链滴

Mybatis 封装通用的增删改查

作者：shengkai
原文链接：https://ld246.com/article/1466502291090
来源网站：链滴

曾经发过一篇文章，大概写的就是阿海多么多么厉害，见到某位同事在Hibernate的基础上封装了下，就以一己之力开发什么什么框架，最后写了个超大的一坨的事。
那么，后续篇来了。阿海不是自负之人，当之前的CRUD框架并没有达到理想的结果时，阿海转向了Mbatis封装。别问我为什么不是Hibernate。我本来就不喜欢Hibernate，即使在之前的一家公司一直强制性的约束使用Hibernate时，也没有对Hibernate产生什么真正的好感，反而屡次发现了Hibernat的一些问题。

或许是因为版本太老了得缘故吧（都是2016年的新世纪了，还在用Hibernate2）总之，bug层出穷，极不稳定，而且还因为版本过老的远古，一些依赖或者受影响的包也必须使用老版本，哎不管怎说，在那段逼逼的时光里敲着逼逼的代码，做着逼逼的事情。重复着逼逼的生活。所以，说到封装，第一时间想到了mybatis。版本嘛，不高不低，就用我最熟练的3.2版本。至于我为什么不用更高的版，主要是我第一次做系统的封装，先用我最拿手的试试，尽管说是最拿手的，但也好长时间没有玩儿了，苦逼的玩儿了一年的hibernate2对于阿海的性格来说，简直就是煎熬。

扯了那么多没用的，接下来说说我这个Hibernate呃，打错了，是我这个MyBatis封装和之前的那Hibernate封装有什么区别和优势。首先，最大的区别就是，Mybatis少不了xml文件。因为这是Mybais的本质，也是他的精华所在。另外呢，就是没有防注入功能。占位付功能，在我封装后的Mybaits，彻底废掉了。也不能说是彻底废掉，只是那些公用方法中的占位付给废掉了。当然，解决SQL注入不一定非得让框架去支持，自己也可以写，随便写个参数拦截器就好了，当然也必须要写，不仅仅是了解决SQL注入，比如html标记转义、敏感字符过滤直来的都是要有的吧？所以说，废掉通用方法中占位付功能并不是一个不明智的选择。

接下来就是与Hibernate之间的缓存效率对比，准确的说是与那个封装之后的Hibernate的缓存对。Hibernate的（缓存+事物）问题相信一些精通的朋友应该知道，当然，即使有一些知道的我还是简单的说一下，毕竟一些朋友还是不知道的。。

在这段文字中，我觉得我应该严肃一下，不再逗逼。好了，且听我说：所谓缓存机制，Hibernate的二级缓存利用了一个叫做“快照”的逗逼机制，本体来说的确挺厉害，然而碰到事物，就...就....这尴尬了-_-!。为什么这么说呢？我先来介绍一下Hibernate的快照原理。当hibernate查询时，会给这语句的结果生成一个缓存和快照，当再次查询时，他会从缓存中找到要查询的对象，然后与快照中的比，当缓存和快照中的id（Hibernate给他定义的对象ID，叫OID，对象更新后OID会变）一致时，不会从数据库中再次查询。而是直接使用缓存中的对象。这样，减少了sql语句的查询。对效率有一的优化。那么问题来了，当开启事务时，每执行一次操作，缓存都会更新一次，而快照则是在提交之才会更新。那么问题来了，当执行完一些sql时，缓存更新，这时候如果遭到回滚的话，快照则不会新。那么这个更新后的缓存就是一个废弃的缓存，就是一个脏数据。如果正好用户没有配置快照的话再次查询出来的数据则不会和数据库中的数据同步。所以就产生了错误数据。

去其他的就不用说了吧？这套基于Mybatis的封装保留了Mybatis的精髓，你依然可以配置mappe，编写xml，使用动态SQL，充分的发挥软编码的特性。
是上面吹了这么多，接下来应该说一下这个封装的结构了。其实也没有什么结构可言，因为之前写过个不算完美的CRUD框架，所以很多东西都可以拿来服用，至少SQL生成这一块和注解这一块基本上以复制粘贴。
通用XML：

<pre class="prettyprint lang-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.aiyi.core.dao.Dao">
<!-- 局部新增 -->
<insert id="addLocal">
${value}
</insert>
<pre><code class="highlight-chroma"><span class="highlight
cl"><!-- 整体新增 -->
<insert id=
add">
    ${value}
</insert&
mp;gt;

<!-- 根据I
查询单个 -->
<select id=
getById" resultType="hashmap">
    ${value}
</select&
mp;gt;

<!-- 根据I
查询单个字段 -->
<select id=
getFieldById" resultType="hashmap">
    ${value}
</select&
mp;gt;

<!-- 根据
件查询单个 -->
<select id=
getByParm" resultType="hashmap">
    ${value}
</select&
mp;gt;

<!-- 根据
件查询单个字段 -->
<select id=
getFieldByParm" resultType="hashmap">
    ${value}
</select&
mp;gt;

<!-- 条件
询列表 -->
<select id=
selectList" resultType="hashmap">
    ${value}
</select&
mp;gt;

```
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 条件
询字段列表 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;select id=
selectListField" resultType="hashmap"&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    ${value}
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;/select&
mp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 更新
部 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;update i
="updateLocal"&amp;gt;${value}&amp;lt;/update&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 更新
部 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;update i
="update"&amp;gt;${value}&amp;lt;/update&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 条件
新局部 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;update i
="updateLocalByPram"&amp;gt;${value}&amp;lt;/update&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 条件
新全部 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;update i
="updateByPram"&amp;gt;${value}&amp;lt;/update&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 根据I
删除 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;delete id
"deleteById"&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    ${value}
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;/delete&
mp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 批量
除 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;delete id
"deleteByparm"&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    ${value}
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;/delete&
mp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 自定
语句查询 --&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;select id=
selectBySql" resultType="hashmap"&amp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">    ${value}
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;/select&
mp;gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl">&amp;lt;!-- 自定
```

语句执行 -->
<update i
="excuse">${value}</update>

<!-- 查询
合条件的数量 -->
<select id=
selectCountByParm" resultType="Long">
    ${value}
</select&
mp;gt;

<!-- 查询
数量 -->
<select id=
selectCount" resultType="Long">
    ${value}
</select&
mp;gt;

<!-- 内查询
-->
<select id=
selectIn" resultType="hashmap">
    ${value}
</select&
mp;gt;

<!-- 获取
一个序列的值 -->
<select id=
fetchSeqNextval" resultType="Long" flushCache="true">
    ${value}
</select&
mp;gt;
</code></pre>
</mapper> </pre>

从这个 mapper 文件中可以看出为什么没有占位符了吧？

接下来就是通用 dao

<pre class="prettyprint lang-java">package com.aiyi.core.dao;

import java.io.Serializable;

import java.util.List;

import java.util.Map;

```
import com.aiyi.core.beans.Po;
import com.aiyi.core.beans.WherePrams;

/**
 * 公共数据库操作层
 * @author 郭胜凯
 * @time 2016年5月3日下午2:55:13
 * @email 719348277@qq.com
 * @param &lt;T&gt; 实体PO类型
 * @param &lt;PK&gt; PO主键类型
 */
public interface Dao&lt;T extends Po, PK extends Serializable
gt; {
    /**
     * 添加不为空的记录（只将不为空字段入库，效率高）
     * @param po
     * @return 受改变的记录数
     */
    public int addLocal(T po);

    /**
     * 记录添加（所有字段入库，效率中）
     * @param po
     * @return
     */
    public int add(T po);

    /**
     * 通过主键获取某个记录
     * @param id 主键
     * @return PO
     */
    public T get(PK id);

    /**
     * 通过主键获取某个字段的值
     * @param id
     * @param fileName
     * @return
```

```
*/
public Serializable getField(PK id, String fileName);
/**
    ● 条件获取一条记录
    ● @param t
    ● @param 条件表达式
    ● @return PO
*/
public T get(WherePrams where);
/**
    ● 条件获取某个记录字段
    ● @param where
    ● @param fileName
    ● @return
*/
public Serializable getFile(WherePrams where, String fileName);
/**
    ● 条件查询列表
    ● @param where 条件表达式
    ● @return PO列表
*/
public List&lt;T&gt; list(WherePrams where);
/**
    ● 查询某个字段列表
    ● @param where 条件表达式
    ● @param fileName 要查询的字段
    ● @return
*/
public Serializable[] listFile(WherePrams where, String fileName);
/**
    ● 查询某些字段
    ● @param where 条件表达式
    ● @param files 要查询的字段集
    ● @return 查询的PO字段列表
*/
```

```
public List&lt;Map&lt;String, Serializable
gt;&gt; listFiles(WherePrams where, String[] files);

/**
```

- 更新不为null的PO字段
- @param po
- @return 受影响的行数

```
*/
public int updateLocal(T po);

/**
```

- 更新PO的所有字段
- @param po
- @return 受影响的行数

```
*/
public int update(T po);

/**
```

- 条件更新不为null的字段
- @param po
- @param 条件表达式
- @return 受影响的行数

```
*/
public int updateLocal(T po, WherePrams where);

/**
```

- 条件更新所有字段
- @param po
- @param 条件表达式
- @return 受影响的行数

```
*/
public int update(T po, WherePrams where);

/**
```

- 删除某个记录
- @param id 主键
- @return 受影响的行数

```
*/
public int del(PK id);

/**
```

- 条件删除某个记录
  - @param where 条件表达式
  - @return 受影响的行数

*/

public int del(WherePrams where);

/**

  - 自定义sql查询
  - @param po 用于封装返回结果的Bean
  - @param sql 用于执行查询的Sql
  - @param args Sql占位付对应的参数
  - @return 结果集合

*/

public List<Map<String, Object
gt;> listBySql(String sql);

/**

  - 执行自定义sql
  - @param sql 用于执行的Sql
  - @param args Sql占位付对应的参数
  - @return 受影响的行数

*/

public int excuse(String sql);

/**

  - 获取指定条件的记录数
  - @param where 条件表达式
  - @return 查询到的记录数

*/

public long count(WherePrams where);

/**

  - 获取对应表中的记录数
  - @return 表中的条数

*/

public long size();

/**

  - 是否存在字段相同的记录（ID以及不为空的字段除外）
  - @param po 参照实体

- @return

*/

public boolean isExist(T po);

/**

- 是否存在指定条件的记录
- @param where 条件表达式
- @return

*/

public boolean isExist(WherePrams where);

/**

- 内查询
- @param fileName 用于内查询的字段
- @param values 字段的值
- @return 查询到的结果集

*/

public List<T> in(String fileName, Serializable[] va
ues);

/**

- 获得下一个序列的值
- @return

*/

public long nextId();

}</pre>

<pre><code class="highlight-chroma"><span class="highlight
cl">   <p>

    </p>
    <p style="tex
-indent:21.25pt;">
        以及DAO的
现类
    </p>
    <p style="tex
-indent:21.25pt;">
</code></pre>

<pre class="prettyprint lang-java">package com.aiyi.core.dao.impl;

import java.io.Serializable;
import java.lang.reflect.Field;

```
import java.net.ConnectException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.LinkedHashMap;
import java.util.List;
import java.util.Map;

import javax.annotation.Resource;

import org.mybatis.spring.SqlSessionTemplate;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Repository;

import com.aiyi.core.annotation.po.FieldName;
import com.aiyi.core.beans.Method;
import com.aiyi.core.beans.Po;
import com.aiyi.core.beans.Pram;
import com.aiyi.core.beans.WherePrams;
import com.aiyi.core.dao.Dao;
import com.aiyi.core.sql.where.C;
import com.aiyi.core.sql.where.SqlUtil;
import com.aiyi.core.util.GenericsUtils;

@SuppressWarnings("unused")
@Repository
public class DaoImpl&lt;T extends Po, PK extends Serializable&gt; implements Dao&lt;T, PK&
t; {

    protected Logger logger = LoggerFactory.getLogger(this.getClass());

    @Resource(name = "sqlSessionTemplateASS")
    private SqlSessionTemplate sqlSessionTemplateASS;

    private Class&lt;T&gt; entityClass;

    private String pkName;                  //主键字段

    private String idName;                  //对应id名称

    private String seq;                     //当前主键

    private String tableName;

    private List&lt;Pram&gt; sqlParms;


    private List&lt;Pram&gt; selectSqlParms;

    private SqlUtil&lt;T&gt; sqlUtil;
```

```
@SuppressWarnings("unchecked")
public DaoImpl(){
    super();

    this.sqlUtil = new SqlUtil&lt;T&gt;();

    this.entityClass = (Class&lt;T&gt;) GenericsUtils.getSuperClassGenricType(this.getClass());

    this.sqlParms = this.sqlUtil.getPramList(this.entityClass);

    this.selectSqlParms = this.sqlUtil.getPramListOfSelect(this.entityClass);

    this.tableName = this.sqlUtil.getTableName(this.entityClass);

    //习惯统一用‘id’做约束了，所以这里我给固定死了，不想固定的话可以修改这里
    this.pkName = "id";

    this.idName = "id";

    this.seq = "id";

}


@Override
public int addLocal(T po) {
    // TODO Auto-generated method stub

    String sql = "insert into " + tableName + " (";
    String prams = "";
    String values = "";

    List&lt;Pram&gt; pramList = SqlUtil.getPramListofStatic(po);

    int index = 0;
    for (int i = 0; i &lt; pramList.size(); i++) {
        if (pramList.get(i).getValue() == null || (pramList.get(i).getValue() + "") .equals("0")) {
            continue;
        }else{
            if(index &gt; 0){
                prams += ",";
                values += ",";
            }
            prams += pramList.get(i).getFile();
            Object value = pramList.get(i).getValue();
            if (value instanceof byte[] ) {
                values += "'" + new String((byte[]) value) + "'";
            }else if(value instanceof String){
                values += "'" + value + "'";
            }else{
                values += value ;
            }
```

```
            index ++;
        }
    }
    sql += prams + ") value (" + values +");";

    SqlUtil.setFileValue(po, "id", nextId());

    logger.debug(sql);
    return sqlSessionTemplateASS.insert("addLocal", sql);

}

@Override
public int add(T po) {
    // TODO Auto-generated method stub
    String sql = "insert into " + tableName + " (";
    String prams = "";
    String values = "";

    List&lt;Pram&gt; pramList = SqlUtil.getPramListofStatic(po);

    for (int i = 0; i &lt; pramList.size(); i++) {
        prams += pramList.get(i).getFile();
        if (pramList.get(i).getValue() == null) {
            values += "null";
        }else{
            values += "'" + pramList.get(i).getValue() + "'";
        }

        if(i &lt; pramList.size() -1){
            prams += ",";
            values += ",";
        }
    }
    sql += prams + ") value (" + values +");";

    SqlUtil.setFileValue(po, "id", nextId());

    return sqlSessionTemplateASS.insert("add", sql);
}

@Override
public T get(PK id) {
    // TODO Auto-generated method stub
    String sql = "select ";
    for (int i = 0; i &lt; selectSqlParms.size(); i++) {
        sql += selectSqlParms.get(i).getFile();
        if(i &lt; selectSqlParms.size() -1){
            sql += ",";
        }else{
            sql += " ";
        }
    }
    sql += " from " + tableName + " where id='" + id + "';";
```

```
    Map&lt;String, Object&gt; resultMap = sqlSessionTemplateASS.selectOne(
                "getById", sql);

    return handleResult(resultMap, this.entityClass);
}

@Override
public Serializable getField(PK id, String fileName) {
    // TODO Auto-generated method stub
    String field = fileName;
    String tabField = "";
    Field f = sqlUtil.getField(this.entityClass, fileName);
    if (null == f) {
        logger.error("查询字段失败(无法找到" + this.entityClass + "中的" + fileName + "字段)");
    }
    FieldName annotation = f.getAnnotation(FieldName.class);
    if (null == annotation) {
        tabField = sqlUtil.toTableString(fileName) + " as " + fileName;
    }else{
        tabField = annotation.name() + " as " + fileName;
    }

    String sql = "select ";
    sql += tabField + " from " + tableName + " where id='" + id + "';";
    Map&lt;String, Object&gt; resultMap = sqlSessionTemplateASS.selectOne(
            "getFieldById", sql);
    return (Serializable) resultMap.get(fileName);
}

@Override
public T get(WherePrams where) {
    // TODO Auto-generated method stub
    String sql = "select ";
    for (int i = 0; i &lt; selectSqlParms.size(); i++) {
        sql += selectSqlParms.get(i).getFile();
        if(i &lt; selectSqlParms.size() -1){
            sql += ",";
        }else{
            sql += " ";
        }
    }
    sql += "from " + tableName + where.getWherePrams() + ";";

    Map&lt;String, Object&gt; resultMap = sqlSessionTemplateASS.selectOne(
            "getByParm", sql);

    return handleResult(resultMap, this.entityClass);
}

@Override
public Serializable getFile(WherePrams where, String fileName) {
    // TODO Auto-generated method stub
    String field = fileName;
    String tabField = "";
```

```
        Field f = sqlUtil.getField(this.entityClass, fileName);
        if (null == f) {
            logger.error("查询字段失败(无法找到" + this.entityClass + "中的" + fileName + "字段)");
        }
        FieldName annotation = f.getAnnotation(FieldName.class);
        if (null == annotation) {
            tabField = sqlUtil.toTableString(fileName) + " as " + fileName;
        }else{
            tabField = annotation.name() + " as " + fileName;
        }

        String sql = "select ";
        sql += tabField + " from " + tableName + where.getWherePrams() + ";";
        Map&lt;String, Object&gt; resultMap = sqlSessionTemplateASS.selectOne(
                "getFieldByParm", sql);
        return (Serializable) resultMap.get(fileName);
    }

    @Override
    public List&lt;T&gt; list(WherePrams where) {
        // TODO Auto-generated method stub

        String sql = "select ";
        for (int i = 0; i &lt; selectSqlParms.size(); i++) {
            sql += selectSqlParms.get(i).getFile();
            if(i &lt; selectSqlParms.size() -1){
                sql += ",";
            }else{
                sql += " ";
            }
        }
        sql += "from " + tableName + where.getWherePrams() + ";";

        List&lt;Map&lt;String, Object&gt;&gt; selectList = sqlSessionTemplateASS.selectList("sel
ctList", sql);

        List&lt;T&gt; list = new ArrayList&lt;&gt;();
        for (Map&lt;String, Object&gt; map : selectList) {
            T t = handleResult(map, this.entityClass);
            list.add(t);
        }

        return list;

    }

    @Override
    public Serializable[] listFile(WherePrams where, String fileName) {
        // TODO Auto-generated method stub

        String field = fileName;
        String tabField = "";
        Field f = sqlUtil.getField(this.entityClass, fileName);
        if (null == f) {
```

```
            logger.error("查询指定字段集失败(无法序列化" + this.entityClass + "中的" + fileName +
字段)");
        }
        FieldName annotation = f.getAnnotation(FieldName.class);
        if (null == annotation) {
            tabField = sqlUtil.toTableString(fileName) + " as " + fileName;
        }else{
            tabField = annotation.name() + " as " + fileName;
        }

        String sql = "select ";
        sql += tabField + " from " + tableName + where.getWherePrams() + ";";
        List&lt;Map&lt;String, Object&gt;&gt; resultMap = sqlSessionTemplateASS.selectList("se
ectListField", sql);

        Serializable[] fields = new Serializable[resultMap.size()];

        for (int i = 0; i &lt; resultMap.size(); i++) {
            if (null != resultMap.get(i)) {
                fields[i] =(Serializable) resultMap.get(i).get(fileName);
            }
        }

        return fields;
    }

    @Override
    public List&lt;Map&lt;String, Serializable&gt;&gt; listFiles(WherePrams where, String[] files)

        // TODO Auto-generated method stub
        String tabField = "";
        int index = 1;
        for (String field : files) {
            Field f = sqlUtil.getField(this.entityClass, field);
            if (null == f) {
                logger.error("查询指定字段集失败(无法序列化" + this.entityClass + "中的" + field + "
段)");
            }
            FieldName annotation = f.getAnnotation(FieldName.class);
            if (null == annotation) {
                tabField += sqlUtil.toTableString(field) + " as " + field;
            }else{
                tabField += annotation.name() + " as " + field;
            }
            if (index &lt; files.length) {
                tabField += ",";
            }

            index ++;
        }

        String sql = "select ";
        sql += tabField + " from " + tableName + where.getWherePrams() + ";";
        List&lt;Map&lt;String, Serializable&gt;&gt; resultMap = sqlSessionTemplateASS.selectList
```

```
"selectListField", sql);

        return resultMap;
    }

    @Override
    public int updateLocal(T po) {
        // TODO Auto-generated method stub

        Serializable id = sqlUtil.getFileValue(po, "id");

        if(null == id){
            return 0;
        }
        String sql = "update " + tableName + " set ";
        List&lt;Pram&gt; prams = sqlUtil.getPramList(po);
        for (int i = 0; i &lt; prams.size(); i++) {
            if(null != prams.get(i).getValue()){
                sql += prams.get(i).getFile() + "=";
                Object value = prams.get(i).getValue();
                if (value instanceof byte[] ) {
                    sql += "'" + new String((byte[]) value) + "'";
                }else if(value instanceof String){
                    sql += "'" + value + "'";
                }else{
                    sql += value ;
                }

//              sql += prams.get(i).getFile() + "='" + prams.get(i).getValue() + "'";
                if (i &lt; prams.size() -1) {
                    sql += ",";
                }
            }
        }
        sql += " where id='" + id +"';";

        return sqlSessionTemplateASS.update("updateLocal", sql);
    }

    @Override
    public int update(T po) {
        // TODO Auto-generated method stub

        Serializable id = sqlUtil.getFileValue(po, "id");

        if(null == id){
            return 0;
        }
        String sql = "update " + tableName + " set ";

        List&lt;Pram&gt; prams = sqlUtil.getPramList(po);

        for (int i = 0; i &lt; prams.size(); i++) {
            if(null != prams.get(i).getValue()){
```

```
                sql += prams.get(i).getFile() + "=";
                Object value = prams.get(i).getValue();
                if (value instanceof byte[] ) {
                    sql += "'" + new String((byte[]) value) + "'";
                }else if(value instanceof String){
                    sql += "'" + value + "'";
                }else{
                    sql += value ;
                }
//              sql += prams.get(i).getFile() + "='" + prams.get(i).getValue() + "'";
                if (i &lt; prams.size() -1) {
                    sql += ",";
                }
            }else{
                sql += prams.get(i).getFile() + "=null";
                if (i &lt; prams.size() -1) {
                    sql += ",";
                }
            }
        }
        sql += " where id='" + id +"';";

        return sqlSessionTemplateASS.update("update", sql);
    }

    @Override
    public int updateLocal(T po, WherePrams where) {
        // TODO Auto-generated method stub

        String sql = "update " + tableName + " set ";
        List&lt;Pram&gt; prams = sqlUtil.getPramList(po);
        for (int i = 0; i &lt; prams.size(); i++) {
            if(null != prams.get(i).getValue()){
                sql += prams.get(i).getFile() + "=";
                Object value = prams.get(i).getValue();
                if (value instanceof byte[] ) {
                    sql += "'" + new String((byte[]) value) + "'";
                }else if(value instanceof String){
                    sql += "'" + value + "'";
                }else{
                    sql += value ;
                }
//              sql += prams.get(i).getFile() + "='" + prams.get(i).getValue() + "'";
                if (i &lt; prams.size() -1) {
                    sql += ",";
                }
            }
        }
        sql += where.getWherePrams() +"';";

        return sqlSessionTemplateASS.update("updateLocalByPram", sql);

    }
```

```
    @Override
    public int update(T po, WherePrams where) {
        // TODO Auto-generated method stub

        String sql = "update " + tableName + " set ";
        Object[] o = new Object[sqlParms.size()];
        for (int i = 0; i &lt; sqlParms.size(); i++) {
            if(null != sqlParms.get(i).getValue()){
                sql += sqlParms.get(i).getFile() + "=" + sqlParms.get(i).getValue();
                if (i &lt; sqlParms.size() -1) {
                    sql += ",";
                }
            }else{
                sql += sqlParms.get(i).getFile() + "=null";
                if (i &lt; sqlParms.size() -1) {
                    sql += ",";
                }
            }
        }
        sql += where.getWherePrams() + "';";

        return sqlSessionTemplateASS.update("updateByPram", sql);

    }

    @Override
    public int del(PK id) {
        // TODO Auto-generated method stub
        String sql = "delete from " + tableName + " where id=" + id;

        return sqlSessionTemplateASS.delete("deleteById", sql);
    }

    @Override
    public int del(WherePrams where) {
        // TODO Auto-generated method stub

        String sql = "delete from " + tableName + where.getWherePrams();

        return sqlSessionTemplateASS.delete("deleteByparm", sql);
    }

    @Override
    public List&lt;Map&lt;String, Object&gt;&gt; listBySql(String sql) {
        // TODO Auto-generated method stub

        List&lt;Map&lt;String, Object&gt;&gt; selectList = sqlSessionTemplateASS.selectList("sel
ctBySql", sql);
        return selectList;
    }

    @Override
    public int excuse(String sql) {
        // TODO Auto-generated method stub
```

```
        return sqlSessionTemplateASS.update("excuse", sql);
    }

    @Override
    public long count(WherePrams where) {
        // TODO Auto-generated method stub

        String sql = "select count(*) from ";

        sql += tableName + where.getWherePrams();

        long count = sqlSessionTemplateASS.selectOne("selectCountByParm", sql);

        return count;
    }

    @Override
    public long size() {
        // TODO Auto-generated method stub
        String sql = "select count(*) from " + tableName;
        long count = sqlSessionTemplateASS.selectOne("selectCount", sql);
        return count;
    }

    @Override
    public boolean isExist(T po) {
        // TODO Auto-generated method stub
        WherePrams wherePrams = Method.createDefault();

        List&lt;Pram&gt; list = SqlUtil.getPramListofStatic(po);

        for (int i = 0; i &lt; list.size(); i++) {
            if (i == 0) {
                wherePrams = Method.where(list.get(i).getFile(), C.EQ, (Serializable)list.get(i).getVal
e());
            }else{
                wherePrams.and(list.get(i).getFile(), C.EQ, (Serializable)list.get(i).getValue());
            }
        }


        return count(wherePrams) &gt; 0;
    }

    @Override
    public boolean isExist(WherePrams where) {
        // TODO Auto-generated method stub
        return count(where) &gt; 0;
    }

    @Override
    public List&lt;T&gt; in(String fileName, Serializable[] values) {
        // TODO Auto-generated method stub
```

```
        String sql = "select ";
        for (int i = 0; i &lt; sqlParms.size(); i++) {
            sql += selectSqlParms.get(i).getFile();
            if(i &lt; selectSqlParms.size() -1){
                sql += ",";
            }else{
                sql += " ";
            }
        }
        sql += "from " + tableName + " where " + fileName + " in ";
        String value = "(";
        for(int i = 0; i &lt; values.length; i++){
            if(i &lt; values.length -1){
                value += values[i] + ",";
            }else{
                value += values[i] + ");";
            }
        }
        sql += value;

        List&lt;Map&lt;String, Object&gt;&gt; selectList = sqlSessionTemplateASS.selectList("sel
ctIn", sql);

        List&lt;T&gt; list = new ArrayList&lt;&gt;();
        for (Map&lt;String, Object&gt; map : selectList) {
            T t = handleResult(map, this.entityClass);
            list.add(t);
        }

        return list;
    }

    private T handleResult(Map&lt;String, Object&gt; resultMap, Class&lt;T&gt; tClazz) {
        if (null == resultMap) {
            return null;
        }
        T t = null;
        try {
            t = tClazz.newInstance();
        } catch (InstantiationException e) {
            logger.error("/**********************************");
            logger.error("实例化Bean失败(" + this.entityClass + ")!"
                    + e.getMessage());
            logger.error("/**********************************");
        } catch (IllegalAccessException e) {
            logger.error("/**********************************");
            logger.error("实例化Bean失败(" + this.entityClass + ")!"
                    + e.getMessage());
            logger.error("/**********************************");
        }
        for (Map.Entry&lt;String, Object&gt; entry : resultMap.entrySet()) {
            String key = entry.getKey();
            Serializable val = (Serializable) entry.getValue();
            try {
```

```
                SqlUtil.setFileValue(t, key, val);
            } catch (Exception e) {
                // TODO: handle exception
                logger.error("/*********************************");
                logger.error("/实例化Bean失败(" + this.entityClass + ")不能赋值到字段(" + key + "):"
                        + e.getMessage());
                logger.error("/*********************************");
            }
        }
        return t;
    }

    /**
     * 获取某表的下一个Id
     */
    public long nextId(){
        String sql = "SELECT auto_increment FROM information_schema.`TABLES` WHERE TABLE
NAME='" + tableName + "'";
        Long idVal = sqlSessionTemplateASS.selectOne("fetchSeqNextval", sql);
        if (null == idVal) {
            logger.error("/*********************************");
            logger.error("/获取id失败，" + tableName + "表异常。请检查是否存在或者配为自增主键")

            logger.error("/*********************************");
            return 0;
        }
        return idVal;

    }

}</pre>

<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">        &lt;p&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">            &lt;br /&gt

</span></span><span class="highlight-line"><span class="highlight-cl">        &lt;/p&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">        &lt;p style="
ext-indent:21.25pt;"&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">            大概就是
个样子，注解我是直接拷贝的之前的CRUD框架，同样，为了效率和兼容MyBaits，这里我直接在类
用了泛型，而不是在方法上用泛型。之前在方法上用了泛型机制，这样的优点就是，只需要实例化一
类就好了。大概调用方法是：
</span></span><span class="highlight-line"><span class="highlight-cl">        &lt;/p&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">        &lt;p style="
ext-indent:21.25pt;"&gt;
</span></span></code></pre>

<pre class="prettyprint lang-java">//假设获取id为1的用户
UserPo user = get(UserPo.class，"id");</pre>

<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">            &lt;p&gt;
```

<br /
gt;
</p>
<p style
"text-indent:21.25pt;">
然而在
率上来说，则是大大的不好。因为每次方法调用时，都要将所传入的class进行解析，效率上大打折扣
在这里，我直接在类上加入了泛型，当一个Dao类初始化时，会自动对泛型字段进行解析，仅解析一
而已，之后每次使用这个Dao类的方法都是用解析好的资源。而不是每次调用方法就要反射一次。再
合Spring让其自动实例化，这样堪称完美。
</p>
<p style
"text-indent:21.25pt;">
那么，
使用过程中，则有了些许的改变，和Mybatis一样，需要些Dao，但是同样不用写实现类，准确的说
是直接集成已实现的Dao类。
</code></pre>

加入我要写一个 UserDao，那么，我不需要写接口和 mapper 文件，直接这样搞：

<pre class="prettyprint lang-java">/**
* 一个基于公用Dao的UserDao
*/
@Repository
public class UserDao extends Dao<UserPo, Integer> {

//在这里，这个UserDao本身拥有了Dao的一些常用的增删改查方法。也可以重写或者增加一些特殊
法。
public List<UserPo> listGirl(){

    //获取用户中的小姑娘，瞎写的。。
}

//再比如说我要重写一下list()这个方法
public List<UserPo> list(WhereParms where){

    return super.list(where);
}

}</pre>

<pre><code class="highlight-chroma"><span class="highlight
cl"><p>

</p&
t;
<p sty
e="text-indent:21.25pt;">
仅此
已，比直接用插件生成dao层还要方便一些，至少我感觉是这样。
</code></pre>

日然后，Service 层可以这么玩儿：

<pre class="prettyprint lang-java">@Service
public class UserServiceImpl{

@Resource
private UserDao userDao;

    /*
    *根据姓名查找用户
    */
public List<UserPo> listUser(String name){
    return userDao.list(Methor.where("name", C.Like, name));
}

}</pre>

<pre><code class="highlight-chroma"><span class="highlight
cl">   <p>
                    &lt
br />
                </
>
                <p
tyle="text-indent:21.25pt;">
                    还
一些若干的小功能，比如说自动日志啦，（在Service层的任意方法中，加入@Log注解可以记录这个
法的进栈时间、执行时所接收的参数，执行时所出现的异常，出栈时间等等）
</code></pre>

再比如说自动事务（加入 @Transactional 注解后该方法拥有事务，一般和 @<a href="https://
d246.com/member/Log" aria-name="Log" class="tooltipped__user" target="_blank">Log</a
 一起用）

<pre class="prettyprint lang-java">@Override
    @Transactional
    @Log
    public ResultBean delUser(int uId, int appId, AdminUser user) {
        // TODO Auto-generated method stub
        ResultBean resultBean = new ResultBean();
        resultBean.setResult(false);
        AdminApp app = adminAppDao.get(appId);
        if (null == app) {
            resultBean.setCode(1);
            resultBean.setMessage("App不存在");
            return resultBean;
        }
        if (app.getAdminUserId() != user.getId()) {
            resultBean.setCode(2);
            resultBean.setMessage("无权操作");
            return resultBean;

```
        }


    userFriendDao.del(Method.where("userId", C.EQ, uId));
    userFriendInfoDao.del(Method.where("userId", C.EQ, uId).or("friendId", C.EQ, uId));

    List&lt;UserFriendGroup&gt; list = userFriendGroupDao.list(Method.where("userId", C.EQ, u
d));
    for (UserFriendGroup userFriendGroup : list) {
        userFriendDao.del(Method.where("friendGroupId", C.EQ, userFriendGroup.getId()));
        userFriendGroupDao.del(userFriendGroup.getId());
    }

    appUserDao.del(uId);
    app.setUserNum(app.getUserNum() - 1);
    adminAppDao.update(app);
    resultBean.setCode(0);
    resultBean.setMessage("删除成功");
    resultBean.setResult(true);
    return resultBean;
}</pre>

<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">                    &lt;p&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
lt;br /&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">                    &lt;
p&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">                    &lt
p style="text-indent:21.25pt;"&gt;
</span></span><span class="highlight-line"><span class="highlight-cl">
—本次装逼结束，成功装完1个逼，成功捕获0个异常。成功率100% &lt;br /&gt;
</span></span></code></pre>


<p>目前暂未发现明什么 Bug，阿海也再用这个封装的 mybatis 写一个自己的项目。等过段时间确
框架没有问题后，阿海再把该封装的 Jar 包以及源码发布。<br>
</p><br>
<p><br>
<br><br>
</p><br>
<p><br>
<br><br>
</p><br>
<p><br>
<br><br>
</p><br>
<p><br>
<br><br>
</p><br>
<p><br>
<span></span><br>
</p><br>
<p><br>
<br><br>
```


