MyBatis-Plus

Pom.xml

```
<dependencies>
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <optional>true</optional>
    </dependency>
    <dependency>
        <groupId>com.baomidou</groupId>
        <artifactId>mybatis-plus-boot-starter</artifactId>
        <version>3.3.0</version>
    </dependency>
    <!--  -->
        <dependency>
                  <groupId>com.baomidou</groupId>
                  <artifactId>dynamic-datasource-spring-boot-starter</artifactId>
                  <version>3.3.0</version>
        </dependency>
</dependencies>
```

application.yaml

```
spring:
  profiles: test
  autoconfigure:
    exclude: com.alibaba.druid.spring.boot.autoconfigure.DruidDataSourceAutoConfigure
  datasource:
    druid:
      stat-view-servlet:
        loginUsername: admin
        loginPassword: 123456
    dynamic:
      primary: master
      druid:
        initial-size: 1
        max-active: 20
        min-idle: 1
        max-wait: 60000
        time-between-eviction-runs-millis: 60000
        min-evictable-idle-time-millis: 300000
        test-while-idle: true
        test-on-borrow: true
        test-on-return: false
        validation-query: SELECT 'x'
        pool-prepared-statements: true
        filters: stat,wall
      datasource:
        master:
          username: dev
          password: 4Lc3nvcYiyDxKDx7
          driver-class-name: com.mysql.jdbc.Driver
          url: jdbc:mysql://127.0.0.1:3306/mms?useUnicode=true&useSSL=false&characterEncoding=utf8
        slave_1:
          username: dev
          password: 4Lc3nvcYiyDxKDx7
          driver-class-name: com.mysql.jdbc.Driver
          url: jdbc:mysql://127.0.0.1:3306/mms?useUnicode=true&useSSL=false&characterEncoding=utf8
```

mybatis-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.
dtd">
<configuration>

        <settings>
                <!--  -->
                <setting name="cacheEnabled" value="true"/>

                <!--  -->
                <setting name="lazyLoadingEnabled" value="true"/>

                <!--  -->
                <setting name="aggressiveLazyLoading" value="true"/>

                <!-- sql   () default:true -->
                <setting name="multipleResultSetsEnabled" value="true"/>

                <!--  () default:true -->
                <setting name="useColumnLabel" value="true"/>

                <!-- JDBC true  default:false  -->
                <setting name="useGeneratedKeys" value="false"/>

                <!--  MyBatis   NONEPARTIAL:  FULL:  -->
                <setting name="autoMappingBehavior" value="PARTIAL"/>

                <!--   SIMPLE:  REUSE: prepared statementsBATCH:   -->
                <setting name="defaultExecutorType" value="SIMPLE"/>

                <!--  -->
                <setting name="mapUnderscoreToCamelCase" value="true"/>

                <!--  session:  statement: ( ) defalut:session -->
                <setting name="localCacheScope" value="SESSION"/>

                <!-- JDBC, ,default:OTHER -->
                <setting name="jdbcTypeForNull" value="NULL"/>
        </settings>
</configuration>
```

Entity

```
@Data
@EqualsAndHashCode(callSuper = true)
@Accessors(chain = true)
@TableName("meeting")
@ApiModel(value="Meeting", description="")
public class Meeting extends BaseEntity {

    @ApiModelProperty(value = "")
    private String name;

    @ApiModelProperty(value = "")
    private Integer status;

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    @ApiModelProperty(value = "")
    private LocalDateTime updateTime;

    @JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss")
    @ApiModelProperty(value = "")
    private LocalDateTime createTime;


    @ApiModelProperty(value = ":")
    private transient String userName;

    @ApiModelProperty(value = ":")
    private transient LocalDateTime expectStartDate;

    @ApiModelProperty(value = ":")
    private transient LocalDateTime expectEndDate;

    @ApiModelProperty(value = ":")
    private transient LocalDateTime createStartDate;

    @ApiModelProperty(value = ":")
    private transient LocalDateTime createEndDate;
}
```

Mapper

```
public interface MeetingMapper extends BaseMapper<Meeting> {
        /**
        *
        */
    Page<Meeting> findMeetingPage(Page<Meeting> meetingPage,
                                  @Param("meeting") Meeting meeting);
}
```

Mapper.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd" >
<mapper namespace="com.demo.mapper.MeetingMapper">

    <select id="findMeetingPage" resultType="com.demo.entity.Meeting">
        SELECT m.*,cur.name AS createName,our.name AS ownerName FROM mms_meeting m
        <if test="meeting != null and meeting.userName !='' and meeting.userName != null">
            LEFT JOIN mms_user cu ON m.creater_id = cu.id OR m.owner_id = cu.id
        </if>
        LEFT JOIN mms_user cur ON m.creater_id = cur.id
        LEFT JOIN mms_user our ON m.owner_id = our.id
        WHERE 1=1
        <if test="meeting != null and meeting.status != null">
            AND m.status=#{meeting.status}
        </if>
        <if test="meeting != null and meeting.type != null">
            AND m.type=#{meeting.type}
        </if>
        <if test="meeting != null and meeting.name !='' and meeting.name != null">
            AND m.name LIKE concat(#{meeting.name},'%')
        </if>
        <if test="meeting != null and meeting.userName !='' and meeting.userName != null">
            AND cu.name LIKE concat(#{meeting.userName},'%')
        </if>
        <if test="meeting != null and meeting.expectStartDate != null">
            AND m.start_time >= #{meeting.expectStartDate}
        </if>
        <if test="meeting != null and meeting.expectEndDate != null">
            AND m.start_time <= #{meeting.expectEndDate}
        </if>
        <if test="meeting != null and meeting.createStartDate != null">
            AND m.create_time >= #{meeting.createStartDate}
        </if>
        <if test="meeting != null and meeting.createEndDate != null">
            AND m.create_time <= #{meeting.createEndDate}
        </if>
        ORDER BY m.create_time DESC
    </select>

</mapper>
```

IService

```java
public interface IMeetingService extends IBaseService<Meeting> {

    Page<Meeting> findMeetingPage(Page<Meeting> meetingPage, Meeting meeting);

}
```

ServiceImpl

```java
@Slf4j
@Service
public class MeetingServiceImpl extends BaseServiceImpl<MeetingMapper, Meeting> implements IMeetingService {
    @Resource
    private MeetingMapper meetingMapper;

    public Page<Meeting> findMeetingPage(Page<Meeting> meetingPage,Meeting meeting) {
        return meetingMapper.findMeetingPage(meetingPage,meeting);
    }
}
```

LambdaQuery

```
@Resource
private IMeetingService meetingService;

Meeting meeting = meetingService.lambdaQuery()
                                .eq(Meeting::getCode,code)
                                .last("limit 1")
                                .one();

Integer meetingNum = meetingService.lambdaQuery()
                                .eq(Meeting::getType, 1)
                                .gt(Meeting::getCreateTime, "2019-01-01 00:00:00")
                                .lt(Meeting::getCreateTime, "2019-01-01 23:59:59")
                                .count();
```

```
@EnableTransactionManagement
@Configuration
@MapperScan("com.demo.mapper")
public class MybatisPlusConfig {
    @Bean
    public PaginationInterceptor paginationInterceptor() {
        return new PaginationInterceptor();
    }
}
```

ID

```
@JsonInclude(JsonInclude.Include.NON_NULL)
@Data
public class BaseEntity implements Serializable {
    @JsonSerialize(using= ToStringSerializer.class)
    @ApiModelProperty(value = "ID",dataType = "java.lang.String")
    @TableId(type = IdType.ID_WORKER)
    protected Long id;
}
```

```
@Slf4j
@Aspect
@Component
public class DataSourceAspect {
    @Resource
    private DynamicRoutingDataSource dynamicRoutingDataSource;

    @Before("execution(* com.demo.service.*.select*(..)) " +
            "|| execution(* com.demo.service.*.get*(..))" +
            "|| execution(* com.demo.service.*.find*(..))" +
            "|| execution(* com.demo.service.*.list*(..))" +
            "|| execution(* com.demo.service.*.count*(..))" +
            "|| execution(* com.demo.service.*.page*(..))" +
            "|| execution(* com.demo.service.*.lambda*(..))" +
            "|| execution(* com.demo.service.*.query*(..))")
    public void setReadDataSourceType() {
        Set<String> keySet = dynamicRoutingDataSource.getCurrentDataSources().keySet();
        List<String> slaveList = new ArrayList<>(keySet);
        slaveList.remove("master");
        Collections.shuffle(slaveList);

        dynamicRoutingDataSource.setPrimary(slaveList.get(0));
        log.debug("dataSource"+slaveList.get(0));
    }

    @Before("execution(* com.demo.service.*.insert*(..)) " +
            "|| execution(* com.demo.service.*.update*(..)) " +
            "|| execution(* com.demo.service.*.delete*(..)) " +
            "|| execution(* com.demo.service.*.save*(..))")
    public void setWriteDataSourceType() {
        dynamicRoutingDataSource.setPrimary("master");
        log.debug("dataSourcemaster");
    }
}
```

```
public class CodeGenerator {
    /**
     * <p>
     *
     * </p>
     */
    public static String scanner(String tip) {
        Scanner scanner = new Scanner(System.in);
        StringBuilder help = new StringBuilder();
        help.append("" + tip + "");
        System.out.println(help.toString());
        if (scanner.hasNext()) {
            String ipt = scanner.next();
            if (StringUtils.isNotEmpty(ipt)) {
                return ipt;
            }
        }
        throw new MybatisPlusException("" + tip + "");
    }

    public static void main(String[] args) {
        //
        AutoGenerator mpg = new AutoGenerator();

        //
```

```
        GlobalConfig gc = new GlobalConfig();
        String projectPath = System.getProperty("user.dir");
        gc.setOutputDir(projectPath+"/api/src/main/java");
        gc.setAuthor("your name");
        gc.setOpen(false);
        gc.setSwagger2(true);
        gc.setFileOverride(true);
        mpg.setGlobalConfig(gc);

        //
        DataSourceConfig dsc = new DataSourceConfig();
        dsc.setDbType(DbType.MYSQL);
        dsc.setUrl("jdbc:mysql://127.0.0.1:3306/mms?useUnicode=true&useSSL=false&characterEncoding=utf8");
        dsc.setDriverName("com.mysql.jdbc.Driver");
        dsc.setUsername("work");
        dsc.setPassword("123456");
        dsc.setTypeConvert(new MySqlTypeConvert(){
            @Override
            public IColumnType processTypeConvert(GlobalConfig globalConfig, String fieldType){
                                //
                if (fieldType.contains("tinyint(1)")){
                                        return DbColumnType.INTEGER;
                                }
                return super.processTypeConvert(globalConfig,fieldType);
            }

        });
        mpg.setDataSource(dsc);

        //
        PackageConfig pc = new PackageConfig();
        //pc.setModuleName(scanner(""));
        pc.setParent("com.demo");
        mpg.setPackageInfo(pc);

        //
        InjectionConfig cfg = new InjectionConfig() {
            @Override
            public void initMap() {
                // to do nothing

            }
        };

        //
        TemplateConfig templateConfig = new TemplateConfig();
        templateConfig.setXml(null);
        mpg.setTemplate(templateConfig);

        //
        StrategyConfig strategy = new StrategyConfig();
        strategy.setNaming(NamingStrategy.underline_to_camel);
        strategy.setColumnNaming(NamingStrategy.underline_to_camel);
        strategy.setTablePrefix("mms_");//
        strategy.setEntityLombokModel(true);
        strategy.setRestControllerStyle(true);
                //
        strategy.setSuperEntityClass("com.demo.commons.entity.BaseEntity");
        strategy.setSuperControllerClass("com.demo.commons.controller.BaseController");
        strategy.setSuperServiceClass("com.demo.commons.service.IBaseService");
        strategy.setSuperServiceImplClass("com.demo.commons.service.BaseServiceImpl");

        strategy.setInclude(scanner(""));
        strategy.setSuperEntityColumns("id");
        strategy.setControllerMappingHyphenStyle(true);
        strategy.setSuperEntityColumns("id");
        mpg.setStrategy(strategy);
        mpg.execute();
    }
}
```