一步一步教你用 Python 实现网络爬虫

【引言】

网络爬虫的入门并没有想象中那么困难，困难的是你有没有勇气踏出第一步。本文将详细介绍利用 Python 语言实现一个基于 Requests 和 BeautifulSoup 技术的网络爬虫，将博客园上特定用户的博客取下来，通过解析获得文章标题和链接，将其形式可视数据并存储到在指定的文本文件中。

【实现思路】

网络爬虫，就是抓取网页数据的程序。

网络爬虫的实现流程包括三个部分：获取网页、解析网页、存储数据。

首先通过 Requests 库向指定的 URL 地址发送 HTTP 请求，从而把整个网页的数据爬取下来，接着通过 BeautifulSoup 模块对页面数据进行解析并对目标数据进行定位，从而将需要的信息抽取出来，最后通过文件操作将数据存储到指定的文本文件中。

【准备工作：引入相关的第三方库】

引入 HTTP 客户端库 Requests：

```
import requests
```

引入网页解析利器 BeautifulSoup 模块：

```
from bs4 import BeautifulSoup
```

【步骤 1：获取网页】

要获取网页的内容，首先要确定该页面的 URL 地址，然后通过 Requests 库的 get()方法向服务发起请求，从而获取网页数据。

```
http://www.cnblogs.com/ALittleBee/default.html?page=1
http://www.cnblogs.com/ALittleBee/default.html?page=2
http://www.cnblogs.com/ALittleBee/default.html?page=3
```

通过对博客页面 URL 的分析，可以发现每个页面直接的规律：URL 地址的大部分内容是相同的，只有最后一个数字在变化，而且这个数据代表的是页面的序号，需要访问那个页面，就把该数字设置为相应的数字就可以了。例如，要访问第 8 个页面的数据，其 URL 的地址为：

```
http://www.cnblogs.com/ALittleBee/default.html?page=8
```

因此，我们要访问博客的所有页面内容，只需要通过一个循环，对 URL 地址

的最后一个数字进行改变即可。例如，要把该博客的前面 10 页数据爬取下来，通过以下的代码就可以实现：

```
url = "http://www.cnblogs.com/ALittleBee/default.html?page="
for i in range( 1,11 ):
    urlf = url + str( i )
    res = requests.get( urlf )
```

是不是很简单呢？没错，使用 Python 就是这么优雅和简单。

为了保证访问的“合法”性，需要通过添加一个请求头参数，将爬虫的请求伪装成合法浏览器的访问，这服务器才不会拒绝这次请求而返回完整的数据。这个功能的实现，只需要在上述的代码基础上稍作修改：

```
user_agent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;)"
headers = { "User-Agent" : user_agent }
url = "http://www.cnblogs.com/ALittleBee/default.html?page="
for i in range( 1,11 ):
    urlf = url + str( i )
    res = requests.get( urlf,   headers = headers )
```

【步骤 2：解析网页】

我们需要的是页面中感兴趣的数据，而不是整个页面的全部信息。对页面内容进行分析的第一步不是编写代码，而是分析爬取下来的页面代码，找到目标数据所在的位置，并对相应的元素进行定位，然后才能将目标信息准确的抽取出来。实现这个网页解析有多种方法，在这里我们选用 BeautifulSoup 模块作为解析器，因为它使用起来更加直观和易懂。

通过 Requests 的 get()方法将网页的全部内容保存在 res 变量中，将 res 变量作为参数以‘lxml’作为解析器产生一个 BeautifulSoup 对象 soup：

```
soup = BeautifulSoup( res.text,   "lxml" )
```

通过对页面 HTML 源码的分析，可以得到目标内容的元素定位：

〈1〉文章标题和链接的目标内容均包含在“class”属性值为“postTitle2”的“a”标签中。

〈2〉文章标题为“a”标签中字符串内容。

〈3〉链接信息为“a”标签中的“href”属性值。

首先利用该 BeautifulSoup 对象 res 的 find_all()方法将符合元素定位信息的“a”标签全部抽取出来，形成一个列表。该列表中的每一个元素实际上就是博客中每篇文章的标题和链接信息。通过遍历列表中的每一个元素，只要把其中的字符串和“href”属性值提取出来，分别赋值给 title 和 link 变量，就可以

很容易的获得目标数据：

```
titles = soup.find_all( 'a', {'class' : 'postTitle2'} )
    for item in titles:
        title = item.text.strip()
        link = item['href']
```

【步骤 3：存储数据】

通过爬虫获取的数据，经过解析后形成有价值的信息。我们需求把这些数据存储起来，可以存储在文件中也可以存储在数据库中。对于入门初学这来说，先学会把爬取的数据存储在指定的文本文件中。

在我们这个爬虫中，把数据存储到本地 E 盘的“blogs.txt”文件中。具体由 Python 的文件操作语句来实现，相当的简单，只有两句话：

```
with open( 'e:\\blogs.txt' , 'a+' )  as  f:
    f.write( title + '\n' + link + '\n )
```

【附录：源码及运行结果】：

```
import requests
from bs4 import BeautifulSoup

url = "http://www.cnblogs.com/ALittleBee/default.html?page="
user_agent = "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Trident/5.0;)"
headers = {"User-Agent":user_agent}
for i in range(1,3):
    urlf = url + str(i)
    res = requests.get(urlf, headers = headers)

    soup = BeautifulSoup(res.text, "lxml")
    titles = soup.find_all('a',{'class':'postTitle2'})
    for item in titles:
        title = item.text.strip()
        link = item['href']

        with open("e:\\MySpider\\blogs.txt","a+") as f:
            f.write(title + '\n' + link + '\n')
```

```
【网络爬虫入门05】分布式文件存储数据库MongoDB的基本操作与爬虫应用
http://www.cnblogs.com/ALittleBee/p/7707509.html
【网络爬虫入门04】彻底掌握BeautifulSoup的CSS选择器
http://www.cnblogs.com/ALittleBee/p/7702560.html
【网络爬虫入门03】爬虫解析利器beautifulSoup模块的基本应用
http://www.cnblogs.com/ALittleBee/p/7700127.html
【网络爬虫入门02】HTTP客户端库Requests的基本原理与基础应用
http://www.cnblogs.com/ALittleBee/p/7669180.html
【网络爬虫入门01】应用Requests和BeautifulSoup联手打造的第一条网络爬虫
http://www.cnblogs.com/ALittleBee/p/7668149.html
【Zigbee技术入门教程-02】一图读懂ZStack协议栈的基本架构和工作机理
http://www.cnblogs.com/ALittleBee/p/7463239.html
【Zigbee技术入门教程-号外】基于Z-Stack协议栈的抢答系统
http://www.cnblogs.com/ALittleBee/p/7419394.html
```