Ebook de Scrapy

1 - Intro Scrapy

Criar um arquivo e faça algo nessa estruta:

```
class BlogSpider(scrapy.Spider):
   name = 'blogspider'
   start_urls = ['https://blog.scrapinghub.com']

   def parse(self, response):
       for title in response.css('.post-header>h2'):
           yield {'title': title.css('a ::text').get()}

       for next_page in response.css('a.next-posts-link'):
           yield response.follow(next_page, self.parse
```

para executar o código

```
scrapy runspider my-project.py
```

2- Iniciar um projeto

```
scrapy startproject nome-projeto
```

3 - Criar um Spider

```
scrapy genspider nome-class-spider url-project
```

4 - Executar o Scrapy modo projeto

```
scrapy crawl nome-class-spider
```

5 - Executar e salvar o arquivo

```
scrapy crawl nome-class-spider -o arquivo.formato
```

6 - Comandos úteis

verificar o conteúdo html do elemento seletor

```
response.get()
response.extract()
response.extract_first()
```

Verificar o texto de uma pesquisa xpath

```
response.xpath('tag/.text()').extract() ou .extract_first()
```

Xpath contains

```
response.xpath('//tag[contains(@attr, '')]')
response.xpath('//tag[contains(text(), 'ipsum')]')
```

Scrapy shell - aceitar a página em pt-br

```
from scrapy import Request

req = Request('url', headers={'Accept-Language':'pt-br'})
fetch(req)
#  response ativado a partir do fetch
<p class="mume-header " id="response-ativado-a-partir-do-fetch"></p>
```

Scrapy FormResquest

```
from scrapy.http import FormRequest
def start_requests(self):
    url='http://imobiliariabelamorada.com.br/filtro/locacao/\
        apartamentos/pato-branco-pr/?busca=1'
    formdata={
        'cat1': '2.locacao',
        'cat3': '4.apartamentos',
        'cidade': '5362.pato-branco-pr',
        'valor':'',
        'cod':''
    }
    yield FormRequest(url, callback=self.parse, formdata=formdata, method='POST')
```

Alterar o robots.txt na class Spider:

```
    def start_requests(self):

        url = 'https://www.almeidaw.com.br'
        form_data = {
            'cat1': '2.locacao',
        }
        # comando que subscreve ROBOTSTXT_OBEY do settings
        meta = {'dont_obey_robotstxt':'False'}
        yield scrapy.FormRequest(url=url, callback=self.parse, formdata=form_data ,method=
```

Start requests completo:

```
    def start_requests(self):

        url = 'https://www.jlo.com/todos/todos/1'

        # headers disponíveis no network
        headers = {
            'Accept': 'text/html,aation/signed-exchange;v=b3;q=9',
            'Accept-Encoding': 'gzip, deflate, br',
            'Accept-Language': 'pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7',
            'Cache-Control': 'max-age=0',
            'Connection': 'keep-alive',
            'Content-Length': '77',
            'Content-Type': 'application/x-www-form-urlencoded',
            'Cookie': '__utmc=226286; __utmz=226LNAD',
            'Host': 'www.jlosso.com.br',
            'Origin': 'https://www.jlosso.com.br',
            'Referer': 'https://www.jlosso.com.br/home',
            'Sec-Fetch-Dest': 'document',
            'Sec-Fetch-Mode': 'navigate',
            'Sec-Fetch-Site': 'same-origin',
            'Sec-Fetch-User': '?1',
            'Upgrade-Insecure-Requests': '1'
        }

        # form de request
        form_data = {
            'cat1': '2.locacao',
            'cat3': '4.carros',
            'valormedio': '',
            'codigo': '',
            'cidade': '5166.pato'
        }

        # desativa o robots.txt
        meta = {'dont_obey_robotstxt':'False'}

        yield scrapy.FormRequest(
                        url=url,
                        callback=self.parse,
                        formdata=form_data,
                        method='POST',
                        headers= headers,
                        meta= meta
                    )
```

FormResquest Shell

exemplo básico de funcionamento:

```
scrapy shell
```

```
url='http://imobiliariabelamorada.com.br/filtro/locacao/ \
    apartamentos/pato-branco-pr/?busca=1'
```

Passando form

```
formdata={
    'cat1': '2.locacao',
    'cat3': '4.apartamentos',
    'cidade': '5362.pato-branco-pr',
    'valor':'',
    'cod':''
    }
```

```
fetch(scrapy.FormRequest(url, formdata=formdata, method='POST'))
```

Passando headers no shell

```
scrapy shell -s USER_AGENT='custom user agent' 'http://www.example.com'
```

ou dentro do ipython

```
url = 'http://www.example.com'
request = scrapy.Request(url, headers={'User-Agent': 'Mybot'})
fetch(request)
```

Desativando robots.txt pelo terminal

Shell

```
scrapy shell -s ROBOTSTXT_OBEY='False' 'http://www.example.com'
```

Crawl

```
scrapy crawl name-spider -s ROBOTSTXT_OBEY='False'
```

Executando scrapy em html localhost

Shell

```
    scrapy shell local/name/page
```

Server local

1. Abra o terminal na pasta onde consta o arquivo .html

   ```
       python -m http.server <port>
   ```

   obs.: A porta default é a 8000, porém, você pode alterar inserindo outro número valor
2. Vá até o local host e clique no arquivo.html
3. Executando scrapy localhost

   ```
       scrapy shell http://0.0.0.0:8000/arquivo.html
   ```

   obs.: No caso de genspider

   ```
       scrapy genspider name localhost
   ```

DELAY no dowload das páginas

Configurando o arquivo settings.py

```
    # See also autothrottle settings and docs
    DOWNLOAD_DELAY = 3
```

ou passando no sheel

```
    scrapy crawl name-spider -s DOWNLOAD_DELAY = Valor
```

Por padrão, o Scrapy não espera um período fixo de tempo entre solicitações,
mas usa um intervalo aleatório entre 0,5 * DOWNLOAD_DELAY e 1,5 * DOWNLOAD_DELAY.

Passar argumento category

```
    scrapy crawl projeto -a category=nome_elemento
```

Ajustar o start_request

```
Nesse método é onde faz o callback para o parse, caso seja passado algum argumento,
é necessário fazer a verificação e/ou alteração do resquet.
```

Scrapy command line - List spiders

```
scrapy list
```

Lista de pseudo-classes

Irmão anterior

```
preceding-sibling::tag
```

Irmão posterior

xpath

```
following-sibling::tag
```

css

```
i.ga-bedrooms-02 + span
```

Pai

```
parent::tag
```

Pegar tag filha em css

```
tag.class > child
```

ou

```
tag.class  child
```

Pegar último filho em css

```
tag:last-child
```

Obter text dentro da tag css

```
response.css('mytag::text') -> Obter texto apenas do nó selecionado.
response.css('mytag ::text') -> Obter texto do nó selecionado e seus nós filhos.
```

Pegar atributo href

```
tag::attr(href)
```

Pegar último elemento selecionado [last()]

```
response.xpath('//div[@class="row"]/text() [last()]').get()
```

Scrapy mais Splash

Instalar e configurar o splash

github scrapy-splash

Instalar

```
    pip install scrapy-splash
```

Executar docker

```
    docker run -p 8050:8050 scrapinghub/splash
```

colocar em settings do spider

```
SPLASH_URL = 'http://localhost:8050/'
DOWNLOADER_MIDDLEWARES = {
    'scrapy_splash.SplashCookiesMiddleware': 723,
    'scrapy_splash.SplashMiddleware': 725,
    'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware': 810,
}
SPIDER_MIDDLEWARES = {
    'scrapy_splash.SplashDeduplicateArgsMiddleware': 100,
}
DUPEFILTER_CLASS = 'scrapy_splash.SplashAwareDupeFilter'
HTTPCACHE_STORAGE = 'scrapy_splash.SplashAwareFSCacheStorage'
```

Exemplo de request com splash

```
from scrapy_splash import SplashRequest

def start_requests(self):
   for url in self.start_urls:
       yield SplashRequest(
           url= url,
           callback= self.parse,
           endpoint='render.html',
           args={'wait': 10}
       )
```

Scrapy mais Selenium

Instalar e configurar o splash

github scrapy-selenium

Instalar

```
    pip install scrapy-selenium
```

colocar em settings do spider

```
from shutil import which

SELENIUM_DRIVER_NAME = 'firefox'
SELENIUM_DRIVER_EXECUTABLE_PATH = which('geckodriver')
SELENIUM_DRIVER_ARGUMENTS=['-headless']  # '--headless' if using chrome instead of firefox
SELENIUM_BROWSER_EXECUTABLE_PATH = which('firefox')

DOWNLOADER_MIDDLEWARES = {
    'scrapy_selenium.SeleniumMiddleware': 800
}
```

Exemplo de request com splash

```
from scrapy_selenium import SeleniumRequest


   def start_requests(self):
       url = self.start_urls[0]
       yield SeleniumRequest(url=url, callback=self.parse, wait_time= 3)
```

Interpretando robots.txt

Guia robots.txt

Básico

```
User-agent: * [para todo bot]
Disallow: / [bloquear o site inteiro]

User-agent: * [para todo bot]
Disallow: /dir/ [bloquear diretório específico]

User-agent: Fetch [para o bot fetch]
Disallow: /private.html [bloquear página específica]
```