**2.6.1 HTML页面解析:**

HTML的标记总是封装在由小于一号(<)和大于号(>)构成的一对尖括号

之中

1、单标记

某些标记称为“单标记”，因为它只需单独使用就能完整地表达意思，这类

标记的语法是:

<标记>

最常用的单标记是<P>，它表示一个段落(Paragraph的结束，并在段落

后面加一空行。

2、双标记

另一类标记称为“双标记”，它由“始标记”和“尾标记”两部分构成，必须成

对使用，其中始标记告诉Web浏览器从此处开始执行该标记所表示的功能，

而尾标记告诉Web浏览器在这里结束该功能。始标记前加一个斜杠(/)即成

为尾标记。这类标记的语法是:

<标记>内容</标记>

其中“内容”部分就是要被这对标记施加作用的部分。例如你想突出对某段

文字的显示，就将此段文字放在一对<EM> </EM>标记中:

<EM>text to emphasize</EM>

3、标记属性

许多单标记和双标记的始标记内可以包含一些属性，其语法是:

<标记属性1属性2属性3 ... >

各属性之间无先后次序，属性也可省略(即取默认值)，例如单标记<HR>

如下图微博的HTML页面:



4.页面链接的提取

页面链接的提取，只有当下载的文档是txt/html格式的页面时在有必要分

析。页面的类型可由该网页头信息分析得出，有些站点返回的应答信息格式不

完整，此时必须通过分析页面URL中的文件扩展名来判别类型。遇到带连接

的标志如<A> , <AREA> , <FRAME>等，就从标记结构的属性中找出目标

URL，并从成对的该标记之间提取正文作为该连接的说明文字(扩展原数

据)。这两个数据就代表了该链接。

一个页面连接提取的过程总结如下:

1判断文件的类型是否是txt/html如果不是则跳过，如果是继续分析。

2读取文件利用正则表达式匹配的方法，寻找标签<a href= >, <area href=>，<base href=>，<frame src=>，<img src=>，<body background=>，<appletcode=>等，记录下该URL 。

3将记录下来的URL按照预定的格式统一完整。以备进行下一步的URL分析。

**2.6.2正则表达式简介**

HTTP协议本质上就是传输HTML所以在写爬虫的时候会遇到许多HTML页面，我们需要从中找出我们需要的内容，如何检索出我们需要的内容是一个关键，在这里可能就需要用到正则表达式。通过正则表达式我们可以匹配出我们需要的内容，接下来就是对正则表达式在爬虫中的应用的介绍。

正则表达式是用来匹配字符串非常强大的工具，在其他编程语言中同样有正则表达式的概念，Python同样不例外，利用了正则表达式，我们想要从返回的页面内容提取出我们想要的内容就易如反掌了。

比如我们想抓取所有图片的 URL 链接，非常直接的做法就是用 findAll("img") 抓取所图片，对吗？但是，有个问题。除了那些明显“多余的”图片（比如， LOGO） 之外，式的网站里都有一些隐藏图片， 用于网页布局留白和元素对齐的空白图片，以及一些不易察觉到的图片标签。总之，你不能仅用商品图片来统计网页上所有的图片。而且网页的布局也可能会变化， 或者，因为某些原因，我们不想通过图片在网页中的位来查找标签。 那么当你想抓取随机分布在网站里的某个元素或数据时，就会出现问题。如，一些网页的最上面可能有一张商品图片，但是在另一些网页上没有。解决这类问题的办法， 就是直接定位那些标签来查找信息

**2.6.3正则表达式相关注解**

(1)数量词的贪婪模式与非贪婪模式:正则表达式通常用于在文本中查找匹配的字符串。Python里数量词默认是贪婪的（在少数语言里也可能是默认非贪婪），总是尝试匹配尽可能多的字符；非贪婪的则相反，总是尝试匹配尽可能少的字符。例如：正则表达式”ab\*”如果用于查”abbbc”，将找到”abbb”。而如果使用非贪婪的数量词”ab\*?”，将找到”a”。

注：我们一般使用非贪婪模式来提取。

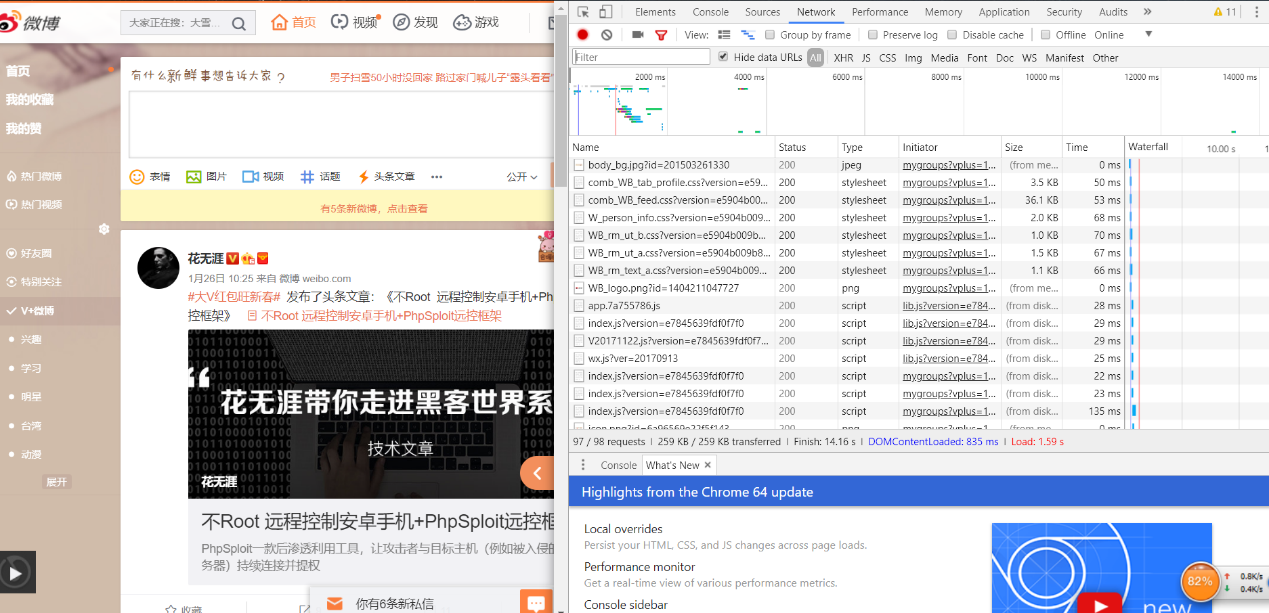
（2）反斜杠问题:与大多数编程语言相同，正则表达式里使用”\”作为转义字符，这就可能造成反斜杠困扰。假如你需要匹配文本中的字符”\”，那么使用编程语言表示的正则表达式里将需要4个反斜杠”\\\\”：前两个和后两个分别用于在编程语言里转义成反斜杠，转换成两个反斜杠后再在正则表达式里转义成一个反斜杠。Python里的原生字符串很好地解决了这个问题，这个例子中的正则表达式可以使用r”\\”表示。同样，匹配一个数字的”\\d”可以写成r”\d”。有了原生字符串，妈妈也不用担心是不是漏写了反斜杠，写出来的表达式也更直观勒。这是Python的re库正则表达式的一些语法:



**2.6.4Chrome编写正则表达式:**



用Chrome抓包:



由于版面原因这里只写出爬取信息的代码:

**def** parse\_tweets(self, response):  
 *""" 抓取微博数据 """* selector = Selector(response)  
 ID = re.findall(**'(\d+)/profile'**, response.url)[0]  
 divs = selector.xpath(**'body/div[@class="c" and @id]'**)  
 **for** div **in** divs:  
 **try**:  
 tweetsItems = TweetsItem()  
 id = div.xpath(**'@id'**).extract\_first() *# 微博ID* content = div.xpath(**'div/span[@class="ctt"]//text()'**).extract() *# 微博内容* cooridinates = div.xpath(**'div/a/@href'**).extract() *# 定位坐标* like = re.findall(**'赞\[(\d+)\]'**.decode(**'utf8'**), div.extract()) *# 点赞数* transfer = re.findall(**'转发\[(\d+)\]'**.decode(**'utf8'**), div.extract()) *# 转载数* comment = re.findall(**'评论\[(\d+)\]'**.decode(**'utf8'**), div.extract()) *# 评论数* others = div.xpath(**'div/span[@class="ct"]/text()'**).extract() *# 求时间和使用工具（手机或平台）* tweetsItems[**"\_id"**] = ID + **"-"** + id  
 tweetsItems[**"ID"**] = ID  
 **if** content:  
 tweetsItems[**"Content"**] = **" "**.join(content).strip(**'[位置]'**.decode(**'utf8'**)) *# 去掉最后的"[位置]"* **if** cooridinates:  
 cooridinates = re.findall(**'center=([\d.,]+)'**, cooridinates[0])  
 **if** cooridinates:  
 tweetsItems[**"Co\_oridinates"**] = cooridinates[0]  
 **if** like:  
 tweetsItems[**"Like"**] = int(like[0])  
 **if** transfer:  
 tweetsItems[**"Transfer"**] = int(transfer[0])  
 **if** comment:  
 tweetsItems[**"Comment"**] = int(comment[0])  
 **if** others:  
 others = others[0].split(**'来自'**.decode(**'utf8'**))  
 tweetsItems[**"PubTime"**] = others[0].replace(**u"\xa0"**, **""**)  
 **if** len(others) == 2:  
 tweetsItems[**"Tools"**] = others[1].replace(**u"\xa0"**, **""**)  
 **yield** tweetsItems  
 **except** Exception, e:  
 **pass** url\_next = selector.xpath(**'body/div[@class="pa" and @id="pagelist"]/form/div/a[text()="下页"]/@href'**.decode(**'utf8'**)).extract()  
 **if** url\_next:  
 **yield** Request(url=self.host + url\_next[0], callback=self.parse\_tweets, dont\_filter=True)