POLITECHNIKA KRAKOWSKA

Scraper do pobierania cen butów z zastosowaniem frameworku Scrapy języka Python

---

Projekt z przedmiotu
Automaty, Języki i Obliczenia

Krzysztof Kulig
IV rok Informatyka
134113

SPIS TREŚCI:

Wstęp **3**

Cel projektu **3**

Technologia **3**

Organizacja działania **4**

Stworzenie projektu Scrapy **4**

Tworzenie Spidera **4**

Przetwarzanie danych wejściowych w modułach projektu Scrapy **5**

Uruchamianie scrapera **5**

Zapisywanie danych wynikowych do pliku **5**

Scrapy shell **6**

Dane wynikowe programu **6**

Zapisywanie danych wyjściowych do bazy danych **6**

Sposób rozbudowy projektu **8**

Propozycje rozbudowy projektu **8**

Literatura **9**

Wstęp

Realizowana przeze mnie praca inżynierska o tytule “Serwis internetowy do handlu odzieżą z edycji limitowanych” wymaga tego, aby umieszczone zostały na stronie internetowej nie tylko informacje dotyczące danego produktu, ale dobrym rozwiązaniem jest umieszczenie również informacji na temat wartości, jaką produkt ten osiąga na innej stronie. Wykonanie modułu, który pobiera automatycznie dane wejściowe ze strony Stockx i wyświetla je jako cenę referencyjną. Dzięki takiemu rozwiązaniu użytkownik będzie mógł sprawdzić, czy sprzedający nie zawyża sztucznie ceny danego produktu, co pozwoli na zmniejszenie odsetka oszukujących użytkowników i znacznie poprawi relacje wśród osób handlujących i kupujących. Zadanie to byłoby ciężkie do zrealizowania bez użycia programu, ponieważ znalezienie i pobranie tak dużej ilości różnych danych zajęłoby dużo czasu. Projekt ten powstał w celu zautomatyzowania tego zadania.

Cel projektu

Celem projektu jest zautomatyzowanie procesu pobierania cen referencyjnych butów z innej, szeroko znanej strony służącej do sprzedaży butów z edycji limitowanych. Funkcja ta zrealizowana została poprzez web scraping, czyli automatyczne pobieranie zawartości danej strony internetowej przy użyciu odpowiednio napisanego programu. W ten sposób osoba tworząca bazę danych z cenami referencyjnymi nie będzie musiała spędzać niezliczonej ilości czasu na ręcznym wyszukiwaniu i zapisywaniu wartości do bazy danych, a zostanie to zrealizowane przy pomocy programu napisanego w Pythonie, który automatycznie pobiera dane ze strony internetowej i zapisuje je bezpośrednio do bazy danych.

Technologia

Do napisania scrapera, który będzie realizował wszystkie zadania użyłem frameworku języka programowania Python o nazwie Scrapy. Służy on do pobierania danych ze stron internetowych. Scrapy pozwala na swobodne określanie poszukiwanych danych za pomocą selektorów CSS.

Organizacja działania

- Określenie adresów stron internetowych, które chcemy przeszukiwać,
- Wskazanie głównego elementu, który zawiera wszystkie informacje,
- Wyciągnięcie z głównego znacznika HTML poszczególnych danych,
- Zapis pobranych danych do pliku o rozszerzeniu CSV.

Stworzenie projektu Scrapy

W celu stworzenia projektu Scrapy, najpierw należy się upewnić, iż na komputerze zainstalowana jest odpowiednia wersja Pythona oraz samego frameworka Scrapy. W tym projekcie wykorzystana została wykorzystania platforma Anaconda, dzięki której zainstalowany został również Scrapy. Aby utworzyć projekt Scrapy, należało otworzyć Anaconda Prompt, który zainstalowany został w trakcie instalacji Anacondy. Następnie należało wpisać polecenie “scrapy startproject nazwa”.

Tworzenie Spidera

Spidery są to klasy zdefiniowane przez użytkownika, które Scrapy używa w celu pobierania informacji ze stron internetowych. Podstawowe elementy Spidera to:

- name - jest to pole, w którym użytkownik określa nazwę. Musi być ona unikatowa w ramach jednego problemu. Znaczy to, że użytkownik nie może nadać dwa razy tej samej nazwy w jednym projekcie.
- allowed_domains - jest to lista ciągów znaków zawierająca nazwy domen, które program może przeszukiwać. Jest to parametr opcjonalny.
- start_urls - lista adresów, które Spider będzie przeszukiwał.
- parse() - jest to metoda, która jest wykorzystywana do obsługi danych wejściowych, które zostaną zwrócone w ramach zapytania skierowanego przez program do serwera strony określonej w liście start_urls.

Przetwarzanie danych wejściowych w modułach projektu Scrapy

Dane wejściowe, które zostały pobrane bezpośrednio ze strony internetowej Grail Point można przetwarzać na różne sposoby. W moim przypadku jest potrzeba, aby dane te zapisać do bazy danych. W pliku items.py można określić strukturę obiektu, który ma zostać utworzony z pobranych danych. Plik pipelines.py służy do definiowania funkcji, które są używane w celu przetwarzania danych wejściowych w taki sposób, aby otrzymać odpowiednio sformatowane dane wyjściowe w wybrane do tego miejsce. Właśnie w pliku pipelines.py odbywa się połączenia z bazą danych i zapisanie pobranych informacji o produktach do tabeli.

Uruchamianie scrapera

Uruchamianie programu scrapującego strony internetowe w przypadku frameworku Scrapy odbywa się w konsoli. Należy wpisać polecenie “scrapy crawl name”, gdzie name to nazwa, którą nadaliśmy swojemu Spiderowi. Po wpisaniu tego polecenia prawidłowo napisany scraper powinien się wykonać. W przypadku, gdy podczas wykonywania pojawią się jakieś błędy, w konsoli zostaną wyświetlone informacje na temat błędów.

Zapisywanie danych wynikowych do pliku

Jeżeli chcemy zapisać dane wynikowe do pliku, możemy dodać do polecenia uruchamiającego spidera “-o fileName.format”, gdzie fileName to nazwa pliku, którą chcemy ustawić, a format to nazwa rozszerzenia pliku, które nas interesuje. Scrapy umożliwia zapis danych do pliku JSON, CSV, XML oraz Python pickle. Możemy zapisać dane do pliku modyfikując moduł “settings.py” naszego projektu. W tym przypadku należy dodać do pliku z ustawieniami słownik FEEDS i określić w nim nazwę pliku oraz jego format.

```
custom_settings = {
    'FEEDS': { 'prices.csv': { 'format': 'csv', 'overwrite': True}}
}
```

Scrapy shell

Dla osób, które zaczynają swoją przygodę z web scrapingiem, Scrapy może okazać się dobrym rozwiązaniem. Wybieranie interesujących nas fragmentów strony internetowej przy pomocy selektorów css jest intuicyjne i stosunkowo proste. Jednak należy nabrać wprawy w odpowiednim przeszukiwaniu struktur stron internetowych. W takim przypadku z pomocą przychodzi nam Scrapy shell. Narzędzie to pozwala nam na sprawdzenie, czy pobieranie danych z interesujących nas stron internetowych jest w ogóle możliwe oraz pozwala na swobodne przemieszczanie się po strukturze strony w celu wyszukania interesujących nas danych.

Dane wynikowe programu

W wyniku wykonania Spidera zostają dodane do bazy danych informacje dotyczące butów zamieszczonych na sprzedaż w serwisie Grail Point. Scraper pobiera nazwę producenta, nazwę modelu oraz jego cenę i dodaje te dane do utworzonej tabeli w bazie danych

Zapisywanie danych wyjściowych do bazy danych

Spider pobiera dane ze strony internetowej informacje dotyczące butów i zapisuje je do odpowiednich obiektów, których budowa określona została w pliku “items.py”. Następnie w pliku “pipelines.py” w funkcji __init__ zostaje zainicjowane połączenie z bazą danych MySQL. W moim przypadku jest to lokalna baza danych, której obsługa jest realizowana przez program XAMPP. Aby połączenie z bazą danych w pliku Python było możliwe, należy zainstalować odpowiednie biblioteki przy pomocy polecenia

```
pip install mysql mysql-connector-python
```

Następnie należy zaimportować mysql.connector do pliku pipelines. Połączenie z bazą danych jest realizowane przy pomocy bloku:

```
self.conn = mysql.connector.connect(
    host = 'localhost',
    user = 'root',
    password = '',
    database = 'aijo'
)
```

Jak widać na powyższym zrzucie ekranu jako argumenty funkcji connect() należy podać dane potrzebne do połączenia się z bazą danych, czyli nazwę hosta, nazwę użytkownika, hasło użytkownika oraz nazwę bazy danych.

Kolejnym krokiem jest utworzenie kursora, który jest używany do wykonywania operacji na bazie danych.

```
self.cur = self.conn.cursor()
```

Po wykonaniu tych operacji należy upewnić się, że tabela, do której chcemy zapisać dane pobrane ze strony internetowej istnieje. Aby to osiągnąć trzeba wykonać operację CREATE TABLE IF EXISTS wewnątrz funkcji execute kursora

```
self.cur.execute("""
CREATE TABLE IF NOT EXISTS prices(
    id int NOT NULL auto_increment,
    company varchar(100),
    name varchar(255),
    price varchar(100),
    PRIMARY KEY (id)
)
""")
```

Kolejnym krokiem w celu zapisania danych do bazy MySQL jest modyfikacja funkcji process_item() w pliku “pipelines.py”, która jest wykonywana za każdym razem gdy Spider pobierze jakieś informacje i przekaże je do odpowiedniego obiektu. Funkcja ta ma za zadanie obsługę samego dodawania pobranych ze strony internetowej informacji dotyczących butów do bazy danych. Jest to wykonywane przy pomocy funkcji execute() wykonanej na utworzonym przez nas wcześniej kursorze.

```
def process_item(self, item, spider):

    self.cur.execute("insert into prices (company, name, price) values (%s, %s, %s)", (
        item["company"][0],
        item["name"][0],
        item["price"]
        ))
```

Aby zatwierdzić i zrealizować wykonane przez funkcję execute polecenie należy wykonać funkcję commit().

```
self.conn.commit()
```

Należy również dodać funkcję close_spider(), która będzie wykonywać się w momencie, gdy nasz spider zakończy pracę i będzie odpowiedzialna za zniszczenie zmiennej, do której przypisano kursor oraz połączenie do bazy danych.

```
def close_spider(self, spider):
    self.cur.close()
    self.conn.close()
```

Sposób rozbudowy projektu

Aby dodać do projektu dodatkowe moduły scrapujące strony internetowe należy dodać kolejne spidery. Trzeba zdefiniować ponownie adresu URL, które program ma przeszukać, zdefiniować dokładne położenie interesujących nas danych poprzez wykorzystanie selektorów CSS. Do pliku items.py należy dodać odpowiednią klasę, której pola będą odpowiadały danym pobranym z internetu. Następnie należy odpowiednio skonfigurować plik pipelines.py poprzez dodanie do niego odpowiednich funkcji formatujących oraz obsługujących dane wejściowe.

Propozycje rozbudowy projektu

Zaimplementowany przeze mnie scraper można łatwo rozbudować i rozszerzyć jego funkcjonalność. Oto kilka pomysłów na rozbudowę przedstawionego w tej dokumentacji scrapera:

- dodanie innych spiderów obsługujących inne kategorie odzieży niż buty,
- zmiana struktury bazy danych w taki sposób, aby była możliwość zapisania do niej produktów o różnych kategoriach.

- dodanie dodatkowej opcji zapisu danych do pliku CSV w celu utworzenia kopii danych poza bazą danych.

Literatura

https://docs.scrapy.org/en/latest/index.html - dokumentacja frameworku Scrapy z przykładami kodu oraz rozległymi opisami poszczególnych elementów.
https://scrapeops.io/python-scrapy-playbook/scrapy-save-data-mysql/ - rozległy opis działań, jakie należy wykonać w celu zapisania pobranych danych do bazy MySQL.