



# Web Scraping

Universidade Federal Fluminense

Daniel dos Santos

15 de dezembro de 2018

# Conteúdo

- Introdução
- HTML
- [rvest](#)
- [RSelenium](#)
- Encerramento

- Pacotes
- O que é web scraping?

# Introdução

# Pacotes

- Essenciais
  - [rvest](#)
  - [RSelenium](#)
- Outros
  - [stringr](#)
  - [dplyr](#)

A documentação dos pacotes encontra-se na pasta da apresentação.

# O que é web scraping ?

web scraping é uma técnica de coleta online que por meio de processos automatizados realiza uma *raspagem* (scraping) de uma página da web, estruturando informações contidas no site.


	name	height	mass	hair_color	skin_color
1	Luke Skywalker	172	77.0	blond	fair
2	C-3PO	167	75.0	NA	gold
3	R2-D2	96	32.0	NA	white, blue
4	Darth Vader	202	136.0	none	white
5	Leia Organa	150	49.0	brown	light
6	Owen Lars	178	120.0	brown, grey	light
7	Beru Whitesun lars	165	75.0	brown	light
8	R5-D4	97	32.0	NA	white, red
9	Biggs Darklighter	183	84.0	black	light
10	Obi-Wan Kenobi	182	77.0	auburn, white	fair

Dados estruturados

FULL CAST AND CREW | TRIVIA | USER REVIEWS | IMDbPro | MORE | SHARE

**O Castelo Animado** (2004) ★ 8.2 271.807 [Rate This](#)

Hauru no ugoku shiro (original title)  
Livre | 1h 59min | Animation, Adventure, Family | 5 August 2005 (Brazil)



When an unconfident young woman is cursed with an old body by a spiteful witch, her only chance of breaking the spell lies with a self-indulgent yet insecure young wizard and his companions in his legged, walking castle.

Director: [Hayao Miyazaki](#)  
Writers: [Hayao Miyazaki](#) (screenplay), [Diana Wynne Jones](#) (novel) (as Daiana Win Jônzu)  
Stars: [Chieko Baishô](#), [Takuya Kimura](#), [Tatsuya Gashûin](#) | [See full cast & crew »](#)

Dados não estruturados

# robots.txt

robots.txt é uma forma de sabermos onde o proprietário do site nos permite fazer scraping. Atualmente, grande parte dos sites já o possuem. Podemos consultá-lo apenas adicionando `/robots.txt` no final da url.

Exemplo: <http://www.amazon.com.br/robots.txt> , ao clicar no link poderemos ver o robots.txt da página da Amazon.

- Estruturas (nodes)
- Inspeccionar Elemento
- xpath

# HTML

# Estruturas (nodes)

- Tags

- título: `<title> ... </title>`
- parágrafo de texto: `<p> ... </p>`
- blocos: `<div> ... </div>`
- tabela: `<table> ... </table>`
- hiperlink (âncora): `<a> ... </a>`

- Class

- exemplo: `<p class = 'CLASSE'> ... </p>`

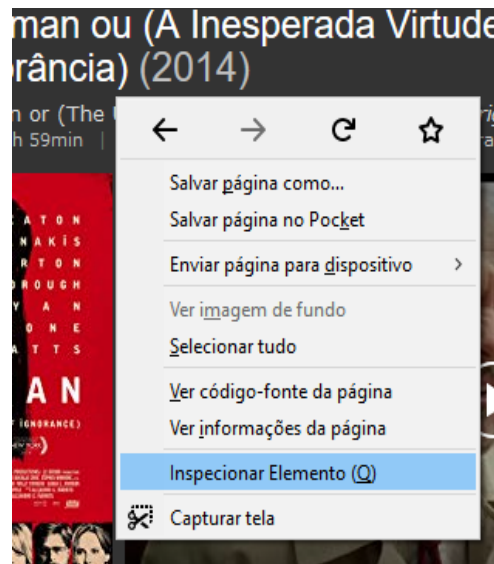
- ID

- exemplo: `<p id = 'ID'> ... </p>`



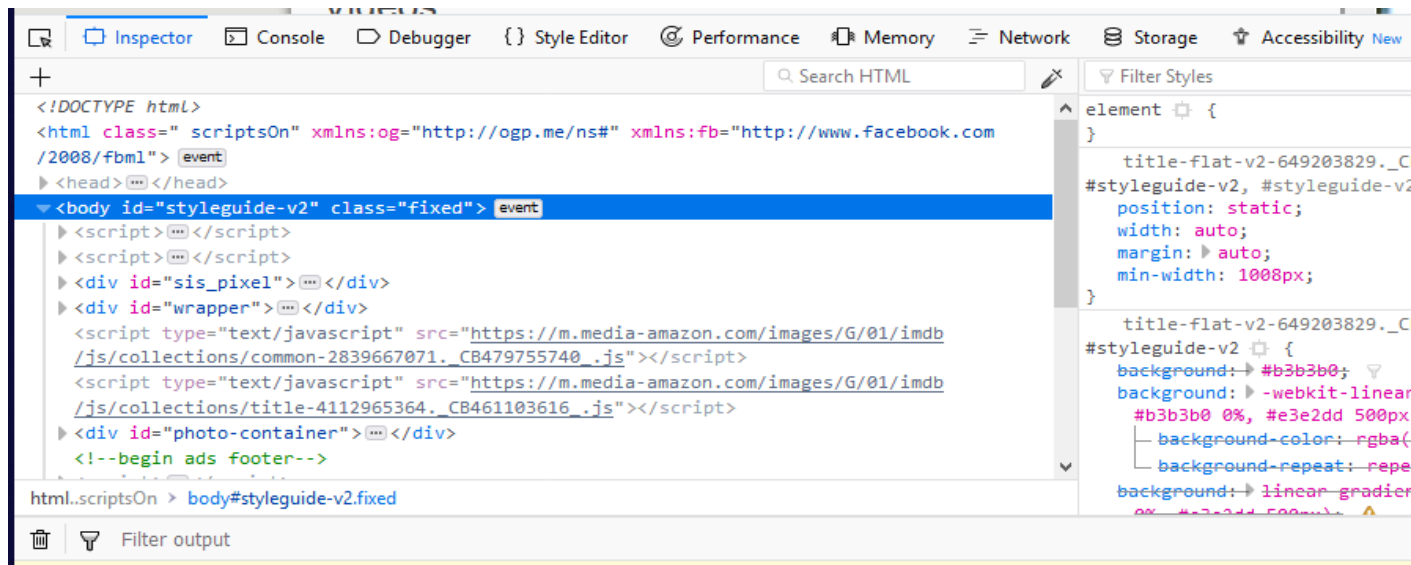
# Inspecionar Elemento

Podemos inspecionar os elementos de uma página clicando com o botão direito do mouse e acessando a opção **Inspecionar Elemento**.



# Inspeção

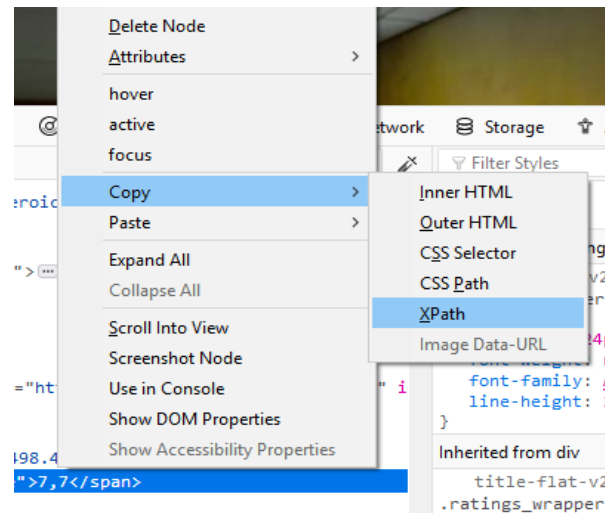
A seguinte janela aparecerá.



# xpath

O xpath assim como tag, class e id é uma forma de localizar elementos de uma página da web.

Em **Inspecionar Elemento** clique com o botão direito na parte desejada e selecione **copiar -> xpath** (pode variar de navegador para navegador).



Copiando o xpath de um node

# Vantagens e desvantagens

- Vantagens
  - Fácil utilização
  - As vezes é a única alternativa ou a mais simples.
- Desvantagens
  - É um endereço fixo, se alguma coisa mudar no site todos eu código pode ser perdido.
  - Não é universal para todos os navegadores.

- rvest
- RSelenium

# Pacotes

# rvest

Principais funções.

- `read_html()`
- `html_node()`
- `html_nodes()`
- `html_text()`
- `html_attr()`
- `html_table()`



# read\_html()

A função `read_html()` lê arquivos em html de forma similar a um `read_csv()`, que lê arquivos no formato `.csv`.

- `HTML <- read_html(https://www.wikipedia.org/)` aqui estamos lendo a página inicial do Wikipedia e armazenando em um objeto.

Data	
HTML	List of 2
node:<externalptr>	
doc :<externalptr>	
attr(*, "class")= chr [1:2] "xml_document" "xml_node"	

O seguinte objeto foi criado

# html\_node() e html\_nodes()

Essas funções extraem partes de um arquivo HTML lido com o `read_html()`. O(s) node(s) que buscamos devem ser passado(s) como uma string, da seguinte forma.

Vamos supor que ao objeto `HTML` está atribuído um arquivo html usando a função `read_html()`.

- `html_node(HTML, 'a')`, extrai apenas uma parte com a tag `a`.
- `html_nodes(HTML, '.title')`, extrai todas as partes que possuem a class `title`, em caso de class devemos colocar `.` antes do nome da classe.
- `html_node(HTML, #price)`, extrai apenas uma parte com o id `price`, em caso de id devemos colocar `#` antes do nome do id.



Podemos ainda buscar por mais de uma parte, exemplo:

- `html_node(HTML, '.price li')`, estamos extraindo apenas uma parte com a tag `li` que esta "dentro" da class `price`.

# **html\_text(), html\_attr() e html\_table()**

Esse conjunto de funções nos permite efetivamente coletar as informações dos nodes.

- `html_text()` coleta o texto daquele node.
- `html_attr()` coleta um atributo do node, por exemplo `href`.
- `html_table()` coleta uma tabela.

# Exercício (rvest)

# Exercício (rvest)

Com o que vimos até agora faça uma coleta simples no site do [IMDB](#) de algum filme que queira e colete as seguintes informações:

- Título
- Sinopse
- Ano
- Avaliação
- Link para a imagem do poster.  
Dica: use a função `html_attr()`

Tente não usar o xpath.

# RSelenium

## Por que usar o RSelenium?

Nem sempre sites não são feitos apenas em **html**, alguns possuem aplicativos interativos feitos em outras linguagens, como por exemplo **JavaScript**. Assim, precisaremos emular um navegador automatizado para interagir com a página web e coletar as informações desejadas e para isso usaremos o **Selenium**.

# Funções

Principais funções.

- `rsDriver()`
- `navigate()`
- `goBack()`
- `goForward()`
- `refresh()`
- `findElement()`
- `findElements()`
- `clickElement()`
- `sendKeysToElement()`



# rsDriver()

Inicia o navegador automatizado, o navegador *default* é o Chrome mas possuímos 2 outras opções, Firefox e phantomJS.

Utilização:

- `rD <- rsDriver(browser = 'firefox')` inicializa o Firefox e armazena em um objeto.
- `remDr <- rD$client` `remDr` é o objeto que será o cliente do servidor, ou seja, ele é responsável por enviar as requisições.

`remDr = Remote Driver`

# **navigate() , goBack() , goForward() e refresh()**

- `navigate()` navega até a url desejada. Utilização.  
`remDr$.navigate('http://www.google.com.br/')` navega até a página inicial do google.
- `goBack()` volta para a página anterior do navegador. Utilização,  
`remDr$.goBack()`
- `goForward()` avança para a página seguinte do navegador. Utilização,  
`remDr$.goForward()`
- `refresh()` atualiza a página do navegador. Utilização, `remDr$.refresh()`



# **findElement() e findElements()**

Essas funções são similares as funções `html_node()` e `html_nodes()`, respectivamente.

Argumentos:

- `using` devemos passar o que iremos usar para a função encontrar o elemento. Valores possíveis: `class name`, `name`, `xpath`, `id`, `css selector`, `tag name` e `link text`.
- `value` os valores do que estamos procurando.

Utilização:

```
remDr$findElement(using = 'class name', value = 'price')
```

# clickElement() e sendKeysToElement()

- clickElement() clica no elemento. Utilização:

```
objeto <- findElement(using = 'class', value = 'cast')  
objeto$clickElement()
```

- sendKeysToElement() aperta teclas e as envia para o site

```
objeto <- findElement(using = 'id', value = 'formulario')  
objeto$sendKeysToElement(list('digite o que quiser aqui', key = "enter"))
```

# Exercício (RSelenium)

# Exercício (RSelenium)

Com o que vimos até agora faça uma coleta simples no site da [Subamarino](#) de algum produto que queira e colete as seguintes informações:

- Nome
  - Preço
  - Código do produto
  - Valor do frete.
- Dica: use a função `sendKeysToElement()`

Tente não usar o xpath.

- Ética e boas práticas
- Agradecimento

# Encerramento

# Ética e boas práticas

- Respeitar o robots.txt
- Não fazer mais de uma consulta por segundo. Basta colocar um `sys.sleep(1)` no seu código.
- Não ultrapassar o limite de informações coletadas por dia, se houver. Entretanto, sempre manter o bom senso.

Essas boas práticas evitam o sobrecarregamento dos servidores e cause prejuízo a essas empresas.

# Captcha

# Agradecimento

Daniel dos Santos  
Universidade Federal Fluminense

[Github](#)