Python&Spider Training

https://github.com/lvyufeng/python_training.git

07/13/2018

Contents

- **Python programming skills**
- **Web content analysis method**
- **How to build a crawler**
- **Anti-crawler and anti-anti crawler**

Python programming skills

Python programming skills

- **File I/O**
- **String processing**
- **Data structure**
- **Iterator**
- **Multithreading**
- **Object oriented**

File I/O

- open file method: open(name[,mode,encoding,buf])
- name: file path
- mode: open mode
- encoding: encoding format
- buf: buffering size

File I/O

- **file read mode:**
- **read([size]): read file([size] byte, read all by default)**
- **readline([size]): read a line**
- **readlines([size]): read all the lines, return a list**

File I/O

- **file write mode:**
- **write(str): write a string to the file**
- **writelines(sequence_of_strings): write multiple lines to the file, parameter could be an iterable objecet**

File I/O

mode	description	note
'r'	read only mode	file must exist
'w'	write only mode	create file if not exist clear file if exist
'a'	append mode	create file if not exist
'r+'/'w+'	write&read mode	
'a+'	write&read&append mode	
'rb','wb','ab','rb+','wb+','ab+':open in binary mode		

File I/O

- **Python file pointer:**
- **seek(offset[,whence]): movable file pointer**
- **offset: could be negetive**
- **whence: offset relative position**

File I/O

- **Python file pointer relative position :**
- **os.SEEK_SET: relative to the starting position**
- **os.SEEK_CUR: relative to current position**
- **os.SEEK_END: relative to the ending position**

String processing

- **str.split(sub_str): string segmentation**
- **str.join(sub_str): string concatenation**
- **str.strip(sub_str)/str.lstrip(sub_str)/str.rstrip(sub_str): remove the first and last (first or last) substrings**
- **str.replace(sub_str,new_str): string replacement**

Data structure

- **List**
- **Dictionary**
- **Set**
- **Tuple**

- **collections——enhanced data structure**

List

```
L = [item,.....]                  #using [] to init a list
L.append(item)                    #append item at the end
L.insert(position, item)            or insert item at a position
L.pop(position)                   #delete the last item or item at the
                                    specified position
L[index] = item                   #replace item
```

Tuple

```
t = ('Adam', 'Lisa', 'Bart')
```

Tuples cannot be changed once defined

Dictionary

Orderless key-value sequence pair

```
d = { 'Adam': 95, 'Lisa': 85, 'Bart': 59 }    #init dict
d['Paul'] = 72                                #add or update item
d.pop(' 'Adam ')                              #remove item
```

Set

A Set is a collection of unorderable hashable values

```
s = set([1, 2, 3])          #define a set
s.add(4)                    #add item
s.remove(4)                 #delete item
```

collections

from collections import *

- namedtuple: set an alias for each location of the tuple
- defaultdict: a key of the corresponding type is automatically generated when the key does not exist.
- deque: a double-ended queue
- Counter: counter of basic data structure
- OrderedDict: ordered dictionary
- ChainMap: multiple maps form a new unit

Iterator

```
l = [1, 2, 3, 4]
i = l.__iter__() # iter(l)
print(i.next())
```

Multithreading

```
from queue import Queue
import threading
detail_url_queue = Queue(maxsize=1000)
for i in range(10):
        html_thread = threading.Thread(target=get_detail_html,
args=(detail_url_queue,))
        html_thread.start()
detail_url_queue.task_done()
detail_url_queue.join()
```

Object oriented

- **Class**
- **Attribute**
- **Method**
- **Inheritance**
- **Polymorphism**
- **Magic method**

Class

```
class ClassName(object):                #define class
        def __init__(self,[...]):      #constructor
                pass
        def __del__(self,[...]):        #destructor
                pass
```

Attribute

```
class Test(object):
        sex = 'male'                                    # direct definition
        def __init__(self,name,age,weight):
                self.name = name                # public
                self._age = age                 # private(fake)
                self.__weight = weight          # indirect access
```

Method

```
class Test(object):
        #@classmethod  called by class name
        #@property       called like property
        def add(self):
                pass
        def _minus(self):
                pass
        def __multiply(self):
                pass
```

Inheritance

```
class A(object):
        def method(self,arg):
                pass
class B(A):
        def method(self,arg):
                super(B,self).method(arg)
```

Polymorphism

- Inheritance
- Method rewriting

```
class A(object):
        def method(self,arg):
                print('arg in A is %s' %(arg) )
class B(A):
        def method(self,arg):
                print('arg in B is %s' %(arg) )
```

Magic method——Instantiation

- __new__(cls): create a object of class
- __init__(self): init a object
- __del__(): recycle object

Magic method——Operator

- __cmp__(self,other): comparison operator
- __eq__(self,other)
- __lt__(self,other)
- __gt__(self,other)
- __add__(self,other): mathematical operator
- __sub__(self,other)
- __mul__(self,other)
- __div__(self,other)
- __or__(self,other): logical Operators
- __and__(self,other)

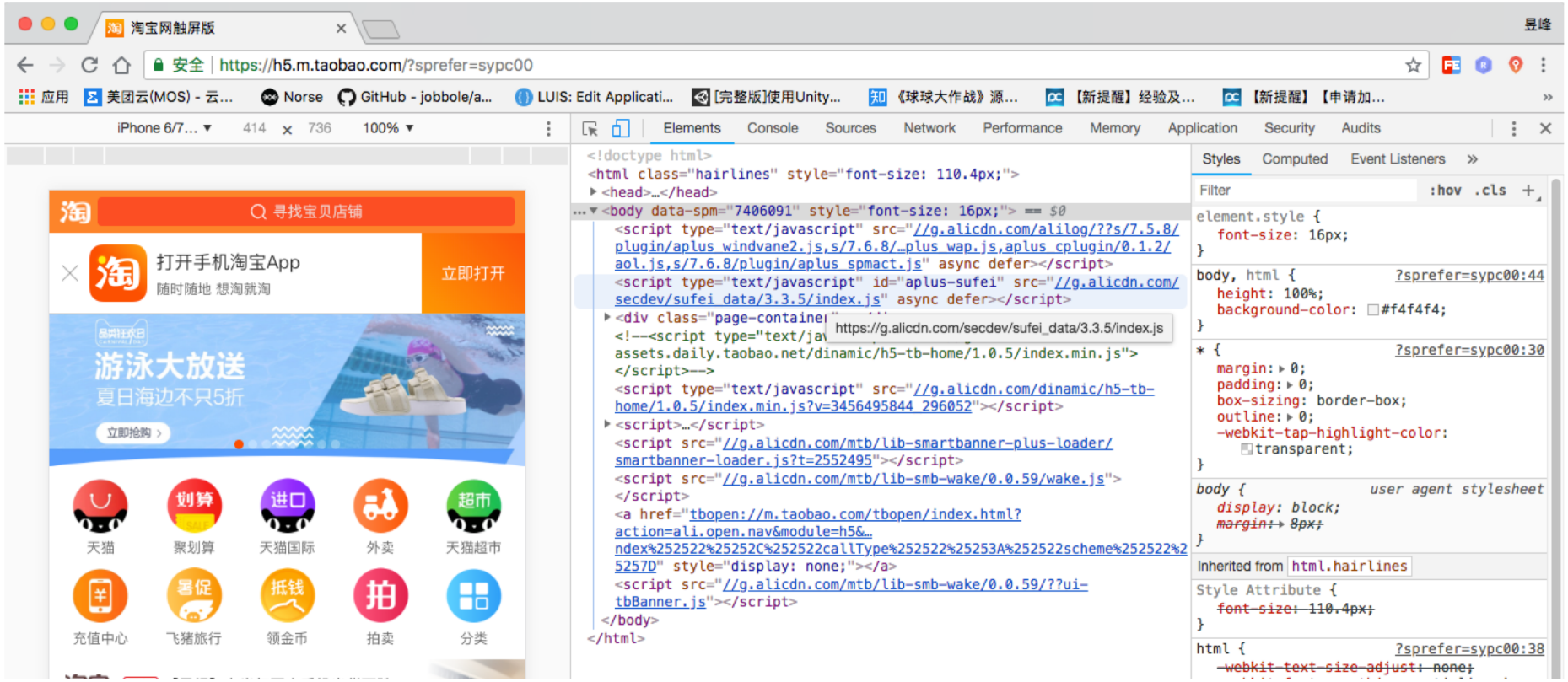
Web content analysis method

Web content analysis method

- **Using the developer mode of Chrome**
- **HTML source code analysis**
- **Network monitoring**

Using the developer mode of Chrome

- **This part of the content is operated on site.**



HTML source code analysis

- **CSS**
- **XPath**
- **selector**
- **Regular expression?**

Network monitoring

- **find API**
- **URL parse**
- **choose useful form parameter**

How to build a crawler

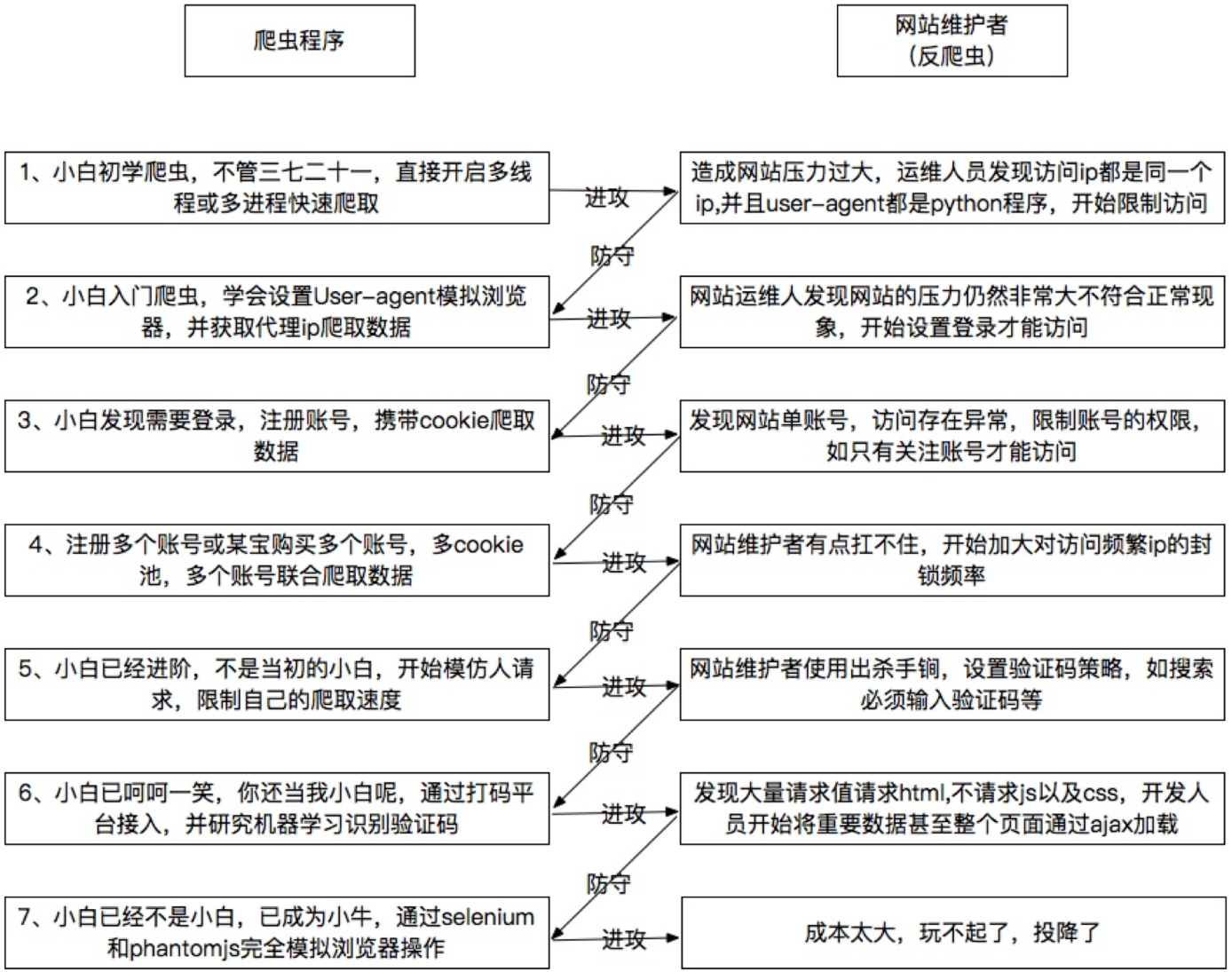
How to build a crawler

- **Static web content: requests + Beatifulsoup**
- **API: requests**
- **Dynamic web content: selenium + chromedriver**

- **Best framework: Scrapy**

Anti-crawler and anti-anti crawler

Anti-crawler and anti-anti crawler



Anti-crawler and anti-anti crawler

- **Old solution: selenium + chromedriver**
  - **have verification code**
  - **too slow**

- **Our solution:**
  - **Taobao,Tmall: use api directly**
  - **JD,suning: api + static web content**
  - **Fast crawl: multithreading + proxy pool**

- **Backup solution:**
  - **Scrapy + mobile web content**