6.2 Ajax 分析方法

还是以上文中的微博为例，我们已经知道了拖动刷新的内容是由 Ajax 加载的，而且页面的 URL 没有变化，那么我们应该到哪去查看这些 Ajax 请求呢？

1. 查看请求

在这里我们还是需要借助于浏览器的开发者工具，我们以 Chrome 浏览器为例来看一下怎样操作。

首先用 Chrome 浏览器打开微博的链接：https://m.weibo.cn/u/2145291155，随后在页面中点击鼠标右键，会出现一个检查的选项，点击它便会弹出开发者工具，如图 6-2 所示：

图 6-2 开发者工具

那么在 Elements 选项卡便会观察到网页的源代码，右侧便是节点的样式。

不过这不是我们想要寻找的内容，我们切换到 Network 选项卡，随后重新刷新页面，可以发现在这里出现了非常多的条目，如图 6-3 所示：

图 6-3 Network 面板结果

前文我们也提到过，这里其实就是在页面加载过程中浏览器与服务器之间发送 Request 和接收 Response 的所有记录。

Ajax 其实有其特殊的请求类型，它叫做 xhr，在上图中我们可以发现一个名称为 getIndex 开头的请求，其 Type 为 xhr，这就是一个 Ajax 请求，我们鼠标点击这个请求，可以查看这个请求的详细信息，如图 6-4 所示：

图 6-4 详细信息

我们在右侧可以观察到其 Request Headers、URL 和 Response Headers 等信息，如图 6-5 所示：

图 6-5 详细信息

其中 Request Headers 中有一个信息为 X-Requested-With:XMLHttpRequest，这就标记了此请求是 Ajax 请求。

随后我们点击一下 Preview，即可看到响应的内容，响应内容是 Json 格式，在这里 Chrome 为我们自动做了解析，我们可以点击箭头来展开和收起相应内容，如图 6-6 所示：

图 6-6 Json 结果

观察可以发现，这里的返回结果是马云的个人信息，如昵称、简介、头像等等，这也就是用来渲染个人主页所使用的数据，JavaScript 接收到这些数据之后，再执行相应的渲染方法，整个页面就被渲染出来了。

另外也可以切换到 Response 选项卡，可以观察到真实的返回数据，如图 6-7 所示：

图 6-7 Response 内容

接下来我们切回到第一个请求，观察一下它的 Response 是什么，如图 6-8 所示：

图 6-8 Response 内容

这是最原始的链接 https://m.weibo.cn/u/2145291155 返回的结果，其代码只有五十行，结构也非常简单，只是执行了一些 JavaScript。

所以说，我们所看到的微博页面的真实数据并不是最原始的页面返回的，而是后来执行 JavaScript 后再次向后台发送了 Ajax 请求，拿到数据后再进一步渲染出来的。

2. 过滤请求

接下来我们再利用 Chrome 开发者工具的筛选功能筛选出所有的 Ajax 请求，在请求的上方有一层筛选栏，我们可以点击 XHR，这样在下方显示的所有请求便都是 Ajax 请求了，如图 6-9 所示：

图 6-9 Ajax 请求

再接下来我们我们不断滑动页面，可以看到在页面底部有一条条新的微博被刷出，而开发者工具下方也一个个地出现 Ajax 请求，这样我们就可以捕获到所有的 Ajax 请求了。

随意点开一个条目都可以清楚地看到其 Request URL、Request Headers、Response Headers、Response Body 等内容，想要模拟请求和提取就非常简单了。

如图所示内容便是马云某一页微博的列表信息，如图 6-10 所示：

图 6-10 微博列表信息

3. 结语

到现在为止我们已经可以分析出来 Ajax 请求的一些详细信息了，接下来我们只需要用程序来模拟这些 Ajax 请求就可以轻松提取我们所需要的信息了。

所以在下一节我们来用 Python 实现 Ajax 请求的模拟，从而实现数据的抓取。

6.3 Ajax 结果提取

仍然是拿微博为例，我们接下来用 Python 来模拟这些 Ajax 请求，把马云发过的微博爬取下来。

1. 分析请求

我们打开 Ajax 的 XHR 过滤器，然后一直滑动页面加载新的微博内容，可以看到会不断有 Ajax 请求发出。

我们选定其中一个请求来分析一下它的参数信息，点击该请求进入详情页面，如图 6-11 所示：

图 6-11 详情页面

可以发现这是一个 GET 类型的请求，请求链接为：https://m.weibo.cn/api/container/getIndex?type=uid&value=2145291155&containerid=1076032145291155&page=2，请求的参数有四个：type、value、containerid、page。

随后我们再看一下其他的请求，观察一下这些请求，发现它们的 type、value、containerid 始终如一。type 始终为 uid，value 的值就是页面的链接中的数字，其实这就是用户的 id，另外

还有一个 containerid，经过观察发现它就是 107603 然后加上用户 id。所以改变的值就是 page，很明显这个参数就是用来控制分页的，page=1 代表第一页，page=2 代表第二页，以此类推。

以上的推断过程可以实际观察参数的规律即可得出。

2. 分析响应

随后我们观察一下这个请求的响应内容，如图 6-12 所示：

图 6-12 响应内容

它是一个 Json 格式，浏览器开发者工具自动为做了解析方便我们查看，可以看到最关键的两部分信息就是 cardlistInfo 和 cards，将二者展开，cardlistInfo 里面包含了一个比较重要的信息就是 total，经过观察后发现其实它是微博的总数量，我们可以根据这个数字来估算出分页的数目。

cards 则是一个列表，它包含了 10 个元素，我们展开其中一个来看一下，如图 6-13 所示：

图 6-13 列表内容

发现它又有一个比较重要的字段，叫做 mblog，继续把它展开，发现它包含的正是微博的一些信息。比如 attitudes_count 赞数目、comments_count 评论数目、reposts_count 转发数目、created_at 发布时间、text 微博正文等等，得来全不费功夫，而且都是一些格式化的内容，所以我们提取信息也更加方便了。

这样我们可以请求一个接口就得到 10 条微博，而且请求的时候只需要改变 page 参数即可，目前总共 138 条微博那么只需要请求 14 次即可，也就是 page 最大可以设置为 14。

这样我们只需要简单做一个循环就可以获取到所有的微博了。

3. 实战演练

在这里我们就开始用程序来模拟这些 Ajax 请求，将马云的所有微博全部爬取下来。

首先我们定义一个方法，来获取每次请求的结果，在请求时 page 是一个可变参数，所以我们将它作为方法的参数传递进来，代码如下：

```
from urllib.parse import urlencodeimport requests

base_url = 'https://m.weibo.cn/api/container/getIndex?'



headers = {

    'Host': 'm.weibo.cn',

    'Referer': 'https://m.weibo.cn/u/2145291155',

    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36',

    'X-Requested-With': 'XMLHttpRequest',

}

def get_page(page):

    params = {

        'type': 'uid',

        'value': '2145291155',

        'containerid': '1076032145291155',

        'page': page

    }

    url = base_url + urlencode(params)
```

```
    try:
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            return response.json()
    except requests.ConnectionError as e:
        print('Error', e.args)
```

首先在这里我们定义了一个 base_url 来表示请求的 URL 的前半部分，接下来构造了一个参数字典，其中 type、value、containerid 是固定的参数，只有 page 是可变参数，接下来我们调用了 urlencode() 方法将参数转化为 URL 的 GET 请求参数，即类似于 type=uid&value=2145291155&containerid=1076032145291155&page=2 这样的形式，随后 base_url 与参数拼合形成一个新的 URL，然后我们用 Requests 请求这个链接，加入 headers 参数，然后判断响应的状态码，如果是 200，则直接调用 json() 方法将内容解析为 Json 返回，否则不返回任何信息，如果出现异常则捕获并输出其异常信息。

随后我们需要定义一个解析方法，用来从结果中提取我们想要的信息，比如我们这次想保存微博的 id、正文、赞数、评论数、转发数这几个内容，那可以先将 cards 遍历，然后获取 mblog 中的各个信息，赋值为一个新的字典返回即可。

```
from pyquery import PyQuery as pq

def parse_page(json):
    if json:
        items = json.get('cards')
        for item in items:
            item = item.get('mblog')
            weibo = {}
```

```
            weibo['id'] = item.get('id')
            weibo['text'] = pq(item.get('text')).text()
            weibo['attitudes'] = item.get('attitudes_count')
            weibo['comments'] = item.get('comments_count')
            weibo['reposts'] = item.get('reposts_count')
            yield weibo
```

在这里我们借助于 *PyQuery* 将正文中的 *HTML* 标签去除掉。

最后我们遍历一下 *page*，一共 *14* 页，将提取到的结果打印输出即可。

```
if __name__ == '__main__':
    for page in range(1, 15):
        json = get_page(page)
        results = parse_page(json)
        for result in results:
            print(result)
```

另外我们还可以加一个方法将结果保存到 *MongoDB* 数据库。

```
from pymongo import MongoClient


client = MongoClient()
db = client['weibo']
collection = db['weibo']
def save_to_mongo(result):
    if collection.insert(result):
```

```
        print('Saved to Mongo')
```

最后整理一下，最后的程序如下：

```
import requestsfrom urllib.parse import urlencodefrom pyquery import PyQuery as pqfrom pymongo import MongoClient



base_url = 'https://m.weibo.cn/api/container/getIndex?'

headers = {

    'Host': 'm.weibo.cn',

    'Referer': 'https://m.weibo.cn/u/2145291155',

    'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_12_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.36',

    'X-Requested-With': 'XMLHttpRequest',

}

client = MongoClient()

db = client['weibo']

collection = db['weibo']

max_page = 14



def get_page(page):

    params = {

        'type': 'uid',

        'value': '2145291155',

        'containerid': '1076032145291155',

        'page': page

    }
```

```
    url = base_url + urlencode(params)
    try:
        response = requests.get(url, headers=headers)
        if response.status_code == 200:
            return response.json()
    except requests.ConnectionError as e:
        print('Error', e.args)


def parse_page(json):
    if json:
        items = json.get('cards')
        for item in items:
            item = item.get('mblog')
            weibo = {}
            weibo['id'] = item.get('id')
            weibo['text'] = pq(item.get('text')).text()
            weibo['attitudes'] = item.get('attitudes_count')
            weibo['comments'] = item.get('comments_count')
            weibo['reposts'] = item.get('reposts_count')
            yield weibo


def save_to_mongo(result):
    if collection.insert(result):
```

```
        print('Saved to Mongo')



if __name__ == '__main__':
    for page in range(1, max_page + 1):
        json = get_page(page)
        results = parse_page(json)
        for result in results:
            print(result)
            save_to_mongo(result)
```

运行程序后样例输出结果如下：

```
{'id': '3938863363932540', 'text': '我们也许不能解决所有的问题，但我们可以尽自己的力量去解决一些问题。移动互联网不能只是让留守孩子多了一个隔空说话的手机，移动互联网是要让父母和孩子一直在一起。过年了，回家吧…… 农村淘宝 2016 团圆贺岁片《福与李》 ', 'attitudes': 21785, 'comments': 40232, 'reposts': 2561}

Saved to Mongo

{'id': '3932968885900763', 'text': '跟来自陕甘宁云贵川六省的 100 位优秀乡村教师共度了难忘的两天，接下来我又得出远门了。。。为了 4000 万就读于乡村学校的孩子，所以有了这么一群坚毅可爱的老师，有了这么多关注乡村教育的各界人士，这两天感动、欣喜、振奋！我们在各自的领域里，一直坚持和努力吧！', 'attitudes': 32057, 'comments': 7916, 'reposts': 2332}

Saved to Mongo
```

查看一下 MongoDB，相应的数据也被保存到 MongoDB，如图 6-14 所示：

图 6-14 保存结果

4. 本节代码

本节代码地址：https://github.com/Python3WebSpider/WeiboList。

5. 结语

本节实例的目的是为了演示 Ajax 的模拟请求过程，爬取的结果不是重点，该程序仍有很多可以完善的地方，如页码的动态计算、微博查看全文等，如感兴趣可以尝试一下。

通过这个实例我们主要是为了学会怎样去分析 Ajax 请求，怎样用程序来模拟抓取 Ajax 请求，了解了相关抓取原理之后，下一节的 Ajax 实战演练会更加得心应手。