



WebScrapping dengan Python Menggunakan Scrapy



THANK YOU!

Melakukan Crawling

```
scrapy crawl olx_motor -o output.json
```

Setelah konfigurasi dilakukan selanjutnya adalah melakukan crawling


output.json :

```
[
{"Harga": "Rp 10.700.000", "Tahun": "2016", "KM": "30.000-35.000 km", "Brand": "Honda"},
{"Harga": "Rp 17.000.000", "Tahun": "2012", "KM": "5.000-10.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 5.100.000", "Tahun": "2010", "KM": "10.000-15.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 33.500.000", "Tahun": "2022", "KM": "0-5.000 km", "Brand": "Honda"},
{"Harga": "Rp 257.000.000", "Tahun": "2014", "KM": "15.000-20.000 km", "Brand": "Kawasaki"},
{"Harga": "Rp 12.000.000", "Tahun": "2018", "KM": "45.000-50.000 km", "Brand": "Honda"},
{"Harga": "Rp 42.500.000", "Tahun": "2019", "KM": "10.000-15.000 km", "Brand": "Kawasaki"},
{"Harga": "Rp 8.600.000", "Tahun": "2005", "KM": "35.000-40.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 9.000.000", "Tahun": "2003", "KM": "10.000-15.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 27.500.000", "Tahun": "2021", "KM": "20.000-25.000 km", "Brand": "Honda"},
{"Harga": "Rp 27.500.000", "Tahun": "2021", "KM": "20.000-25.000 km", "Brand": "Honda"},
{"Harga": "Rp 4.000.000", "Tahun": "2011", "KM": "50.000-55.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 27.500.000", "Tahun": "2021", "KM": "20.000-25.000 km", "Brand": "Honda"},
{"Harga": "Rp 27.500.000", "Tahun": "2021", "KM": "20.000-25.000 km", "Brand": "Honda"},
{"Harga": "Rp 17.000.000", "Tahun": "2019", "KM": null, "Brand": "Honda"},
{"Harga": "Rp 18.000.000", "Tahun": "2020", "KM": "5.000-10.000 km", "Brand": "Honda"},
{"Harga": "Rp 26.500.000", "Tahun": "2021", "KM": "20.000-25.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 8.200.000", "Tahun": "2017", "KM": null, "Brand": "Yamaha"},
{"Harga": "Rp 55.500.000", "Tahun": "2022", "KM": "5.000-10.000 km", "Brand": "Yamaha"},
{"Harga": "Rp 17.000.000", "Tahun": "2023", "KM": "0-5.000 km", "Brand": "Yamaha"}
]
```



```
1  yield {
2      'Harga': harga.strip() if harga else None,
3      'Tahun': tahun if tahun else None,
4      'KM': km if km else None,
5      'Brand': brand if brand else None,
6  }
```


yield {...}: Setelah mengekstrak data, kita menghasilkan data tersebut sebagai Python dictionary. Python yield digunakan seperti return statement tapi ini mengembalikan generator. Generator adalah iterator yang dapat kita iterasi hanya sekali karena hanya menghasilkan nilai satu kali dan tidak menyimpan nilai-nilai tersebut. Jika pernyataan None, data tersebut tidak dihasilkan dan mengembalikan None.



```
1 def parse_motor_detail(self, response):
2     harga = response.css('span[data-aut-id="itemPrice"]::text').extract_first()
3     tahun = response.css('span[data-aut-id="value_m_year"]::text').extract_first()
4     km = response.css('span[data-aut-id="value_mileage"]::text').extract_first()
5     brand = response.css('span[data-aut-id="value_make"]::text').extract_first()
6
```

`def parse_motor_detail(self, response):` : Ini adalah callback untuk parsing halaman detail motor.

Di dalam metode ini, kita mengekstrak detail seperti harga, tahun, jarak tempuh, dan merek motor. Kita mengekstrak ini dengan menggunakan CSS Selector.



```
1 def parse(self, response):
2     motor_links = response.xpath('//a[contains(@href, "/item/")]@href').getall()
3
4     for link in motor_links:
5         absolute_url = response.urljoin(link)
6         yield scrapy.Request(url=absolute_url, callback=self.parse_motor_detail)
7
8     next_page = response.css('a.next::attr(href)').extract_first()
9     if next_page:
10        yield scrapy.Request(url=next_page, callback=self.parse)
```

def parse(self, response): : Ini adalah metode yang Scrapy gunakan sebagai callback, fungsi ini dipanggil oleh scrapy sekali mendapatkan response.

motor_links = response.xpath('//a[contains(@href, "/item/")]@href').getall()

Dalam metode ini, kita pertama mendapatkan semua link motor dari halaman web dengan menggunakan Xpath.

Kemudian untuk setiap link di motor_links, kita menghasilkan Request baru ke URL absolut motor tersebut, dengan callback ke metode parse_motor_detail.

```
1 custom_settings = {  
2     'DOWNLOAD_DELAY': 0.5,  
3     'USER_AGENT': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'  
4 }
```

custom_settings = {...}: Ini adalah konfigurasi custom yang didefinisikan untuk spider ini. Setting 'DOWNLOAD_DELAY' digunakan untuk menghindari banning dari website yang bersangkutan dengan memberikan delay 0.5 detik setiap download. 'USER_AGENT' adalah string identifikasi yang digunakan oleh protokol HTTP untuk mendefinisikan jenis dan versi browser web.



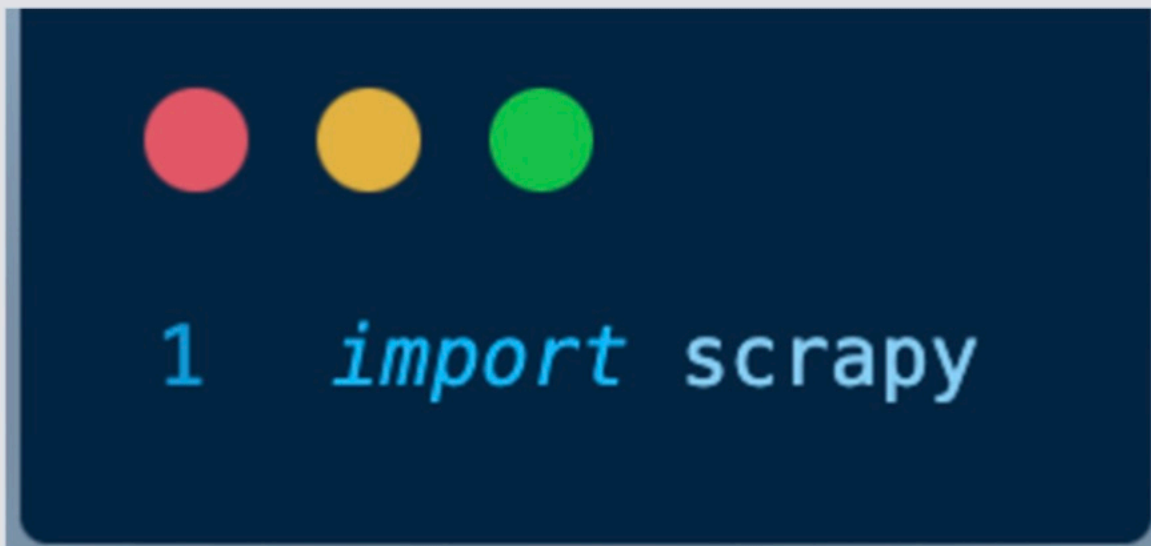
```
1 class OLXMotorSpider(scrapy.Spider):  
2     name = "olx_motor"  
3     start_urls = [  
4         'https://www.olx.co.id/motor-bekas_c200',  
5     ]
```

class OLXMotorSpider(scrapy.Spider): :
Ini adalah deklarasi class spider.
Spider adalah class yang
didefinisikan pengguna dan Scrapy
menggunakan untuk scrape
informasi dari situs web.

name = "olx_motor" : Ini adalah
nama identifikasi spider yang
harus unik dalam sebuah proyek.

start_urls = ['<https://www.olx.co.id/motor-bekas_c200>'] : Ini adalah list URL dimana spider akan
mulai melakukan scraping.

Konfigurasi spider_olx.py



import scrapy : Bagian ini import library scrapy, sebuah framework python untuk web scraping yang menyediakan semua fitur yang diperlukan untuk web scraping seperti mengikuti link, mengekstrak data, dll.

Membuat Spider olx

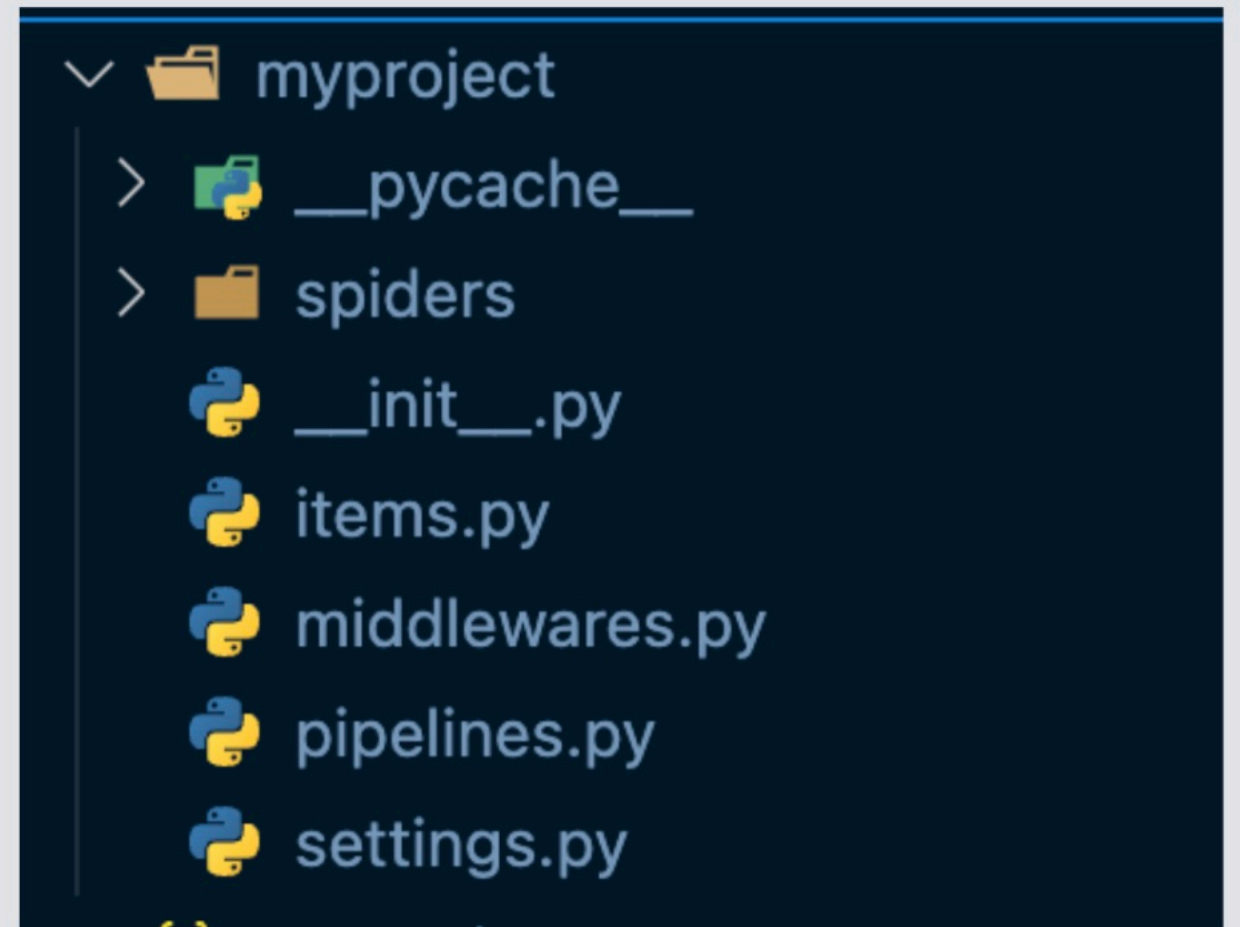
```
scrapy genspider olx_spider https://www.olx.co.id/motor-bekas_c200
```

Tahap ketiga yang harus dilakukan adalah membuat spider yang akan melakukan scrapping

Membuat Scrappy Project

```
scrapy startproject myproject
```

Tahap kedua yang harus dilakukan adalah membuat direktori project scrappy kita



Install Scrappy

```
pip install scrapy
```

Tahap pertama yang harus dilakukan adalah menginstall library scrapy pada python environment kita