[鼠绘漫画网 1](#_Toc23830)

[一拳超人 1](#_Toc4528)

[方案1，公众号抓包 1](#_Toc9987)

[方案2，解决网页图片懒加载的问题 1](#_Toc22610)

[方案3，解析资源库的URL规则 2](#_Toc4224)

[总结 3](#_Toc25559)

[海贼王 3](#_Toc5534)

[百度贴吧 10](#_Toc24212)

[黑咪，监狱学园 10](#_Toc964)

[剑风传奇 10](#_Toc813)

[飞丸网 10](#_Toc10869)

[东京食尸鬼 10](#_Toc11461)

[Wanted 11](#_Toc17226)

# 鼠绘漫画网

爬取鼠绘漫画网的资源。

## 一拳超人

### 方案1，公众号抓包

即便成功抓包也涉及到脚本的问题，因为我不可能一话一话的点过去。

而我不会python，我也不会shell。

pass。

### **方案2，解决网页图片懒加载的问题**

可以先用crawler4j试试。

2018-03-03 10:51:23

试过了，这吊东西不靠谱，获取的网页源代码还是什么都没有，就算是因为我没研究出来，那这吊东西还是不靠谱，一个好的工具就应该简便易用。搞那么多配置，或入口藏的深的要死，老子哪有那功夫去研究。

鼠标滑动过后，图片才会显示，看起来像是vue.js导致的。

那么现在有两种做法，找到那段js，然后…… ……然后个毛，我js那么差，pass。

第二种做法是继续找工具，可以直接获取js加载后的网页源代码。

2018-03-03 11:22:23

找到了，叫Selenium。

crawler4j还有jsoup这种工具都是直接访问url，那么服务器返回什么，他们就接收什么，静态资源还好说，但这种js加载出来的动态html代码，这些工具就获取不到了，而selenium，这个工具会调用浏览器。因为是靠浏览器去访问的，所以碰到js后，浏览器会自动执行，等浏览器执行完js，渲染出整个页面后，selenium在从浏览器那获取html。

（启动程序后，看着浏览器自动弹开有那么点诡异，跟中毒似的）

我用的是Chrome，所以要下载Chrome的驱动提供给selenium。

<dependency>

<groupId>org.seleniumhq.selenium</groupId>

<artifactId>selenium-java</artifactId>

<version>3.10.0</version>

</dependency>

System.*getProperties*().setProperty("webdriver.chrome.driver","D:\\Workspace\\tools\\chromeDriver\\chromedriver.exe");

WebDriver webDriver = **new** ChromeDriver();

webDriver.get("http://hanhuazu.cc/cartoon/post?id=2292");

String responseBody = webDriver.getPageSource();

System.***out***.println(responseBody);

webDriver.close();

### **方案3，解析资源库的URL规则**

第1话，第1页

http://pic02.ishuhui.com/cartoon/book-7/53/1-6775/0001.jpg?0c850560

第128话，第1页

<http://pic03.ishuhui.com/cartoon/book-7/53/128-10463/0001.jpg?ef791431>

* ？后跟的是随机数，防缓存的，直接过
* 从pic02，变到pic03中间有没有pic02呢

~~我猜鼠绘分域名是因为资源量过大。~~

这你麻痹的就尴尬了，第128话，第1页，我用02，或是01都能正常访问。

[http://pic02.ishuhui.com/cartoon/book-7/53/128-10463/0001.jpg?ef791431](http://pic03.ishuhui.com/cartoon/book-7/53/128-10463/0001.jpg?ef791431)

也就是说，这些域名最后解析的都是同一个ip，那么这个问题我就算解决了。

* cartoon，book-7

cartoon是固定的。

book-7有可能变成book-1，其它的还没见过。这东西好比一个大文件夹。

* /53/1-6775/、/53/128-10463/

这一段的话，53是一拳超人的这部漫画的代号，1表示第一话，128表示第128话。关键是6775以及10463，这些代表某一话的文件资源位置。

这个迭代太麻烦了，资源位置不对会报404，超出这一话的长度也会报404，book-7或是book-1不对还是报404。

我现在不知道怎么去判断这个404是在哪个环节抛出来的。

另外一个问题是访问的太频繁可能被鼠绘服务器禁用IP，限制访问间隔时间的话，像海贼王那种几百话的总计上千页的漫画，又不知道什么时候能爬完。

但这个问题又有点多余，因为我规则还没解析出来。。。。

### **总结**

那就方案二了。

## 海贼王

我本来是想分析鼠绘资源库的url规则，然后迭代的，但这样成本实在太大了，尤其是要冒着被禁ip的风险，不划算。

剩下可行的方案，一个是搞明白，为什么公众号可以显示图片，而用浏览器，这些图片却被屏蔽了。

还有就是抓包，不管是浏览器还是微信，与互联网通信绕不开网卡。或者这么说吧，上网的本质是与服务器上传下载数据，所谓互联网也就是一个范围为全球的局域网。那么其他计算机上的数据要想进入我们的计算机，就要通过网卡写入我们计算机上的缓存区，然后写进硬盘里或者用完之后被回收。而网卡驱动上是提供了对外调用的函数的，因此写个程序是可以实时抓取全部进入本机的数据。

然后我看了下，如何解析数据包，然后就没有然后了……

老老实实分析那些图片url被鼠绘藏哪了吧。

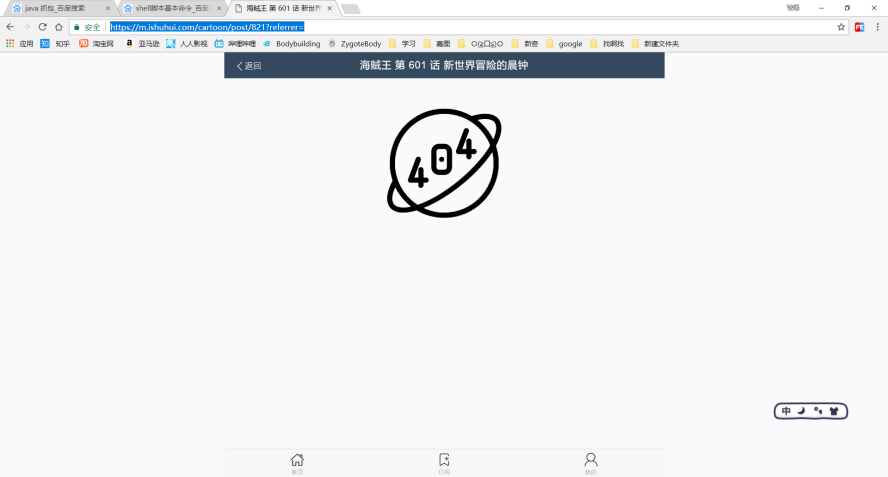
爬虫的套路其实就是分析网页源码，然后提取自己需要的url。微信公众号是H5，换言之，这里头的内容是可以在浏览器里打开的。鼠绘公众号里的海贼王漫画也能在浏览器中打开，但里头和图片相关的url被屏蔽了。

以海贼王第600话为例，从公众号里复制这一话的url只能得到这个http://m.ishuhui.com/cartoon/book/1，而这个仅代表海贼王这部漫画。

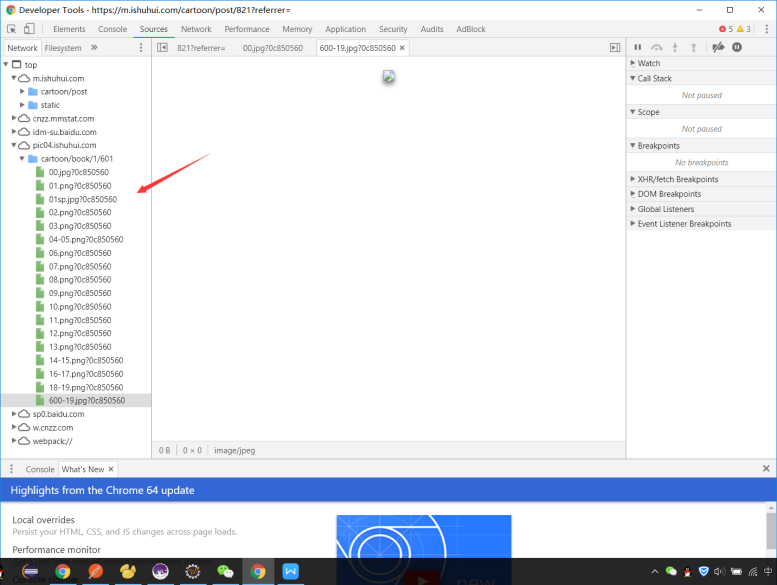
但是点分享给某人，嘿，这一话的url就跳出来了<https://m.ishuhui.com/cartoon/post/821>



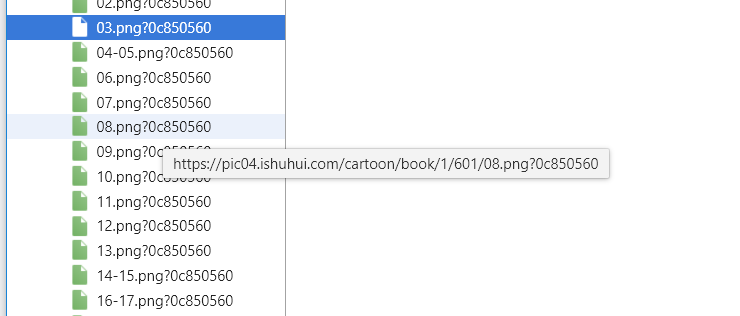
用浏览器访问这个连接的话，显示如下内容。



不过呼出控制台后却看到了有意思的东西



把鼠标移上去

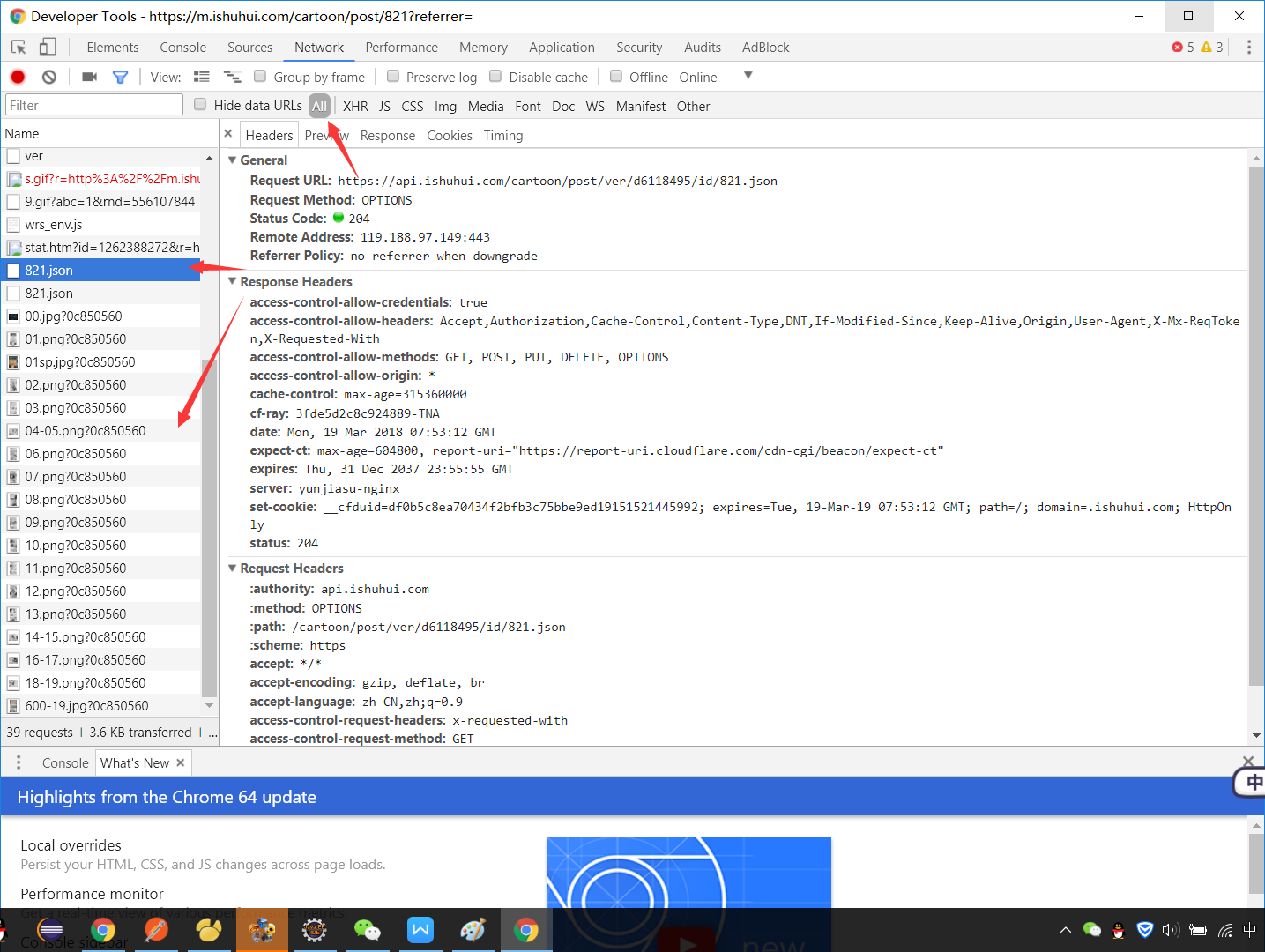


右键新标签打开



excetied !

他妈的，点开network后惊喜更多



821.json的内容

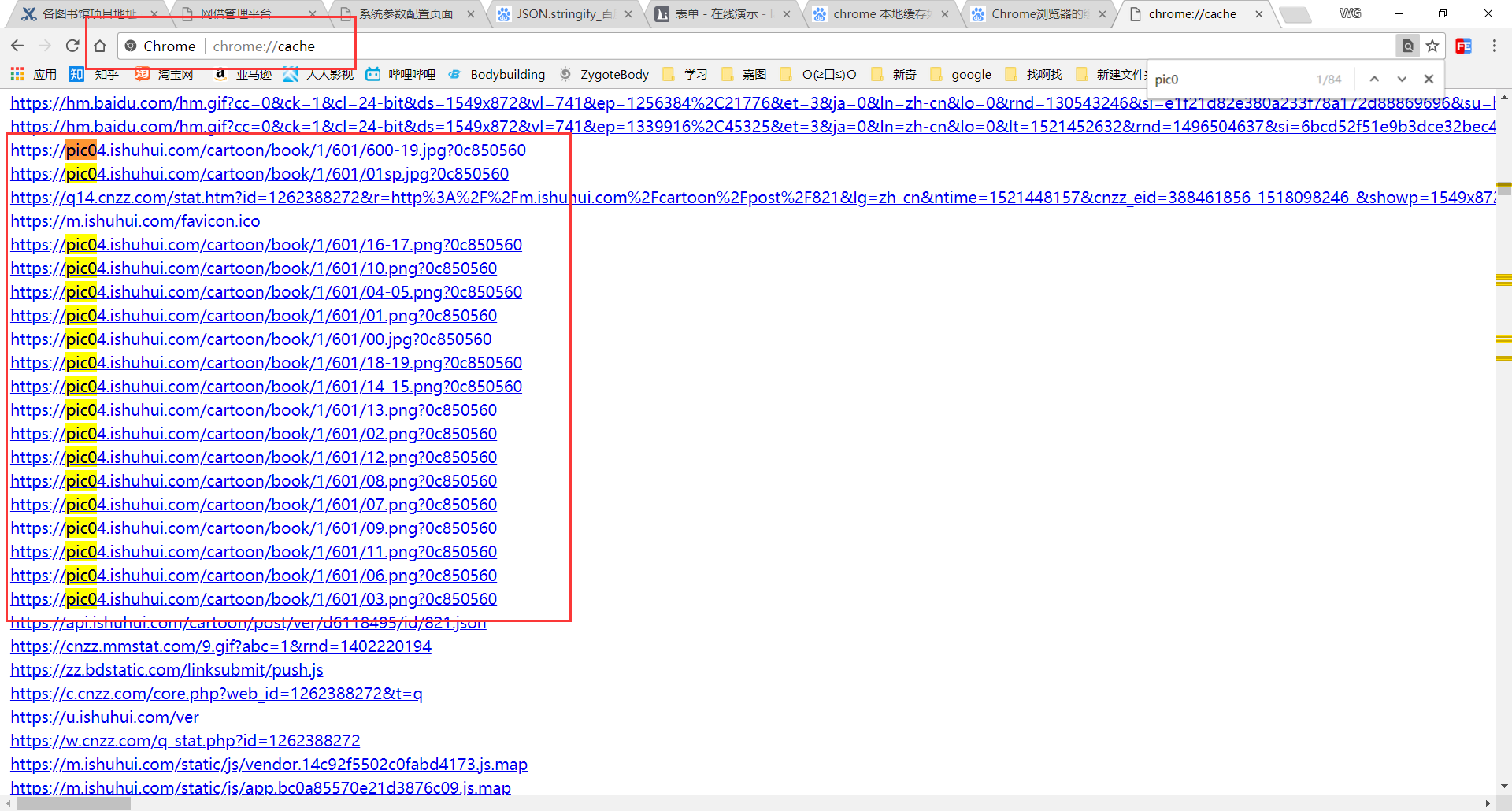


拷了一部分

content\_img:"{"00.jpg":"/upload/cartoon/book/1/601/00.jpg","01.png":"/upload/cartoon/book/1/601/01.png","01sp.jpg":"/upload/cartoon/book/1/601/01sp.jpg","02.png":"/upload/cartoon/book/1/601/02.png","03.png":"/upload/cartoon/book/1/601/03.png","04-05.png":"/upload/cartoon/book/1/601/04-05.png","06.png":"/upload/cartoon/book/1/601/06.png","07.png":"/upload/cartoon/book/1/601/07.png","08.png":"/upload/cartoon/book/1/601/08.png","09.png":"/upload/cartoon/book/1/601/09.png","10.png":"/upload/cartoon/book/1/601/10.png","11.png":"/upload/cartoon/book/1/601/11.png","12.png":"/upload/cartoon/book/1/601/12.png","13.png":"/upload/cartoon/book/1/601/13.png","14-15.png":"/upload/cartoon/book/1/601/14-15.png","16-17.png":"/upload/cartoon/book/1/601/16-17.png","18-19.png":"/upload/cartoon/book/1/601/18-19.png","600-19.jpg":"/upload/cartoon/book/1/601/600-19.jpg"}"

到这里的话，又出现了一种可取的方式，因为控制台里的那些图片，肯定是存在Chrome缓存里的。那么把这些东西都提取出来，也实现了获取图片url的目的。

（// 20180915，该功能仅限于chrome 66使用）



那末，如果我能把每一话的地址都获取出来的话（<https://m.ishuhui.com/cartoon/post/821>这种地址），然后用Chrome加载一遍，不管有没有图片，由于js的原因，这些图片url是不是全都进Chrome的缓存了呢？

to be continued

这些地址获取不出来啊，他妈的。还是只有通过分享才会跳出来，分享按钮的功能，我都想不通是js做的还是c#做的。

又只能找规律了。

600话 <https://m.ishuhui.com/cartoon/post/821>

650话 [https://m.ishuhui.com/cartoon/post/8](https://m.ishuhui.com/cartoon/post/821)70

700话 [https://m.ishuhui.com/cartoon/post/](https://m.ishuhui.com/cartoon/post/821)918

750话 [https://m.ishuhui.com/cartoon/post/](https://m.ishuhui.com/cartoon/post/821)968

800话 [https://m.ishuhui.com/cartoon/post/](https://m.ishuhui.com/cartoon/post/821)1018

800话往后的内容，有个叫海贼小站的网站可以直接保存，不用这么麻烦。

用selenium访问每一个url，之后再访问Chrome缓存页，鉴于Chrome缓存页只是一个HTML，所以倒是能很方便的右键保存。

原本以为这只是个体力活，但里头还藏着两个坑。

其一，Chrome的缓存是有上限的，我目测是360M左右，因此200个url一次跑完，最后也只缓存了大概50来话的内容。至少要分成4次，保险起见我分了5次。

其二，Chrome的窗口与窗口之间不共享缓存（不知道是不是叫窗口，就是已经开了一个Chrome的情况下，再开一个Chrome）。所以，selenuim跑完后，不可以调用它的close()方法。

WebDriver driver = *getWebDriver*();

// 1019

**for**(**int** i = 980; i < 1020; i++) {

String url = "https://m.ishuhui.com/cartoon/post/" + i;

driver.get(url);

Document s = Jsoup.*parse*(driver.getPageSource());

System.***out***.println(s.title());

// 这里很想吐槽，为什么selenuim的title方法不能直接把title输出出来，害的我还要在用Jsoup解析一遍，不 // 过在解析一遍倒是也有好处，不会因为访问过快被鼠绘封ip。

}

合并从Chrome缓存页那拿到的HTML，用jsoup获取<a>标签，适当过滤并排序。

BufferedWriter writer = **new** BufferedWriter(**new** FileWriter("D:\\Workspace\\test\\imgurl.txt"));

Document document = Jsoup.*parse*(**new** File("C:\\Users\\Administrator\\Desktop\\新建文本文档 (2).txt"), "UTF-8");

Elements elements = document.getElementsByTag("a");

List<String> urlList = Lists.*newArrayList*();

**for** (Element temp : elements) {

String text = temp.text();

**if** (StringUtils.*startsWith*(text, "https://pic")) {

urlList.add(text);

}

}

Collections.*sort*(urlList);

**for** (String text : urlList)

writer.write(text + "\r\n");

writer.close();

赞 ！



到这基本结束了，毕竟最重要的url已经到手。不过我想用每一话的标题为文件夹命名。标题在做最开始的苦力活时，我保存了一份。

// 获取每一话的名字

Map<Integer, String> map = Maps.*newHashMap*();

File file = **new** File("C:\\Users\\Administrator\\Desktop\\新建文本文档.txt");

BufferedReader reader = Files.*newReader*(file, Charset.*forName*("UTF-8"));

String temp = **null**;

**while** ((temp = reader.readLine()) != **null**) {

**if** (StringUtils.*isNotBlank*(temp)) {

**int** no = NumberUtils.*toInt*(temp.substring(0, 3));

**if** (no > 0) {

map.put(no, temp.substring(3));

}

}

}

// 开始保存

Map<Integer, List<String>> urlListMap = Maps.*newHashMap*();

// 引用重用，懒得起名字了

file = **new** File("D:\\Workspace\\test\\imgurl.txt");

reader = Files.*newReader*(file, Charset.*forName*("UTF-8"));

**while** ((temp = reader.readLine()) != **null**) {

**if** (StringUtils.*isNotBlank*(temp)) {

**int** no = NumberUtils.*toInt*(temp.substring(41, 44));

**if** (no == 0) {

**continue**;

}

**if** (urlListMap.get(no) != **null**) {

urlListMap.get(no).add(temp);

} **else** {

List<String> list = Lists.*newArrayList*();

list.add(temp);

urlListMap.put(no, list);

}

}

}

// 合并map，方便我重用ShuHuiUtil里的方法

Map<String, List<String>> finalMap = Maps.*newHashMap*();

**for** (Entry<Integer, List<String>> entry : urlListMap.entrySet()) {

**int** no = entry.getKey();

String name = map.get(no); // 取标题名称

**if** (StringUtils.*isBlank*(name)) {

**continue**;

}

name = "海賊王 第 " + no + " " + name;

finalMap.put(name, entry.getValue());

}

ShuHuiUtil.*saveToLocal*(finalMap);

中间缺了十几话，有一部分是鼠绘url造成的，另一部分可能是我过滤条件写的不好，下次有空补一下吧。

下一个目标是《监狱学园》

：)

// 尚缺 680、681

# 百度贴吧

## 黑咪，监狱学园

## 剑风传奇

写功能给自己用，第一个要弄明白的问题，是自己要追求过程，还是追求结果。不搞明白这个，遇到问题人容易慌乱。

像爬漫画这件事，遇到瓶颈了，就容易在“为了技术而技术”与“老子买个会员算了”的心态间徘徊，并渐渐放弃。

我纠结的地方，是按下启动键后我就什么都不要干，然后等着程序给我生成资源就好了。但世上哪有自动化程度那么高的程序，为人服务的东西终究还是要有人参与啊。

贴吧的资源，获取图片的url很简单，jsoup就搞定了。主要是话与话直接不连续，还没有规律，另外就是分页。

目前来说只能跑一段，然后停下来，我主动收集一波，改参数，然后继续跑。

分页的问题不知道怎么解决 :-(

话说回来这个类都快500行了啊。

https://github.com/J0-J0/demo/blob/master/src/main/java/com/jojo/crawler/MangaCrawler.java

# 飞丸网

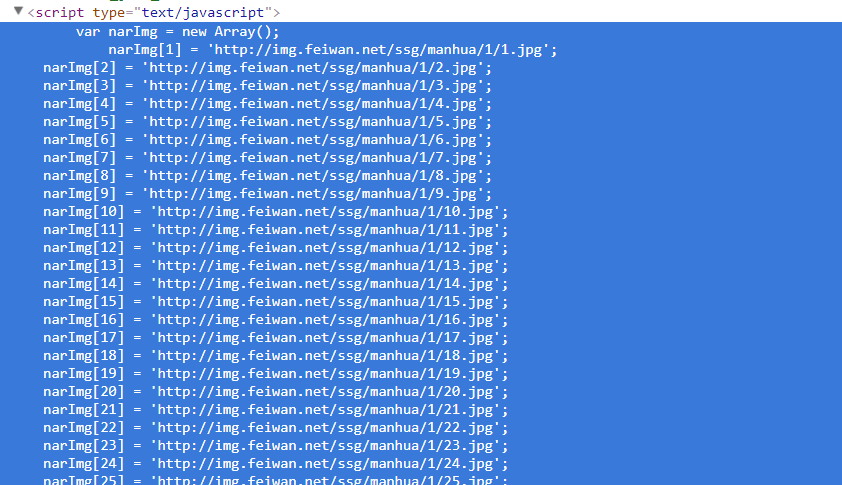
## 东京食尸鬼

这网站的爬取容易度和海贼小站有的一拼。

http://ssg.feiwan.net/manhua/1.html

首先，网站的管理者，给每一部漫画都搞了个独立域名。restful url 风格，最后一部分表示该漫画中的某一话。

其次，这家伙把某一话的所有图片url放在<script>块里。



说实在的也就是从jsoup变成正则表达式

# Wanted

一拳超人原作版

进击的巨人

亚人