mybatis-plus是基于mybatis，相对于mybatis，有许多特性比较好用。比如：

* 分页查询
* 表字段自动转换为实体类属性等

使用mybatis-plus与Spring Data JPA有点类似。

### 1、添加pom引用

<dependency>

<groupId>com.baomidou</groupId>  
 <artifactId>mybatis-plus-boot-starter</artifactId>  
 <version>3.0.7.1</version>  
</dependency>

### 2、配置

官方提供了[基于spring boot的配置](https://baomidou.oschina.io/mybatis-plus-doc/" \l "/spring-boot)。其中column-underline: true特别好用，会自动将下划线格式的表字段，转换为以驼峰格式命名的属性。

mybatis-plus:

global-config:  
 db-config:  
 id-type: *auto* field-strategy: *not\_empty  
 #驼峰下划线转换* column-underline: true  
 *#逻辑删除配置* logic-delete-value: 0  
 logic-not-delete-value: 1  
 db-type: *mysql* refresh: true  
 configuration:  
 map-underscore-to-camel-case: true  
 cache-enabled: false

### 3、使用

1）创建实体类，与JPA特别相似，其中

* @TableName注解是为了指定此实体类对应数据库的哪一张表；
* @TableId指的是注解，type属性指定的是该主键自增的方式，AUTO代表自增，UUID代表使用UUID增加主键；
* @TableField，此注解主要用在非主键实体属性上。下面是此注解的所有属性：

|  |  |
| --- | --- |
| 值 | 描述 |
| value | 字段值（驼峰式命名，可省略） |
| update | 预处理set字段自定义注入 |
| condition | 预处理WHERE实体条件自定义运算规则 |
| el | 详看注释说明 |
| exist | 是否为数据库表字段（默认true存在，false不存在） |
| strategy | 字段验证 |
| fill | 字段填充标记（FieldFill，配合自动填充使用） |

@TableName("test")

public class User {  
  
 @TableId(value = "id",type = IdType.*AUTO*)  
 private Integer id;  
  
 private String name;  
  
 getter、setter...

}

2）创建mapper，与JPA有相像之处。使用@Mapper为了把mapper这个Dao交给Spring管理，且不可以再写mapper映射文件；继承BaseMapper可以方便使用mybatis-plus方法，泛型需要写对用的实体类。此处直接使用select \* from，mybatis-plus会根据实体类自动驼峰转下划线映射到表的字段中。

@Repository

public interface UserMapper extends BaseMapper<User> {  
  
 @Select("select \* from t\_user where id=#{id} and deleted\_at is null")  
 User queryByIdAndDeletedAtIsNull(@Param("id") Integer id);  
}

### 4、条件构造器

1）查询单挑数据（非主键查询）

只要在一个实体内写入属性，然后再用mybatis-plus自带的selectOne()方法即可查询（注意：查询的限制条件必须此条数据是唯一的，不然查询后会报错）。不建议使用。

PatrolJobLogEntity jobLogEntity = new PatrolJobLogEntity();

jobLogEntity.setJobId(12);

jobLogEntity.setTaskFinished(30);

patrolJobLogMapper.selectOne(jobLogEntity);

2）查询数量，含有拼接方式查询

使用mybatis-plus条件查询需要使用到EntityWrapper构造器，它能添加查询条件，如：.eq(“job\_id”,13)表示查询job\_id=13的记录，isNull(“deleted\_at”)表示deleted\_at必须为null（注意：此处查询的条件是使用的是数据库表中的字段名称）。

EntityWrapper<PatrolTaskEntity> w = new EntityWrapper<>().eq("job\_id",13).isNull("deleted\_at");

List<PatrolTaskEntity> tasks = patrolTaskMapper.selectCount(w);

3）查询列表

也需要使用EntityWrapper构造器，使用mybatis-plus的selectList()方法。

EntityWrapper<PatrolTaskEntity> w = new EntityWrapper<>().eq("job\_id",13).isNull("deleted\_at");

List<PatrolTaskEntity> tasks = patrolTaskMapper.selectList(w);

### 5、分页

1）添加配置文件，表示开启mybatis-plus分页功能

@EnableTransactionManagement

@Configuration  
public class MybatisPlusConfig {  
  
 @Bean  
 public PaginationInterceptor paginationInterceptor() {  
 return new PaginationInterceptor();  
 }  
}

2）写查询语句，此处的Pagination必须给出，不然无法完成分页功能。

@Mapper

public interface BannerLogMapper extends BaseMapper<BannerLogEntity> {

    @Select("select \* from t\_pub\_banner\_logs")

    List<BannerLogEntity> queryLogByPage(Pagination page);

}

3）进行分页查询代码块，只需要将要查询的第几页和页面大小写入Page对象中。然后进行查询，查询后分页属性(当前页，总页数，总条数)，是警告插件自动会写到传入page对象中。

Page<BannerLogEntity> page = new Page<>(pageNo,pageSize);

List<BannerLogEntity> bannerLogs = bannerLogMapper.queryLogByPage(page);

page.getTotal();

page.getCurrent();

page.getPages();