花瓣网爬虫的分析和设计

更新时间：2019-06-24 17:57:04

> 困难只能吓倒懦夫懒汉，而胜利永远属于敢于等科学高峰的人。
>
> ——茅以升

如果你还没有完全理解本专栏前面章节的内容，甚至没有将源代码都跑一遍的话我建议你先不要急于进入本章节。本章的内容有点特殊，可以说是将本专栏内的各种内容都融合其中。

本章将会以采集花瓣网的优质图片作为课题展开。花瓣网是早年典型的C2C(Copy To China)类型的成功代表之一，她的原型来自于Pinterest是一种瀑布流式图片墙，用户可以在浏览各种网站的时候通过浏览器的插件将喜欢的图片收集到花瓣网的图集中，花瓣网就相当于一个由用户提供内容(UGC)并进行个人分类的图片索引平台。用户在花瓣网会检测当前浏览器是否已被滑动到底部，当检测成功就会自动加载下一批的图片让用户有无需一页一页地翻着来看图，业界称这种加载方式为"无限加载"，因为图片的显示是随机的，所以往下拉总是会拉出新的一批图片。

以花瓣网为课题是因为对于爬虫项目来说它具有一些典型的特点：

1. 通过跟踪登录用户来实现反爬判定
2. 它是一个重度使用Javascript进行局部加载的网站
3. 它是以图片为主的网站

跟踪登录用户实现反爬判定

这是一种非常绝妙的反爬与保护数据的办法，如果你用匿名用户访问花瓣网当你没翻几页以后就会弹出一个登录对话框来阻止你进一步的操作，如果是人在浏览通常都会注册一个用户然后继续浏览。但如果对于是爬虫而言就光这一招就能破掉我们在前几章中学到的所有招数。

因为你的爬虫再也不是隐藏在阴暗中的路人甲，你的访问行为必须与一个登录账号绑定。而登录账号又必须通过手机验证，那么是不是就是说即使你能用爬虫通过代码的方式实现了登录并开始爬取工作，一但被网站认定你是爬虫那么你的账号可能就会被永封，那你只能换个手机来重新注册一个新的用户账号来继续你的爬工作了。这种高成本就很好的限制住了大规模的爬网行为，至少如果要大规模地爬网你就得先有一大堆用不同手机号验证过的账号才能展开，不是吗？

其次，网站还得知了你的手机号，全国手机大多实名制一但你爬了过多不应该爬的数据网站会很快地定位到你并让你承担法律负责，这一点确实值得重视，否则就真是得不尝失了。

重度使用 Javascript 进行局部加载

前面章节所介绍的爬网技术是对已经渲染好的网页内容进行提取，但如果网页是需要执行一些 Javascript 脚本才会生成某些元素的话，那之前所学也同样泡汤。随着Angular、React和Vue这些重前端技术的普及这类重度应用前端的网页也越来越多，如果要爬取这些页面我们还会在蜘蛛上加一把功夫让蜘蛛将获取到的网页能执行其上的脚本显现其真容后才可能爬到想要的数据 。

以图片为主的网站

对于 Scrapy 爬虫技术还有一个重要的内容还没有介绍，那就是图片爬虫，如何将图片作为最终爬取结果存下来是 Scrapy 的一个重要的学习内容。

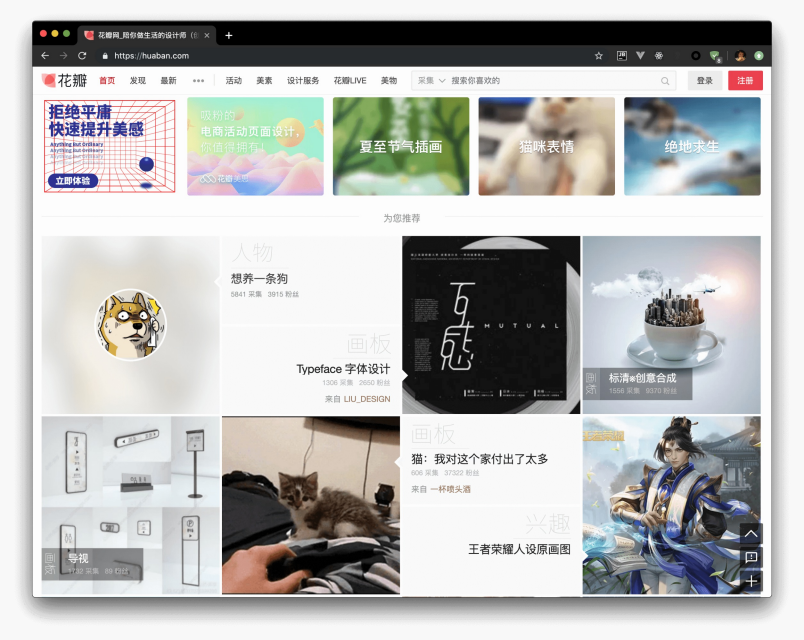
既然花瓣网具有这么多的限制那么是不是就意味着我们就束手无策了呢？答案当然是否定的，能反爬就能反反爬，能反反爬就会有更高明的反爬机制出现，爬网与反爬网技术总是互相牵制又互相促进的。

花瓣网、淘宝网、京东、微博甚至是微信都是采用限制匿名用户的方式来检测蜘蛛的，但不限制"人"，那就是说只要我们所开发的蜘蛛的行为能更像人，这种反爬机制也可以被攻破，只是我们所需要付出的代价就更高一些，这个代价就是时间。

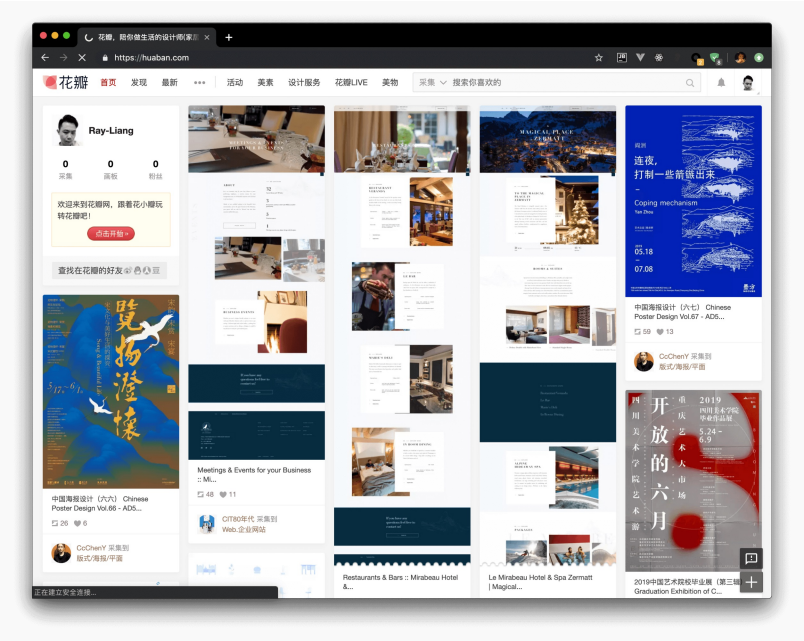
在本章介绍的就是一种**高匿的拟人爬虫**，这种爬虫的特点是：

- 完全模仿人的浏览行为，让反爬机制难以分辨
- 速度慢，只能以人的速度来浏览速度自然快不了，想要快速就多注册几个账号

分析花瓣网



花瓣网登录前与登录后能看到的内容是完全不同的：



所以在开始之前你需要先注册一个账号并通过手机验证后就能看以上的画面了。

这个页面是根据你注册时所选定的兴趣由网站主动过滤出来的内容，这个页面可以作为"种子"成为爬虫的入口。

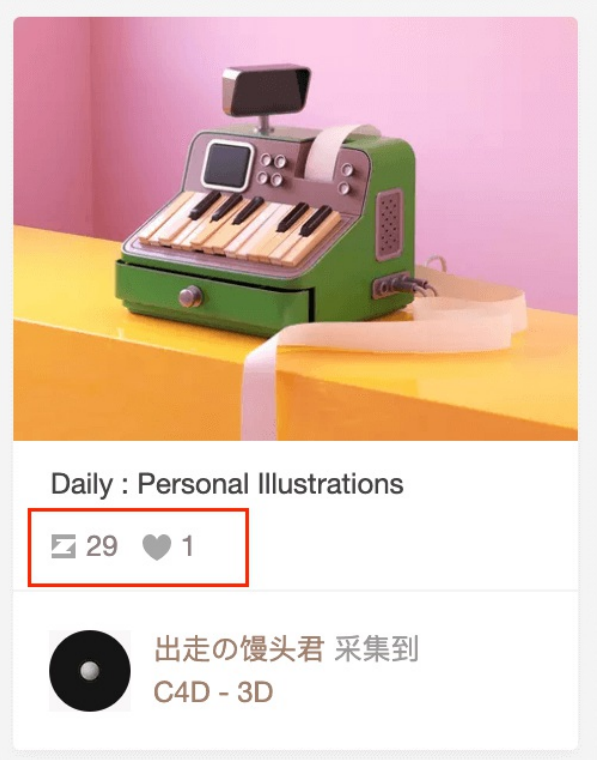
也可以用花瓣的查询页址：

```
https://huaban.com/search/?q=App&type=pins
```

参数`q`后面附带的就是查询关键字。

分析数据结构

花瓣的数据结构相对简单，从花瓣中我们爬取的重点目标是图片、描述与原网站链接三样就足够了。另外，在这么多的图里面我们可以加个条件只爬取`转采>250`并且`喜欢>25`属性优质图片，这样可以保证不去爬取那些垃圾图片以免浪费空间。



这样我们可以很快地得出以下的数据结构

字段	说明
`imgUrl`	图片地址
`link`	原网站链接
`desc`	描述

我将上图上一个Pin单元内的HTML进行截取后，将对爬虫分析有用的部分保留下来后得到以下的HTML代码，以便于分析讲解：

```
<div data-id="2520376569"
     data-seq="2520376569"
     data-source="xwallet.pundix.com"
     data-created-at="1561088939">
    <a class="img long x layer-view loaded">
        <img src="//hbimg.huabanimg.com/07ca5a928446d37c043f284f0c6c1b24263647872cd69b-hSntEt_/fw/480"
             width="236"
             height="814"
             alt="CIT80年代采集到Web.企业网站" >
    </a>
    <p data-raw="Pundi X – Making cryptocurrency accessible to everyone"
       data-formatted="Pundi X – Making cryptocurrency accessible to everyone"
       class="description">Pundi X – Making
        cryptocurrency accessible to<a class="show-more">...</a></p>
    <p class="stats less">
      <span title="转采" class="repin"><i></i>26</span>
      <span title="喜欢" class="like"><i></i>1</span>
    </p>
</div>
```

这部分内容只要用一个`#waterfall>div`的CSS选择器就能选出所有当前展示的图片项。

其中`desc`比较容易选出直接用`#waterfall>div .description`就可以了。而原网站链接在这个元素结构内并没有找到，如果你去点击浏览器中的图片就会发现它会弹出一个层然后将原图片放大显示，再点击后会进入https://huaban.com/go/?pin_id=<拼版ID>再进行原网站转跳，也就是说我们并不需要去记录原网站地址，而只要将顶层`<div>`元素的`data-id`属性记下来，有需要时直接匹配花瓣的转跳页面来进行跳转就可以了。

最后，也是我们要爬取的重点在`<img>`元素的`src`，你会发这里的地址有点怪，它是长这样的：

```
//hbimg.huabanimg.com/07ca5a928446d37c043f284f0c6c1b24263647872cd69b-hSntEt_/fw/480
```

如果复制出来直接贴到浏览器的地址栏会马上转跳到一你文件夹的一个完全不存在的地址，浏览器会将上述地址解释为：

```
file:///hbimg.huabanimg.com/07ca5a928446d37c043f284f0c6c1b24263647872cd69b-hSntEt_/fw/480
```

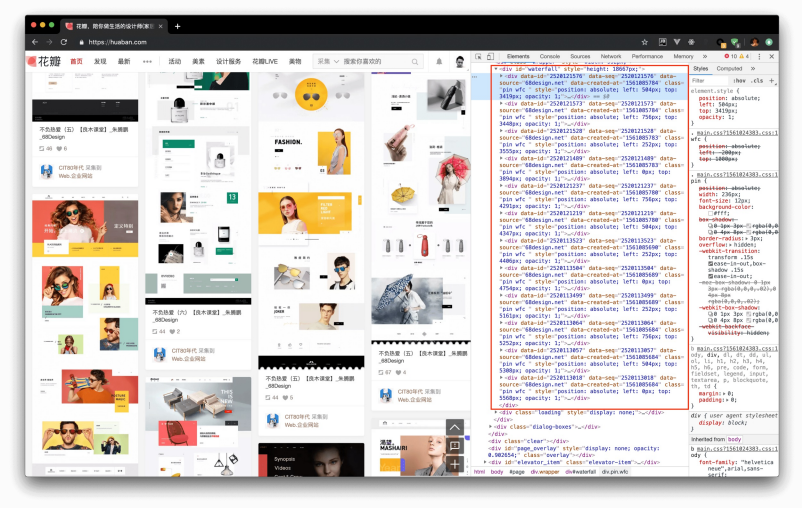
解决办法就是将`files://`协议换成`https://`就可以了。

但是这个图并不是原图，而是一张缩略图！这也不是问题，这个URL本身就暴露了原图的地址，只要将`/fw/480`部份删除掉就可以得到原图地址了，也就是：

```
https://hbimg.huabanimg.com/07ca5a928446d37c043f284f0c6c1b24263647872cd69b-hSntEt_
```

加载方法

既然花瓣网采用的是"无限加载"模式，那么每次会加载多少个元素呢？不断加载新元素不会让浏览器的内存消耗光吗？带着这两个问题我们可以打开Chrome的开发者模型一边滚动网页触发页面加载一边观察网页元素的变化：



只要多加载几次你就会发现，你所见到的网页元素的总数是维持在一定数量之内，当新加载的元素达到一定数量时，排在最前面的元素就会被删除。这个就是让浏览器不断加载新图片而又不会因为元素过多消耗内存过大而导致浏览器崩溃的秘密所在了！

我们得到一个规则：加载一定数量的图片后就有相应数量的元素被删除。

一定要记住这一点，这样我们在提取数据时就要注意了。

花瓣爬虫的设计思路

无论开发什么样的规模的项目我都会遵循着分析、设计、测试最后编码这样的一个流程来执行。可能有些开发人员认为前三部都是没有必要的，想到哪写到哪最后运行一下就能调通了。没有经过充分思考就动手是写不出高质量代码的，分析、设计与测试是能让我们在编码之前发现很思路上的错误甚至是可以让我们绕开很多被隐藏起来的大坑的。

虽然我们已经基本上学会了如何来开发一个爬虫项目了，但是我们所面对的网站是多种多样的，每个网站都会有其自身的特点，又或者是开发人员针对爬虫故意埋下的坑。所以在动手之前应该充分地分析，然后进行思路的整理得到一份最简单的设计或者是实现步骤，对于自己不确定的技术内容通过编写测试来验证，这反倒会让你的开发速度倍增。

那接下来我们就来设想一下这个花瓣网的蜘蛛应该如何来开发。

首先，这是一个"高匿、慢速的拟人爬虫"这是对爬虫性质的介定，为了达到这个目标爬虫就不能采用前章书中提到的随机代理中间件与随机UA中间件了，因为你正常地用浏览器上网的时候是不会打开一个网页就换一个浏览器，又或者上着上着网IP地址就会自己变了的，开了这两个中间件只会暴露了你是爬虫的真实身份。

其次，你得先注册的一个用户，然后就是用这个用户身份进行登录。而且是用爬虫来模拟你登录。这是进入花瓣网这个大门的第一步，不能登录就更别说去提取数据了。

然后，上文在对网页进行分析的时候基本上就对`Item`类的结构分析完整了，直接编写`Item`类是完全没有问题的，只要能正确分析出通过Javascript生成的元素就可以保证能提取网页内容了。经过之前的观察我们知道数据的加载是通过页面滑动到底部被触发的，那我们可不可以直接生成这些URL请求而不是模拟用户滑动的操作来触发加载呢？

答案是肯定的！根据我分析Chrome的网络请求我找到了这个分页数据的请求地址：

```
https://huaban.com/search/?q=app&type=pins&page=20&per_page=20
```

这个地址一共有三个可变参数：

- `q` ——查询字符串，这个可以随意你想看什么就输入什么关键字进行匹配
- `page`——这是起始页的页码，意思就是从哪个位置开始加载
- `pre_page`——每次加载多少个图片

最后，只要将图片地址下载到本地并生成一个新的文件名，将`Item`的数据写到一个MongoDB里面那么这个爬虫就可以完成了。

小结

我在前几章的内容中都一直渗透着如何先设计再动手实践，谋定而后动这是我们作为开发人员在开发项目时很必要的一种做法。否则只会是以修修补补式地开发，那最终只会做出一个千疮百孔的连自己都不想再碰的怪物。

23 分布式网络爬虫

打开花瓣网的大门

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论