

Scrapy

Trabalhando com *Scrapy* – Parte 2

Scrapy

Extraindo dados com *Scrapy Shell*

Vamos usar o *Scrapy Shell* para extrair dados de <http://quotes.toscrape.com> para entender melhor os conceitos antes de aplicá-los em nosso *spider*.

Execute a linha a seguir no Linux:

```
$ scrapy shell 'http://quotes.toscrape.com/page/1/'
```

Ou, no Windows, usando aspas duplas:

```
C:\> scrapy shell "http://quotes.toscrape.com/page/1/"
```

Scrapy

Você verá algo como isso:

```
evaldowolkers@evaldo ~  
File Edit View Search Terminal Help  
[]  
2017-12-03 01:28:14 [scrapy.extensions.telnet] DEBUG: Telnet console listening on 127.0.0.1:6023  
2017-12-03 01:28:14 [scrapy.core.engine] INFO: Spider opened  
2017-12-03 01:28:15 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/1/> (referer: None)  
[s] Available Scrapy objects:  
[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)  
[s] crawler <scrapy.crawler.Crawler object at 0x7f74a51ceda0>  
[s] item {}  
[s] request <GET http://quotes.toscrape.com/page/1/>  
[s] response <200 http://quotes.toscrape.com/page/1/>  
[s] settings <scrapy.settings.Settings object at 0x7f74a384bac8>  
[s] spider <DefaultSpider 'default' at 0x7f74a33ad278>  
[s] Useful shortcuts:  
[s] fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)  
[s] fetch(req) Fetch a scrapy.Request and update local objects  
[s] shelp() Shell help (print this help)  
[s] view(response) View response in a browser  
>>>
```

```
Command Prompt - scrapy shell "http://quotes.toscrape.com/page/1/"  
2017-12-19 19:01:49 [scrapy.core.engine] INFO: Spider opened  
2017-12-19 19:01:50 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://quotes.toscrape.com/page/1/> (referer: None)  
[s] Available Scrapy objects:  
[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)  
[s] crawler <scrapy.crawler.Crawler object at 0x00000216E82067B8>  
[s] item {}  
[s] request <GET http://quotes.toscrape.com/page/1/>  
[s] response <200 http://quotes.toscrape.com/page/1/>  
[s] settings <scrapy.settings.Settings object at 0x00000216E93E2A58>  
[s] spider <DefaultSpider 'default' at 0x216e966b9e8>  
[s] Useful shortcuts:  
[s] fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirects are followed)  
[s] fetch(req) Fetch a scrapy.Request and update local objects  
[s] shelp() Shell help (print this help)  
[s] view(response) View response in a browser
```

Scrapy

Usando o *shell* você pode tentar selecionar elementos usando CSS com o objeto de resposta:

```
[s] response <200 http://quotes.toscrape.com/page/1/>
[s] settings <scrapy.settings.Settings object at 0x7f74a384bac8>
[s] spider <DefaultSpider 'default' at 0x7f74a33ad278>
[s] Useful shortcuts:
[s] fetch(url[, redirect=True]) Fetch URL and update local objects (by default, redirec
[s] fetch(req) Fetch a scrapy.Request and update local objects
[s] shelp() Shell help (print this help)
[s] view(response) View response in a browser
>>> response.css('title')
[<Selector xpath='descendant-or-self::title' data='<title>Quotes to Scrape</title>'>]
>>>
```

O resultado da execução de ***response.css('title')*** é um objeto semelhante a uma lista chamado *SelectorList*, que representa uma lista de objetos *Selector* que envolve elementos *XML/HTML* e permitem que você execute mais consultas para afinar a seleção ou extração de dados.

Scrapy

Para extrair o texto do título, você pode usar desta forma:

```
>>> response.css('title::text').extract()  
['Quotes to Scrape']
```

Temos duas coisas para observar aqui: Primeiro é que adicionamos `::text` à consulta CSS para informar que queremos selecionar apenas os elementos de texto diretamente dentro do elemento `<title>`, caso não seja especificado `::text`, teríamos obtido o elemento de título completo, incluindo as tags:

```
>>> response.css('title').extract()  
['<title>Quotes to Scrape</title>']
```

A segunda coisa a observar é que o resultado da chamada `.extract()` é uma lista, porque estamos lidando com um instância de `SelectorList`, sendo assim, para pegar o primeiro resultado usamos:

```
>>> response.css('title::text').extract_first()  
'Quotes to Scrape'
```

Scrapy

Em vez de usar o `extract_first()` você também pode fazer assim:

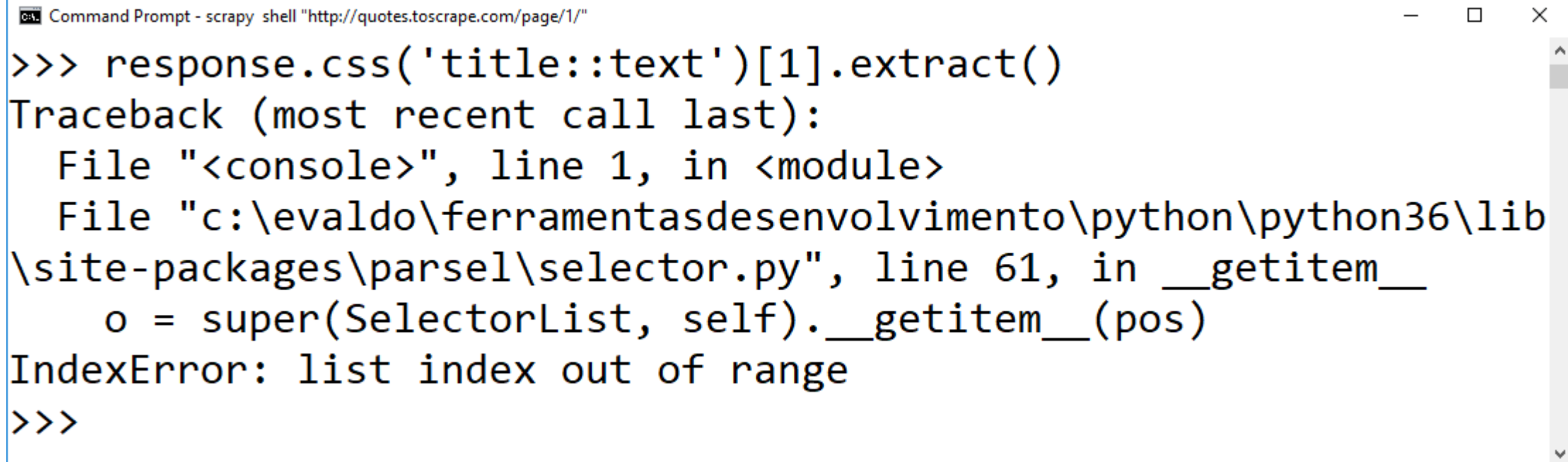
```
>>> response.css('title::text')[0].extract()  
'Quotes to Scrape'
```

No entanto, usando `extract_first()` evitamos um *IndexError* porque será retornando *None* quando não encontrar nenhum elemento que corresponda à seleção.

Desta forma teremos um código que evitará erros em caso de não encontrarmos o que estamos procurando.

Scrapy

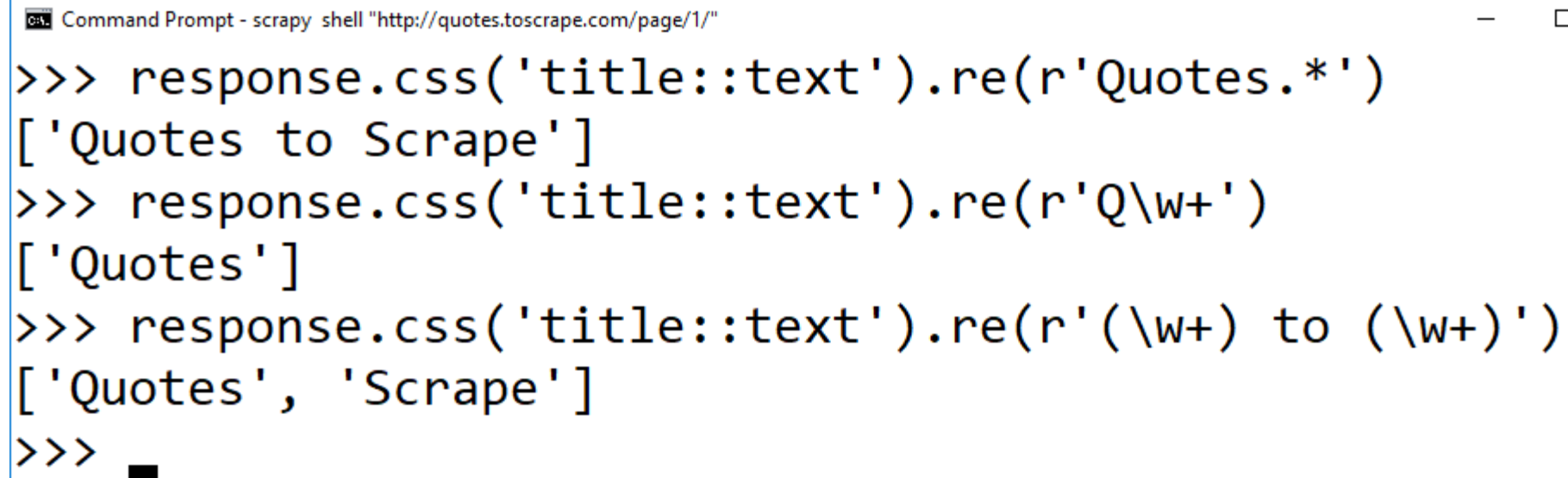
Veja o erro de índice ao tentar retornar o índice [1]:



```
Command Prompt - scrapy shell "http://quotes.toscrape.com/page/1/"
>>> response.css('title::text')[1].extract()
Traceback (most recent call last):
  File "<console>", line 1, in <module>
  File "c:\evaldo\ferramentasdesenvolvimento\python\python36\lib\site-packages\parsel\selector.py", line 61, in __getitem__
    o = super(SelectorList, self).__getitem__(pos)
IndexError: list index out of range
>>>
```

Scrapy

Além dos métodos *extract()* e *extract_first()*, também podemos usar o método *re()* para extrair dados usando expressões regulares (Neste curso tem uma aula sobre expressões regulares).



```
Command Prompt - scrapy shell "http://quotes.toscrape.com/page/1/"

>>> response.css('title::text').re(r'Quotes.*')
['Quotes to Scrape']
>>> response.css('title::text').re(r'Q\w+')
['Quotes']
>>> response.css('title::text').re(r'(\w+) to (\w+)')
['Quotes', 'Scrape']
>>> █
```


Scrapy

XPath

Além do CSS, os seletores *Scrapy* também suportam expressões *XPath*.

O *XPath* é o resultado de um esforço para fornecer uma sintaxe e semântica comuns para a funcionalidade compartilhada entre *XSL Transformations* e *XPointer*. O objetivo principal do *XPath* é referenciar partes de um documento *XML*. O *XPath* fornece facilidades para manipulação de *strings*, números e booleanos. *XPath* utiliza uma sintaxe compacta não-*XML* para facilitar o uso de *XPath* com *URLs* e valores de atributos *XML*.

XPath obteve seu nome porque usa uma notação de caminho como em *URLs* para navegar pela estrutura hierárquica de um documento *XML*.

Scrapy

XPath

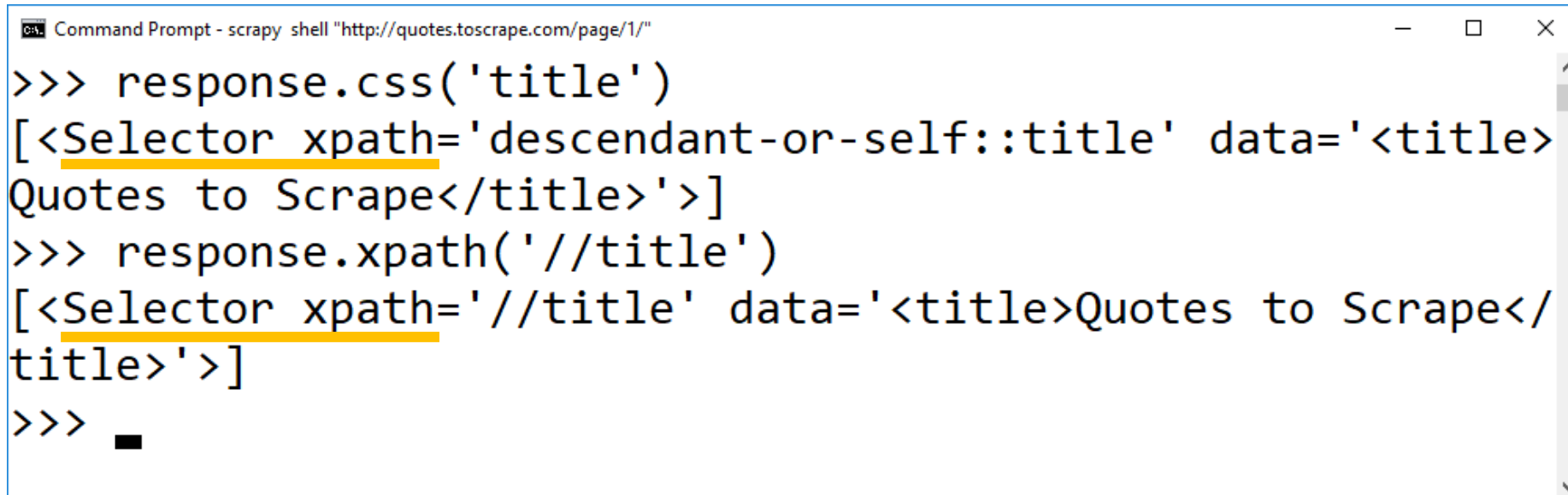
Veja sua utilização no *Scrapy Shell*.

```
Command Prompt - scrapy shell "http://quotes.toscrape.com/page/1/"
>>>
>>> response.xpath('//title')
[<Selector xpath="//title" data='<title>Quotes to Scrape
</title>'>]
>>> response.xpath('//title/text()').extract_first()
'Quotes to Scrape'
>>>
```

Scrapy

XPath

As expressões *XPath* são muito poderosas e são a base do *Scrapy Selectors*. Na verdade, os seletores CSS são convertidos para *XPath*, podemos ver isso na representação dos objetos seletores no *shell*.



```
Command Prompt - scrapy shell "http://quotes.toscrape.com/page/1/"
>>> response.css('title')
[<Selector xpath='descendant-or-self::title' data='<title>
Quotes to Scrape</title>'>]
>>> response.xpath('//title')
[<Selector xpath='//title' data='<title>Quotes to Scrape</
title>'>]
>>> █
```

Scrapy

Extraindo citações e autores

Vamos agora ver como extrair citações e autores do site.

Cada citação em <http://quotes.toscrape.com> é representada por elementos HTML que se parecem com isto:

```
<div class="quote">
  <span class="text">"The world as we have created it is a process of our
  thinking. It cannot be changed without changing our thinking."</span>
  <span>
    by <small class="author">Albert Einstein</small>
    <a href="/author/Albert-Einstein">(about)</a>
  </span>
  <div class="tags">
    Tags:
    <a class="tag" href="/tag/change/page/1/">change</a>
    <a class="tag" href="/tag/deep-thoughts/page/1/">deep-thoughts</a>
    <a class="tag" href="/tag/thinking/page/1/">thinking</a>
    <a class="tag" href="/tag/world/page/1/">world</a>
  </div>
</div>
```

Scrapy

Extraindo citações e autores

```
1  <div class="quote"> -> div.quote
2  <span class="text"> -> span.text
3  <small class="author"> -> small.author
4  <div class="tags"> -> div.tags
5  <a class="tag" href="/tag/change/page/1/"> -> a.tag
6
7
8  <span class="text">"The world as we have created it is a process of our
9    thinking. It cannot be changed without changing our thinking."</span>
10 -> span.text::text = "The world...thinking"
11
12 by <small class="author">Albert Einstein</small>
13 -> small.author::text = Albert Einstein
14
15 <a class="tag" href="/tag/change/page/1/">change</a>
16 -> a.tag::text = change
```

Scrapy

Extraindo citações e autores

Vamos executar o Scrapy Shell para extrairmos alguns dados.

C:\> scrapy shell "http://quotes.toscrape.com"

No Linux use aspas simples:

\$ scrapy shell 'http://quotes.toscrape.com'

Scrapy

Extraindo citações e autores

Recebemos uma lista de seletores para os elementos HTML com:

```
>>> response.css("div.quote")
```

```
Command Prompt - scrapy shell "http://quotes.toscrape.com"
ope itemtype="h">, <Selector xpath="descendant-or-self::div[@class and contains
(concat(' ', normalize-space(@class), ' '), ' quote ')]" data='<div class="quot
e" itemscope itemtype="h">, <Selector xpath="descendant-or-self::div[@class and
contains(concat(' ', normalize-space(@class), ' '), ' quote ')]" data='<div cl
ass="quote" itemscope itemtype="h">, <Selector xpath="descendant-or-self::div[@
class and contains(concat(' ', normalize-space(@class), ' '), ' quote ')]" data
='<div class="quote" itemscope itemtype="h">, <Selector xpath="descendant-or-se
lf::div[@class and contains(concat(' ', normalize-space(@class), ' '), ' quote
')]" data='<div class="quote" itemscope itemtype="h">, <Selector xpath="descend
ant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '),
' quote ')]" data='<div class="quote" itemscope itemtype="h">, <Selector xpath
="descendant-or-self::div[@class and contains(concat(' ', normalize-space(@clas
s), ' '), ' quote ')]" data='<div class="quote" itemscope itemtype="h">, <Sele
ctor xpath="descendant-or-self::div[@class and contains(concat(' ', normalize-sp
ace(@class), ' '), ' quote ')]" data='<div class="quote" itemscope itemtype="h'
>, <Selector xpath="descendant-or-self::div[@class and contains(concat(' ', nor
malize-space(@class), ' '), ' quote ')]" data='<div class="quote" itemscope ite
mtype="h">]
>>>
```

Scrapy

Extraindo citações e autores

Cada um dos seletores retornados por nossa consulta permite executarmos novas consultas sobre seus elementos.

Vamos atribuir o primeiro seletor a uma variável, para que possamos executar nossos seletores CSS em uma citação específica:

```
>>> citacao = response.css("div.quote")[0]
```


Scrapy

Extraindo citações e autores

Agora, podemos extrair o título, autor e as *tags* da citação usando o objeto que criamos.

```
Command Prompt - scrapy shell "http://quotes.toscrape.com"
>>> texto = citacao.css("span.text::text").extract_first()
>>> texto
'“The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking.”'
>>> autor = citacao.css("small.author::text").extract_first()
>>> autor
'Albert Einstein'
>>>
```

Scrapy

Extraindo citações e autores

Como as *tags* são uma lista de *strings*, podemos usar o método *.extract()* para obter todas:

```
Command Prompt - scrapy shell "http://quotes.toscrape.com"
>>>
>>> tags = citacao.css("div.tags a.tag::text").extract()
>>> tags
['change', 'deep-thoughts', 'thinking', 'world']
>>>
>>>
>>>
>>> █
```

Scrapy

Extraindo citações e autores

Como podemos extrair cada parte individualmente, podemos iterar sobre todos os elementos das citações e colocá-los em um dicionário *Python*.

```
Command Prompt - scrapy shell "http://quotes.toscrape.com"
>>> for citacao in response.css("div.quote"):
...     texto = citacao.css("span.text::text").extract_first()
...     autor = citacao.css("small.author::text").extract_first()
...     tags = citacao.css("div.tags a.tag::text").extract()
...     print(dict(texto=texto, autor=autor, tags=tags))
...
{'texto': '"The world as we have created it is a process of our thinking. It cannot be changed without changing our thinking."', 'autor': 'Albert Einstein', 'tags': ['change', 'deep-thoughts', 'thinking', 'world']}
{'texto': '"It is our choices, Harry, that show what we truly are, far more than our abilities."', 'autor': 'J.K. Rowling', 'tags': ['abilities', 'choices']}
{'texto': '"There are only two ways to live your life. One is as though nothing is a miracle. The other is as though everything is a miracle."', 'autor': 'Albert Einstein', 'tags': ['inspirational', 'life', 'live', 'miracle', 'miracles']}
{'texto': '"The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid."', 'autor': 'Jane Austen', 'tags': ['aliteracy', 'books', 'classic', 'humor']}
{'texto': '"Imperfection is beauty, madness is genius and it's better to be abs
```

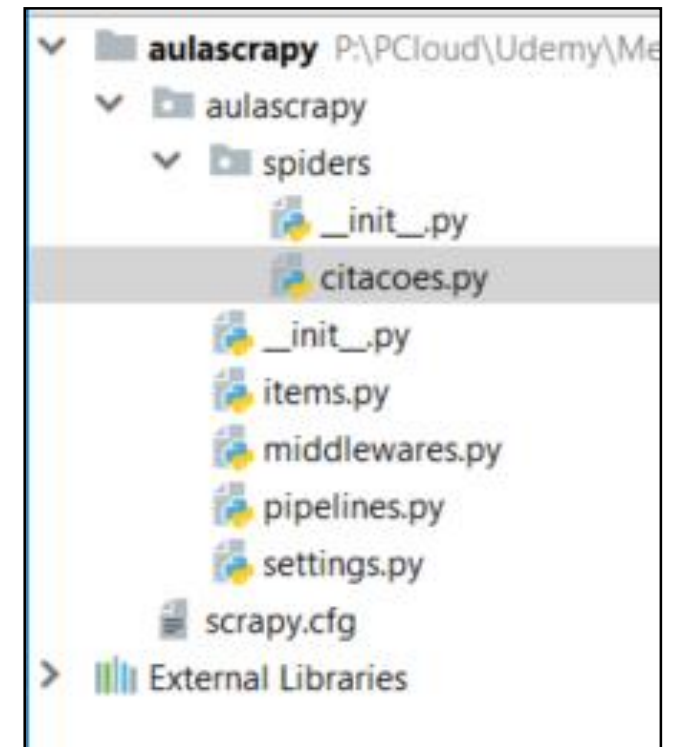
Scrapy

Exemplo

Vamos criar o projeto scrapy:

\$ scrapy startproject aulascrapy

Vamos criar o arquivo *citacoes.py* e vamos salvar na pasta *spiders*.



Scrapy

Exemplo

```
import scrapy

class CitacoesSpider(scrapy.Spider):
    name = "citacoes"
    start_urls = [
        "http://quotes.toscrape.com/page/1/",
        "http://quotes.toscrape.com/page/2/",
    ]

    def parse(self, response):
        for citacao in response.css('div.quote'):
            yield {
                'texto': citacao.css('span.text::text').extract_first(),
                'autor': citacao.css('small.author::text').extract_first(),
                'tags': citacao.css('div.tags a.tag::text').extract,
            }
```

Scrapy

Exemplo

Acesse a pasta principal do projeto (onde fica o arquivo scrapy.cfg) e execute o spider usando:

\$ scrapy crawl citacoes

```
Command Prompt
alize-space(@class), ' '), ' tags ')]/descendant-or-self::*a[@class and contains(concat(' ', normalize-space(@class), ' '), ' tag ')]/text()" data='paraphrased'>]>}
2017-12-19 21:16:11 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'texto': '"A woman is like a tea bag; you never know how strong it is until it's in hot water."', 'autor': 'Eleanor Roosevelt', 'tags': <bound method SelectorList.extract of [<Selector xpath="descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '), ' tags ')]/descendant-or-self::*a[@class and contains(concat(' ', normalize-space(@class), ' '), ' tag ')]/text()" data='misattributed-eleanor-roosevelt'>]>}
2017-12-19 21:16:11 [scrapy.core.scrapers] DEBUG: Scraped from <200 http://quotes.toscrape.com/page/1/>
{'texto': '"A day without sunshine is like, you know, night."', 'autor': 'Steve Martin', 'tags': <bound method SelectorList.extract of [<Selector xpath="descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '), ' tags ')]/descendant-or-self::*a[@class and contains(concat(' ', normalize-space(@class), ' '), ' tag ')]/text()" data='humor', <Selector xpath="descendant-or-self::div[@class and contains(concat(' ', normalize-space(@class), ' '),
```

FIM