

СКРАПИНГ

28.10.2020

Что такое скрапинг

Автоматизированный сбор данных из Интернета

Раньше:

анализ интерфейсных данных,
интеллектуальный анализ данных,
сбор веб- данных

В настоящее время:

Любой сбор данных, кроме сбора данных
с использованием человека+ браузера

Задачи для скраперов

- ▶ Сбор HTML данных с домена
- ▶ Парсинг данных для получения информации
- ▶ Хранение информации
- ▶ Перемещение по страницам

Соединение с Интернетом

A → TCP IP → B

Локальный маршрутизатор Боба получает эту последовательность и интерпретирует ее как пакет с помощью собственного MAC-адреса и направляет на IP-адрес Алисы. Маршрутизатор заменяет в заголовке пакета обратный адрес на свой и посылает пакет дальше.

Пакет Боба проходит несколько промежуточных серверов, которые направляют его по правильному физическому/проводному пути на сервер Алисы.

Сервер Алисы получает пакет на свой IP-адрес.

Сервер Алисы считывает порт назначения пакета (почти всегда это порт 80 для веб-приложений, это что-то вроде «номера квартиры» в пакетной передаче данных, где IP-адрес является «улицей») в заголовке и передает его в соответствующее приложение – приложение веб-сервера.

Веб-сервер принимает поток данных от серверного процессора.

Эти данные говорят что-то вроде:

- это GET-запрос;
- запрашивается следующий файл: index.html.

Веб-сервер находит соответствующий HTML-файл, записывает его в новый пакет для отправки Бобу и посылает его через свой локальный маршрутизатор обратно на компьютер Боба точно таким же вышеописанным способом.

Python

```
from urllib.request import urlopen  
html = urlopen("http://pythonscraping.com/pages/page1.html")  
print(html.read())
```

```
$python scrapetest.py
```

```
from urllib.request import urlopen
```

3.9

cpython / Lib / urllib /

Go to file

This branch is 656 commits ahead, 1085 commits behind master.

Pull request Compare



miss-islington and ronaldoussoren bpo-41471: Ignore invalid prefix lengths in system proxy settings on ...

✓ aa7b03b 7 дней назад History

..



__init__.py

Make a new urllib package .

13 лет назад



error.py

bpo-30022: Get rid of using EnvironmentError and IOError (except test... (

4 года назад



parse.py

bpo-39481: PEP 585 for a variety of modules (GH-19423)

7 месяцев назад



request.py

bpo-41471: Ignore invalid prefix lengths in system proxy settings on ...

7 дней назад



response.py

bpo-12707: deprecate info(), geturl(), getcode() methods in favor of ...

14 месяцев назад



robotparser.py

bpo-35922: Fix RobotFileParser when robots.txt has no relevant crawl ...

17 месяцев назад

BeautifulSoup

```
$pip install beautifulsoup4
```

```
>python setup.py install
```

```
$ virtualenv scrapingEnv
```

```
▶ $ cd scrapingEnv/
```

```
▶ $ source bin/activate
```

```
(scrapingEnv)ryan$ pip install beautifulsoup4  
(scrapingEnv)ryan$ python  
> from bs4 import BeautifulSoup
```


Запуск

- ▶ `from urllib.request import urlopen`
- ▶ `from bs4 import BeautifulSoup`
- ▶ `bsObj = BeautifulSoup(urlopen("http://www.pythonscraping.com/pages/page1.html"))`
- ▶ `print(bsObj.h1)`

<h1>An Interesting Title</h1>

ВОЗМОЖНЫЕ ОШИБКИ

- ▶ Страница не найдена
- ▶ 404 Page Not Found
- ▶ Сервер не найден
- ▶ 500 Internal Server Error

```
if html is None:  
    print("URL is not found")  
else:  
    #программа продолжает работу
```

```
try:  
    html = urlopen("http://www.pythonscraping.com/pages/page1.html")  
except HTTPError as e:  
    print(e)  
    #возвратить null, прервать или выполнять операции по "Плану B"  
else:  
    #программа продолжает работу. Примечание: если возвращается null или прерывается в #exception catch, оператор "else" использовать не нужно
```



```
try:
    badContent = bsObj.nonExistingTag.anotherTag
except AttributeError as e:
    print("Tag was not found")
else:
    if badContent == None:
        print ("Tag was not found")
    else:
        print(badContent)
```



```
from urllib.request import urlopen
from urllib.error import HTTPError
from bs4 import BeautifulSoup
def getTitle(url):
    try:
        html = urlopen(url)
    except HTTPError as e:
        return None
    try:
        bsObj = BeautifulSoup(html.read())
        title = bsObj.body.h1
    except AttributeError as e:
        return None
    return title
title = getTitle("http://www.pythonscraping.com/pages/page1.html")
if title == None:
    print("Title could not be found")
else:
    print(title)
```


Как заглянуть в HTML код

- ▶ `bsObj.findAll("table") [4].findAll("tr")[2].find("td").findAll("div")[1].find("a")`

Скрапер

```
"<span class="red">Heavens! what a virulent attack!</span>" replied <span class="green">the prince</span>, not in the least disconcerted by this reception.
```

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
html = urlopen("http://www.pythonscraping.com/pages/warandpeace.html")
bsObj = BeautifulSoup(html)
```

```
nameList = bsObj.findAll("span", {"class":"green"})
for name in nameList:
    print(name.get_text())
```


find() и findAll()

```
findAll(tag, attributes, recursive, text, limit, keywords)  
find(tag, attributes, recursive, text, keywords)
```

```
findAll({"h1", "h2", "h3", "h4", "h5", "h6"})
```

tag — именованная метка, читается /tæg/; более правильное название — дескриптор). В SGML (**HTML**, WML, AmigaGuide, языках семейства XML) — элемент языка разметки гипертекста. Текст, содержащийся между начальным и конечным **тегом**, отображается и размещается в соответствии со свойствами, указанными в начальном теге.

```
nameList = bsObj.findAll(text="the prince")  
print(len(nameList))  
  
allText = bsObj.findAll(id="text")  
print(allText[0].get_text())
```



```
bsObj.tag.subTag.anotherSubTag
```

В целом функции BeautifulSoup всегда работают с потомками выбранного тега. Например, `bsObj.body.h1` выбирает первый тег `h1`, который является потомком тега `body`. Она не найдет теги, расположенные вне `body`.

Аналогично `bsObj.div.findAll("img")` найдет первый тег `div` в документе, а затем извлечет все теги `img`, которые являются потомками тега `div`.

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen("http://www.pythonscraping.com/pages/page3.html")
bsObj = BeautifulSoup(html)

for child in bsObj.find("table", {"id": "giftList"}).children:
    print(child)
```



```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import datetime
import random
import re

random.seed(datetime.datetime.now())
def getLinks(articleUrl):
    html = urlopen("http://en.wikipedia.org"+articleUrl)
    bsObj = BeautifulSoup(html)
    return bsObj.find("div", {"id":"bodyContent"}).findAll("a",
                                                            href=re.compile("^(/wiki/)((?!:).)*$"))
links = getLinks("/wiki/Kevin_Bacon")
while len(links) > 0:
    newArticle = links[random.randint(0, len(links)-1)].attrs["href"]
    print(newArticle)
    links = getLinks(newArticle)
```


Краулинг сайта

- Извлечение всех страниц сайта

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

pages = set()
def getLinks(pageUrl):
    global pages
    html = urlopen("http://en.wikipedia.org"+pageUrl)
    bsObj = BeautifulSoup(html)
    try:
        print(bsObj.h1.get_text())
        print(bsObj.find(id="mw-content-text").findAll("p")[0])
        print(bsObj.find(id="ca-edit").find("span").find("a").attrs['href'])
    except AttributeError:
```



```
print("This page is missing something! No worries though!")
for link in bsObj.findAll("a", href=re.compile("^(/wiki/)")):
    if 'href' in link.attrs:
        if link.attrs['href'] not in pages:
            #Мы получили новую страницу
            newPage = link.attrs['href']
            print("-----\n"+newPage)
            pages.add(newPage)
            getLinks(newPage)
getLinks("")
```


ДОПОЛНИТЕЛЬНЫЕ ВОЗМОЖНОСТИ

- ▶ `pip install --upgrade oauth2client`

- ▶ [Работа с Google](#)

Простой метапоисковый алгоритм на Python

```
from bs4 import BeautifulSoup
import requests
query = input('What are you searching for?:  ')
url = 'http://www.google.com/search?q='
page = requests.get(url + query)
soup = BeautifulSoup(page.text)
h3 = soup.find_all("h3",class_="r")
for elem in h3:
    elem=elem.contents[0]
    link=("https://www.google.com" + elem["href"])
    print(link)
```

```
from bs4 import BeautifulSoup
import requests
query = input('What are you searching for?:  ')
number = input('How many pages:  ')
url = 'http://www.google.com/search?q='
page = requests.get(url + query)
for index in range(int(number)):
    soup = BeautifulSoup(page.text)
    next_page=soup.find("a",class_="fl")
    next_link=("https://www.google.com"+next_page["href"])
    h3 = soup.find_all("h3",class_="r")
    for elem in h3:
        elem=elem.contents[0]
        link=("https://www.google.com" + elem["href"])
        print(link)
    page = requests.get(next_link)
```



```
from bs4 import BeautifulSoup
import requests
dict=""
query = input('What are you searching for?: ' )
url = 'http://www.google.com/search?q='
page = requests.get(url + query)
soup = BeautifulSoup(page.text)
h3 = soup.find_all("h3",class_="r")
for elem in h3:
    elem=elem.contents[0]
    elem = elem["href"]
    if "wikipedia" in elem:
        link=("https://www.google.com" + elem)
        break
page = requests.get(link)
soup = BeautifulSoup(page.text)
text = soup.find(id="mw-content-text")
p= text.find("p")
while p != None:
    dict+=p.get_text()+"\n"
    p = p.find_next("p")
dict=dict.split()
```