

# Selenium

Automatizando navegadores da web.

Realizando scraping de JavaScript.

# Selenium

O que é o Selenium?



Selenium é um conjunto de ferramentas para automatizar navegadores da web.

Site oficial:

<https://selenium.dev/>

Documentação:

<https://selenium.dev/documentation/en/>

<https://selenium-python.readthedocs.io/>

# Selenium

## O que é o Selenium?

O Selenium é muito importante, mas, em sua essência, é um conjunto de ferramentas para automação do navegador da Web que usa as melhores técnicas disponíveis para controlar remotamente as instâncias do navegador e emular a interação do usuário com o navegador.

Permite que os usuários simulem atividades comuns executadas pelos usuários finais; inserir texto em campos, selecionar valores suspensos e caixas de seleção e clicar em links nos documentos. Ele também fornece muitos outros controles, como movimento do mouse, execução arbitrária de JavaScript e muito mais.

Embora usado principalmente para testes front-end de sites, o Selenium é, em sua essência, uma biblioteca de agentes de usuário de navegador. As interfaces são onipresentes para sua aplicação, o que incentiva a composição com outras bibliotecas para se adequar ao seu objetivo.

# Selenium

O que é o Selenium?

Estas são as ferramentas disponíveis.

 <b>Selenium WebDriver</b> <p>If you want to create robust, browser-based regression automation suites and tests, scale and distribute scripts across many environments, then you want to use Selenium WebDriver, a collection of language specific bindings to drive a browser - the way it is meant to be driven.</p> <p><b>DOWNLOAD</b></p>	 <b>Selenium IDE</b> <p>If you want to create quick bug reproduction scripts, create scripts to aid in automation-aided exploratory testing, then you want to use Selenium IDE; a Chrome and Firefox add-on that will do simple record-and-playback of interactions with the browser.</p> <p><b>DOWNLOAD</b></p>	 <b>Selenium Grid</b> <p>If you want to scale by distributing and running tests on several machines and manage multiple environments from a central point, making it easy to run the tests against a vast combination of browsers/OS, then you want to use Selenium Grid.</p> <p><b>DOWNLOAD</b></p>
---	---	---



# Selenium

## O que é o Selenium WebDriver?

O Selenium WebDriver aceita comandos e os envia para um navegador. Isso é implementado através de um driver de navegador específico, que envia comandos para um navegador e recupera resultados.

# Selenium

## O que é o Selenium WebDriver?

A instalação do Selenium é bem diferente da instalação de outras ferramentas comerciais. Para usar o Selenium em seu projeto de automação, você precisa instalar as bibliotecas de ligações (bindings libraries) de linguagens para a linguagem de sua escolha. Além disso, você precisará de binários do WebDriver para os navegadores nos quais deseja automatizar e executar o teste.

# Selenium

## O que é o Selenium WebDriver?

A maioria dos drivers de navegador realmente inicia e acessa um aplicativo de navegador (como Firefox, Google Chrome, Internet Explorer, Safari ou Microsoft Edge); também há um driver de navegador HtmlUnit, que simula um navegador usando o navegador HtmlUnit, que é executado sem uma interface gráfica. É possível também usar o Chrome Driver e o Gecko Driver (Firefox) em modo headless (sem interface gráfica).

# Selenium

Como utilizo o Selenium WebDriver no Python?

Para utilizar o *Selenium WebDriver* com *Python*, existe um pacote (bindings libraries) de mesmo nome que pode ser instalado através do comando:

```
$ pip install selenium
```



# Selenium

## Baixando o Mozilla Gecko Driver

Vamos agora baixar o driver da Mozilla no link a seguir:

<https://github.com/mozilla/geckodriver/releases/download/v0.26.0/geckodriver-v0.26.0-win64.zip>

O arquivo deve ser descompactado e sua pasta deve ser adicionada ao path do Windows.

# Selenium

## Primeiro exemplo

Vamos criar uma página em PHP e executar localmente para testar nosso primeiro exemplo.

*Esta aula depende do conhecimento adquirido na seção “Rastreado Formulários e Login” pois vamos usar o PHP para rodar páginas web localmente.*

Crie um arquivo chamado exemplo1.php e coloque o conteúdo a seguir.

# Selenium

## Primeiro exemplo

```
<html>
  <head>
    <title>Olá Aluno!!!</title>
  </head>
  <body>
    <h1>Curso Python Web Scraping</h1>
    <form action="destino.php" method="POST">
      Informe os dados abaixo:<br><br>
      <table>
        <tr>
          <td>Nome:</td>
          <td><input type="text" name="nome" id="nome" size="50" maxLength="100"></input></td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

# Selenium

## Primeiro exemplo

```
...  
    <tr>  
        <td>E-mail:</td>  
        <td><input type="text" name="email" id="email" size="50" maxLength="100"></input></td>  
    </tr>  
    <tr>  
        <td>Celular:</td>  
        <td><input type="text" name="celular" id="celular" size="20" maxLength="14"></input></td>  
    </tr>  
    <tr>  
        <td><input type='submit' name='enviar' id='enviar' value='Enviar dados'></td>  
    </tr>  
</table>  
</form>  
</body>  
</html>
```

# Selenium

## Primeiro exemplo

Acesse a pasta onde o arquivo PHP foi criado e inicie com o comando:

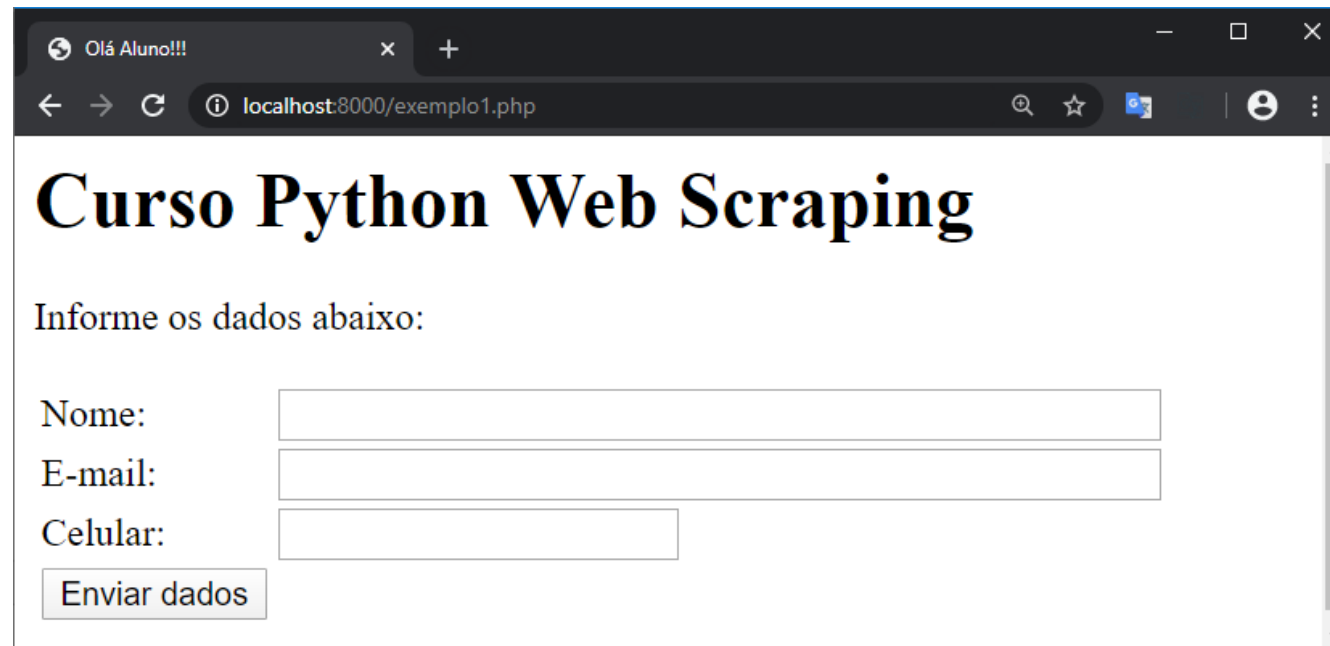
```
$ php -S localhost:8000
```

# Selenium

## Primeiro exemplo

Abra o navegador e acesse a página :

<http://localhost:8000/exemplo1.php>



A screenshot of a web browser window. The address bar shows 'localhost:8000/exemplo1.php'. The page title is 'Curso Python Web Scraping'. Below the title, it says 'Informe os dados abaixo:'. There are three input fields labeled 'Nome:', 'E-mail:', and 'Celular:'. Below these fields is a button labeled 'Enviar dados'.

# Selenium

## Primeiro exemplo

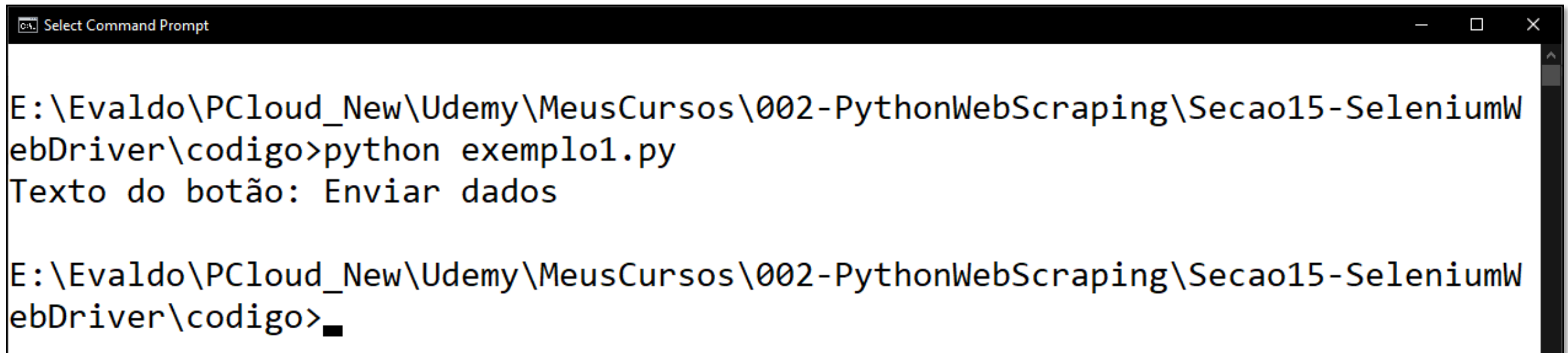
Agora vamos escrever o arquivo exemplo1.py

```
from selenium import webdriver
# webdriver representa o Browser
driver = webdriver.Firefox()
# Navegar até a página exemplo1
driver.get("http://localhost:8000/exemplo1.php")
# Find Element By retorna um objeto WebElement
objeto = driver.find_element_by_id("enviar")
# Pegando o atributo value para obter o texto do botão
print('Texto do botão:', objeto.get_attribute('value'))
driver.close()
```

# Selenium

## Primeiro exemplo

Ao executar o arquivo exemplo1.py, você verá que será aberta uma janela do Firefox e fechada em seguida. Veja o texto impresso no prompt.



```
Select Command Prompt

E:\Evaldo\PCloud_New\Udemy\MeusCursos\002-PythonWebScraping\Secao15-SeleniumWebDriver\codigo>python exemplo1.py
Texto do botão: Enviar dados

E:\Evaldo\PCloud_New\Udemy\MeusCursos\002-PythonWebScraping\Secao15-SeleniumWebDriver\codigo>
```



# Selenium

## Segundo exemplo

Vamos agora criar o arquivo exemplo2.php.

```
<html>
  <head>
    <title>Olá Aluno!!!</title>
    <script src="http://code.jquery.com/jquery-1.11.3.min.js"></script>
    <script>
      function sleep(time){
        return new Promise((resolve) => setTimeout(resolve, time));
      }

      sleep(5000).then(() => {
        $("form").append("<tr><td><input type='submit' name='enviar' id='enviar' value='Enviar dados'></td></tr>");
      });
    </script>
  </head>
  ...
```

# Selenium

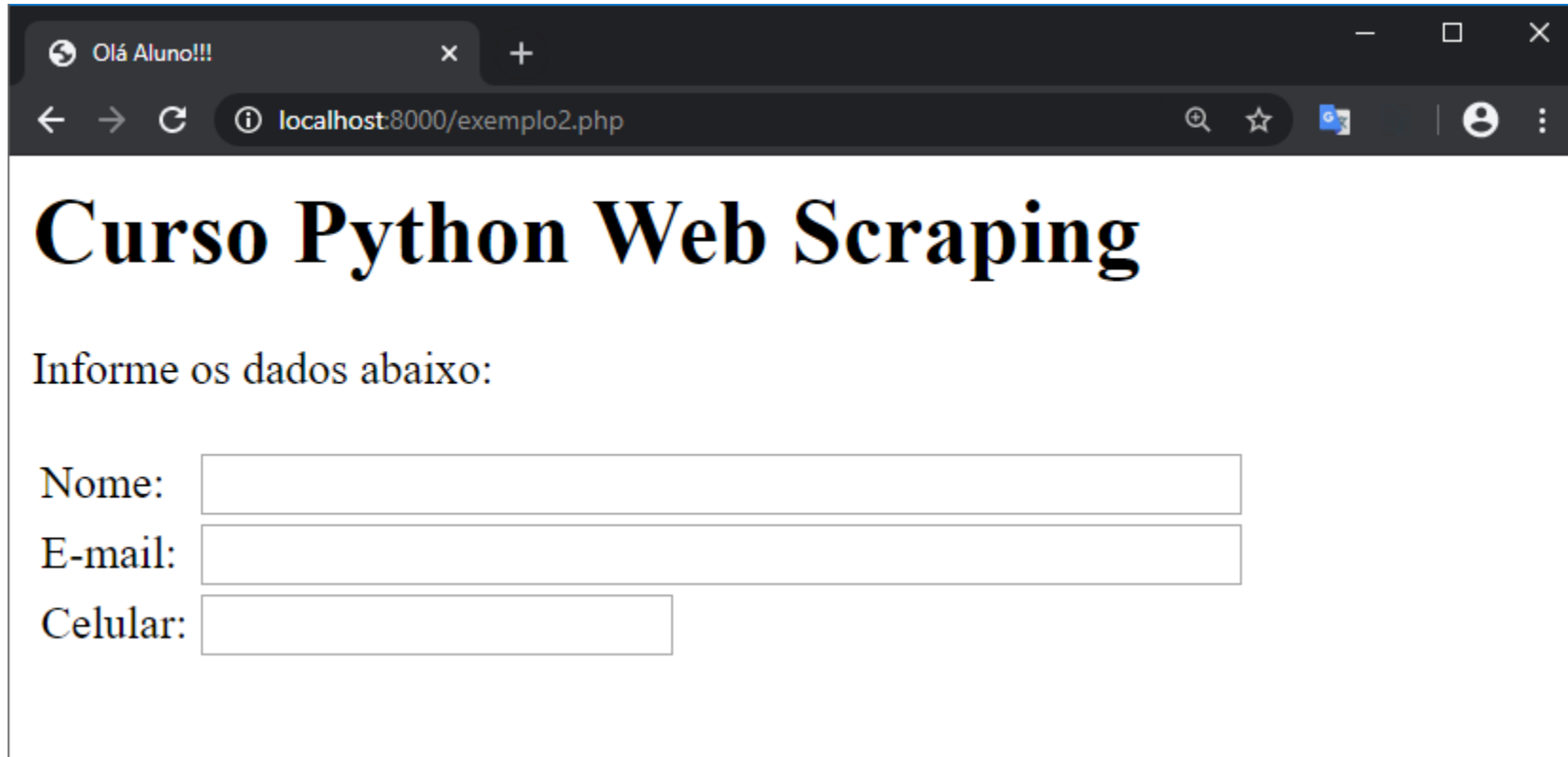
## Segundo exemplo

```
...  
<body>  
<h1>Curso Python Web Scraping</h1>  
<form action="destino.php" method="POST">  
  Informe os dados abaixo:<br><br>  
  <table>  
    <tr>  
      <td>Nome:</td>  
      <td><input type="text" name="nome" id="nome" size="50" maxlength="100"></input></td>  
    </tr>  
    <tr>  
      <td>E-mail:</td>  
      <td><input type="text" name="email" id="email" size="50" maxlength="100"></input></td>  
    </tr>  
    <tr>  
      <td>Celular:</td>  
      <td><input type="text" name="celular" id="celular" size="20" maxlength="14"></input></td>  
    </tr>  
  </table>  
</form>  
</body>  
</html>
```

# Selenium

## Segundo exemplo

Quando você acessar a página a mesma será exibida assim:



A screenshot of a web browser window. The address bar shows 'localhost:8000/exemplo2.php'. The page title is 'Olá Aluno!!!'. The main content of the page is a form titled 'Curso Python Web Scrapping'. Below the title, it says 'Informe os dados abaixo:'. There are three input fields: 'Nome:', 'E-mail:', and 'Celular:'. Each field has a corresponding text input box.

Olá Aluno!!!

localhost:8000/exemplo2.php

## Curso Python Web Scrapping

Informe os dados abaixo:

Nome:

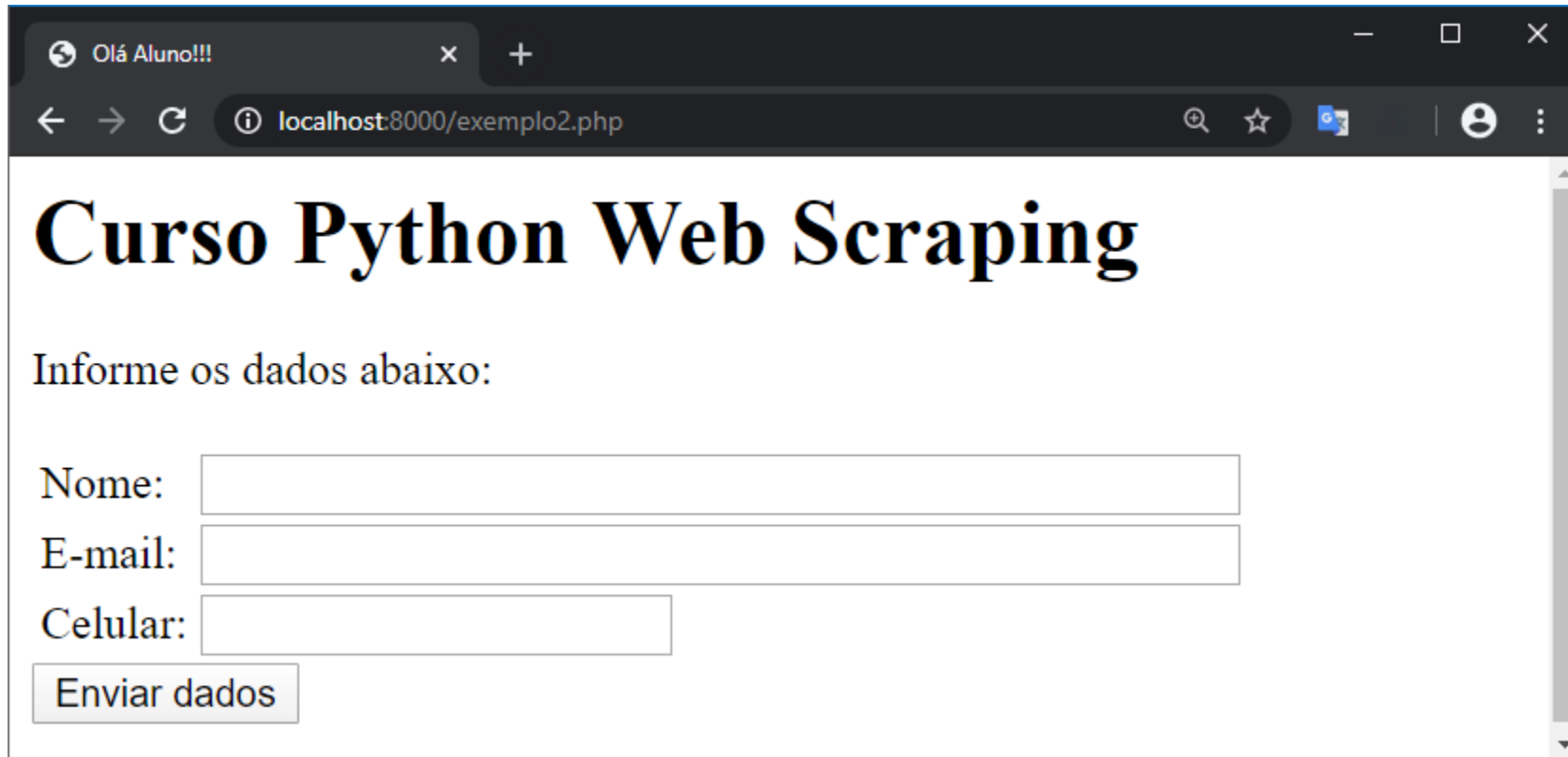
E-mail:

Celular:

# Selenium

## Segundo exemplo

Depois de alguns segundos aparecerá um botão.



A screenshot of a web browser window. The address bar shows 'localhost:8000/exemplo2.php'. The page title is 'Olá Aluno!!!'. The main content of the page is a form titled 'Curso Python Web Scrapping'. Below the title, it says 'Informe os dados abaixo:'. There are three input fields: 'Nome:', 'E-mail:', and 'Celular:'. Below these fields is a button labeled 'Enviar dados'.

# Selenium

## Segundo exemplo

Vamos agora tentar ler o conteúdo da página criada usando *request*.

Para isso crie o arquivo exemplo2\_1.py

```
from urllib.request import urlopen  
html = urlopen("http://localhost:8000/exemplo2.php")  
print(html.read().decode('utf-8'))
```

# Selenium

## Segundo exemplo

Execute o arquivo exemplo2\_1.py e observe que o botão “Enviar dados” não aparece no resultado.

# Selenium

## Segundo exemplo

Agora vamos criar o arquivo exemplo2\_2.py.

```
from selenium import webdriver

print("Início")
driver = webdriver.Firefox()
driver.get("http://localhost:8000/exemplo2.php")
print(driver.find_element_by_id("enviar").get_attribute('value'))
driver.close()
print("Fim")
```

# Selenium

## Segundo exemplo

Execute o arquivo exemplo2\_2.py e observe que ocorrerá o erro a seguir:

Message: Unable to locate element: [id="enviar"]

Isso porque, como o botão com o id “enviar” só é gerado após alguns segundos, porque está sendo criado via JavaScript, o sistema não consegue encontrar este elemento na página.



# Selenium

## Segundo exemplo

O conteúdo da página que conseguimos pegar não é o mesmo que vemos no browser após alguns segundos.

Para resolver esta questão vamos criar o arquivo exemplo2\_3.py.

# Selenium

## Segundo exemplo

```
from selenium import webdriver

def aguardar(driver):
    while True:
        try:
            if driver.find_element_by_id("enviar"):
                print(driver.find_element_by_id("enviar").get_attribute('value'))
                return
        except:
            print('Deu erro')
            pass
    ...
```

# Selenium

## Segundo exemplo

```
...  
print("Início")  
driver = webdriver.Firefox()  
driver.get("http://localhost:8000/exemplo2.php")  
aguardar(driver)  
driver.close()  
print("Fim")
```

# Selenium

## Segundo exemplo

Ao executar o arquivo exemplo2\_3.py, você verá que o sistema vai retornar o valor do botão “Enviar dados”

# Selenium

## Terceiro exemplo

No terceiro exemplo vamos usar o arquivo exemplo1.php que é nosso formulário, vamos criar o arquivo que será executado para mostrar na tela os dados do formulário (destino.php) e o programa em Python para executar o formulário e retornar os dados exibidos na página destino (exemplo3.py).

# Selenium

## Terceiro exemplo

### Exemplo3.py

```
from selenium import webdriver

driver = webdriver.Firefox()
driver.get("http://localhost:8000/exemplo1.php")
nome = driver.find_element_by_name("nome")
email = driver.find_element_by_name("email")
celular = driver.find_element_by_name("celular")
enviar = driver.find_element_by_name("enviar")
nome.send_keys("Evaldo Wolkers")
email.send_keys("evaldowolkers@gmail.com")
celular.send_keys("(28)99948-3074")
enviar.click()
print(driver.page_source)
driver.close()
```

# Selenium

## Terceiro exemplo

### destino.php

```
<html>
  <head>
    <title>Olá Aluno!!!</title>
  </head>
  <body>
    <?php
      $nome = $_POST['nome'];
      $email = $_POST['email'];
      $celular = $_POST['celular'];

      if (empty($nome) or empty($email) or empty($celular))
      {echo "Favor informar todos os campos.";}
      else{echo "Olá " . $nome . "<br>Seu e-mail é: " . $email . "<br>Seu celular é: " . $celular;}
    ?>
  </body>
</html>
```

# Selenium

## Quarto exemplo

```
from selenium import webdriver
from time import sleep

driver = webdriver.Firefox()
driver.get("https://www.pythonparatodos.com.br")
# current_url: Retorna a URL corrente
print(driver.current_url)
driver.get("http://www.pythonparatodos.com.br/cursos")
sleep(3)
# Pressiona o botão Voltar
driver.back()
sleep(3)
...
```



# Selenium

## Quarto exemplo

```
...  
# Pressiona o botão Avançar  
driver.forward()  
sleep(3)  
# Recarrega a página  
driver.refresh()  
# Retorna o título da página  
print(driver.title)  
# Salvar uma imagem (screenshot) com a página atual  
driver.save_screenshot('teste.png')  
driver.close()
```

# Selenium

## Quinto exemplo

```
from selenium import webdriver

driver = webdriver.Firefox()
driver.get("http://localhost:8000/exemplo1.html")
driver.execute_script('window.alert("Script Executado")')

# Não vou fechar para ver o script na tela
#driver.close()
```

# FIM