ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
ФАКУЛЬТЕТ МАТЕМАТИКИ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ
ДО ВИКОНАННЯ ЛАБОРАТОРНИХ РОБІТ З DATA SCRAPING

(для студентів спеціальності 124 «Системний аналіз»)

Ужгород – 2021

Методичні рекомендації до виконання лабораторних робіт із Data scraping (для студентів спеціальності 124 «Системний аналіз») / Упоряд.: Ю.В. Андрашко. Ужгород: УжНУ, 2021. 24 с

Упорядники:

Андрашко Юрій Васильович, к.т.н., доцент кафедри системного аналізу та теорії оптимізації;

Рецензент: Брила Андрій Юрійович, кандидат фізико-математичних наук, доцент, доцент кафедри системного аналізу та теорії оптимізації

Рекомендовано до друку Науково-методичною комісією факультету математики та цифрових технологій (протокол №4 від 11 січня 2021 року).

Зміст

Вступ

Метою вивчення Data Scraping є формування у здобувачі вмінь та навиків збору та обробки інформації із відкритих джерел даних в мережі Інтернет та підготовка їх для подальшого опрацювання.

Інформація – це відомості про навколишнє середовище (об'єкти, явища, події, процеси тощо), які зменшують міру існуючої невизначеності та неповноти знань про нього.

Дані – це інформація, що подана у формалізованому вигляді, прийнятому для опрацювання автоматичними засобами за можливою участю людини.

Система - будь-який об'єкт, який одночасно розглядається як єдине ціле, і як об'єднана сукупність різнорідних елементів, для досягнення певної мети

Інформаційні система – це система яка забезпечує збір, зберігання, обробку, пошук, видачу даних.

Скрапінг – структуризація даних з веб-сторінок, які призначені для перегляду людиною за допомогою браузера.

Технології скрапінгу:

- Ручна обробки даних.
- Пошук шаблонів (регулярні вирази).
- HTML-аналізатори.
- DOM аналіз.
- Платформи вертикальної агрегації .
- Розпізнавання семантичних анотацій.
- Використання комп'ютерного зору.

Фреймворк Scrapy

Scrapy – це Python фреймворк з відкритим кодом призначений для зручного збору даних із веб сайтів.

Для установки необхідно виконати команду

```
pip install scrapy
```

Після встановлення можливо самостійно створити павука або скористатися командою

```
scrapy genspider [-t template] <name> <domain>
```

Де `template` – це шаблон павука, може приймати одне зі значень `basic, crawl, csvfeed, xmlfeed`. За замовчуванням використовується шаблон `basic`, який і будемо розглядати надалі

`name` – це ім'я павук. Буде згенеровано відповідний файл що містить `NameSpider` та властивість `name = 'name'`

`domain` – домен, де розташований сайт з якого потрібно зібрати дані.

Приклад згенерованого павука наведено нижче:

```python
import scrapy


class NameSpider(scrapy.Spider):
    name = 'name'
    allowed_domains = ['domain']
    start_urls = ['http://domain/']

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

Найпростіший павук містить метод `parse`. Для запуску павука необхідно виконати команду

```
scrapy runspider name.py
```

В результаті виконання команди в консоль буде виведено звіт про результати запуску павука.

Розглянемо детальніше як функціонує фреймворк Scrapy. На рис 1. наведено схему функціонування Scrapy.

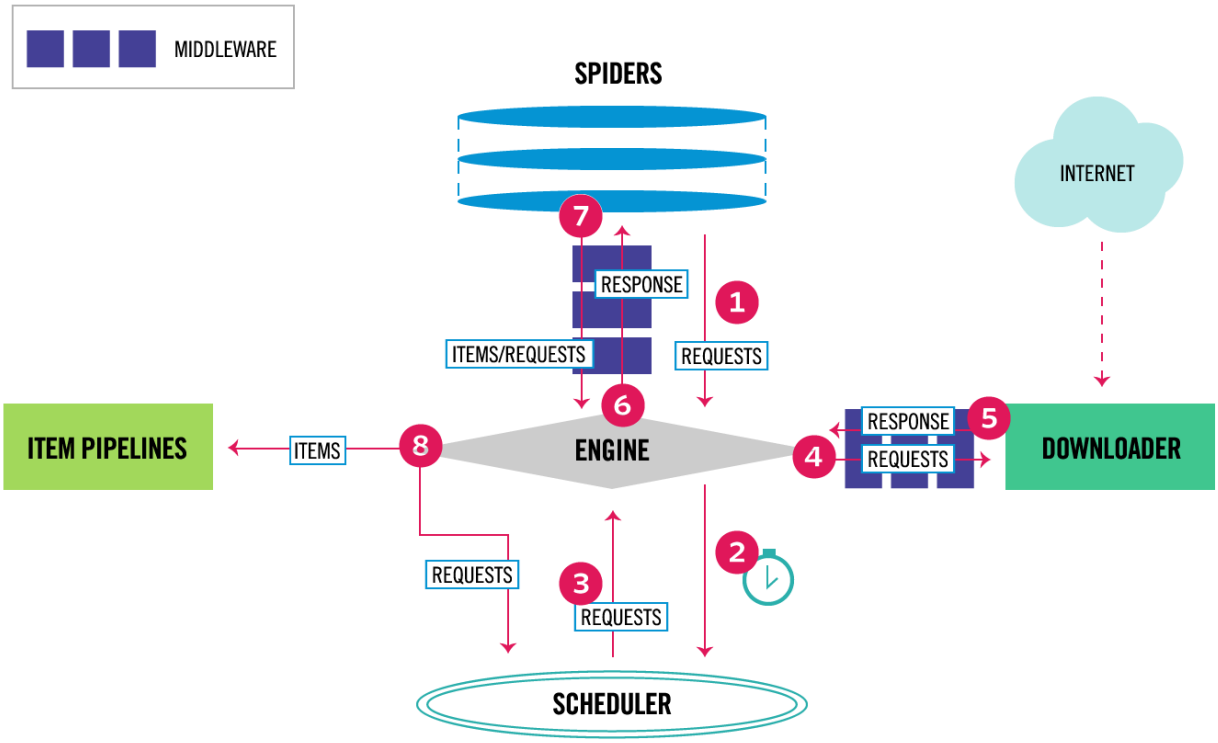


Рис. 1. Схема функціонування Scrapy

При запуску павука для кожного елемента списку що зберігається у властивості `start_urls` генерується початковий запит (1) . Всі запити через `spider middleware` та середовище виконання фреймворка потрапляють потрапляють до планувальника (2). Планувальник зберігає чергу запитів і відповідно до налаштувань Scrapy чи конкретного проєкту (таких як затримка між запитами, кількість одночасних запитів) вибирає запити і передає їх через середовище виконання фреймворка (3) та `downloader middleware` у завантажувач (4). Завантажувач відповідає за встановлення з'єднання з ресурсом через мережу інтернет, надсилання http запиту та отримання відповіді. Завантажувач може підтримувати багато паралельних з'єднань для одночасного надсилання декількох запитів, а також спроби повторного підключення якщо ресурс недоступний. Отримана відповідь через `downloader middleware` (5)

середовище виконання фреймворка та `spider middleware (6)` потрапляє до павука. За замовчуванням відповідь передається як перший параметр методу `parse`. Даний метод повинен здійснити розбір відповіді та виділити дані, які необхідно зібрати із сторінки та посилання на наступні сторінки, на яких можна продовжити збір даних. Павук повинен повертати об'єкт, який представляє зібрані дані, або об'єкт що є нащадком `scrapy.http.Request`. Через `spider middleware` та середовище виконання фреймворка (7) всі відповіді потрапляють до `item pipelines`, а запити – до планувальника (8). Через `item pipelines` відбувається збереження результатів виконання до файлу, вивід в консоль, тощо. Запити ж повторюють шлях початкових запитів (3)-(4)-(5)-(6).

Виконання павука завершується коли планувальник не містить жодного запиту, а обробка всіх запитів та відповідей завершилась. Виконання звершується виводом звіту.

Лабораторна робота №1. Створення павука

Завдання 1. Використовуючи сервіс https://hotline.ua/ зібрати дані (назва моделі, ціна, адреса зображення) про:

1. Телевізори.
2. Ігрові приставки.
3. Відеокарти.
4. Смартфони.
5. Ноутбуки.

Завдання 2. Використовуючи сервіс «Наукова періодика України» (http://www.irbis-nbuv.gov.ua//cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=F&I21DBN=UJRN&P21DBN=UJRN&S21CNR=20&Z21ID=) зібрати дані про

1. всі випуски (Назва, рік, номер) для видань, назва яких починається на літеру "н".

2. всі статті (назва, автори, сторінки), опубліковані в Управління розвитком складних систем

3. всі періодичні видання (назва, місто видання, посилання на сторінку видання).

4. всі випуски (Назва, url обгортки) для видань що поступили в 2020 році.

5. всі статті (назва, автори, сторінки), опубліковані в Науковий вісник Ужгородського університету. Серія : Математика і інформатика

Розглянемо приклад виконання 1 завдання. Для цього спочатку зайдемо на сайт та знайдемо сторінку яка містить дані про ноутбуки https://hotline.ua/computer/noutbuki-netbuki/ рис (2).

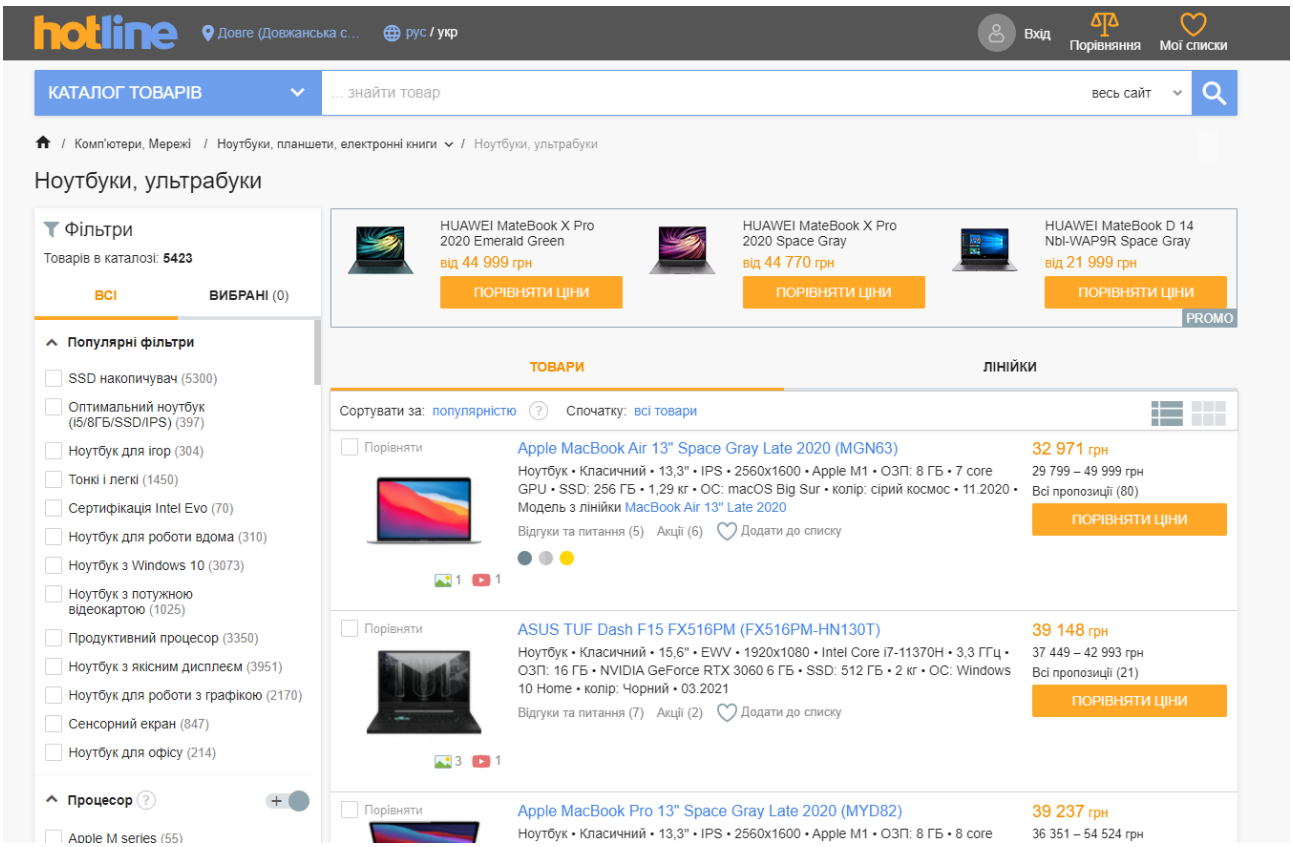


Рис. 2. Приклад сторінки що містить дані.

Згенеруємо павук виконавши команду

```
scrapy genspider laptop hotline.ua
```

Далі змінимо властивість `start_urls` на список, що буде містити єдину початкову адресу `['https://hotline.ua/computer/noutbuki-netbuki/']`

Тепер можна запустити павука командою

`scrapy runspider laptop.py`

Та побачити звіт про успішний запуск.

Далі необхідно здійснити аналіз html коду сторінки. Для цього слід в браузері відкрити сторінку та натиснути клавішу F12 (рис. 3).

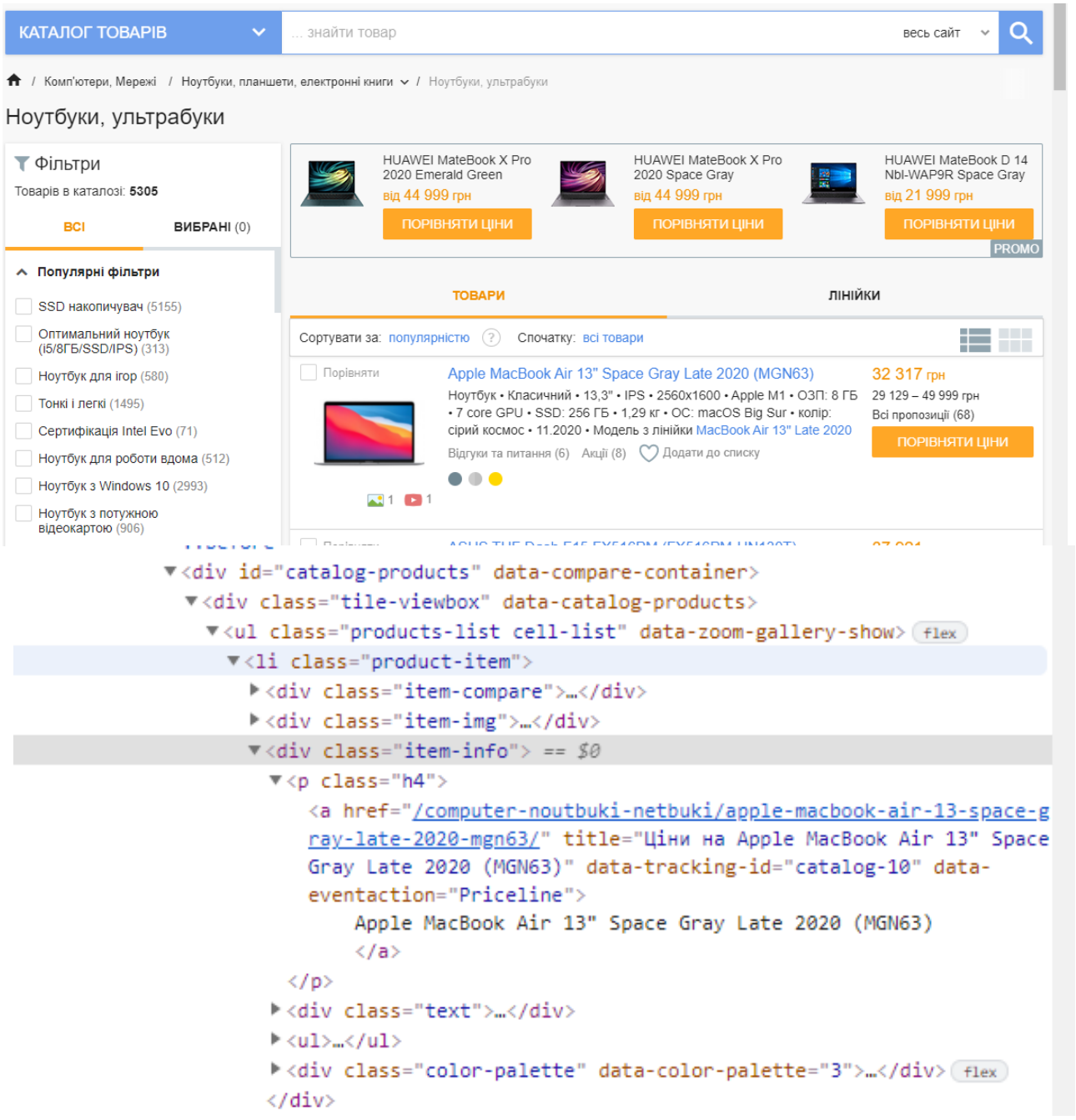


Рис. 3. Дослідження структури сторінки.

Для отримання певного елементу у відповіді Scrapy підтримує запити за допомогою CSS селекторів та XPATH запити. Скористаємось CSS селекторами які відомі. Для отримання колекції елементів списку, кожен із яких містить дані про один ноутбук слід виконати запит `response.css("li.product-item")`. Отримана колекція є ітерованою тому переберемо її циклом `for`.

```
for product in response.css("li.product-item"):
```

На кожному кроці будемо отримувати модель ноутбука. Для цього встановимо що вона записана як текст в посиланні, що вкладено в заголовок 4 рівня. Тому для отримання моделі ноутбука в текстовому виді виконаємо запит

```
product.css("p.h4 a::text").get()
```

Аналогічно отримуємо ціну та URL адресу зображення виконуючи запити `product.css("div.price-md span.value::text").get()` та `product.css("img.img-product::attr(src)").get()`, відповідно.

Отримавши відповідні значення необхідно сформувати об'єкт відповіді та повернути його. Для повернення слід використовувати оператор `yield`.

```
if img != None:
    img_url = "https://hotline.ua/"+img

if model != None and price != None:
    laptop = {}
    laptop["model"] = model.strip()
    laptop["price"] = float(price.replace("\xa0",""))
    laptop["img_url"] = img_url
    yield laptop
```

В наведеному рішенні також відсікаються порожні відповіді та здійснюється очистка даних.

Після збору всіх даних на цій сторінці необхідно перейти на наступну. Дослідивши структуру сторінки можна прийти до висновку що найзручніший спосіб – використання кнопки переходу на наступну сторінку у пагінації (рис 4).

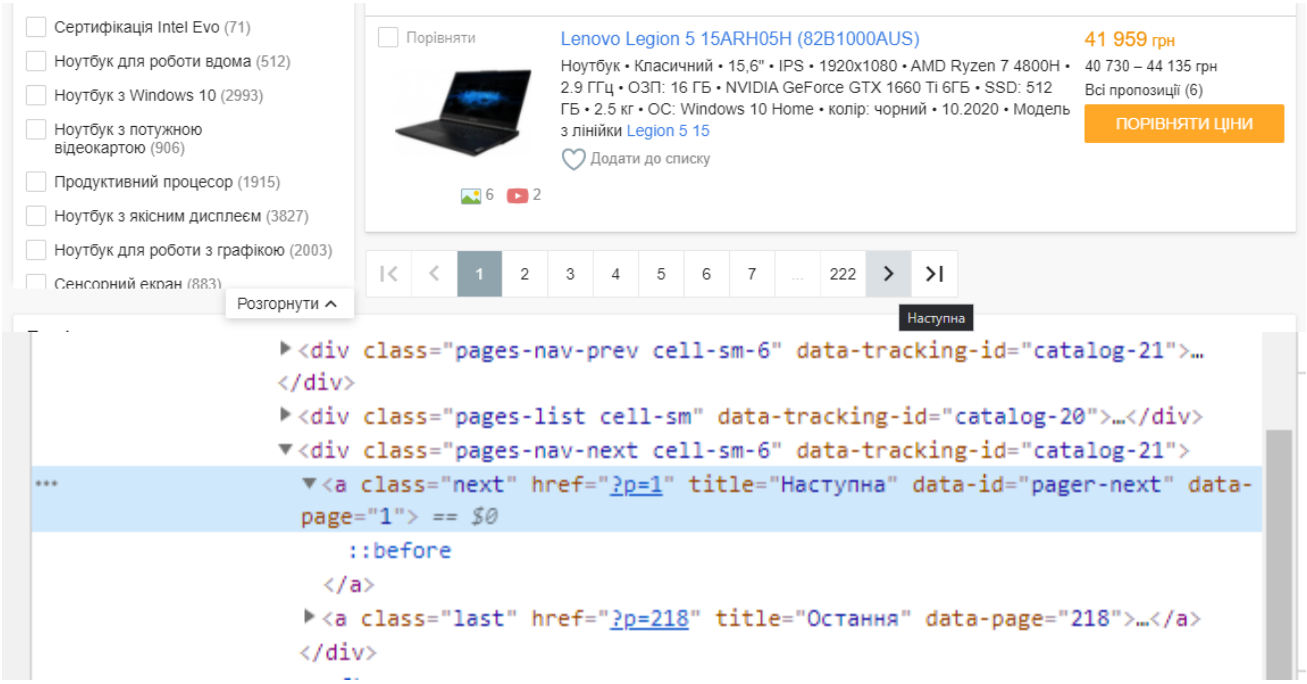


Рис. 1.4. Кнопка переходу на наступну сторінку.

Знаходимо кнопку за відповідним запитом і якщо вона містить URL наступної сторінки то формуємо запит для переходу.

```
next = response.css("a.next::attr(href)").get()
if next != None:
      next_url = "https://hotline.ua/computer/noutbuki-netbuki/"+next
      yield scrapy.Request(next_url)
```

Запустивши павук в консолі Ви побачите відповідь на 1 завдання.

```
2020-05-10 19:09:14 [scrapy.core.scraper] DEBUG: Scraped from <200 https://hotline.ua/computer/noutbuki-netbuki/?p=8>
{'model': 'HP ENVY x360 15-ed1017ur Silver (2X1Q9EA)', 'price': 38491.0, 'img_url': 'https://hotline.ua//img/tx/277/277357564_s265.jpg'}
2020-05-10 19:09:14 [scrapy.core.scraper] DEBUG: Scraped from <200 https://hotline.ua/computer/noutbuki-netbuki/?p=8>
{'model': 'Apple MacBook Pro 13" Space Gray 2020 (MWP52)', 'price': 60290.0, 'img_url': 'https://hotline.ua//img/tx/227/227100780_s265.jpg'}
2020-05-10 19:09:14 [scrapy.core.scraper] DEBUG: Scraped from <200 https://hotline.ua/computer/noutbuki-netbuki/?p=8>
{'model': 'MSI GF75 Thin 10SCXR (GF7510SCXR-003US)', 'price': 25261.0, 'img_url': 'https://hotline.ua//img/tx/276/276480610_s265.jpg'}
2020-05-10 19:09:14 [scrapy.core.scraper] DEBUG: Scraped from <200 https://hotline.ua/computer/noutbuki-netbuki/?p=8>
```

Лабораторна робота №2. Збереження результатів до файлу

Завдання 1. Використовуючи сервіс https://hotline.ua/ зібрати дані (назва моделі, ціна, адреса зображення) про:

1. Телевізори.
2. Ігрові приставки.
3. Відеокарти.
4. Смартфони.
5. Ноутбуки.

Зібрані дані зберегти до:

1. JSON файлу
2. XML файлу
3. CSV файлу
4. JSON файлу
5. XML файлу

Завдання 2. Використовуючи сервіс «Наукова періодика України» (http://www.irbis-nbuv.gov.ua//cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=F&I21DBN=UJRN&P21DBN=UJRN&S21CNR=20&Z21ID=) зібрати дані про

1. всі випуски (Назва, рік, номер) для видань, назва яких починається на літеру "н".
2. всі статті (назва, автори, сторінки), опубліковані в Управління розвитком складних систем
3. всі періодичні видання (назва, місто видання, посилання на сторінку видання).
4. всі випуски (Назва, url обгортки) для видань що поступили в 2020 році.
5. всі статті (назва, автори, сторінки), опубліковані в Науковий вісник Ужгородського університету. Серія : Математика і інформатика

Зібрані дані зберегти до

1. CSV файлу

2. JSON файлу

3. XML файлу

4. CSV файлу

5. JSON файлу

Scrapy містить широкі можливості для збереження результатів збору даних у файл. Для виконання завдання 3 та збереження результатів збору даних у файл можна створити проєкт Scrapy. Для цього необхідно виконати команду

```
scrapy startproject hotline
```

У поточному катологзі буде створено підкаталог hotline що буде містити проєкт зі такою структурою (рис 5):

settings.py –файл що містить налаштування проєкту;

piplines.py – опис `pipline` користувача для обробки результатів збору даних;

middelwares.py – опис `spider` та `downloader middleware` користувача;

items.py – опис типів результатів збору даних;

каталог spiders – опис павуків проєкту.

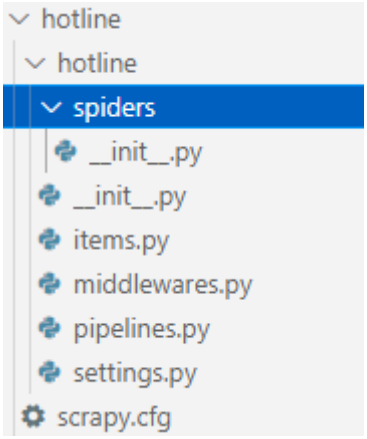


Рис. 5. Структура проєкту Scrapy

Опишемо результати збору даних. Для цього в файлі items.py створимо клас нащадок scrapy.Item всі поля якого є об’єктами класу scrapy.Field

```
class LaptopItem(scrapy.Item):
    model = scrapy.Field()
    price = scrapy.Field()
```

```
    img_url = scrapy.Field()
```

Перемістимо Павук `laptop.py` до каталогу `spiders` та модифікуємо його так, щоб результат був не звичайним об'єктом, а екземпляром класу `LaptopItem`

```
import scrapy
from  hotline.items import LaptopItem

class LaptopSpider(scrapy.Spider):
    name = 'laptop'
    allowed_domains = ['hotline.ua']
    start_urls = ['https://hotline.ua/computer/noutbuki-netbuki/']

    def parse(self, response):
        for product in response.css("li.product-item"):
            model = product.css("p.h4 a::text").get()
            price = product.css("div.price-md span.value::text").get()
            img = product.css("img.img-product::attr(src)").get()
            if img != None:
                img_url = "https://hotline.ua/"+img

            if model != None and price != None:
                laptop = LaptopItem()
                laptop["model"] = model.strip()
                laptop["price"] = float(price.replace("\xa0",""))
                laptop["img_url"] = img_url
                yield laptop


        next = response.css("a.next::attr(href)").get()
        if next != None:
            next_url = "https://hotline.ua/computer/noutbuki-netbuki/"+next
            yield scrapy.Request(next_url)
```

Тепер запустимо проєкт командою

```
scrapy crawl laptop
```

Як і раніше, вивід результатів здійснюється в консоль. Можна побачити послідовність повідомлень про зібрані дані виду:

```
2020-05-10 19:37:52 [scrapy.core.scraper]
DEBUG: Scraped from <200 https://hotline.ua/computer/noutbuki-netbuki/?p=4>
{'img_url': 'https://hotline.ua//img/tx/277/277249941_s265.jpg',
 'model': 'Apple MacBook Air 13" Space Gray 2020 (Z0YJ0)',
 'price': 29180.0}
```

Для виконання завдання необхідно виконати запуск проєкта з параметром `-o`. Для цього в консолі слід виконати команду

```
scrapy crawl laptop -o file.xml
```

Результати буде записано до файлу `file.xml` в форматі xml

Лабораторна робота №3. Збереження результатів до бази даних

Завдання. Використовуючи сервіс https://hotline.ua/ зібрати та зберегти до бази даних дані (назва моделі, ціна, адреса зображення) про

1. Телевізори.
2. Ігрові приставки.
3. Відеокарти.
4. Смартфони.
5. Ноутбуки.

SQLite – це бібліотека написана на мові C, яка забезпечує роботу з легкою локальною базою даних на основі файлу, яка не вимагає окремого серверного процесу і дозволяє отримувати доступ до бази даних за допомогою мови запитів SQL. Деякі програми можуть використовувати SQLite для внутрішнього зберігання даних. Для роботи із цією базою даних спочатку необхідно встановити API інтерфейс для бази даних Sqlite 3.x. Для цього виконати команду.

```
pip install pysqlite3
```

Щоб використовувати модуль, спочатку потрібно створити об’єкт `Connection`, який представляє базу даних. Якщо дані будуть зберігатися у файлі `example.db` то необхідно створити з’єднання con таким чином:

```
import sqlite3
con = sqlite3.connect('example.db')
```

Коли у вас є підключення, ви можете створити об'єкт `Cursor` і викликати його метод `execute ()` для виконання команд SQL. Наприклад, для додавання у таблицю `stocks` нових записів здійснюється таким чином:

```
cur = con.cursor()
cur.execute("INSERT INTO stocks VALUES ('2006-01-05','BUY','RHAT',100,35.14)")
```

Щоб отримати дані після виконання запиту `SELECT`, ви можете або розглядати курсор як ітератор, викликати метод `fetchone ()` курсора, щоб отримати один відповідний рядок, або викликати `fetchall ()`, щоб отримати список відповідних рядків. Наприклад, друк всіх даних із таблиці `stocks` можна здійснити

```
for row in cur.execute('SELECT * FROM stocks ORDER BY price'):
        print(row)
```

Запис у базу даних – це обробка результату збору даних. Для цього доцільно додати `pipline` користувача. Для цього спочатку в файлі settings.py потрібно додати налаштування `ITEM_PIPELINES`. Дане налаштування є словником, ключем якого є клас, а значенням – ціле число від 1 до 999, яке визначає порядок обробки. Чим менше число – тим раніше клас отримає результат для обробки.

```
ITEM_PIPELINES = {
   'hotline.pipelines.HotlinePipeline': 300,
}
```

Клас для обробки результатів містить метод `process_item` до якого потрапляють всі результати обробки та павук який їх повернув як параметри `item` та `spider`. Метод повинен повертати `item` для передачі до наступного обробника.

```
class HotlinePipeline:
    def process_item(self, item, spider):
        return item
```

Клас для обробки результатів також може генерувати виняткову ситуацію `scrapy.exceptions.DropItem` якщо необхідно обірвати оброку. При генеруванні даної виняткової ситуації обробка item припиняється та не

передається до наступного обробника. Наприклад, можна відфільтрувати всі ноутбуки що коштують дорожче 25000 грн.

```
class FilterPipeline:
    MAX_PRICE = 25000

    def process_item(self, item, spider):
        if item["price"] > self.MAX_PRICE:
            raise DropItem(f"{item['model']} is too  expensive")
        return item
```

Також можна використовувати обробку для розрахунку залежних полів. Якщо необхідно розрахувати вартість в доларах США то можна скористатися таким обробником:

```
class CalculateUSDPricePipeline:
    USD_COURSE = 27.62

    def process_item(self, item, spider):
        item["priceUSD"] = item["price"] / self.USD_COURSE
        return item
```

Метод `open_spider` виконується один раз при запуску павука. Він використовується для ініціалізації величин. Для підрахунку унікальних моделей скористаємось типом даних множина. При запуску павука ініціалізуємо попорожню множину та на кожному кроці перевіримо наявність в множині. За умови наявності моделі у множині генеруємо виняткову ситуацію `DropItem`:

```
class FilterUniquePipline:
    def open_spider(self, spider):
        self.unique_items = set()

    def process_item(self, item, spider):
        if item["model"] in self.unique_items:
            raise DropItem("Not unique")
        self.unique_items.add(item["model"])
        return item
```

Для роботи із SQLite базою даних зручно використовувати SQLiteStudio (рис 6).

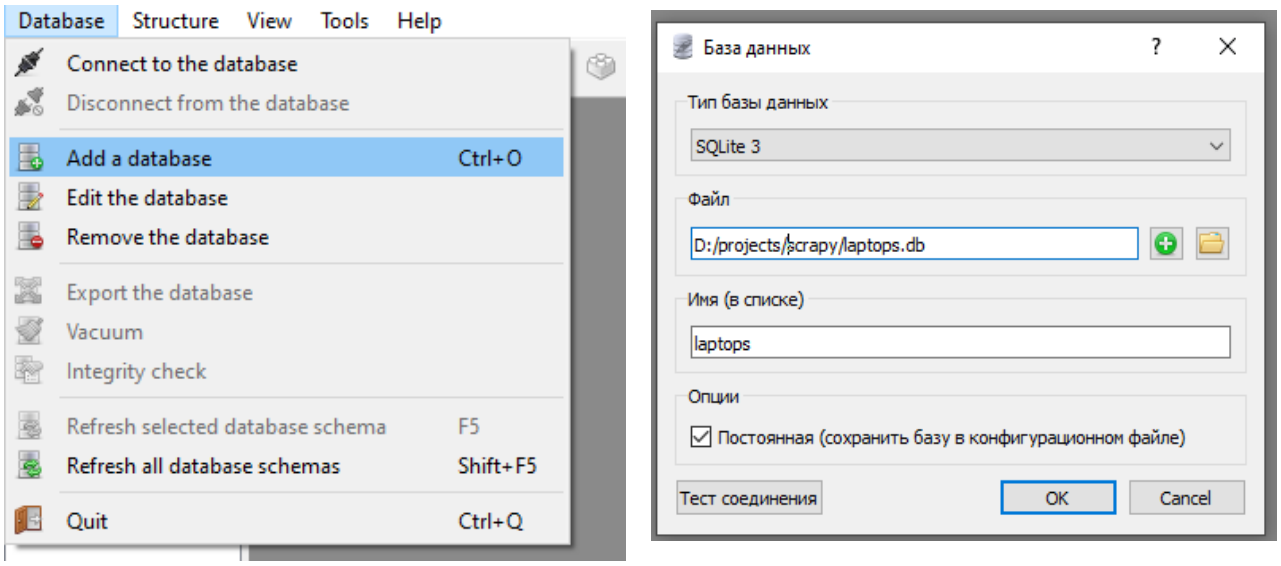


Рис. 6. Створення бази даних в SQLiteStudio.

У SQLiteStudio створюємо таблицю в режимі конструктора (рис 7). SQL код створення генерується автоматично (рис 8.)

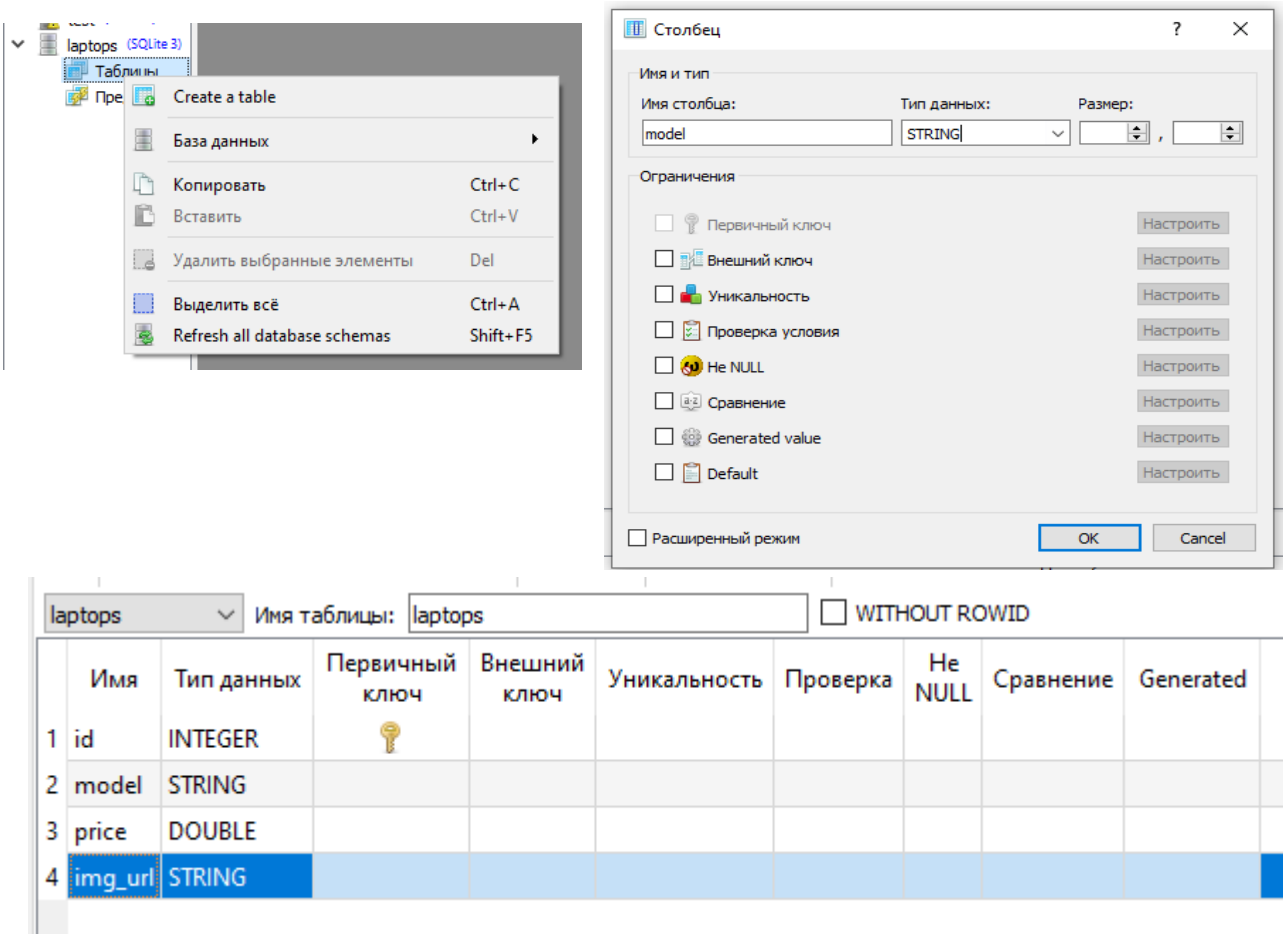


Рис. 7. Створення таблиці в SQLiteStudio

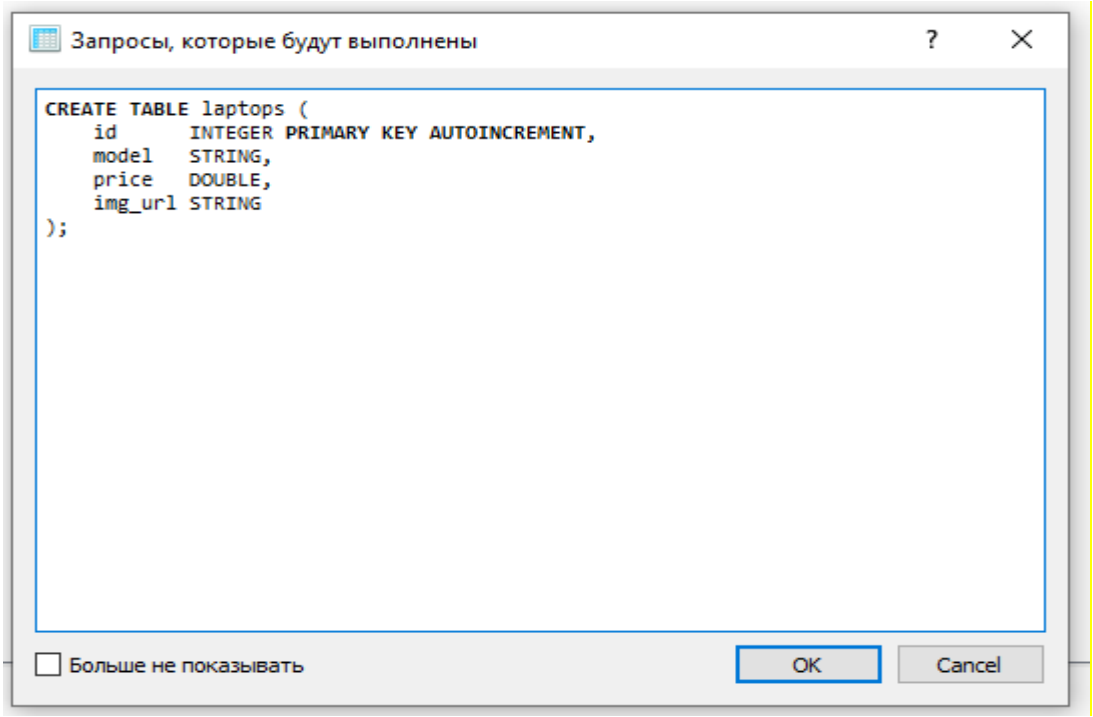


Рис. 8. SQLкод створення таблиці

Для роботи із SQLite базою даних використовуємо Також генеруємо `pipline`. Для роботи із базою даних використовуємо `pysqlite3` – API 2.0 інтерфейс для доступу до бази даних до Sqlite 3.x При запуску павука створюється підключення до бази даних. Та створюємо курсор доступу.

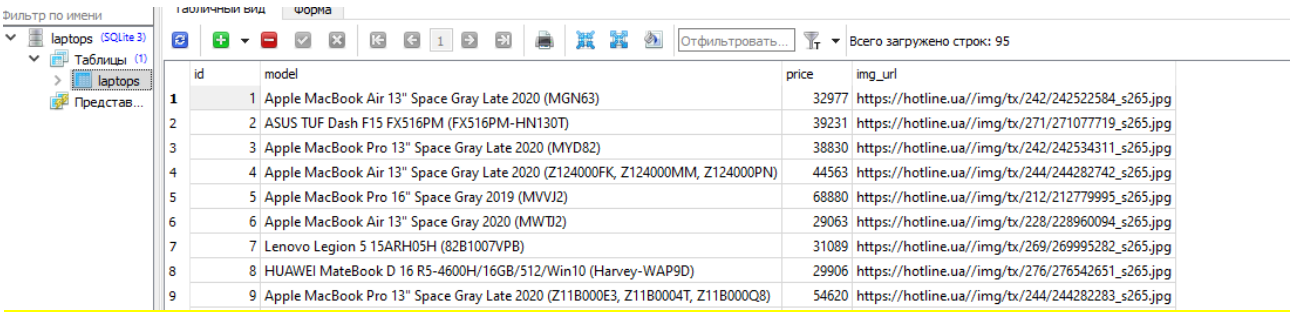
Також слід уникати дублювання даних в базі. Для забезпечення унікальності записів при старті павука завантажуємо всі моделі із бази даних у множину.

При оброці результату відбувається перевірка типу з використанням функції `isinstance`. Даний пайплайн опрацьовує тільки результати типу `LaptopItem`. Результати інших типів передаються наступному обробнику. Для результатів збору даних про ноутбуки перевіряються на унікальність з використанням множини. Використання множини дає змогу значно підвищити швидкодію у порівнянні із методом коли при обробці кожного результату виконується запит до множини на перевірку унікальності.

Метод `close_spider` виконується один раз при завершенні виконання збору даних. Перериваємо підключення до бази даних.

```
import sqlite3
from hotline.items import LaptopItem


class SaveToDBPipeline:
    DB_NAME = "laptops.db"

    def open_spider(self, spider):
        self.connection = sqlite3.connect(self.DB_NAME)
        cursor = self.connection.cursor()
        self.model_set = set()
        for model in cursor.execute("SELECT model FROM laptops"):
            self.model_set.add(model[0])

    def process_item(self, item, spider):
        if not isinstance(item, LaptopItem):
            return item
        cursor = self.connection.cursor()
        if not item["model"] in self.model_set:
            cursor.execute("INSERT INTO laptops (model, price, img_url) VALUES
            (?,?,?)", [item["model"], item["price"], item["img_url"]])
            self.model_set.add(item["model"])
            self.connection.commit()
        return item

    def close_spider(self, spider):
        self.connection.close()
```

Після виконання павука результати збору даних, записані в базу даних переглядаємо в SQLiteStudio (рис 9.)



	id	model	price	img_url
1	1	Apple MacBook Air 13" Space Gray Late 2020 (MGN63)	32977	https://hotline.ua//img/tx/242/242522584_s265.jpg
2	2	ASUS TUF Dash F15 FX516PM (FX516PM-HN130T)	39231	https://hotline.ua//img/tx/271/271077719_s265.jpg
3	3	Apple MacBook Pro 13" Space Gray Late 2020 (MYD82)	38830	https://hotline.ua//img/tx/242/242534311_s265.jpg
4	4	Apple MacBook Air 13" Space Gray Late 2020 (Z124000FK, Z124000MM, Z124000PN)	44563	https://hotline.ua//img/tx/244/244282742_s265.jpg
5	5	Apple MacBook Pro 16" Space Gray 2019 (MVVJ2)	68880	https://hotline.ua//img/tx/212/212779995_s265.jpg
6	6	Apple MacBook Air 13" Space Gray 2020 (MWTJ2)	29063	https://hotline.ua//img/tx/228/228960094_s265.jpg
7	7	Lenovo Legion 5 15ARH05H (82B1007VPB)	31089	https://hotline.ua//img/tx/269/269995282_s265.jpg
8	8	HUAWEI MateBook D 16 R5-4600H/16GB/512/Win10 (Harvey-WAP9D)	29906	https://hotline.ua//img/tx/276/276542651_s265.jpg
9	9	Apple MacBook Pro 13" Space Gray Late 2020 (Z11B000E3, Z11B0004T, Z11B000Q8)	54620	https://hotline.ua//img/tx/244/244282283_s265.jpg

Рис. 9. Приклад таблиці із зібраними даними

При повторному запуску павука результати збору даних не повинні суттєво відрізнятись. Пропонується самостійно модифікувати пайплайн таким чином, щоб при знаходженні моделі в базі даних відбувалось оновлення даних.

Лабораторна робота №4. Розгортання проєкту

Завдання. Розгорнути проєкт створений при виконанні лабораторної роботи 3.

Для того, щоб розгорнути проєкт необхідно виконати такі дії:

1. Установити сервер. Для цього в консолі слід виконати команду

   `pip install scrapyd`

2. Запустити сервер виконавши в окремому вікні консолі.

   `Scrapyd`

   Важливо не закривати це вікно, адже це може призвести до зупинки сервера. Важливо також запам'ятати адресу сервера. За замовчуванням це `http://localhost:6800/`

3. Створити egg файл. Для цього

   a. Редагувати файл конфігурації `scrapy.cfg`

```
[settings]
default = laptops.settings

[deploy]
url = http://localhost:6800/
project = laptops
```

   b. Створити файл конфігурації, наприклад `deploy.py`. Важливо щоб `name` та `entry_point` містили правильну назву проєкту, вказану при його створенні. В цьому прикладі це `hotline`

```
from setuptools import setup, find_packages
setup(
    name="hotline",
    entry_points={'scrapy': ['settings = hotline.settings']},
    version="1.0.1",
    packages=find_packages(),
)
```

   c. Скомпілювати проєкт. Вказати в консолі команду

      `python deploy.py bdist_egg`

   d. Перевірити наявність `egg` файлу в папці `dist`

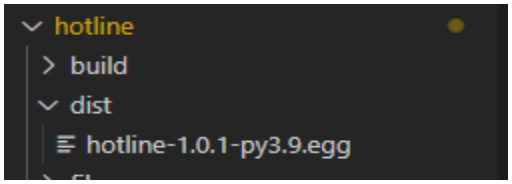


Рис. 9. Приклад egg файлу

4.Завантажити проєкт на сервер. Для цього вказати в консолі команду

```
curl http://localhost:6800/addversion.json -F project=hotline -F version=1.0.1 -F egg=@dist\hotline-1.0.1-py3.9.egg
```

`project` – назва проєкту, рекомендується вказувати як в параметрі `name` в файлі конфігурації

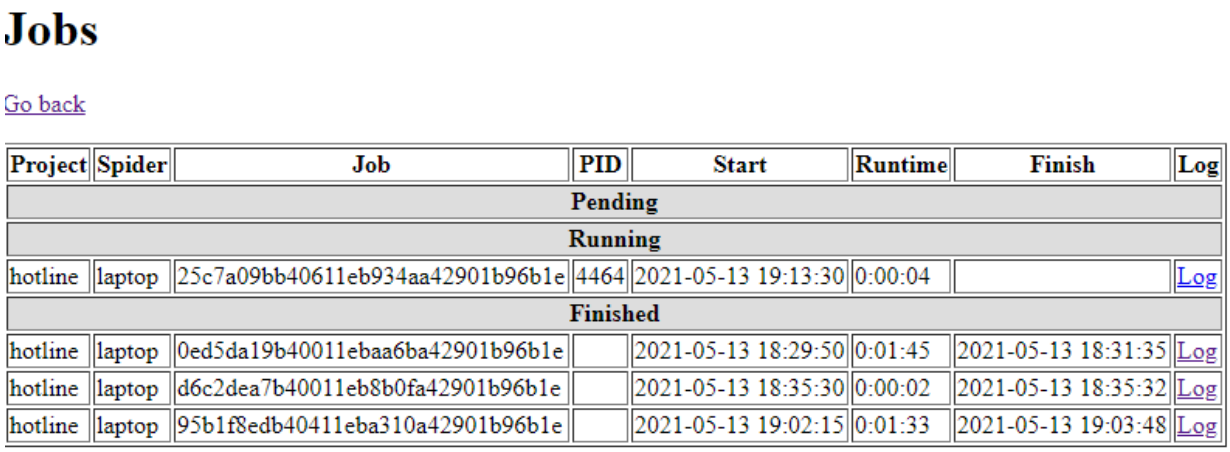
`version` – версія проєкту, як було вказано в параметрі `version` в файлі конфігурації

`egg` – шлях до `egg` файлу.

5. Запустити паук на виконання. Для цього вказати в консолі команду

```
curl http://localhost:6800/schedule.json -d project=hotline -d spider=laptop
```

6. Перевірити статус процесу можна у браузері за його адресою `http://localhost:6800`



Jobs

Go back

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Running							
hotline	laptop	25c7a09bb40611eb934aa42901b96b1e	4464	2021-05-13 19:13:30	0:00:04		Log
Finished							
hotline	laptop	0ed5da19b40011ebaa6ba42901b96b1e		2021-05-13 18:29:50	0:01:45	2021-05-13 18:31:35	Log
hotline	laptop	d6c2dea7b40011eb8b0fa42901b96b1e		2021-05-13 18:35:30	0:00:02	2021-05-13 18:35:32	Log
hotline	laptop	95b1f8edb40411eba310a42901b96b1e		2021-05-13 19:02:15	0:01:33	2021-05-13 19:03:48	Log

Рис 10. Перевірка стану процесів на сервері scrapyd

Якщо при запуску павука на сервері виникає помилка виду `spawnProcess not available since pywin32 is not installed`

То необхідно встановити відповідний пакет виконавши в консолі команду

```
pip install pywin32
```

та перезапустити сервер `scrapyd`.

Список рекомендованих джерел

1. Ryan Mitchell. (2016). Web Scraping with Python: Collecting Data from the Modern Web.
2. Katharine Jarmul, Richard Lawson (2017). Python Web Scraping (2nd ed.)
3. Richard Lawson (2015). Web Scraping with Python: Successfully scrape data from any website with the power of Python.
4. Scrapy 2.5 documentation. URL: https://docs.scrapy.org/en/latest/
5. sqlite3 – DB-API 2.0 interface for SQLite databases URL: https://docs.python.org/3/library/sqlite3.html
6. SQLiteStudio. URL: https://sqlitestudio.pl/

Навчально-методичне видання

Андрашко Юрій Васильович

Методичні рекомендації до виконання лабораторних робіт із Data scraping (для студентів спеціальності 124 «Системний аналіз»)

В авторській редакції

Рекомендовано до друку Науково-методичною комісією факультету математики та цифрових технологій (протокол №4 від 11 січня 2021 року)