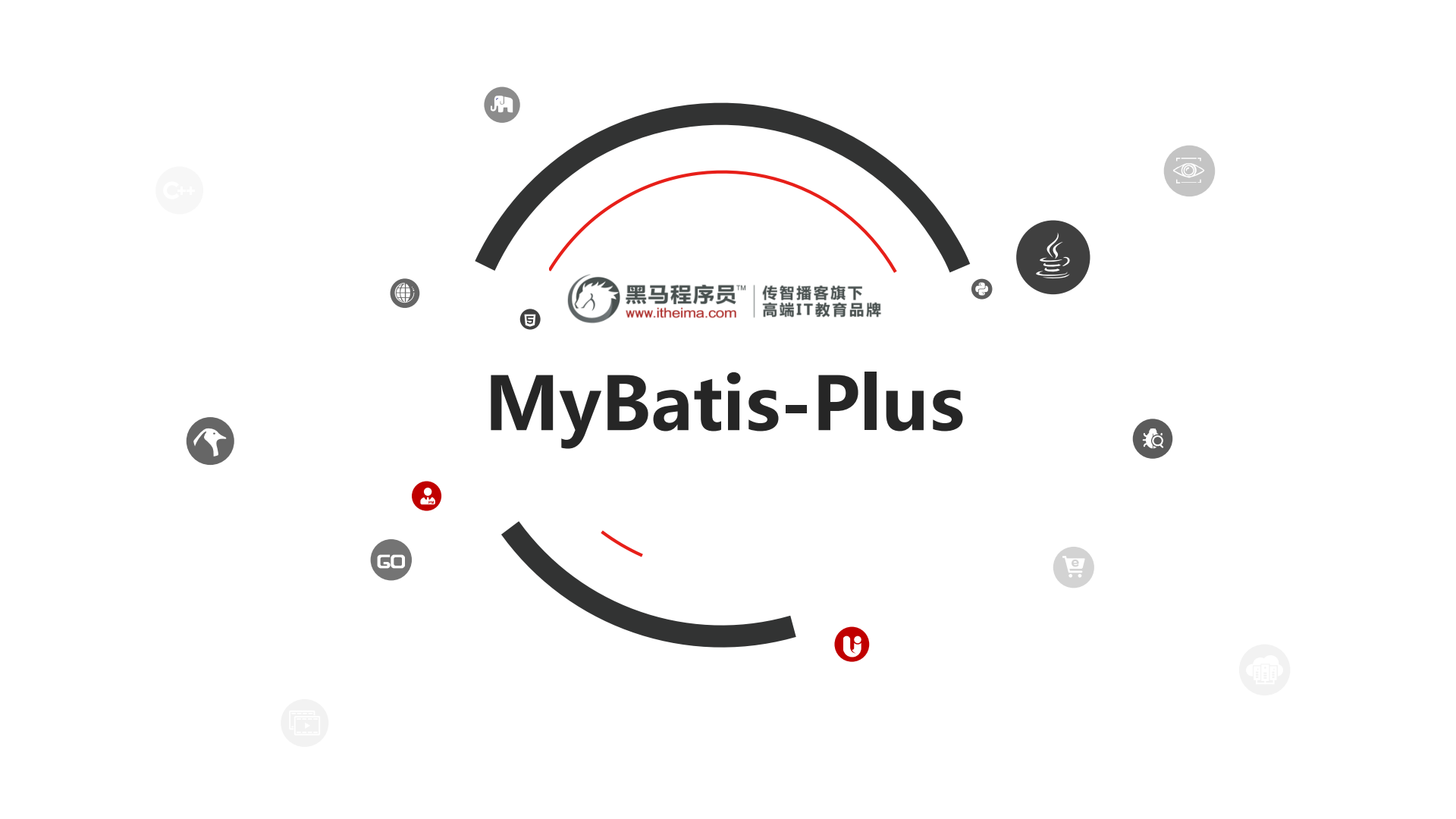

黑马程序员™
www.itheima.com
传智播客旗下
高端IT教育品牌
MyBatis-Plus

目录 Contents

目录 Contents

MyBatis-Plus 概述

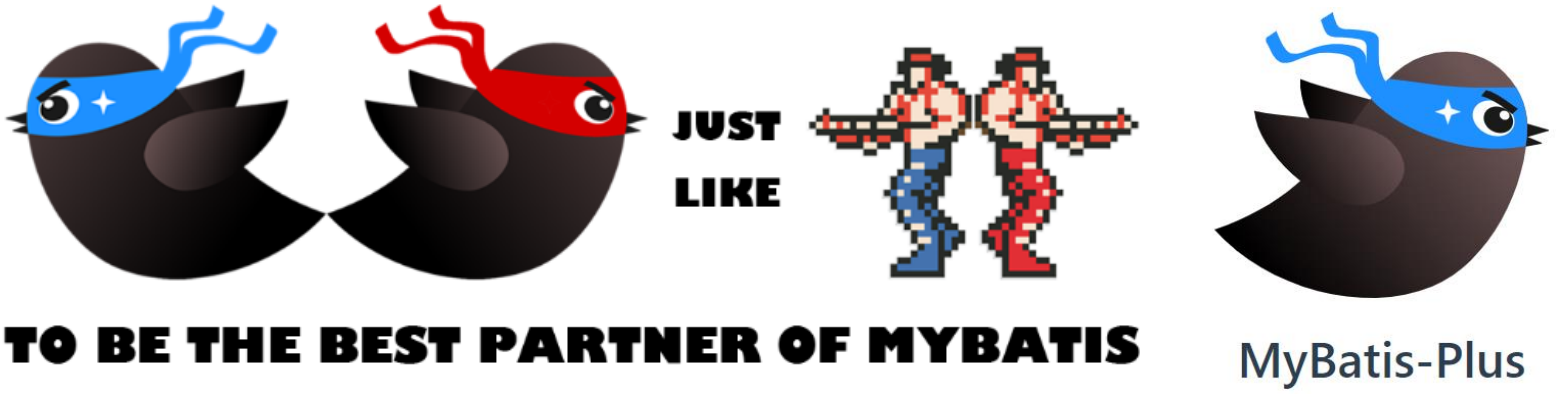
MyBatis-Plus 概述

MyBatis

MyBatis-Plus 概述

MyBatis-Plus 介绍

- MyBatis-Plus（简称 MP）是一个 MyBatis 的增强工具，在 MyBatis 的基础上只做增强不做改变，为简化开发、提高效率而生。
- 官网：https://mybatis.plus/ 或 https://mp.baomidou.com/



目录 Contents

MyBatis-Plus 快速入门

案例：需求

SpringBoot 整合 MyBatis-Plus，并实现根据Id查询功能。

案例：实现步骤

① 数据库环境准备
② 创建SpringBoot工程，引入MyBatis-Plus起步依赖
③ 编写DataSource相关配置
④ 编写mapper
⑤ 测试

目录 Contents

MyBatis-Plus CRUD

- 添加操作
- 修改操作
- 删除操作
- 查询操作

MyBatis-Plus 概述

添加操作

```java
// 插入一条记录
int insert(T entity);
```

MyBatis-Plus 概述

删除操作

```java
// 根据 entity 条件，删除记录
int delete(@Param(Constants.WRAPPER) Wrapper<T> wrapper);
// 删除（根据ID 批量删除）
int deleteBatchIds(@Param(Constants.COLLECTION) Collection<? extends Serializable> idList);
// 根据 ID 删除
int deleteById(Serializable id);
// 根据 columnMap 条件，删除记录
int deleteByMap(@Param(Constants.COLUMN_MAP) Map<String, Object> columnMap);
```

MyBatis-Plus 概述

修改操作

```java
// 根据 whereEntity 条件，更新记录
int update(@Param(Constants.ENTITY) T entity, @Param(Constants.WRAPPER) Wrapper<T> updateWrapper);
// 根据 ID 修改
int updateById(@Param(Constants.ENTITY) T entity);
```

MyBatis-Plus 概述

查询操作

```java
// 根据 ID 查询
T selectById(Serializable id);
// 根据 entity 条件，查询一条记录
T selectOne(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);

// 查询（根据ID 批量查询）
List<T> selectBatchIds(@Param(Constants.COLLECTION) Collection<? extends Serializable> idList);
// 根据 entity 条件，查询全部记录
List<T> selectList(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
// 查询（根据 columnMap 条件）
List<T> selectByMap(@Param(Constants.COLUMN_MAP) Map<String, Object> columnMap);
// 根据 Wrapper 条件，查询全部记录
List<Map<String, Object>> selectMaps(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
// 根据 Wrapper 条件，查询全部记录。注意： 只返回第一个字段的值
List<Object> selectObjs(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);

// 根据 entity 条件，查询全部记录（并翻页）
IPage<T> selectPage(IPage<T> page, @Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
// 根据 Wrapper 条件，查询全部记录（并翻页）
IPage<Map<String, Object>> selectMapsPage(IPage<T> page, @Param(Constants.WRAPPER) Wrapper<T> que
// 根据 Wrapper 条件，查询总记录数
Integer selectCount(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
```

目录 Contents

MyBatis-Plus 条件构造器

- 基本比较查询
- 逻辑查询
- 排序查询
- 模糊查询
- 指定查询字段
- LambdaQueryWrapper
- 删除&修改条件构造器

MyBatis-Plus 条件构造器

基本比较查询

- eq( ) : 等于 =
- ne( ) : 不等于 <>
- gt( ) : 大于 >
- ge( ) : 大于等于 >=
- lt( ) : 小于 <
- le( ) : 小于等于 <=
- between( ) : BETWEEN 值1 AND 值2
- notBetween( ) : NOT BETWEEN 值1 AND 值2
- in( ) : in
- notIn( ) : not in

MyBatis-Plus 条件构造器

逻辑查询

- or( )： 让紧接着下一个方法用or连接

MyBatis-Plus 条件构造器

模糊查询

- like
- notLike
- likeLeft
- likeRight

MyBatis-Plus 条件构造器

排序查询

- orderBy
- orderByAsc
- orderByDesc

MyBatis-Plus 条件构造器

select

- select：指定需要查询的字段

MyBatis-Plus 条件构造器

查询方法

```java
// 根据 ID 查询
T selectById(Serializable id);
// 根据 entity 条件，查询一条记录
T selectOne(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);

// 查询（根据ID 批量查询）
List<T> selectBatchIds(@Param(Constants.COLLECTION) Collection<? extends Serializable> idList);
// 根据 entity 条件，查询全部记录
List<T> selectList(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
// 查询（根据 columnMap 条件）
List<T> selectByMap(@Param(Constants.COLUMN_MAP) Map<String, Object> columnMap);
// 根据 Wrapper 条件，查询全部记录
List<Map<String, Object>> selectMaps(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
// 根据 Wrapper 条件，查询全部记录。注意： 只返回第一个字段的值
List<Object> selectObjs(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);

// 根据 entity 条件，查询全部记录（并翻页）
IPage<T> selectPage(IPage<T> page, @Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
// 根据 Wrapper 条件，查询全部记录（并翻页）
IPage<Map<String, Object>> selectMapsPage(IPage<T> page, @Param(Constants.WRAPPER) Wrapper<T> que
// 根据 Wrapper 条件，查询总记录数
Integer selectCount(@Param(Constants.WRAPPER) Wrapper<T> queryWrapper);
```

MyBatis-Plus 条件构造器

LambdaQueryWrapper

- 为了避免代码中出现硬编码，可以使用 **LambdaQueryWrapper** 完成条件构造

MyBatis-Plus 条件构造器

删除&修改条件构造器

```java
// 根据 entity 条件，删除记录
int delete(@Param(Constants.WRAPPER) Wrapper<T> wrapper);
// 删除（根据ID 批量删除）
int deleteBatchIds(@Param(Constants.COLLECTION) Collection<? extends Serializable> idList);
// 根据 ID 删除
int deleteById(Serializable id);
// 根据 columnMap 条件，删除记录
int deleteByMap(@Param(Constants.COLUMN_MAP) Map<String, Object> columnMap);
```

```java
// 根据 whereEntity 条件，更新记录
int update(@Param(Constants.ENTITY) T entity, @Param(Constants.WRAPPER) Wrapper<T> updateWrapper);
// 根据 ID 修改
int updateById(@Param(Constants.ENTITY) T entity);
```

目录 Contents

Mybatis-Plus Service封装

- Mybatis-Plus 为了开发更加快捷，对业务层也进行了封装，直接提供了相关的接口和实现类。我们在进行业务层开发时，可以继承它提供的接口和实现类，使得编码更加高效。

目录 Contents

代码生成器

- AutoGenerator 是 MyBatis-Plus 的代码生成器，通过 AutoGenerator 可以快速生成 Entity、Mapper、Mapper XML、Service、Controller 等各个模块的代码，极大的提升了开发效率。

传智播客旗下高端IT教育品牌