> 如果你也在用 MyBatis，建议尝试该分页插件，这一定是**最方便**使用的分页插件。

分页插件的必要性

互联网应用中，分页可谓无处不在，在每个需要展示数据的地方，都能找到分页的影子。在日常开发中，为了追求效率，通常使用数据库的物理分页。这时，对于一个业务逻辑SQL，大多数情况需要输出两段SQL来达到分页效果：count查询总数和limit分页，这无疑增加了大量的工作量。对于这种大量的、相似的、非业务逻辑的代码，抽象出公共插件是势在必行的。

分页插件原理

Mybatis给开发者提供了一个拦截器接口，只要实现了该接口，就可以在Mybatis执行SQL前，作一些自定义的操作。分页插件就是在此基础上开发出来的，对于一个需要分页的SQL，插件会拦截并生成两段SQL。举一个简单的例子：

原SQL：

```
select * from table where a = '1'
```

拦截后的查询总数SQL：

```
select count(*) from table where a = '1'
```

拦截后的分页SQL：

```
select * from table where a = '1' limit 5,10
```

这样我们只需要根据业务逻辑开发原SQL，不需关心分页语法对原SQL的影响，拦截器已经为我们处理好了。更多拦截器的信息可以参考：

- QueryInterceptor源码
- Executor 拦截器高级教程 - QueryInterceptor 规范

支持的数据库

该插件目前支持以下数据库的物理分页:

1. Oracle
2. Mysql
3. MariaDB
4. SQLite
5. Hsqldb
6. PostgreSQL
7. DB2
8. SqlServer(2005,2008)
9. Informix
10. H2
11. SqlServer2012
12. Derby
13. Phoenix

与SpringMVC集成

Maven依赖

```
<!-- 分页插件 -->
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper</artifactId>
    <version>5.0.0</version>
</dependency>
```

Spring配置文件

只需要在原来配置Mybatis的 `SqlSessionFactoryBean` 的地方加上分页插件的配置即可，具体区别请看以下的对比：

原来的配置方式：

```
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dynamicDataSource"/>
    <property name="mapperLocations">
        <list>
            <value>classpath:mapper/*.xml</value>
```

```
            <value>classpath:mapper/*/*.xml</value>
        </list>
    </property>
</bean>
```

加上分页插件的配置方式：

```
<bean id="sqlSessionFactory"
class="org.mybatis.spring.SqlSessionFactoryBean">
    <property name="dataSource" ref="dynamicDataSource"/>
    <property name="mapperLocations">
        <list>
            <value>classpath:mapper/*.xml</value>
            <value>classpath:mapper/*/*.xml</value>
        </list>
    </property>
    <!-- 配置分页插件 -->
    <property name="plugins">
        <array>
            <bean class="com.github.pagehelper.PageInterceptor">
                <property name="properties">
                    <value>
                        helperDialect=postgresql
                        reasonable=true
                    </value>
                </property>
            </bean>
        </array>
    </property>
</bean>
```

可以看到仅仅是加了`<property name="plugins">`的配置。在`<property name="properties">`里可以配置分页参数，一般情况下配置数据库类型`helperDialect`即可。完整的参数如下：

```
<!-- 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"/>
<!-- 3.5.0版本可用 - 为了支持startPage(Object params)方法 -->
<!-- 增加了一个`params`参数来配置参数映射，用于从Map或ServletRequest中取值 -->
<!-- 可以配置pageNum,pageSize,count,pageSizeZero,reasonable,不配置映射的用默认值 -->
<property name="params"
value="pageNum=start;pageSize=limit;pageSizeZero=zero;reasonable=heli;count=contsql"/>
```

在代码中使用

在需要进行分页的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，然后分页插件会把分页信息封装到 `PageInfo` 中。

```
// startPage(第几页, 多少条数据)
PageHelper.startPage(pageIndex, pageSize);
// Mybatis查询方
List<InstanceVO> list = instanceDao.select(instance);
// 用PageInfo对结果进行包装
PageInfo pageInfo = new PageInfo(list);
```

以这种对原SQL无侵害的方法，就可以得到分页的效果和详细的分页信息。PageInfo包含了非常全面的分页属性：

```
public class PageInfo<T> implements Serializable {
    private static final long serialVersionUID = 1L;
    //当前页
    private int pageNum;
    //每页的数量
    private int pageSize;
    //当前页的数量
    private int size;
    //由于startRow和endRow不常用，这里说个具体的用法
```

```
    //可以在页面中"显示startRow到endRow 共size条数据"
    //当前页面第一个元素在数据库中的行号
    private int startRow;
    //当前页面最后一个元素在数据库中的行号
    private int endRow;
    //总记录数
    private long total;
    //总页数
    private int pages;
    //结果集
    private List<T> list;
    //第一页
    private int firstPage;
    //前一页
    private int prePage;
    //下一页
    private int nextPage;
    //最后一页
    private int lastPage;
    //是否为第一页
    private boolean isFirstPage = false;
    //是否为最后一页
    private boolean isLastPage = false;
    //是否有前一页
    private boolean hasPreviousPage = false;
    //是否有下一页
    private boolean hasNextPage = false;
    //导航页码数
    private int navigatePages;
    //所有导航页号
    private int[] navigatepageNums;
    …
}
```

具体的例子

原SQL：

```
select
id,name,create_time,create_user_id,update_time,update_user_id,is_delete
from xxx.aaa
where
    ( is_delete = ? )
```

拦截后的SQL（源自Mybatis日志信息）：

```
# count
select count(0) from xxx.aaa WHERE (is_delete = ?)
```

```
# 分页
select

id,name,create_time,create_user_id,update_time,update_user_id,is_delete
from xxx.aaa
where
    ( is_delete = ? ) LIMIT 3
```

返回的json数据：
返回数据用一个自定义类 `Result` 来封装， `models` 是业务数据， `paging` 是分页信息。

```
{
    "models":[
        {
            "createTime":1508890619000,
            "createUserId":"888888",
            "id":3,
            "isDelete":0,
            "name":"【TEST】",
            "updateTime":1512373972000,
            "updateUserId":"888888"
        },
        {
            "createTime":1508890619000,
            "createUserId":"888888",
            "id":4,
            "isDelete":0,
            "name":"bbb",
            "updateTime":1508891132000,
            "updateUserId":"888888"
        },
        {
            "createTime":1508890619000,
            "createUserId":"888888",
            "id":5,
            "isDelete":0,
            "name":"ccc",
```

```
            "updateTime":1508891132000,
            "updateUserId":"888888"
        }
    ],
    "paging":{
        "endRow":3,
        "firstPage":1,
        "hasNextPage":true,
        "hasPreviousPage":false,
        "isFirstPage":true,
        "isLastPage":false,
        "lastPage":5,
        "navigateFirstPage":1,
        "navigateLastPage":5,
        "navigatePages":8,
        "navigatepageNums":[1,2,3,4,5],
        "nextPage":2,
        "pageNum":1,
        "pageSize":3,
        "pages":5,
        "prePage":0,
        "size":3,
        "startRow":1,
        "total":15
    },
    "resultCode":"100",
    "success":true,
    "valid":true
}
```


总结

使用分页插件时，不需要在分页的地方手写分页SQL和count的SQL，不需要更改已有的业务代码，只需要在执行SQL前调用一句代码即可实现分页，并得到丰富的分页信息。有了这些分页信息，前端可以选用多种分页方法，非常方便！

参考

- https://github.com/pagehelper/Mybatis-PageHelper/blob/master/README_zh.md
- https://gitee.com/free/Mybatis_PageHelper
- http://www.ciphermagic.cn/mybatis-page.html