通用mapper

核心包

```
<dependency>
    <groupId>tk.mybatis</groupId>
    <artifactId>mapper-core</artifactId>
    <version>1.1.5</version>
</dependency>

<dependency>
    <groupId>tk.mybatis</groupId>
    <artifactId>mapper-extra</artifactId>
    <version>1.1.5</version>
</dependency>

<dependency>
    <groupId>tk.mybatis</groupId>
    <artifactId>mapper-base</artifactId>
    <version>1.1.5</version>
</dependency>
```

基础接口

Select

接口：`SelectMapper<T>` 方法：`List<T> select(T record);` 说明：根据实体中的属性值进行查询，查询条件使用等号

接口：`SelectByPrimaryKeyMapper<T>` 方法：`T selectByPrimaryKey(Object key);` 说明：根据主键字段进行查询，方法参数必须包含完整的主键属性，查询条件使用等号

接口：`SelectAllMapper<T>` 方法：`List<T> selectAll();` 说明：查询全部结果，select(null)方法能达到同样的效果

接口：`SelectOneMapper<T>` 方法：`T selectOne(T record);` 说明：根据实体中的属性进行查询，只能有一个返回值，有多个结果是抛出异常，查询条件使用等号

接口：`SelectCountMapper<T>` 方法：`int selectCount(T record);` 说明：根据实体中的属性查询总数，查询条件使用等号

Insert

接口：`InsertMapper<T>` 方法：`int insert(T record);` 说明：保存一个实体，null的属性也会保存，不会使用数据库默认值

接口：`InsertSelectiveMapper<T>` 方法：`int insertSelective(T record);` 说明：保存一个实体，null的属性不会保存，会使用数据库默认值

Update

接口：`UpdateByPrimaryKeyMapper<T>` 方法：`int updateByPrimaryKey(T record);` 说明：根据主键更新实体全部字段，null值会被更新

接口：`UpdateByPrimaryKeySelectiveMapper<T>` 方法：`int updateByPrimaryKeySelective(T record);` 说明：根据主键更新属性不为null的值

Delete

接口：`DeleteMapper<T>` 方法：`int delete(T record);` 说明：根据实体属性作为条件进行删除，查询条件使用等号

接口：`DeleteByPrimaryKeyMapper<T>` 方法：`int deleteByPrimaryKey(Object key);` 说明：根据主键字段进行删除，方法参数必须包含完整的主键属性

base组合接口

接口：`BaseSelectMapper<T>` 方法：包含上面Select的4个方法

接口：`BaseInsertMapper<T>` 方法：包含上面Insert的2个方法

接口：`BaseUpdateMapper<T>` 方法：包含上面Update的2个方法

接口：`BaseDeleteMapper<T>` 方法：包含上面Delete的2个方法

CRUD组合接口

接口：`BaseMapper<T>` 方法：继承了base组合接口中的4个组合接口，包含完整的CRUD方法

高级接口

批量插入数据

接口：`InsertListMapper<T>`

方法：`int insertList(List<? extends T> recordList)`

说明：传入集合，将代替传统的循环插入，提高效率，成功返回1，失败返回0

重点：这个接口是**tk.mybatis.mapper.additional.insert**包下的，此包下的这个方法没有主键策略，需要自己提前设置好主键，生成的sql语句如下，oracle并不支持下面的插入sql语句

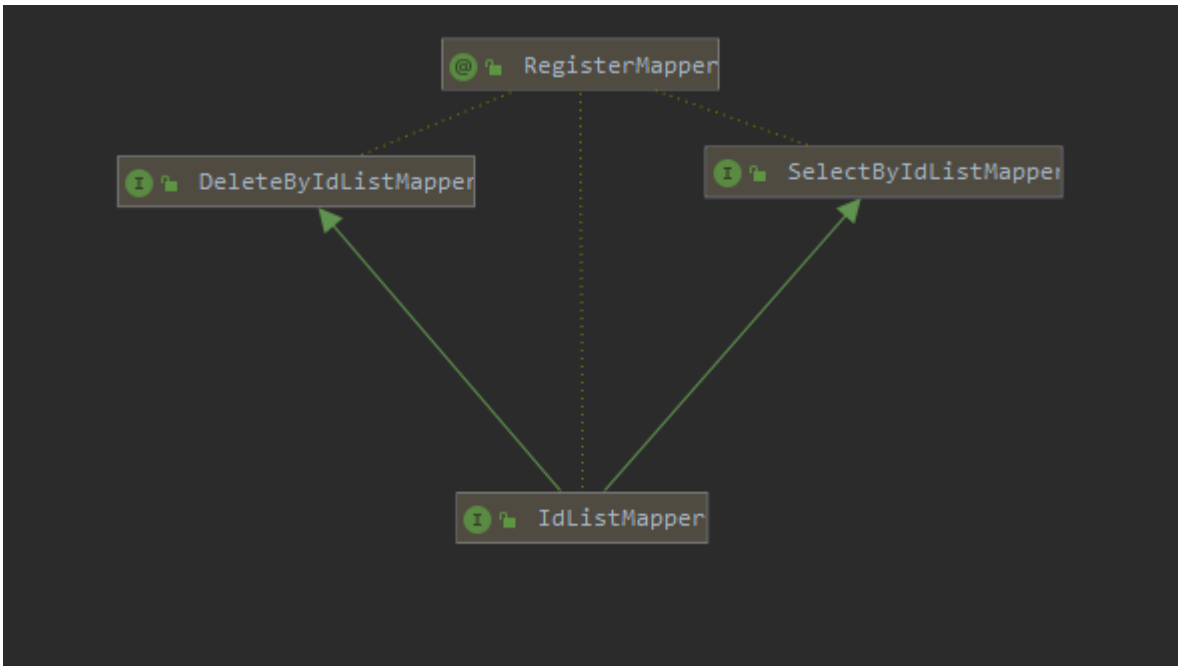
```
insert into table_name ("id", "name") values(1, "zhangsan"), (2, "lisi"), (3, "wangwu");
```

在**tk.mybatis.mapper.common.special**包下，还有一个此方法，也是批量插入，但是此方法存在主键 策 略，但是，集合的批量插入主键是不回显的。

当一个接口继承了其它通用mapper接口时，需要在此接口上`@tk.mybatis.mapper.annotation.RegisterMapper`注解。例如：

```
@RegisterMapper
public interface ITotalMapper<T, K> extends BaseMapper<T>, ExampleMapper<T>,
InsertListMapper<T>, IdListMapper<T, K> {
}
```

批量查询数据



接口：`SelectByIdListMapper<T, PK>`

方法：`List<T> selectByIdList(@Param("idList") List<PK> idList)`

说明：根据集合查询记录，集合里存放的是主键

批量删除数据

接口：`DeleteByIdListMapper<T, PK>`

方法：`int deleteByIdList(@Param("idList") List<PK> idList)`

说明：根据主键的集合删除记录，成功返回 1，失败返回 0

Example方法

接口：`SelectByExampleMapper<T>` 方法：`List<T> selectByExample(Object example);` 说明：根据Example条件进行查询 **重点：**这个查询支持通过`Example`类指定查询列，通过`selectProperties`方法指定查询列

接口：`SelectCountByExampleMapper<T>` 方法：`int selectCountByExample(Object example);` 说明：根据Example条件进行查询总数

接口：SelectOneByExampleMapper

方法：`T selectOneByExample(Object example)`

说明：根据Example条件查询满足条件的一条记录

接口：`UpdateByExampleMapper<T>` 方法：`int updateByExample(@Param("record") T record, @Param("example") Object example);` 说明：根据Example条件更新实体`record`包含的全部属性，null值会被更新

接口：`UpdateByExampleSelectiveMapper<T>` 方法：`int updateByExampleSelective(@Param("record") T record, @Param("example") Object example);` 说明：根据Example条件更新实体`record`包含的不是null的属性值

接口：`DeleteByExampleMapper<T>` 方法：`int deleteByExample(Object example);` 说明：根据Example条件删除数据

Example组合接口

接口：`ExampleMapper<T>` 方法：包含上面Example中的5个方法

主键策略

MYSQL使用主键自增

```
@Id
@KeySql(useGeneratedKeys = true)
private Long id;

或

@Id
@GeneratedValue(generator = "JDBC")
private Long id;
```

oracle使用序列自增

```
@Id
@KeySql(sql = "select SEQ_ID.nextval from dual", order = ORDER.BEFORE)   //begore就是在插入前回显
private Integer id;

或

@Id
@GeneratedValue(
  strategy = GenerationType.IDENTITY,
  generator = "select SEQ_ID.nextval from dual")
private Integer id;
```

不推荐第二种方式，使用`@GeneratedValue`时也要配置一个 **ORDER** 全局参数