Neuedu

东软睿道内部公开
文件编号： D000-

Mybatis框架技术

版本：3.6.0

第2章 Mybatis核心配置与DAO开发

本章教学目标

- 掌握Mybatis核心配置；
- 掌握Mybatis核心接口和类；
- 掌握Mybatis原始dao开发方法；
- 掌握Mybatis mapper代理开发方法；

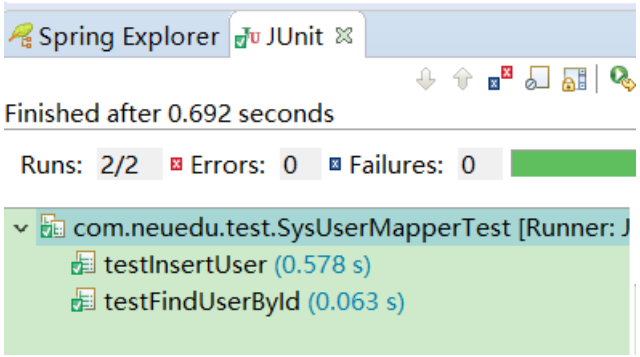
本章教学内容

Neuedu

节	知识点	掌握程度	难易程度	教学形式	对应在线微课
Mybatis核心配置	SqlMapConfig.xml 全局配置文件	掌握		线下	SqlMapConfig.xml 全局配置文件
	mapper.xml映射文件	掌握		线下	mapper.xml映射文件
Mybatis核心接口和类	SqlSessionFactoryBuilder	掌握		线下	SqlSessionFactoryBuilder
	SqlSessionFactory	掌握		线下	SqlSessionFactory
	SqlSession	掌握		线下	SqlSession
Mybatis开发dao的方法	原始dao开发方法	掌握		线下	原始dao开发方法
	原始dao开发问题	掌握		线下	原始dao开发问题
	mapper代理开发	掌握	难	线下	mapper代理开发
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，使用mapper代理开发方法，完成《跨境电商系统》用户信息的增加及查询查功能。
  - 创建Maven工程ch02-mybatis-project
  - 编写SqlMapConfig.xml 和Mapper.xml
  - 编写mapper接口
  - 编写测试代码
- Junit测试输出效果如下图：



CONTENTS 目录

Mybatis核心配置文件

- SqlMapConfig.xml 全局配置文件
- mapper.xml映射文件

SqlMapConfig.xml 全局配置文件

Neuedu

- SqlMapConfig的XML配置文件包含了影响MyBatis 行为的设置和属性信息,文档结构如下:
  - configuration 配置
    - properties 属性
    - settings 设置
    - typeAliases 类型命名
    - typeHandlers 类型处理器
    - objectFactory 对象工厂
    - plugins 插件
    - Environments 环境
    - mappers 映射器

SqlMapConfig.xml 全局配置文件

Neuedu

- properties全局参数
  - 加载属性文件

```
<properties resource="config/db.properties">
<!--properties中还可以配置一些属性名和属性值  -->
<!-- <property name="jdbc.driver" value=""/> -->
</properties>
```

- 全部代码参见：ch02-mybatis01工程

- MyBatis 将按照下面的顺序来加载属性：
  - 在 properties 元素体内定义的属性首先被读取。
  - 然后会读取properties 元素中resource或 url 加载的属性，它会覆盖已读取的同名属性。
  - 最后读取parameterType传递的属性，它会覆盖已读取的同名属性。

SqlMapConfig.xml 全局配置文件

Neuedu

- settings全局参数
  - Mybatis框架在运行时可以调整一些运行参数，比如：开启二级缓存、开启延迟加载等
  - 全局参数将会影响Mybatis的运行行为。
  - 详细参见MyBatis3帮助文档（中文版）.chm

Setting(设置)	Description（描述）	Valid Values(验证值组)	Default(默认值)
cacheEnabled	在全局范围内启用或禁用缓存配置任何映射器在此配置下。	true | false	TRUE
lazyLoadingEnabled	在全局范围内启用或禁用延迟加载。禁用时，所有协会将热加载。	true | false	TRUE
aggressiveLazyLoading	启用时，有延迟加载属性的对象将被完全加载后调用懒惰的任何属性。否则，每一个属性是按需加载。	true | false	TRUE
multipleResultSetsEnabled	允许或不允许从一个单独的语句（需要兼容的驱动程序）要返回多个结果集。	true | false	TRUE
useColumnLabel	使用列标签，而不是列名。在这方面，不同的驱动有不同的行为。参考驱动文档或测试两种方法来决定你的驱动程序的行为如何。	true | false	TRUE
useGeneratedKeys	允许JDBC支持生成的密钥。兼容的驱动程序是必需的。此设置强制生成的键被使用，如果设置为true，一些驱动会不兼容性，但仍然可以工作。	true | false	FALSE

SqlMapConfig.xml 全局配置文件

Neuedu

- typeAliases全局参数
  - 在mapper.xml中，定义的statement需要parameterType指定输入参数的类型、需要resultType指定输出结果的映射类型。如果在指定类型时输入类型全路径，不方便进行开发，可以针对parameterType或resultType指定的类型定义一些别名，方便开发。

```
<typeAliases>
    <!-- 针对单个别名定义 type：类型的路径 alias：别名 -->
    <!-- <typeAlias type="com.neuedu.pojo.User" alias="user"/> -->
    <!-- 批量别名定义
    指定包名，mybatis自动扫描包中的po类，自动定义别名，别名就是类名（首字母大写或小写都可以）
    -->
    <package name="com.neuedu.pojo"/>
</typeAliases>
```

  - 全部代码参见：ch02-mybatis01工程

SqlMapConfig.xml 全局配置文件

Neuedu

- typeHandlers全局参数
  - Mybatis中通过typeHandlers(类型处理器)完成jdbc类型和java类型的转换。通常情况下，Mybatis提供的类型处理器满足日常需要，不需要自定义

类型处理器	Java类型	JDBC类型
BooleanTypeHandler	Boolean，boolean	任何兼容的布尔值
ByteTypeHandler	Byte，byte	任何兼容的数字或字节类型
ShortTypeHandler	Short，short	任何兼容的数字或短整型
IntegerTypeHandler	Integer，int	任何兼容的数字和整型
LongTypeHandler	Long，long	任何兼容的数字或长整型
FloatTypeHandler	Float，float	任何兼容的数字或单精度浮点型
DoubleTypeHandler	Double，double	任何兼容的数字或双精度浮点型
BigDecimalTypeHandler	BigDecimal	任何兼容的数字或十进制小数类型
StringTypeHandler	String	CHAR和VARCHAR类型
ClobTypeHandler	String	CLOB和LONGVARCHAR类型
NStringTypeHandler	String	NVARCHAR和NCHAR类型
NClobTypeHandler	String	NCLOB类型
ByteArrayTypeHandler	byte[]	任何兼容的字节流类型
BlobTypeHandler	byte[]	BLOB和LONGVARBINARY类型
DateTypeHandler	Date（java.util）	TIMESTAMP类型
DateOnlyTypeHandler	Date（java.util）	DATE类型
TimeOnlyTypeHandler	Date（java.util）	TIME类型
SqlTimestampTypeHandler	Timestamp（java.sql）	TIMESTAMP类型
SqlDateTypeHandler	Date（java.sql）	DATE类型
SqlTimeTypeHandler	Time（java.sql）	TIME类型
ObjectTypeHandler	任意	其他或未指定类型
EnumTypeHandler	Enumeration类型	VARCHAR-任何兼容的字符串类型，作

SqlMapConfig.xml 全局配置文件

Neuedu

- environments全局参数
  - environments主要用于配置数据库相关，而且可以在里面配置多个environment。
  - transactionManager
    - MyBatis中有两种事务管理器类型
    - JDBC：使用了JDBC的提交和回滚设置。
    - MANAGED(托管)：而它让容器来管理事务的整个生命周期（比如spring、jee应用服务器的上下文）
  - dataSource
    - 用来配置基本的JDBC数据源连接信息
    - 有三种内建的数据源类型UNPOOLED|POOLED|JNDI
  - 和spring整合后 environments配置将废除

SqlMapConfig.xml 全局配置文件

Neuedu

- mappers全局参数
  - 既然 MyBatis的行为已经由上述元素配置完了,我们现在就要定义SQL映射语句了。 但是, 首先我们需要告诉MyBatis到哪里去找到这些语句。

```
<mapper class="com.neuedu.mapper.UserMapper"/>
```

  - 这些语句简单告诉了MyBatis去哪里找映射文件。其余的细节就是在每个SQL映射文件中了,下面的部分我们来讨论SQL映射文件。

mapper.xml映射文件

- MyBatis 真正强大之处就在这些映射语句，也就是它的魔力所在。SQL映射文件的配置却非常简单对比JDBC 代码使用SQL映射文件配置可以节省95%的代码量。
- 需要配置的基本元素
  1. cache – 配置给定模式的缓存
  2. cache-ref – 从别的模式中引用一个缓存
  3. resultMap – 这是最复杂而却强大的一个元素了，它描述如何从结果集中加载对象
  4. sql – 一个可以被其他语句复用的SQL 块
  5. insert – 映射INSERT 语句
  6. update – 映射UPDATE 语句
  7. delete – 映射DELEETE 语句
  8. select – 映射SELECT语句

mapper.xml映射文件

- select查询语句是使用 MyBatis 时最常用的元素之一。
- 全部代码参见：ch02-mybatis01工程

```
<!-- 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=#{id}
</select>
```

mapper.xml映射文件

- insert插入语句是使用 MyBatis 时最常用的元素之一。

```
<!-- 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,sex,address)
    values (seq_user.nextval,#{username},#{birthday},#{sex},#{address})

    <!--
    keyProperty：将查询到主键值设置到parameterType指定的对象的哪个属性
    order：SELECT LAST_INSERT_ID()执行顺序，相对于insert语句来说它的执行顺序
    resultType：指定SELECT LAST_INSERT_ID()的结果类型
     -->
    <selectKey keyProperty="id" order="AFTER" resultType="int">
        select seq_user.currval from dual
    </selectKey>

</insert>
```

mapper.xml映射文件

- delete删除语句是使用 MyBatis 时最常用的元素之一。

```
<!-- 删除 用户
根据id删除用户，需要输入 id值
 -->
<delete id="deleteUser" parameterType="java.lang.Integer">
    delete from T_USER where id=#{id}
</delete>
```

mapper.xml映射文件

- update更新语句是使用 MyBatis 时最常用的元素之一。

```
<!-- 根据id更新用户
分析：
需要传入用户的id
需要传入用户的更新信息
parameterType指定user对象，包括 id和更新信息，注意：id必须存在
#{id}：从输入 user对象中获取id属性值
 -->
<update id="updateUser" parameterType="com.neuedu.pojo.User">
    update T_USER set username=#{username},birthday=#{birthday},sex=#{sex},address=#{address}
     where id=#{id}
</update>
```

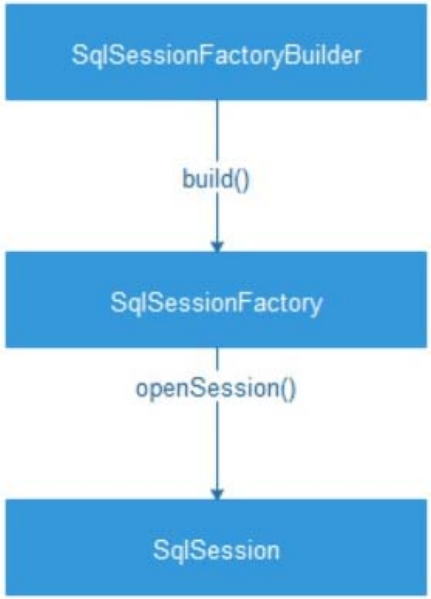
课堂练习（5分钟）

- 1、简述SqlMapConfig.xml 全局配置文件的作用，以及经常配置的项有哪些？
- 2、简述mapper.xml映射文件的作用，以及增删改查语句的映射方法

CONTENTS 目录

Mybatis核心接口和类

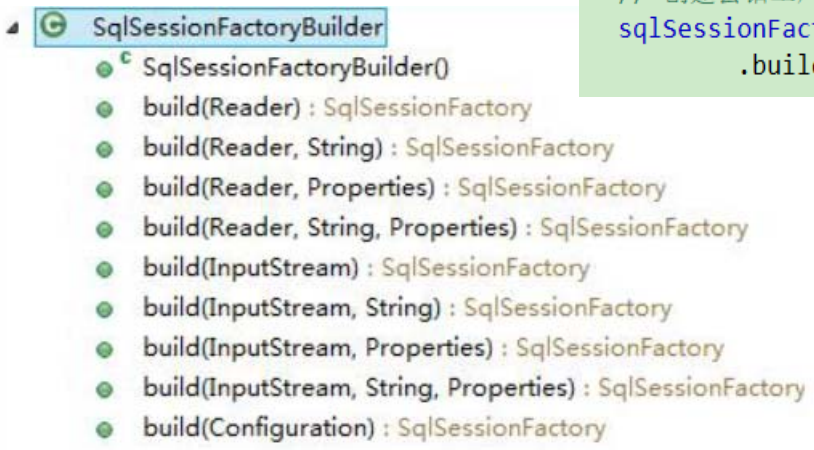
- （1）每个MyBatis的应用程序都以一个SqlSessionFactory对象的实例为核心；
- （2）首先获取SqlSessionFactoryBuilder对象，可以根据XML配置文件或Configuration类的实例构建该对象；
- （3）然后获取SqlSessionFactory对象，该对象实例可以通过SqlSessionFactoryBuilder对象来获得。
- （4）有了SqlSessionFactory对象之后，就可以进而获取SqlSession实例，SqlSession对象中完全包含以数据库为背景的所有执行SQL操作的方法。可以用该实例直接执行已映射的SQL语句。



SqlSessionFactoryBuider

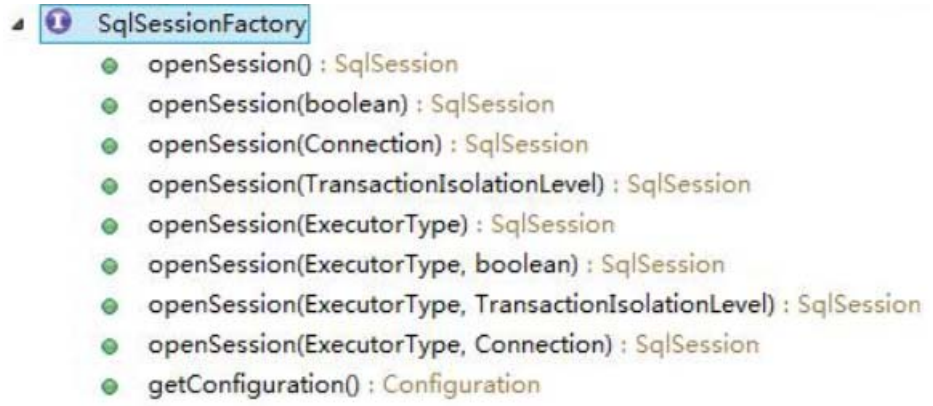
- SqlSessionFactoryBuilder负责构建SqlSessionFactory，并且提供了多个build()方法的重载。
- SqlSessionFactoryBuilder 对象可以从 XML 配置文件,或从 Configuration 类的实例中构建 SqlSessionFactory 对象。
- 全部代码参见：ch01-mybatis01工程

```
// mybatis配置文件
String resource = "config/SqlMapConfig.xml";
// 得到配置文件流
InputStream inputStream = Resources.getResourceAsStream(resource);
// 创建会话工厂，传入mybatis的配置文件信息
sqlSessionFactory = new SqlSessionFactoryBuilder()
        .build(inputStream);
```



SqlSessionFactory

- SqlSessionFactory的作用就是创建SqlSession实例的工厂。
- 通过SqlSessionFactory提供的openSession()方法来获取SqlSession实例。
- SqlSessionFactory一旦被创建，在应用执行期间都存在。
- 全部代码参见：ch01-mybatis01工程




```
SqlSession sqlSession = sqlSessionFactory.openSession();
```

SqlSession

- SqlSession是一个面向用户（程序员）的接口。它提供了面向数据库执行SQL命令所需的所有方法。
- SqlSession对应着一次数据库会话， SqlSession是线程不安全的


```
User user = sqlSession.selectOne("test.findUserById", id);
```

```
List<User> list = sqlSession.selectList("test.findUserByName", name);
```

全部代码参见：ch01-mybatis01工程

课堂练习（3分钟）

- 1、简述Mybatis核心接口和类，以及各自的作用
- 2、写出Mybatis核心接口和类的创建代码

Neuedu

CONTENTS 目录

Mybatis开发dao的方法

- 原始dao开发方法
  - 程序员需要写dao接口和dao实现类。
- mapper代理开发
  - 程序员只需要mapper接口和mapper.xml映射文件，Mybatis可以自动生成mapper接口实现类代理对象。

原始dao开发方法

- 1、程序员需要写dao接口和dao实现类。
- 2、需要向dao实现类中注入SqlSessionFactory，在方法体内通过SqlSessionFactory创建SqlSession
- 全部代码参见：ch02-mybatis01工程

```
public interface UserDao {
    public User findUserById(int id) throws Exception;
    public void insert(User user) throws Exception;
    public void delete(int id) throws Exception;
}
```

```
public class UserDaoImpl implements UserDao {
    private SqlSessionFactory sqlSessionFactory;
    public UserDaoImpl(SqlSessionFactory sqlSessionFactory) {
        this.sqlSessionFactory = sqlSessionFactory;
    }
    @Override
    public User findUserById(int id) throws Exception {
        SqlSession sqlSession = this.sqlSessionFactory.openSession();
        User user =  sqlSession.selectOne("test.findUserById",id);
        sqlSession.close();
        return user;
    }
```

课堂练习（20分钟）

- 使用原始dao开发方法，实现订单商品系统用户信息的更新和删除

原始dao开发问题

- 1、dao接口实现类方法中存在大量模板方法，设想能否将这些代码提取出来，大大减轻程序员的工作量。
- 2、调用sqlsession方法时将statement的id硬编码了
- 3、调用sqlsession方法时传入的变量，由于sqlsession方法使用泛型，即使变量类型传入错误，在编译阶段也不报错，不利于程序员开发。

- 全部代码参见：ch02-mybatis01工程

```
@Test
public void testFindUserById() throws Exception {
    UserDao dao = new UserDaoImpl(sqlSessionFactory);
    User u = dao.findUserById(2);
    System.out.println(u.getUserName());
}
```

mapper代理开发

Neuedu

- Mapper代理方法
  - 程序员只需要mapper接口和mapper.xml映射文件，Mybatis可以自动生成mapper接口实现类代理对象。
  - 程序员编写mapper接口需要遵循一些开发规范。

- mapper代理开发规范
  - 1、在mapper.xml中namespace等于mapper接口地址
  - 2、mapper.java接口中的方法名和mapper.xml中statement的id一致
  - 3、mapper.java接口中的方法输入参数类型和mapper.xml中statement的parameterType指定的类型一致。
  - 4、mapper.java接口中的方法返回值类型和mapper.xml中statement的resultType指定的类型一致。

mapper代理开发

Neuedu

- mapper代理开发方法流程
  - 1、创建接口，方法名、参数、返回值与mapper.xml中映射配置保持一致

```
public interface UserMapper {
    public User findUserById(int id) throws Exception;
    public List<Map<String, Object>> findUserList(User user) throws Exceptior
    public int findUserCount(User user) throws Exception;
}
```

  - 2、修改mapper.xml配置文件,namespace和接口地址一致

```
<mapper namespace="com.neuedu.mapper.UserMapper">
```

  - 3、在SqlMapConfig.xml中加载mapper.xml
  - 4、测试运行

```
UserMapper mapper = sqlSession.getMapper(UserMapper.class);
User user = new User();
user.setUserName("zhang");
List<Map<String, Object>> users = mapper.findUserList(user);
```

  - 全部代码参见：ch02-mybatis01工程

mapper代理开发

Neuedu

- 批量加载mapper

```
<mappers>
    <!--通过resource方法一次加载一个映射文件 -->
    <!--
    <mapper resource="config/sqlmap/User.xml"/>
    <mapper resource="config/sqlmap/UserMapper.xml"/>
    -->
    <!--通过mapper接口加载单个 映射文件前提是：使用的是mapper代理方法-->
    <!--
    <mapper class="com.neuedu.mapper.UserMapper"/>
    -->
    <!--批量加载mapper,指定mapper接口的包名，mybatis自动扫描包下边所有mapper接口进行加载,
        前提是：使用的是mapper代理方法-->
    <package name="com.neuedu.mapper"/>
</mappers>
```

- 全部代码参见：ch02-mybatis01工程

课堂练习（20分钟）

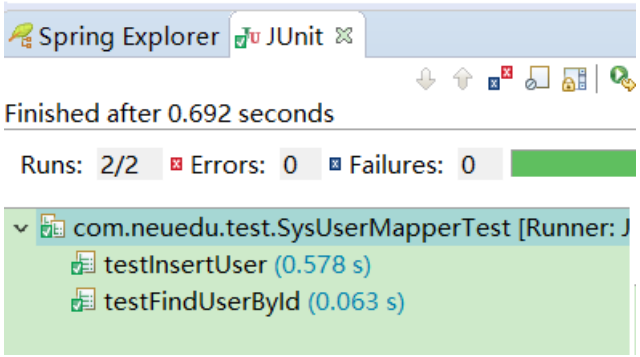
- 使用mapper代理，实现订单商品系统的用户信息的更新和删除

Neuedu

CONTENTS 目录

本章实战项目任务实现

- 通过本章内容的学习，使用mapper代理开发方法，完成《跨境电商系统》用户信息的增加及查询查功能。
  - 创建Maven工程ch02-mybatis-project
  - 编写SqlMapConfig.xml 和Mapper.xml
  - 编写mapper接口
  - 编写测试代码
- Junit测试输出效果如下图：



本章重点总结

- 掌握Mybatis核心配置；
- 掌握Mybatis核心接口和类；
- 掌握Mybatis原始dao开发方法；
- 掌握Mybatis mapper代理开发方法；

课后作业【必做任务】

Neuedu

- 1、使用传统的dao开发方法，完成《跨境电商系统》用户信息的增删改查功能。
  - 创建Maven工程ch02-mybatis-khzy01
  - 编写SqlMapConfig.xml 和Mapper.xml
  - 编写mapper接口
  - 编写测试代码

课后作业【必做任务】

Neuedu

- 2、使用mapper代理开发方法，完成《跨境电商系统》用户信息的更新和查询功能。
  - 创建Maven工程ch02-mybatis-khzy02
  - 编写SqlMapConfig.xml 和Mapper.xml
  - 编写mapper接口
  - 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）