputStream);
    SqlSession session = factory.openSession();
    User u = session.selectOne("test.findUserById",2);
    u.setUserName("fff");
    session.update("test.updateUser", u);
    session.commit();
    session.close();
}
```

全部代码参见：ch01-mybatis01工程

课堂练习(30分钟)

- 1、 Mybatis 有两种事务管理类型，分别是( )。
- A.JDBC B.JDO C.JTA D.MANAGED

- 2、Mybatis的数据源类型有三种，分别是( )。
- A. UNPOOLED B.POOLED C.DBCP D.JNDI

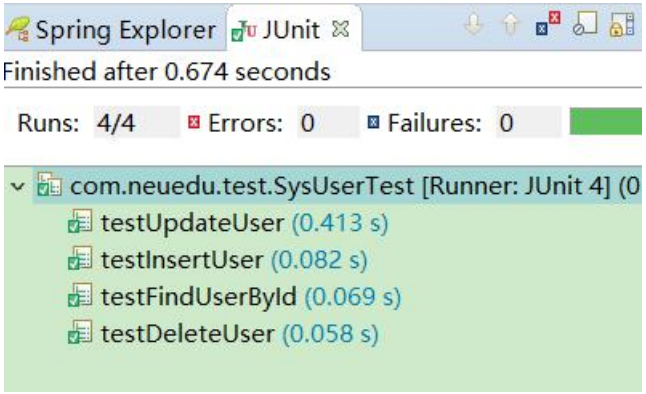
- 3、理解并独立完成课堂案例

CONTENTS 目录

本章实战项目任务实现

- 通过本章内容的学习，完成《跨境电商系统》用户信息的增删改查功能。
  - 创建Maven工程ch01-mybatis-project
  - 编写基础配置文件SqlMapConfig.xml
  - 在mapper文件中编写增删改查的映射配置
  - 编写测试代码

- Junit测试输出效果如下图：



本章重点总结

- 了解什么是Mybatis框架简介；
- 理解Mybatis框架功能架构；
- 掌握Mybatis框架项目创建；
- 掌握Mybatis框架的配置文件；
- 掌握Mybatis框架的sql映射和测试代码；

课后作业【必做任务】

- 使用Mybatis框架完成《订单商品管理系统》的商品的增删改查功能。
  - 创建Maven工程ch01-mybatis-khzy
  - 编写基础配置文件SqlMapConfig.xml
  - 在mapper文件中编写增删改查的映射配置
  - 编写测试代码

课后作业【线上任务】

- 线上任务
  - 安排学员线上学习任务（安排学员到睿道实训平台进行复习和预习的任务，主要是进行微课的学习）