



# VirtualEnv / Scrapy

**VirtualEnv** é uma ferramenta (container) para criar ambientes Python isolados. Neste caso será utilizado para conseguir “rodar” o Scrapy nele.

Instalação:

```
pip install virtualenv
```



Para criar um container no VirtualEnv execute o comando:

```
Virtualenv nome
```

Será criada uma pasta com o nome que você digitou.

Execute então o seguinte comando:

```
source /nome/bin/activate
```



Para instalar o Scrapy execute o seguinte comando:

```
pip install virtualenv
```

Para iniciar um novo projeto Scrapy execute o comando a seguir:

```
scrapy startproject nomedoprojeto
```

Vá para a pasta do projeto e execute o comando para iniciar um novo *spider*:

scrapy genspider nome dominio.com


```
class ListaSpider(scrapy.Spider):
    name = 'Lista'
    allowed_domains = ['listanainternet.com.br']
    start_urls = ['http://www.listanainternet.com.br/inicio']

    def parse(self, response):
        for item in response.css("html body div.main div.cat div.row-fluid ul.span3 li.btn-cat0"):
            link = item.css("a::attr(href)").extract_first()
            yield response.follow(link, self.parse_categoria)

    def parse_categoria(self, response):
        for link in response.css("html body div.main div.container div.row-fluid div.span8 div.item div.item-inner a::attr(href)").extract():
            yield response.follow(link, self.parse_empresas)

    def parse_empresas(self, response):
        for link in response.css("html body div.main div.container div.row-fluid div.span8 div.item.item_ad div.item-inner a::attr(href)").extract():
            yield response.follow(link, self.parse_empresa)

    def parse_empresa(self, response):
        nome = response.css('html body div.main div.row-fluid div.span8.content div.title.blue-lis font::text').extract_first()
        endereco = response.css("html body div.main div.row-fluid div.span8.content div.contacts.row-fluid div.span7.address p::text").extract_first()
        telefone = response.css("html body div.main div.row-fluid div.span8.content div.contacts.row-fluid div.span5.phone span::text").extract_first()
        telefone = re.sub(r'\s', '', telefone)
        empresa = AgendaEnarItem(nome = nome, endereco = endereco, telefone = telefone)
        yield empresa
```



Será gerado um arquivo *spider* na pasta *spiders*, esse é um exemplo de *spider* que navega pelo site com a função:

```
response.follow(link_coletado_no_site,  
funcao_para_pegar_html_da_nova_pagina)
```

No arquivo `Items.py` você coloca o objeto da empresa, esse arquivo fica como a seguir:

```
import scrapy

class AgendaEnariItem(scrapy.Item):
    nome = scrapy.Field()
    endereco = scrapy.Field()
    telefone = scrapy.Field()
```

Quando o *spider* é iniciado, entra o `pipelines.py`, ele tem função para iniciar elementos junto com o *spider*. Segue exemplo do *pipeline* para salvar no MongoDB:

```
import pymongo
from scrapy.conf import settings

import logging

class AgendaEnariPipeline(object):
    collection_name = 'empresas'


    def __init__(self, mongo_uri, mongo_db):
        self.mongo_uri = mongo_uri
        self.mongo_db = mongo_db

    @classmethod
    def from_crawler(cls, crawler):
        return cls(
            mongo_uri=crawler.settings.get('MONGO_URI'),
            mongo_db=crawler.settings.get('MONGO_DATABASE', 'items')
        )

    def open_spider(self, spider):
        self.client = pymongo.MongoClient(self.mongo_uri)
        self.db = self.client[self.mongo_db]

    def close_spider(self, spider):
        self.client.close()

    def process_item(self, item, spider):
        self.db[self.collection_name].insert_one(dict(item))
        return item
```

- 
- No `open_spider` inicia-se o banco.
  - No `close_spider`, é fechada a conexão.
  - No `process_item`, é salvo o item no banco e é usado para remover o “lixo” do item.