SelectBaseMapper与UpdateBaseMapper的变化。

@RegisterMapper

public interface SelectBaseMapper<T> {

@SelectProvider(type=MySelectProvider.class,method = "dynamicSQL")

List<T> selectPage(BaseExample baseExample);

}

这里SelectBaseMapper的变化最主要的就是能直接返回对象。

@RegisterMapper

public interface UpdateBaseMapper {

@UpdateProvider(type=MyUpdateProvider.class,method="dynamicSQL")

public int updateObject( UpdateBaseExample baseExample);

}

以上两个类中我们用到了3个注解，分别是@RegisterMapper，@SelectProvider，@UpdateProvider。

@RegisterMapper：

    这个注解由tk.mybatis:mapper提供，主要作用是将此Mapper接口注册到通用Mapper中。使用这个注解时有一个要求，就是不能被mybatis扫描Mapper接口时扫描到，而我们作为一个单独的项目正好符合这一点。

@SelectProvider

    这个注解由mybatis为我们提供，主要作用是表明被注解的方法的内容提供者。而method的意思是执行提供者时的方法。这里我们的方法为dynamicSQL。这个方法是MapperTemplate为我们提供的一个方法。该方法仅仅用来初始化ProviderSqlSource(而ProviderSqlSource的作用是执行@SelectProvider注解中type对应的类型中的被该注解修饰的方法。拿上面的SelectBaseMapper来举例，就是执行MySelectProvider类中的selectPage方法这里只要求方法名相同即可)。

@UpdateProvider

    这个注解由mybatis为我们提供，主要作用同@SelectProvider。

四、XML如何消失？

   这里既是本章的重点了。其实XML并不是消失，只是不从xml文件中读取，转而变成动态生成了。这里查找和更新的原理相同，我主要拿查找来举例。动态生成的地方就是我们注解的@SelectProvider中type的类型中对应的被注解方法。即MySelectProvider中的selectPage方法。

public class MySelectProvider extends MapperTemplate{

//此构造方法是父类MapperTemplate要求必须填写的

public MySelectProvider(Class<?> mapperClass, MapperHelper mapperHelper) {

super(mapperClass, mapperHelper);

}

    //传入MappedStatement，MappedStatement就相当于是我们XML中整个<select></select>语句的封装，包含了传入类型和返回类型等信息

 public String selectPage(MappedStatement ms)

{

Class<?> entityClass = getEntityClass(ms);

//修改返回值类型为实体类型

setResultType(ms, entityClass);

StringBuilder sql = new StringBuilder("SELECT ");

sql.append(MyExampleSqlHelp.getColumn());// 在sql中加上查找的字段

sql.append(MyExampleSqlHelp.isUseAlias()); //是否使用别名，，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.isUseLeftJoin()); //是否使用了左连接查询，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.useWhereAndEqualsWhere()); //是否使用了大于，小于，不等于，等于查询，并且判断有没有使用等于查询，如果有则在sql中加上

sql.append(MyExampleSqlHelp.useGreaterThan());//是否使用了大于查询，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.useLessThan());//是否使用了小于查询，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.useNotEquals());//是否使用了不等于查询，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.notUseWhere());//再次判断是否使用了查询

sql.append(MyExampleSqlHelp.useIn());//是否使用了in查询，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.useLike());//是否使用了like查询，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.useGroup());//是否使用了分组功能，如果使用了则在sql中加上

sql.append(MyExampleSqlHelp.useOrder());//是否使用了排序功能，如果使用了则在sql中加上

return sql.toString();

}

}

这里我们自定义了一个MyExampleSqlHelp类，是为了分段管理sql，防止混乱。

/\*\*

\* 是否使用了不等于查询

\* @return

\*/

public static String useNotEquals() {

return "<if test =\"@com.example.test.util.ExampleOGNL@useNotEqualsWhere(\_parameter)\">\r\n" +

" <foreach collection=\"notEqualsWhereKey\" item=\"key\" index=\"index\" separator=\"AND\"> " +

" ${key} != #{notEqualsWhereValue[${index}]} " +

" </foreach>\r\n" +

" </if>\r\n" +

" </where>\r\n" +

" </if>";

}

我们查看MyExampleSqlHelp类的useNotEquals()方法可以发现，其实这里的语句与xml中极为类似，只是我们在xml中的直接判断我们改为了@com.example.test.util.ExampleOGNL@useNotEqualsWhere(\_parameter)这种形式，这是MyBatis中OGNL表达式的形式，意思是当执行test中的判断条件时转到com.example.test.util.ExampleOGNL类的useNotEqualsWhere(\_parameter)方法中执行。

/\*\*

\* 是否使用了不等于查询

\* @param parameter 查询参数

\*/

public static boolean useNotEqualsWhere(Object parameter) {

if(parameter != null) {

BaseExample sbe = (BaseExample)parameter;

List<String> notEqualsWhere = sbe.getNotEqualsWhereKey();

if(notEqualsWhere != null && notEqualsWhere.size() >0) {

return true;

}

}

return false;

}

这里非常重要的一点就是该方法的返回值必须是其调用方需要的类型。比如我们在上面执行<if>标签的test，那么我们的返回值就必须是boolean或者Boolean类型。

这段方法的逻辑也很简单，就是判断传入参数中是否存在notEqualsWhere属性，并且有没有值。

经过上述的讲解，我们基本完成了对xml消失这一条件的实现。同时，现在我们的整个项目更加简洁，并能打成jar包直接使用。

打成jar包之后再好不过的就是能直接从maven中心仓库里下载啦。那么，在下一篇文章中我将讲解如何上传jar包到中心仓库中。希望各位看官老爷们能喜欢。

最后的最后，附上项目github地址:https://github.com/satant/pj.mybatis

maven下载依赖，如果要使用，version务必2.2.0以上，且只需继承BaseExampleMapper接口即可。

<!-- https://mvnrepository.com/artifact/com.github.satant/pj.mybatis -->

        <dependency>

            <groupId>com.github.satant</groupId>

            <artifactId>pj.mybatis</artifactId>

            <version>2.2.0</version>

        </dependency>

---------------------

作者：亲爱的撒旦

来源：CSDN

原文：https://blog.csdn.net/qq\_30770095/article/details/80249406

版权声明：本文为博主原创文章，转载请附上博文链接！

h\_collect join h\_doctor

on h\_collect.coll\_new\_doc\_id = doctor.id