第9章 网络爬虫

提纲

- 9.1 网络爬虫概述
- 9.2 urllib模块
- 9.3 Beautiful Soup

9.1 网络爬虫概述

- 什么是爬虫？
  - 网络爬虫又被称为网页蜘蛛(Web spider)，网络机器人。把互联网比喻成一个蜘蛛网，那么Spider就是在网上爬来爬去的蜘蛛。
  - 爬虫过程：从网站某一个页面（通常是首页）开始，读取网页的内容，找到在网页中的其它链接地址，然后通过这些链接地址寻找下一个网页，这样一直循环下去，直到把这个网站所有的网页都抓取完为止。
- 定义
  - 网络爬虫是一个自动抓取网页的程序或脚本。
    - ✓网络爬虫的基本操作是抓取网页
    - ✓网络爬虫是搜索引擎的重要组成部分。

9.1-1统一资源定位符URL

- 如何才能随心所欲地获得自己想要的页面？
  - 先从URL开始。

- 浏览网页
  - 抓取网页的过程其实和读者平时使用IE、chrome等浏览器浏览网页的道理是一样的。
    - ✓比如，浏览器的地址栏中输入www.baidu.com 这个地址
    - ✓打开网页的过程其实就是浏览器作为一个浏览的“客户端”，向服务器端发送了 一次请求，把服务器端的文件“抓”到本地，再进行解释、展现。
    - ✓HTML是一种标记语言，用标签标记内容并加以解析和区分。
    - ✓浏览器的功能是将获取到的HTML代码进行解析，然后将原始的代码转变成我们直接看到的网站页面。

9.1-1统一资源定位符URL

- 统一资源定位符： Uniform Resource Locator
  - 通俗地说，URL是Internet上描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上。
  - 采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。
- URL的格式由三部分组成：
  - 1. 协议(或称为服务方式)
  - 2. 存有该资源的主机IP地址(有时也包括端口号)
  - 3. 主机资源的具体地址，如目录和文件名等
    - ✓第1部分和第2部分用“://”符号隔开，
    - ✓第2部分和第3部分用“/”符号隔开。

9.1-1统一资源定位符URL

- URL例子
  - http协议

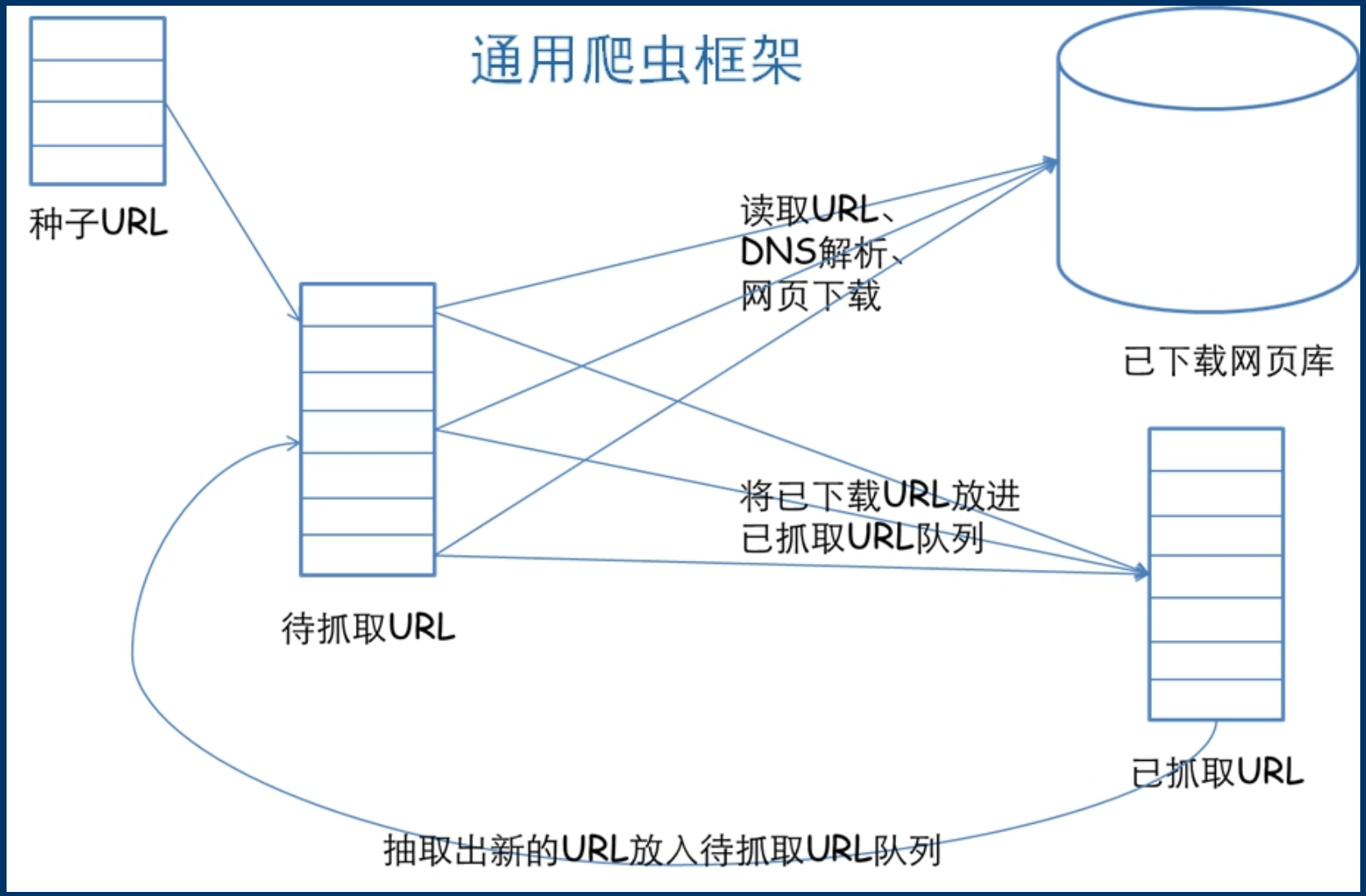
    http://www.zjgsu.edu.cn/Channel_1/index.html
    - ✓1. 使用超级文本传输协议HTTP，提供超级文本信息服务的资源。
    - ✓2. 计算机域名为www.zjgsu.edu.cn
    - ✓3. 超级文本文件(文件类型为.html)是在目录 / Channel_1下的 index.html
  - ftp的URL

    ftp://test:test@192.168.0.1:21/profile
    - ✓1. ftp协议，用户名和密码为：test、test
    - ✓2. 服务器IP和port为：192.168.0.1:21
    - ✓3. 登录后转到profile文件夹

9.1-2 爬虫的工作流程

- 基本工作流程
  - 1. 首先选取一部分精心挑选的种子URL；
  - 2. 将这些URL放入待抓取URL队列；
  - 3. 从队列中取出待抓取URL，解析DNS，并且得到主机的IP，并将URL对应的网页下载下来，存储进已下载网页库中。此外，将这些URL放进已抓取URL队列。
  - 4. 分析已抓取URL队列中的URL，分析其中的其他URL，并且将URL放入待抓取URL队列，从而进入下一个循环。

9.1-2 爬虫的工作流程


通用爬虫框架
种子URL
读取URL、
DNS解析、
网页下载
已下载网页库
将已下载URL放进
已抓取URL队列
待抓取URL
已抓取URL
抽取出新的URL放入待抓取URL队列

9.1-2 爬虫的工作流程

- Python爬虫工具分类：
  - 简单爬虫设计
    - ✓下载网页：urllib（入门简单爬虫库），Requests
    - ✓解析网页：Beautiful Soup
    - ✓模拟交互，处理JS动态网页：Selenium
  - 高级爬虫框架Scrapy
  - 分布式爬虫设计
    - ✓分布式队列
    - ✓布隆过滤器 (Bloom Filter)

9.2 urllib模块

urllib 库

- Urllib库是Python中一个功能强大、用于操作URL，在做网络爬虫时经常用到的库。
- 在Python2.x中，分为Urllib库和Urllib2库，Python3.x之后都合并到Urllib库中，使用方法稍有不同。
  - ✓在Pytho2.x中使用import urllib2——-对应的，在Python3.x中会使用import urllib.request，urllib.error
  - ✓在Pytho2.x中使用import urlopen——-对应的，在Python3.x中会使用import urllib.request.urlopen
  - ✓在Pytho2.x中使用import urlparse——-对应的，在Python3.x中会使用import urllib.parse

9.2 urllib模块

- **urllib 库的四个基本模块**
  - urllib.request：可以用来发送request和获取request的结果
  - urllib.error：包含urllib.request产生的异常
  - urllib.parse：用来解析和处理URL
  - urllib.robotparse：用来解析页面的robots.txt文件

9.2-1 基本操作

1.使用urlopen获取网页

- **urllib.request.urlopen(url, data=None, [timeout,]*, cafile=None, capath=None, cadefault=False, context=None)**
  - ✓url：需要打开的网址，可以是一个string，或者一个Request对象
  - ✓data：Post方式提交的数据
  - ✓timeout：设置网站的访问超时时间

```
3 #simplest_crawler.py
4 import urllib.request as
5 response = ur.urlopen('http://python.org/')
```

HTTPResponse对象

```
<http.client.HTTPResponse object at 0x0000232308686D8>
Python主页:
b'<!doctype html>\n<!--[if lt IE 7]>   <html class="no-js ie6
lt-ie7 lt-ie8 lt-ie9">   <![endif]-->\n<!--[if IE 7]>
<html class="no-js ie7 lt-ie8 lt-ie9">          <![endif]-->
\n<!--[if IE 8]>      <html class="no-js ie8 lt-ie9">
<![endif]-->\n<!--[if gt IE 8]><!--><html class="no-js"
lang="en" dir="ltr">  <!--<![endif]-->\n\n<head>\n    <meta
charset="utf-8">\n    <meta http-equiv="X-UA-Compatible"
content="IE=edge">\n\n    <link rel="prefetch" href="//
```

9.2-1 基本操作

- **urlopen返回的对象提供方法：**
  - read() , readline() ,readlines() , fileno() , close() ： 对HTTPResponse类型数据进行操作
  - info()：返回HTTPMessage对象，表示远程服务器返回的头信息
  - getcode()：返回Http状态码。如果是http请求，200请求成功完成;404网址未找到
  - geturl()：返回请求的url

9.2-1 基本操作

2. 使用urllib.request.Request包装请求，再通过urlopen获取页面

```
#request_urlopen.py
import urllib.request as ur
url = r'http://www.lagou.com/zhaopin/Python/?labelWords=label'
headers = {        #伪装成浏览器
    'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; WOW64) \
                    AppleWebKit/537.36(KHTML, like Gecko) '
                   r'Chrome/45.0.2454.85 Safari/537.36 115Browser/6.0.3',
    'Referer': r'http://www.lagou.com/zhaopin/Python/?labelWords=label',
    'Connection': 'keep-alive' }
req = ur.Request(url, headers=headers)
print(req)
page = ur.urlopen(req).read()
page = page.decode('utf-8')
print(page)

```

```
<urllib.request.Request object at 0x0000012C875A0048>
<!DOCTYPE html>
<html>
<head>
    <!-- meta -->
    <meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible"
```

9.2-1 基本操作

Post数据

- **urllib.request.Request()**中的data参数默认为None， 当data参数不为空的时候，urlopen()提交方式为Post

```
#post_urlopen.py
from urllib import request, parse
url = r'http://www.lagou.com/jobs/positionAjax.json?'
headers = {          #伪装成浏览器
    'User-Agent': r'Mozilla/5.0 (Windows NT 6.1; WOW64) \
                    AppleWebKit/537.36 (KHTML, like Gecko) '
                   r'Chrome/45.0.2454.85 Safari/537.36 115Browser/6.0.3',
    'Referer': r'http://www.lagou.com/zhaopin/Python/?labelWords=label',
    'Connection': 'keep-alive' }
data = {
    'first': 'true',
    'pn': 1,
    'kd': 'Python' }
data = parse.urlencode(data).encode('utf-8')
req = request.Request(url, headers=headers, data=data)
page = request.urlopen(req).read()
page = page.decode('utf-8')

```

9.2-1 基本操作

Post数据(续)

- urllib.parse.urlencode(query, doseq=False, safe='', encoding=None, errors=None)
  urlencode( )主要作用就是将url附上要提交的数据

```
11 data = {
12     'first': 'true',
13     'pn': 1,
14     'kd': 'Python' }
15 data = parse.urlencode(data).encode('utf-8')
16 req = request.Request(url, headers=headers, data=data)
```

- 经过urlencode( )转换后的data数据为?first=true?pn=1?kd=Python，最后提交的url为:
  http://www.lagou.com/jobs/positionAjax.json?first=true?pn=1?kd=Python

9.2-2 简单抓取网页

不同网页爬取

```
#tieba_crawler.py
import urllib.request    #主要用于打开和阅读url
from os.path import join #用于串联完整路径

wp=r"D:\Teaching\@LessonCourse\Python Programming and Practice\MyLecture\pyegs\ch9\html"
print("模拟抓取百度贴吧python、java、C#页面,并写入指定路径文件")
def tieba_baidu(url,s):
    header={'User-Agent': r'Mozilla/5.0 (Windows NT 6.3; WOW64) \
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/43.0.235'}#伪装成浏览器
    for i in range(len(s)):
        file_name=join(wp,s[i]+".html")
        print("正在下载"+s[i]+"页面，并保存为"+file_name)
        req=urllib.request.Request(url+s[i],headers=header)
        m=urllib.request.urlopen(req).read()
        with open(file_name,"wb") as file:
            file.write(m)

if __name__=="__main__":
    url="http://tieba.baidu.com/f?kw="
    s_tieba=["python","java","c#"]
    tieba_baidu(url,s_tieba)
```

9.2-2 简

下载网页

```python
#picture_crawler.py
import urllib.request    #主要用于打开和阅读url
import os,re
import urllib.error      #用于错误处理

print("爬取指定页面的jpg格式的图片")
dirpath="D:\Teaching\@LessonCourse\Python Programming and Practice\MyLecture\pyegs\ch9\pic"
def baidu_tieba(url,s):
    '''根据传入的地址和关键字列表进行图片抓取'''
    for i in range(len(s)):
        count=1
        file_name=os.path.join(dirpath,s[i]+".html")
        print("正在下载"+s[i]+"页面，并保存为"+file_name)
        m=urllib.request.urlopen(url+s[i]).read()
        dirname=s[i]
        new_path=os.path.join(dirpath,dirname)
        if not os.path.isdir(new_path):
            os.makedirs(new_path) #创建目录保存每个网页上的图片
        page_data=m.decode()
        page_image=re.compile('<img src=\"(.+?)\"')    #匹配图片的pattern
        for image in page_image.findall(page_data):#用正则表达式匹配所有的图片
            pattern=re.compile(r'http://.*.jpg$')   #匹配jpg格式的文件
            if pattern.match(image): #如果匹配 则获取图片信息，若不匹配，进行下一个页面的匹配
                try:
                    image_data=urllib.request.urlopen(image).read()    #获取图片信息
                    image_path=os.path.join(dirpath,dirname,str(count)+".jpg")   #给图片命名
                    count+=1
                    with open(image_path,"wb") as image_file:
                        image_file.write(image_data)                    #将图片写入文件
                except urllib.error.URLError as e:
                    print("Download failed")
            with open(file_name,"wb") as file:    #将页面写入文件
                file.write(m)

if __name__=="__main__":
    url="http://tieba.baidu.com/f?kw="
    s_tieba=["python","java","c#"]
    baidu_tieba(url,s_tieba)
```

9.3 Beautiful Soup

- Beautiful Soup为python抓取网页数据的一个库
  - Beautiful Soup提供一些简单的、python式的函数用来处理导航、搜索、修改分析树等功能。
  - Beautiful Soup是一个工具箱，通过解析文档为用户提供需要抓取的数据，因为简单，所以不需要多少代码就可以写出一个完整的应用程序。
  - Beautiful Soup自动将输入文档转换为Unicode编码，输出文档转换为utf-8编码。
  - Beautiful Soup已成为和lxml、html6lib一样出色的python解释器，为用户灵活地提供不同的解析策略或强劲的速度。
  - Python正则表达式的使用较为繁琐，而 Beautiful Soup在数据提取方面友好易用。

9.3 Beautiful Soup

一个简单例子：抓取文章标题

```
3 #bs4_title_crawler.py
4 from urllib import request
5 from bs4 import BeautifulSoup
6
7 #获取简书首页的全部文章标题
8 url = r'http://www.jianshu.com'
9 # 模拟真实浏览器进行访问
10 headers = {'User-Agent':'Mozilla/5.0 (Windows NT 10.0; WOW64) \
11     AppleWebKit/537.36 (KHTML, like Gecko) Chrome/55.0.2883.87 Safari/537.36'}
12 page = request.Request(url, headers=headers)
13 page_info = request.urlopen(page).read()
14 page_info = page_info.decode('utf-8')
15 # 将获取到的内容转换成BeautifulSoup格式，并将html.parser作为解析器
16 soup = BeautifulSoup(page_info, 'html.parser')
17 # 以格式化的形式打印html
18 # print(soup.prettify())
19 titles = soup.find_all('a', 'title')  # 查找所有a标签中class='title'的语句
20 # 打印查找到的每一个a标签的string
21 #print(titles)
22 for title in titles:
23     print(title.string)
```