# تحقیق پایان ترم
# برنامه سازی وب

علی احمدی کافشانی
97105703

حامد عبدی
96109782

تا به حال شده است به سایتی مراجعه کنید و اطلاعات موجود در آن سایت را ببینید و به فکر تغییری در آن اطلاعات یا جمع اوری آن‌ها و انجام آزمایش های موردنظر خود بر روی ان‌ها باشید؟

اگر جواب شما بله است پس شما به scrapy نیاز دارید.
Scrapy یک ابزار بر پایه پایتون است که میتواند سایت‌ها را به صورتی که شما تعریف میکنید پیمایش کند و اطلاعات آن‌ها را استخراج نماید. با توجه به این تعریف هر سایتی و به صورت کلی‌تر هر api قابل پیمایش است.
برای راه اندازی این ابزار به نسخه حداقل 3.6 پایتون نیازمندیم. سپس با دستور زیر این ابزار را نصب میکنیم.

pip install Scrapy

در زمان تألیف این مقاله نسخه نهایی و مورد استفاده ما 2.5.1 است.

حال به اجزای مختلفی که scrapy می‌تواند داشته باشد نگاه می‌اندازیم
به صورت کلی یک برنامه scrapy از یک یا چند عنکبوت تشکیل شده است که می‌تواند به صورت مورد نظر سایت را پیمایش کند.
هر عنکبوت یک اسم و یک لیست از url های موردنظر برای شروع و یک یا چند تابع که چگونه این url ها را پیمایش کند دارد.
به صورت مثال در این مقاله ما سایت کافه بازار را برای به دست آوردن اسم پکیج و تعداد ستاره‌های هر برنامه پیمایش میکنیم.
ابتدا به مراجعه به سایت cafebazaar.ir با محیط سایت و api ها آشنا می‌شویم. سپس به صفحه یک برنامه می‌رویم و متوجه می‌شویم که در پایین هر اپ 2 الی 3 سری اپلیکشن پیشنهادی آمده هست. پس آدرس و اطلاعات این اپلیکشن ها در جواب هایی که از سمت سرور فرستاده می‌شود دیده خواهند. با جستجو و مشاهده درخواست های شبکه بین مرورگر و بازار به درخواست جالب AppDetailsV2Request میرسیم. این تک درخواست اطلاعات برنامه انتخاب شده و همچنین برنامه‌های مرتبط به آن را که ما نیاز داریم فراهم میکند. حال با این اطلاعات به سمت نوشتن عنکبوت میرویم.

ابتدا کد زیر را مینویسیم که میتواند با اطلاعات یک برنامه اطلاعات 30 برنامه پیشنهاد شده زیر آن را استخراج نماید.

```
import requests

package_name = "com.nbadigital.gametimelite"
data = {
"properties": {
        "language": 2,
        "clientVersion": "web",
        },
        "singleRequest": {"appDetailsV2Request": {"packageName": package_name}},
}

response = requests.post(
        "https://api.cafebazaar.ir/rest-v1/process/AppDetailsV2Request",
        json=data,
)

apps_rows = response.json()["singleReply"]["appDetailsV2Reply"]["extraContentPageBodyInfo"]["pageBody"]["rows"]
for i in apps_rows:
        for j in i["simpleAppList"]["apps"]:
                info = j["info"]
                print(info["packageName"], info["rate"])
```

در ابتدا باید یک پروژه scrapy را ایجاد کنیم. این کار را با دستور
python -m scrapy startproject <project_name>
انجام میدهیم. ما نام برنامه خود را cafeBazaar میزاریم. در پوشه ایجاد شده چند فایل موجود است که آن‌ها را به اختصار توضیح میدم:

scrapy.cfg & settings.py

همانطور که مشخص است تنظیمات برنامه در این قسمت قرار میگیرد

items.py

تعریف item ها

pipelines.py & middlewares.py

نحوه برخورد با item ها و درخواست ها در بین راه سرور و مرحله نهایی اجرا در این دو قسمت قرار میگیرد و به طور کلی middleware هستند.
و در نهایت پوشه spiders را داریم که خالی است. برای شروع ما یک عنکبوت به صورت زیر در آن تعریف میکنیم

```python
import json
import scrapy


data = []


class CafeBazaarSpider(scrapy.Spider):
    name = "cafeBazaarspider"
    start_urls = []
    base_url = "https://api.cafebazaar.ir/rest-v1/process/AppDetailsV2Request"
    global_counter = 0

    def start_requests(self):
        yield scrapy.FormRequest(
            self.base_url,
            method="POST",
            callback=self.parse,
            body=self.get_request_body("com.nbadigital.gametimelite"),
        )

    def get_request_body(self, package_name: str) -> str:
        return json.dumps(
            {
                "properties": {
                    "language": 2,
                    "clientVersion": "web",
                },
                "singleRequest": {"appDetailsV2Request": {"packageName": package_name}},
            }
        )

    def parse(self, response):
        ...
```

در این کد url که در مرحله قبل پیدا کردیم تنها url ما است پس ارایه start_urls خالی است و چون به صورت پیشفرض درخواست GET scrapy انجام میدهد ما در start_request این رفتار را جایگزین کرده و با POST که به آن نیاز داریم درخواست خود را صدا میزنیم.
در ابتدا هم یک لیست خالی برای نگه داری برنامه‌های دیده شده تولید می‌شود و سپس این اطلاعات در انتهای برنامه ذخیره میشود.

سپس باید یک تابع parse برای آن بنویسیم که به طور خلاصه یعنی عنکبوت چگونه باید با این درخواست  و جواب به دست آمده از آن رفتار کند. همانطور که به صورت دستی درخواست ها را ایجاد میکردیم در تابع parse درخواست ها را هندل بکنیم.

برای این کار ابتدا با استفاده از امکانات scrapy یک item میسازیم. برای این کار در فایل items.py

```
import scrapy
class CafebazaarItem(scrapy.Item):
package_name = scrapy.Field()
rating = scrapy.Field()
```

اگر به اطلاعات دیگری از برنامه نیاز داشتیم میتوانیم قسمت مربوط به آن را اضافه کنیم و آن را هم در درخواست های خود استخراج کنیم.
حال فایل pipelines.py را به صورت زیر تکمیل میکنیم. این قسمت وظیفه نگهداری ایتم ها در حافظه موقت و سپس ذخیره آن‌ها در حافظه دایم که برای ما یک فایل است را دارد. توجه شود که میتوان در یک دیتابیس هم این اطلاعات را ذخیره کرد.

```
data = []
class CafebazaarPipeline:
def process_item(self, item, spider):
        if item not in data:
                data.append(item)
        return item
def close_spider(self, spider):
        with open("cafeBazaarData.text", "w+") as file:
                for i in data:
                        file.write(f"{i['package_name']} | {i['rating']}\n")
```

تابع process_item قبل از ساخت هر item صدا زده می‌شود و ما با این قابلیت یک حافظه موقت ایجاد می‌کنیم. تابع close_spider هم برای دخیره در حافظه دایم است.
سپس باید در تنظیمات این pipeline را فعال کنیم. پس در فایل settings.py

```
ITEM_PIPELINES = {
'cafeBazaar.pipelines.CafebazaarPipeline': 100,
}
```

کد زیر را اضافه میکنیم.
عدد 100 یک اولویت را مشخص میکند و با توجه به اینکه ما فقط یک pipeline داریم معنی خاصی نمیدهد.
در مرحله آخر تابع parse را در عنکبوت مانند کد قبلی خودمان کامل میکنیم.

```
def parse(self, respons):
        apps_rows = response.json()["singleReply"]["appDetailsV2Reply"]
                "extraContentPageBodyInfo"
        ]["pageBody"]["rows"]
                for i in apps_rows:
                        for j in i["simpleAppList"]["apps"]:
                                yield CafebazaarItem(
                                package_name=j["info"]["packageName"], rating=j["info"]
["rate"]
                                )

                                yield scrapy.Request(
                                self.base_url,
                                method="POST",
                                callback=self.parse,
                                body=self.get_request_body(j["info"]["packageName"]),
```

)

در این کد ابتدا ایتم CafebazaarItem ساخته شده و yield می‌شود تا pipeline آن را ذخیره کند سپس درخواست جدید با این ایتم ساخته شده صدا زده می‌شود تا باعث ایجاد ایتم های بیشتر و کامل شدن اطلاعات ما شود

نتیجه انجام این crawl در کافه بازار بعد از حدود 30 دقیقه به صورت زیر در فایل cafeBazaarData.text موجود است.