

Web Crawlers para Gente Grande com Python e Scrapy

Gileno Alves Santa Cruz Filho

Gileno, quem?



**O que são Web
Crawlers?**

Crawler X Scrapping

Scraping

- Extrair Dados Estruturados de algo não estruturado
 - HTML
 - XML
 - IMAGENS (OCR)
 - PDF
-

Crawler

- Visitar Páginas Web
- Seguir regras de visitaçã
- Não necessariamente precisa extrair informação estruturada (fazer scraping)

Porque criá-los?



Informação == dinheiro

Google

E-Commerce

- Qual o valor que meus concorrentes estão cobrando?
- Quais produtos estão sendo vendidos na loja X?
- Qual a variação de preço?
- Produto está disponível?

Imóveis

- Qual preço praticado na região X?
- O apartamento X está abaixo do preço praticado no mercado?
- Que imóveis estão disponíveis na cidade Y?

Cotação Dolar



Dólar Hoje

US\$ 1,00



R\$ 3,92

DÓLAR COMERCIAL



 **SELECIONE A CIDADE...** ▼

DÓLAR TURISMO ⓘ

Compare as cotações em sua cidade

Código Fonte

```
91  
92 <input type="hidden" id="id-moeda" value="8" />  
93 <input type="hidden" id="toggle" value="0" />  
94 <input type="hidden" value="3,92" id="taxa-comercial">  
95 <input type="hidden" id="taxa-turismo" value="0" />  
96 <input type="hidden" id="input-alterado" value="" />  
97 <input type="hidden" id="alerta" value="0" />  
98  
99 <input type="hidden" id="ss" value="0" />  
100
```

urllib

```
import urllib.request
```

```
import re
```

```
url = 'https://www.melhorcambio.com/dolar-hoje'
```

```
headers = {
```

```
    'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,  
like Gecko) Chrome/63.0.3239.132 Safari/5'
```

```
}
```

```
req = urllib.request.Request(url, headers=headers)
```

```
with urllib.request.urlopen(r) as response:
```

```
    html = response.read().decode('utf-8')
```

```
preco = re.findall(r'<input type="hidden" value="(.)" id="taxa-comercial">', html)[0]
```

```
print(preco)
```

Usando Requests

```
import requests
```

```
import re
```

```
url = 'https://www.melhorcambio.com/dolar-hoje'
```

```
headers = {
```

```
'User-Agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like  
Gecko) Chrome/63.0.3239.132 Safari/5'
```

```
}
```

```
req = requests.get(url, headers=headers)
```

```
html = req.text
```

```
preco = re.findall(r'<input type="hidden" value="(.)" id="taxa-comercial">', html)[0]
```

```
print(preco)
```


Scrapy



Spiders

```
<table data-apvhdrId="512">
  <tbody>
    <tr data-id="1">
      <td data-itemid="001">Beer</td>
      <td data-quantity="10">10 bottles</td>
      <td data-unitcost="11.00">11.00</td>
      <td data-amount="110.00">110.00</td>
    </tr>
    <tr data-id="2">
      <td data-itemid="002">Vodka</td>
      <td data-quantity="20">20 bottles</td>
      <td data-unitcost="100.00">100.00</td>
      <td data-amount="2000.00">2000.00</td>
    </tr>
  </tbody>
</table>
```

```
[
  {"apvhdrId":512, "id":1, "itemid":001, "quantity":10, "unitcost":"10.00", "amount":"110.00"},
  {"apvhdrId":512, "id":2, "itemid":001, "quantity":20, "unitcost":"100.00", "amount":"2000.00"}
]
```

Items (html -> dados estruturados)

Usando Scrapy (dolar_hoje.py)

```
import scrapy
```

```
import re
```

```
class DolarSpider(scrapy.Spider):
```

```
    name = 'dolar_hoje'
```

```
    start_urls = ['https://www.melhorcambio.com/dolar-hoje']
```

```
    custom_settings = {
```

```
        'USER_AGENT': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
```

```
        Chrome/63.0.3239.132 Safari/5'
```

```
    }
```

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

```
        html = response.text
```

```
        preco = re.findall(
```

```
            r'<input type="hidden" value="(.)" id="taxa-comercial">', html
```

```
        )[0]
```

```
        self.log(preco)
```

Seletor

xpath / css

XPath

- Linguagem de consulta para arquivos XML
- Uma espécie de SQL para arquivos XML (pode ser usado em HTML)

Usando xpath e css

```
import scrapy
```

```
import re
```

```
class DolarSpider(scrapy.Spider):
```

```
    name = 'dolar_hoje'
```

```
    start_urls = ['https://www.melhorcambio.com/dolar-hoje']
```

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

```
        preco = response.xpath('//input[@id="taxa-comercial"]/@value')
```

```
        self.log(preco.extract_first())
```

```
        preco = response.css('#taxa-comercial')[0]
```

```
        self.log(preco.attrib['value'])
```

Items

Retornando Items

```
import scrapy
```

```
class VivaRealSpider(scrapy.Spider):
```

```
    name = 'vivareal'
```

```
    start_urls =
```

```
    ['https://www.vivareal.com.br/venda/pernambuco/recife/bairros/boa-viagem/apartamento_residencial/']
```

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

```
        for item in response.xpath("//div[contains(@class, 'results-list')]/div"):
```

```
            yield {
```

```
                'title': item.xpath("./h2/a/text()").extract_first().strip()
```

```
            }
```

Scrapy Project

scrapy startproject tdc

```
└─ tdc
  └─ tdc
    > __pycache__
    └─ spiders
      > __pycache__
      ├── __init__.py
      ├── __init__.py
      ├── items.py
      ├── middlewares.py
      ├── pipelines.py
      ├── settings.py
      └── scrapy.cfg
```

Item Pipeline

pipelines.py

```
class TdcPipeline(object):  
  
    def process_item(self, item, spider):  
        # Faz alguma limpeza  
        # Salva no banco de dados  
        return item
```

.... settings.py

```
ITEM_PIPELINES = {  
    tdc.pipelines.TdcPipeline: 300,  
}
```

yield Request

Criando novas requisições

```
import scrapy
```

```
class VivaRealSpider(scrapy.Spider):
```

```
    name = 'vivareal'
```

```
    start_urls =
```

```
['https://www.vivareal.com.br/venda/peernambuco/recife/bairros/boa-viagem/apartamento_residencial/']
```

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

```
        for item in response.xpath("//div[contains(@class, 'results-list')]/div"):
```

```
            href = item.xpath("./h2/a/@href").extract_first()
```

```
            yield scrapy.Request(
                f'https://www.vivareal.com.br{href}', self.parse_detail
            )
```

```
    def parse_detail(self, response):
```

```
        yield {
            'title': response.xpath("//title/text()").extract_first().strip()
        }
```

CrawlSpider

Criando um CrawlSpider

```
import scrapy
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
class VivarealSpider(CrawlSpider):
    name = 'vivareal_crawl'
    start_urls = ['https://www.vivareal.com.br/venda/peernambuco/recife/bairros/boa-viagem/apartamento_residencial/']
    rules = (
        Rule(
            LinkExtractor(allow='/venda/peernambuco/recife/bairros/boa-viagem/apartamento_residencial/')
        ),
        Rule(
            LinkExtractor(
                allow='/imovel/',
            ), callback='parse_imovel'
        )
    )
    def parse_imovel(self, response):
        yield {
            'title': response.xpath("//title/text()").extract_first()
        }
```

Regras da CrawlSpider

```
rules = (  
    Rule(  
        LinkExtractor(allow='/venda/pernambuco/recife/bairros/boa-viagem/apartamento_residencial/')  
    ),  
    Rule(  
        LinkExtractor(  
            allow='/imovel/',  
        ), callback='parse_imovel'  
    )  
)
```

Plugins e Settings

settings.py

```
SPIDER_MODULES = [tdc.spiders']
```

```
NEWSPIDER_MODULE = 'tdc.spiders'
```

```
USER_AGENT = 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/63.0.3239.132 Safari/5'
```

```
# Obey robots.txt rules
```

```
ROBOTSTXT_OBEY = True
```

```
# Configure maximum concurrent requests performed by Scrapy (default: 16)
```

```
#CONCURRENT_REQUESTS = 32
```

```
DOWNLOAD_DELAY = 0.5
```

plugins

- <https://github.com/scrapy-plugins>
- Scrapy Splash

Deploy

Algumas opções

- Scrapy
- ScrapingHub
- SpiderKeeper

**Talk is cheap,
Show me the
code**

Obrigado!

Dúvidas?

@gilenofilho

contato@gilenofilho.com.br

<https://www.pycursos.com>

<https://github.com/gileno/tdcrecife2019>