**Mybatis-plus-taste**

发现自己一直逛的超市也叫taste，巧

这里作为taste过程记录的新位置，在这里直接贴图，贴代码都会比较方便，希望能够节省出更多的时间。

# **201903151053**

跑测试用例的时候，如果发现有相关Statement找不到的情况，可以检查一下target相应目录下面是否有XXXMapper.xml.若不存在，则需要手动复制一份对应的mapper.xml配置过去，才可以。

# **201903151345**

发现一个问题，将数据库连接相关的配置存放到属性文件中，mybatis-config.xml用properties标签指定了属性文件配置，调试跟踪源码之后，发现也确实有去读取属性文件内容，但是配置连接池信息处，引用了属性文件(jdbc.properties)中的属性，却没有得到正确的转换，比如driver，它在配置DataSource对象的时候，拿到对应driver的值变成了“${jdbc.driver}”，而不是”com.mysql.jdbc.Driver”。

于是来了兴趣，找了mybatis配置DataSource对象的相关源码，它也没有去读出properties文件内的key，直接将字面量赋值到DataSource中的相应属性。这会是什么原因？难道是先后顺序吗?

# **201903152330**

还是上面说的问题，对DataSource类成员变量设置值的代码有翻到了，觉得mybatis中实现中并没做相关处理——从属性文件中获取同名属性

# **201903171036**

版本号：d5f17395ca91e37d2aeba03144090b272e6d5051

现在在测试分页插件.测试代码挺简单的，测试结果也十分准确。

这次测试的背后知识点是：Mybatis下RowBounds分页实现,在分页拦截器中禁用RowBounds内存分页，使用拼接后带分页语法的sql进行查询。

有关RowBounds，我引用别人的文章过来提供参考。

【原文地址】https://blog.csdn.net/u010077905/article/details/38469653

RowBounds类，用于实现分页查询。RowBounds中有两个属性，offset和limit。



RowBounds在处理分页时，只是简单的把offset之前的数据都skip掉，超过limit之后的数据不取出(内存分页)，上图中的代码取自MyBatis中的DefaultResultSetHandler类。跳过offset之前的数据是由方法skipRows处理，判断数据是否超过了limit则是由shouldProcessMoreRows方法进行判断。简单点说道，就是先把数据全部查询到ResultSet，然后从ResultSet中取出offset和limit之间的数据，这就实现了分页查询。

小结论：内存分页是从数据库中查出所有数据后，mybatis框架排除掉范围外数据。从性能上来说，这种分页方式不是很好。

**201903171246**

分页插件相关的开发以及测试程序都已经完整了，接下来分支版本是补全test用例，同步一下官方的Readme,提交一次git.

完成了上面的事情，那么框架的版本号也跟着推进到 4b6a7d4d9e08e1c987cfec73d4d78f593a3d28a0

**201903171519**

版本号：0bacdf4908fb4d8430edb9683e226e4840842dff

增加spring的支持，在wiki包下展示了结合spring的相关mybatis的配置，才能够如常使用plus增强的功能。

**201903212259**

版本号：3389a0a89d7d4e8e4eebebedbc246ea9b6d8ea3a

来到了这里，主要内容是提供Mybatis-plus在Spring多环境中的支持。但是我理解的可能有点不到位，也缺乏去试验的兴趣。在这个版本之后的commit说是增加对web mvc的支持，这就让我比较疑惑，为什么会有这些.？所以我决定观察一下，若没有相关的测试用例，我就不尝试那一部分了。

**201903222117**

版本号：7d999652e14e062d0efc433f28c1eec6fc87b821

这次是对分页插件中，查询总数量方法的修改，写得很明显，不支持含有”order by”关键字地count查询。虽然觉得处理得有点马虎，但是这还是版本的初期，肯定是以大方向为主，细节属于日常优化阶段处理。

这次的疑问是：

~~为什么order by对查询数据总数有影响？~~哈哈，这个问题本身就问的不对。count()会对order by产生什么影响？

在网上找到：count() 会造成order by 排序无效，这点倒是正确，count()把结果归并为一个数据，而order by的目的可能是为了弄一个有序数据集出来。

但是，这里貌似不影响结果。

原来，这里是不可以按照个人想法去盲猜的，下结论的根据是怎样，心里要明白才好。

我从count方法中找出查找总记录数的sql，实验一下：

SELECT COUNT(0) FROM ( originalSQL ) AS TOTAL;

这里sql和我盲猜用的查询总记录数sql，两条语句根本就不同，结果也很难走到一块过去。

实践一下mybatis-plus用的这一条就明白了。



经过实践，我看到这种select count(0)的方式并没有什么特别，照样能够计算出正确的结果。所以就不明白了官方为什么要做这次的修改。

**201903232231**

版本号:3adee82963762c966e6398d39a82b8ff44271f23

添加批量操作 hubin 2016/2/15 18:14

新的api: deleteByIds(List<Long>), insertBatch(List<T>)

**201904021745**

版本号: 3adee82963762c966e6398d39a82b8ff44271f23

还是这一个分支，这次跑通了测试，能够执行新api：批量删除与批量增加。

这次的taste中，发现原本作为 AutoInjector 类的静态常量（代表sql语句，mapper-statement-id），被抽取到SqlMethod枚举类中,代码逻辑变得更加清晰起来。

commit.

**201904021758**

版本号：8769c675ec212d85863a80f13b019886520e2cae

需要纠正一些，上一个版本中，批量删除api并不属于mapper增强的sql部分，所以我上面说的是有误的。

不过，在这次的版本中，大致地了解到mp开始为mapper类增强了批量删除，批量修改的功能。

**201904004**

[1159]

版本号：8769c675ec212d85863a80f13b019886520e2cae，6670ee72eb8d13baab6872578eae846d126f9bd4

结合这两个来看，mp一开始打算做批量修改的增强，后面我在taste这部分的过程中实在是别扭到极点，印象中mysql也没有类似可以用来修改多个对象的update语句啊.现在结合6670ee版本的修改后，也就释疑了。mp框架中后面也是舍弃掉批量修改这部分，毕竟正确的sql语句都没。。

[1211]

探索到一个地方，languageDriver.createSqlSource()方法，它是做什么的呢？

**201904005**

[1228]

taste了deleteBatch增强部分的测试，有一个要注意的地方。如果不了解的话，出错时候可能会找的一头雾水。



上图展示了解决问题的方法。这到底是一个怎样的问题？

像涉及到批量的增强操作：删除，插入。都是借助mybatis框架提供的<foreach>那些标签来实现动态数据需求。直接引用foreach标签的测试结果如下；



上面的错误中，仔细看多几次即可明白，它的报错原因是使用了参数#{item}，但是由于foreach标签并没有被解析，得不到参数item，可使用的参数只有collection或者list。在这里，只需要想办法让foreach标签得到解析就可以了。

在mp的做法中，我总结了该方法的步骤

1. 在有引用到mybatis特殊标签（暂时不知如何叫，明白是哪个就好）时候，需要做一下标记，用<script>标签包含起来。
2. 使用XMLLanguageDriver对象来构造SqlSource对象。它内部会判断sql字符串是否包含script标签，若有，会按照mybatis那一套标签解析去eval此字符串.



[2241]

版本号：6670ee72eb8d13baab6872578eae846d126f9bd4

此版本taste结束。