Python 网页抓取基础

石今

根据网站的地址规律，生成一组地址，从网页中获得想要信息，以最简单的ftp为例：

ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/ASN_BINARY/Mammalia/

/gene/DATA/ASN_BINARY/Mammalia/ 的索引

名称	大小	修改日期
[上级目录]		
All_Mammalia.ags.gz	1.7 GB	2017/2/21 上午2:31:00
Bos_taurus.ags.gz	42.8 MB	2017/2/21 上午2:30:00
Canis_familiaris.ags.gz	20.5 MB	2017/2/21 上午2:30:00
Homo_sapiens.ags.gz	198 MB	2017/2/21 上午2:30:00
Mus_musculus.ags.gz	109 MB	2017/2/21 上午2:30:00
Pan_troglodytes.ags.gz	32.6 MB	2017/2/21 上午2:30:00
Rattus_norvegicus.ags.gz	60.8 MB	2017/2/21 上午2:30:00
Sus_scrofa.ags.gz	26.2 MB	2017/2/21 上午2:31:00

上面的每一项，下载地址为ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/ASN_BINARY/Mammalia/<压缩包名称>
如果需要批量下载，则需要获得页面中压缩包名称的集合

为获取名称列表，最简单的方式就是Ctrl + S，保存整个页面的源码。或者，按F12，右键源码最上方的HTML元素，copy – copy outerHTML，将整个页面的源码保存为一个文本文件。通过python re模块，以行为单位读取页面的源码，可以发现，含有所需名称的元素通常为如下形式：

```
<a class="icon file" draggable="true"
href="/gene/DATA/ASN_BINARY/Mammalia/All_Mammalia.ags.gz">All_Mammalia.ags.gz</a>
```

我们可以直接使用m = re.match(r”.+href=\”(.+)\”>.+ >”,line)进行匹配，读取m.group(1)即可获得想要的结果

在掌握了最简单的方法后，我们希望以一种更聪明的方法来定位所要元素。

```
Elements  Console  Sources  Network  Timeline  Profiles  Resources  Security  Audits  1
<!DOCTYPE html>
<html i18n-values="dir:textdirection;lang:language" dir="ltr" lang="undefined" i18n-processed>
  ▶<head>…</head>
  ▼<body>
    ▶<div id="listingParsingErrorBox" i18n-values=".innerHTML:listingParsingErrorBoxText">…</div>
     <span id="parentDirText" style="display:none" i18n-content="parentDirText">[上级目录]</span>
     <h1 id="header" i18n-content="header">/gene/DATA/ASN_BINARY/Mammalia/ 的索引</h1>
    ▼<table>
      ▶<thead>…</thead>
      ▼<tbody id="tbody">
        ▶<tr>…</tr>
        ▼<tr>
          ▼<td data-value="All_Mammalia.ags.gz">
             <a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/All_Mammalia.ags.gz">All_Mammalia.ags.gz</a> == $0
           </td>
           <td class="detailsColumn" data-value="1875902569">1.7 GB</td>
           <td class="detailsColumn" data-value="1487615460">2017/2/21 上午2:31:00</td>
         </tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
        ▶<tr>…</tr>
```

```
Styles  Computed  Event Listeners  »
Filter                    :hov  .cls  +
element.style {
}
a.icon:hover {                                  (index):156
   text-decoration: underline;
}
a.file {                                        (index):160
   background:▶
      url("…
      R2AYBvyQ8L0Gk8MwREBrTfKe4TpTzwhArXWi8H…
      ") left top no-repeat;
}
a.icon {                                        (index):151
   -webkit-padding-start: 1.5em;
   ~~text-decoration: none;~~
}
a[Attributes Style] {
   -webkit-user-drag: element;
   -webkit-user-select: none;
}
a:-webkit-any-link {        user agent stylesheet
   color: -webkit-link;
   ~~text-decoration: underline;~~
   cursor: auto;
}
Inherited from table
table {                                         (index):137
```

在用F12打开控制台后，点击左上方用红框标出的审核按钮，选择页面中的元素，即可直接导向源代码所在的位置。

此时，我们选择copy – copy Xpath，即可获得元素的Xpath路径， Xpath为xml语言用于确定某部分位置的语言。在之后的scrapy模块中我们将使用到。

一种处理网页源代码的模块是BeautifulSoup模块，相较于使用re模块这种原始的方法，BeautifulSoup模块可以更快捷的实现我们想要的功能。

html.txt new.py

```python
with open("html.txt","r") as r:
    content = r.read()
    
# print content

from bs4 import BeautifulSoup


soup = BeautifulSoup(content,'html.parser',from_encoding='utf-8')

links = soup.find_all("a")

for link in links:
    print link
```

```
:\>python new.py
a href="http://code.google.com/p/chromium/issues/entry">鍔ュ慷閬欒绝</a>
a href="LOCATION">鍘熶  鍒楀〃</a>
a class="icon up" href="/gene/DATA/ASN_BINARY/Mammalia/..">[涓嬭骇鏋缂綍]</a>
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/All_Mammalia.ags.gz">All_Mammalia.ags.gz</a
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Bos_taurus.ags.gz">Bos_taurus.ags.gz</a>
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Canis_familiaris.ags.gz">Canis_familiaris.a
gz</a>
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Homo_sapiens.ags.gz">Homo_sapiens.ags.gz</a
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Mus_musculus.ags.gz">Mus_musculus.ags.gz</a
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Pan_troglodytes.ags.gz">Pan_troglodytes.ags
</a>
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Rattus_norvegicus.ags.gz">Rattus_norvegicus
s.gz</a>
a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Sus_scrofa.ags.gz">Sus_scrofa.ags.gz</a>
```

```python
with open("html.txt","r") as r:
    content = r.read()

# print content

from bs4 import BeautifulSoup


soup = BeautifulSoup(content,'html.parser',from_encoding='utf-8')

links = soup.find_all(attrs = {"class":"icon file"})

for link in links:
    print link
```

```
Z:\>python new.py
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/A11_Mammalia.ags.gz">A11_Mammalia.ags.gz</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Bos_taurus.ags.gz">Bos_taurus.ags.gz</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Canis_familiaris.ags.gz">Canis_familiaris.ags
.gz</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Homo_sapiens.ags.gz">Homo_sapiens.ags.gz</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Mus_musculus.ags.gz">Mus_musculus.ags.gz</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Pan_troglodytes.ags.gz">Pan_troglodytes.ags.g
z</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Rattus_norvegicus.ags.gz">Rattus_norvegicus.a
gs.gz</a>
<a class="icon file" draggable="true" href="/gene/DATA/ASN_BINARY/Mammalia/Sus_scrofa.ags.gz">Sus_scrofa.ags.gz</a>
```

```python
with open("html.txt","r") as r:
    content = r.read()

# print content

from bs4 import BeautifulSoup


soup = BeautifulSoup(content,'html.parser',from_encoding='utf-8')

links = soup.find_all(attrs = {"class":"icon file"})

for link in links:
    print link["href"]
```

```
Z:\>python new.py
/gene/DATA/ASN_BINARY/Mammalia/All_Mammalia.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Bos_taurus.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Canis_familiaris.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Homo_sapiens.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Mus_musculus.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Pan_troglodytes.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Rattus_norvegicus.ags.gz
/gene/DATA/ASN_BINARY/Mammalia/Sus_scrofa.ags.gz
```

值得一提的是，在find_all()函数中，可以直接使用正则表达式，如soup.find_all(href=re.compile(“ASN_BINARY”)), 更多关于BeautifulSoup的用法留给大家去发现。

使用scrapy模块来编写网络爬虫

```
sanger-dev@login-0-1: 09:53:19 ~/app/program/Python/bin
$scrapy --help
Scrapy 1.3.0 - no active project

Usage:
  scrapy <command> [options] [args]

Available commands:
  bench         Run quick benchmark test
  commands
  fetch         Fetch a URL using the Scrapy downloader
  genspider     Generate new spider using pre-defined templates
  runspider     Run a self-contained spider (without creating a project)
  settings      Get settings values
  shell         Interactive scraping console
  startproject  Create new project
  version       Print Scrapy version
  view          Open URL in browser, as seen by Scrapy

  [ more ]      More commands available when run from project directory

Use "scrapy <command> -h" to see more info about a command
sanger-dev@login-0-1: 09:53:40 ~/app/program/Python/bin
$pwd
/mnt/ilustre/users/sanger-dev/app/program/Python/bin
```

由于我们已经将上图中的目录加入了环境变量，可以直接键入scrapy startproject <项目名称>，来创建一个项目，键入后将会在文件夹中找到一个以项目名称命名的文件夹。

以一个命名为cancer的项目为例，进入文件夹后，有一个以cancer命名的文件夹，一个我自己建立的log.txt文件， 一个scrapy.cfg配置文件。在cancer文件夹中，存在items.py、settings.py等一些文件，和一个命名为spiders的文件夹。我们的爬虫脚本将放在这个文件夹中。

```
.
├── cancer
│   ├── __init__.py
│   ├── __init__.pyc
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   ├── settings.pyc
│   └── spiders
│       ├── __init__.py
│       ├── __init__.pyc
│       ├── Molecular Profiling of Acute Myeloid Leukemia - My Cancer Genome.html
│       ├── new.py
│       └── new.pyc
├── log.txt
└── scrapy.cfg

2 directories, 14 files
```

为保险起见，编写及测试爬虫脚本之前，我们先对settings.py进行设置。在settings.py中，大部分配置保持默认设置即可。而DOWNLOAD_DELAY我们将其取消注释，并设置为一个不小于一定数值的整数(如30)。这里我们将DOWNLOAD_DELAY设置为3，表示3s进行一次爬取。

```
# Configure maximum concurrent requests performed by Scrapy (default: 16)
#CONCURRENT_REQUESTS = 32

# Configure a delay for requests for the same website (default: 0)
# See http://scrapy.readthedocs.org/en/latest/topics/settings.html#download-delay
# See also autothrottle settings and docs
DOWNLOAD_DELAY = 3
# The download delay setting will honor only one of:
#CONCURRENT_REQUESTS_PER_DOMAIN = 16
#CONCURRENT_REQUESTS_PER_IP = 16

# Disable cookies (enabled by default)
#COOKIES_ENABLED = False

# Disable Telnet Console (enabled by default)
#TELNETCONSOLE_ENABLED = False

# Override the default request headers:
#DEFAULT_REQUEST_HEADERS = {
#   'Accept': 'text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8',
#   'Accept-Language': 'en',
#}
```

进入spiders文件夹，打开new.py

```
import scrapy
import re, os

class DmozSpider(scrapy.spiders.Spider):
    name = "mycancer"
    allowed_domains = ["www.mycancergenome.org"]
    """
    lst = []
    with open("/mnt/ilustre/users/sanger-dev/app/program/Python/bin/cancer/cancer/spiders/Molecular Profiling of Acute Myeloid Leukemia - My Cancer Genome.html") as r:
        for line in r:
            if not line.startswith("<li class=\"submenu_li\">"):
                pass
            else:
                line = r.next()
                if line.find("href=\"") != -1:
                    url = re.match(".+href=\"(.+)\">",line).group(1)
                    lst.append(url)
    start_urls = lst
    """
    start_urls = ["https://www.mycancergenome.org/content/disease/acute-lymphoblastic-leukemia/"]
```

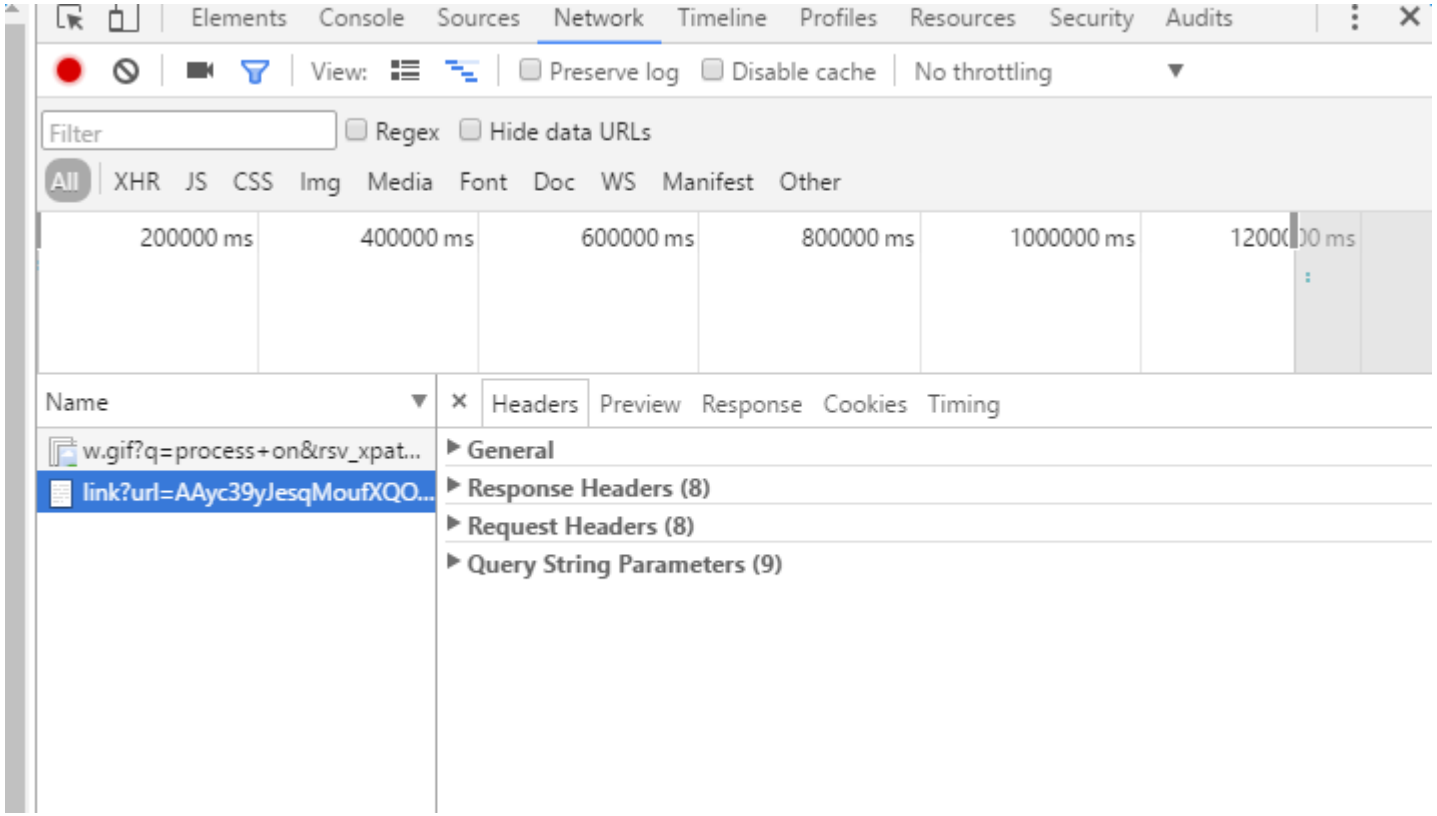
我们先定义一个爬虫类，该类继承自scrapy.spiders.Spider，其中name我们自定义一个名字，allowed_domains为指定爬虫爬取的网页范围，start_urls表示起始的爬取位点。

```
    def parse(self,response):
        """
        filename = response.url.split("/")[-2]
        filename = os.path.join("/mnt/ilustre/users/sanger-dev/sg-users/shijin/scrapy", filename)
        with open(filename,"wb") as f:
            f.write(response.body)
        with open("/mnt/ilustre/users/sanger-dev/sg-users/shijin/scrapy/list.txt","a") as w:
            w.write(response.url + "\t" + filename + "\n")


        for line in response:
            if line.startswith("<li class=\"submenu_li\">"):
                print line
                """
        # print response.body
        file_name = response.url.split("/")[-4] + "_" + response.url.split("/")[-3] + "_" + response.url.split("/")[-2]
        file_path = os.path.join("/mnt/ilustre/users/sanger-dev/sg-users/shijin/scrapy", file_name)
        with open(file_path,"wb") as f:
            f.write(response.body)
        for i in response.xpath('//ul/li/a/@href').extract():
            # print i
            yield scrapy.Request(i, callback=self.parse)
(END)
```

之后，我们要加上一个parse函数，该函数的参数为网站的response。我们通过对网站的源码的处理，获得下一步的链接，使用python生成器来生成进一步要爬取的网站。

网页的http请求



同样是在控制台中，选择network工具窗口，当提交某一访问时（打开某一超链接），可以看到Request信息和Response信息。其中response会作为parse()方法的参数传入。

数据库爬虫的几条tips

1. 构建数据库时，需要知道数据来源，即，对抓取网页进行保留备份，以方便对数据进行验证
2. 在保留网页后，根据网页在网站中的相对位置进行整理分类。
3. 对网页元素进行整理分类，建立表格、数据库

```
file_name = response.url.split("/")[-4] + "_" + response.url.split("/")[-3] + "_" + response.url.split("/")[-2
file_path = os.path.join("/mnt/ilustre/users/sanger-dev/sg-users/shijin/scrapy", file_name)
with open(file_path,"wb") as f:
    f.write(response.body)
```

这里，我保存了访问网站的全文，文件起名方式为：对地址进行分割，取后三位，以下划线进行连接。这样可以在结果文件夹中直接看到网站的一个结构信息。也可以写一个list文件，对文件夹中的网页地址与文件名称的对应关系进行记录。

```
for i in response.xpath('//ul/li/a/@href').extract():
    # print i
    yield scrapy.Request(i, callback=self.parse)
```

这里，我使用了xpath来对下一步要访问的地址进行定位。如果不会写xpath路径，可以使用之前说的copy Xpath来获得获得元素的xpath位置。

退回到项目文件夹的根目录，运行爬虫：
scrapy crawl mycancer

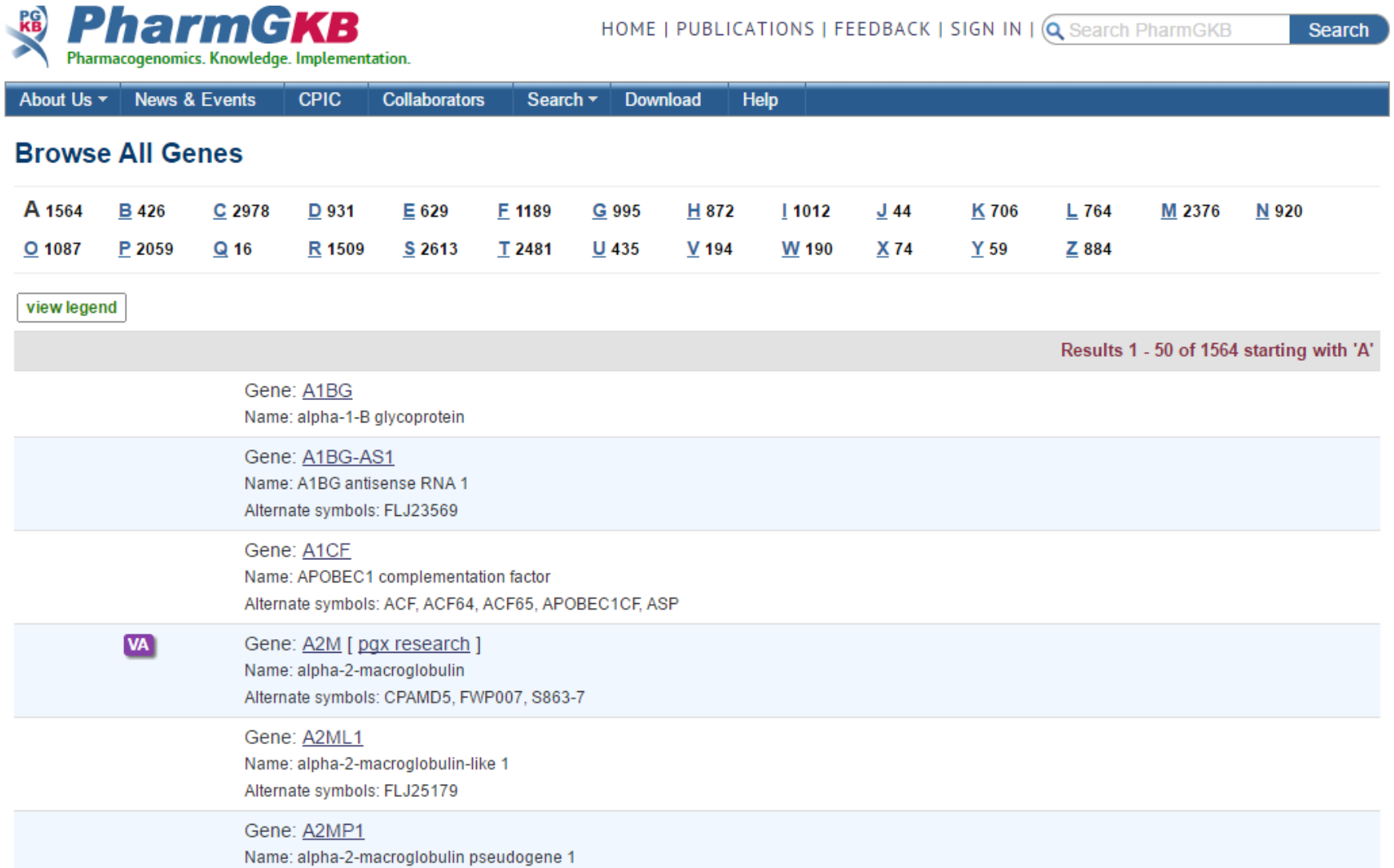
可以使用nohup命令， nohup scrapy crawl mycancer &> log.txt &

页面抓取的完成总共用了两个小时左右。

```
2016-12-28 18:51:27 [scrapy.extensions.logstats] INFO: Crawled 1365 pages (at 19 pages/min), scraped 0 items (at 0 items/min)
2016-12-28 18:51:29 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://www.mycancergenome.org/content/gene/stat6/> (referer: https://www.mycancergenome.org/content/gene/ret/)
2016-12-28 18:51:29 [scrapy.core.engine] INFO: Closing spider (finished)
2016-12-28 18:51:29 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/exception_count': 145,
 'downloader/exception_type_count/twisted.internet.error.TCPTimedOutError': 2,
 'downloader/exception_type_count/twisted.internet.error.TimeoutError': 49,
 'downloader/exception_type_count/twisted.web._newclient.ResponseFailed': 27,
 'downloader/exception_type_count/twisted.web._newclient.ResponseNeverReceived': 67,
 'downloader/request_bytes': 1508244,
 'downloader/request_count': 1799,
 'downloader/request_method_count/GET': 1799,
 'downloader/response_bytes': 114633377,
 'downloader/response_count': 1654,
 'downloader/response_status_count/200': 1365,
 'downloader/response_status_count/301': 288,
 'downloader/response_status_count/404': 1,
 'dupefilter/filtered': 756974,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2016, 12, 28, 10, 51, 29, 482466),
 'log_count/DEBUG': 1883,
 'log_count/ERROR': 5,
 'log_count/INFO': 134,
 'offsite/domains': 82,
 'offsite/filtered': 1601,
 'request_depth_max': 8,
 'response_received_count': 1366,
 'scheduler/dequeued': 1798,
 'scheduler/dequeued/memory': 1798,
 'scheduler/enqueued': 1798,
 'scheduler/enqueued/memory': 1798,
 'spider_exceptions/ValueError': 5,
 'start_time': datetime.datetime(2016, 12, 28, 8, 45, 27, 545265)}
2016-12-28 18:51:29 [scrapy.core.engine] INFO: Spider closed (finished)
(END)
```

使用ajax方法的网页的爬取方法：

案例： https://www.pharmgkb.org/search/browseAlpha.action?browseKey=allGenes



为什么要将这种网站单独拿出来讲？因为在这种网页中，点击Next 50 或选择页面，地址栏中网址都是不变的。

Gene: ABCA6
Name: ATP-binding cassette, sub-family A (ABC1), member 6
Alternate symbols: EST155051

Gene: ABCA7
Name: ATP-binding cassette, sub-family A (ABC1), member 7
Alternate symbols: ABCX

VA
Gene: ABCA8 [pgx research]
Name: ATP-binding cassette, sub-family A (ABC1), member 8
Alternate symbols: KIAA0822

Gene: ABCA9
Name: ATP-binding cassette, sub-family A (ABC1), member 9
Alternate symbols: EST640918

DL VA VIP
Gene: ABCB1 [VIP annotation] [pgx research]
Name: ATP-binding cassette, sub-family B (MDR/TAP), member 1
Alternate symbols: ABC20, CD243, GP170, MDR1, P-gp

Gene: ABCB10
Name: ATP-binding cassette, sub-family B (MDR/TAP), member 10
Alternate symbols: EST20237, M-ABC2, MTABC2

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

Next 50 >>

Jump to page: 1

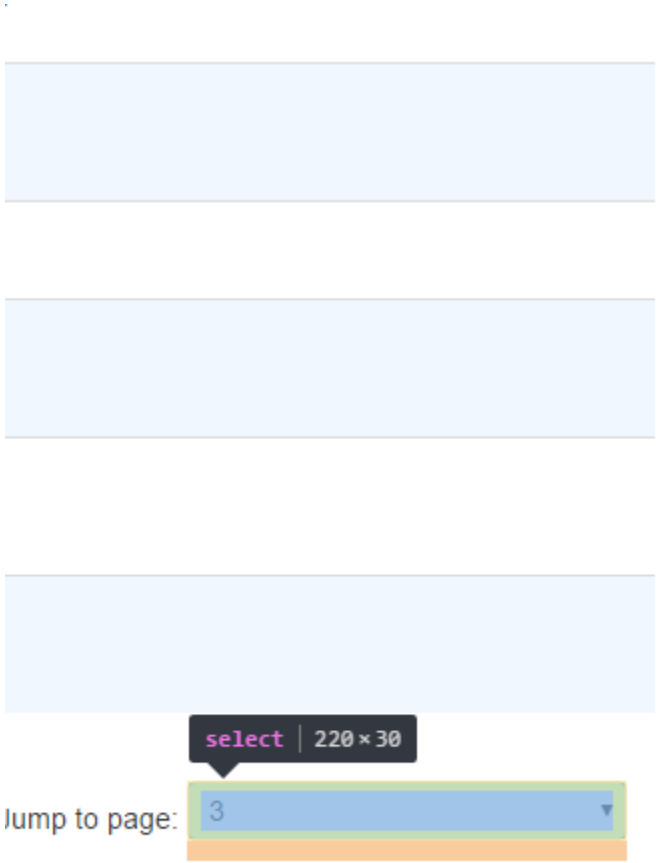
Search index last updated on 17-2-21 上午4:25.

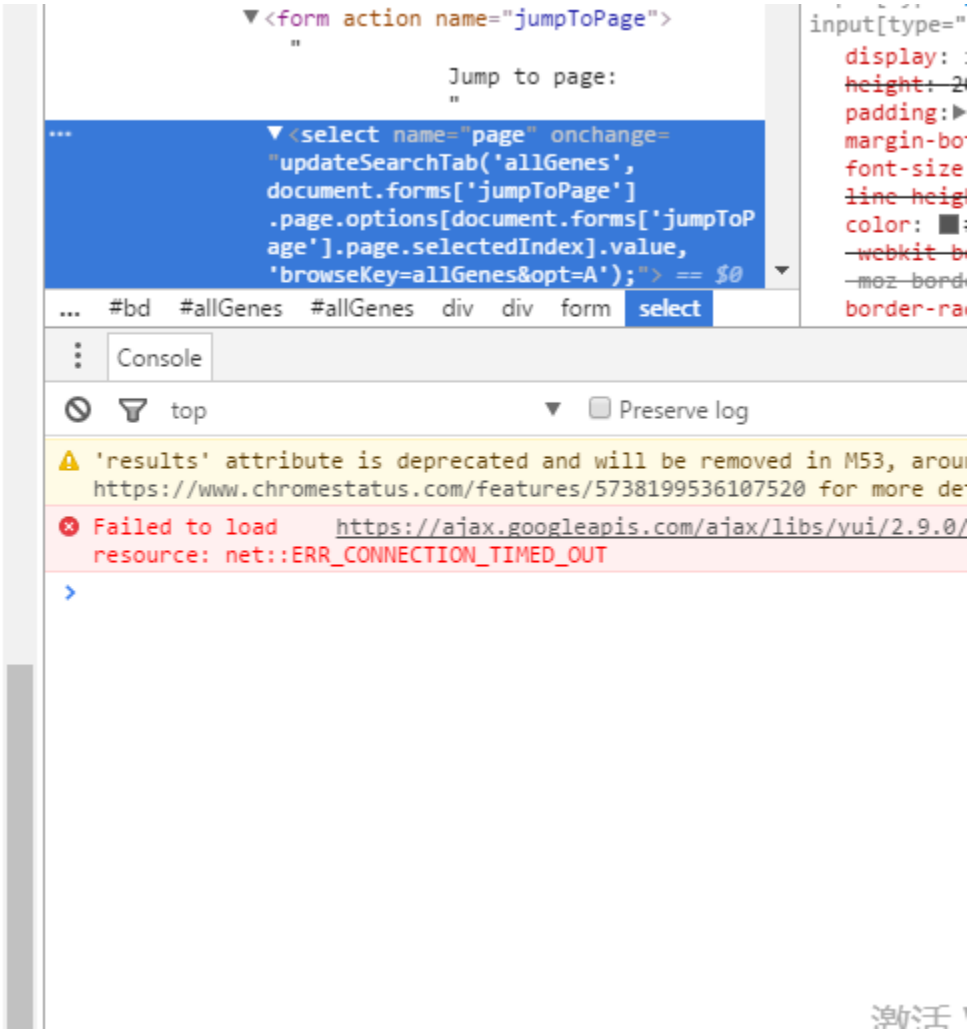
那么我们便打开控制台，点击network工具窗口，此时点击一个next 50，获得一个response信息，我们把response header复制如下：

GET /search/browseAlpha.action?returnType=ajax&browseKey=allGenes&opt=A&startIndex=100 HTTP/1.1
Host: www.pharmgkb.org
Connection: keep-alive
Accept: text/html, */*; q=0.01
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.106 Safari/537.36
Referer: https://www.pharmgkb.org/search/browseAlpha.action?browseKey=allGenes
Accept-Encoding: gzip, deflate, sdch, br
Accept-Language: zh-CN,zh;q=0.8
Cookie: JSESSIONID=33778F551F5D49EED12E4689324D2A48; _gat=1; _ga=GA1.2.1496219068.1483931103; _gali=yui_3_16_0_1_1487732837167_32

进行许多次比较，我们发现，只有startIndex产生变化

选中下图中的下拉菜单元素，对其进行复制





我们发现了一组value和页面数字的对应关系，那么是不是更改startIndex=100中的100为这些value值，就可以获得网站的内容了呢？

```
<select name="page" onchange="updateSearchTab('allGenes', document.forms['jumpToPage'].page.options[document.forms['jumpToPage'].page.selectedIndex].value, '
            <option value="0">1</option>
            <option value="50">2</option>
            <option value="100" selected="true">3</option>
            <option value="150">4</option>
            <option value="200">5</option>
            <option value="250">6</option>
            <option value="300">7</option>
            <option value="350">8</option>
            <option value="400">9</option>
            <option value="450">10</option>
            <option value="500">11</option>
            <option value="550">12</option>
            <option value="600">13</option>
            <option value="650">14</option>
            <option value="700">15</option>
            <option value="750">16</option>
            <option value="800">17</option>
            <option value="850">18</option>
```

答案是一定的。不过为了验证想法，我们对updateSearchTab函数进行搜索，得到结果：

```
<select name="page" onchange="updateSearchTab('allGenes', document.forms['jumpToPage'
            <option value="0">1</option>
            <option value="50">2</option>
            <option value="100" selected="true">3</option>
```

```
<script type="text/javascript">
  var browseId = 'allGenes';
  // called by UI controls
  function updateSearchTab(tabId, startIndex, queryArgs) {
    $.ajax({
        url: '/search/browseAlpha.action?returnType=ajax&' + queryArgs + '&startIndex=' + startIndex,
        context: this,
        type: 'GET',
        dataType: 'html'
      })
      .done(function(response) {
        $('#' + browseId).html(response);
      });
  }
</script>
```

函数中的tabId，startIndex，queryArgs三个参数与select中的参数进行对应，url的组成由传入参数的变化而变化。获得了url信息后，我们可以使用之前讲过的方法来获取所需要的页面。

作业：
抓取https://www.pharmgkb.org网站

谢谢观看！