5.2 Selenium 编写爬虫程序

任务：

现在很多网站的网页都不是静态的 HTML 文档，大部分都包含 JsvaScript 程序，很多信息都是通过 JavaScript 程序处理后才显示出来的，因此我们要使用 Selenium 模拟浏览器访问网站的方法来获取网页文档。

5.2.1 JavaScript 控制网页

网页上的信息不一定都是静态的 HTML 数据，实际上很多信息是通过 JavaScript 处理后得到的，那么怎么样去爬取这些数据呢？我们先来设计一个包含 JavaScript 的网页文档，看看怎么样爬取它的数据。

1、创建网页模版

创建 project5 项目，在它下面的 templates 文件夹中设计一个网页文件 phone.html，它包含三个，第一个的 ID 是"hMsg"，它的信息是确定的"Html Message"；第二个 ID 是"jMag"，它的信息是在网页转载时由 JavaScript 的程序赋予值"JavaScript Message"；第三个的 ID 是"sMsg"，它的信息是网页在转载时通过 Ajax 的方法向服务器提出 GET 请求获取的，服务器返回字符串值"Server Message"，phone.html 模版文件如下：

```
<script>
    function init()
    {
        http=new XMLHttpRequest();
        http.open("get","/show",false);
        http.send(null);
        msg=http.responseText;
        document.getElementById("sMsg").innerHTML=msg;
        document.getElementById("jMsg").innerHTML="JavaScript Message";
    }
</script>
<body onload="init()">
Testing<br>
<span id="hMsg">Html Message</span><br>
<span id="jMsg"></span><br>
<span id="sMsg"></span>
</body>
```

2、创建服务器程序

服务器程序 server.py 显示出 phone.html 文件，其中 index 函数读取该文件并发送出去，show 函数是在接受地址"/show"请求后发送"Server Message".

```
import flask
app=flask.Flask(__name__)
@app.route("/")
def index():
    return flask.render_template("phone.html")
```

```
@app.route("/show")
def show():
    return "Server Message"
app.run()
```

3、浏览器浏览

启动服务器 server.py 程序，浏览 web 地址"http://127.0.0.1:5000"，结果如图 5-2-1 所示。

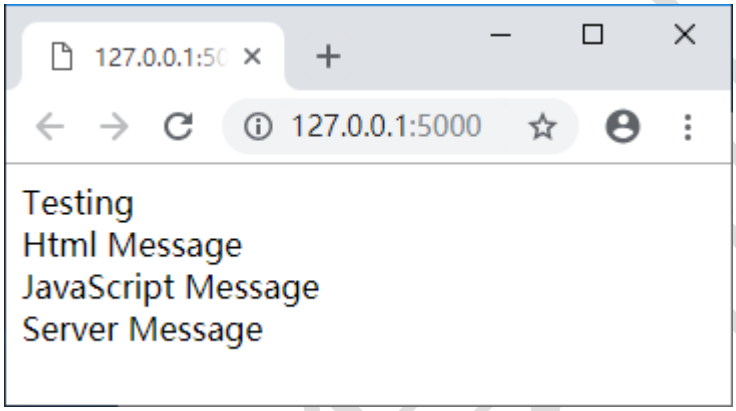

图 5-2-1 测试网站

5.2.2 普通爬虫程序的问题

1、普通客户端程序

设计一个客户端程序，它通过 urllib.request 直接访问"http://127.0.0.1:5000"，程序如下：

```
import urllib.request
resp=urllib.request.urlopen("http://127.0.0.1:5000");
data=resp.read()
data=data.decode()
print(data)
```

执行该程序，我们看到输出的结果如下：

```
<script>
    function init()
    {
        http=new XMLHttpRequest();
        http.open("get","/show",false);
        http.send(null);
        msg=http.responseText;
        document.getElementById("sMsg").innerHTML=msg;
        document.getElementById("jMsg").innerHTML="JavaScript Message";
    }
</script>
<body onload="init()">
<span id="hMsg">Html Message</span><br>
<span id="jMsg"></span><br>
```

```
<span id="sMsg"></span>
</body>
```

这个结果就是 phone.html 文件，注意这个结果中没有 id="iMsg"与 id="sMsg"的的信息。

2、编写普通爬虫程序

设计一个客户端程序，它通过 urllib.request 直接访问"http://127.0.0.1:5000"获取 HTML 代码，使用 BeautifulSoup 解析得到数据，程序如下：

```
from bs4 import BeautifulSoup
import urllib.request
resp=urllib.request.urlopen("http://127.0.0.1:5000");
html=resp.read()
html=html.decode()
soup=BeautifulSoup(html,"lxml")
hMsg=soup.find("span",attrs={"id":"hMsg"}).text
print(hMsg)
jMsg=soup.find("span",attrs={"id":"jMsg"}).text
print(jMsg)
sMsg=soup.find("span",attrs={"id":"sMsg"}).text
print(sMsg)
```

执行该程序结果如下：

```
Html Message
```

显然如果我们通过该方法获取网页 HTML 文档然后进行数据爬取，那么只能爬取 hMsg 的信息"HTML Message"，但是爬取不到 jMsg 与 sMsg 的信息的！

因为 jMsg 与 sMsg 的信息不是静态地嵌入在网页中的，而是通过 JavaScript 与 Ajax 动态产生的，通过 urllib.request.urlopen 得到的网页中没有这样的动态信息，如果要得到这些信息就必须让爬虫程序能够执行对应的 JsvaScript 程序，下面介绍的 Selenium 框架有这个功能。

5.2.3 安装 Selenium 框架

前面已经分析了要获取 jMsg 与 sMsg 的这些信息就必须在获取网页后客户端能按要求执行对应的 JsvaScript 程序。显然一般的客户端程序没有这个能力去执行 JavaScript 程序，我们必须寻找一个能像浏览器那样工作的插件来完成这个工作，它就是 selenium。selenium 就是一个没有显示界面的浏览器，它能与通用的浏览器（例如 chrome,firefox 等）配合工作。我们要先安装 selenium 与 chrome 的驱动程序。

(1) 安装 selenium

```
pip install selenium
```

执行该命令即可完成安装。

(2) 安装 chrome 驱动程序

要 selenium 与浏览器配合工作就必须安装浏览器对应的驱动程序，例如要与 chrome 配合就要先到 http://chromedriver.chromium.org/网站下载 chromedrive.exe 的驱动程序，然后把它复制到 python 的 scripts 目录下。

5.2.4 编写 Selenium 爬虫程序

1、使用 Selenium 查看网页代码

按下列步骤编写客户端程序：

(1) 程序先从 selenium 引入 webdriver，引入 chrome 程序的选择项目 Options：

```
from selenium import webdriver
from selenium.webdriver.driver.options import Options
```

(2) 设置启动 chrome 时不可见：

```
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
```

(3)创建 chome 浏览器:

```
chrome= webdriver.Chrome(chrome_options=chrome_options)
```

这样创建的 chrome 浏览器是不可见的，如果仅仅使用：

```
chrome= webdriver.Chrome()
```

创建 chrome 浏览器，那么在程序执行时会弹出一个 chrome 浏览器。

(4) 使用 driver.get(url)方法访问网页：

```
driver.get("http://127.0.0.1:5000")
```

(5) 通过 driver.page_source 获取网页 HTML 代码：

```
html=driver.page_source
print(html)
```

(5) 使用 driver.close()关闭浏览器：

```
driver.close()
```

根据这样的规则编写客户端程序如下：

```
from selenium import webdriver
from selenium.webdriver.driver.options import Options
chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
chrome= webdriver.Chrome(chrome_options=chrome_options)
driver.get("http://127.0.0.1:5000")
html=driver.page_source
print(html)
driver.close()
```

执行这个程序，结果如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head><script>
    function init()
    {
        http=new XMLHttpRequest();
        http.open("get","/show",false);
        http.send(null);
        msg=http.responseText;
        document.getElementById("sMsg").innerHTML=msg;
        document.getElementById("jMsg").innerHTML="JavaScript Message";
    }
```

```
</script></head>
<body onload="init()">
<span id="hMsg">Html Message</span><br />
<span id="jMsg">JavaScript Message</span><br />
<span id="sMsg">Server Message</span>
</body>
</html>
```

由此可见我们得到的 HTML 文档时执行完 JavaScript 程序后的文档，其中包含了 jMsg 与 sMsg 的信息！

3、编写 Selenium 爬虫程序

Selenium 模拟浏览器访问网站的方法来获取网页文档，然后从中爬取要的数据，这样的爬虫程序功能就比较强大了，设计爬虫程序如下：

```
from selenium import webdriver
from selenium.webdriver.driver.options import Options
from bs4 import BeautifulSoup
chrome_options = Options()
chrome_options.add_argument('--headless')
chrome_options.add_argument('--disable-gpu')
chrome = webdriver.Chrome(chrome_options=chrome_options)
driver.get("http://127.0.0.1:5000")
html=driver.page_source
soup=BeautifulSoup(html,"lxml")
hMsg=soup.find("span",attrs={"id":"hMsg"}).text
print(hMsg)
jMsg=soup.find("span",attrs={"id":"jMsg"}).text
print(jMsg)
sMsg=soup.find("span",attrs={"id":"sMsg"}).text
print(sMsg)
```

执行该程序，结果我们爬取到了所有数据：

```
Html Message
JavaScript Message
Server Message
```

由此可见，采用 Selenium 的结构主要是模拟浏览器去访问网页，并充分执行网页中的 JavaScript 程序，使得网页的数据充分下载，这样再用爬虫程序去爬取数据就必要稳妥了。