

表示

网络

可视化

数据

计算

特征分析

第五章 网络爬虫与信息提取

5.1 爬虫基本原理及获取数据

同济大学 计算机基础教研室



1. 爬虫原理及网页构造

2. 引例及Requests库

3. 问题及限制

4. 实例

网络连接

自助售货机：

选择商品→投入硬币/纸币/扫码→弹出相应的商品

浏览网页：

在浏览器中输入一个网址后敲击回车，看到网站的页面信息。

即：浏览器请求了网站的服务器，获取到网络资源。



电脑

www.tongji.edu.cn

request
(请求头和消息体)



response
(HTML文件)



服务器



(1) 本机电脑(购买者)带着请求头和消息体(硬币和商品需求)向服务器(自助售货机)发起一次Request请求(购买)

爬虫：

模拟浏览器发送请求，获得html代码。

html包括标签和文字，再从中提取想要的信息。

(2) 相应的服务器(自助售货机)会返回相应的HTML文件(相应的商品)作为Response

爬虫原理

网络爬虫(Web Spider) 如果把互联网看作一张大网，那么爬虫就是在大网上爬来爬去的蜘蛛，碰到想要的食物就把他抓取出来。



爬虫原理

网络爬虫就是一种按照一定的规则，根据网页的地址（即URL）自动地抓取网页信息的程序或者脚本。

网络连接需要：

(1) 电脑的Request请求

(2) 服务器端的Response回应

爬虫需要：

(1) 模拟电脑对服务器发起Request请求

(2) 接收服务器端的Response的内容，并且解析提取所需信息

引例1——爬取百度新闻页面 (news.baidu.com)

引入Requests库

import requests

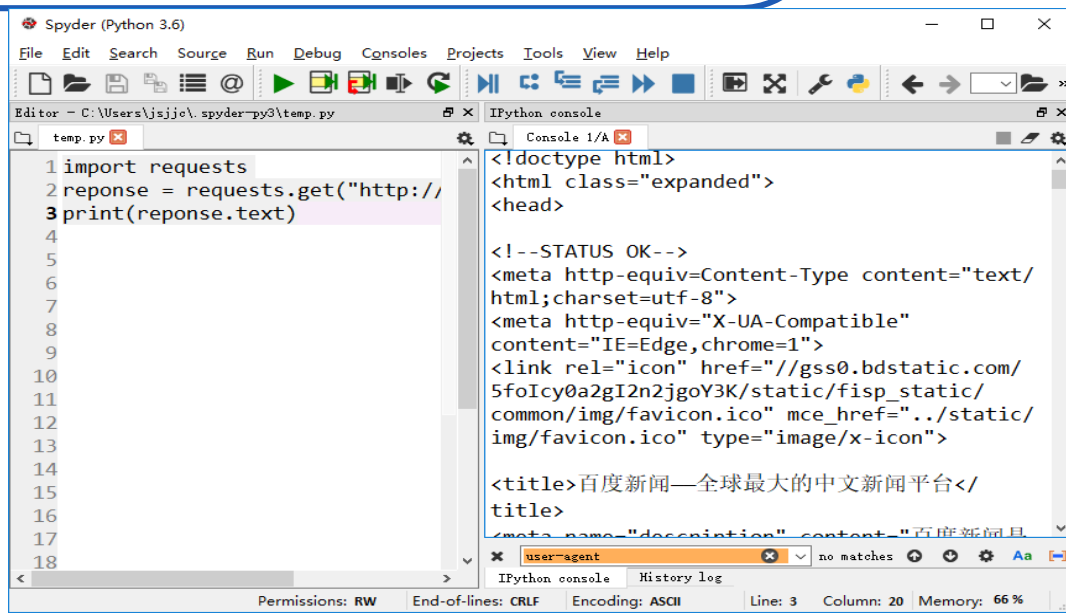
发起GET请求

response = requests.get("http://news.baidu.com/")

输出响应内容

print(response.text)

URL地址



The screenshot shows the Spyder Python IDE interface. The Editor window displays a Python script in `temp.py` with the following code:

```
1 import requests
2 reponse = requests.get("http://
3 print(reponse.text)
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
```

The IPython console window shows the output of the script, which is the HTML content of the Baidu News page:

```
<!doctype html>
<html class="expanded">
<head>

<!--STATUS OK-->
<meta http-equiv=Content-Type content="text/
html; charset=utf-8">
<meta http-equiv="X-UA-Compatible"
content="IE=Edge,chrome=1">
<link rel="icon" href="//gss0.bdstatic.com/
5foIcy0a2gI2n2jgoY3K/static/fisp_static/
common/img/favicon.ico" mce_href="../static/
img/favicon.ico" type="image/x-icon">

<title>百度新闻—全球最大的中文新闻平台</
title>
<meta name="description" content="百度新闻是
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: CRLF, Encoding: ASCII, Line: 3, Column: 20, Memory: 66 %.

网页html代码

- ✓ Chrome浏览器输入URL地址“news.baidu.com”打开页面
- ✓ 在网页处单击右键，选择“检查”，打开开发者工具
- ✓ Elements选项卡包含了网页经JavaScript处理的最终HTML代码。
- ✓ Network选项卡中包含了浏览器收到的网页源代码。
- ✓ (IE中“查看源文件”或Firefox中“查看页面源代码”)



```
function query2Json(n) {
  if (!n)
    return {}
  for (var i = 0; i < n.length; i++) {
    e = n[i]
    t[e] = {}
  }
  return t
}

var UserMonitor = function() {
  function n(n) {
    if (n) {
      var r = "log_" + (new Date).getTime()
      , t = window[r] = new Image()
      t.onload = t.onerror = function() {
        window[r] = null
      }
    }
  }
}
```

JavaScript语言
交互和动画效果

```
<!doctype html>
<html class="expanded">
<head>
  <!--ST
  <meta charset="utf-8" type="text/html" content="t
  <meta http-equiv="X-UA-Compatible" content="IE=
  <link href="//gss0.bdstatic.c
  <title>百度新闻—海里中文资讯平台</title>
  <meta name="description" content="百度新
  <script type="text/javascript">
    document.write("<script type='t
  </script>
  <script type="text/javascript"> window.N
  <script type="text/javascript" src="//gs
  </script>
```

HTML代码
超文本标记语言

```
<div class="l-left-col" alog-group="focus-top-left">
  <div id="left-col-wrapper">
    <div class="recommend-tip" id="recommend">...</div>
    <div id="headline-tabs" class="mod-headline-tab">
      <ul class="clearfix">
        <li class="active">...
          ::after
        </ul>
        <a id="tab-left" href="#" class="mod-recommend"
          href="#" class="mod-recommend" if="true" mon=
          "m=53&...
        </div>
        <div class="...
        </div>
      </div>
      <div class="l-right-col" alog-group="focus-top-right">...</div>
      ::after
    </div>
    <div class="mod-localnews column clearfix" id="local_news">...</div>
    <ul id="goTop" class="mod-sidebar">...</ul>
    <style>...</style>
    <div class="civilnews">...</div>
```

经js处理的
HTML代码

```
Filter
element.style {
}

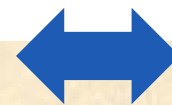
.hotnews li {
  display: block;
  padding: 6px 0 1px;
  display: -webkit-box;
```

CSS样式
层叠样式表

html源代码由很多尖括号构成的标签组织而成

标签之间存在上下游关系

文件



标签树

```
<!doctype html>
```

```
<html xmlns="http://www.w3.org/1999/xhtml" class="sui">
```

```
  <head>...</head>
```

```
  <body class="s-manhattan-index" style>
```

```
    <div id="s_is_index_css" style="display:none;">...</div>
```

```
    <textarea id="s_is_result_css" style="display:none;">
```

```
    <textarea id="s_index_off_css" style="display:none;">
```

```
    <div id="wrapper">
```

```
      <div class="s-skin-container s-isindex-wrap">
```

```
        <div id="head" class>
```

```
          <div id="s_top_wrap" class="s-top-wrap" style>
```

```
          <div id="s_upfunc_menus" class="s-upfunc-menus">
```

```
          <div id="u_sp" class="s-isindex-wrap s-sp-menu">
```

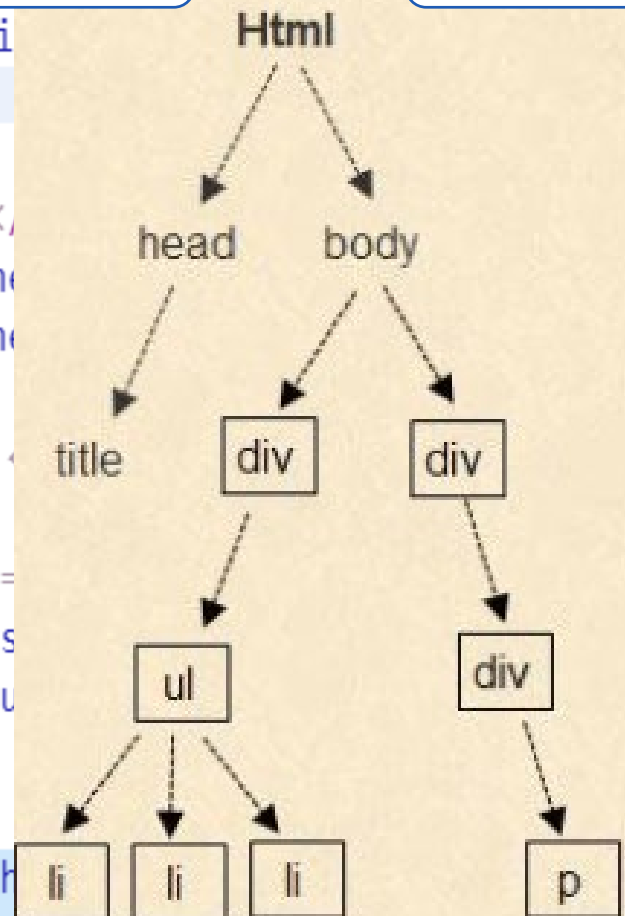
```
            <style>...</style>
```

```
            <div class="clear"></div>
```

```
            <div id="head_wrapper" class="s-isindex-wrap">
```

```
              <img ">...</div> == $0
```

```
            <div id="s_wrap" class="s-isindex-wrap">...</div>
```



<p class="title">.....</p>

国内 国际 军事 财经 娱乐 体育 互联网 科技 游戏 女人 汽车 房产

热点要闻

- 习近平：全面建成小康社会 乘势而上
- 当游客偶遇总书记 习近平鼓励企业不断勇攀科技高峰
- 总书记来到我们家 走进山西转型综改示范区
- 从民法总则到民法典草案·中国民法制度将迎来新时代



Elements Console Sources Network Performance

Memory Application Security Audits

Styles Computed Event Listeners DOM Breakpoints

```
Filter
element.style {
}
.clearfix {
  zoom: 1;
}
.clearfix {
  zoom: 1;
}
ul, ol, li {
  list-style: none;
}
```

经js处理的
HTML代码

CSS样式
层叠样式表

Tips: Chrome中审查元素与网页源代码区别

✓ 审查元素

- 看到实时性的内容(经过js的修改)，即最终的html代码
- 定位网页元素、及时调试、修改、定位、追踪检查、查看嵌套、修改样式和查看js动态输出信息。

✓ 网页源代码

- 看到最开始浏览器收到HTTP响应内容，
- 查看源代码只是把网页输出的源代码直接打开，既不能动态变化，也不能修改。

引例2-爬取页面的新闻标题和链接

引入requests库和bs4库

```
import requests
```

```
from bs4 import BeautifulSoup
```

爬取网页

```
url = "http://news.baidu.com"
```

```
response = requests.get(url)
```

解析网页

```
soup = BeautifulSoup(response.text, 'html.parser')
```

定位div hotnews

```
divs = soup.find('div', class_='hotnews')
```

```
for n in range(0,5):
```

定位a标签

```
a = divs.find('li', class_='hdline{}'.format(n)).find('a')
```

取新闻标题、链接

```
title = a.get_text()
```

```
href = a.get('href')
```

写入文件

```
with open('hotnews.txt', 'a', encoding='utf-8') as f:
```

```
    f.write("标题: " + title + "\n")
```

```
    f.write("链接: " + href + "\n")
```

```
    f.write("\n")
```

hotnews - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

标题: 习近平心中的“14.1178亿”

链接: http://www.xinhuanet.com/politics/xxjxs/2021-05/11/c_1127433527.htm

标题: 短短七个字 蕴含着领袖的深思

链接: [http://app.cctv.com/special/cportal/detail/arti/index.html?](http://app.cctv.com/special/cportal/detail/arti/index.html?id=ArtiKQ8PagP8Pni219Ijg3GY210425&fromapp=cctvnews&version=808&allow_comment=1)

id=ArtiKQ8PagP8Pni219Ijg3GY210425&fromapp=cctvnews&version=808&allow_comment=1

标题: 《平“语”近人(国际版)》第一集海外热播

链接: <http://m.news.cctv.com/2021/05/11/ARTIq6aVdR2PSRykaczEsI8U210511.shtml>

标题: 香港青年在深圳

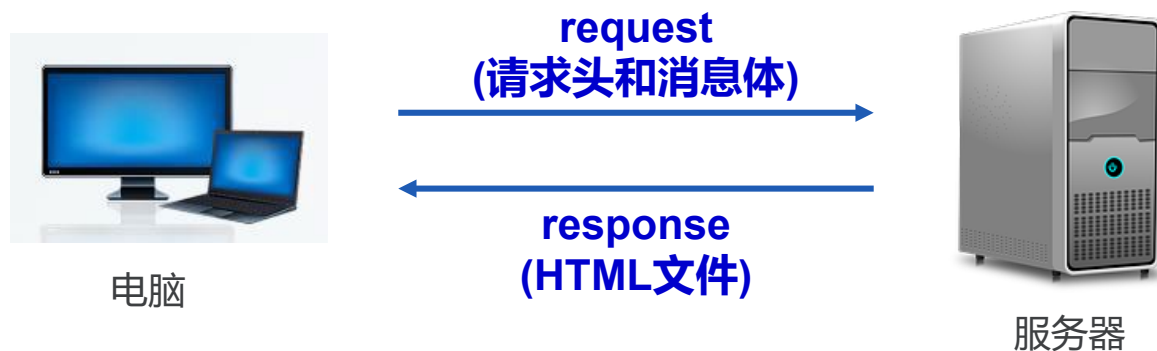
链接: [https://s.cyol.com/shuzibao/cmsfile/cms_json/zqzx/Newspaper/2/2021-05-](https://s.cyol.com/shuzibao/cmsfile/cms_json/zqzx/Newspaper/2/2021-05-11/Content/nw.D110000zgqnb_20210511_3-01.html?isshow=1)

11/Content/nw.D110000zgqnb_20210511_3-01.html?isshow=1

标题: 唱军歌说党史 | 保卫黄河: “为抗战发出怒吼”

链接: https://mp.weixin.qq.com/s/OTN7_JUOMnwUUuq7scVRDg

爬虫基本流程



- 网络爬虫就是要模拟用户使用浏览器访问网页的过程，即先模拟电脑对服务器发起Request请求，再接受服务器端的Response响应内容，根据需要进行解析和提取，并保存所需的信息。

发起请求



获取响应
内容



解析内容



保存数据

爬虫基本流程

1.发起请求：

- ✓ 向目标站点发送一个Request（可以包含额外的headers等信息），即发起请求，然后等待服务器响应。



- ✓ 相当于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(我们打开浏览器并输入网址，然后点击回车)。

爬虫基本流程

2. 获取响应内容：

- ✓ 如果服务器能正常响应，我们会得到一个**Response**，即获取到的内容，类型可能有HTML、Json字符串、二进制数据（图片，视频等）等。



- ✓ 相当于服务器接收客户端的请求，进而将网页HTML文件发送给浏览器。

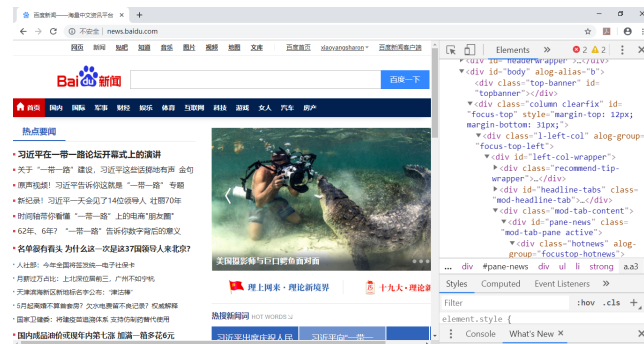
爬虫基本流程

3.解析内容:

✓ 得到的内容

- 也许是HTML，可使用正则表达式，网页解析库（如Beautiful Soup）进行解析
- 也许是Json，可直接转为Json对象解析
- 也许是二进制数据，可进行保存或者进一步处理

- ### ✓ 相当于浏览器把服务器端的文件获取到本地，再进行解释并且展现出来。



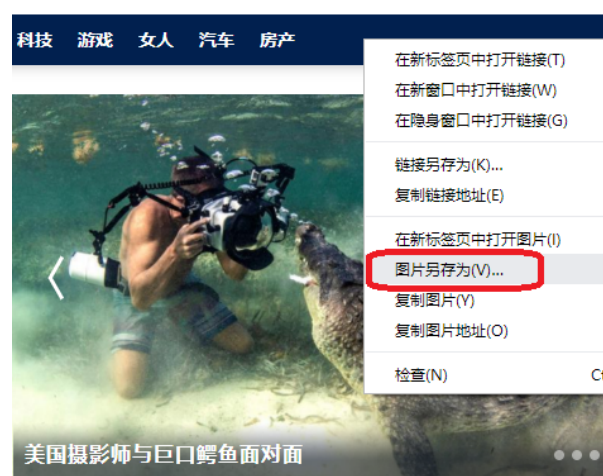
爬虫基本流程

4.保存数据:

✓ 保存的方式

- 把数据存为文本
- 把数据保存到数据库
- 把数据保存为jpg、mp4 等格式...

✓ 相当于浏览网页时，下载网页上的图片或者视频



爬虫基本流程

爬虫就是获取网页并提取和保存信息的自动化程序

- ✓ 在快速获取大量数据时，通过爬虫程序来自动抓取并进行异常处理、错误重试等等操作，确保爬虫保持高效运行。

Urllib.Request / Requests 库

- Python3中可使用`urllib.request`和`requests`进行网页爬取
 - ✓ `urllib`库：python内置，无需额外安装，但使用不够方便。
 - ✓ `requests`库：第三方库，需安装。基于urllib3，采用Apache2 Licensed开源协议的HTTP库。`requests`库简单易用，比urllib更加方便。
- 课内使用`requests`库获取网页的HTML信息,安装方法：
 - ✓ IDLE：在cmd命令行中 `pip install requests`
 - ✓ Anaconda：在prompt中 `conda install requests`
- 检测：
 - ✓ 在交互式环境（IDLE、Spyder等）中输入 `import requests` 不报错，说明requests模块成功安装

引例扩展

(1).为了使用requests，需要首先引入requests库：

```
import requests
```

(2).使用requests来发送http请求，如get请求：

```
r = requests.get('http://www.baidu.com/')
```

(3).获得的Response对象r，即http请求的响应结果。
显示Response对象r的一些属性，如：

#请求的状态码

```
print(r.status_code)
```

#检测r的类型

```
print(type(r))
```

#获取请求信息的url

```
print(r.url)
```



status_code

headers

body

response

Response

- response对象包含了三部分主要信息

- 响应状态

r.status_code

200	404	502	301
成功	找不到页面	服务器错误	跳转页面

- 响应头(Response Header)

r.headers

- ✓ 包括内容类型，长度，服务器信息，设置Cookie等信息。

- 响应体(Response body)

r.text

- ✓ 主要包含请求资源的内容，如网页的HTML代码、图片的二进制数据等。

乱码?

从headers猜测response对象的编码方式。若header中不存在charset,则认为是ISO-8859-1

```
# 显示Response对象的编码方式  
print(r. encoding)
```

ISO-8859-1

根据网页内容分析出的编码方式。
原则来说更加准确。

```
# 显示Response对象的文本编码  
print(r. apparent_ encoding)
```

UTF-8

```
# 修改编码方式后重新显示  
r. encoding= r. apparent_ encoding  
print(r. text)
```

r.text根据r.encoding显示网页内容

Response对象的主要属性

属 性		说 明
r.status_code		http请求的返回状态，200表示成功，其他（如404）表示失败
r.text	处理过的Unicode型的数据	http响应内容的字符串形式，即url对应的页面内容
r.content	bytes型的原始二进制数据	http响应内容的二进制形式，即url对应内容的二进制形式
r.encoding		从http header中猜测的响应内容编码方式
r.apparent_encoding		从内容中分析出的响应内容编码方式，备选编码方式

得到的内容

- 网页文本
 - ✓ HTML文档、Json格式文本等
- 图片、音频、视频
 - ✓ 二进制文件，保存为相应的格式
- 其他

```
r=requests.get ('http://www.baidu.com/img/baidu_jgylogo3.gif')  
print(r.content) # 二进制文件使用content
```

#保存图片

```
with open('logo.gif','wb') as f:  
    f.write(r.content)  
print('Ok')
```

Requests库的异常类型

异常	说明
ConnectionError	网络连接出现错误， 如DNS查询失败、拒绝连接
HTTPError	HTTP协议层面出现错误
URLRequired	URL缺失造成的异常
TooManyRedirects	超过最大重定向次数产生异常
ConnectTimeout	连接远程服务器超时异常
Timeout	请求URL超时，产生超时错误

超时设置

- ✓ 利用 `timeout` 变量来配置最大请求时间，如果在`timeout`时间内请求内容没有返回，将产生一个`timeout`的异常。

```
import requests  
resp = requests.get('http://www.baidu.com', timeout=0.5)  
print(resp.status_code)
```

- 如果在指定时间内没有返回，就会抛出`timeout`异常。
- 如果远端服务器很慢，可以传入一个`None`作为`timeout`值，使得`requests`永远等待。
- `timeout` 仅对连接过程有效：只限制请求的时间，与响应体的下载无关。即：当返回的 `response` 包含很大内容，下载需要一定时间，与`timeout`没有关系。

连接、读取超时

```
requests.get('http:// www.baidu.com', timeout=(6.005, 0.01))
```

分别指定连接和读取的超时时间，服务器在指定时间没有应答，抛出

requests.exceptions.**ConnectTimeout / ReadTimeout**

- timeout=([连接超时时间], [读取超时时间])
- 连接：客户端连接服务器并发送http请求
- 读取：客户端等待服务器发送第一个字节之前的时间

请求不成功（状态码）

```
bad_r = requests.get('http://httpbin.org/status/404')
```

```
bad_r.raise_for_status()
```

如果 HTTP 请求返回了不成功的状态码（4XX客户端错误，或者5XX服务端错误）， Response.raise_for_status() 抛出 requests.exceptions.**HTTPError**

Requests库的异常处理

```
import requests
from requests.exceptions import ConnectTimeout, RequestException
try:
    response = requests.get('http://www.baidu.com', timeout=0.005)
    print(response.status_code)
except ConnectTimeout:          #超时异常
    print('timeout')
except RequestException:       #请求异常
    print('reqerror')
```

异常处理

当你不确定会发生什么错误时，
尽量使用try...except来捕获异常


```
import requests
url =
"http://vacations.ctrip.com/tour/detail/p12906212s2.html#ctm_ref=va_youxue_s
2_lst_p1_l2_1_txt"
try:
    r = requests.get(url, timeout=0.5)
    r.raise_for_status()
    print(r.text[:500])
except:
    print("爬取失败")
```

爬取网页的通用代码框架

- 通用代码框架可以有效处理和避免访问和爬取网页过程中可能出现的错误，提高访问和爬取的效率。

```
import requests                                #加载Requests库
def get_html(url):                             #定义get_html函数
    try:
        r=requests.get(url,timeout=40)         #设定get函数参数，超时限制40s
        r.raise_for_status()                  #有效判断网络连接状态，错误将捕获异常
        r.encoding = r.apparent_encoding      #设置编码格式
        return r.text                         #返回网页文本内容
    except:
        return "产生异常"                    #返回异常提示
if __name__ == "__main__":                    #直接运行时（非以模块导入）
    url = "http://www.baidu.com"              #给url赋值
    print(get_html(url))                      #打印函数内容
```

HTTP基本知识

- **URL**：统一资源定位符。HTTP协议一般采用url作为**定位网络资源**的标识

`http://[host]:[port]/[path]`

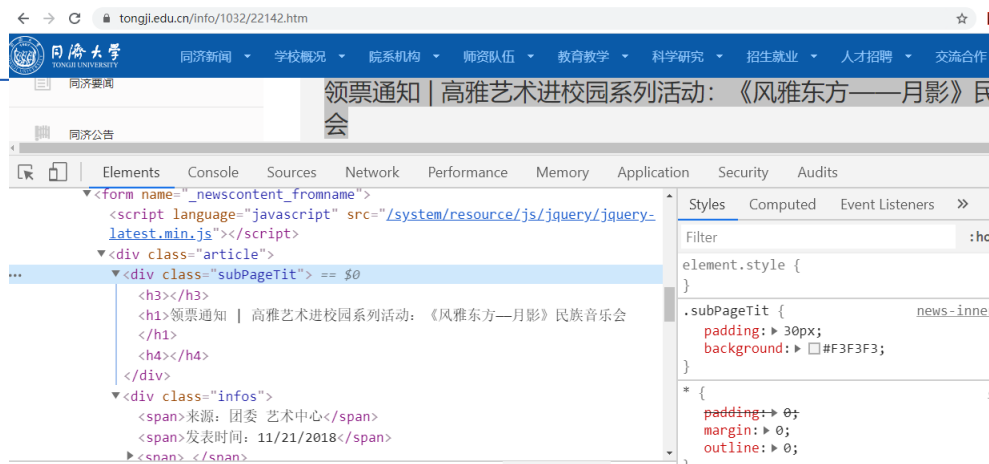
合法的internet主机，域名或ip地址

端口号，省略时默认端口号为80

资源在该主机或ip地址服务器上所包含的内部路径

`https://www.tongji.edu.cn/info/1032/22142.htm`

- **超文本**：HyperText



HTTP基本知识

- **HTTP**(HyperText Transfer Protocol)**超文本传输协议**。
- 用于将超文本数据从网络传输到本地浏览器而制定的传送协议
- 保证高效而准确地传送超文本文档

- 基于**请求与响应模式**的**无状态**的**应用层协议**。

用户发起请求，
服务器做相关响应

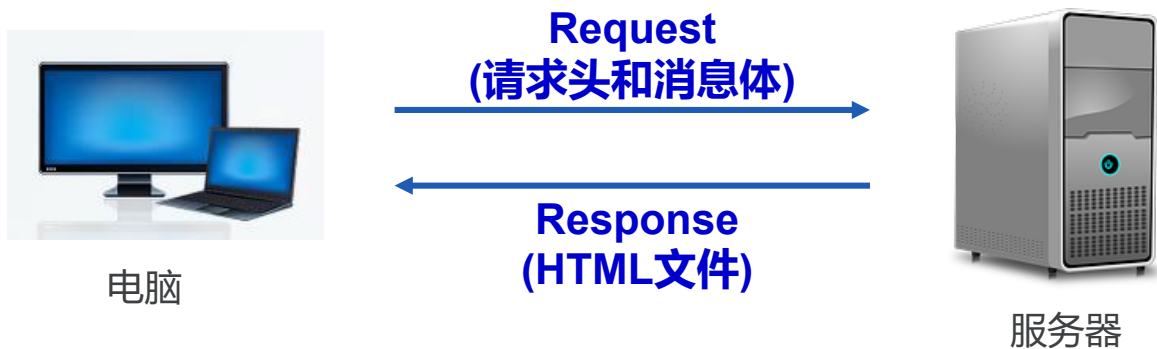
第一次请求跟第二次请求之
间，并没有相关的关联

协议工作在TCP
协议之上

- **HTTPS**(Hyper Text Transfer Protocol over Secure Socket Layer)**以SSL为安全基础的HTTP协议**

<https://www.tongji.edu.cn>

爬虫基本流程



- 1.发起请求
- 2.获取响应内容



Elements Console So **Network** Performance Memory Application Security Audits 4

Search X ☐ Preserve log ☐ Disable cache Online

Aa .* Search Filter ☐ Hide data URLs **All** XHR JS CSS Img Media Font Doc WS Manifest Other

☐ Has blocked cookies

Name	Method	Status	Type	Initiator	Size	Time	Waterfall
www.baidu.com	GET	200	docume...	Other	70.9 kB	447 ms	
super_min-fe4f97903e.css	GET	200	stylesheet	(index)	(disk ca...	5 ms	
weather-7bffde5120.css	GET	200	stylesheet	(index)	(disk ca...	5 ms	
card_min-03bfc881a3.css	GET	200	stylesheet	(index)	(disk ca...	5 ms	
bd logo1.png	GET	200	png	(index)	(memor...	0 ms	

62 requests | 130 kB transferred | 1.5 MB resources | Finish: 11.07 s | DOMContentLoaded: 752 ms | Load: 914 ms



Elements Console So **Network** Performance Memory Application Security Audits 4

Search X ☐ Preserve log ☐ Disable cache Online

Aa .* Search Filter ☐ Hide data URLs **All** XHR JS CSS Img Media Font Doc WS Manifest Other

☐ Has blocked cookies

Name	Method	Status	Type	Initiator	Size	Time	Waterfall
www.baidu.com	GET	200	docume...	Other	70.9 kB	447 ms	
super_min-fe4f97903e.css	GET	200	stylesheet	(index)	(disk ca...	5 ms	
weather-7bffde5120.css	GET	200	stylesheet	(index)	(disk ca...	5 ms	
card_min-03bfc881a3.css	GET	200	stylesheet	(index)	(disk ca...	5 ms	
bd logo1.png	GET	200	png	(index)	(memor...	0 ms	

62 requests | 130 kB transferred | 1.5 MB resources | Finish: 11.07 s | DOMContentLoaded: 752 ms | Load: 914 ms



baidu.com

上海: 20°C 优 43 | 换肤 消息

抗击

设为首页

关于百度

About Baidu

百度推广

使用百度前必读

意见反馈

帮助中心

隐藏卡片



Elements

Console

Sources

Network

Performance

Memory

Application

Security

Audits



Preserve log



Disable cache

Online



Filter



Hide data URLs

All

XHR

JS

CSS

Img

Media

Font

Doc

WS

Manifest

Other

Has

Name



www.baidu.com



super_min-fe4f97903e.css



weather-7bffde5120.css



card_min-03bfc881a3.css



bd_logo1.png



bd_logo1.png?qua=high



logo_top-e3b63a0b1b.png



baidu_resultlogo@2.png



loading-f9b765014b.gif?v=md5



default_b91837ae.png



init-5ceabe80c4.css



newmusic_min_1b1ebf56.css



jquery-1-edb203c114.10.2.js

84 requests | 161 kB transferred | 1.6 MB resource

General

Request URL

Request Method

Status Code

Remote Address

Referrer Policy

error-when-downgrade

Response Headers

Request Headers

Accept: text/html,application/xhtml+xml,application/xml;q=0.

Accept-Encoding: gzip, deflate, br

Accept-Language: zh-CN,zh;q=0.9,en;q=0.8

Cache-Control: max-age=0

Connection: keep-alive



Console

What's New



请求

- 请求是由客户端向服务端发出的，包括：**request method**请求方式；**request URL**请求网址；**request header**请求头；**request Body**请求体
- 浏览器发信息给该网址所在服务器，即向服务器提出请求。
- **请求方式**主要有GET、POST、HEAD、PUT、DELETE。
 - ✓ **GET请求**的请求参数会显示在URL链接的后面。
 - ✓ 如用百度搜索“同济”，请求的URL链接变为：
<https://www.baidu.com/s?wd=同济>
 - ✓ **POST请求**的请求参数会存放在Request内，并不会出现在 URL 链接的后面。如登录信息。

➤ 请求URL

- ✓ 统一资源定位符，即网址。URL可以确定一张网络图片、一个音频、一个网页文档等，其包含的信息指出了文件的位置以及浏览器如何处理它。

➤ 请求头（Headers）

- ✓ 包含请求时的头部信息，例如User-Agent（指定浏览器的请求头），Cookies、Host等。

➤ 请求体

- ✓ 额外携带的数据，例如登录表单提交的登录信息

HTTP操作方法

方法	说明
GET	向特定资源发起请求，请求页面，返回页面内容
HEAD	获取html网页头(类似于GET，但不返回网页具体内容)
POST	向html提交数据进行处理请求(如提交表单或上传文件)，数据包含在请求体中
PUT	从客户端向服务器传送的数据取代指定文档中的内容
PATCH	提交局部修改请求
DELETE	请求服务器删除指定的页面
OPTIONS	允许客户端查看服务器的性能

- HTTP协议，通过URL对资源做定位，通过常用的方法，对资源进行管理，每一次操作都是独立无状态的。
- 在HTTP协议的世界里，网络通道跟服务器都是黑盒子，能看到的就是url链接以及对url链接的相关操作。

Requests对象的方法

方 法	解 释
<code>requests.request()</code>	构造一个请求，支持以下各种方法
<code>requests.get()</code>	获取html的主要方法，对应http协议get
<code>requests.head()</code>	获取html头部信息
<code>requests.post()</code>	向html网页提交post请求
<code>requests.put()</code>	向html网页提交put请求
<code>requests.patch()</code>	向html提交局部修改的请求
<code>requests.delete()</code>	向html提交删除请求

- GET和POST是表单提交数据的两种基本方式
- GET是从服务器上获取数据，POST是向服务器传送数据
- GET请求数据通过域名后缀url传送，用户可见，不安全
- POST请求数据通过在请求报文正文里传输，相对比较安全

request方法

```
requests.request(method, url, **kwargs)
```

请求方式，
分别对应get等7种方法

页面链接

控制访问
参数(13个)

'GET'

'HEAD'

'POST'

'PUT'

'PATCH'

'DELETE'

'OPTIONS'

params: 增加到url中的参数

data: 向服务器提供或提交资源时使用

json: 作为内容部分向服务器提交json数据

headers: 向某url访问时发起的http的头字段

cookies: 从http协议中解析cookie

auth: 支持http认证功能

files: 向服务器传输文件使用

timeout: 设定的超时时间

proxies: 设定访问代理服务器

allow_redirects: 是否允许对url进行重定向

stream: 对获取的内容是否进行立即下载

verify: 认证ssl证书

cert: 本地ssl证书路径

其他方法的具体形式

```
requests.get(url, params=None, **kwargs)
```

```
requests.head(url, **kwargs)
```

```
requests.post(url, data=None, json=None, **kwargs)
```

```
requests.put(url, data=None, **kwargs)
```

```
requests.patch(url, data=None, **kwargs)
```

```
requests.delete(url, **kwargs)
```

Get方法

拟爬取的网站地址

url中的额外参数，字典或者字节序列，作为参数增加到 url 中

控制访问参数(12个)

```
requests.get(url, params= None, **kwargs)
```

获取html网页内容，返回一个response对象

➤ 基本GET请求

```
import requests  
r = requests.get('http://httpbin.org/get')  
print(r.text)
```

r =

```
requests.get("http://httpbin.org/get?name=Sharon&age=10")
```

- 在get请求中，经常使用params关键字，以一个字典来传递这些参数。
- 将一些键值对以?key1=value1&key2=value2的模式增加到url中。

Get方法

```
requests.get(url,params= None,**kwargs)
```

获取html网页内容，返回一个response对象

➤ 带参数的GET请求

#通过params，将data键值对增加到url

```
import requests
data = {'name' : 'Sharon', 'age' : 10}
r = requests.get('http://httpbin.org/get', params=data)
print(r.url)
```

等价于：r =
requests.get("http://httpbin.org/get?name=Sharon&age=10")

添加headers

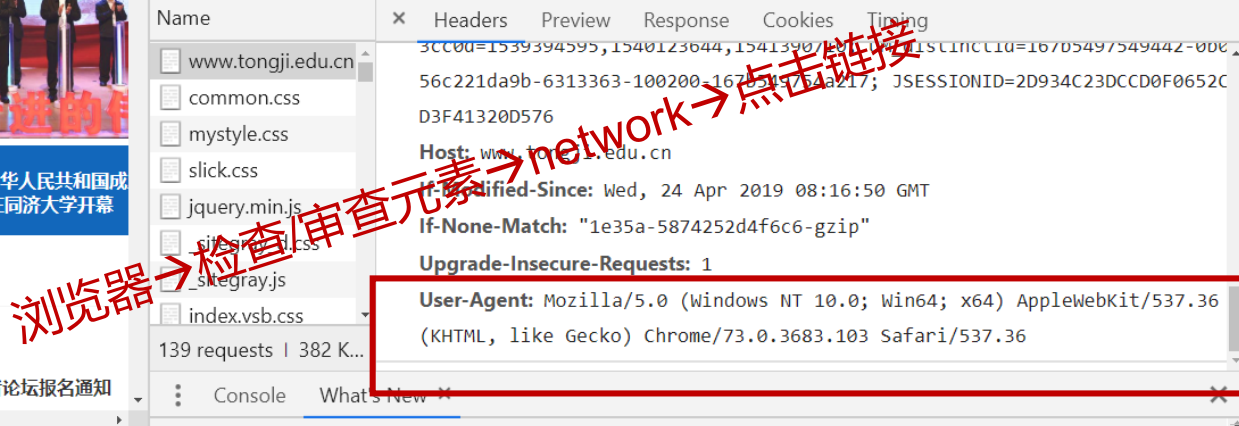
- 以防有些网站禁止爬虫访问，添加浏览器的User-Agent信息

```
import requests
```

```
headers = {'User-Agent': "Mozilla/5.0 (Windows NT 10.0;  
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/74.0.3729.108 Safari/537.36"}
```

```
r = requests.get('http://www.baidu.com', headers=headers)
```

```
print(r.text)
```



User-Agent: 向访问网站提供你所使用的浏览器类型及版本、操作系统及版本、浏览器内核、等信息的标识。

Headers信息

```
import requests  
r = requests.get("http://httpbin.org/get")  
print(r.request.headers['User-Agent'])
```

```
{'User-Agent': 'python-requests/2.18.4', 'Accept-Encoding': 'gzip,  
deflate', 'Accept': '*/*', 'Connection': 'keep-alive'}
```

在请求中指定请求代理:

```
headers = {'User-Agent': 'Sharon'}  
r = requests.get("http://httpbin.org/get",  
headers=headers)  
print(r.request.headers)
```

```
{'User-Agent': 'Sharon', 'Accept-Encoding': 'gzip, deflate', 'Accept':  
'*/*', 'Connection': 'keep-alive'}
```


知乎页面

```
import requests  
r = requests.get("https://www.zhihu.com/explore")  
print(r.text)
```

```
import requests  
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0;  
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/81.0.4044.122 Safari/537.36'}  
r = requests.get("https://www.zhihu.com/explore",  
headers=headers)  
print(r.text)
```

<https://www.zhihu.com/robots.txt>

传递关键字爬取百度新闻搜索

```
keyword = input("input keyword:")  
url = "http://news.baidu.com/ns"  
kv = {"word" : keyword}  
headers = {"User-Agent": "Mozilla/5.0 (Windows NT 10.0;  
WOW64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/63.0.3239.84 Safari/537.36"}  
r=requests.get(url,timeout=30,params=kv,headers=hea  
ders)  
print(r.text)  
print(r.url)
```

字典，字节序列或文件对象

POST方法

拟爬取的网站地址

JSON格式的数据

```
requests.post(url,data = None,json = None,**kwargs)
```

向服务器传送数据

➤ 基本post请求

在requests中，以form表单形式发送post请求，只需将请求的参数构造成一个字典，然后与params不同的是，data提交的数据并不接在url

链接里，而是放在url对应位置的地方（例如form里）作为数据来存储。

```
import requests
data = {"name": "yiqing", "age": 18}
response = requests.post("http://httpbin.org/post", data=data)
print(response.text)
```

```
"args": {},
"data": "",
"files": {},
"form": {
  "age": "18",
  "name": "yiqing"
}
```

POST方法

```
requests.post(url,data = None,json = None,**kwargs)
```

✓ 带参数post请求

在requests中，以form表单形式发送post请求，只需将请求的参数构造成一个字典，然后传给 data参数

```
import requests
data = {"name": "Sharon", "age": 10}
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/63.0.3239.84 Safari/537.36"}
response = requests.post("http://httpbin.org/post", data=data,
headers=headers)
print(response.text)
```

文件上传

➤ 直接用 files 参数

文件上传：用open方式打开文件，与file形成键值对，定义成字典，并将其与files做关联，同时对应到相关url。以之向链接提交文件。

建立Utf-8 格式的txt文件: test.txt并上传

```
import requests
url = 'http://httpbin.org/post'
fs = {'file': open('G:/test.txt', 'rb')}
r = requests.post(url, files=fs)
print(r.text)
```

```
"files": {
  "file": "Hello,Tongji\r\n"
},
```

上传图片文件logo.gif

```
import requests
fs = {'file': open('logo.gif', 'rb')}
resp = requests.post('http://httpbin.org/post', files=fs)
print(resp.text)
```

网络爬虫按尺寸分类

- 如Google、百度，作用是建立全依赖的搜索引擎，规模很大。
- 需要一个能够爬取全internet所有资源的爬虫，对爬取速度要求非常高，必须定制开发。

全internet

- 网站上内容很多，对应的数据量大，爬取速度必须足够快，网站本身数据更新的速度。用例如Scrapy之类更为常见。

网站

- 规模很小，获取网络的数据量也很小，对爬取速度要求不高。

requests库实现。或者一系列网页能够完成的作用，最为常见。

网页

网络爬虫引发的问题

对服务器性能的骚扰

- 服务器默认按照**人数**来约定访问能力
- 网络爬虫用**计算机快速运算能力**（十万/s~几十万/s）去爬取网站内容以获取相关资源，对网站来说形成骚扰，为服务器带来巨大资源开销

内容层面的法律风险

- 网站服务器上的**数据**往往有产权归属
- 如果爬虫爬取网站提供的**数据**以进行**牟利**，有可能会带来法律上的风险

个人隐私泄露

- 网络爬虫具备一定的**突破能力**
- 如果爬取网站上受保护的**个人隐私数据**，将给个人带来很大的隐私泄露风险。

对网络爬虫的限制

- 服务器或者网站的所有者可通过**来源审查**限制网络爬虫。

判断所有请求网络链接**HTTP头部**的**user-agent**字段，对于不是预定的浏览器可以限制其访问。

- 发布**Robots协议**，通知所有的爬虫本网站可爬取的策略和规则。

但是，Robots协议仅仅是通过发布来体现，至于是否遵守仍是由网络爬虫自身决定。

Robots协议 (Robots Exclusion Protocol 网络爬虫排除标准)

- **作用：**通知网络爬虫本网站可爬取的策略和规则，哪些页面可以抓取，哪些不允许
- 访问网站时**首先将检查协议文件**，以确定访问的范围
User-agent 表明的是爬虫来源，如果是*代表所有的爬虫；
Disallow 代表不允许这个爬虫访问的资源目录或网址；
Allow代表允许该爬虫访问的资源目录或网址。

<https://www.jd.com/robots.txt>

User-agent: EtaoSpider
Disallow: /

针对EtaoSpider
禁止访问任何部分

User-Agent: *
Disallow: /pop/*.html
Disallow: /?*

针对所有爬虫
禁止访问pop目录下所有html网址
禁止访问网站中所有包含问号开头 (?) 的网址

优化

- 异常处理？文件夹或路径存在？文件没有关闭？不能用原始文件名保存图片？

```
import requests
import os
url =
"https://gd2.alicdn.com/imgextra/i2/2895983329/O1CN011aSiyVlqb4ya7wU_!!2895983329.jpg"
root = "e://"
path = root + url.split('/')[-1]
try:
    if not os.path.exists(root):
        os.mkdir(root)
    if not os.path.exists(path):
        r = requests.get(url)
        with open(path, 'wb') as f:
            f.write(r.content)
        print("文件保存成功")
    else :
        print("文件已存在")
except:
    print("爬取失败")
```

课后实验

- 爬取百度/必应首页html代码，显示response对象的text内容
 - ✓ 观察response对象的状态码、编码方式和http header
 - ✓ 修改header，并传递参数到url
 - ✓ 通过传递关键字“ 同济大学”，爬取搜索的结果
 - ✓ 选取一张结果图片，将其保存到本地
- 对于用户输入的任意关键词，获取 360 搜索网站 (<https://www.so.com/>)
 - ✓ 任意搜索一个关键词KeyWord，然后观察其url地址。
 - ✓ 根据用户输入的关键词，设置好当前url，使用requests库的get()方法实现对指定url的请求，并观察response响应对象的url链接和text内容。

政府工作报告-词云绘制图

- 1.引入第三方库
- 2.爬取网页文件
- 3.词云绘制



#引入第三方库

```
import requests
```

```
from bs4 import BeautifulSoup
```

```
from wordcloud import WordCloud,STOPWORDS
```

```
import matplotlib.pyplot as plt
```

```
import numpy as np
```

```
import PIL.Image as Image #图像处理
```

#爬取网页文件

```
url="http://www.gov.cn/premier/2022-03/12/content_5678750.htm"
```

```
r=requests.get(url)
```

```
r.encoding = r.apparent_encoding
```

```
soup=BeautifulSoup(r.text,"html.parser")
```

```
content=soup.find("div", class_="pages_content").text
```

```
print(content)
```


#词云图绘制

```
mask = np.array(Image.open(r'五角星.png'))
```

#通过generate(content)方法，将文本content生成一个WordCloud对象

```
w=WordCloud(# 设置字体格式
```

```
font_path='C:/Windows/Fonts/simhei.ttf',
```

```
background_color='white',#默认为黑色
```

```
stopwords=STOPWORDS.add(“各位代表”),#过滤词汇
```

```
mask=mask, # 设置遮罩图片
```

```
max_words=1000, # 最多显示词数
```

```
max_font_size=40) # 字体最大值
```

```
c=w.generate(content)
```



```
c.to_file("pic1.png") #将词云输出为图像文件
```

```
plt.subplot(111)
```

```
plt.imshow(c) # 显示词云图
```

```
plt.axis('off') # 不显示坐标轴
```

```
plt.show() # 显示图像
```