第5章

动态页面采集技术与Python实现

5.1 动态页面内容的生成与交互

5.1.1 页面内容的生成方式

动态页面区别于静态页面的最主要特征是页面内容的生成方式，动态页面的内容的生成方式可以分成两类，即服务器端生成、客户端生成。

1. 服务器端生成

采用这种方式进行管理的内容包括新闻信息、通知、广告等，它是 Web 页面内容生成的主要途径。基于这种方式可以很方便地进行内容管理，目前市场上有很多 Web 内容管理系统（Content Management System，CMS）就提供了在后台进行 Web 内容管理的平台。通过这种方式可以方便地应对 Web 内容繁杂、制作发布、扩展性等方面的问题。

在这种内容生成方式中，页面的主要内容与页面的结构和表现方式一般是分离的。页面主要内容可以存储在各种数据库系统中，而决定结构和表现方式的 HTML

标签和语句则是存储在 Web 服务器上，因此在应用架构上采用的是 Client/Server/Database 模式。在 Web 页面中使用脚本语言连接数据库、向数据库发起查询请求、对查询结果进行格式化等，最终生成包含具体内容的页面，并推送给客户端。

在 Web 页面中经常使用的脚本语言有 JSP、ASP、PHP 等，使用这些语言连接数据库、查询数据库、生成给用户的 HTML 文档。一个简单的例子是用户登录，用户访问 login.htm，在输入用户名和密码之后，将这些信息提交给 results.jsp，在该文件中进行脚本的动态执行，从而完成上述过程。

1）login.htm

```
<%@ page contentType="text/html;charset=gb2312" %>
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>登录</title>
    </head>
    <body>
            <form id="loginForm" name="login" action="results.jsp" method="get"
target="_blank">
                <p>用户名</p>
                  <input type="text" name="username" size="44">
                <p>密码</p>
                  <input type="text" name="password" size="44">
                  <input type="submit" value="提交" />
            </form>
    </body>
</html>
```

2）results.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import = "java.sql.ResultSet,java.sql.Statement,java.sql.SQLException,java.
sql.Connection,java.sql.DriverManager,javax.servlet.*, javax.servlet.http.*, java.io.*,
java.net.URLEncoder" %>
<%
        String username, passwd;
        username = request.getParameter("username");
        passwd = request.getParameter("password");
    String drivername="com.microsoft.jdbc.sqlserver.SQLServerDriver";
    String uname="sqlinjection";
    String userpassword="sql";
    String dbname="sqlinject";
    String url = "jdbc:microsoft:sqlserver://127.0.0.1:1433;DatabaseName=sql";
    Class.forName(drivername).newInstance();
```

```
    Connection conn = DriverManager.getConnection(url,"sqluser", "sql1234");
    Statement stmt = conn.createStatement();
    ResultSet rs = stmt.executeQuery (""select * from users where username = '" + username + "'
and passwd = " + passwd");
  if(rs.next()){
      out.print("<center><h3>登录成功!</h3></center>");
  } else {
      out.print("<center><h3>登录不成功!</h3></center>");
  }
    stmt.close();
    conn.close();
    %>
```

为了执行这些脚本代码，在数据库中存储有 users 表，该表包含两个字段，即 username 和 passwd。这样可以在后台配置 users 表中的记录，从而来决定返回给浏览器的信息。

另一种在服务器上进行内容生成的途径是在 HTML 文档中嵌入 SSI(Server Side Include)指令。包含这种指令的文件的默认扩展名是.stm、.shtm 或.shtml，这样当客户端访问这类文件时，Web 服务器端会对这些文件进行读取和解析，把文件中包含的 SSI 指令解释出来，最终生成 HTML 文档推送给客户端。与内容生成有关的常见指令是 include，使用方法是在 HTML 文档中的合适位置插入命令：

```
<!-- #include file = "文件名" -->
```

其中，文件名是一个用相对路径表示的文件，例如<!-- #include file="head_news.htm"-->将 head_news.htm 文件的内容插入到当前页面。

这种通过指令嵌入来生成动态页面的典型场合是当不同 Web 页面包含某些相同的内容时，将这些内容定义为一个独立的文件，然后嵌入到其他 Web 页面中。例如，在新浪新闻页面中都有一个如图 5-1 所示的导航条，就可以将这些内容定义为一个独立的 HTML 文档。这样可以省去在不同新闻页面中编写导航条的麻烦。

图 5-1　导航条

2. 客户端生成

根据这种内容生成方式，内容是在客户端上生成，而客户端主要是浏览器。受限

于浏览器的能力，客户端生成的内容一般是轻量级的、局部的，例如给用户提示警告信息、显示定时时间等。

在这种生成方式下，Web 页面中需要嵌入一定的脚本或插件。常用的脚本语言包括 JavaScript、VBScript、ActionScript 等，插件包括 Active X 控件、Flash 插件等。这些脚本或插件具备对浏览器事件做出响应、读写 HTML 中的元素、创建或修改 Cookie 等功能，这些功能的实现要求客户端具有执行脚本、下载并执行插件的能力。通过在浏览器内执行这些脚本或插件功能，实现 Web 页面内容的生成，并进行动态更新。

由于需要在浏览器的进程空间中执行代码，所以这种方式可能会影响客户端安全。许多恶意的脚本代码就通过这种途径对本地计算机产生安全风险，所以浏览器一般都会提供让用户进行自行配置的界面，如图 5-2 所示。

图 5-2　浏览器执行脚本的配置

5.1.2　动态页面交互的实现

动态页面的交互是指浏览器和 Web 服务器之间的命令参数传递方式，按照命令参数的不同提供方式，主要有用户提供和 Cookie 提供两种方式。不管是哪种方式，通常使用 JavaScript 等脚本语言来生成带参数的 URL，最终发送给服务器。URL 的发送有通过 Ajax 引擎和非 Ajax 引擎两种。下面分别介绍这里提到的一些关键技术。

1. 通过 URL 传递请求参数

URL 即统一资源定位，用来表示资源在互联网上的地址。URL 具有以下的形式：

```
协议://域名部分:端口号/目录/文件名.文件后缀?参数1=值#标志&参数2=值#标志
```

其中，对于 Web 页面请求而言，协议可以是 http 或 https。如果使用默认的 80 作为端口号，则可以省略。? 表示第一个参数的开始，起到分隔的作用。URL 可以携带多个参数及其值，在上面给出的形式中包含了两个参数。基本形式是"参数＝值"，不同参数之间用 & 连接起来。# 标志表示书签，用于访问一个 Web 页面中的特定部分。带参数的 URL 的两个例子如下：

```
https://search.jd.com/Search?keyword=互联网大数据&enc=utf-8
https://www.baidu.com/s?ie=utf-8&f=8&rsv_bp=0&rsv_idx=1&tn=baidu&wd=bigdata
```

在用户参与的情况下，生成动态页面所需要的参数一般是通过输入框、下拉列表框、按钮等基本的 HTML 控件来完成。生成 URL 则是浏览器在控件的相关事件中调用 JavaScript 等语言编写的脚本来完成的。

图 5-3 所示为京东的搜索页面部分，它包含了一个输入框和一个按钮，相应的 HTML 源代码如下：

```
<div class="form">
    <input type="text" onkeydown="javascript:if(event.keyCode==13) search('key');"
autocomplete="off" id="key" accesskey="s" class="text" />
    <button onclick="search('key');return false;" class="button cw-icon"><i></i>搜
索</button>
</div>
```

图 5-3　输入搜索

当按下回车键或单击按钮时，浏览器执行相应的脚本，最终生成包含参数的 URL，具有如下形式：

```
https://search.jd.com/Search?keyword=互联网大数据&enc=utf-8
```

表示搜索的关键词是“互联网大数据”，编码方式是 utf-8。

2. 通过 Cookie 获取命令参数

在 Cookie 中记录了一些客户端和服务器之间交互的参数，例如购物网站上用户设定的城市、登录用户名和口令等，这样对于需要用户登录的页面，就可以自动读取 Cookie 内容作为请求的参数。具体过程是这样的，在浏览器的地址栏中输入 URL，浏览器根据域名及作用范围来寻找相应的 Cookie 文件，如果文件存在，则将该 Cookie 中的内容读出，并填充在 HTTP 请求头上发送给服务器。

图 5-4 所示为在 Chrome 浏览器的开发者视图下看到的访问淘宝网时的 HTTP 请求头，可以看出 Cookie 是以“Cookie：name1＝value1；name2＝value2；...”的形式传递参数给服务器的。

图 5-4　访问淘宝网时的 HTTP 请求头

3. Ajax

Ajax 是一种基于 JavaScript 并整合 XHTML、XML、DOM 等技术实现的客户端/服务器端动态页面编程框架。1999 年微软公司发布 IE5，引入一个新功能，即允许 JavaScript 脚本向服务器发起 HTTP 请求，这个技术在后来被称为 Ajax，其全称是 Asynchronous JavaScript and XML。

支持 Ajax 的浏览器配置有 Ajax 引擎，Ajax 通过 XMLHttpRequest 和 Web 服务器进行异步通信，利用 iframe 技术实现按需获取数据。Ajax 通常用于在后台与服务器进行少量数据交换，在不重新加载整个网页的情况下对网页的局部进行更新，最大程度地减少冗余请求，避免给服务器造成太大的负担，也更有助于提高用户体验。

Ajax 这种类型的动态页面机制通常是在一定条件下触发的，归纳起来主要有以下两种途径。

1）页面中的定时器

Web 页面按照一定的时间间隔自动执行脚本，从而发起 Ajax 请求。典型的页面例子是包含股票实时行情数据的 Web 页面。图 5-5 所示为新浪股票行情页面，页面上的当前价格等信息按照一定的时间自动更新。

图 5-5　Web 页面中的实时内容显示

2）鼠标或键盘事件驱动

Web 页面中要显示的内容比较多，但是为了用户体验，一般事先只显示部分内容，随着用户阅读的进行，不断滚动鼠标或按 PgDn 键自动获取更多内容并显示。在这个过程中获取更多内容的方式，通常也是 Ajax 一种请求。在各种新闻网站，例如新浪新闻中，一个典型的页面可参见“http://news.sina.com.cn/o/2018-11-06/doc-ihmutuea7351575.shtml”，当用户在浏览器上向下翻页时会不断出现更多的新闻列表。另一个典型的例子是各种电商网站上的商品评论，通常事先显示评论的少数记录，之后随着用户不断按 PgDn 键或向下滚动鼠标，会有更多的评论记录显示出来。

5.2　动态页面采集技术

根据 5.1 节中关于动态网页内容生成与交互方式的叙述，动态网页的采集技术也与交互方式直接相关。采集技术可以归纳为以下 4 种类型。

(1) 构造带参数的 URL,利用参数传递动态请求;

(2) 构造 Cookie 携带参数,利用 HTTP 头部传递动态请求的参数;

(3) 离线分析 Ajax 的动态请求,使用静态页面采集技术或者通过 Cookie、POST 等形式发送请求参数;

(4) 模拟浏览器技术。

在这 4 种方式中最简单的是第一种,只要拼接出正确的 URL 即可。当页面本身使用 Cookie 传递参数时,爬虫就可以通过 Cookie 发送请求参数。Ajax 动态请求技术是当前使用最广泛的,其基本步骤包括分析页面、获取 URL、获取动态请求参数以及传递参数。模拟浏览器技术实际上是调用浏览器内核来模拟真实用户的点击和按键等操作,因此在执行效率上要比其他方式低得多,但是能够执行一些比较复杂的脚本,模拟复杂的交互操作。

这 4 种技术手段适用于不同的动态页面交互方式,在进行爬虫程序设计时应当根据页面的具体情况选择。

5.3 使用带参数的 URL

带参数的 URL 具有以下形式:

```
协议://域名部分:端口号/目录/文件名.文件后缀?参数1=值#标志&参数2=值#标志
```

其中,参数 1、参数 2 等多个参数通过 & 连接,参数名称可以通过网页的 HTML 源代码获得,也可以直接观察页面请求时的 URL 获得。这种方法比较简单、直接,但是需要注意参数值的编码方式。

图 5-6 所示为京东的搜索页面部分,它包含了一个输入框和一个按钮,上面是生成的 URL。

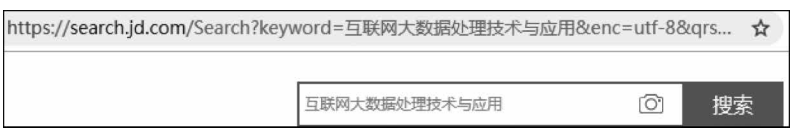

图 5-6 搜索与对应的 URL

"https://search.jd.com/Search?keyword=互联网大数据处理技术与应用&enc=utf-8"表示搜索的关键词是"互联网大数据处理技术与应用",编码方式是 utf-8。

因此，爬虫在采集类似动态页面时就可以直接填写关键词，构成完整的带参数的URL，然后发送给 Web 服务器。

然而，参数在 URL 中可能是经过编码的，例如同样的搜索命令在当当网中显示查询的关键词是经过编码后的，如图 5-7 所示。

图 5-7　经过编码的带参数 URL

在 2.2 节中对网页字符编码进行了归纳，主要有 utf-8、gbk、gb2312、unicode，这些编码方式同样适用于 URL 的参数。字符串编码的转换方式见 2.2 节所述。通过以下简单的编码测试，可以发现图 5-7 所示的例子对 URL 的参数采用了 gbk 编码。

```
>>> n = '互联网大数据处理技术与应用'      # unicode
>>> n.encode('gbk')
b'\xbb\xa5\xc1\xaa\xcd\xf8\xb4\xf3\xca\xfd\xbe\xdd\xb4\xa6\xc0\xed\xbc\xbc\xca\xf5\xd3
\xeb\xd3\xa6\xd3\xc3'
>>>
```

综合上述内容来看，这种动态页面采集技术比较简单，关键在于构建合适的URL。以下是一个例子，将 URL 的参数存放在字典中，则可以通过字典和字符串的相关运算符来拼接带参数的 URL。

```
url = 'https://search.jd.com/Search'
#以字典存储查询的关键词及属性
qrydata = {
    'keyword':'互联网大数据',
    'enc':'utf-8',
}
lt = []
for k,v in qrydata.items():
    lt.append(k + '=' + str(v))
query_string = '&'.join(lt)

url = url + '?' + query_string
print(url)
```

这段程序最终生成的 URL 是“https://search.jd.com/Search? keyword=互联网大数据 &enc=utf-8”。

5.4 利用 Cookie 和 Session

在第 3 章对 Cookie 的一些基本知识及其在 HTTP 协议中的使用方法进行了介绍，在这里要掌握如何在爬虫中实现基于 Cookie 的动态交互过程。该过程分为两个环节，一是 Cookie 的获得(或构造)，二是将 Cookie 传递到服务器。

由于 Cookie 中的内容通常是由服务器生成的，客户端只是简单地将内容存储到本地。因此，在一般情况下需要通过合适的手段获得 Cookie 内容。最简单、准确的方式是通过浏览器的开发者工具或开发者模式，例如在 Google 浏览器中可以跟踪到访问 thelion.com 时的 HTTP 请求信息，能够看出该网站使用了 Cookie，如图 5-8 所示。只要将 Cookie 的属性值复制出来，保存成文本文件即可，例如 thelion.txt。

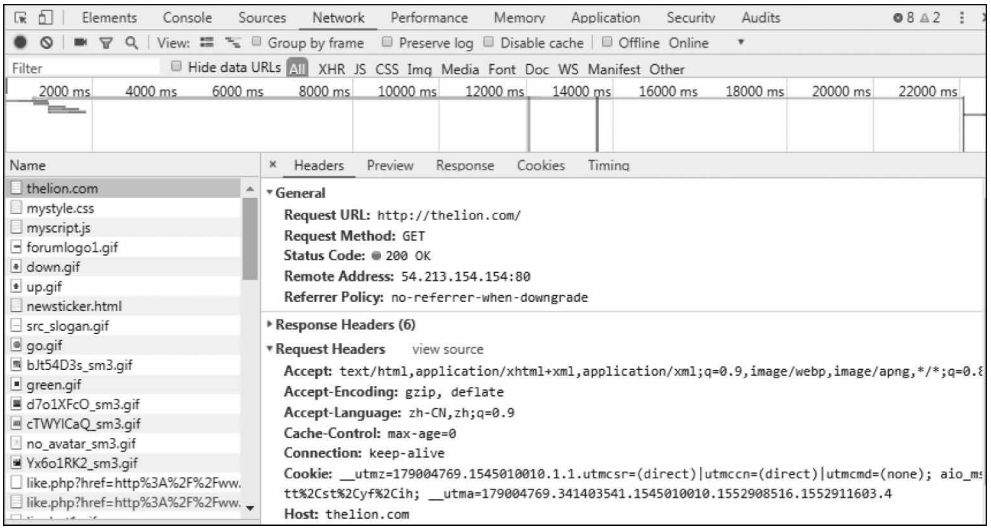

图 5-8 通过浏览器获得 Cookie 内容

从图中可以看出 Cookie 是以“Cookie：name1=value1；name2=value2；...”的形式存在的，这里指出了所传递的参数及参数值。因此在 Python 中可以将 Cookie 的内容以字典的形式存储。由于 Cookie 的传递是通过 HTTP 协议的请求头，所以需要在 requests.get()中指定 cookies 的属性值。以下是完整的例子。

```
Prog-4-cookie-demo.py
#Python 3.6 环境
import requests

f = open(r'taobao-hk.txt')                    #打开所保存的 cookies 内容文件
cookies = {}                                  #初始化 cookies 字典变量
for line in f.read().split(';'):
#按照字符进行划分读取.若将其设置为 1,就会把字符串拆分成两份
     name,value = line.strip().split('=',1)
     cookies[name] = value                    #为字典 cookies 添加内容

r = requests.get("https://www.taobao.com/",cookies = cookies) #cookies 中的内容作为参数
```

如 3.5.2 节所述，Cookie 存在被截获而导致个人敏感信息泄露的风险，此外连续多次访问同一台 Web 服务器时，反复申请与释放资源容易造成时间消耗。为了避免这些问题，在爬虫请求时可以使用 Session 技术。在 Python 中使用 Session 的方法是通过 requests 的 Session 类，以下例子展示了使用 Session 进行页面采集的方法。

```
import requests
def bySession(URL):
     sess = requests.Session()                #获得 Session 对象
     response = sess.get(URL)                 #使用 get、post 等方法提交请求
     #后续可以使用 response 对象的各种方法获得响应信息
```

但是也要注意，Session 是由服务器创建，Session 的长时间存在容易增加服务器的存储成本。因此，在设计友好的爬虫时，需要在信息泄露风险、时间消耗和服务器的存储成本之间综合权衡。

5.5　使用 Ajax：以评论型页面为例

Web 页面可以使用 JavaScript 等脚本语言生成带参数的 URL，并最终可能通过 Ajax 引擎发送到服务器上执行，因此可以获得最终发给服务器的 URL 作为爬虫的爬行任务。在这种动态页面访问方式中，最重要的关键技术问题是要寻找到 Ajax 动态加载的请求 URL 地址。此外，如果在向 Web 服务器发送请求，需要携带参数，则可以采取 5.3 节介绍的做法，也有的网页可能要求通过 POST 方式动态提交参数。

同样，上述问题都可以采用最常用的方法，即通过浏览器的开发者工具或开发者模式来检测请求信息。

视频讲解

5.5.1 获取 URL 地址

很多实时性比较强的网站都采用 Ajax 进行内容的动态加载，例如提供实时天气、股票行情等的网站。这类信息的自动获取就需要寻找相应的 URL，以新浪财经的股票实时信息为例，其网址是“https://finance.sina.com.cn/stock/”。在浏览器中输入该网址，进入浏览器的开发者模式。Google Chrome 浏览器提供的分析功能比较合适，在开发者模式下选择 Network、JS，如图 5-9 所示，即可看到页面中加载的 JS。

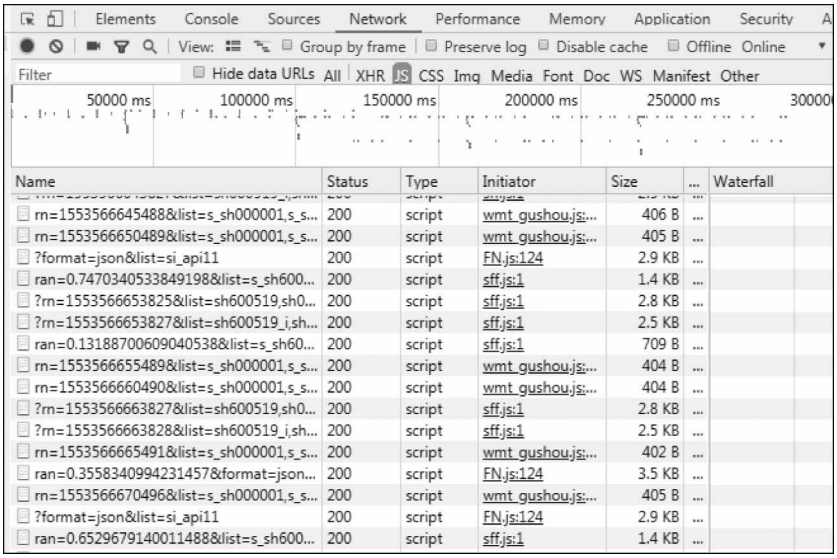

图 5-9 在开发者模式下查看 JS

如果 JS 是定时重复执行的，可以结合页面数据的更新情况寻找对应的 JS，单击之后，即可看到关于该 JS 发送 URL 的请求头和响应信息，如图 5-10 所示。图中 Request URL 对应的值即为请求数据的 URL。然后直接使用程序模拟请求，就可以从接口获取到返回的数据，一般情况下 Ajax 返回的数据是以 JSON 形式封装的，这样信息提取过程就会比较容易。

第二种获得 URL 地址的方法是通过请求头的 Referer 属性，3.4.3 节中提到，在请求头中该属性表示所请求的 URL 是哪个页面中的链接。当网站信息结构或交互

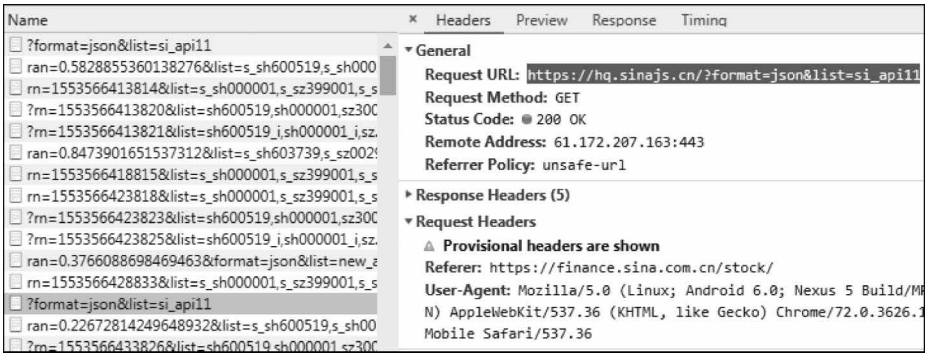

图 5-10　查看 JS 发送的请求信息

过程比较复杂时，通过第一种方法不一定能获得 Request URL，这时可以分析请求头的 Referer 属性。下面以携程酒店评论信息页面的请求为例来说明。

图 5-11 所示为酒店评论的入口页面，页面上提示有 600 条评论，当将鼠标指针移动到“酒店点评”处时可以看出，其超链接指向了“javascript:void(0)”。在其他很多网站的动态页面中都有类似的链接，显然爬虫通过这个链接无法获得真正评论的 URL。

图 5-11　酒店评论的入口页面

如果要获得这些评论，首先要找到请求的 URL。在进入开发者模式之后，通过鼠标点击操作可以在评论信息页面检查对应的请求过程。如图 5-12 所示，通过 Network 下的 XHR 选项可以在请求头的 Referer 属性中找到评论的 URL，即图中选中的部分。这里的 XHR 类型是指通过 XMLHttpRequest 方法发送的请求，可以在这里检查页面中发送的请求。

但是当遇到加密的 JS 时，要分析并找到请求地址就会非常困难，需要耐心寻找

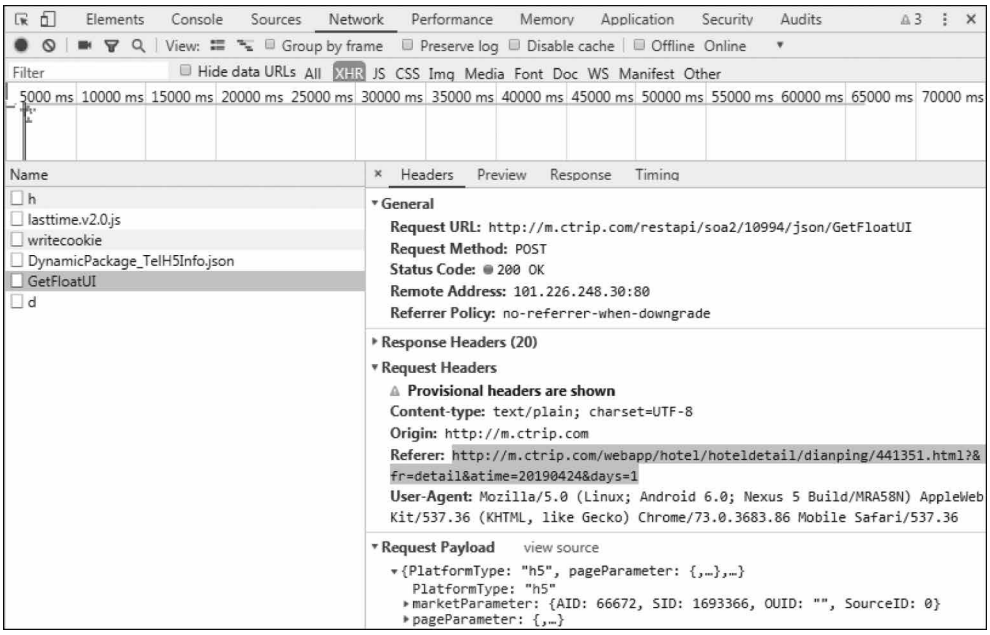

图 5-12　通过 Referer 属性寻找所需要的 URL

页面特征，以及在不同页面交互的过程中寻找动态请求之间的关系，这样就有可能获得最终发送给 Web 服务器的请求信息。

5.5.2　获取动态请求参数

一般情况下，Ajax 的动态请求使用带参数的 URL，这时可以直接使用前面提到的方法来构造 URL。页面还可以通过提交(POST)数据的方式向服务器发送请求的动态参数，在携程、亚马逊等许多存在用户评论的网站上广泛使用这种技术。在这种方式下，最终表现出来的 URL 并不是以“https://search.jd.com/Search?keyword=互联网大数据&enc=utf-8”这种通过关键字-值对表示的 URL，而是通过 POST 的方式提交数据。因此，只要能获得 POST 的数据及形式就可以向服务器发送请求了。

图 5-13 是携程酒店评论页面在浏览器开发者模式下的结果，可以看出在请求时采用了 Payload 的参数传递方式，这是 Ajax 的一种典型方式，在许多类似的动态页面中都存在。从图中可以看出请求的 Content-Type 是 application/json，具体参数在 Payload 中。

因此，只要将这些参数复制出来写到程序中。例子如下：

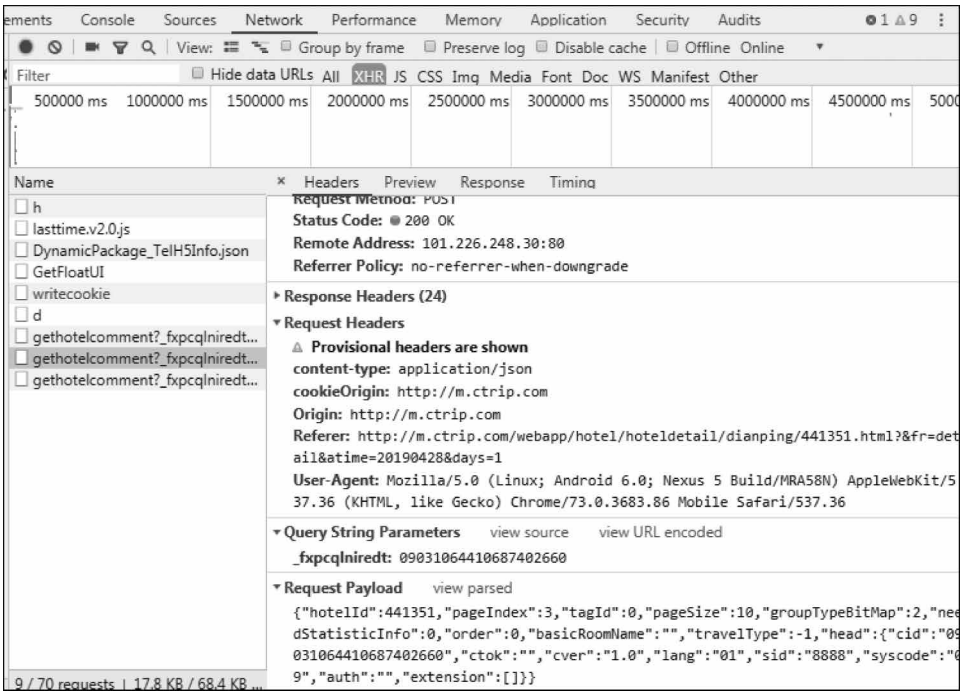

图 5-13　POST 请求参数 Request Payload

```
import requests
import json
#以下 payload 数据来自浏览器看到的结果
payload = {"hotelId":441351,"pageIndex":2,"tagId":0,"pageSize":10,"groupTypeBitMap":2,
"needStatisticInfo":0,"order":0,"basicRoomName":"","travelType": - 1,"head":{"cid":
"09031064410687402660","ctok":"","cver":"1.0","lang":"01","sid":"8888","syscode":
"09","auth":"","extension":[]}}
payloadHeader = {
        'Content - Type': 'application/json',
}
#封装成为 JSON 形式
data = json.dumps(payload)
#以 POST 方法发送 URL 请求,同时指定所携带的参数给函数参数 data
res = requests.post(url, data = json.dumps(payload), headers = payloadHeader)
res.encoding = 'utf - 8'
```

那么这段程序运行后,即可通过 res.text()获取到评论记录信息。由于是动态请求,所以在 Payload 中包含了丰富的信息来实现页面的动态内容。在页面没有经过混淆等处理的情况下,可以直接通过参数名称大体判断其含义。但是如果参数经过了混淆处理,则需要通过一定的观察和分析才能具体确定每个参数的含义。

5.6 模拟浏览器——以自动登录邮箱为例

视频讲解

模拟浏览器有3种实现方式，一种是以模拟特定浏览器的header信息方式来实现对浏览器的模拟；一种是使用浏览器内核(例如WebKit)；还有一种是直接在浏览器上开发组件(Firefox/Chrome)实现动态页面的采集。其中，第一种方式只是简单地在爬虫程序调用requests.get()时指定headers参数，以下例子是一种基本用法。

```
useragent = 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/72.0.3626.121 Mobile Safari/537.36'
http_headers = {
    'User-Agent':useragent,
    'Accept': 'text/html'
    #其他头部属性
}
page = requests.get(url, headers = http_headers)   #URL要请求的网址
```

其中，useragent的值可以通过在浏览器的开发者模式下查看请求头，获取对应的User-Agent。

在页面的JS脚本比较复杂、Ajax交互较多或存在不同页面之间大量数据交换的情况下，使用浏览器组件模拟浏览器进行页面内容的采集比较合适。这里以Selenium为例介绍具体方法。Selenium是一套完整的Web应用程序测试系统，包含了测试的录制(Selenium IDE)、编写及运行(Selenium Remote Control)和测试的并行处理(Selenium Grid)。它可以模拟真实浏览器，支持多种浏览器，在爬虫中主要用来解决JavaScript渲染问题。

这里以爬虫自动登录邮箱，查看有没有新邮件为例。为达到目的，需要经过安装配置、页面结构分析和程序实现3个步骤，以下分别介绍。

1. 安装配置

在Python下安装Selenium，执行pip install selenium即可，如图5-14所示。安装完成后，下载chromedriver (http://chromedriver.storage.googleapis.com/index.html)，这里以Chrome为例。chromedriver的版本有很多，用户一定要下载与计算机上Chrome浏览器版本相对应的版本。在下载zip包之后将其解压，发现里面仅有一个chromedriver.exe文件，需要放到Chrome浏览器安装目录(即chrome.exe所在

的目录)里面,如图 5-15 所示。

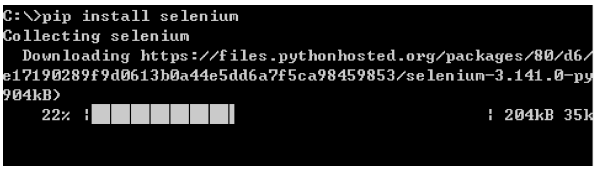

图 5-14　安装 Selenium

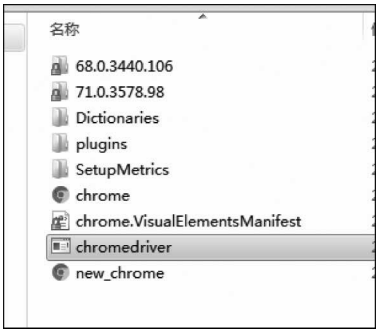

图 5-15　将 chromedriver.exe 放到 Chrome 浏览器安装目录里

最后将 chromedriver.exe 所在的目录名称添加到操作系统的 path 环境变量中,即完成安装配置过程。

2. 页面结构分析

这里以登录邮箱为例(https://mail.fudan.edu.cn/),爬虫自动输入用户名、密码,单击“登录”按钮,相应的页面部分如图 5-16 所示。通过页面的源代码寻找界面控件对应的控件名称,用户名为 uid、密码为 password、“登录”按钮为 Button。

图 5-16　邮箱登录页面的部分

登录以后，通过浏览器开发者模式下的 Source 选项找到信息的位置，即新邮件数，如图 5-17 所示。

图 5-17　寻找所需要的信息(新邮件数)

3. 程序实现

在上述步骤分析完成之后就可以写出 Python 程序了，以下是主体函数。

```
from bs4 import BeautifulSoup
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
import time
import getpass

def start_browser(uid, passwd):   #参数为用户 ID 和口令
    #调用图形界面
    chrome_options = Options()
    chrome_options.add_argument('--headless')
    chrome_options.add_argument('--disable-gpu')
    browser = webdriver.Chrome(chrome_options=chrome_options)
    browser.get('https://mail.fudan.edu.cn/')   #指定 URL
    time.sleep(2)                               #延迟,等待返回

    #模拟填写登录信息
    browser.find_element_by_id("uid").send_keys(uid)
```

```
browser.find_element_by_id("password").send_keys(passwd)
#模拟单击“登录”按钮
browser.find_element_by_class_name("Button").click()
time.sleep(2)

#登录以后的页面中划分了 Frame,切换到相应的 Frame 中进行信息提取
browser.switch_to.frame('welcome')
html = browser.page_source
#使用 BeautifulSoup,具体用法在第 6 章介绍
soup = BeautifulSoup(html, 'html.parser')
try:
   params = soup.select('.fNewMail')[0].text.strip()
   print("有" + str(params) + "封新邮件!")
except IndexError as e:
   print("没有新邮件!")

#关闭浏览器
browser.quit()
return
```

从这段程序可以看出,通过模拟浏览器获取页面内容的主要步骤是:通过webdriver.Chrome初始化浏览器对象,使用浏览器对象进行URL的访问,根据页面中的输入框或按钮的名称调用send_keys或click方法模拟键盘输入或鼠标点击,接着通过浏览器对象的page_source属性获得页面内容,最后通过quit方法关闭浏览器对象。

通过浏览器组件来模拟浏览器的优点是使用简单,复杂的事都交给框架去处理。但是其缺点也是很明显的,在执行过程中需要启动控制台进程,执行速度慢,需要动态地执行JS,并模拟人的浏览器操作。

思考题

1. 页面内容的生成方式有哪几种?

2. 通过URL传递请求参数时URL的特征是什么?如何获得实际传递的参数和值?

3. 找一个实际的Web页面,其中包含Ajax请求,并通过浏览器的开发者工具查看请求的URL。

4. 编写程序,实现通过POST发送请求参数。

5. 学习使用模拟浏览器Selenium进行页面内容采集的方法。