# Selenium

PUCRS

Pontifícia Universidade Católica
do Rio Grande do Sul

## – Aula 08 –
## Coleta, Preparação e Análise de Dados

Prof. Me. Lucas R. C. Pessutto

Slides adaptados do material do Prof. Lucas Silveira Kupssinskü e do Prof. Luan Fonseca Garcia

# Ferramentas vistas para web scraping

- Para obter páginas
  - requests
- Para obter informações de uma página
  - BeautifulSoup
  - Regex
  - Seletores CSS
- Problema:
  - Não “navegamos” pela página como um usuário

# Selenium

- Projeto **open source** que fornece um conjunto de ferramentas para **automatizar** testes em web browsers.
- **Selenium WebDriver**: comanda um web browser como se fosse um usuário real.
- **Selenium IDE**: extensão para diversos browsers que permite gravar e re-executar testes em um browser.
- **Selenium Grid**: uso do WebDriver em um grid de máquinas para testar diversos browser/máquinas.

# WebDriver

- É um protocolo **independente** de **plataforma** ou **linguagem** para programas externos controlarem o comportamento de um browser.

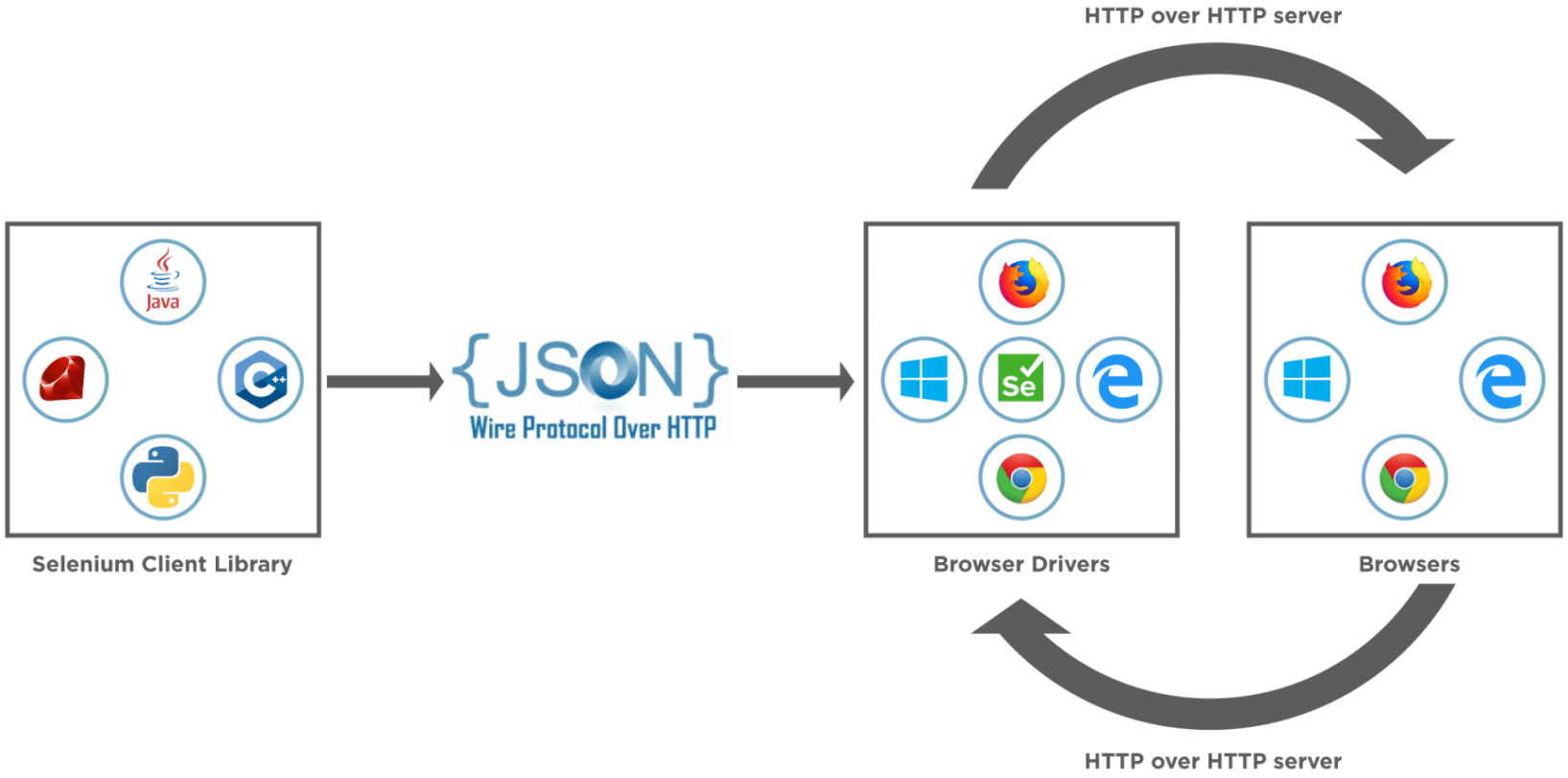
- Um conjunto de interfaces permite descobrir e manipular nodos **DOM** em documentos web e também controlar o **comportamento** de usuários.

- Permite que desenvolvedores web escrevam testes automatizados para simular o comportamento de um agente externo.

# WebDriver - História

- Ideia começou em 2004 como uma ferramenta interna na ThoughtWorks para automatização de testes.
- Em 2009 desenvolvedores da ThoughtWorks e do Google resolveram unir seus projetos de automação de testes e passaram a chamar de Selenium WebDriver.
- Em 2012 negociaram com a W3C para que virasse um padrão da web.
- Em 2018 foi lançada uma recomendação da W3C definindo uma API.

# Selenium – WebDriver Architecture




Fonte da imagem: https://www.toolsqa.com/selenium-webdriver/selenium-webdriver-architecture/

# Selenium – Motivos para utilizar no Scraping

| Feature | Descrição |
|---|---|
| Suporte a páginas dinâmicas | Consegue acessar páginas que são renderizadas/carregadas dinamicamente após ações do usuário. |
| Funciona em conjunto com o browser | Desenvolvedores de navegadores web desenvolvem e distribuem seus webdrivers para Selenium, garantindo qualidade e funcionalidade da integração do Selenium com os browsers. |
| Technology Agnostic | Embora navegadores sejam bastante distintos em sua implementação, o Selenium fornece uma API comum de acesso. |
| Simulação de usuário | Como o Selenium simula um usuário navegando no browser, o website “não tem” como identificar que se trata de um script automatizado e não de uma pessoa que está navegando. |
| Visualização dos resultados | É possível acompanhar o progresso do scraping através de uma janela do navegador. |

# Driver

É o programa que de fato se **comunica** com o browser para que ele realize ações.

Cada browser específico vai precisar de um driver específico, que em geral é desenvolvido pelo mesmo desenvolvedor do browser.

A API WebDriver na prática se comunica com o driver, que então pede para o browser executar algo.

A partir da versão **4.6** do Selenium todo controle de escolha da versão do driver correta para o browser e sua versão é feita pelo Selenium Manager, sem necessidade de intervenção do desenvolvedor.

# Comandos Comuns para Navegador

- webdriver.get(“link”)
  - Driver manda browser carregar na sessão aberta o documento html com o endereço passado como parâmetro.
- webdriver.refresh()
  - Atualiza a sessão navegador.
- webdriver.maximize_window()
  - Maximiza a janela do navegaor.
- webdriver.fullscreen_window()
  - Deixa o navegador em tela cheia.
- webdriver.get_screenshot_as_png()
  - Tira uma screenshot do navegador.

# Selenium – Um primeiro exemplo

```python
from selenium import webdriver
from selenium.webdriver.common.by import By


driver = webdriver.Chrome()

driver.get("https://www.google.com")

driver.title # => "Google"

driver.implicitly_wait(0.5)

search_box = driver.find_element(By.NAME, "q")
search_button = driver.find_element(By.NAME, "btnK")

search_box.send_keys("Selenium")
search_button.click()

driver.find_element(By.NAME, "q").get_attribute("value") # => "Selenium"

driver.quit()
```

# Objeto WebElement

Quando chamamos a função *get()*, o driver realiza uma chamada http e faz o navegador acessar o site cujo endereço foi passado por parâmetro.

Isto faz com que seja possível ter acesso aos elementos web utilizando a estrutura de árvore DOM criada pelo navegador.

Podemos encontrar elementos através da função *find_element(),* que procura na árvore o elemento que corresponde aos filtros passados na chamada e retorna um objeto do tipo WebElement.

# Selenium find_element()

Quando executamos a função *find_element()* precisamos definir por qual critério ela irá buscar o elemento web.

As possibilidades são estas:

- **ID** = busca pelo atributo id
- **XPATH** = busca pelo XPATH do elemento
- **LINK_TEXT** = busca pelo texto de um link
- **PARTIAL_LINK_TEXT** = busca pelo texto parcial de um link
- **NAME** = busca pelo atributo "name“
- **TAG_NAME** = busca pelo nome da tag
- **CLASS_NAME** = busca pelo nome do atributo classe
- **CSS_SELECTOR** = busca pelo nome do seletor CSS

Podemos realizar uma busca dentro de um WebElement, o que restringe ela apenas aos seus descendentes na árvore DOM.

# Outros comandos

- webelement.send_keys(‘texto’)
  - Envia ao elemento um texto.
  - Útil para preencher um campo de texto.

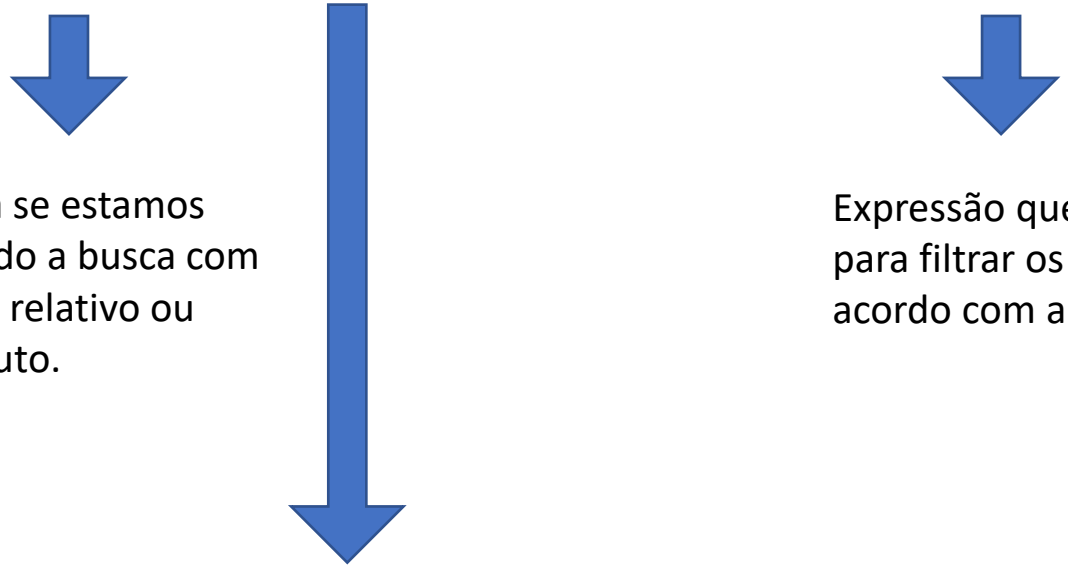
- webelement.click()
  - Executa uma ação de clique no elemento dentro do navegador.
  - Útil para apertar botões, abrir menus, selecionar checkbox, etc.
- webelement.get_attribute(‘nome do atributo’)
  - Retorna o valor da propriedade ou atributo com do WebElement.
  - Se a propriedade não existir, retorna None.
  - Ex: webelement.get_attribute(“href”) retorna o texto do link representado por um elemento do tipo *a (anchor)*

# XPath

- É uma sintaxe que permite identificar e localizar elementos em um documento XML ou HTML
- *Path like*
- XPath usa caminhos para definir elementos XML
- XPath é um padrão W3C
- Pode ser utilizado diretamente na busca do navegador

# Xpath – Sintaxe geral

```
//nomeelemento[expressao]
```



Indica se estamos fazendo a busca com XPath relativo ou absoluto.

Indica qual a tag estamos buscando (* para qualquer tag).

Expressão que pode ser utilizada para filtrar os elementos de acordo com algum critério

# Xpath – alguns exemplos

`//p`

*Todos elementos de nome* ***p***

# Xpath – alguns exemplos

## /html/head/title

*Todos elementos com nome **title** que são filhos do elemento **head** que é filho do elemento **html***

```
<html>
▼<head>
    <title>The Dormouse's story</title>
  </head>
▼<body>
••• ▼<p class="title"> == $0
      <b> The Dormouse's story </b>
    </p>
  ▼<p class="story">
      " Once upon a time there were three little sisters; and their names were "
      <a class="sister" href="http://example.com/elsie" id="link1"> Elsie </a>
      " , "
      <a class="sister" href="http://example.com/lacie" id="link2"> Lacie </a>
      " and "
      <a class="sister" href="http://example.com/tillie" id="link3"> Tillie </a>
      " ; and they lived at the bottom of a well. "
    </p>
    <p class="story"> ... </p>
  </body>
</html>
```

# Xpath – alguns exemplos

```
//*[@id='link1']
```

*Todos elementos que tenham um atributo **id** com valor **link1***

```
<html>
▼<head>
    <title>The Dormouse's story</title>
  </head>
▼<body>
  ▼<p class="title">
      <b> The Dormouse's story </b>
    </p>
  ▼<p class="story">
      " Once upon a time there were three little sisters; and their names were "
...   <a class="sister" href="http://example.com/elsie" id="link1"> Elsie </a> == $0
      " , "
      <a class="sister" href="http://example.com/lacie" id="link2"> Lacie </a>
      " and "
      <a class="sister" href="http://example.com/tillie" id="link3"> Tillie </a>
      " ; and they lived at the bottom of a well. "
    </p>
    <p class="story"> ... </p>
  </body>
</html>
```

# XPath – alguns exemplos

```
//*[contains(@id, 'link')]
```

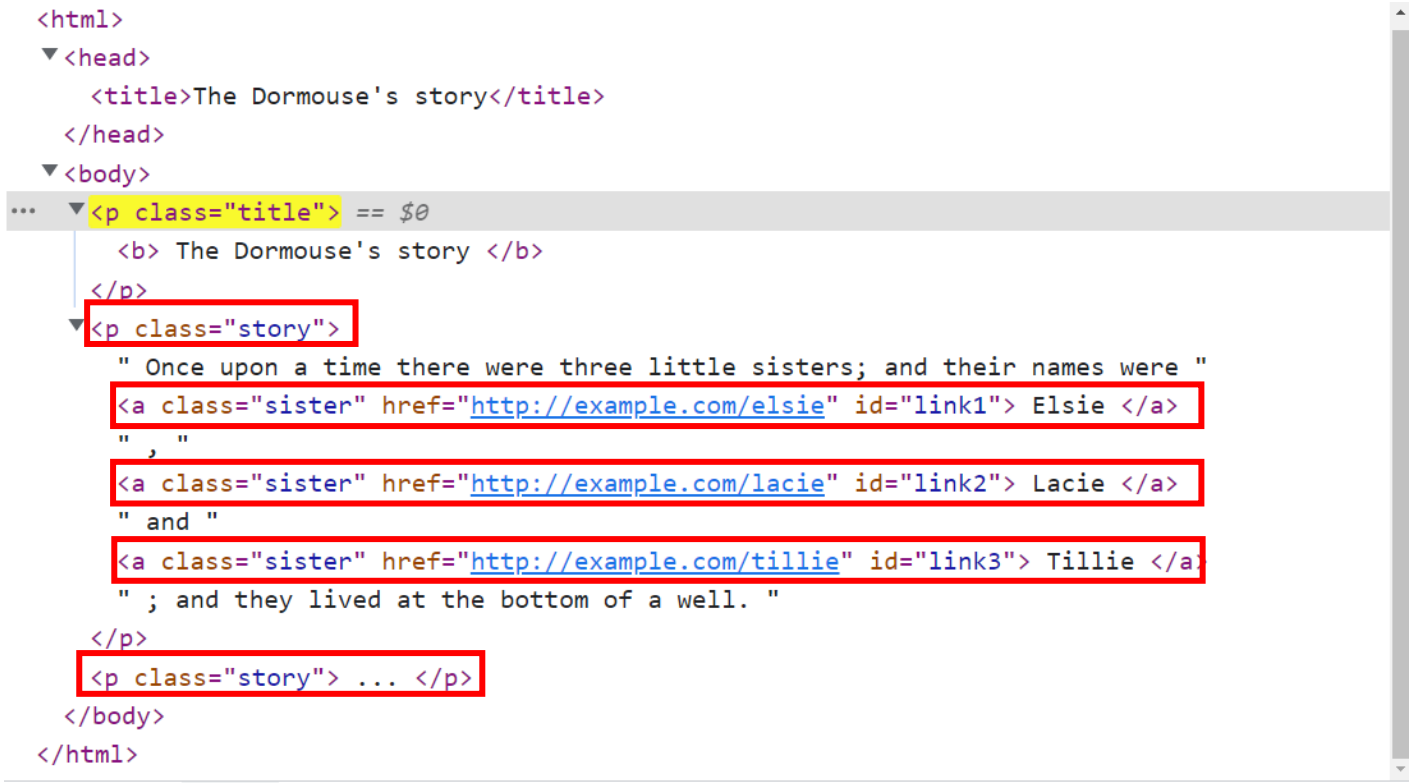
*Todos elementos que tenham um atributo **id** com valor **link***

```
<html>
▼<head>
    <title>The Dormouse's story</title>
  </head>
▼<body>
••• ▼<p class="title"> == $0
      <b> The Dormouse's story </b>
    </p>
  ▼<p class="story">
      " Once upon a time there were three little sisters; and their names were "
      <a class="sister" href="http://example.com/elsie" id="link1"> Elsie </a>
      " , "
      <a class="sister" href="http://example.com/lacie" id="link2"> Lacie </a>
      " and "
      <a class="sister" href="http://example.com/tillie" id="link3"> Tillie </a>
      " ; and they lived at the bottom of a well. "
    </p>
    <p class="story"> ... </p>
  </body>
</html>
```

# XPath – alguns exemplos

*Todos elementos que tenham um atributo **classe** que o valor comece com **s***

```
//*[starts-with(@class, 's')]
```



# XPath – alguns exemplos

*Todos elementos que tenham um atributo **id** com valor **link1** ou **link2***

```
//*[@id='link1' or @id='link2']
```

```
<html>
 <head>
   <title>The Dormouse's story</title>
 </head>
 <body>
  <p class="title"> == $0
    <b> The Dormouse's story </b>
  </p>
  <p class="story">
    " Once upon a time there were three little sisters; and their names were "
    <a class="sister" href="http://example.com/elsie" id="link1"> Elsie </a>
    " , "
    <a class="sister" href="http://example.com/lacie" id="link2"> Lacie </a>
    " and "
    <a class="sister" href="http://example.com/tillie" id="link3"> Tillie </a>
    " ; and they lived at the bottom of a well. "
  </p>
  <p class="story"> ... </p>
 </body>
</html>
```

# XPath – Predicados

Predicados são usados dentro de colchetes para definir restrições para o resultado do XPath

| Path Expression | Result |
|---|---|
| /bookstore/book[1] | Seleciona a primeira tag book que é filha de uma tag bookstore. |
| /bookstore/book[last()] | Seleciona a última tag book que é filha de uma tag bookstore. |
| /bookstore/book[last()-1] | Seleciona a penultima tag book que é filha de bookstore |
| /bookstore/book[position()<3] | Seleciona os primeiros dois livros filhos de bookstore |
| //title[@lang] | Seleciona todas as tags title que possuem lang como atributo |
| //title[@lang='en'] | Seleciona todas as tags title que possuem lang como atributo com valor en |
| /bookstore/book[price>35.00] | Seleciona todas as tags book que tenham como filho uma tag price com conteúdo maior que 35 |
| /bookstore/book[price>35.00]/title | Seleciona os títulos dos livros que tenham como filho uma tag price com conteúdo maior que 35 |

# XPath – Suporte em python

- Suporte limitado a Xpath no modulo padrão do python
  - xml.etree.ElementTree
- Para um suporte mais abrangente de XPath devem ser utilizadas outras opções:
  - lxml
  - selenium

# Referências

- Documentação do Selenium:
  - https://www.selenium.dev/documentation/
- Document Object Model:
  - https://www.w3.org/TR/WD-DOM/introduction.html
- XPath:
  - https://en.wikipedia.org/wiki/XPath

# Demo Selenium + XPath