

# گزارش تمرین اول درس مبانی بازیابی اطلاعات

Crawling with scrapy

گروه ۶

امیررضائی – هادی زمانی

نیازمندی ها :

- Pyhon 3.7.0
- Postgresql 12
- پکیج های درون فایل requirements.txt

شرح پروژه :

جهت اجرای پروژه از فایل `crawl.py` شروع میکنیم در این فایل بعد از ران کردن عملیات `crawl` شروع میشود و اطلاعات در دیتابیس ذخیره میشود و پس از اتمام عملیات به مدت یک ساعت صبر میکند و سپس دوباره عملیات را شروع می‌شود. و اطلاعات هم در دیتابیس و هم در فایل `items.json` در پوشه `crawler` ذخیره میشود.

جهت گرفتن خروجی جداگانه و دستی `json` کافی است برنامه را با دستور

```
$ scrapy crawl news -o filename.json
```

اجرا کنیم. ( نمونه خروجی `json` تحت عنوان `sample.json` در فایل ها موجود میباشد.)

جهت تبدیل داده ها به انکودینگ `utf-8` ، در فایل `setting.py` دستور زیر را استفاده میکنیم.

```
FEED_EXPORT_ENCODING = 'utf-8'
```

در این تکلیف سایت هدف ( emalls.ir ) دارای محدودیت تعداد ریکوئست بود که برای رفع این مشکل از google bot user-agent استفاده شده است:

```
USER_AGENT = "Mozilla/5.0 (compatible; Googlebot/2.1;+http://www.google.com/bot.html) "
```

راه های دیگری برای رفع محدودیت های 403 هست که از آن ها میتوان به ۲ پکیج زیر اشاره کرد.

- Scrapy-user-agent
- Scrapy-proxy-pool

تنظیمات مربوط در فایل `setting.py` گذاشته شده است(عکس در پایین) که به دلخواه از یکی از این پکیج ها میشود استفاده کرد. برای استفاده از هر کدام کافی است تکه کد مربوط را `uncomment` کنیم.البته ممکن است برای برخی از سایت ها عمل نکنند پس بهترین راه همان استفاده از `google bot user-agent` است. همچنین در فایل `utils.py` تعداد زیادی از `user agent` ها آورده شده است که میتوان با استفاده از تابع `get_random_agent()` میتوان از هر کدام به صورت رندوم در ریکوئست ها استفاده کرد. برای این کار باید `user agent` جدید را در `headers` ریکوئست بگذاریم. مثال :

```
yield response.follow(detail, callback=self.parse_detail , headers = {'User-agent': get_random_agent()})
```

که چون ما از گوگل بات استفاده کرده ایم به این نیازی نداریم.

```
# USER AGENT BYPASS
# DOWNLOADER_MIDDLEWARES = {
#     'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': None,
#     'scrapy_user_agents.middlewares.RandomUserAgentMiddleware': 400,
# }

# PROXY BYPASS
# PROXY_POOL_ENABLED = True
# DOWNLOADER_MIDDLEWARES = {
#     # ...
#     'scrapy_proxy_pool.middlewares.ProxyPoolMiddleware': 610,
#     'scrapy_proxy_pool.middlewares.BanDetectionMiddleware': 620,
#     # ...
# }
```

نحوه‌ی کارکرد دیتابیس :

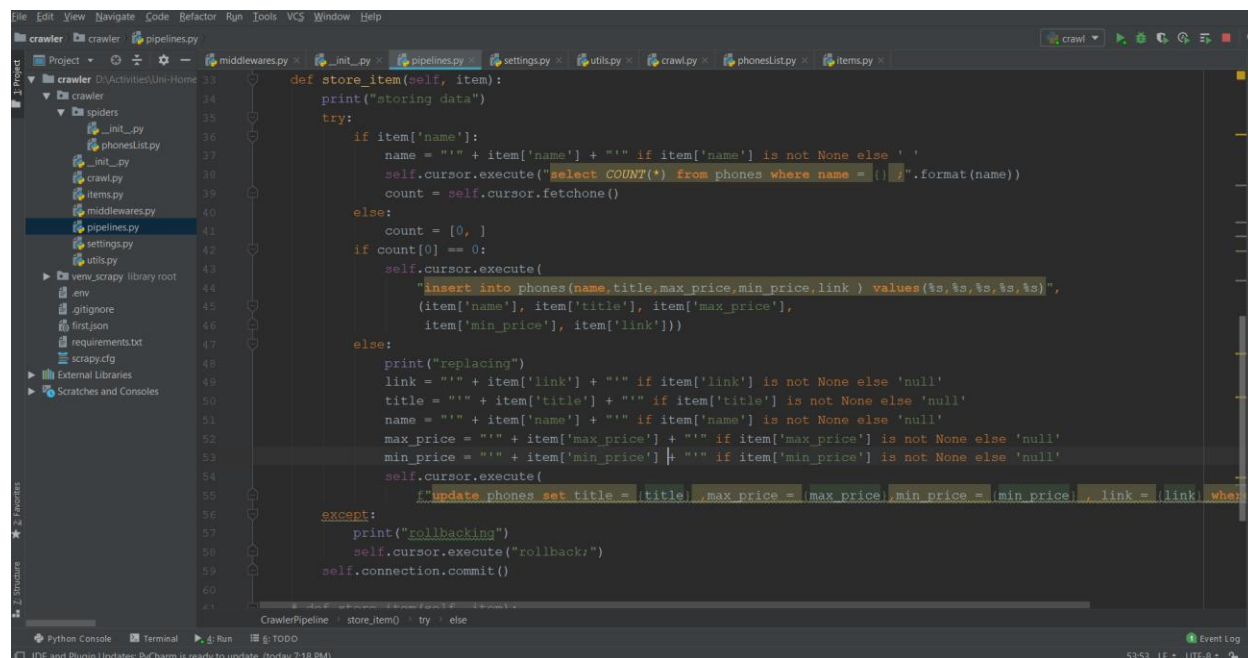
در این پروژه از دیتابیس postgresql استفاده شده است و تنظیمات مربوط در فایل pipelines.py آورده شده است و اطلاعات دیتابیس در فایل env. موجود است و میتوانید به دلخواه تغییر دهید و نیاز است یک جدول به اسم phones ساخته شود. برای اطمینان میتوانید از command زیر استفاده کنید.

```
CREATE TABLE phones (
  id serial PRIMARY KEY,
  name varchar,
  title varchar,
  max_price varchar,
  min_price varchar,
  link varchar
);
```

در فایل pipelines از دو تابع store\_items استفاده شده است. از آنجایی که برای عملیات crawl در بار اول دیتای duplicate نداریم جهت سرعت ذخیره سازی و بهینه بودن دیتابیس بدون شرط اطلاعات را ذخیره میکنیم . (مثلا برای ذخیره‌ی آبجکت ۱۰۰۱ ام باید برای چک کردن اینکه قبلا ذخیره شده است یا نه بر روی ۱۰۰۰ تا سطر کوئری بزند که این عدد به تدریج بالاتر هم میرود و نیازی نیست برای ذخیره سازی در بار اول این کار انجام شود).

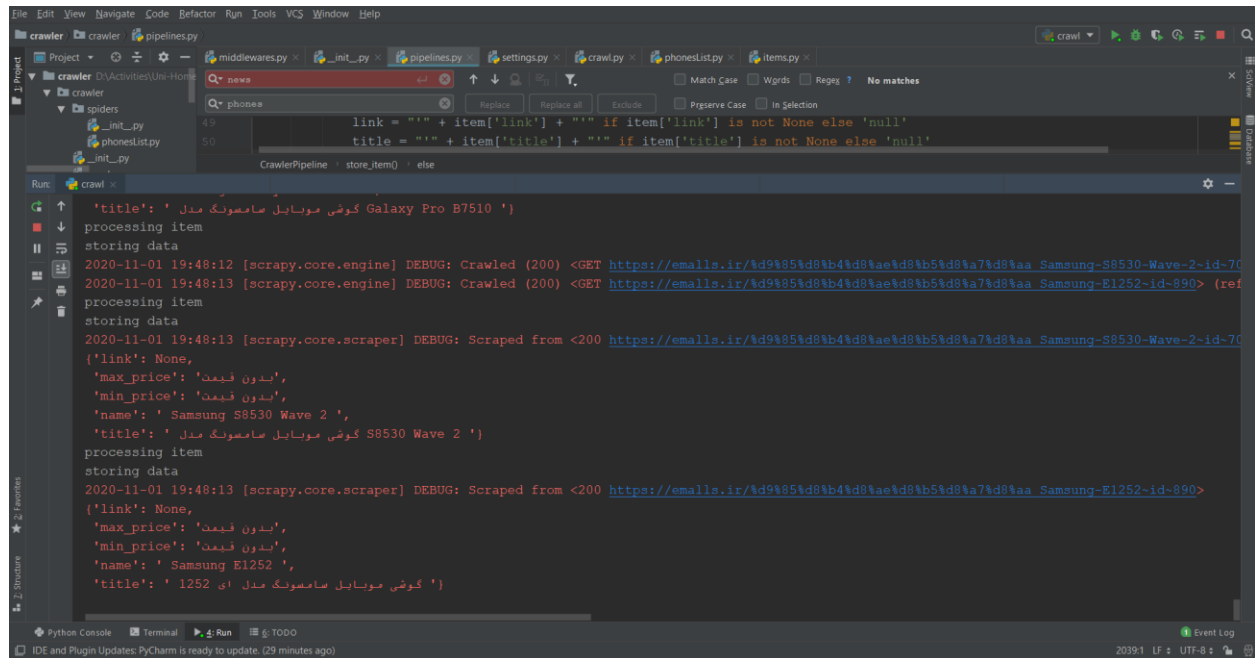
```
# def store_item(self, item):
#
#     self.cursor.execute(
#
#         "insert into phones(name,title,max_price,min_price,link ) values(%s,%s,%s,%s,%s)",
#         (item['name'], item['title'], item['max_price'],
#          item['min_price'], item['link']))
#
#     self.connection.commit()
```

در دفعات بعدی جهت جلوگیری از ذخیره ی مجدد صفحات و duplicate نبودن داده ها از تکه کد زیر استفاده میکنیم :



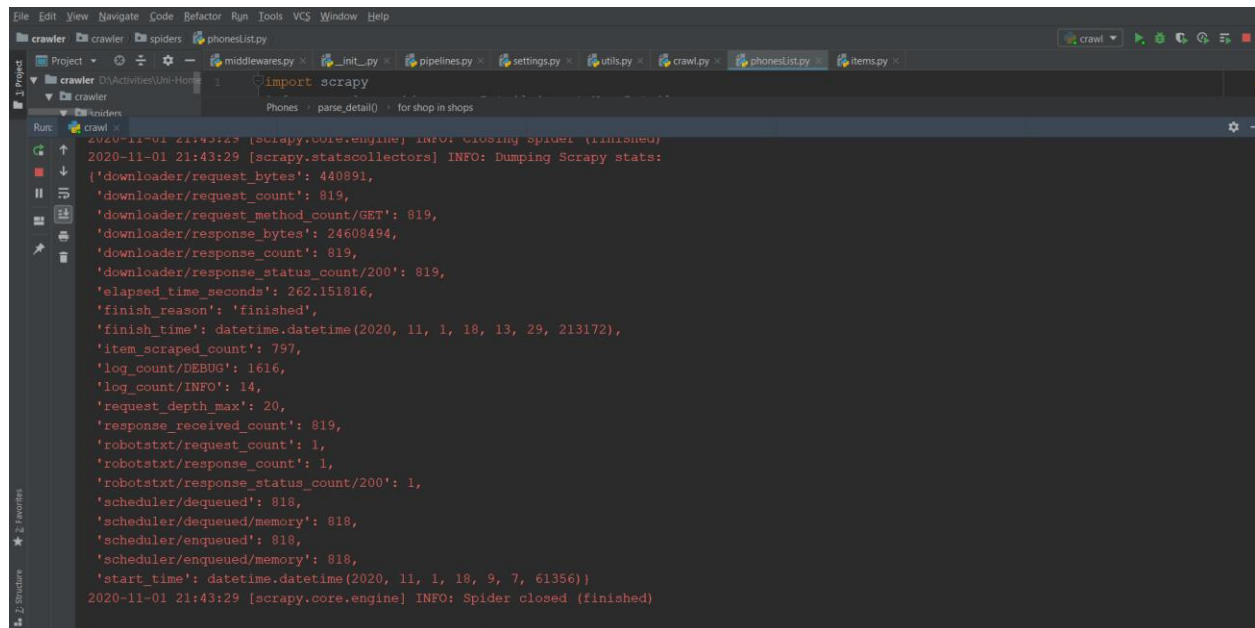
```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
crawler crawler pipelines.py
Project
  crawler
    spiders
      __init__.py
      phonesList.py
    __init__.py
    crawl.py
    items.py
    middlewares.py
    pipelines.py
    settings.py
    utils.py
  venv_scrapy library root
  env
  gitignore
  first.json
  requirements.txt
  scrapy.cfg
  External Libraries
  Scratches and Consoles
2 Favorites
2 Structure
CrawlerPipeline > store_item() > try : else
def store_item(self, item):
    print("storing data")
    try:
        if item['name']:
            name = "" + item['name'] + "" if item['name'] is not None else ' '
            self.cursor.execute("select COUNT(*) from phones where name = {}".format(name))
            count = self.cursor.fetchone()
        else:
            count = [0, ]
        if count[0] == 0:
            self.cursor.execute(
                "insert into phones(name,title,max_price,min_price,link ) values(%s,%s,%s,%s,%s)",
                (item['name'], item['title'], item['max_price'],
                 item['min_price'], item['link']))
        else:
            print("replacing")
            link = "" + item['link'] + "" if item['link'] is not None else 'null'
            title = "" + item['title'] + "" if item['title'] is not None else 'null'
            name = "" + item['name'] + "" if item['name'] is not None else 'null'
            max_price = "" + item['max_price'] + "" if item['max_price'] is not None else 'null'
            min_price = "" + item['min_price'] + "" if item['min_price'] is not None else 'null'
            self.cursor.execute(
                f"update phones set title = {title} ,max price = {max_price} ,min price = {min_price} , link = {link} where name = {name}")
    except:
        print("rollbacking")
        self.cursor.execute("rollback;")
    self.connection.commit()
```

اسکرین شات از پروژه‌ی در حال اجرا :



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
crawler crawler pipelines.py
Project crawler
spiders
  init.py
  phonesList.py
  _init.py
  pipelines.py
  settings.py
  crawl.py
  phonesList.py
  items.py
  scrapy
    import scrapy
    class CrawlerPipeline:
      def store_item(self, item):
        link = "" + item['link'] + "" if item['link'] is not None else 'null'
        title = "" + item['title'] + "" if item['title'] is not None else 'null'
        # ... (rest of the code) ...

Run: crawl
2020-11-01 19:48:12 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://emalls.ir/%d9%85%d8%b4%d8%ae%d8%b5%d8%a7%d8%aa Samsung-S8530-Wave-2-id-70>
2020-11-01 19:48:13 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://emalls.ir/%d9%85%d8%b4%d8%ae%d8%b5%d8%a7%d8%aa Samsung-E1252-id-890> (ref
processing item
storing data
2020-11-01 19:48:13 [scrapy.core.scrapy] DEBUG: Scraped from <200 https://emalls.ir/%d9%85%d8%b4%d8%ae%d8%b5%d8%a7%d8%aa Samsung-S8530-Wave-2-id-70>
('link': None,
 'max_price': 'بدون قیمت',
 'min_price': 'بدون قیمت',
 'name': ' Samsung S8530 Wave 2 ',
 'title': ' گوشی موبایل سامسونگ مدل S8530 Wave 2 ')
processing item
storing data
2020-11-01 19:48:13 [scrapy.core.scrapy] DEBUG: Scraped from <200 https://emalls.ir/%d9%85%d8%b4%d8%ae%d8%b5%d8%a7%d8%aa Samsung-E1252-id-890>
('link': None,
 'max_price': 'بدون قیمت',
 'min_price': 'بدون قیمت',
 'name': ' Samsung E1252 ',
 'title': ' 1252 گوشی موبایل سامسونگ مدل ای ')
Python Console Terminal Run
IDE and Plugin Updates: PyCharm is ready to update. (29 minutes ago)
2020-11-01 19:48:13 [scrapy.core.scrapy] DEBUG: Scraped from <200 https://emalls.ir/%d9%85%d8%b4%d8%ae%d8%b5%d8%a7%d8%aa Samsung-E1252-id-890>
```



```
File Edit View Navigate Code Refactor Run Tools VCS Window Help
crawler crawler spiders phonesList.py
Project crawler
spiders
  init.py
  phonesList.py
  _init.py
  pipelines.py
  settings.py
  crawl.py
  phonesList.py
  items.py
  scrapy
    import scrapy
    class CrawlerPipeline:
      def store_item(self, item):
        link = "" + item['link'] + "" if item['link'] is not None else 'null'
        title = "" + item['title'] + "" if item['title'] is not None else 'null'
        # ... (rest of the code) ...

Run: crawl
2020-11-01 21:43:29 [scrapy.core.engine] INFO: Closing spider (finished)
2020-11-01 21:43:29 [scrapy.statscollectors] INFO: Dumping Scrapy stats:
{'downloader/request_bytes': 440891,
 'downloader/request_count': 819,
 'downloader/request_method_count/GET': 819,
 'downloader/response_bytes': 24608494,
 'downloader/response_count': 819,
 'downloader/response_status_count/200': 819,
 'elapsed_time_seconds': 262.151816,
 'finish_reason': 'finished',
 'finish_time': datetime.datetime(2020, 11, 1, 18, 13, 29, 213172),
 'item_scraped_count': 797,
 'log_count/DEBUG': 1616,
 'log_count/INFO': 14,
 'request_depth_max': 20,
 'response_received_count': 819,
 'robotstxt/request_count': 1,
 'robotstxt/response_count': 1,
 'robotstxt/response_status_count/200': 1,
 'scheduler/dequeued': 818,
 'scheduler/dequeued/memory': 818,
 'scheduler/enqueued': 818,
 'scheduler/enqueued/memory': 818,
 'start_time': datetime.datetime(2020, 11, 1, 18, 9, 7, 61356)}
2020-11-01 21:43:29 [scrapy.core.engine] INFO: Spider closed (finished)
```