

Web Crawlers com Scrapy



Gileno Alves Santa Cruz Filho

Gileno, quem?



**Centro de
Informática**
UFPE



PyCursos



Python User Group Pernambuco

**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 visitaço
- 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



DÓLAR COMERCIAL

R\$

4,85



DÓLAR TURISMO ⓘ



SELECIONE A CIDADE...



✓ Compare as cotações do **Dólar** em sua cidade ✓

Quero comprar

Quero vender

Remessa online >>

Código Fonte

```
348 <input type="hidden" id="nome-moeda" value="Dólar" />
349 <input type="hidden" id="artigo" value="do" />
350 <input type="hidden" id="id-moeda" value="8" />
351 <input type="hidden" id="toggle" value="0" />
352 <input type="hidden" value="4,85" id="taxa-comercial">
353 <input type="hidden" id="taxa-turismo" value="0" />
354 <input type="hidden" id="input-alterado" value="" />
```

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-apvhidrid="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>

```

```

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

```

Items (html -> dados estruturados)

Usando Scrapy

```
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 OlxSpider(scrapy.Spider):
```

```
    name = "olx"
```

```
    start_urls = ["https://pe.olx.com.br/imoveis/aluguel"]
```

```
    custom_settings = {
```

```
        "USER_AGENT": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like  
        Gecko) Chrome/99.0.4844.82 Safari/537.36"
```

```
    }
```

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

```
        items = response.xpath('//*[id="ad-list"]/li')
```

```
        for item in items:
```

```
            yield {
```

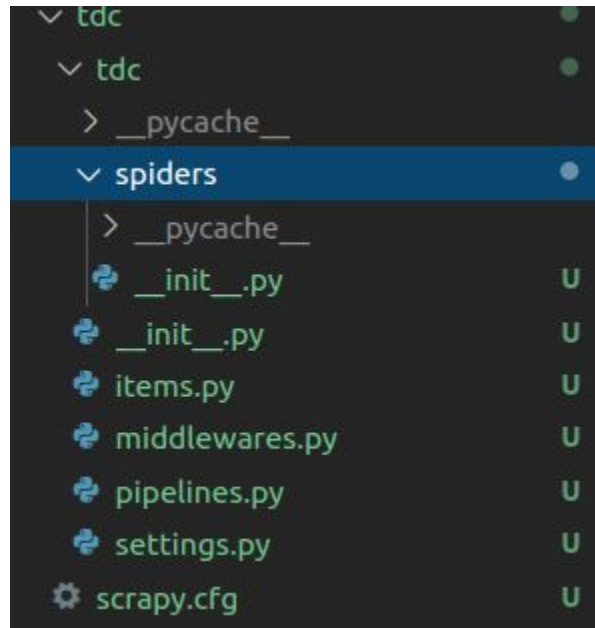
```
                "url": item.xpath("./a/@href").extract_first(),
```

```
                "titulo": item.xpath("./h2/text()").extract_first(),
```

```
            }
```

Scrapy Project

scrapy startproject tdc



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 OlxSpider(scrapy.Spider):
    name = 'olx'
    start_urls = ['https://pe.olx.com.br/imoveis/aluguel']
    def parse(self, response):
        items = response.xpath('//*[@id="ad-list"]/li')
        for item in items:
            url = item.xpath("./a/@href").extract_first()
            if url:
                yield scrapy.Request(url=url, callback=self.parse_detail)
    def parse_detail(self, response):
        preco = response.xpath("//h2[@font-weight='400']/text()").extract_first()
        yield {
            'url': response.url,
            'titulo': response.xpath("//title/text()").extract_first(),
            'preco': preco,
        }
```

Plugins e Settings

settings.py

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

```
USER_AGENT = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)  
Chrome/99.0.4844.82 Safari/537.36'
```

```
# 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

Scrapinghub -> Zyte

The screenshot displays the Zyte dashboard interface. On the left is a dark blue sidebar with a hamburger menu icon and the 'zyte' logo. Below the logo are navigation links: 'Dashboard' (with a home icon), 'TOOLS' (with a list icon), 'Smart Proxy Manager' (with a snowflake icon), 'Automatic Extraction' (with a magnifying glass icon), 'Zyte Data API' (with a monitor icon), 'Scrapy Cloud' (with a cloud icon and a dropdown arrow), 'Splash' (with a gear icon), 'Settings' (with a settings icon), 'Documentation' (with a document icon), 'Help center' (with a question mark icon), 'Contact support' (with a speech bubble icon), and 'System status' (with a status icon). The main content area has a dark blue header with a search bar, a 'Welcome back, Gileno Filho!' message, and a user profile icon. Below the header, the breadcrumb 'gileno / Dashboard' is shown. The main section is titled 'Scrapy Cloud Projects' with a cloud icon and a notification badge '1'. A 'Create Project' button is in the top right. A list of projects is shown, with 'crawler-escolas' visible. Below this is a 'Tools' section with five cards: 'Smart Proxy Manager' (described as setting spiders free from captchas), 'Automatic Extraction' (described as extracting data automatically at scale), 'Zyte Data API' (described as a smart browser API for solving captchas), 'Scrapy Cloud' (described as a platform to deploy and scale web spiders), and 'Splash' (described as a lightweight, scriptable headless browser). Each card includes a 'Try it free for 14 days' link, except for 'Scrapy Cloud' which has a 'Create your first project' link.

zyte

Search

Welcome back, Gileno Filho!

gileno / Dashboard

Scrapy Cloud Projects 1

Create Project

crawler-escolas

Tools

- Smart Proxy Manager**
Set your spiders free. No captchas, just successful requests.
[Try it free for 14 days](#)
- Automatic Extraction**
Extract data automatically at scale from any website.
[Try it free for 14 days](#)
- Zyte Data API**
Smart browser API that solves complex banning and blocking.
[Try it free for 14 days](#)
- Scrapy Cloud**
Deploy, run and scale your web spiders.
[Create your first project](#)
- Splash**
Lightweight, scriptable headless browser with an HTTP API

Spider Keeper

SpiderKeeper

Projects

demo

JOBS

Dashboard

Periodic Jobs

SPIDERS

Dashboard

Deploy

PROJECT

Running Stats

SERVICE

Usage Stats

Job Dashboard

Run

Next Jobs

#	Job	Spider	Args	Priority	Wait
637	12	demo	foo=bar	HIGH	0 h 0 m

Running Jobs

#	Job	Spider	Args	Priority	Runtime	Started	Log	Running On	Action
634	12	demo	foo=bar	HIGH	0 h 0 m	2017-04-11 14:10:04	Log	localhost:6800	Stop

Completed Jobs

#	Job	Spider	Args	Priority	Runtime	Started	Log	Status
454	11	demo		NORMAL	0 h 0 m	2017-04-10 23:49:29	Log	FINISHED

SpiderKeeper.

Version 1.0.0

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

Obrigado! Dúvidas?



@gilenofilho

contato@gilenofilho.com.br

<https://www.pycursos.com>

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