Mybatis-plus

1. 入门

1.1 起步

```
<!--lombok - 代码去冗余工具-->
<dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
</dependency>

<!--mysql支持-->
<dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.13</version>
</dependency>

<dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
        <version>3.0.1</version>
</dependency>
```

spring-boot配置文件：

```
spring:
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: jdbc:mysql://localhost:3306/4mybatis_plus?seUnicode=true&characterEncoding=UTF-
8&serverTimezone=UTC
    username: root
    password: 3
```

添加@MapperScan注解：

```
import org.mybatis.spring.annotation.MapperScan;

@SpringBootApplication
@MapperScan("com.example.demo.multi.springBoot.mybatisPlus.mapper")
public class DemoMultiSpringBootApplication {

        public static void main(String[] args) {
                SpringApplication.run(DemoMultiSpringBootApplication.class, args);
        }
}
```

建立**Mapper.java：

```
import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.example.demo.multi.springBoot.mybatisPlus.entity.User;

public interface UserMapper extends BaseMapper<User> {

}
```

建立实体类：

```
import lombok.Data;

@Data // lombok工具
public class User {
    private Long id;
    private String name;
    private Integer age;
    private String email;
}
```

1.2 扩展配置

```
mybatis-plus:
  config-location: classpath:mybatis-config.xml
  mapper-locations: classpath*:mybatis/*.xml
  type-aliases-package: com.baomidou.mybatisplus.samples.quickstart.entity
  type-aliases-super-type: java.lang.Object
  type-handlers-package: com.baomidou.mybatisplus.samples.quickstart.handler
```

1.2.1 typeAliasesSuperType

该配置请和 typeAliasesPackage 一起使用，如果配置了该属性，则仅仅会扫描路径下以该类作为父类的域对象 。

1.2.2 checkConfigLocation ?

- 类型：`boolean`
- 默认值：`false`

作用，没有理解：

> 启动时是否检查 MyBatis XML 文件的存在，默认不检查

```
mybatis-plus:
  check-config-location: false
```

1.2.3 executorType

通过该属性可指定 MyBatis 的执行器，MyBatis 的执行器总共有三种：

- ExecutorType.SIMPLE：该执行器类型不做特殊的事情，为每个语句的执行创建一个新的预处理语句（PreparedStatement）
- ExecutorType.REUSE：该执行器类型会复用预处理语句（PreparedStatement）
- ExecutorType.BATCH：该执行器类型会批量执行所有的更新语句

```
mybatis-plus:
  executor-type: simple
```

1.2.4 configurationProperties ?

指定外部化 MyBatis Properties 配置，通过该配置可以抽离配置，实现不同环境的配置部署

```
mybatis-plus:
  configuration-properties: classpath:mybatis/config.properties
```

1.2.5 mybatis原生配置

1.2.6 懒加载

1.2.7 autoMappingBehavior

- 类型：`AutoMappingBehavior`
- 默认值：`partial`，原生mybatis也是默认的这个值

MyBatis 自动映射策略，关于介绍，可以看这里，此属性有3个值：

- AutoMappingBehavior.NONE：不启用自动映射
- AutoMappingBehavior.PARTIAL：只对非嵌套的 resultMap 进行自动映射
- AutoMappingBehavior.FULL：对所有的 resultMap 都进行自动映射

如果开启了自动映射，每个resultMap都可以单独关闭自动映射：

```
<resultMap id="userResultMap" type="User" autoMapping="false">
    <result property="password" column="hashed_password"/>
</resultMap>
```

1.2.8 autoMappingUnknownColumnBehavior

自动映射时**未知列**或**未知属性**处理策略，总共有 3 种可选值：

- AutoMappingUnknownColumnBehavior.NONE：不做任何处理 (默认值)
- AutoMappingUnknownColumnBehavior.WARNING：以日志的形式打印相关警告信息
- AutoMappingUnknownColumnBehavior.FAILING：当作映射失败处理，并抛出异常和详细信息

```
mybatis-plus:
  configuration:
    auto-mapping-unknown-column-behavior: none
```

1.2.9 callSettersOnNulls

作用：在返回类型为map的时候特别有用，即使字段值为空，结果（map）中也会包含字段对应的（map的）key。

1.3 全局配置

1.3.1 refresh

- 类型：`boolean`
- 默认值：`false`

是否自动刷新 Mapper 对应的 XML 文件，默认不自动刷新。如果配置了该属性，Mapper 对应的 XML 文件会自动刷新，更改 XML 文件后，无需再次重启工程，由此节省大量时间。

1.3.2 columnLike

- 类型：`boolean`
- 默认值：`false`

plus自带，是否开启 LIKE 查询，即根据 entity 自动生成的 where 条件中 String 类型字段 是否使用 LIKE，默认不开启。

1.3.4 logicDeleteValue

逻辑删除：当使用plus提供的deleteById之类的删除时，不是物理删除，而是自动使用逻辑删除；

直接上代码：

```
import com.baomidou.mybatisplus.core.injector.ISqlInjector;
import com.baomidou.mybatisplus.extension.injector.LogicSqlInjector;
import org.springframework.context.annotation.Bean;
```

```
import org.springframework.context.annotation.Configuration;

@Configuration
public class MyBatisPlusConfiguration {

    @Bean
    public ISqlInjector sqlInjector() {
        return new LogicSqlInjector();
    }
}
```

```
import com.baomidou.mybatisplus.annotation.TableLogic;

public class User {
    private Integer id;
    private String name;
    private Integer age;
    private String email;
    @TableLogic  // 关键注解
    private Boolean delFlag;
}
```

yml配置：

```
mybatis-plus:
  global-config:
    db-config:
      logic-delete-value: true
      logic-not-delete-value: false
```

数据库字段：`del_flag`

1.4 复杂配置

1.4.1 添加xml配置文件

```
mybatis-plus:
  config-location: classpath:mybatis-config.xml
```

1.4.2 Mapper.xml文件配置

```
mybatis-plus:
  mapper-locations: classpath*:mybatis/*.xml
```

1.4.3 别名父类

该配置请和 typeAliasesPackage 一起使用，如果配置了该属性，则仅仅会扫描路径下以该类作为父类的域对象 。

1.4.4 typeHandlersPackage

1.4.5 typeEnumsPackage

枚举类 扫描路径，如果配置了该属性，会将路径下的枚举类进行注入，让实体类字段能够简单快捷的使用枚举属性，具体使用请结合 枚举注入 查看。

1.4.6 checkConfigLocation

启动时是否检查 MyBatis XML 文件的存在，默认不检查。

```
mybatis-plus:
  check-config-location: false
```

1.4.7 设置执行器

通过该属性可指定 MyBatis 的执行器，MyBatis 的执行器总共有三种：

- ExecutorType.SIMPLE：该执行器类型不做特殊的事情，为每个语句的执行创建一个新的预处理语句（PreparedStatement）
- ExecutorType.REUSE：该执行器类型会复用预处理语句（PreparedStatement）
- ExecutorType.BATCH：该执行器类型会批量执行所有的更新语句

```
mybatis-plus:
  executor-type: simple
```

1.4.8 mybatis.properties配置

```
mybatis-plus:
  configuration-properties: classpath:mybatis/config.properties
```

1.4.9 原生配置

configuration

1.4.10 全局策略（globalConfig）

1.4.11 懒加载

aggressiveLazyLoading #

```
mybatis-plus:
  configuration:
    aggressive-lazy-loading: true
```

当设置为 true 的时候，懒加载的对象可能被任何懒属性全部加载，否则，每个属性都按需加载。需要和 lazyLoadingEnabled 一起使用。

1.4.12 结果集映射

MyBatis 自动映射策略，通过该配置可指定 MyBatis 是否并且如何来自动映射数据表字段与对象的属性，总共有 3 种可选值：

- AutoMappingBehavior.NONE：不启用自动映射
- AutoMappingBehavior.PARTIAL：只对非嵌套的 resultMap 进行自动映射
- AutoMappingBehavior.FULL：对所有的 resultMap 都进行自动映射

```
mybatis-plus:
  configuration:
    auto-mapping-behavior: partial
```

1.4.13 未知列和属性映射策略

- AutoMappingUnknownColumnBehavior.NONE：不做任何处理 (默认值)
- AutoMappingUnknownColumnBehavior.WARNING：以日志的形式打印相关警告信息
- AutoMappingUnknownColumnBehavior.FAILING：当作映射失败处理，并抛出异常和详细信息

```
mybatis-plus:
  configuration:
    auto-mapping-unknown-column-behavior: none
```

1.4.14 缓存全局开关

全局地开启或关闭配置文件中的所有映射器已经配置的任何缓存，默认为 true

```
mybatis-plus:
  configuration:
    cache-enabled: true
```

1.4.15 数据库类型配置

```
mybatis-plus:
  global-config:
    db-config:
      db-type: mysql
```

2. 代码生成器

3. 打印SQL日志

3.1 spring-boot

```
logging:
  level:
    root: debug
```

4. ActiveRecord模式

5. 分页插件

5.1 SpringBoot

```
//Spring boot方式
@EnableTransactionManagement
@Configuration
@MapperScan("com.baomidou.cloud.service.*.mapper*")
public class MybatisPlusConfig {

    /**
     * 分页插件
     */
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        return new PaginationInterceptor();
    }
}
```

5.2 xml配置方式

```
<!-- spring xml 方式 -->
<plugins>
    <plugin interceptor="com.baomidou.mybatisplus.plugins.PaginationInterceptor">
        <property name="sqlParser" ref="自定义解析类、可以没有" />
```

```
        <property name="dialectClazz" value="自定义方言类、可以没有" />
    </plugin>
</plugins>
```

5.3 使用示例

```
public interface UserMapper{//可以继承或者不继承BaseMapper
    /**
     * <p>
     * 查询 : 根据state状态查询用户列表，分页显示
     * 注意!!: 如果入参是有多个,需要加注解指定参数名才能在mapper.xml中取值
     * </p>
     *
     * @param page 分页对象,mapper.xml中可以从里面进行取值,传递参数 Page 即自动分页,必须放在第一位(你可以继承Page实现自己的分页对象)
     * @param state 状态
     * @return 分页对象
     */
    IPage<User> selectPageVo(Page page, @Param("state") Integer state);
}
```

```
<select id="selectPageVo" resultType="com.baomidou.cloud.entity.UserVo">
    SELECT id,name FROM user WHERE state=#{state}
</select>
```

```
public IPage<User> selectUserPage(Page<User> page, Integer state) {
    // 不进行 count sql 优化，解决 MP 无法自动优化 SQL 问题，这时候你需要自己查询 count 部分
    // page.setOptimizeCountSql(false);
    // 当 total 为非 0 时(默认为 0),分页插件不会进行 count 查询
    // 要点!! 分页返回的对象与传入的对象是同一个
    return userMapper.selectPageVo(page, state));
}
```

6. 缓存

6.1 缓存与序列化

如果开启了缓存，但查询结果（实体类）没有实现序列化接口，会报错（NotSerializableException）；

7. 热加载配置

- 方式一：

  创建MybatisMapperRefresh对象，并丢给spring容器管理；

- 方式二：

  直接使用refresh配置 #

  ```
  mybatis-plus:
    global-config:
      refresh: true
  ```

  简单粗暴

扩展

项目根目录

```
System.getProperty("user.dir")
```