# 使用mybatis-plus处理数据库一张表格

创建springboot项目

导入依赖：lombook+mysql\_mybatis-plus

<dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-boot-starter</artifactId>  
 <version>Latest Version</version>  
</dependency>  
  
<dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
</dependency>  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
</dependency>

配置application.properties

配置数据库信息（mysql版本是8jdbc前加.cj）

#配置日志  
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.commons.JakartaCommonsLoggingImpl

spring.datasource.url=jdbc:mysql://localhost:3306/car\_data?useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8&useSSL=false  
spring.datasource.username=wangjie  
spring.datasource.password=19961016wj  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

编写实体类（字段和数据库保持一致，引入lombook依赖，这里简化配置getset等方法）

@Data  
public class User {

@TableId(type= IdType.*AUTO*) //使ID自增  
 private Long id;  
 private String name;  
 private Integer age;  
 private String email;  
}

编写mapper

@Repository //代表持久层  
public interface UserMapper extends BaseMapper<User>{  
}

## 查询测试

@Test//查询所有用户  
void select() {  
 //查询所有用户  
 List<User> user=userMapper.selectList ( null );  
 System.*out*.println (user);  
 //查询指定id用户  
 User user1=userMapper.selectById ( 1l );  
 //查询多个用户  
 List<User> userList=userMapper.selectBatchIds ( Arrays.*asList* (1,2,3) );  
}

@Test//查询某个或多个参数不为空的用户  
void select(){  
 QueryWrapper<User> queryWrapper=new QueryWrapper <> ( );  
 queryWrapper  
 .isNotNull ( "name" )  
 .le ( "age",20 );  
 System.*out*.println ( userMapper.selectList ( queryWrapper ) );  
}

@Test//根据某个参数区间查询用户  
void test(){  
 QueryWrapper <User> queryWrapper=new QueryWrapper <> ( );  
 queryWrapper.between ( "age",20,30 );  
 System.*out*.println ( userMapper.selectList ( queryWrapper ) );  
}

@Test//模糊查询i  
void test1(){  
 QueryWrapper<User> queryWrapper=new QueryWrapper <> ( );  
 queryWrapper.like ( "name","i" );  
 System.*out*.println ( userMapper.selectList ( queryWrapper ) );  
}  
}

## 分页查询

//分页插件  
@Bean  
public MybatisPlusInterceptor mybatisPlusInterceptor1() {  
 MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();  
 interceptor.addInnerInterceptor(new PaginationInnerInterceptor ( DbType.*H2*));  
 return interceptor;  
}

测试代码，#页数#每页的尺寸

Page<User> page=new Page <> ( 2,3 );  
userMapper.selectPage ( page,null);  
System.*out*.println ( page.getRecords () );

## 删除测试

@Test  
void delete(){  
 //根据id删除用户  
 userMapper.deleteBatchIds ( Arrays.*asList* ( 1553000769238630402l,1553000769238630403l ) );  
 //根据条件删除  
 HashMap<String,Object> map=new HashMap <> ( );  
 map.put ( "name","杰哥" );  
 userMapper.deleteByMap ( map );  
}

逻辑删除（类似回收站操作）

//数据库添加字段实体类添加字段注解

@TableLogic  
private Integer deleted;

Application.properties更新配置

mybatis-plus.global-config.db-config.logic-delete-field=1  
mybatis-plus.global-config.db-config.logic-not-delete-value=0

## 插入测试

@Test  
void insert(){  
 User user=new User ();  
 user.setName ( "杰哥" );  
 user.setAge ( 18 );  
 userMapper.insert (user );  
 System.*out*.println (user);  
}

## 更新测试

@Test  
void change(){  
 User user = new User ();  
 user.setId (2l);  
 user.setName ( "张三" );  
 userMapper.updateById ( user );  
}

## 排序测试

@Test//排序  
void test2(){  
 QueryWrapper<User> queryWrapper=new QueryWrapper <> ( );  
 queryWrapper.orderByAsc ( "age" );  
 userMapper.selectList ( queryWrapper );  
}

## 数据库数据生成时间与更新时间

方法一：数据库直接设置

方法二：代码级别

//构建实体类参数

@TableField(fill= FieldFill.*INSERT*)  
private Date createTime;  
@TableField(fill = FieldFill.*INSERT\_UPDATE*)  
private Date updateTime;

编写处理器处理这个注解

@Component//给spring托管  
public class ObjectHandler implements MetaObjectHandler{  
 @Override  
 public void insertFill(MetaObject metaObject) {  
 //源码setFieldValByName(String fieldName, Object fieldVal, MetaObject metaObject)  
 this.setFieldValByName ( "createTime" ,new Date ( ),metaObject);  
 }  
 @Override  
 public void updateFill(MetaObject metaObject) {  
 this.setFieldValByName ( "updateTime",new Date ( ),metaObject );  
 }  
}

## 乐观锁

取出获取当前version，更新时带上当前version，更新成功版version+1，banbenha

数据库中添加version字段 默认版本是1

实体类中添加字段

@Version  
private Integer version;

配置插件

@MapperScan("com.jie.mapper")  
@EnableTransactionManagement  
@Configuration//配置类  
public class MybatisPlusConfig {  
 //注册乐观锁插件  
 @Bean  
 public MybatisPlusInterceptor mybatisPlusInterceptor() {  
 MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();  
 interceptor.addInnerInterceptor(new OptimisticLockerInnerInterceptor ());  
 return interceptor;  
 }  
}

测试乐观锁

@Test//测试乐观锁  
void MybatisPlusInterceptor(){  
 User user=userMapper.selectById ( 1l );  
 user.setName ( "王杰" );  
 user.setAge ( 1100 );  
 user.setEmail ( "2231817@qq.com" );  
 //模拟线程插队，user会操作失败  
 User user1=userMapper.selectById ( 1l );  
 user1.setName ( "阿凡达" );  
 userMapper.updateById ( user1 );  
 userMapper.updateById ( user );  
}

## 代码生成器

导入依赖

<dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-boot-starter</artifactId>  
 <version>3.4.1</version>  
</dependency>  
<dependency>  
 <groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-generator</artifactId>  
 <version>3.4.1</version>  
</dependency>  
<dependency>  
 <groupId>mysql</groupId>  
 <artifactId>mysql-connector-java</artifactId>  
</dependency>  
<dependency>  
 <groupId>org.apache.velocity</groupId>  
 <artifactId>velocity-engine-core</artifactId>  
 <version>2.3</version>  
</dependency>  
<dependency>  
 <groupId>io.swagger</groupId>  
 <artifactId>swagger-annotations</artifactId>  
 <version>1.5.22</version>  
</dependency>

配置数据库

server.port=8089  
#配置日志  
mybatis-plus.configuration.log-impl=org.apache.ibatis.logging.commons.JakartaCommonsLoggingImpl  
spring.datasource.url=jdbc:mysql://localhost:3306/car\_data?useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8&useSSL=false  
spring.datasource.username=wangjie  
spring.datasource.password=19961016wj  
spring.datasource.driver-class-name=com.mysql.jdbc.Driver

代码生成

//代码自动生成器  
public class JieCode {  
 public static void main(String[] args) {  
 //新建一个代码生成器对象  
 AutoGenerator autoGenerator=new AutoGenerator ();  
 //配置策略  
  
 //全局配置  
 GlobalConfig globalConfig=new GlobalConfig ();  
 globalConfig.setOutputDir ( "C:\\Java\_project\\spring\_project\\auto\_create\\src\\main\\java");  
 globalConfig.setAuthor ( "杰哥" );  
 globalConfig.setOpen ( false );//是否打开资源管理器  
 globalConfig.setFileOverride ( false );  
 globalConfig.setSwagger2 ( true );  
 autoGenerator.setGlobalConfig ( globalConfig );  
 //数据源配置  
 DataSourceConfig dataSourceConfig = new DataSourceConfig ();  
 dataSourceConfig.setUrl ( "jdbc:mysql://localhost:3306/car\_data?useUnicode=true&characterEncoding=utf8&serverTimezone=GMT%2B8&useSSL=false" );  
 dataSourceConfig.setDriverName ( "com.mysql.cj.jdbc.Driver" );  
 dataSourceConfig.setUsername ( "wangjie" );  
 dataSourceConfig.setPassword ( "19961016wj" );  
 dataSourceConfig.setDbType ( DbType.*MYSQL* );  
 autoGenerator.setDataSource ( dataSourceConfig );  
 //包配置  
 PackageConfig packageConfig = new PackageConfig ();   
 packageConfig.setModuleName ("blog");  
 packageConfig.setParent ( "com.jie" );  
 packageConfig.setEntity ( "pojo" );  
 packageConfig.setMapper ( "mapper" );  
 packageConfig.setService ( "service" );  
 packageConfig.setController ( "controller" );  
 autoGenerator.setPackageInfo ( packageConfig );  
 //策略配置  
 StrategyConfig strategy = new StrategyConfig();  
 strategy.setInclude("user");  
 strategy.setNaming( NamingStrategy.*underline\_to\_camel*);//下划线转驼峰命名  
 strategy.setColumnNaming(NamingStrategy.*underline\_to\_camel*);;  
 strategy.setEntityLombokModel(true);//是否使用lombok开启注解  
 strategy.setLogicDeleteFieldName ( "deleted" );//逻辑删除  
  
 //自动填充配置  
 TableFill createTime =new TableFill ( "create\_time", FieldFill.*INSERT* );  
 TableFill updateTime =new TableFill ( "update\_time", FieldFill.*INSERT\_UPDATE* );  
 ArrayList <TableFill> tableFills = new ArrayList <> ();  
 tableFills.add ( createTime );  
 tableFills.add ( updateTime );  
 strategy.setTableFillList ( tableFills);  
  
 strategy.setVersionFieldName ( "version" ); //乐观锁配置  
 autoGenerator.setStrategy ( strategy );  
 //执行  
 autoGenerator.execute ();  
 }  
  
}