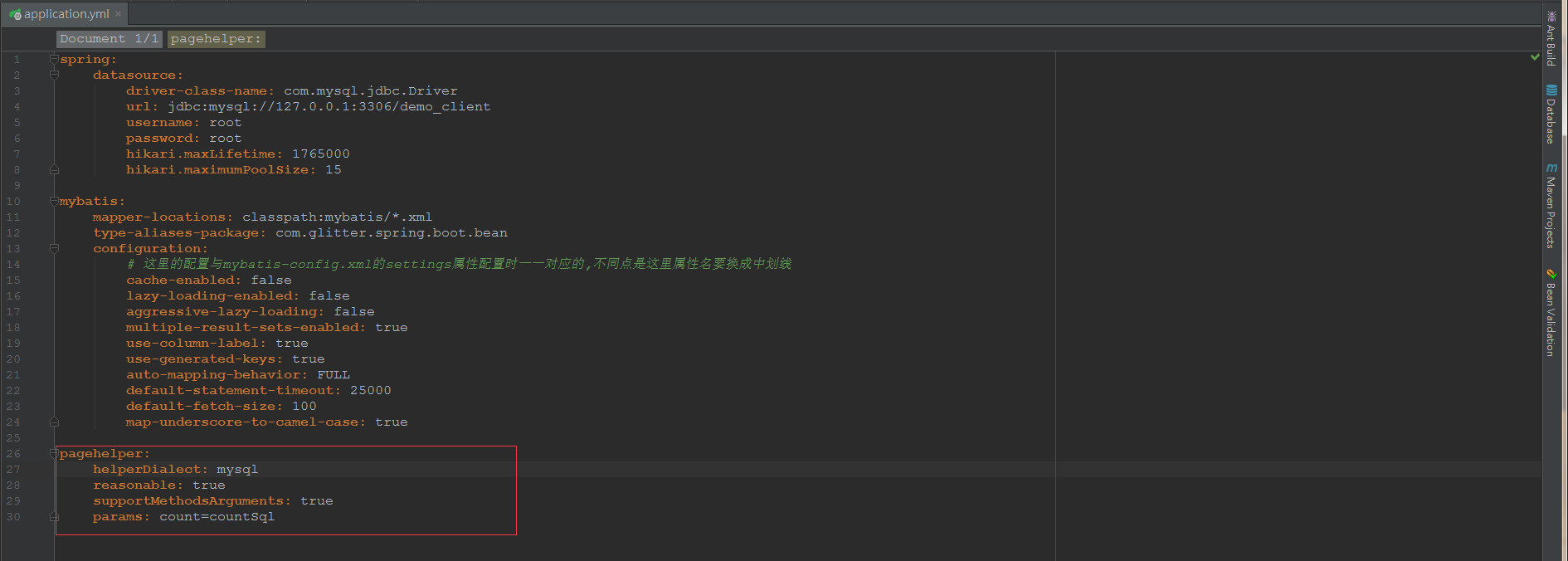
1.官方集成文档。

https://github.com/pagehelper/Mybatis-PageHelper/blob/master/README\_zh.md

https://github.com/pagehelper/Mybatis-PageHelper/blob/master/wikis/zh/HowToUse.md

2.该插件不做任何配置也可以直接使用，其默认的运行模式已经介绍过，这里不再赘述，只是如果什么也配置，那么他的执行行为是按照其默认的内置参数值来执行的，其效果未必是我们想要的，所以，我们还是要配置一些他的属性，以便符合我们的分页要求。

3.配置插件的特性。

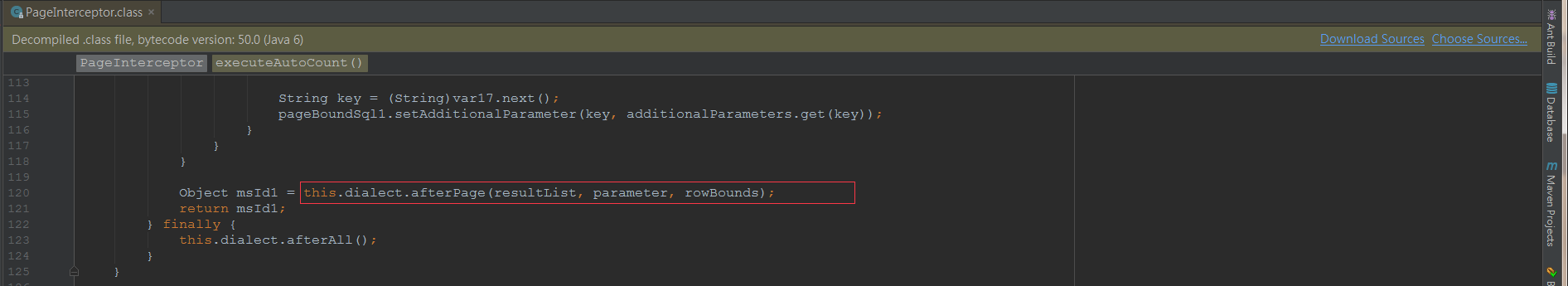


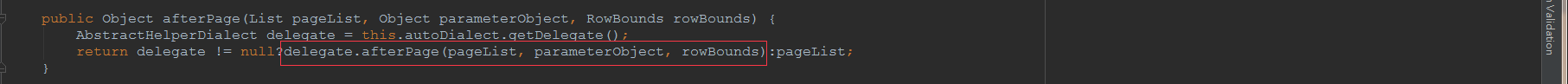
4.另外，该插件的中间运算对象Page类继承了ArrayList。

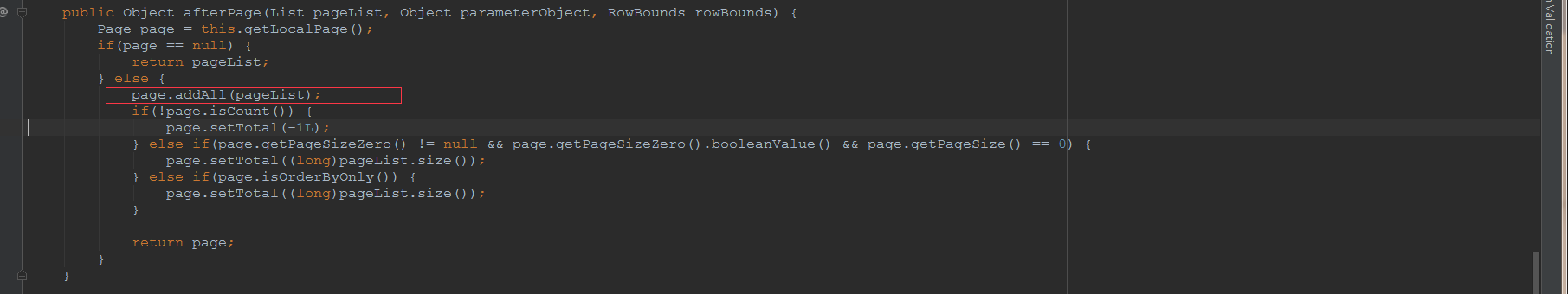
public class Page<E> extends ArrayList<E> implements Closeable {

结果集合实际上是用Page对象做的接收，也就是说这里的userInfos实际上就是Page对象，具体可以看其源码。

List<UserInfo> userInfos = userInfoDao.findAllList();







5.分页结束后，其最后会将Page对象转换成PageInfo对象进行返回。

package com.github.pagehelper;  
  
import com.github.pagehelper.Page;  
import com.github.pagehelper.PageSerializable;  
import java.util.Collection;  
import java.util.List;  
  
public class PageInfo<T> extends PageSerializable<T> {  
 private int pageNum;  
 private int pageSize;  
 private int size;  
 private int startRow;  
 private int endRow;  
 private int pages;  
 private int prePage;  
 private int nextPage;  
 private boolean isFirstPage;  
 private boolean isLastPage;  
 private boolean hasPreviousPage;  
 private boolean hasNextPage;  
 private int navigatePages;  
 private int[] navigatepageNums;  
 private int navigateFirstPage;  
 private int navigateLastPage;  
  
 public PageInfo() {  
 this.isFirstPage = false;  
 this.isLastPage = false;  
 this.hasPreviousPage = false;  
 this.hasNextPage = false;  
 }  
  
 public PageInfo(List<T> list) {  
 this(list, 8);  
 }  
  
 public PageInfo(List<T> list, int navigatePages) {  
 super(list);  
 this.isFirstPage = false;  
 this.isLastPage = false;  
 this.hasPreviousPage = false;  
 this.hasNextPage = false;  
 if(list instanceof Page) {  
 Page page = (Page)list;  
 this.pageNum = page.getPageNum();  
 this.pageSize = page.getPageSize();  
 this.pages = page.getPages();  
 this.size = page.size();  
 if(this.size == 0) {  
 this.startRow = 0;  
 this.endRow = 0;  
 } else {  
 this.startRow = page.getStartRow() + 1;  
 this.endRow = this.startRow - 1 + this.size;  
 }  
 } else if(list instanceof Collection) {  
 this.pageNum = 1;  
 this.pageSize = list.size();  
 this.pages = this.pageSize > 0?1:0;  
 this.size = list.size();  
 this.startRow = 0;  
 this.endRow = list.size() > 0?list.size() - 1:0;  
 }  
  
 if(list instanceof Collection) {  
 this.navigatePages = navigatePages;  
 this.calcNavigatepageNums();  
 this.calcPage();  
 this.judgePageBoudary();  
 }  
  
 }  
  
 public static <T> PageInfo<T> of(List<T> list) {  
 return new PageInfo(list);  
 }  
  
 public static <T> PageInfo<T> of(List<T> list, int navigatePages) {  
 return new PageInfo(list, navigatePages);  
 }  
  
 private void calcNavigatepageNums() {  
 int startNum;  
 if(this.pages <= this.navigatePages) {  
 this.navigatepageNums = new int[this.pages];  
  
 for(startNum = 0; startNum < this.pages; ++startNum) {  
 this.navigatepageNums[startNum] = startNum + 1;  
 }  
 } else {  
 this.navigatepageNums = new int[this.navigatePages];  
 startNum = this.pageNum - this.navigatePages / 2;  
 int endNum = this.pageNum + this.navigatePages / 2;  
 int i;  
 if(startNum < 1) {  
 startNum = 1;  
  
 for(i = 0; i < this.navigatePages; ++i) {  
 this.navigatepageNums[i] = startNum++;  
 }  
 } else if(endNum > this.pages) {  
 endNum = this.pages;  
  
 for(i = this.navigatePages - 1; i >= 0; --i) {  
 this.navigatepageNums[i] = endNum--;  
 }  
 } else {  
 for(i = 0; i < this.navigatePages; ++i) {  
 this.navigatepageNums[i] = startNum++;  
 }  
 }  
 }  
  
 }  
  
 private void calcPage() {  
 if(this.navigatepageNums != null && this.navigatepageNums.length > 0) {  
 this.navigateFirstPage = this.navigatepageNums[0];  
 this.navigateLastPage = this.navigatepageNums[this.navigatepageNums.length - 1];  
 if(this.pageNum > 1) {  
 this.prePage = this.pageNum - 1;  
 }  
  
 if(this.pageNum < this.pages) {  
 this.nextPage = this.pageNum + 1;  
 }  
 }  
  
 }  
  
 private void judgePageBoudary() {  
 this.isFirstPage = this.pageNum == 1;  
 this.isLastPage = this.pageNum == this.pages || this.pages == 0;  
 this.hasPreviousPage = this.pageNum > 1;  
 this.hasNextPage = this.pageNum < this.pages;  
 }

6 .如果我们使用PageInfo作为返回对象不习惯的话，可以进一步再对其进行包装，然后返回我们自己的PageInfo对象。

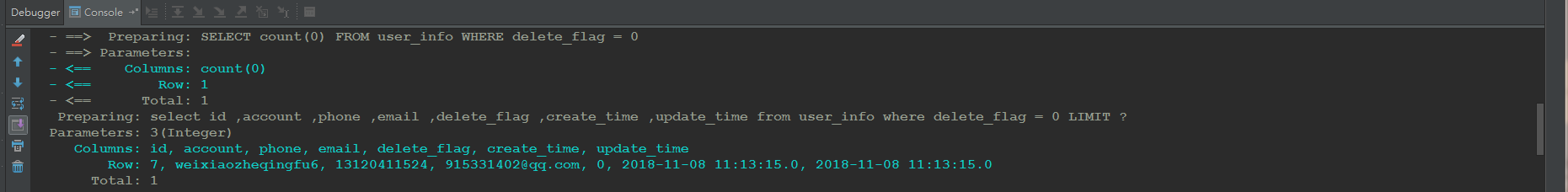
public class PageInfo<T> implements Serializable {  
  
 private static final long *serialVersionUID* = -4647032854428993236L;  
  
 */\*\* 每页记录数 \*/* private int pageSize;  
 */\*\* 页码 \*/* private int pageNum;  
 */\*\* 总记录数 \*/* private long totalRecords;  
 */\*\* 总页数\*/* private int totalPages;  
 */\*\* 结果集 \*/* private List<T> data;  
  
 */\*\* 首页 \*/* private int firstPageNum;  
 */\*\* 尾页 \*/* private int lastPageNum;  
 */\*\* 上一页 \*/* private int prePageNum;  
 */\*\* 下一页 \*/* private int nextPageNum;  
 */\*\* 是否有上一页 \*/* private boolean hasPrePage;  
 */\*\* 是否有下一页 \*/* private boolean hasNextPage;  
  
 public PageInfo(List<T> data){  
 com.github.pagehelper.PageInfo pageInfoGithub = new com.github.pagehelper.PageInfo(data);  
 if(null != pageInfoGithub){  
 this.pageNum = pageInfoGithub.getPageNum();  
 this.pageSize = pageInfoGithub.getPageSize();  
 this.totalRecords = pageInfoGithub.getTotal();  
 this.totalPages = pageInfoGithub.getPages();  
 this.firstPageNum = pageInfoGithub.getFirstPage();  
 this.lastPageNum = pageInfoGithub.getLastPage();  
 this.prePageNum = pageInfoGithub.getPrePage();  
 this.nextPageNum = pageInfoGithub.getNextPage();  
 this.hasPrePage = pageInfoGithub.isHasPreviousPage();  
 this.hasNextPage = pageInfoGithub.isHasNextPage();  
 }  
 this.data = data;  
 }  
  
  
 public int getPageSize() {  
 return pageSize;  
 }  
  
 public int getPageNum() {  
 return pageNum;  
 }  
  
 public long getTotalRecords() {  
 return totalRecords;  
 }  
  
 public int getTotalPages() {  
 return totalPages;  
 }  
  
 public List<T> getData() {  
 return data;  
 }  
  
 public int getFirstPageNum() {  
 return firstPageNum;  
 }  
  
 public int getLastPageNum() {  
 return lastPageNum;  
 }  
  
 public int getPrePageNum() {  
 return prePageNum;  
 }  
  
 public int getNextPageNum() {  
 return nextPageNum;  
 }  
  
 public boolean isHasPrePage() {  
 return hasPrePage;  
 }  
  
 public boolean isHasNextPage() {  
 return hasNextPage;  
 }

7.代码中使用方式。

@Override  
public PageInfo<UserInfo> getUserInfosPage(Integer pageNum, Integer pageSize) {  
 PageHelper.*startPage*(pageNum, pageSize);  
 List<UserInfo> userInfos = userInfoDao.findAllList();  
 PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);  
 return pageInfo;  
}

8.执行效果，不同的情况与我的自定义分页插件执行结果可能会有出入，两个插件的计算策略都说的通，就看习惯用哪个。然后使用的时候与各方尤其是前端沟通好即可。

另外，它的这个插件计算结果带导航，我的插件比较简约，没有导航功能，就是中间的…，如果需要，后期可以加上。另外现在一般都是后端返回极简的PageInfo，那些导航还有上一下下一页首页尾页之类的一般都交给前端分页插件自行计算了。



<http://localhost:8080/userInfo/getUserInfosPage?pageNum=-1&pageSize=3>

{

"code": "0",

"message": "操作成功！",

"data": {

"total": 1,

"list": [

{

"id": 7,

"account": "weixiaozheqingfu6",

"password": null,

"phone": "13120411524",

"email": "915331402@qq.com",

"phoneVerified": null,

"emailVerified": null,

"fullName": null,

"nickName": null,

"age": null,

"height": null,

"remark": null,

"deleteFlag": false,

"registerTime": null,

"createTime": "2018-11-08T03:13:15.000+0000",

"updateTime": "2018-11-08T03:13:15.000+0000"

}

],

"pageNum": 1,

"pageSize": 3,

"size": 1,

"startRow": 1,

"endRow": 1,

"pages": 1,

"prePage": 0,

"nextPage": 0,

"isFirstPage": true,

"isLastPage": true,

"hasPreviousPage": false,

"hasNextPage": false,

"navigatePages": 8,

"navigatepageNums": [

1

],

"navigateFirstPage": 1,

"navigateLastPage": 1,

"firstPage": 1,

"lastPage": 1

}

}

9.我们如果觉得这个插件自带的PageInfo的返回结果内容太多，或者对其字段命名不太习惯，那么我们可以对其再做一次包装，返回自己的PageInfo对象。

一切都不变，只要项目中有自己的PageInfo对象类，并且PageInfo<UserInfo> pageInfo = new PageInfo(userInfos);代码导入的是自己的PageInfo的类即可了。

执行效果如下：

<http://localhost:8080/userInfo/getUserInfosPage?pageNum=-1&pageSize=3>

{

"code": "0",

"message": "操作成功！",

"data": {

"pageSize": 3,

"pageNum": 1,

"totalRecords": 1,

"totalPages": 1,

"data": [

{

"id": 7,

"account": "weixiaozheqingfu6",

"password": null,

"phone": "13120411524",

"email": "915331402@qq.com",

"phoneVerified": null,

"emailVerified": null,

"fullName": null,

"nickName": null,

"age": null,

"height": null,

"remark": null,

"deleteFlag": false,

"registerTime": null,

"createTime": "2018-11-08T03:13:15.000+0000",

"updateTime": "2018-11-08T03:13:15.000+0000"

}

],

"firstPageNum": 1,

"lastPageNum": 1,

"prePageNum": 0,

"nextPageNum": 0,

"hasPrePage": false,

"hasNextPage": false

}

}