MyBatis Generator_工具引入

MyBatis 生成器

最后发布时间：2019 年 11 月 24 日 | 版本：1.4.0

用户指导
介绍
什么是新的?
快速入门指南

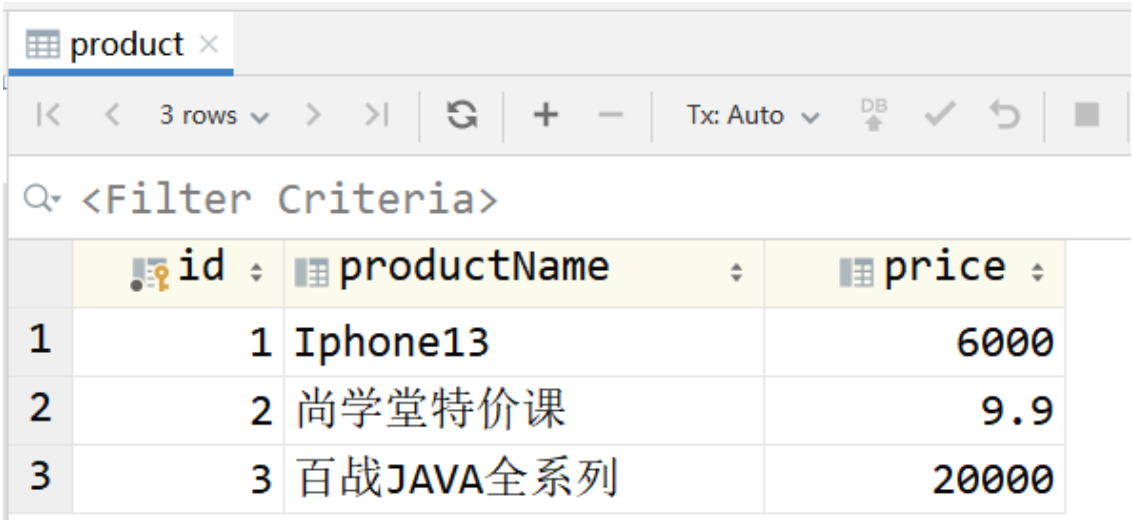
MyBatis 生成器介绍

MyBatis Generator (MBG) 是 MyBatis MyBatis的代码生成器。它将为所有版本的 MyBatis 生成代码。它将内省一个数据库表（或多个表）并生成可用于访问表的工件。这减少了设置对象和配置文件以与数据库表交互的初始麻烦。MBG 试图对大量简单的 CRUD（创建、检索、更新、删除）的数据库操作产生重大影响。您仍然需要为连接查询或存储过程编写 SQL 和对象代码。

MyBatis Generator（MBG）是MyBatis官方提供的代码生成器。它可以根据数据库的表结构自动生成POJO类、持久层接口与映射文件，极大减少了代码的编写量，提高开发效率。

MBG可以作为项目引入使用，也可以作为Maven插件使用，其中作为Maven插件使用更加方便快捷。

1. 准备数据库表



	id	productName	price
1	1	Iphone13	6000
2	2	尚学堂特价课	9.9
3	3	百战JAVA全系列	20000

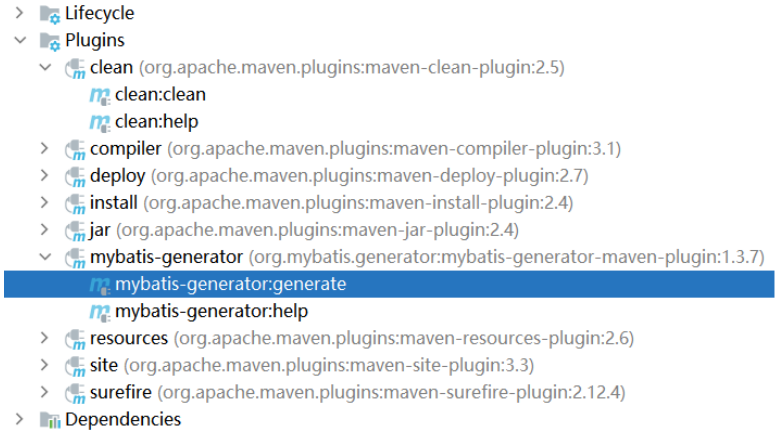
2. 在pom文件中配置MBG插件

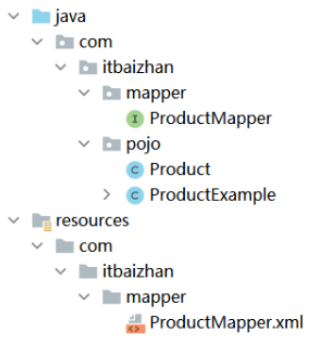
```
<build>
    <plugins>
        <plugin>
            <groupId>org.mybatis.generator</groupId>
            <artifactId>mybatis-generator-maven-plugin</artifactId>
            <version>1.3.7</version>
            <configuration>
                <!-- MBG配置文件位置 -->

<configurationFile>src/main/resources/generatorConfig.xml</configurationFile
>
                <!-- 运行显示详情 -->
                <verbose>true</verbose>
                <!-- 允许覆盖文件 -->
                <overwrite>true</overwrite>
            </configuration>
        </plugin>
    </plugins>
</build>
```

3. 编写MBG配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE generatorConfiguration
        PUBLIC "-//mybatis.org//DTD MyBatis Generator Configuration 1.0//EN"
        "http://mybatis.org/dtd/mybatis-generator-config_1_0.dtd">

<generatorConfiguration>
    <!-- jdbc的jar包位置，插件需要连接数据库 -->
    <classPathEntry location="F:\repository\mysql\mysql-connector-
java\8.0.26\mysql-connector-java-8.0.26.jar"/>

    <context id="default" targetRuntime="MyBatis3">
        <!-- 是否去除自动生成的注释-->
        <commentGenerator>
            <property name="suppressAllComments" value="true"/>
        </commentGenerator>

        <!--数据库连接参数-->
        <jdbcConnection driverClass="com.mysql.jdbc.Driver"
                        connectionURL="jdbc:mysql://localhost:3306/mybatis"
                        userId="root"
                        password="root"></jdbcConnection>

        <!-- 类型处理器，在数据库类型和java类型之间的转换控制-->
        <javaTypeResolver>
            <property name="forceBigDecimals" value="false"/>
        </javaTypeResolver>

        <!-- targetProject:JAVA类路径  targetProject:生成的POJO类的包-->
        <javaModelGenerator targetProject="src/main/java"
targetPackage="com.itbaizhan.pojo">
            <!-- 是否生成子包 -->
            <property name="enableSubPackages" value="false"/>
            <!-- 设置是否在getter方法中，对String类型字段调用trim()方法 -->
            <property name="trimStrings" value="true"/>
        </javaModelGenerator>

        <!-- targetProject:配置文件路径 targetPackage:生成映射文件的位置 -->
        <sqlMapGenerator targetProject="src/main/resources"
targetPackage="com.itbaizhan.mapper">
            <!-- 是否生成子包 -->
            <property name="enableSubPackages" value="false"/>
        </sqlMapGenerator>

        <!-- targetPackage：JAVA类路径 targetProject:生成的持久层接口包 -->
        <javaClientGenerator targetProject="src/main/java"
targetPackage="com.itbaizhan.mapper" type="XMLMAPPER">
            <!-- 是否生成子包 -->
            <property name="enableSubPackages" value="false"/>
        </javaClientGenerator>

        <!-- 数据库表，表名不要和其他库中的表名一样 -->
        <table tableName="product"></table>
    </context>
</generatorConfiguration>
```

4. 运行插件，自动生成POJO，持久层接口，映射文件：





- Product.java：POJO类
- ProductMapper.java：持久层接口
- ProductMapper.xml：映射文件
- ProductExample.java：查询扩展类，该类可以构造复杂的查询条件。
  - Criterion：代表一个字段。
  - GeneratedCriteria：抽象类，生成查询条件的工具。
  - Criteria：GeneratedCriteria的子类，生成查询条件的工具。

5. 在配置文件中注册生成的映射文件

```
<mappers>
    <mapper class="com.itbaizhan.mapper.ProductMapper">        </mapper>
</mappers>
```

实时学习反馈

1. 使用MyBaits Generator生成的文件不包括：

A POJO类

B 持久层接口类

C 映射文件

D 核心配置文件

2.MyBaits Generator生成的文件中，哪个类可以构造复杂查询条件？

A POJO类

B 持久层接口类

C Example类

D 映射文件

答案

1=>D 2=>C

MyBatis Generator_增删改方法


```
public class TestMBG {
    InputStream is = null;
    SqlSession session = null;
    ProductMapper productMapper = null;

    @Before
    public void before() throws IOException {
        is = Resources.getResourceAsStream("SqlMapConfig.xml");
        SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();
        SqlSessionFactory factory = builder.build(is);
        session = factory.openSession();
        productMapper = session.getMapper(ProductMapper.class);
    }

    @After
    public void after() throws IOException {
        session.close();
        is.close();
    }

    // 新增
    @Test
    public void testAdd(){
        Product product = new Product("百战Python课", 15000.0);
        productMapper.insert(product);
        session.commit();
    }
    // 修改
    @Test
    public void testUpdate(){
        Product product = new Product(5,"百战Python课", 25000.0);
        productMapper.updateByPrimaryKey(product);
```

```
        session.commit();
    }
    // 删除
    @Test
    public void testDelete(){
        productMapper.deleteByPrimaryKey(5);
        session.commit();
    }

}
```

实时学习反馈

1. Mybaits Generator生成的方法中，根据Id删除是什么方法：

A `deleteByPrimaryKey`

B `delete`

C `deleteOne`

D `destory`

答案

1=>A

MyBatis Generator_查询方法

```
// 根据id查询
@Test
public void testFindById() {
    Product product = productMapper.selectByPrimaryKey(1);
    System.out.println(product);
}

// 查询所有
@Test
public void testFindAll() {
    // 查询扩展对象，可以构建查询条件
    ProductExample productExample = new ProductExample();
    List<Product> products = productMapper.selectByExample(productExample);
    products.forEach(System.out::println);
}

// 根据商品名查询
@Test
public void testFindByName(){
    // 查询扩展对象，可以构建查询条件
    ProductExample productExample = new ProductExample();
    // 构建查询条件
    ProductExample.Criteria criteria = productExample.createCriteria();
    criteria.andProductnameLike("%尚学堂%");
    // 查询
```

```
    List<Product> products = productMapper.selectByExample(productExample);
    products.forEach(System.out::println);
}
```

实时学习反馈

1. Mybaits Generator生成的方法中，查询所有使用什么方法：

A `deleteByPrimaryKey`

B `selectByExample`

C `selectByPrimaryKey`

D `select`

答案

1=>B

MyBatis Generator_复杂查询

```
// 多条件and查询
@Test
public void testFindAnd() {
    // 查询扩展对象，可以构建查询条件
    ProductExample productExample = new ProductExample();
    // 构建查询条件
    ProductExample.Criteria criteria = productExample.createCriteria();
    criteria.andProductnameLike("%百战%");
    criteria.andPriceBetween(0.0,20000.0);

    // 查询
    List<Product> products = productMapper.selectByExample(productExample);
    products.forEach(System.out::println);
}

// 多条件or查询
@Test
public void testFindOr() {
    // 查询扩展对象，可以构建查询条件
    ProductExample productExample = new ProductExample();
    // 构建查询条件
    ProductExample.Criteria criteria = productExample.createCriteria();
    criteria.andProductnameLike("%百战%");

    ProductExample.Criteria criteria1 = productExample.createCriteria();
    criteria1.andPriceBetween(0.0,10000.0);

    productExample.or(criteria1);

    // 查询
    List<Product> products = productMapper.selectByExample(productExample);
    products.forEach(System.out::println);
```

```
}
```

实时学习反馈

1. 在Mybaits Generator中，`Criteria`类是类的内部类：

A `GeneratedCriteria`

B 持久层接口

C `POJO`

D `Example`

答案

1=>D