1. **Mybatis分页插件 - PageHelper说明**

如果你也在用Mybatis，建议尝试该分页插件，这个一定是**最方便**使用的分页插件。

该插件目前支持Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库分页。

[点击提交BUG](http://git.oschina.net/free/Mybatis_PageHelper/issues/new?issue%5Bassignee_id%5D=&issue%5Bmilestone_id%5D=)

# 版本说明

## 最新版本为3.7.5

PageInfo中的judgePageBoudary方法修改：

isLastPage = pageNum == pages && pageNum != 1;

//改为

isLastPage = pageNum == pages;

## 版本3.4.1

* 重大bug修复，SqlParser解析sql失败的时候返回了不带count(\*)的sql，导致查询失败。
* 产生原因，由于SqlParser在系统中出现的位置修改后，导致这里出现错误。
* 强烈推荐各位更新到最新版本。

## 版本3.4.0

* 增加了对@SelectProvider注解方法的支持，不使用这种方式的不影响
* 对基本逻辑进行修改，减少反射调用和获取BoundSql次数
* 对支持的数据库全部通过完整测试
* 虽然变化大，但是影响范围小，不特殊通知各位了。

# 使用方法

分页插件项目中的正式代码一共有个5个Java文件，这5个文件的说明如下：

* Page<E>[必须]：分页参数类，该类继承ArrayList，虽然分页查询返回的结果实际类型是Page<E>,但是可以完全不出现所有的代码中，可以直接当成List使用。返回值不建议使用Page，建议仍然用List。如果需要用到分页信息，使用下面的PageInfo类对List进行包装即可。
* PageHelper[必须]：分页插件拦截器类，对Mybatis的拦截在这个类中实现。
* PageInfo[可选]：Page<E>的包装类，包含了全面的分页属性信息。
* SqlParser[可选]：提供高效的count查询sql。主要是智能替换原sql语句为count(\*)，去除不带参数的order by语句。需要jsqlparser-0.9.1.jar支持。
* SqlUtil[必须]：分页插件工具类，分页插件逻辑类，分页插件的主要实现方法都在这个类中。

## 引入分页插件

引入分页插件一共有下面3种方式，推荐使用引入分页代码的方式，这种方式易于控制，并且可以根据自己需求进行修改。

### 引入分页代码

将本插件中的com.github.pagehelper包（[点击进入gitosc包](http://git.oschina.net/free/Mybatis_PageHelper/tree/master/src/main/java/com/github/pagehelper) | [点击进入github包](https://github.com/pagehelper/Mybatis-PageHelper/tree/master/src/main/java/com/github/pagehelper)）下面的三个类Page,PageHelper和SqlUtil放到项目中，如果需要使用PageInfo（强大的分页包装类），也可以放到项目中。如果想使用更高效的count查询，你也可以将SqlParser放到SqlUtil相同的包下，使用SqlParser时必须使用jsqlparser-0.9.1.jar。

### 引入Jar包

如果你想使用本项目的jar包而不是直接引入类，你可以在这里下载各个版本的jar包（点击Download下的jar即可下载）

* <https://oss.sonatype.org/content/repositories/releases/com/github/pagehelper/pagehelper/>
* <http://repo1.maven.org/maven2/com/github/pagehelper/pagehelper/>

由于使用了sql解析工具，你还需要下载这个文件（这个文件完全独立，不依赖其他）：

* SqlParser.jar：<http://search.maven.org/remotecontent?filepath=com/github/jsqlparser/jsqlparser/0.9.1/jsqlparser-0.9.1.jar>
* SqlParser - github地址：<https://github.com/JSQLParser/JSqlParser>

### 使用maven --> 使用这一个

添加如下依赖：

<dependency>

<groupId>com.github.pagehelper</groupId>

<artifactId>pagehelper</artifactId>

<version>3.7.5</version>

</dependency>

<dependency>

<groupId>com.github.jsqlparser</groupId>

<artifactId>jsqlparser</artifactId>

<version>0.9.1</version>

</dependency>

## 在Mybatis配置xml中配置拦截器插件:

<!--

plugins在配置文件中的位置必须符合要求，否则会报错，顺序如下:

properties?, settings?,

typeAliases?, typeHandlers?,

objectFactory?,objectWrapperFactory?,

plugins?,

environments?, databaseIdProvider?, mappers?

-->

<plugins>

<!-- com.github.pagehelper为PageHelper类所在包名 -->

<plugin interceptor="com.github.pagehelper.PageHelper">

<property name="dialect" value="mysql"/>

<!-- 该参数默认为false -->

<!-- 设置为true时，会将RowBounds第一个参数offset当成pageNum页码使用 -->

<!-- 和startPage中的pageNum效果一样-->

<property name="offsetAsPageNum" value="true"/>

<!-- 该参数默认为false -->

<!-- 设置为true时，使用RowBounds分页会进行count查询 -->

<property name="rowBoundsWithCount" value="true"/>

<!-- 设置为true时，如果pageSize=0或者RowBounds.limit = 0就会查询出全部的结果 -->

<!-- （相当于没有执行分页查询，但是返回结果仍然是Page类型）-->

<property name="pageSizeZero" value="true"/>

<!-- 3.3.0版本可用 - 分页参数合理化，默认false禁用 -->

<!-- 启用合理化时，如果pageNum<1会查询第一页，如果pageNum>pages会查询最后一页 -->

<!-- 禁用合理化时，如果pageNum<1或pageNum>pages会返回空数据 -->

<property name="reasonable" value="true"/>

</plugin>

</plugins>

这里的com.github.pagehelper.PageHelper使用完整的类路径。

其他五个参数说明：

1. 增加dialect属性，使用时必须指定该属性，可选值为oracle,mysql,mariadb,sqlite,hsqldb,postgresql,**没有默认值，必须指定该属性**。
2. 增加offsetAsPageNum属性，默认值为false，使用默认值时不需要增加该配置，需要设为true时，需要配置该参数。当该参数设置为true时，使用RowBounds分页时，会将offset参数当成pageNum使用，可以用页码和页面大小两个参数进行分页。
3. 增加rowBoundsWithCount属性，默认值为false，使用默认值时不需要增加该配置，需要设为true时，需要配置该参数。当该参数设置为true时，使用RowBounds分页会进行count查询。
4. 增加pageSizeZero属性，默认值为false，使用默认值时不需要增加该配置，需要设为true时，需要配置该参数。当该参数设置为true时，如果pageSize=0或者RowBounds.limit = 0就会查询出全部的结果（相当于没有执行分页查询，但是返回结果仍然是Page类型）。
5. 增加reasonable属性，默认值为false，使用默认值时不需要增加该配置，需要设为true时，需要配置该参数。具体作用请看上面配置文件中的注释内容。

### 如何选择配置这些参数

单独看每个参数的说明可能是一件让人不爽的事情，这里列举一些可能会用到某些参数的情况。

首先dialect属性是必须的，不需要解释。其他的参数一般情况下我们都不必去管，如果想了解何时使用合适，你可以参考以下场景：

#### 场景一

如果你仍然在用类似ibatis式的命名空间调用方式，你也许会用到rowBoundsWithCount，分页插件对RowBounds支持和Mybatis默认的方式是一致，默认情况下不会进行count查询，如果你想在分页查询时进行count查询，以及使用更强大的PageInfo类，你需要设置该参数为true。

#### 场景二

如果你仍然在用类似ibatis式的命名空间调用方式，你觉得RowBounds中的两个参数offset,limit不如pageNum,pageSize容易理解，你可以使用offsetAsPageNum参数，将该参数设置为true后，offset会当成pageNum使用，limit和pageSize含义相同。

#### 场景三

如果觉得某个地方使用分页后，你仍然想通过控制参数查询全部的结果，你可以配置pageSizeZero为true，配置后，如可以通过设置pageSize=0或者RowBounds.limit = 0就会查询出全部的结果。

#### 场景四

如果你分页插件使用于类似分页查看列表式的数据，如新闻列表，软件列表，你希望用户输入的页数不在合法范围（第一页到最后一页之外）时能够正确的响应到正确的结果页面，那么你可以配置reasonable为true，这时如果pageNum<1会查询第一页，如果pageNum>总页数会查询最后一页。

## Spring配置方法 --> 不使用这一种

首先需要在Spring中配置org.mybatis.spring.SqlSessionFactoryBean。然后配置配置Mybatis的具体配置有两种方式，一种是用mybatis默认的xml配置，另一种就是完全使用spring的属性配置方式。

### mybatis默认的xml配置

配置configLocation属性指向上面的mybatis-config.xml文件。有关分页插件的配置都在mybatis-config.xml，具体配置内容参考上面的mybatis-config.xml。

### 使用spring的属性配置方式

使用spring的属性配置方式，可以使用plugins属性像下面这样配置：

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

<property name="dataSource" ref="dataSource"/>

<property name="mapperLocations">

<array>

<value>classpath:mapper/\*.xml</value>

</array>

</property>

<property name="typeAliasesPackage" value="com.isea533.ssm.model"/>

<property name="plugins">

<array>

<bean class="com.github.pagehelper.PageHelper">

<property name="properties">

<value>

dialect=hsqldb

reasonable=true

</value>

</property>

</bean>

</array>

</property>

</bean>

属性配置按照上面的方式配置，每个配置独立一行即可。

## 如何在代码中使用

阅读前后请注意看[重要提示](https://github.com/pagehelper/Mybatis-PageHelper" \l "%E9%87%8D%E8%A6%81%E6%8F%90%E7%A4%BA)

首先该分页插件支持以下两种调用方式：

//第一种，RowBounds方式的调用

List<Country> list = sqlSession.selectList("x.y.selectIf", null, new RowBounds(1, 10));

//第二种，Mapper接口方式的调用，推荐这种使用方式。 --> 使用这种

PageHelper.startPage(1, 10); // 必须在查询条件之前使用

List<Country> list = countryMapper.selectIf(1);

下面分别对这两种方式进行详细介绍

### RowBounds方式的调用

List<Country> list = sqlSession.selectList("x.y.selectIf", null, new RowBounds(1, 10));

使用这种调用方式时，你可以使用RowBounds参数进行分页，这种方式侵入性最小，我们可以看到，通过RowBounds方式调用只是使用了这个参数，并没有增加其他任何内容。

分页插件检测到使用了RowBounds参数时，就会对该查询进行**物理分页**。

关于这种方式的调用，有两个特殊的参数是针对RowBounds的，你可以参看上面的[场景一](https://github.com/pagehelper/Mybatis-PageHelper" \l "%E5%9C%BA%E6%99%AF%E4%B8%80)和[场景二](https://github.com/pagehelper/Mybatis-PageHelper" \l "%E5%9C%BA%E6%99%AF%E4%BA%8C)

**注：**不只有命名空间方式可以用RowBounds，使用接口的时候也可以增加RowBounds参数，例如：

//这种情况下也会进行物理分页查询

List<Country> selectAll(RowBounds rowBounds);

### PageHelper.startPage静态方法调用

在你需要进行分页的Mybatis方法前调用PageHelper.startPage静态方法即可，紧跟在这个方法后的第一个**Mybatis查询方法**会被进行分页。

#### 例一：

SqlSession sqlSession = MybatisHelper.getSqlSession();

CountryMapper countryMapper = sqlSession.getMapper(CountryMapper.class);

try {

//获取第1页，10条内容，默认查询总数count

PageHelper.startPage(1, 10);

//紧跟着的第一个select方法会被分页

List<Country> list = countryMapper.selectIf(1);

assertEquals(2, list.get(0).getId());

assertEquals(10, list.size());

//分页时，实际返回的结果list类型是Page<E>，如果想取出分页信息，需要强制转换为Page<E>，

//或者使用PageInfo类（下面的例子有介绍）

assertEquals(182, ((Page) list).getTotal());

} finally {

sqlSession.close();

}

#### 例二：

SqlSession sqlSession = MybatisHelper.getSqlSession();

CountryMapper countryMapper = sqlSession.getMapper(CountryMapper.class);

try {

//获取第1页，10条内容，默认查询总数count

PageHelper.startPage(1, 10);

//紧跟着的第一个select方法会被分页

List<Country> list = countryMapper.selectIf(1);

//后面的不会被分页，除非再次调用PageHelper.startPage

List<Country> list2 = countryMapper.selectIf(null);

//list1

assertEquals(2, list.get(0).getId());

assertEquals(10, list.size());

//分页时，实际返回的结果list类型是Page<E>，如果想取出分页信息，需要强制转换为Page<E>，

//或者使用PageInfo类（下面的例子有介绍）

assertEquals(182, ((Page) list).getTotal());

//list2

assertEquals(1, list2.get(0).getId());

assertEquals(182, list2.size());

} finally {

sqlSession.close();

}

### 例三，使用PageInfo的用法： --> 使用这个方法

//获取第1页，10条内容，默认查询总数count

PageHelper.startPage(1, 10);

List<Country> list = countryMapper.selectAll();

//用PageInfo对结果进行包装

PageInfo page = new PageInfo(list);

//测试PageInfo全部属性

//PageInfo包含了非常全面的分页属性

assertEquals(1, page.getPageNum());

assertEquals(10, page.getPageSize());

assertEquals(1, page.getStartRow());

assertEquals(10, page.getEndRow());

assertEquals(183, page.getTotal());

assertEquals(19, page.getPages());

assertEquals(1, page.getFirstPage());

assertEquals(8, page.getLastPage());

assertEquals(true, page.isFirstPage());

assertEquals(false, page.isLastPage());

assertEquals(false, page.isHasPreviousPage());

assertEquals(true, page.isHasNextPage());

本项目中包含大量测试，您可以通过查看测试代码了解使用方法。

测试代码地址：<http://git.oschina.net/free/Mybatis_PageHelper/tree/master/src/test/java/com/github/pagehelper/test>

# 重要提示

## PageHelper.startPage方法重要提示

只有紧跟在PageHelper.startPage方法后的**第一个**Mybatis的**查询（Select方法）**方法会被分页。

## 分页插件不支持带有for update语句的分页

对于带有for update的sql，会抛出运行时异常，对于这样的sql建议手动分页，毕竟这样的sql需要重视。

## 分页插件不支持关联结果查询

原因以及解决方法可以看这里：

<http://my.oschina.net/flags/blog/274000>

分支插件不支持关联结果查询，但是支持关联嵌套查询。只会对主sql进行分页，嵌套的sql不会被分页。

# 相关链接

对应于oschub的项目地址：<http://git.oschina.net/free/Mybatis_PageHelper>

对应于github的项目地址：<https://github.com/pagehelper/Mybatis-PageHelper>

Mybatis-Sample（分页插件测试项目）：<http://git.oschina.net/free/Mybatis-Sample>

Mybatis项目：<https://github.com/mybatis/mybatis-3>

Mybatis文档：<http://mybatis.github.io/mybatis-3/zh/index.html>

Mybatis专栏：

* [Mybatis示例](http://blog.csdn.net/column/details/mybatis-sample.html)
* [Mybatis问题集](http://blog.csdn.net/column/details/mybatisqa.html)

作者博客：

* <http://my.oschina.net/flags/blog>
* <http://blog.csdn.net/isea533>

作者QQ： 120807756

作者邮箱： [abel533@gmail.com](mailto:abel533@gmail.com)

Mybatis工具群： 211286137 (Mybatis相关工具插件等等)