第19章 网络爬虫基础

随着大数据时代的到来，每一天都会有数以万亿级字节的数据产生。数据从何而来？企业产生的用户数据，例如百度指数、阿里指数、TBI腾讯浏览指数、新浪微博指数等。政府或机构公开的数据，例如中华人民共和国国家统计局数据、世界银行公开数据、联合国数据、纳斯达克数据等。数据是否可以进行交易？数据堂、国云数据市场、贵阳大数据交易所等提供买卖数据的平台，麦肯锡、埃森哲、艾瑞咨询等企业提供数据管理咨询服务。为什么要做爬虫？如果需要的数据市场上没有，可以直接从允许爬取的网站上获取。读者尤其需要注意的是爬虫必须遵循一定规则，本章中介绍的案例仅供读者用于安全研究和学习。此外，爬虫与反爬虫技术相互博弈，这也就意味着爬虫与反爬虫技术都会与时俱进，因此，本章主要介绍利用Python语言的内置模块与第三方模块编写基本的网络爬虫脚本，从而实现对网站信息的定向采集。

**19.1 网络爬虫技术基础**

在开始正式介绍具体的爬虫技术之前，读者还应分清几个基本概念。HTML（Hyper Text Markup Language）是超文本标记语言。它包括一系列标签，通过这些标签统一网络上的文档格式，使得因特网上的资源形成了一个逻辑整体。HTML文本是指由HTML命令组成的描述性文本。HTML命令可以说明文字、图像、动画、声音、表格与链接等，在HTML4.0版本中，只定义了70余种标记，并且这种标记是固定的，也就是说HTML是不可扩展的，它的最新版本是HTML5。浏览器将HTML文本翻译为可识别的信息，即网页。

随着Web应用的不断发展，HTML的局限性逐渐显露，如HTML无法描述数据、可读性差、搜索时间长等。为此，开发人员设计了XML（Extensible Markup Language，可扩展标记语言），它可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言，非常适合万维网传输。XML文件格式是纯文本格式，在许多方面类似于HTML，XML由XML元素组成，每个XML元素包括一个开始标记、一个结束标记以及两个标记之间的内容。另外，XML可以被各种应用程序处理。

HTTP（Hyper Text Transfer Protocol，超文本传输协议），这是一个简单的请求-响应协议。HTTP指定了客户端发送给服务器的消息类型以及得到响应的类型。URL（uniform resource locator，统一资源定位系统），是一种用于指定信息位置的表示方法。常见的网页中大部分使用HTTP URL语法，具体形式如下：<http://<host>:<port>/<path>?<searchpart>>

其中<host>表示域名；<port>表示端口号，若省略:<port>则通常使用缺省的端口80；<path>是一个HTTP选择器；<searchpart>则是查询字符串；当<path>和<searchpart>都不存在时，/<path>?<searchpart>可以整体省略。

更多关于HTML和XML的技术细节请参阅文献[]。

根据需求不同，爬虫可以分为两类。一类为通用爬虫，另外一类是聚焦爬虫。通用爬虫是搜索引擎（百度，谷歌，雅虎，必应等）系统重要的组成部分，它将互联网中采集的网页保存至本地，形成网页镜像备份。搜索引擎将通用爬虫爬取的网页进行各种预处理，最后由搜索引擎根据特定的排序算法呈现给用户。由此可见，通用爬虫功能有限，它并不能针对用户的需求进行有针对性的搜索。而聚焦爬虫在实施网页抓取时，会对内容进行处理筛选，只抓取与需求相关的网页信息。

一个网络爬虫（下文简称爬虫）的基础工作流程如下图19-1所示。



图19-1 爬虫基础工作流程

本章也将按照图19-1的顺序介绍爬虫的相关技术，但在保存数据环节本章中使用的是Python自带的文件处理模块，如需使用SQLite数据库进行存储，读者可以参考本书的第12章，如需选择其他数据库，如MySQL，读者可参考其官方网址提供的帮助文档。

**19.2爬虫环境搭建**

本小节介绍Python爬虫的环境搭建方法，首先需要下载并安装与爬虫相关的模块。

1. Requests模块

Requests是一个优雅而简单的Python HTTP库，它主要负责发送HTTP请求并获取响应，从而实现源码采集。在PyCharm的Terminal中输入安装指令即可完成安装，如下：

pip install requests

2. Selenium模块

Selenium是一个Web项目的自动化测试模块，它也可以完成发送HTTP请求并获取响应的功能。其安装方式如下：

pip install selenium

3. lxml模块

lxml是XML和HTML的解析器，其主要功能是高性能的解析和提取XML和HTML中的数据。lxml和正则一样，也是用C语言实现的，也可以利用XPath语法来定位特定的元素及节点信息。在本章中它用于解析HTML。其安装方式如下：

pip install lxml

接下来，为了配合Selenium模块实现网络源码采集，需要在计算机中安装谷歌Chrome浏览器。安装完毕后确认Chrome浏览器的版本，在浏览器主界面点解右上角“三个点”样式的图标，弹出菜单后点击菜单中的“设置”选项，进入谷歌浏览器设置界面，再点击左侧导航栏最下方的“关于Chrome”的选项，即可查看到浏览器版本，如图19-2所示。

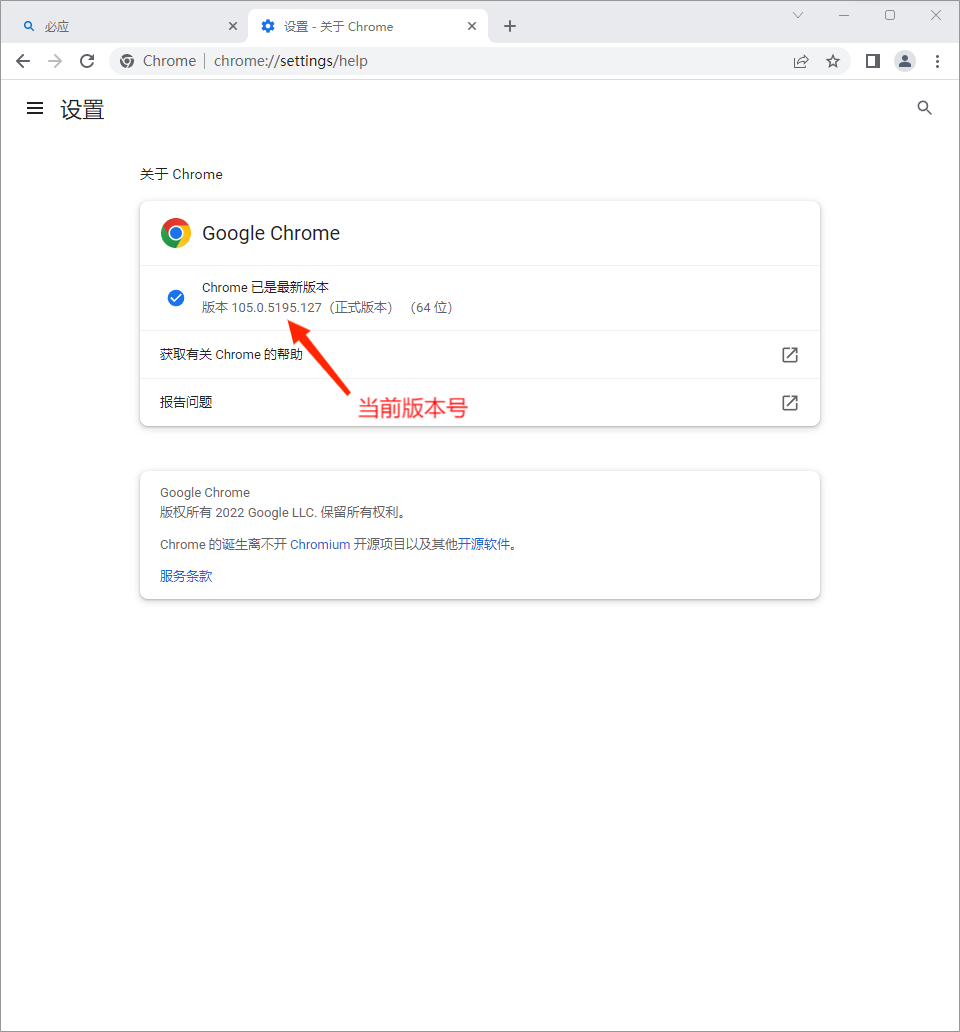


图19-2 查看Chrome浏览器版本

通过上述步骤后发现Chrome浏览器版本为105.0.5195.127（正式版本）（64位），通过浏览器镜像网站下载相对应版本的chromeDriver驱动文件，读者需注意的是还应选择与操作系统相匹配的驱动文件，例如作者使用的是Windows32操作系统，那么应下载chromedriver\_win32.zip文件，如图19-3所示。

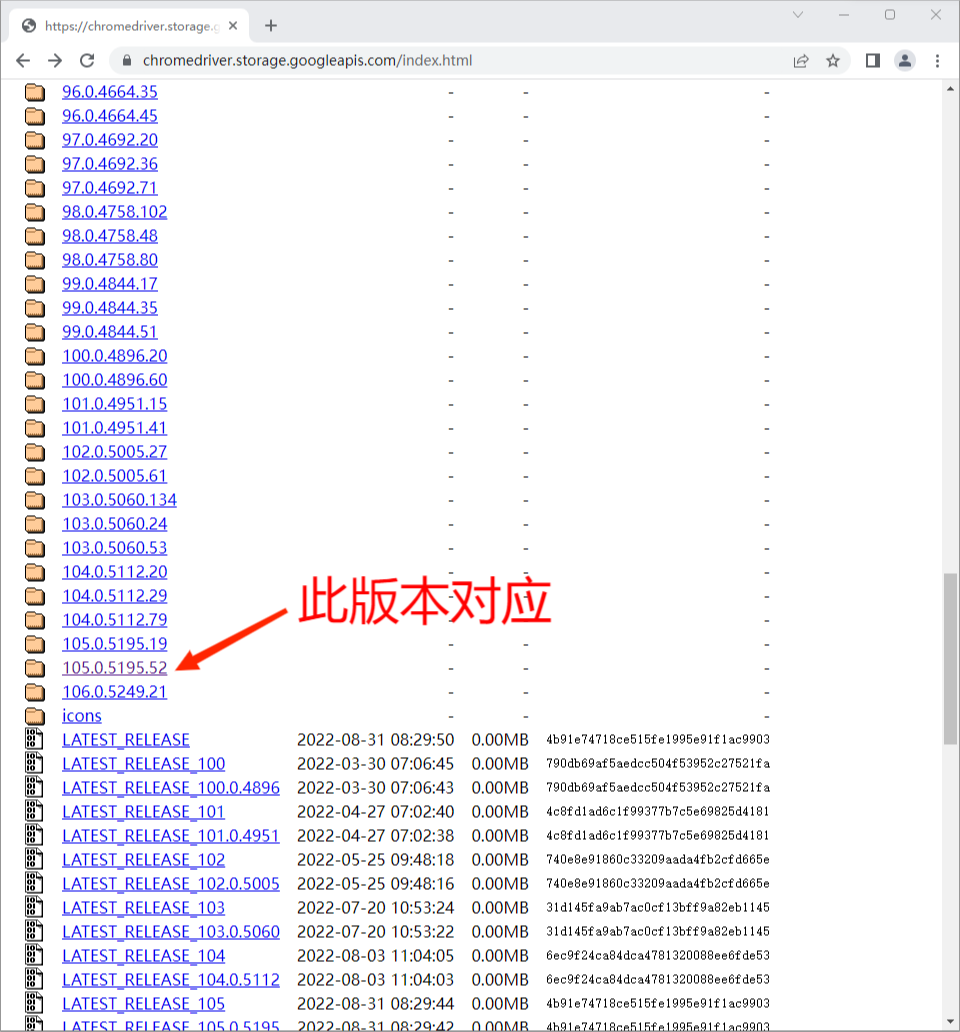


图19-3（1） 选择对应Chrome浏览器的版本号

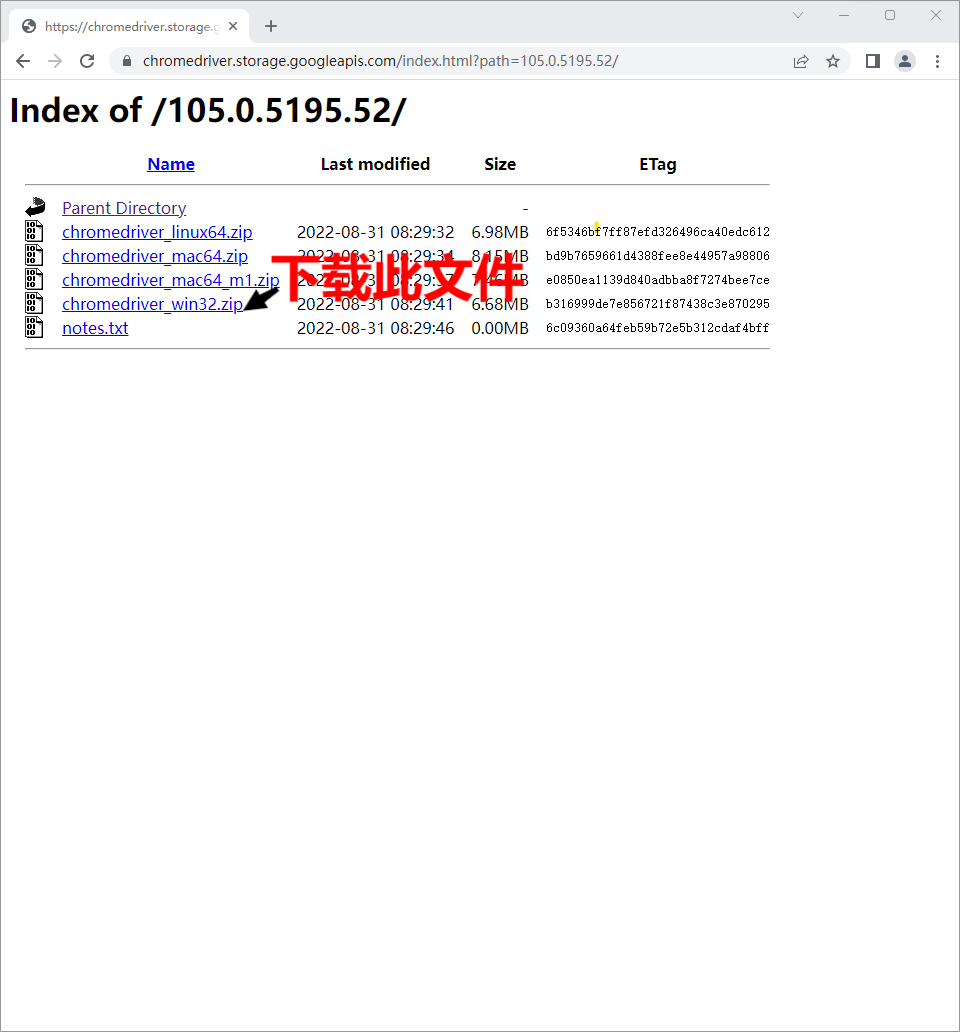


图19-3（2） 根据操作系统选择对应chromeDriver

图19-3 选择chromeDriver驱动文件

下载完该文件后，将文件放入指定路径（读者应明确该路径信息，便于后续实验）中解压缩即可。进入PyCharm编辑以下代码：

from selenium import webdriver

import time

browser= webdriver.Chrome(executable\_path='D:\chromedriver\chromedriver.exe')

browser.get('http://www.baidu.com')

time.sleep(15)

browser.close()

注意executable\_path选项中要添加之前的谷歌浏览器驱动所在的路径，执行成功后会自动弹出Chrome浏览器并显示百度主页面，如图19-4所示。

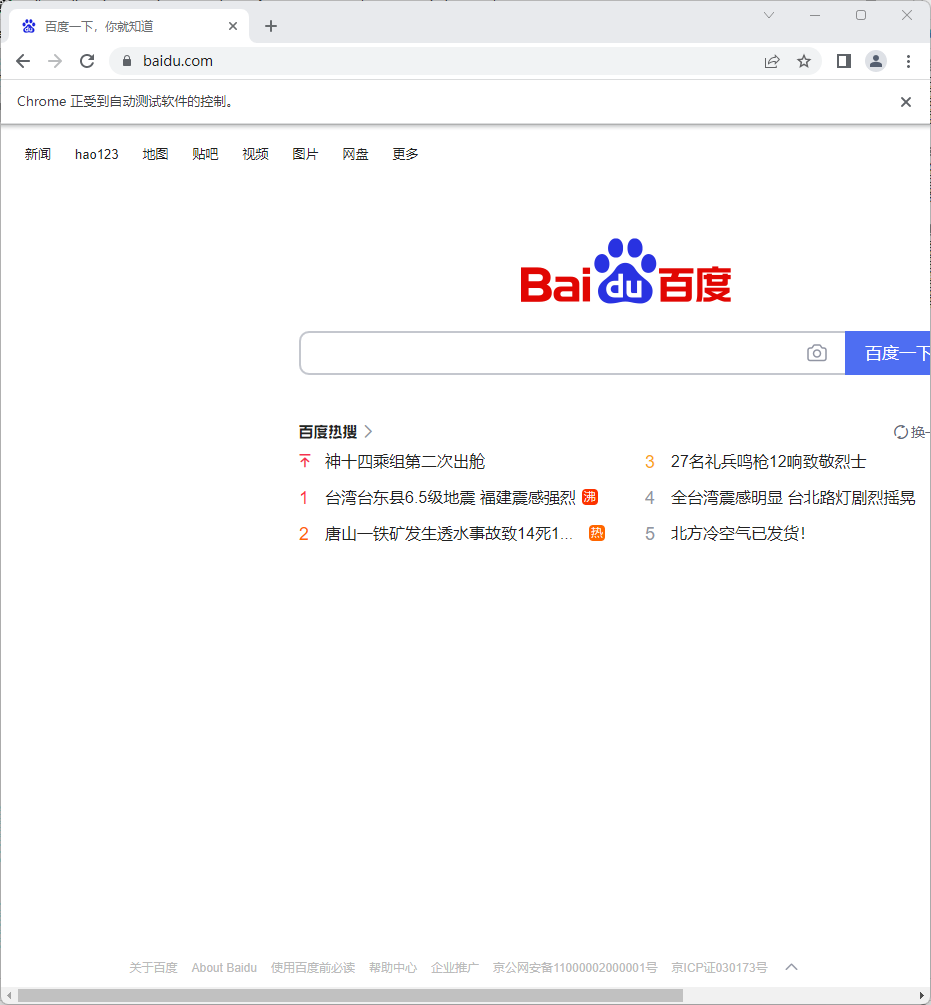


图19-4 使用Selenium模块弹出Chrome浏览器并访问指定网页

**19.3 源码采集**

在聚焦网络爬虫技术中，数据基本上都是从网络源码中采集，所以采集网页的HTML源码尤为重要，本节中分别利用urllib内置库、Requests第三方模块和Selenium第三方模块进行网页的HTTP请求并采集网页源码。三个库的对比如表19-1所示

表19-1 urllib、Requests和Selenium的对比

|  |  |  |  |
| --- | --- | --- | --- |
|  | urllib | Requests | Selenium |
| 安装方面 | Python自带，无需额外安装。 | 需自行安装，安装指令:pip install requests | 需自行安装，安装指令:pip install selenium |
| 优点 | 无需安装，可直接导入使用。 | 相比于urllib，requests更加简便也更加快捷。 | Selenium的测试直接在浏览器中进行，更加接近于真人操作。 |
| 缺点 | 每次访问都要重新建立连接，耗费资源。 | 无法处理纯Javascript制作的网站 | 速度慢且资源耗费大 |

**19.3.1 使用urllib采集源码**

urllib是Python的标准库之一，主要用于发送Get请求然后返回HTTP响应，从而抓取源码。它可以处理cookie，同时也支持挂代理访问，基本满足源码采集需求。

urllib包中，使用频率最高的是urllib.request模块，urllib.request模块用于模拟浏览器的一个请求发起过程，这里介绍其中最常用的函数urlopen和其中的部分方法。

urlopen函数用于实现对目标url的访问，其中read() , readline() ,readlines()方法都是用于读取网页内容的，区别在于read()读取网页的全部内容，readline()读取网页的一行内容，readlines()读取全部内容并返回列表；getcode()方法用于返回网页的状态码，在HTTP请求中，若返回200则代表访问成功，404则代表未找到网页；geturl()方法可以返回请求的url。

下面，通过采集“哈尔滨职业技术学院”官方网站为例，介绍源码采集的过程，其代码如下：

#导入urllib包中的request模块

from urllib import request

#声明变量base\_url用于存储待采集的网址

base\_url = 'http://www.hzjxy.org.cn/'

#利用request.urlopen(传入采集网址)获取相应

response = request.urlopen(base\_url)

#打开用于存储网页源代码的文本文档

fp = open('yuanma.txt','a',encoding='utf-8')

#利用read函数读取响应值得注意的是字符集要按照网页源码为准并采集源码

html = response.read().decode('utf-8')

#将网页源代码显示在控制台上

print(html)

#将网页源代码写入文本文档

fp.write(html)

#关闭打开的文本文档

fp.close()

运行效果如图1.6所示。

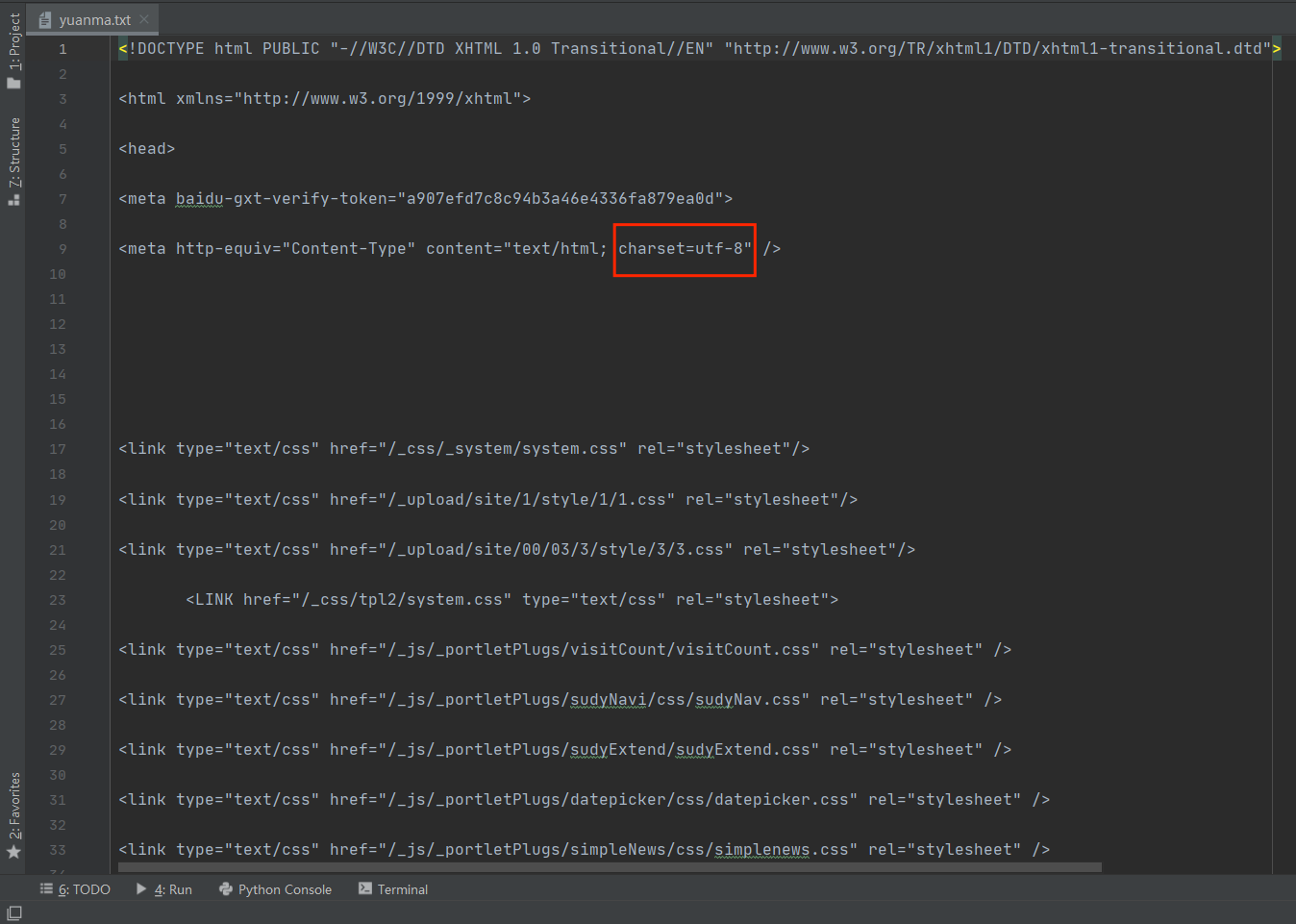


图1.6 确认字符集

当然也可以利用urllib模块加入请求头，一般网站都会验证请求头报文以确保服务端能够正确响应请求内容，为了能够模拟真实用户的发送请求所以在爬取网站时通常会加入请求头，当然并不是一定要加入全部请求头信息，一般都要加入User-Agent字段如果涉及到模拟登录也可以加入Cookie来保持登录状态，其他字段根据反爬手段在决定是否加入，请求头信息如图1.7所示。

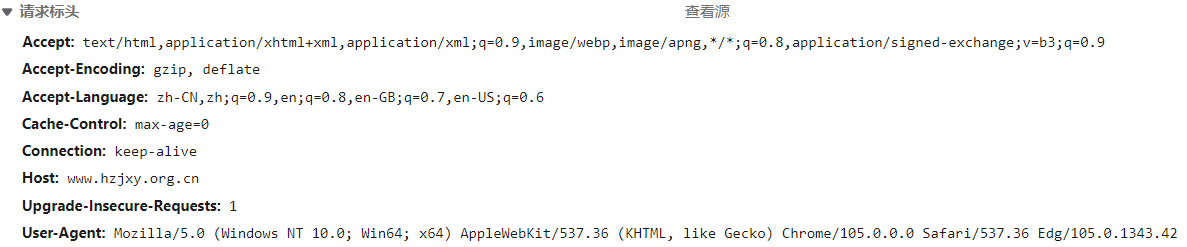


图1.7 请求报头信息

Accept：浏览器可接受的MIME（Multipurpose Internet Mail Extensions，多功能Internet邮件扩展）类型。

Accept-Encoding：浏览器能够进行解码的数据编码方式，比如gzip。Servlet能够向支持gzip的浏览器返回经gzip编码的HTML页面。许多情形下这可以减少5到10倍的下载时间。

Accept-Language：浏览器所希望的语言种类，当服务器能够提供一种以上的语言版本时要用到。

Cache-Control：通用消息头字段，被用于在 HTTP 请求和响应中，通过指定指令来实现缓存机制。

Connection：表示是否需要持久连接。如果Servlet看到这里的值为“Keep-Alive”，或者看到请求使用的是HTTP 1.1（HTTP 1.1默认进行持久连接），它就可以利用持久连接的优点，当页面包含多个元素时（例如Applet，图片），显著地减少下载所需要的时间。要实现这一点，Servlet需要在应答中发送一个Content-Length头，最简单的实现方法是：先把内容写入ByteArrayOutputStream，然后在正式写出内容之前计算它的大小。

Host：初始URL中的主机和端口。

html Upgrade-Insecure-Requests:1HTTP Upgrade-Insecure-Requests 请求头向服务器发送一个客户端对HTTPS加密和认证响应良好，并且可以成功处理的信号，可以请求所属网站所有的HTTPS资源。

User-Agent：浏览器类型。

利用urllib模块，加入请求头进行源码采集，请求头中只加入了User-Agent字段，代码如下：

from urllib import request

# 构建请求头字典

headers = {

'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36 Edg/105.0.1343.42'}

# 待采集的网址

base\_url = 'http://www.hzjxy.org.cn/'

# 构建带有请求头的网址

full\_url = request.Request(url=base\_url,headers=headers)

# 获取响应

response = request.urlopen(full\_url)

# 源码采集

fp = open('before.txt','a',encoding='utf-8')

html = response.read().decode('utf-8')

fp.write(html)

#关闭打开的文本文档

fp.close()

运行效果如图1.8所示。



图1.8 加入请求头的源码采集片段

利用urllib模块，加入请求头进行源码采集，请求头中不加入User-Agent字段时，代码如下：

from urllib import request

# 待采集的网址

base\_url = 'http://www.hzjxy.org.cn/'

# 构建带有请求头的网址

full\_url = request.Request(url=base\_url)

# 获取响应

response = request.urlopen(full\_url)

# 源码采集

fp = open('after.txt','a',encoding='utf-8')

html = response.read().decode('utf-8')

fp.write(html)

#关闭打开的文本文档

fp.close()

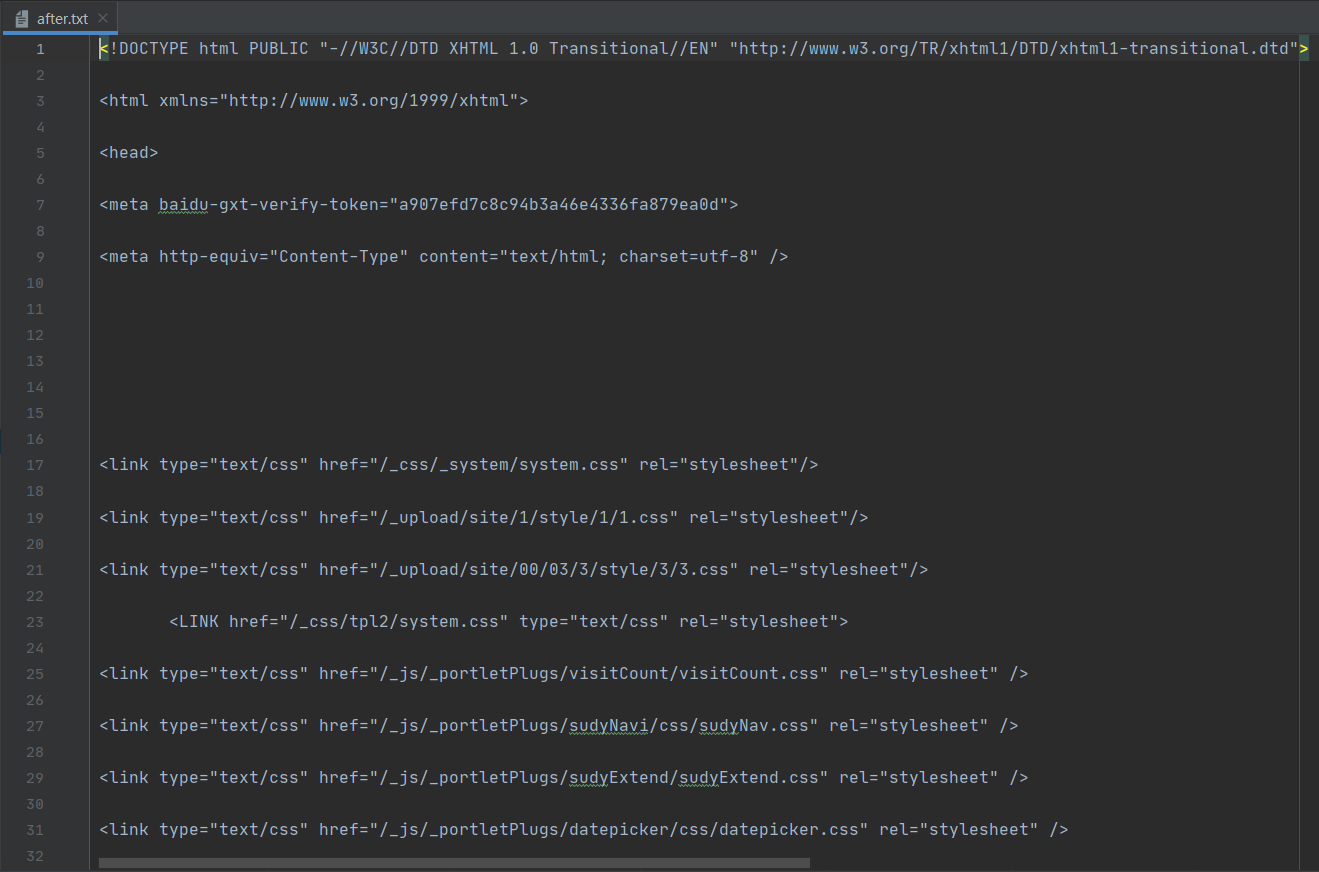


图1.9 不加入请求头的源码采集片段

从两种代码的运行结果来看，是否加入请求头对于此网站的源码采集结果没有明显的影响，这说明了此网站并没有对请求头进行验证。

**19.3.2 使用Requests采集源码**

本章节我们同样也介绍，普通采集模块和带有请求头的方式进行网站源码的采集。requests模块是python基于urllib开发的，专用于发送HTTP请求的第三方库，它可以直接构建常用的get和post请求并发起，而urllib一般要先构建get或者post请求，然后再发起请求。不难看出，相较于前身urllib，requests的使用更加简洁，也更加人性化。下面介绍requests模块中最常用的三种方法：

1.构造一个请求：requests.request()

2.获取网页中的信息，与HTTP中的GET方法相对应：requests.get()

3.向网页提交信息，与HTTP中的POST方法相对应：requests.post()

示例如下：

1.通过requests发送get请求并获取源码，代码如下：

# 导入requests第三方库

import requests

# 待采集网址

base\_url = 'http://www.hzjxy.org.cn/'

# 利用requests模块发送get请求访问网址

response = requests.get(url=base\_url)

# 利用网页源码进行解码

response.encoding = 'utf-8'

# 获取源码

html = response.text

2.带有请求报头信息进行网站源码采集，代码如下：

# 导入requests第三方库

import requests

# 构建请求头字典

headers = {

'User-Agent':'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36 Edg/105.0.1343.42'}

# 待采集网址

base\_url = 'http://www.hzjxy.org.cn/'

# 利用requests模块发送get请求访问网址并传入请求头

response = requests.get(url=base\_url,headers=headers)

# 利用网页源码进行解码

response.encoding = 'utf-8'

# 获取源码

html = response.text

此外，为了方便采集媒体信息可以使用content属性获取二进制流，关于content的使用.....。

**19.3.3 使用Selenium采集源码**

相比较上述两种通过代码进行发送get请求获取响应的方式不同，Selenium是通过谷歌浏览器驱动调用浏览器进行实际的网站请求并获取源码，从反爬虫的角度上看，此方法和真实的用户进行网站请求几乎一样，可规避一些基本的反爬技术，但很难通过具有严格反爬验证的网站。另外，在爬取网页时每次都需调用浏览器，相对于1.2.1节和1.2.2节中介绍的两种采集源码方法，使用Selenim模型进行源码采集的效率较低。

Selenium的基础用法如下：

1.调用selenium打开浏览器

s = Service(r'D:\\chromedriver\\chromedirver.exe')

browser = webdriver.Chrome(service = s)

2.使用selenium输入字符或键盘操作

input.send\_keys('人工智能')

input.send\_keys(Keys.ENTER)

利用Selenium采集源码，代码如下：

from selenium import webdriver

import time

# 初始化一个浏览器

browser = webdriver.Chrome(executable\_path='D:\chromedriver\chromedriver.exe')

# 发起HTTP请求

browser.get('http://www.hzjxy.org.cn/')

# 等待网页加载

time.sleep(15)

# 获取源码

html = browser.page\_source

# 给浏览器截图

browser.save\_screenshot('哈尔滨职业技术学院.png')

**19.4 数据解析**

**19.4.1 bs4 模块**

bs4，即BeautifulSoup，它是python的一个HTML 或 XML 解析库，最主要的功能就是从网页中抓取数据。当使用的bs4解析器不同时，返回的结果也可能不同，几种常用解析器的比较如表19-2

表19-2 解析器之间的比较

|  |  |  |  |
| --- | --- | --- | --- |
| 解析器 | 使用API | 优点 | 缺点 |
| python标准库 | BeautifulSoup(values,'html.parser') | 内置标准库，执行速度适中，文档容错能力强 | 旧版本容错能力差，需要安装C依赖 |
| LXML,HTML解析器 | BeautifulSoup(values,'lxml') | 速度快、文档容错能力强 | 需要安装C语言库 |
| LXML,XML解析器 | BeautifulSoup(values,'xml') | 速度快，唯一一个支持XML的解析器 | 需要安装C语言库 |
| html5lib | BeautifulSoup(values,'html5lib') | 提供最好的容错性，以浏览器解析文档，生成HTML5格式的文档 | 速度慢 |

bs4模块的常用方法如下

1.解析html字符串：初始化html对象并把对象赋值给soup变量，之后就可以用soup的各个方法和属性解析这段html代码，示例：

soup = BeautifulSoup(html,'html.parser')

2.节点选择器：调用节点的名称，但需要注意的是如果有多个要匹配的节点，那么只能匹配到第一个这样的节点，后面的节点会被忽略。示例：

soup.title

3.搜索内容：分为soup.find()和soup.find\_all()两种方法，soup.find()方法只返回第一个匹配到的对象，而soup.find\_all()则会返回所有匹配到的对象

4.格式化输出：soup.prettify()可以将要解析的字符串以标准的字符格式输出

5.获取元素的属性可以使用attrs属性，返回值是字典数据类型，利用属性的键可以获取值。

轮播图

**19.4.2 lxml 模块**

lxml是XML和HTML的解析器，其主要功能是解析和提取XML和HTML中的数据；与正则相同，lxml也是用C语言实现的，也可以利用XPath语法，来定位特定的元素及节点信息。

lxml库中大部分功能都位于lxml.etree 模块中，导入lxml.etree模块的常见方式：

from lxml import etree

Lxml模块的常用方法如下

1.从字符串常量中解析HTML文档或片段

etree.HTML()

etree.XML()

2.查找第一个匹配的子元素，若没有则返回None

f = etree.XML("<p><a x='777'>aText</a></p>")

f.find("a")

3.返回所有匹配的子元素，若找不到则返回[]

f = etree.XML("<p><a x='777'>aText<b/><c/><b/></a></p>")

f.findall(".//a[@x]")

XPath即为XML路径语言（XML Path Language），它是一种用来确定XML文档中某部分位置的语言。XPath基于XML的树状结构，提供在数据结构树中找寻节点的能力。起初XPath的提出的初衷是将其作为一个通用的、介于XPointer与XSL间的语法模型。但是XPath很快地被开发者采用来当作小型查询语言，常用规则如下表：

表19-3 XPath常用规则

|  |  |
| --- | --- |
| **表达式** | **描述** |
| nodename | 选取此节点的所有子节点。 |
| / | 从根节点选取。 |
| // | 从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。 |
| . | 选取当前节点。 |
| .. | 选取当前节点的父节点。 |
| @ | 选取属性。 |

利用网络爬虫技术采集“哈尔滨职业技术学院”所有图书馆（网址：[新闻 (hzjxy.org.cn)](http://lib.hzjxy.org.cn/xw/list1.htm)）模块新闻信息并保存至文本文档中，采集字段以及页数如图1.10所示：

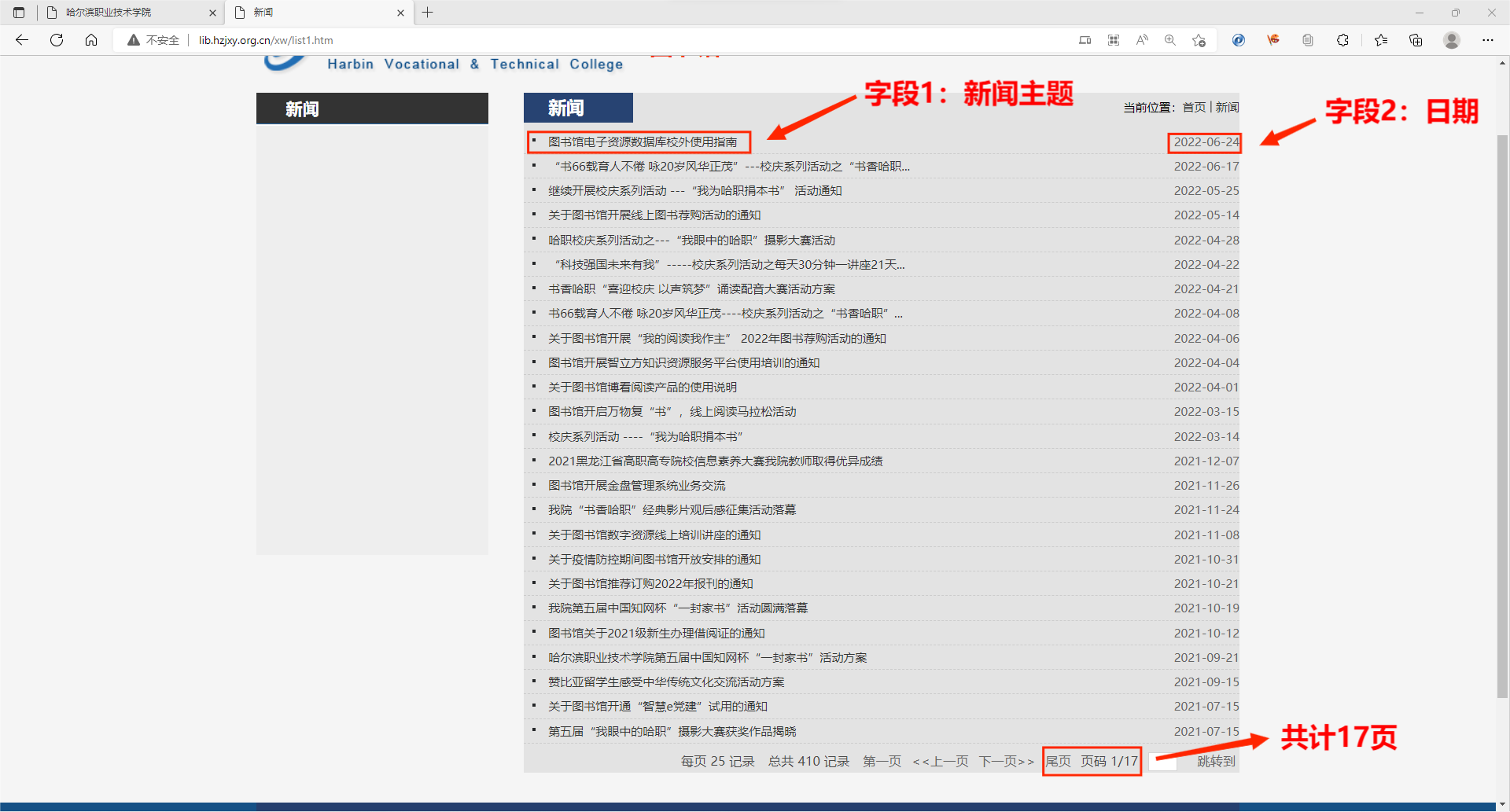


图1.10 指定字段

代码如下：

import requests

from lxml import etree

# 请求头部分

headers = {'User-Agent':''}

# 保存信息文件

fp = open('news.txt','a',encoding='utf-8')

# 待采集网址

base\_url = 'http://www.hzjxy.org.cn/'

# 数据采集部分

def htmls():

# 制作翻页

for num in range(1,18):

base\_url = 'http://lib.hzjxy.org.cn/xw/list{}.htm'.format(num)

# 发送请求并获取响应

response = requests.get(url=base\_url,headers=headers)

# 解码

response.encoding = 'utf-8'

# 获取源码

html = response.text

clean(html)

# 数据解析

def clean(html):

# 解析html

htmls = etree.HTML(html)

# 新闻标题

title = htmls.xpath('//div[@id="wp\_news\_w6"]/ul/li/div/span/a/@title')

# 日期

times = htmls.xpath('//div[@id="wp\_news\_w6"]/ul/li/div/span[@class="Article\_PublishDate"]/text()')

# print(times)

saves(title,times)

# 信息保存

def saves(title,time):

# 遍历信息并保存

for tt,ti in zip(title,time):

# 信息格式化

full\_info = '新闻标题：{}\t日期：{}\n'.format(tt,ti)

# 信息写入

fp.write(full\_info)

if \_\_name\_\_ == '\_\_main\_\_':

htmls()

fp.close()

**19.5 小结**

本章主要介绍了爬虫的基本概念、环境搭建、通过Python标准库以及第三方模块进行源码采集和数据解析等内容。另外，还有一些其他的第三方模块，如PyQuery、Beautifulsoap等也可完成爬虫功能，读者可以自行查阅这些模块的安装方法和应用属性以创建爬虫脚本，不断提高网络数据提取的效率和准确性。