26.Python Scrapy2

1. Python 고급 Crawling 기술 Scrapy Project 생성1

1. All Category Project 개요

- 전체 Category search후 관련 Sub Category Crawling 하는 Program

2. **scrapy Project 생성**

1) C:₩Users₩ktg₩python-env> # Directory 이동 만들기

2) virtScrapy₩Scripts₩activate.bat → (virtScrapy) C:₩Users₩ktg₩python-env> # 가상 환경 전환됨

Scrapy1에서 만들었다면 Skip

3) (virtScrapy) C:₩Users₩ktg₩python-env>pip install scrapy # 가상 환경 에서 scrapy Framework 설치하기

4) (virtScrapy) C:₩Users₩ktg₩python-env>cd C:₩PyCharmProject₩Sources # 가상 환경 에서 만들 Project로 이동

5) (virtScrapy) C:₩PyCharmProject₩Sources>scrapy startproject ktgScrapy20 # 가상 환경 에서 . All *Category* Project 만들기

6) spider (크롤러 이름) 생성 방법

- 크롤링 프로젝트 내에, 여러 가지 크롤러(scrapy에서는 spider라고 함) 있을 수 있으므로, 각 크롤러의 이름을 지정

[1] scrapy genspider 이용

① (virtScrapy) C:₩PyCharmProject₩Sources>cd C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20

② (virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>

scrapy genspider gmarket6 corners.gmarket.co.kr/Bestsellers

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>

scrapy genspider gmarket7 corners.gmarket.co.kr/Bestsellers

[2] pycharm 으로 작성

① ktgScrapy20₩ktgScrapy20/spiders 디 렉토리에 gmarket6,7 py 파일(템플릿)이 생김

② 직접 scrapy genspider 명령을 사용하지 않고, 만들어도 됨

2. Python 고급 Crawling 기술 Scrapy Project 생성2

3. spider (크롤러) 실행

1 (virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>scrapy crawl gmarket6

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>scrapy crawl gmarket7

2) 다양한 데이터 포멧으로 아이템 저장하기(gmarket5 예시)

① csv, xml, json 포멧

② 터미널 환경에서, ecommerce 폴더에서 다음 명령

scrapy crawl 크롤러명 -o 저장할파일명 -t 저장포멧

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>scrapy crawl gmarket7 -o gmarket7.csv -t csv

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>scrapy crawl gmarket7 -o gmarket7.xml -t xml

json 파일을 확인하면, 한글문자가 깨져나옴

(virtScrapy) C:₩PyCharmProject₩Sources₩ktgScrapy20₩ktgScrapy20>scrapy crawl gmarket7 -o gmarket7.json -t json

3. Python 고급 Crawling 기술 Scrapy 기본 구조1

1. Scrapy 기본 Template 구조

1) 클래스 이름은 마음대로 정하면 됨, 단 scrapy.Spider 를 상속

2) name이 크롤러(spider)의 이름

3) allowed_domains는 옵션 (삭제해도 무방함)

- 별도 상세 설정으로 허용된 주소 이외의 주소는 크롤링 못하게끔 하는 기능을 위한 변수

4) start_urls 가 중요함. 크롤링할 페이지 주소를 나타냄.

- parse 함수는 클래스의 메서드로 response를 반드시 인자로 받아야 함
- response에 start_urls 에 기록된 주소의 크롤링 결과가 담아 오기 때문

5) response 확인하기

- response.text 에 크롤링된 데이터가 담겨져 있음

2. gmarket.py 예시 문장

```
# -*- coding: utf-8 -*-
import  scrapy


class GmarketSpider(scrapy.Spider):
    name = 'gmarket7'          # 'gmarket6
    def start_requests(self):
        yield scrapy.http.Request(url='http://corners.gmarket.co.kr/Bestsellers',
                                  callback=self.parse_mainPages)
```

3. Python 고급 Crawling 기술 Scrapy 기본 구조2

1. settings.py

```
# Obey robots.txt rules
ROBOTSTXT_OBEY = False

# 한글 Setting
FEED_EXPORT_ENCODING = 'utf-8'


# Log gmarket7
LOG_FILE="log.txt"
DUPEFILTER_CLASS = 'scrapy.dupefilters.BaseDupeFilter'
```

4. Python 고급 Crawling 기술 Crawler 적용예시1

3. gmarket6.py(Spider) 예시

1) 전체 Category 파악

① start_urls이 없어도 가장 먼저 실행 되는 함수 정의
start_requests(self):

② scrapy.http.Request(url) 주소 호출

③ 2번 수행후 callback에 의해 parse(self, response) 호출

2) **parse_mainPag에서** 각각의 Sub Category까지 callback

① callback은 parse를 다른 함수(**parse_mainPage**)로 이름 수정 가능

② category_links에 = response.css ('**div.gbest-cate ul.by-group li a::attr(href)**'). getall()을 추가함으로써 css로 전달하는 추가 Filtering 가능

③ category_links에 따라 해당 URL Callback 이때 meta를 Dictionary 형식으로 Parameter 전달가능

3) 각각의 Sub Category까지 crawling

① 해당하는 Item 각각 을 가져옴
(ranking, title, ori_price, dis_price, discount_percent)

② 조건에 맞는 연산 수행 (전처리 및 조건 수행)

③ 해당 값 출력

1) gmarket6.py 예시 문장 (start_requests(self)

```
def start_requests(self):
    yield scrapy.http.Request
            (url='http://corners.gmarket.co.kr/Bestsellers',
             callback=self.parse_mainPage)
```

호출

2) gmarket6.py 예시 문장 (**parse_mainPage**(self, response))

```
def parse_mainPage(self, response):
    category_links = response.css('div.gbest-cate ul.by-group li a::attr(href)').getall()
    category_names = response.css('div.gbest-cate ul.by-group li a::text').getall()
    for index, category_link in enumerate(category_links):
        yield scrapy.http.Request(
                url='http://corners.gmarket.co.kr'+category_link,
                callback=self.parse_mainCategory,
                meta={'mainCategoryName':category_names[index]}
        )
```

호출

3) gmarket6.py 예시 문장 (**parse_mainCategory**(self, response))

```
def parse_mainCategory(self, response):
    print('parse_mainCategory->', response.meta['mainCategoryName'])
    # 해당 css의 범위를 줄인 형태의 best_items css
    best_items = response.css('div.best-list')
    for  index,item in enumerate(best_items[1].css('li')):
        ranking = index+1
        title = item.css('a.itemname::text').get()
        ori_price = item.css('div.o-price::text').get()

        .....................
    print(ranking , title, ori_price, dis_price, discount_percent)
```

4. Python 고급 Crawling 기술 Crawler 적용예시2

3. gmarket7.py(Spider) 예시

1) 전체 Category 파악

① start_urls이 없어도 가장 먼저 실행 되는 함수 정의
start_requests(self):

② scrapy.http.Request(url) 주소 호출

③ 2번 수행후 callback에 의해 parse(self, response) 호출

2) **parse_mainPages에서** 각각의 **parse_items,**
parse_subcategory 까지 callback

① ***1st Category*** callback(**parse_items**)은 해당 Item을 추출하고 **yield 하여 추가로 Data Extract 가능**

② ***2nd category*** **crawling 은 main Category 아래 sub category를 Crawling 하여 역시 같은** callback(**parse_items**)은 해당 Item을 추출하고 **yield 하여 추가로 Data Extract 가능**

③ gmarket best 의 main **category 선택후 sub category 각각을 탐방 Item을 저장**

1) start_requests(self)

```
def start_requests(self):
    yield scrapy.http.Request
            (url='http://corners.gmarket.co.kr/Bestsellers',
             callback=self. parse_mainPages)
```

1.호출

2) **def parse_mainPages**(self, response):

```
category_links = response.css('div.gbest-cate ul.by-group li a::attr(href)').getall()
category_names = response.css('div.gbest-cate ul.by-group li a::text').getall()
# 1st Category  (2.호출) Main Page ALL 부분의 모든 Item 탐방
for index, category_link in enumerate(category_links):
        yield scrapy.http.Request(callback=self.parse_items, ...
# 2nd category crawling ((3.호출)
for index, category_link in enumerate(category_links):
    yield scrapy.http.Request(callback=self. parse_subcategory,  ....
```

3.호출

2.호출

3) **def parse_subcategory**(self, response):

```
subcategory_links = response.css('div.navi.group > ul > li >
        a::attr(href)').getall()
subcategory_names = response.css('div.navi.group > ul > li >
        a::text').getall()
for index, subcategory_link in enumerate(subcategory_links):
        yield scrapy.Request
    (url='http://corners.gmarket.co.kr' + subcategory_link,
     callback=self.parse_items,
```

4.호출

4) **def parse_items**(self, response):

```
    print('parse_mainCategory->',
response.meta['mainCategoryName'])
    # 해당 css의 범위를 줄인 형태의 best_items css
    best_items = response.css('div.best-list')
    for  index,item in enumerate(best_items[1].css('li')):
        doc = Ktgscrapy20Item()
        ranking = index+1
        title = item.css('a.itemname::text').get()
        ori_price = item.css('div.o-price span::text').get()
        dis_price = item.css('div.s-price strong span span::text').get()
        discount_percent = item.css('div.s-price em::text').get()
          .....................
        doc['main_category_name'] = response.meta['mainCategoryName']
        doc['sub_category_name'] = response.meta['subCategory_name']
        doc['ranking'] = ranking
        yield  doc
```