Crawler as a service

Wynne 2020/2/8

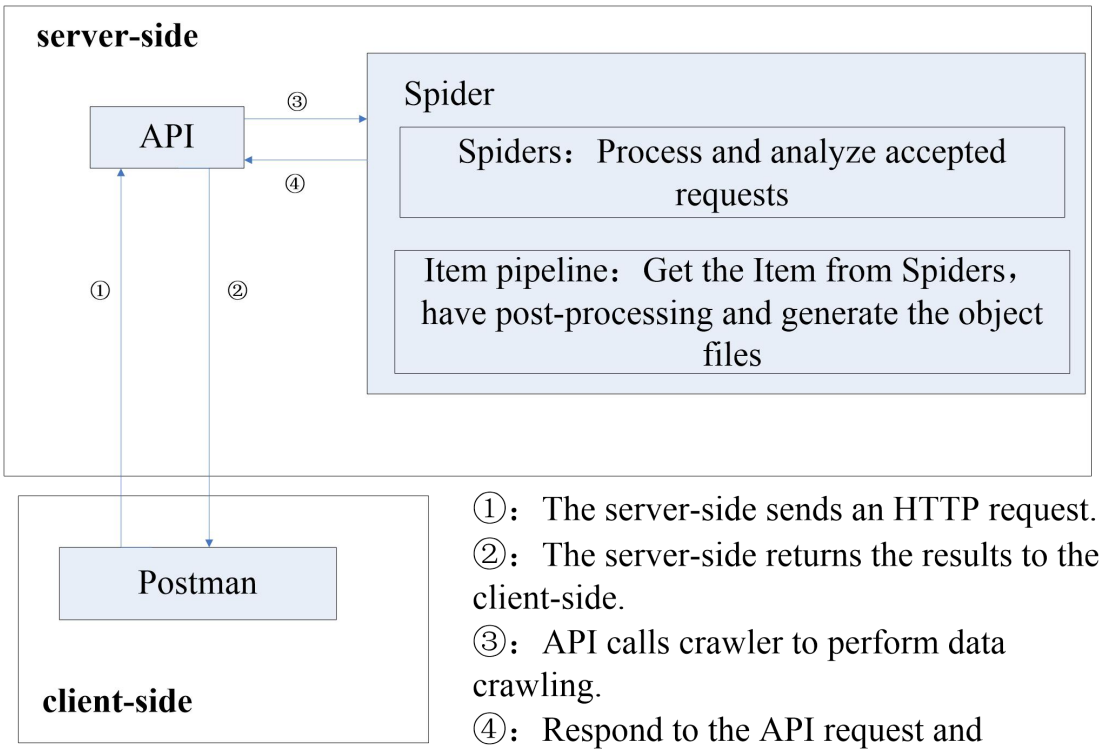
Overview

Goal

The purpose of this project is mainly to improve my skills, supplemented by in-depth understanding of python project, and to conduct in-depth research and expansion of crawler project with my tutor, mainly including learning flask architecture and establishing an API server.

Overall architecture

The system is divided into two important parts, API and crawler.From the structure can be divided into the client side and the server side.The client includes curl and postman to send an HTTP request to the server.The server side includes the dbmovie crawler section and the API server section.This system main operation process is as follows: the postman/curl from client sends an HTTP request to the API server has been deployed successfully, the server by calling the crawler command to execute the crawler task, will download code via pipelines for processing, to generate the target file, and then through the server returns a result, will be returned to the client and displays a crawler content.As shown in figure 1.



①：The server-side sends an HTTP request.
②：The server-side returns the results to the client-side.
③：API calls crawler to perform data crawling.
④：Respond to the API request and generate the results.

figure 1

Demo video

http://www.iqiyi.com/w_19sb2b9fgh.html

Crawler

Development environment set up

scrapy installation process：

1) Installation of scrapy frame

Dos commands are as follows:(Install the Twisted library first, or you'll get an error)

$ pip install Twisted-18.4.0-cp36-cp36m-win_amd64.whl

$ pip install -U scrapy

2) Project creation

The commands are as follows:

$ scrapy startproject DoubanMovie

You can use the following commands to view the project folder structure:

$ tree /f

Project folder structure description:

items.py: encapsulated entity class for collecting data
pipelines. py: handling of collected data
settings. py: framework core configuration file
spiders: crawler main script file

3) Create crawler script

$ scrapy genspider dbmovie https://movie.douban.com/top250

4) Test the website connection, at this point there will be 403 errors, need to set the Header Header information.

$ scrapy shell https://movie.douban.com/top250

At this point, we need to create a new one called "rotate_useragent.py" in the DoubanMovie directoryand tp implement automatic random selection in it.

The code is as follows:

```
class RotateUserAgentMiddleware(UserAgentMiddleware):
def __init__(self, user_agent=''):
self.user_agent = user_agent
def process_request(self, request, spider):
ua = random.choice(self.user_agent_list)
if ua:
print(ua)
request.headers.setdefault('User-Agent', ua)
```

And make changes in the "setting.py" file to configure rotate_useragent.py into the framework

```
DOWNLOADER_MIDDLEWARES = {
'DoubanMovie.middlewares.DoubanmovieDownloaderMiddleware': 543,
'scrapy.contrib.downloadermiddleware.useragent.UserAgentMiddleware':None,
'DoubanMovie.rotate_useragent.RotateUserAgentMiddleware':400
}
```

At this point, the browser can be accessed normally with shell view (hint: 200)

5) Set "items.py" to determine the collection data object

The code is as follows:

```
import scrapy
class DoubanmovieItem(scrapy.Item):
rank = scrapy.Field()
name = scrapy.Field()
pass
```

6) Write "dbmovie.py" to parse HTML tags to get data

The code is as follows:

```
class DbmovieSpider(scrapy.Spider):
name = 'dbmovie'
allowed_domains = ['douban.com']
start_urls = ['https://movie.douban.com/top250/']
def parse(self, response):
currentPage_movie_item = response.xpath('//div[@class = "item"]')
for movie_item in currentPage_movie_item:
```

```
        movie = DoubanmovieItem()
         movie['rank'] = movie_item.xpath('div[@class = "pic"]/em/text()').extract()
        movie['name'] = movie_item.xpath('div[@class = "info"]/div[@class = "hd"]/a/
span[@class = "title"][1]/text()').extract()
         yield movie
         nextPage=response.xpath('//span[@class="next"]/a/@href')
         if nextPage:
        url=response.urljoin(nextPage[0].extract())
         yield scrapy.Request(url,self.parse)
```

7) Compile "pipelines.py" to set up console output

The code is as follows:

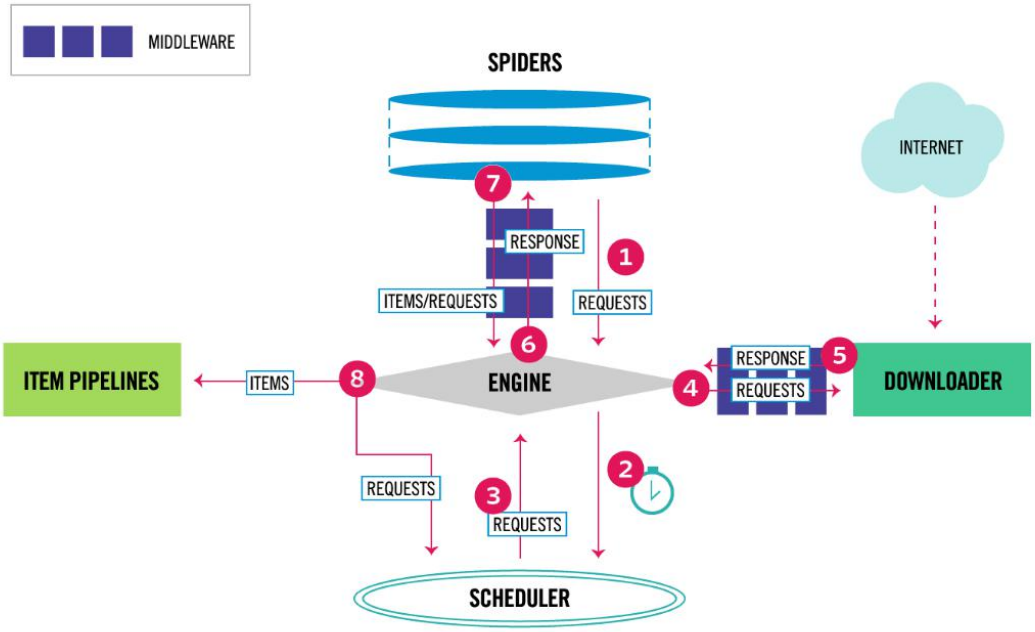
```
class DoubanmoviePipeline(object):
def __init__(self):
path = 'D:\codeSavePath\production practice\DoubanMovie\TOP250.csv'
self.file = open(path,'a+',encoding='utf-8')
self.writer = csv.writer(self.file)
def process_item(self, item, spider):
print(item)
print('电影排名:{0}'.format(item['rank'][0]))
self.writer.writerow((item['rank'],item['name']))
return item
def close_spider(self,spider):
self.file.close()
```

8) Run

```
$ scrapy crawl dbmovie
```

Crawler background

The frame diagram of crawler is as follows, this diagram is from:
https://www.cnblogs.com/miaoning/p/11626563.html



The main components of a crawler include：

- **Engine**:The engine is responsible for controlling the flow of data between all components of the system and triggering events when something happens.
- **Scheduler**:The scheduler receives requests from the engine and queues them for later output (including the engine).
- **Downloader**:The Downloader crawls the web page and sends the crawled data to the engine, which then sends it to the Spiders.
- **Spiders**:For the processing of downloaded data.
- **Item Pipeline**:A data-related component responsible for cleanup, validation, and persistence (such as storing items in a database).
- **Downloader middlewares:**Used for information transfer between the engine and the loader.

Crawler processing steps:

1. SPIDERS send an initial request to the ENGINE;
2. The engine arranges the current request in the SCHEDULER and asks the next request to enter the engine;
3. The scheduler returns a request to the engine (current request);
4. the engine sends the request to the DOWNLODER through the download middleware;
5. once the page is downloaded, the loader generates a response (about the page) and sends the response to the engine through the download middleware;
6. the engine receives the response from the loader and sends the response to the crawler through the crawler middleware;
7. the crawler processes the response and returns the tailored data to the engine with a new request through the crawler middleware;
8. the engine sends processing items to the data for permanent data preservation,

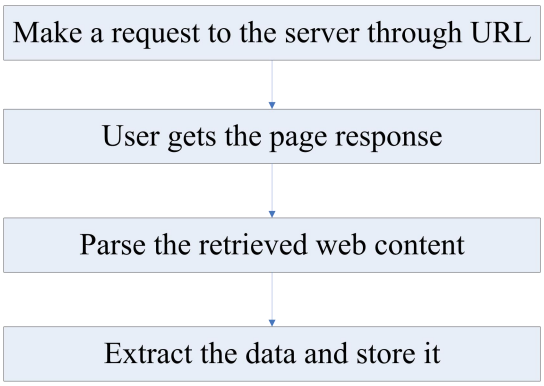
then sends processed requests to the scheduler and asks if there is another request;

9. this process repeats from step 1 until there are no requests from the scheduler.

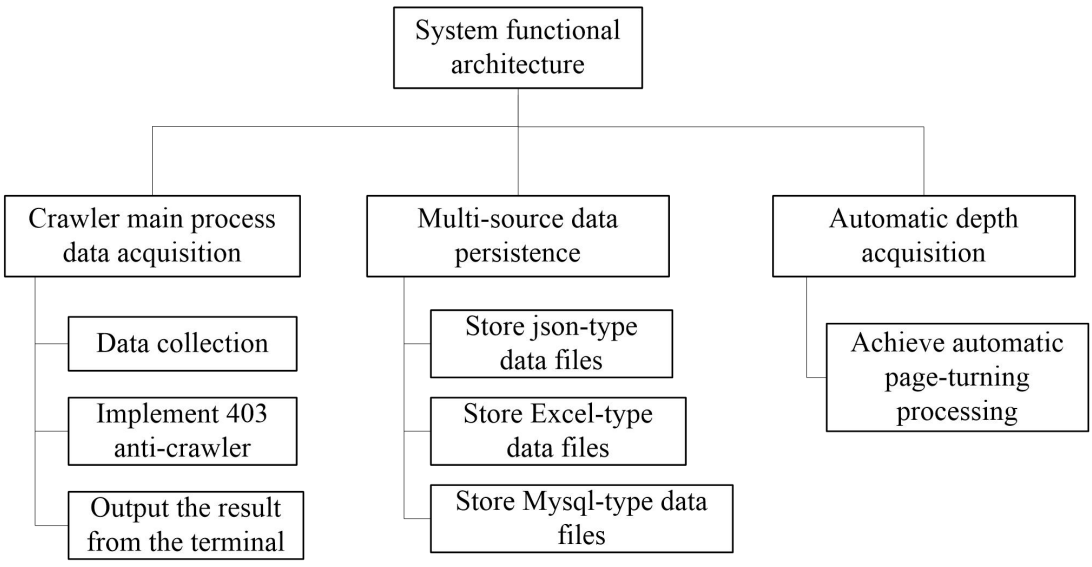
Crawler pipelines design

Based on Scrapy framework, the crawler mainly through the url sends a request to the server, using the method of xpath to extract the position of the information you want, in the case of the server operating normally,it will access the data through a loop, and users can receive the requested page response and get the binary format of data. Then the parser will parse the content of the page. Finally the pipelines extract and process the binary data, output the format you want and save it, such as json format, save locally.

Working principle of crawler:



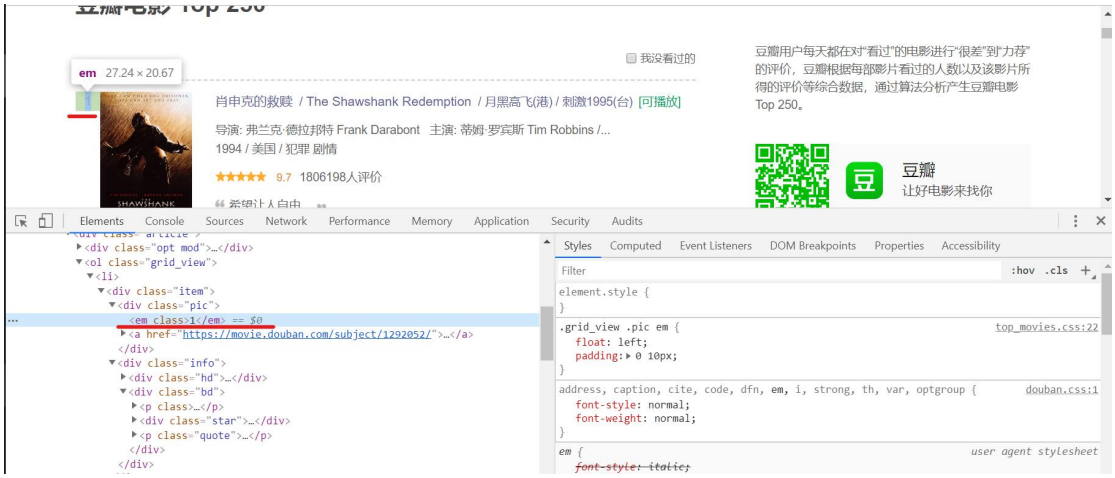
System functional architecture：



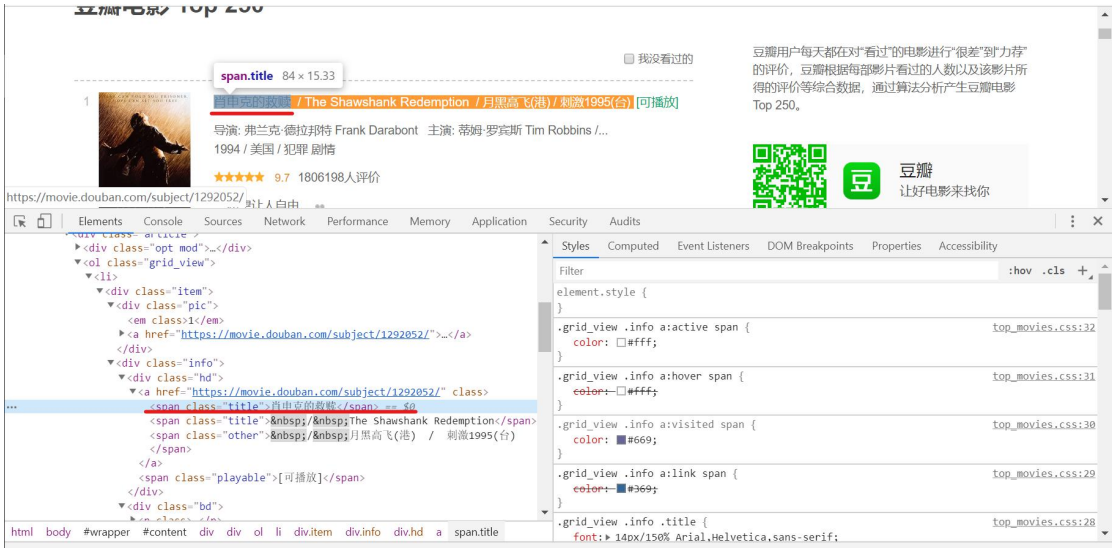
1) crawler main process data acquisition

Data collection is mainly realized by extracting fields in xpath, which uses path expressions to select nodes in web pages and to confirm specific information contents through their elements, attributes, etc.

In crawler project：



Rank's Xpath extraction method:
movie['rank'] = movie_item.xpath('div[@class = "pic"]/em/text()').extract()



Name's Xpath extraction method:
movie['name'] = movie_item.xpath('div[@class = "info"]/div[@class = "hd"]/a/span[@class = "title"][1]/text()').extract()

Many websites have anti-crawler mechanisms, so what is the anti-crawler mechanism?When our crawlers are first made, they are usually simple, fast, but low camouflage.If the website we crawl doesn't have an anti-crawler mechanism, then we can simply crawl a lot of data, but if the website has an anti-crawler mechanism, such as checking header information or making statistics for IP access frequency.Once the website monitors recognizes crawlers, such as the same IP address and the user-agent is always python, the website will restrict access to your IP. At this point, we need to crack the anti-crawler mechanism. Anti-anti-crawler mechanism can simulate different browser behavior and change proxy servers and gateways at a certain frequency to crack the anti-crawler detection of websites.

First of all, we can quickly get a list of user-agents through "rotate_useragent.py" to achieve automatic random selection, as shown below:

```python
    def process_request(self, request, spider):
        #这句话用于随机轮换user-agent
        ua = random.choice(self.user_agent_list)
        #若有ua
        if ua:
            # 输出自动轮换的user-agent
            print(ua)
            #把值传入request中
            request.headers.setdefault('User-Agent', ua)

```

Then extend the middleware, write a list of useragents, and take the common browser request headers as a list, as shown below:

```python
user_agent_list = [\
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1"\
    "Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11",\
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1092.0 Safari/536.6",\
    "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6",\
    "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/19.77.34.5 Safari/537.1",\
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.9 Safari/536.5",\
    "Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5",\
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",\
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 6.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3",\
    "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.0 Safari/536.3",\
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24",\
    "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.24"
]
```

2) Multi-source data persistence

Through the processing of the data, we can finally store the data as: json format, Excel format, database format.

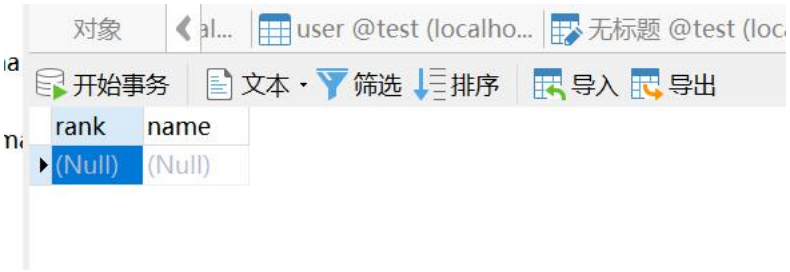
Json storage is a great option when the amount of data is not very large. When the data is passed in from an item, it is dict converted to json by the "json.dumps" method.Note that because "json.dumps" serializes the default ASCII encoding for Chinese, you need to specify "ensure_ascii = False" for the final output, as shown below:

```python
class DoubanmoviePipeline(object):
    def __init__(self):
        self.file = open('JsonTOP250!.json', 'w', encoding='utf-8')
    #运行时调用
    def process_item(self, item, spider):
        #将dict类型的数据转成str
        line = json.dumps(dict(item), ensure_ascii=False) + "\n"
        #写入
        self.file.write(line)
        return item
    #关闭爬虫的方法
    def spider_closed(self, spider):
        #调用close进行关闭
        self.file.close()
```

CSV type is one of the simplest type of saving. It only requires the improt of CSV module into the project and a call to "CSV. Writer" to export the CSV file, as shown below:

```python
class DoubanmoviePipeline(object):
    #定义一个构造函数
    def __init__(self):
        #存放路径
        path = 'D:\codeSavePath\production practice\DoubanMovie\TOP250.csv'
        #存放文件格式
        self.file = open(path,'a+',encoding='utf-8')
        #存放文件读写格式
        self.writer = csv.writer(self.file)
    #pipeline必须调用的方法，返回item或dict对象
    def process_item(self, item, spider):
        #将item打印出来
        print(item)
        #打印电影排名
        print('电影排名:{0}'.format(item['rank'][0]))
        #按行保存
        self.writer.writerow((item['rank'],item['name']))
        #返回item
        return item
```

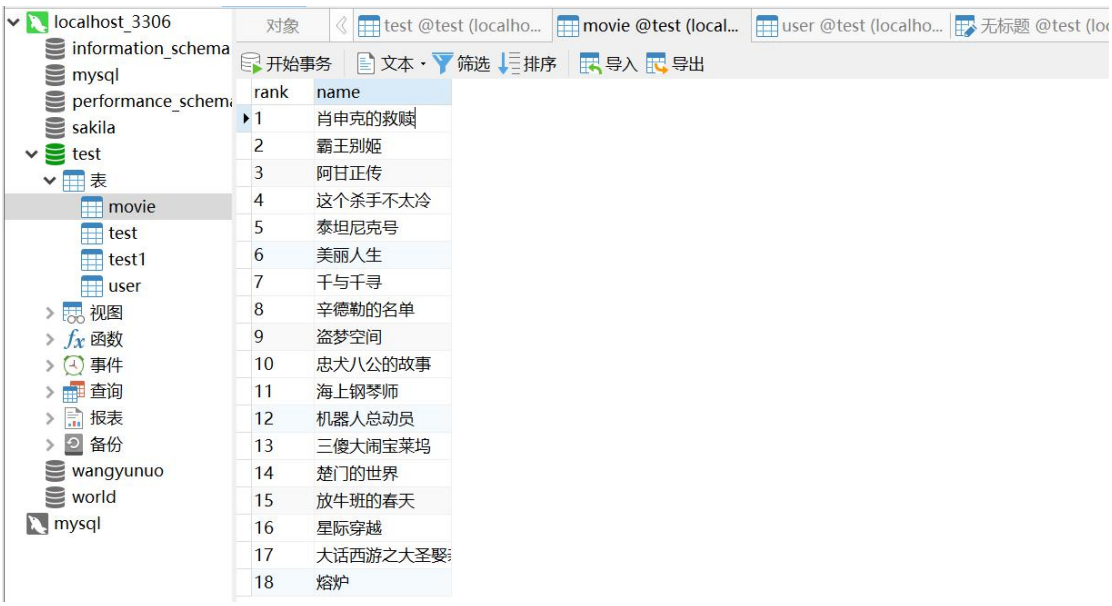
To save the data in Mysql, you first need to create a new table "test" in the Mysql database, as shown below:



You then need to connect the database to the crawler in python code, mainly by matching the name of the database with the name of the table.Use SQL statements to store the crawled results directly into the database, as shown below:

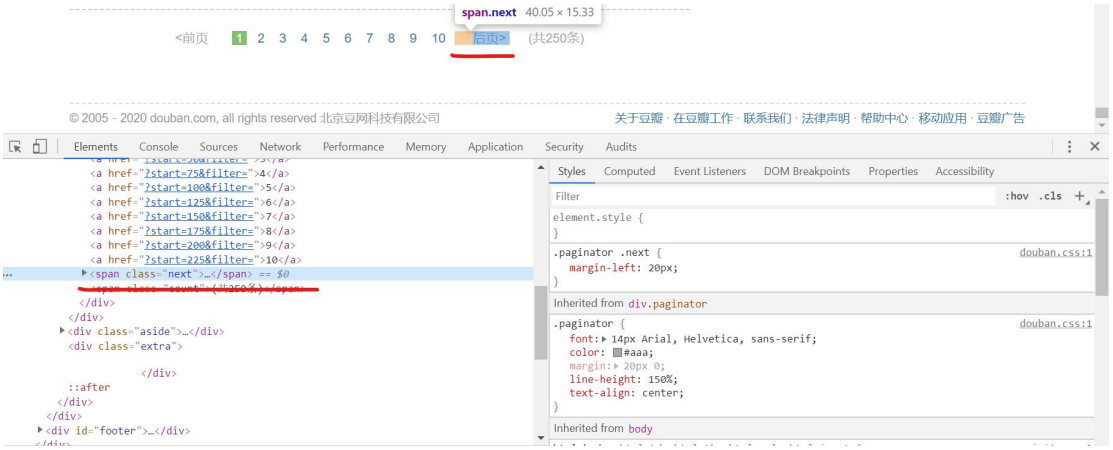
```python
class DoubanmoviePipeline(object):
    def process_item(self, item, spider):
        dbObject = dbHandle()
        cursor = dbObject.cursor()
        cursor.execute("USE test")
        sql = "INSERT INTO movie(rank,name) VALUES(%s,%s)"
      抛出执行时的异常
        try:
            cursor.execute(sql,
                           ( item['rank'], item['name']))
            cursor.connection.commit()
        except BaseException as e:
            print("错误在这里>>>>>>>>>>>>>", e, "<<<<<<<<<<<<<错误在这里")
            dbObject.rollback()
        return item
```

After running the crawler, the results of the database are as follows:



3) Automatic depth acquisition

The automatic depth acquisition module mainly deals with the page-turning function of the website. If automatic page-turning is not added, the crawler will run once and only be able to crawl the contents of one page, which cannot guarantee that it can crawl all the information ranked at 250.By automatically turning the page, the URL of the next page can be obtained through the xpath method, and then judge whether the "next page" is valid. If so, splice its URL to the previous page, and complete the crawling of all page information after sending the request, as shown below:




```
#自动请求翻页实现爬虫的深度采集
    nextPage=response.xpath('//span[@class="next"]/a/@href')
    #判断nextPage是否有效
    if nextPage:
        #拼接下一页的地址
        url=response.urljoin(nextPage[0].extract())
        #发送url后页请求
        yield scrapy.Request(url,self.parse)
```

Crawler test

start the spiders（command： scrapy crawl + spiders project name）
After the terminal enters the crawler project folder, enter the following command：

```
PS D:\codeSavePath\production practice> cd DoubanMovie
PS D:\codeSavePath\production practice\DoubanMovie> scrapy crawl dbmovie
```

Return the result（part of the result）：

```
Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3
2020-02-05 15:24:29 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://movie.douban.com/top250> (referer: None)
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['肖申克的救赎'], 'rank': ['1']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['霸王别姬'], 'rank': ['2']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['阿甘正传'], 'rank': ['3']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['这个杀手不太冷'], 'rank': ['4']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['美丽人生'], 'rank': ['5']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['泰坦尼克号'], 'rank': ['6']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['千与千寻'], 'rank': ['7']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['辛德勒的名单'], 'rank': ['8']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['盗梦空间'], 'rank': ['9']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['忠犬八公的故事'], 'rank': ['10']}
2020-02-05 15:24:29 [scrapy.core.scraper] DEBUG: Scraped from <200 https://movie.douban.com/top250>
{'name': ['海上钢琴师'], 'rank': ['11']}
```

API

Api background

Flask is a lightweight, customizable framework written in Python that is more flexible, lightweight, safe, and easy to use than other frameworks of its type.The construction of the Flask is based on the example on the official website. I wrote a minimal Flask framework and tested it.

1）Install the extension package

$pip install flask

2）Write sample code

```python
from flask import Flask
from flask_restful import Resource, Api

app = Flask(__name__)
api = Api(app)

class HelloWorld(Resource):
    def get(self):
        return {'hello': 'world'}

api.add_resource(HelloWorld, '/')

if __name__ == '__main__':
    app.run(debug=True)
```

3）result

4） After the API is successfully deployed, the terminal returns an address, and we can open another new terminal and test it with curl to get the task.

$curl http://127.0.0.1:5000/

```
PS D:\codeSavePath\production practice> cd git-flask
PS D:\codeSavePath\production practice\git-flask> curl http://127.0.0.1:5000/


StatusCode        : 200
StatusDescription : OK
Content           : {
                        "hello": "world"
                    }

RawContent        : HTTP/1.0 200 OK
                    Content-Length: 25
                    Content-Type: application/json
                    Date: Fri, 17 Jan 2020 11:54:29 GMT
                    Server: Werkzeug/0.16.0 Python/3.6.5

                    {
                        "hello": "world"
                    }

Forms             : {}
Headers           : {[Content-Length, 25], [Content-Type, application/json], [Date, Fri, 17 Jan 2020 11:54:29 G
                    MT], [Server, Werkzeug/0.16.0 Python/3.6.5]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 25


PS D:\codeSavePath\production practice\git-flask>
```

Api design

Api construction process: import the required Flask module into the project with the command "pip install", such as Flask, flask_restful,Api, etc.First you need to create an instance of the Flask. The app is an instance of the Flask, and it can take the name of the package or module as an argument, but it usually passes "__name__".

```
33    app = Flask(__name__)
34    api = Api(app)
```

In class, it inherits from the flask_rest.resource class and defines methods from the request, such as get,post. I mainly defined a request submitted by "get". In this method, I called the "Popen" method to run the instructions of the crawler project. In "Popen", the "cwd" parameter can be used to specify where the shell command should run.

```
89      p = Popen('scrapy runspider --loglevel=INFO dbmovie.py',shell= True,stdout=PIPE,stderr=PIPE,cwd='D:\\codeSavePath\\production p
90      p.wait()
91      print(p.stdout.readlines())
92      print(p.stderr.readlines())
```

If you want to output the contents of the crawl to the terminal, you can simply read the absolute path of the stored file and output it.

```
filename = "D:\codeSavePath\production practice\DoubanMovie\DoubanMovie\spiders\JsonTOP250!.json"
with open(filename,encoding='UTF-8') as fp:
    content = fp.readlines()
    return content,200
```

A route is then bound through “api.add_resource”.

```
194
195     api.add_resource(ToTest,'/crawl')
196
```

Finally, start the API with app.run(debug=True,port=9191) in the main function.

```
production practice > git-flask > main.py > ...
1   import os
2   from api import *
3   #os.system("python api.py") #应该import进去
4   if __name__ == '__main__':
5       app.run(debug=True,port=9191)
6
```

Api server deployment

Once this setup is complete, I can run main.py to deploy a crawl request of my own.

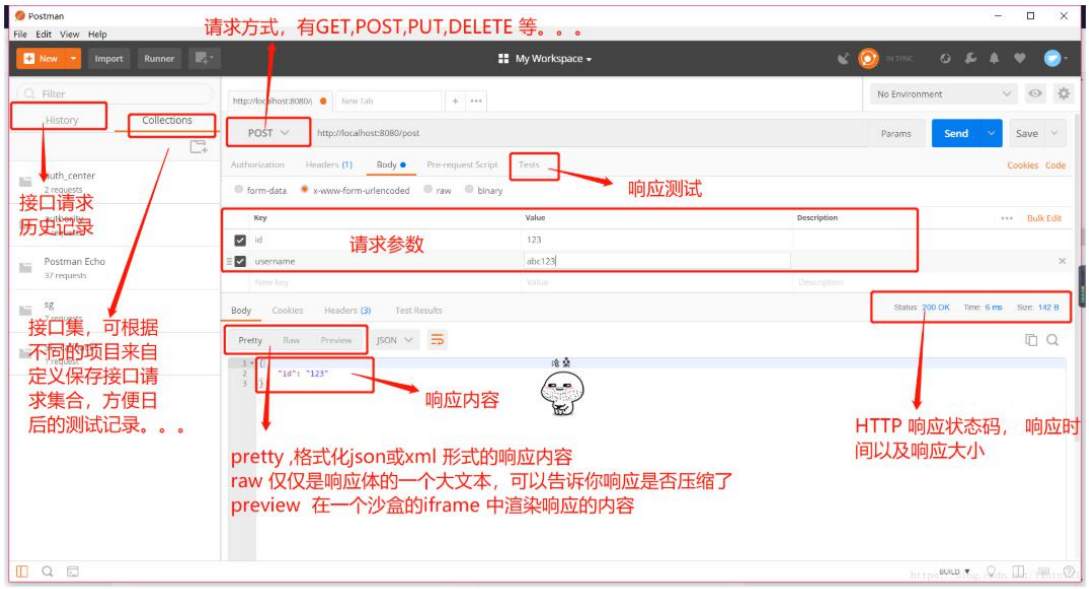
```
PS D:\codeSavePath\production practice\git-flask> python main.py
 * Serving Flask app "api" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 260-341-837
 * Running on http://127.0.0.1:9191/ (Press CTRL+C to quit)
```

Api test

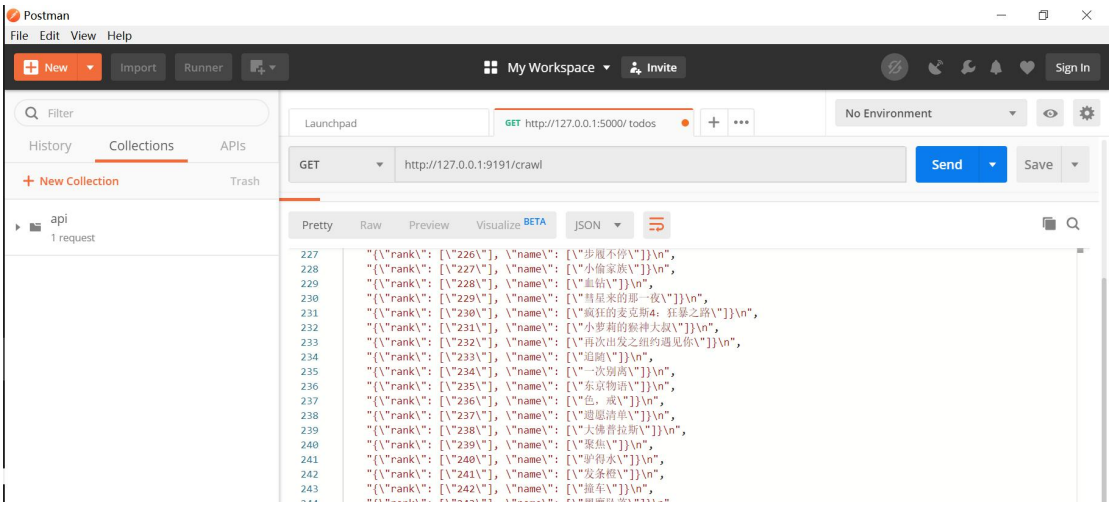
1) curl： Curl is a file transfer utility that uses URL syntax to work from the command line. Test as following：

```
PS D:\codeSavePath\production practice> cd git-flask
PS D:\codeSavePath\production practice\git-flask> curl http://127.0.0.1:5000/


StatusCode        : 200
StatusDescription : OK
Content           : {
                        "hello": "world"
                    }

RawContent        : HTTP/1.0 200 OK
                    Content-Length: 25
                    Content-Type: application/json
                    Date: Fri, 17 Jan 2020 11:54:29 GMT
                    Server: Werkzeug/0.16.0 Python/3.6.5

                    {
                        "hello": "world"
                    }

Forms             : {}
Headers           : {[Content-Length, 25], [Content-Type, application/json], [Date, Fri, 17 Jan 2020 11:54:29 G
                    MT], [Server, Werkzeug/0.16.0 Python/3.6.5]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 25


PS D:\codeSavePath\production practice\git-flask>
```

2) postman： This is a visual software, used to simulate HTTP requests, to help people in the back end of the unit tools.It can customize the request URL, request type [GET,POST, etc.], and add Head information and HTTP body information, etc., so that we can test HTTP requests simply and intuitively.



Return the result：



appendix

Problems encountered and solutions

1) Crawler module part：

- Call the crawler module with the main function

The original crawler can only be called by shell command: scrapy crawl dbmovie can only be called by shell command in the crawler directory. The crawler part cannot be used as a module for other projects. However, if it can be called in python code, the portable of crawler will be greatly improved.

Solutions and use of technology：

Create a new.py file in the crawler spiders directory and use the subprocess.popen () module to generate a subroutine that calls the shell command scrapy crawl dbmovie in the subroutine using a combination of different parameters in Popen.

```
#version 7
import subprocess
subprocess.Popen("scrapy crawl dbmovie", shell=True)
```

- When climbing data, can only climb to get a page, without page turn Settings.

When the crawler is in operation, because it cannot get the page turning button, it can only crawl the information of one page. If it wants to get all the information of 250 movies, it needs to let the program recognize the page turning itself, then splicing the obtained url to the previous page, so as to get the url of all the page numbers, and then crawl the information.

```
nextPage=response.xpath('//span[@class="next"]/a/@href')
if nextPage:
url=response.urljoin(nextPage[0].extract())
yield scrapy.Request(url,self.parse)
```

- When the crawler sends the request, 403 error codes may appear.

This is because many websites have anti-crawler technology, which prevents the crawler from directly accessing the web page. Therefore, dynamic agents need to be created to simulate the crawler as a browser to access the web page.

Solutions and use of technology：

You need to create a dynamic proxy list, randomly select the user agent header information in the list, and disguise the request.Bind every request of the crawler and send it to the visiting url.

def process_request(self, request, spider):

ua = random.choice(self.user_agent_list)

if ua:

print(ua)

request.headers.setdefault('User-Agent', ua)

```
user_agent_list = [\
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari
    "Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1092.0 Safari
    "Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6"
    "Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/19.77.34.5 Safari/
    "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.9 Safari/536
    "Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5
    "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari
    "Mozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3"
    "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.10
```

```
CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11
2020-02-05 15:24:30 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://movie.douban.com/top250?start=25&filter=> (refer
er: https://movie.douban.com/top250)
```

2) API server sction：

- Unable to call the crawler project section within the API project.

there is always an error: the crawler file cannot be found because the execution of the crawler is under the crawler foldeer.

Solutions and use of technology：

After searching the data, I found that the scrapy command is a local command that can only be executed in the current file, but the runspider command is a global command that can run in other files.Using the Popen method, there is a parameter called 'cwd' that specifies the subprocess working directory.Then the parameter shell=True, shell command can be used normally in windows.

p = Popen('scrapy runspider --loglevel=INFO dbmovie.py',shell= True,stdout=PIPE,stderr=PIPE,cwd='D:\\codeSavePath\\production practice\\DoubanMovie\\DoubanMovie\\spiders')

p.wait()

- When you output 'stdout' and 'stderr' using the 'subprocess.run' method, the result is not the final standard output and standard error output.

```
"Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3\r\nMozilla/5.0
    (Windows NT 6.2; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/19.77.34.5 Safari/537.1\r\nMozilla/5.0 (Windows
    NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1062.0 Safari/536.3\r\nMozilla/5.0 (Macintosh;
    Intel Mac OS X 10_8_0) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3\r\nMozilla/5.0 (Windows
    NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) Chrome/22.0.1207.1 Safari/537.1Mozilla/5.0 (X11; CrOS i686
    2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11\r\nMozilla/5.0 (Windows NT 6.0)
    AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5\r\nMozilla/5.0 (Windows NT 5.1) AppleWebKit/
    536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3\r\nMozilla/5.0 (X11; Linux x86_64) AppleWebKit/536.5
    (KHTML, like Gecko) Chrome/19.0.1084.9 Safari/536.5\r\nMozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML,
    like Gecko) Chrome/19.0.1062.0 Safari/536.3\r\nMozilla/5.0 (Windows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko)
    Chrome/19.0.1063.0 Safari/536.3\r\nMozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/
    19.0.1062.0 Safari/536.3\r\nMozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1
    Safari/536.3\r\n"
```

Solutions and use of technology：

stdout=PIPE,stderr=PIPE

print(p.stdout.readlines())

print(p.stderr.readlines())

After a successful run, I can output standard output and standard error output (partly)：

```
 * Detected change in 'D:\\codeSavePath\\production practice\\git-flask\\api.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 260-341-837
 * Running on http://127.0.0.1:9191/ (Press CTRL+C to quit)
[b'Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3\r\n', b'Mozilla/5.0 (
Windows NT 6.0) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5\r\n', b'Mozilla/5.0 (Windows NT 6.2)
 AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3\r\n', b'Mozilla/5.0 (Windows NT 6.0) AppleWebKit/536
.5 (KHTML, like Gecko) Chrome/19.0.1084.36 Safari/536.5\r\n', b'Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.3 (KHTML, like
 Gecko) Chrome/19.0.1062.0 Safari/536.3\r\n', b'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.1 (KHTML, like Gecko) C
hrome/22.0.1207.1 Safari/537.1Mozilla/5.0 (X11; CrOS i686 2268.111.0) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.11
32.57 Safari/536.11\r\n', b'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_0) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.
0.1063.0 Safari/536.3\r\n', b'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0
Safari/536.3\r\n', b'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/535.24 (KHTML, like Gecko) Chrome/19.0.1055.1 Safari/535.2
4\r\n', b'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1061.1 Safari/536.3\r\n', b
'Mozilla/5.0 (Windows NT 6.2) AppleWebKit/536.6 (KHTML, like Gecko) Chrome/20.0.1090.0 Safari/536.6\r\n', b'Mozilla/5.0 (Wi
ndows NT 5.1) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0.1063.0 Safari/536.3\r\n']
[b'2020-02-04 09:29:17 [scrapy.utils.log] INFO: Scrapy 1.8.0 started (bot: DoubanMovie)\r\n', b'2020-02-04 09:29:17 [scrapy
.utils.log] INFO: Versions: lxml 4.4.2.0, libxml2 2.9.5, cssselect 1.1.0, parsel 1.5.2, w3lib 1.21.0, Twisted 19.10.0, Pyth
on 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD64)], pyOpenSSL 19.1.0 (OpenSSL 1.1.1d  10 Sep 2
019), cryptography 2.8, Platform Windows-10-10.0.18362-SP0\r\n', b"2020-02-04 09:29:17 [scrapy.crawler] INFO: Overridden se
ttings: {'BOT_NAME': 'DoubanMovie', 'LOG_LEVEL': 'INFO', 'NEWSPIDER_MODULE': 'DoubanMovie.spiders', 'ROBOTSTXT_OBEY': True,
```

Summarize what I've learned in other parts

1) Github related knowledge：

steps of uploading the project to the github：

If you want to upload a file to text, just place it in the test folder where you want it, and type the following command in git bash:

cd d:/git(Store path)

cd test(Storage folder)

git add .

git commit -m 'Notes filled in'

git push

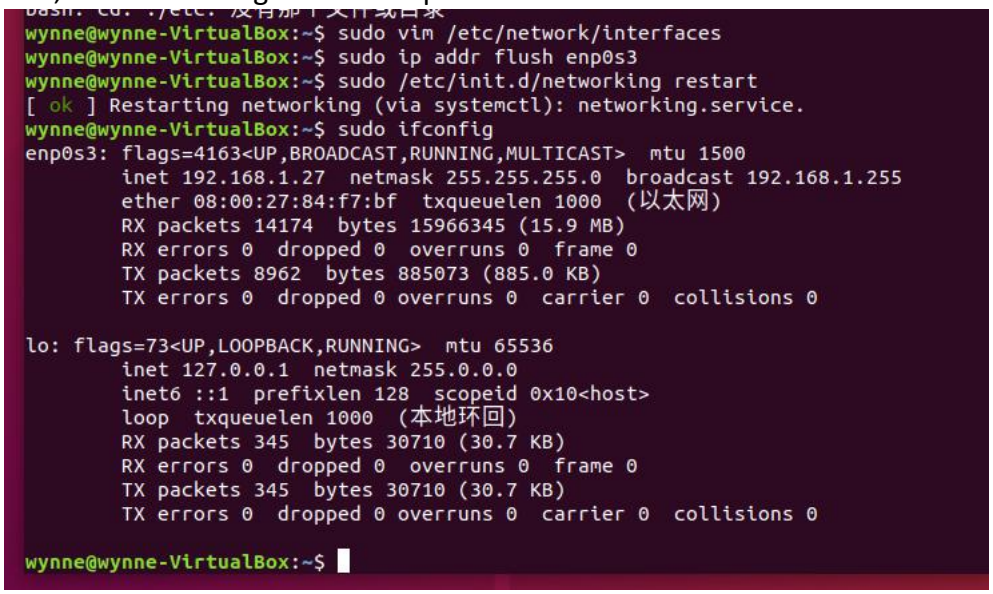
2) Create a temporary file, use the 'tempfile.NamedTemporaryFile' method, and use the timestamp prefix as a file.

```
now = time.strftime("%Y-%m-%d-%H_%M_%S",time.localtime(time.time()))
aa = tempfile.NamedTemporaryFile(mode='w+b',dir='D:',prefix=now,suff
ix='.json')
print("-----这是 api 中的临时文件-------")
print(aa.name)
```

```
 * Debugger is active!
 * Debugger PIN: 260-341-837
 * Running on http://127.0.0.1:9191/ (Press CTRL+C to quit)
-----这是api中的临时文件-------
D:\codeSavePath\production practice\git-flask\2020-02-04-17_22_140pcjcyue.json
[b'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.3 (KHTML, like Gecko) Chrome/19.0
```

Unresolved issues and future expectations

1) When I create a Linux virtual machine using Oracle VM VirtualBox, the system is fine, but networking has been a problem.



2) future expectations:

- Write the files obtained by the crawler in a temporary file and delete them after use.
- Switch to another your favorite or meaningful site and crawl the data.
- Specify the number of crawlers that can be retrieved, such as the first 100.
- Multiple crawler processes occur simultaneously.