第 13 章　网络爬虫入门与应用

13.1 HTML 与 JavaScript 基础

- 如果只是编写爬虫程序的话，毕竟不是开发网站，所以只要能够看懂 HTML 代码基本上就可以了，不要求能编写。
- 当然，对于一些高级爬虫和特殊的网站，还需要具有深厚的 JavaScript 功底，或者 JQuery 、 AJAX 等知识。

13.1.1 HTML 基础

（ 1 ） h 标签

在 HTML 代码中，使用 h1 到 h6 表示不同级别的标题，其中 h1 级别的标题字体最大， h6 级别的标题字体最小。该标签的用法为：

<h1> 一级标题 </h1>

<h2> 二级标题 </h2>

<h3> 三级标题 </h3>

13.1.1 HTML 基础

（ 2 ） p 标签

在 HTML 代码中， p 标签表示段落，用法为：

 这是一个段落

13.1.1 HTML 基础

（ 3 ） a 标签

在 HTML 代码中， a 标签表示超链接，使用时需要指定链接地址（由 href 属性来指定）和在页面上显示的文本，用法为：

```
<a href="http://www.baidu.com"> 点这里 </a>
```

13.1.1 HTML 基础

（ 4 ）img 标签

在 HTML 代码中， img 标签用来显示一个图像，并使用 src 属性指定图像文件地址，可以使用本地文件，也可以指定网络上的图片。例如：

```
<img src="Python 可以这样学 .jpg" width="200" height="300" />

<img src="http://www.tup.tsinghua.edu.cn/upload/bigbookimg/072406-01.jpg"
width="200" height="300" />
```

13.1.1 HTML 基础

（ 5 ）table 、 tr 、 td 标签

在 HTML 代码中， table 标签用来创建表格， tr 用来创建行， td 用来创建单元格，用法为：

```
<table border="1">
    <tr>
        <td> 第一行第一列 </td>
        <td> 第一行第二列 </td>
    </tr>
    <tr>
        <td> 第二行第一列 </td>
```

工于建构 成于创造

13.1.1 HTML 基础

（ 6 ）ul 、 ol 、 li

在 HTML 代码中， ul 标签用来创建无序列表， ol 标签用来创建有序列表， li 标签用来创建其中的列表项。例如，下面是 ul 和 li 标签的用法：

```
<ul id="colors" name="myColor">
    <li> 红色 </li>
    <li> 绿色 </li>
    <li> 蓝色 </li>
</ul>
```

13.1.1 HTML 基础

（ 7 ） div 标签

在 HTML 代码中， div 标签用来创建一个块，其中可以包含其他标签，例如：

```
<div id="yellowDiv" style="background-color:yellow;border:#FF0000 1px solid;">
    <ol>
        <li> 红色 </li>
        <li> 绿色 </li>
        <li> 蓝色 </li>
    </ol>
</div>
<div id="reddiv" style="background-color:red">
    <p> 第一段 </p>
```

13.1.2 JavaScript 基础

（ 1 ）在网页中使用 JavaScript 代码的方式

- 可以在 HTML 标签的事件属性中直接添加 JavaScript 代码。例如，把下面的代码保存为 index.html 文件并使用浏览器打开，单击按钮“保存”，网页会弹出提示“保存成功”。

```
<html>
    <body>
        <form>
            <input type="button" value=" 保存 " onClick="alert(' 保存成功 ');">
        </form>
    </body>
</html>
```

13.1.2 JavaScript 基础

- 对于较多但仅在个别网页中用到的 JavaScript 代码，可以写在网页中的 <script> 标签中。例如，下面的代码保存为 index.html 并使用浏览器打开，会发现页面上显示的是“动态内容”而不是“静态内容”。

```
<html>
    <body>
        <div id="test">静态内容</div>
    </body>
    <script type="text/javascript">
        document.getElementById("test").innerHTML=" 动态内容 ";
    </script>
</html>
```

13.1.2 JavaScript 基础

- 如果一个网站中会用到大量的 JavaScript 代码，一般会把这些代码按功能划分到不同函数中，并把这些函数封装到一个扩展名为 js 的文件中，然后在网页中使用。例如，和网页在同一个文件夹中的 myfunctions.js 内容如下：

```
function modify(){
    document.getElementById("test").innerHTML=" 动态内容 ";
}
```

- 在下面的页面文件中，把外部文件 myfunctions.js 导入，然后调用了其中的函数：

```
<html>
<head>
    <script type="text/javascript" src="myfunctions.js"></script>
</head>
```

13.1.2 JavaScript 基础

（ 2 ）常用 JavaScript 事件

- 把下面的代码保存为 index.html 并使用浏览器打开，会发现在每次页面加载时都会弹出提示，但在页面上进行其他操作时，并不会弹出提示。

```
<html>
    <body onLoad="alert('页面开始加载');">
        <div id="test">静态内容</div>
    </body>
</html>
```

13.1.2 JavaScript 基础

- 除了常用的事件之外，还有一些特殊的方式可以执行 JavaScript 代码。例如，下面的代码演示了在链接标签 <a> 中使用 href 属性指定 JavaScript 代码的用法。

```
<html>
    <script type="text/javascript">
        function test(){alert(' 提示信息 ');}
    </script>
    <body>
        <a href="javascript:test();"> 点这里 </a>
    </body>
</html>
```

13.1.2 JavaScript 基础

（ 3 ）常用 JavaScript 对象

- 下面的代码演示了 prompt() 方法的用法，将其保存为文件 index.html 并使用浏览器打开，会提示用户输入任意内容，然后在页面上输出相应的信息。

```
<html>
    <script type="text/javascript">
        var city = prompt(" 请输入一个城市名称： ", " 烟台 ");
        document.write(" 你输入的是： "+city);
    </script>
    <body></body>
</html>
```

13.1.2 JavaScript 基础

- 把下面的代码保存为文件 index.html ，此时页面上会显示图像文件 1.jpg 的内容，单击该图像时会切换成为 2.jpg 的内容。

```
<html>
    <body>
        <img name="img1" src="1.jpg"
             onClick="document.img1.src='2.jpg';" />
    </body>
</html>
```

13.2 urllib 基本应用与爬虫案例

- Python 3.x 标准库 urllib 提供了 urllib.request 、 urllib.response 、 urllib.parse 和 urllib.error 四个模块，很好地支持了网页内容读取功能。再结合 Python 字符串方法和正则表达式，可以完成一些简单的网页内容爬取工作，也是理解和使用其他爬虫库的基础。

13.2.1 urllib 的基本应用

1. 读取并显示网页内容

```
>>> import urllib.request
>>> fp = urllib.request.urlopen(r'http://www.python.org')
>>> print(fp.read(100))                    # 读取 100 个字节
>>> print(fp.read(100).decode())           # 使用 UTF8 进行解码
>>> fp.close()                             # 关闭连接
```

13.2.1 urllib 的基本应用

2. 提交网页参数

（ 1 ）下面的代码演示了如何使用 GET 方法读取并显示指定 url 的内容。

```
>>> import urllib.request
>>> import urllib.parse
>>> params = urllib.parse.urlencode({'spam': 1, 'eggs': 2, 'bacon': 0})
>>> url = "http://www.musi-cal.com/cgi-bin/query?%s" % params
>>> with urllib.request.urlopen(url) as f:
    print(f.read().decode('utf-8'))
```

13.2.1 urllib 的基本应用

（ 2 ）下面的代码演示了如何使用 POST 方法提交参数并读取指定页面内容。

```
>>> import urllib.request
>>> import urllib.parse
>>> data = urllib.parse.urlencode({'spam': 1, 'eggs': 2, 'bacon': 0})
>>> data = data.encode('ascii')
>>> with urllib.request.urlopen("http://requestb.in/xrbl82xr",
                                data) as f:
    print(f.read().decode('utf-8'))
```

13.2.1 urllib 的基本应用

3. 使用 HTTP 代理访问页面

```
>>> import urllib.request
>>> proxies = {'http': 'http://proxy.example.com:8080/'}
>>> opener = urllib.request.FancyURLopener(proxies)
>>> with opener.open("http://www.python.org") as f:
    f.read().decode('utf-8')
```

13.2.2 urllib 爬虫案例

- **例 13-1** 爬取公众号文章中的图片。

✓ 第 1 步 确定公众号文章的地址，以微信公众号“Python 小屋”里的一篇文章为例，文章标题为“报告 PPT（163 页）：基于 Python 语言的课程群建设探讨与实践”，地址为：

https://mp.weixin.qq.com/s?__biz=MzI4MzM2MDgyMQ==&mid=2247486249&idx=1&sn=a37d079f541b194970428fb2fd7a1ed4&chksm=eb8aa073dcfd2965f2d48c5ae9341a7f8a1c2ae2c79a68c7d2476d8573c91e1de2e237c98534&scene=21#wechat_redirect

13.2.2 urllib 爬虫案例

- 第 2 步 在浏览器（以 Chrome 为例）中打开该文章，然后单击鼠标右键，选择“查看网页源代码”，分析后发现，公众号文章中的图片链接格式为：

13.2.2 urllib 爬虫案例

- 第 3 步 根据前面的分析，确定用来提取文章中图片链接的正则表达式：

```
pattern = 'data-type="png" data-src="(.+?)"'
```

13.2.2 urllib 爬虫案例

- 第 4 步 编写并运行 Python 爬虫程序，代码如下：

```
from re import findall

from urllib.request import urlopen


url = 'https://mp.weixin.qq.com/s?__biz=MzI4MzM2MDgyMQ==&mid=2247486249&i
dx=1&sn=a37d079f541b194970428fb2fd7a1ed4&chksm=eb8aa073dcfd2965f2d48c5ae9
341a7f8a1c2ae2c79a68c7d2476d8573c91e1de2e237c98534&scene=21#wechat_redire
ct'

with urlopen(url) as fp:

    content = fp.read().decode()


pattern = 'data-type="png" data-src="(.+?)"'
# 查找所有图片链接地址
```

13.3 scrapy 爬虫案例

- **例 13-2** 使用 scrapy 框架编写爬虫程序。
- 第 1 步 使用 pip 命令安装好 scrapy 之后，在命令提示符环境中执行下面的命令创建一个项目 MyCraw ：

```
scrapy startproject MyCraw
```

13.3 scrapy 爬虫案例

- 第 2 步 然后编写 Python 程序 MyCraw\MyCraw\spiders\MySpider.py ，用于爬取指定页面的内容，把网页内容和图片分别保存为文件， MySpider.py 的代码如下：

code\MySpider.py

13.3 scrapy 爬虫案例

- 第 3 步 在命令提示符环境中执行下面的命令，运行爬虫程序。

```
scrapy crawl mySpider
```

13.3 scrapy 爬虫案例

- **例 13-3** 使用 scrapy 框架编写爬虫程序，爬取天涯小说。

- 第 1 步 以天涯小说“宜昌鬼事之大宗师”为例，首先确定第一页的链接为：

http://bbs.tianya.cn/post-16-1126849-1.shtml

- 第 2 步 然后查看并分析网页源代码，确定作者 ID ，确定如何查找作者发表的帖子而过滤其他跟帖，并确定该小说不同页的 URL 之间有什么规律。

13.3 scrapy 爬虫案例

- 第 3 步 把这些问题都确定之后，创建爬虫项目。

1 ）进入命令提示符（ cmd ）环境，切换至 Python 安装目录的 scripts 目录，执行命令 scrapy startproject xiaoshuo 创建爬虫项目 xiaoshuo ，

2) 进入该目录，编写 Python 程序文件 \spiders\spiderYichangGuishi.py ，

3 ）然后在命令提示符环境中执行命令 scrapy crawl spiderYichangGuishi ，稍等片刻即可爬取小说全文并生成记事本文档。

code\spiderYichangGuishi.py

13.4 BeautifulSoup 用法简介

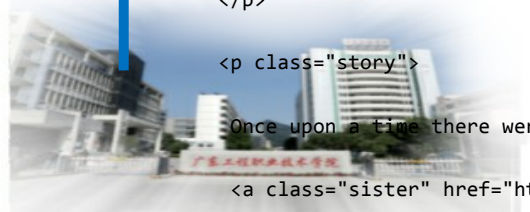
```
>>> from bs4 import BeautifulSoup
>>> BeautifulSoup('hello world!', 'lxml')       # 自动添加标签
<html><body><p>hello world!</p></body></html>

>>> BeautifulSoup('<span>hello world!', 'lxml') # 自动补全标签
<html><body><span>hello world!</span></body></html>
```

13.4 BeautifulSoup 用法简介

```
>>> html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters; and their names were
<a href="http://example.com/elsie" class="sister" id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister" id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
```

13.4 BeautifulSoup 用法简介

```
>>> soup = BeautifulSoup(html_doc, 'html.parser') # 也可以指定 lxml 或其他解析器

>>> print(soup.prettify())                          # 以优雅的方式显示出来
<html>
 <head>
  <title>
   The Dormouse's story
  </title>
 </head>
 <body>
  <p class="title">
   <b>
    The Dormouse's story
   </b>
  </p>
  <p class="story">
   Once upon a time there were three little sisters; and their names were
   <a class="sister" href="http://example.com/elsie" id="link1">
```



13.4 BeautifulSoup 用法简介

```
>>> soup.title                                  # 访问 <title> 标签的内容
<title>The Dormouse's story</title>
>>> soup.title.name                             # 查看标签的名字
'title'
>>> soup.title.text                             # 查看标签的文本
"The Dormouse's story"
>>> soup.title.string                           # 查看标签的文本
"The Dormouse's story"
>>> soup.title.parent                           # 查看上一级标签
<head><title>The Dormouse's story</title></head>
>>> soup.head
<head><title>The Dormouse's story</title></head>
```

13.4 BeautifulSoup 用法简介

```
>>> soup.body                          # 查看 body 标签内容
<body>
<p class="title"><b>The Dormouse's story</b></p>
<p class="story">Once upon a time there were three little sisters; and their names were
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<p class="story">...</p>
</body>
```

13.4 BeautifulSoup 用法简介

```
>>> soup.p                                    # 查看段落信息
<p class="title"><b>The Dormouse's story</b></p>
>>> soup.p['class']                           # 查看标签属性
['title']
>>> soup.p.get('class')                       # 也可以这样查看标签属性
['title']
>>> soup.p.text                               # 查看段落文本
"The Dormouse's story"
>>> soup.p.contents                           # 查看段落内容
[<b>The Dormouse's story</b>]
>>> soup.a
<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>
```

13.4 BeautifulSoup 用法简介

```
>>> soup.find_all('a')                    # 查找所有 <a> 标签

[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>, <a class
="sister" href="http://example.com/lacie" id="link2">Lacie</a>, <a class="sister"
href="http://example.com/tillie" id="link3">Tillie</a>]

>>> soup.find_all(['a', 'b'])          # 同时查找 <a> 和 <b> 标签

[<b>The Dormouse's story</b>, <a class="sister" href="http://example.com/elsie" id
="link1">Elsie</a>, <a class="sister" href="http://example.com/lacie" id="link2">L
acie</a>, <a class="sister" href="http://example.com/tillie" id="link3">Tillie</a
>]
```

13.4 BeautifulSoup 用法简介

```
>>> import re
>>> soup.find_all(href=re.compile("elsie"))
                                        # 查找 href 包含特定关键字的标签
[<a class="sister" href="http://example.com/elsie" id="link1">Elsie</a>]
>>> soup.find(id='link3')               # 查找属性 id='link3' 的标签
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>
>>> soup.find_all('a', id='link3')     # 查找属性 'link3' 的 a 标签
[<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>]
>>> for link in soup.find_all('a'):
    print(link.text,':',link.get('href'))

Elsie : http://example.com/elsie
```

13.4 BeautifulSoup 用法简介

```
>>> print(soup.get_text())                    # 返回所有文本
The Dormouse's story

The Dormouse's story

Once upon a time there were three little sisters; and their names were

Elsie,

Lacie and

Tillie;

and they lived at the bottom of a well.

...

>>> soup.a['id'] = 'test_link1'               # 修改标签属性的值

>>> soup.a

<a class="sister" href="http://example.com/elsie" id="test_link1">Elsie</a>
```

13.4 BeautifulSoup 用法简介

```
>>> for child in soup.body.children:      # 遍历直接子标签
    print(child)


<p class="title"><b>The Dormouse's story</b></p>
<p class="story">Once upon a time there were three little sisters; and their names were
<a class="sister" href="http://example.com/elsie" id="test_link1">test_Elsie</a>,
<a class="sister" href="http://example.com/lacie" id="link2">Lacie</a> and
<a class="sister" href="http://example.com/tillie" id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>
<p class="story">...</p>
```

13.4 BeautifulSoup 用法简介

```
>>> test_doc = '<html><head></head><body><p></p><p></p></body></heml>'
>>> s = BeautifulSoup(test_doc, 'lxml')
>>> for child in s.html.children:          # 遍历直接子标签
    print(child)

<head></head>
<body><p></p><p></p></body>
>>> for child in s.html.descendants:       # 遍历子孙标签
    print(child)

<head></head>
<body><p></p><p></p></body>
```

13.5 requests 基本操作与爬虫案例

- Python 扩展库 requests 可以使用比标准库 urllib 更简洁的形式来处理 HTTP 协议和解析网页内容，也是比较常用的爬虫工具之一，完美支持 Python 3.x，使用 pip 可以直接在线安装。
- 安装成功之后，使用下面的方式导入这个库：

```
>>> import requests
```

13.5.1 requests 基本操作

（1）增加头部并设置访问代理

```
>>> url = 'https://api.github.com/some/endpoint'
>>> headers = {'user-agent': 'my-app/0.0.1'}
>>> r = requests.get(url, headers=headers)
```

13.5.1 requests 基本操作

（ 2 ）访问网页并提交数据

```
>>> payload = {'key1': 'value1', 'key2': 'value2'}
>>> r = requests.post("http://httpbin.org/post", data=payload)
>>> print(r.text)          # 查看网页信息，略去输出结果
>>> url = 'https://api.github.com/some/endpoint'
>>> payload = {'some': 'data'}
>>> r = requests.post(url, json=payload)
>>> print(r.text)          # 查看网页信息，略去输出结果
>>> print(r.headers)       # 查看头部信息，略去输出结果
>>> print(r.headers['Content-Type'])
```

13.5.1 requests 基本操作

（ 3 ）获取和设置 cookies

- 下面的代码演示了使用 get() 方法获取网页信息时 cookies 属性的用法：

```
>>> r = requests.get("http://www.baidu.com/")
>>> r.cookies                    # 查看 cookies
<RequestsCookieJar[Cookie(version=0, name='BDORZ', value='27315', port=No
ne, port_specified=False, domain='.baidu.com', domain_specified=True, dom
ain_initial_dot=True, path='/', path_specified=True, secure=False, expire
s=1521533127, discard=False, comment=None, comment_url=None, rest={}, rfc
2109=False)]>
```

13.5.1 requests 基本操作

- 下面的代码演示了使用 get() 方法获取网页信息时设置 cookies 参数的用法：

```
>>> url = 'http://httpbin.org/cookies'
>>> cookies = dict(cookies_are='working')
>>> r = requests.get(url, cookies=cookies)  # 设置 cookies
>>> print(r.text)
{
  "cookies": {
    "cookies_are": "working"
  }
}
```

13.5.2 requests 爬虫案例

- **例 13-4** 使用 requests 库爬取微信公众号“Python 小屋”文章“Python 使用集合实现素数筛选法”中的所有超链接。

13.5.2 requests 爬虫案例

```
>>> import requests

>>> url = 'https://mp.weixin.qq.com/s?__biz=MzI4MzM2MDgyMQ==&mid=2247486531&idx=1&
sn=7eeb27a03e2ee8ab4152563bb110f248&chksm=eb8aa719dcfd2e0f7b1731cfd8aa74114d68facf
1809d7cdb0601e3d3be8fb287cfc035002c6#rd'

>>> r = requests.get(url)

>>> r.status_code        # 响应状态码

200

>>> r.text[:300]         # 查看网页源代码前 300 个字符

'<!DOCTYPE html>\n<!--headTrap<body></body><head></head><html></html>--><html>\n
 <head>\n        <meta http-equiv="Content-Type" content="text/html; charset=utf-
8">\n<meta http-equiv="X-UA-Compatible" content="IE=edge">\n<meta name="viewport"
content="width=device-width,initial-scale=1.0,maximum-scale='

>>> '筛选法' in r.text   # 测试网页源代码中是否包含字符串'筛选法'

True
```

13.5.2 requests 爬虫案例

```
>>> links = re.findall(r'<a .*?href="(.+?)"', r.text)
                                    # 使用正则表达式查找所有超链接地址
>>> for link in links:
    if link.startswith('http'):
        print(link)
```

13.5.2 requests 爬虫案例

```
>>> from bs4 import BeautifulSoup
>>> soup = BeautifulSoup(r.content, 'lxml')
>>> for link in soup.findAll('a'):  # 使用 BeautifulSoup 查找超链接地址
    href = link.get('href')
    if href.startswith('http'):      # 只输出绝对地址
        print(href)
```

13.5.2 requests 爬虫案例

- **例 13-5** 读取并下载指定的 URL 的图片文件。

```
>>> import requests
>>> picUrl = r'https://www.python.org/static/opengraph-icon-200x200.png'
>>> r = requests.get(picUrl)
>>> r.status_code
200
>>> with open('pic.png', 'wb') as fp:
    fp.write(r.content)                    # 把图像数据写入本地文件
```

13.6 selenium 爬虫案例

- **例 13-6** 使用 selenium 编写爬虫程序，获取指定城市的当前天气。

✓ 第 1 步 首先，查看一下本地计算机 Windows 操作系统的内部版本号，以我的 Win10 为例，步骤为：依次单击开始 ==> 设置 ==> 系统 ==> 关于，找到下图中的操作系统内部版本号：

13.6 selenium 爬虫案例

- 第 2 步 打开网址 https://developer.microsoft.com/en-us/microsoft-edge/tools/webdriver/ ，下载合适版本的驱动，并放到 Python 安装目录下。
- 第 3 步 打开命令提示符环境，使用 pip 安装扩展库 selenium 。
- 第 4 步 编写如下程序代码：

```
import re
from selenium import webdriver


# 指定引擎
driver = webdriver.Edge()
city = input('请输入要查询的城市：').lower()
# 获取指定 URL 的信息，并进行渲染
driver.get(r'http://openweathermap.org/find?q={0}'.format(city))
```