

Automação com Selenium usando Python

Selenium Projects

	Selenium Webdriver	Manipula navegadores Web nativamente, suportando várias linguagens de programação.
	Selenium IDE	Extensão do Chrome, Firefox e Edge. gravar e reproduzir interações com o navegador.
	Selenium Grid	Usa o Selenium Webdriver para rodar testes em várias máquinas ao mesmo tempo.

**Selenium Webdriver**

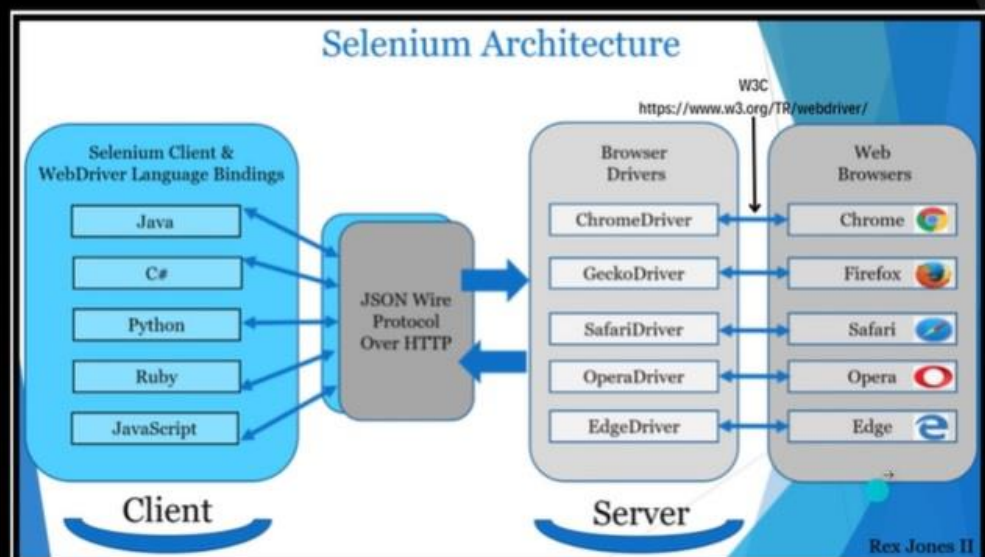
É uma Biblioteca / Módulo para interagir com navegadores

Também pode ser considerado uma API

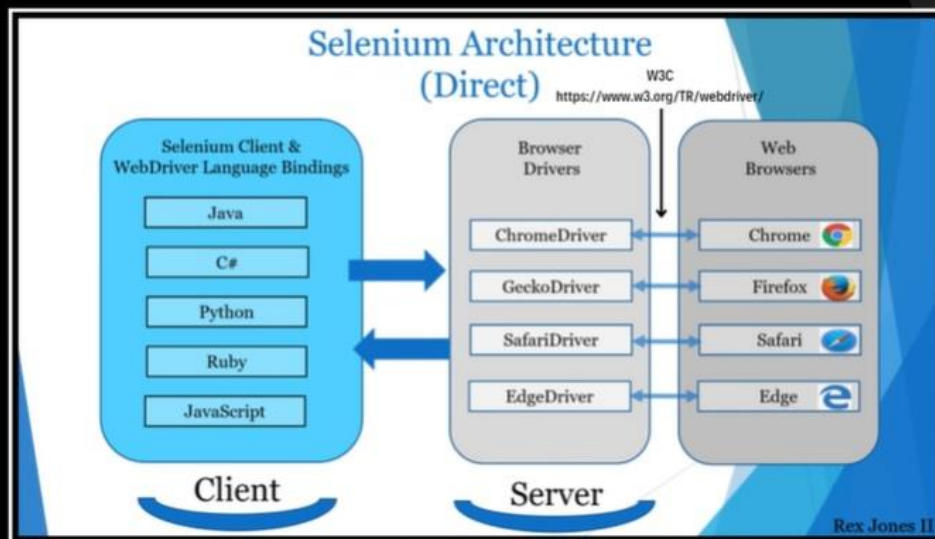
Usado para automatizar testes de GUI

Graphic User Interface

Selenium 3



Selenium 4



Nesse curso usaremos o Selenium webdriver, com a linguagem de programação python. Para isso precisamos instalar o python e também a IDE Pycharm, onde iremos fazer os testes.

<https://www.selenium.dev> – site oficial do selenium

<https://www.selenium.dev/downloads/> - para efetuar a instalação do selenium

pip install selenium - basta digitar no terminal.

No entanto iremos utilizar um ambiente virtual para o uso do selenium. Para isso, no terminal do Pycharm digite:

Python -m venv venv - para criar um ambiente virtual

Observação: Ao criar um projeto no Pycharm, automaticamente ele cria um ambiente virtual (venv), o comando acima pode ser utilizado caso ocorra algum erro.

Na sequência precisamos ativar o venv que foi criado, para isso basta digitar no terminal:

Venv\Scripts\Activate.ps1 – para ativar o venv

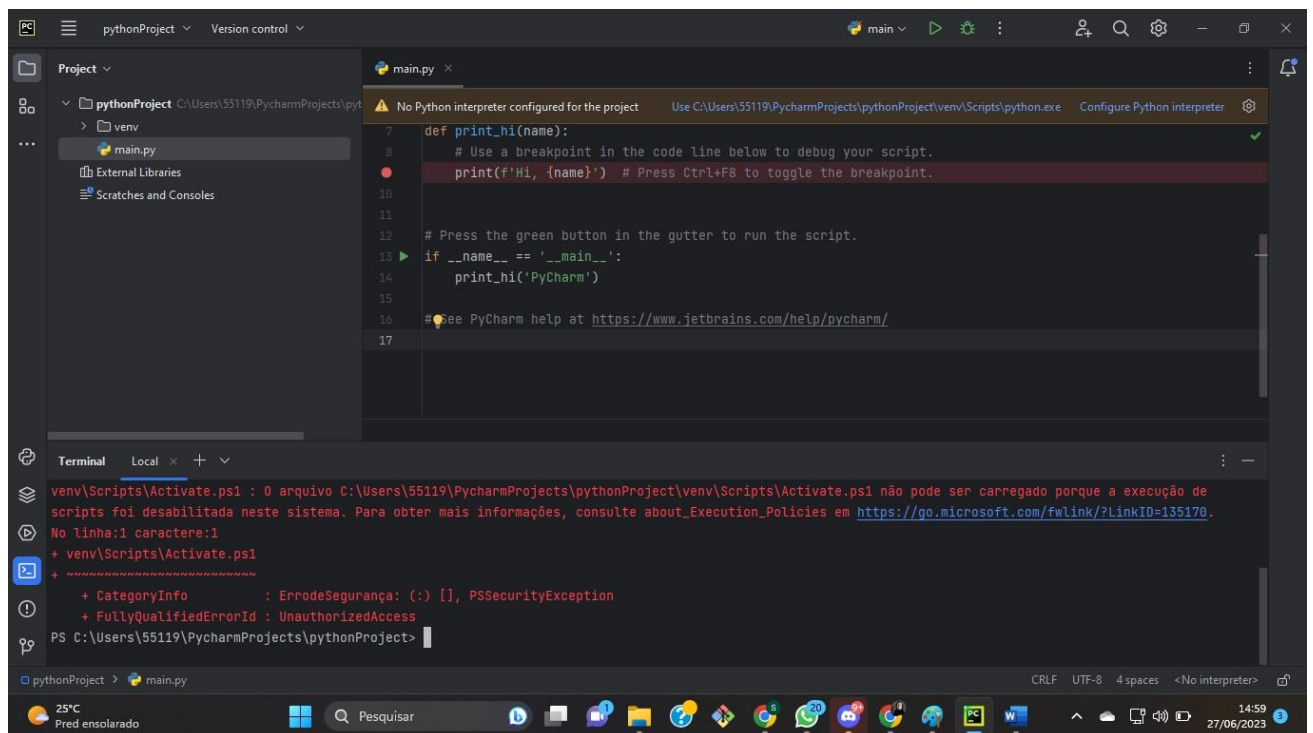
Agora podemos instalar o selenium:

Pip install selenium – digite no terminal

Para saber se a instalação foi feita basta digitar:

Pip show selenium

Erro ao ativar o venv:



Olá Simone, esse erro é por causa de uma política de segurança do PowerShell, para prevenir scripts maliciosos.

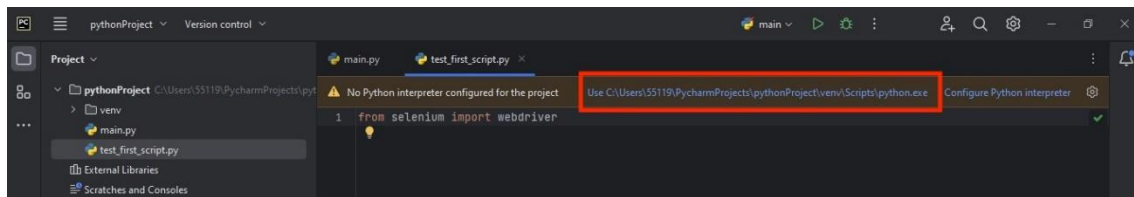
Pra permitir a execução do script de ativação do ambiente virtual, rode o seguinte comando no terminal:

Set-ExecutionPolicy Unrestricted -Scope Process

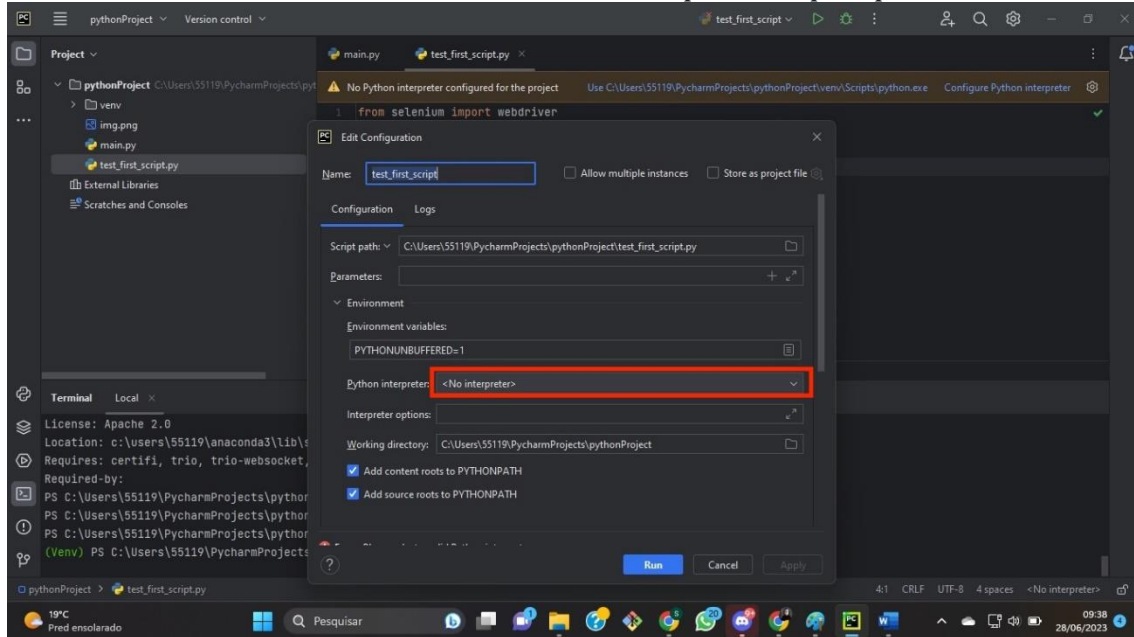
Esse comando vai habilitar a execução do script na sessão aberta do Powershell.

Outro erro:

Clica nessa opção e vê se resolve:



Se não resolver, clica nessa abaixo, e me manda o print do que aparecer:



Opa, fiz a primeira opção, deu certo, obrigada professor!! Mas uma dúvida, eu acabei instalando o selenium no windows, e ele também aparece no venv, tem algum problema? Por que tem que ser num ambiente virtual? Não entendi.



[Leonardo](#) — Instrutor

Que bom que deu certo Simone!

E não tem problema estar nos 2. Mas eu recomendo sempre trabalhar no `venv` do projeto, pq dessa forma vc isola as configurações daquele projeto, e se caso vc precise ter outro projeto na sua maquina, usando outras versões, estando cada projeto com seu `venv`, vai evitar conflitos e problemas que dariam se vc tivesse instalado tudo local.

Observação: Problemas resolvidos

Mapeando Elementos da tela

Tipos de Locators

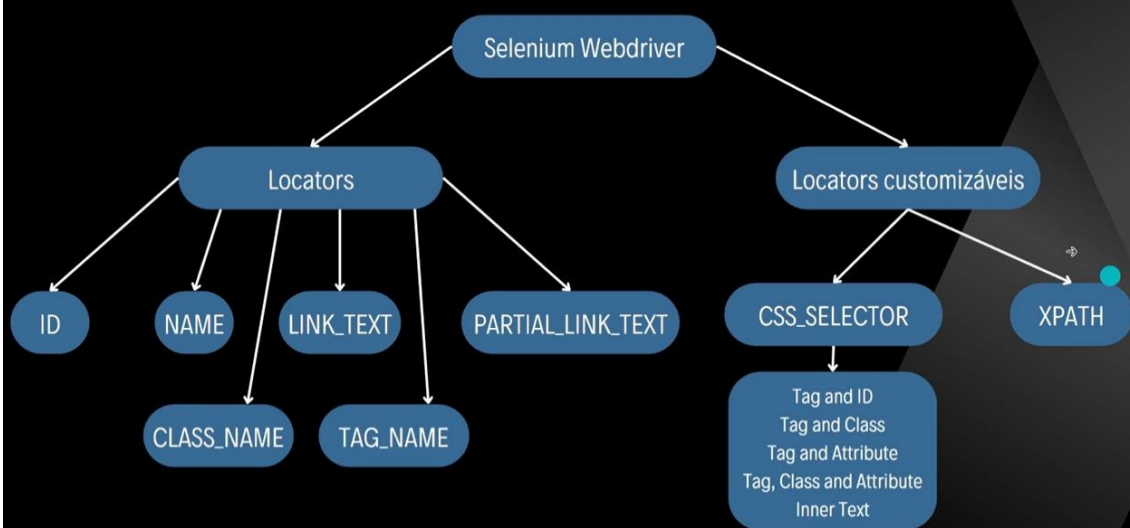
Locators

Tag Atributo Valor Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

Estrutura de um elemento

Tipos de Locators



O que é um locator? Para os novatos de plantão, um locator (localizador) é o nome que damos para um código que serve como endereço de um elemento na página web. Através desse locator, conseguimos fazer com que um framework de


automação web como Selenium, Cypress, Robot, etc., encontre determinado elemento da página

Locators

Tag Atributo Valor Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.ID
`browser.find_element(By.ID, "password")`




No exemplo acima, o comando `browser.find_element(By.ID, "password")` vai procurar na página o elemento ID password. By é uma classe.

Locators

Tag Atributo Valor Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.NAME
`browser.find_element(By.NAME, "password")`



Outro exemplo, nesse caso o elemento é o NAME.

Locators

href: atributo de link

Link text

```
<a href="https://twitter.com/saucelabs" target="_blank" rel="noreferrer">Twitter</a>
```

By.LINK_TEXT

```
browser.find_element(By.LINK_TEXT, "Twitter")
```

No exemplo acima a palavra twitter vira um link com o uso da tag em html href. O elemento é o Link_text.

Locators

href: atributo de link

Link text

```
<a href="https://twitter.com/saucelabs" target="_blank" rel="noreferrer">Twitter</a>
```

By.PARTIAL_LINK_TEXT

```
browser.find_element(By.PARTIAL_LINK_TEXT, "Twi")
```

Acima uma outra forma de link com texto parcial.

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.CLASS_NAME

Encontra a primeira ocorrência

```
browser.find_element(By.CLASS_NAME, "input_error form_input")  
browser.find_elements(By.CLASS_NAME, "input_error form_input")
```

Retorna todas as ocorrências



Acima dois outros locators (Procura o nome da classe) que veremos na prática seus funcionamentos.

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.TAG_NAME

Encontra a primeira ocorrência

```
browser.find_element(By.TAG_NAME, "input")  
browser.find_elements(By.TAG_NAME, "input")
```

Retorna todas as ocorrências



Outro locator que procura o nome do input, no exemplo acima provavelmente há o input de login e o input de senha na tela.

Locators

Tag Atributo Atributo Valor

↓ ↓ ↓ ↓

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name#id_value (tag é opcional)

```
browser.find_element(By.CSS_SELECTOR, "input#password")
```

```
browser.find_element(By.CSS_SELECTOR, "#password")
```



Esse locator é de CSS, e acima é mostrada as duas maneiras de utiliza-lo com ou sem a tag (no caso input), usa-se o # para separar a tag do valor.

Locators

Tag Atributo Atributo Valor

↓ ↓ ↓ ↓

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name.class_value (tag é opcional)

```
browser.find_element(By.CSS_SELECTOR, "input.input_error")
```

```
browser.find_element(By.CSS_SELECTOR, ".input_error")
```



Esse locator acima usa . como separador, novamente é opcional usar a tag (input no caso) input_error é o valor da classe.

Locators

Tag Atributo Atributo Valor

```
<input class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

By.CSS_SELECTOR

tag_name[attribute=value] (tag é opcional)

```
browser.find_element(By.CSS_SELECTOR, "input[type=password]")
```

```
browser.find_element(By.CSS_SELECTOR, "[type=password]")
```

No csso acima é usado o atributo do valor e o valor, no caso o type (atributo) e o valor que é password (valor) utilizando como separador o [].

Locators

Tag Atributo Atributo Valor

```
{input} class="input_error form_input" placeholder="Password" type="password" data-test="password" id="password" name="password" autocorrect="off" autocapitalize="none" value fdprocessedid="84hy06">
```

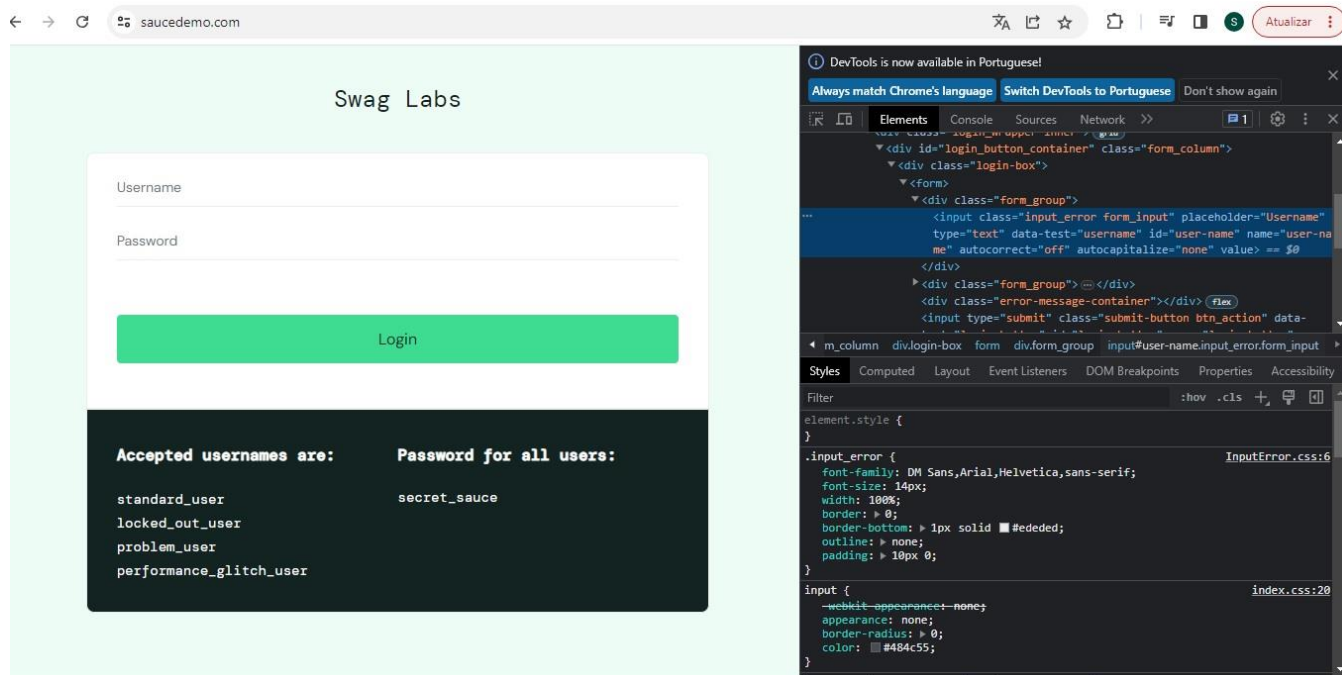
By.CSS_SELECTOR

tag_name.class_name[attribute=value]

```
browser.find_element(By.CSS_SELECTOR, "input.input_error[type=password]")
```

No caso acima é usado a tag com o nome da classe, atributo e valor.

Na sequencia vamos acessar o site <https://www.saucedemo.com/> e inspecionar ele, clicando com o botão direito do mouse e depois clicar em "Inspecionar", ou pressionar a tecla F12, abaixo a página acessada:



Acima em destaque a inspeção do campo Username. Para fazer uma busca no código basta pressionar CTRL + F e digitar o locator como por exemplo:

Input#user-name

Dessa forma ele vai buscar o ID, lembrando que poderia ser sem o Input também:

#user-name

Outro locator que podemos localizar é o **input.form_input** para achar uma classe, nesse site vai achar 2 locators o do campo username e o do campo password.

Observação: caso queira achar apenas um elemento (no caso Username) durante a busca temos que ser mais específicos e digitar algo a mais além da classe, como no exemplo abaixo:

input.form_input[placeholder=Username]

O próximo exemplo é achar o atributo, lembrando que pode ser encontrado sem a tag input, assim:

[data-test=username]