Automação com Selenium usando Python parte 2

Simone Dos Santos
Uma outra dúvida, no vídeo vc digita input.form_input e ele acha o locator, mas na vida real como a gente vai saber o nome do que precisa buscar? Esse input.form_input por exemplo não foi citado anteriormente, o nome dos locators são padronizados? Existe uma lista deles? Temos que decorar? Como é que é que faz pra saber o que buscar?

Leonardo — Instrutor
Simone, no minuto `3:50` da aula eu mostro de onde tirei o `input.form_input`. Não é padronizado, é o valor contido no atributo `class` do elemento. Dá uma revisada nesse trecho.

De qualquer forma, segue mais uma explicação:

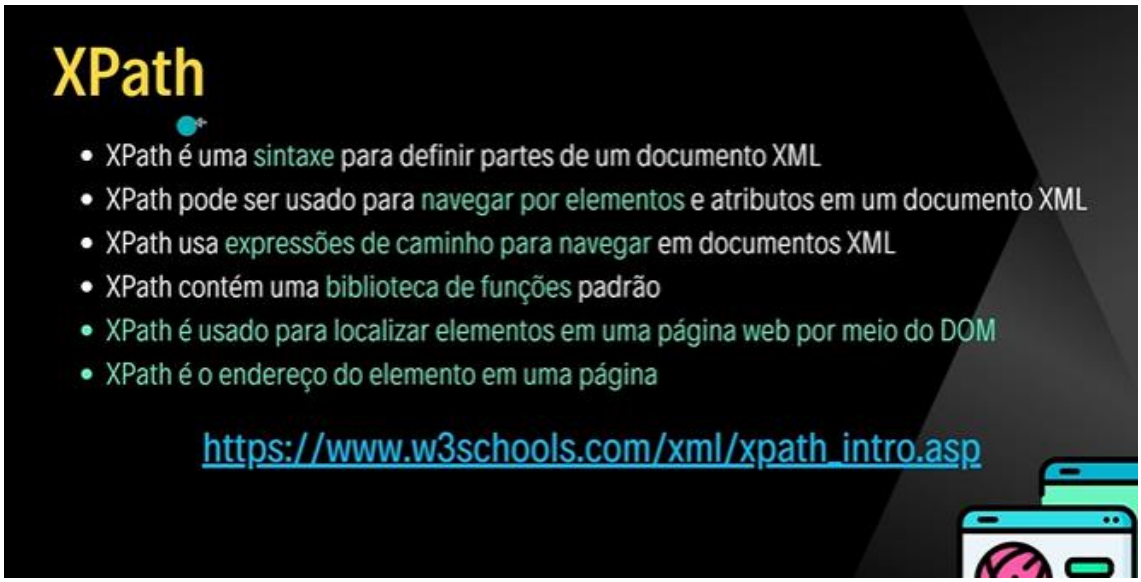
`input` > é a `tag` que estou procurando

`.` > significa que estou procurando pela `class`

`form_input` > é uma das classes do elemento, os valores estão separados por um espaço (input_error **form_input**)

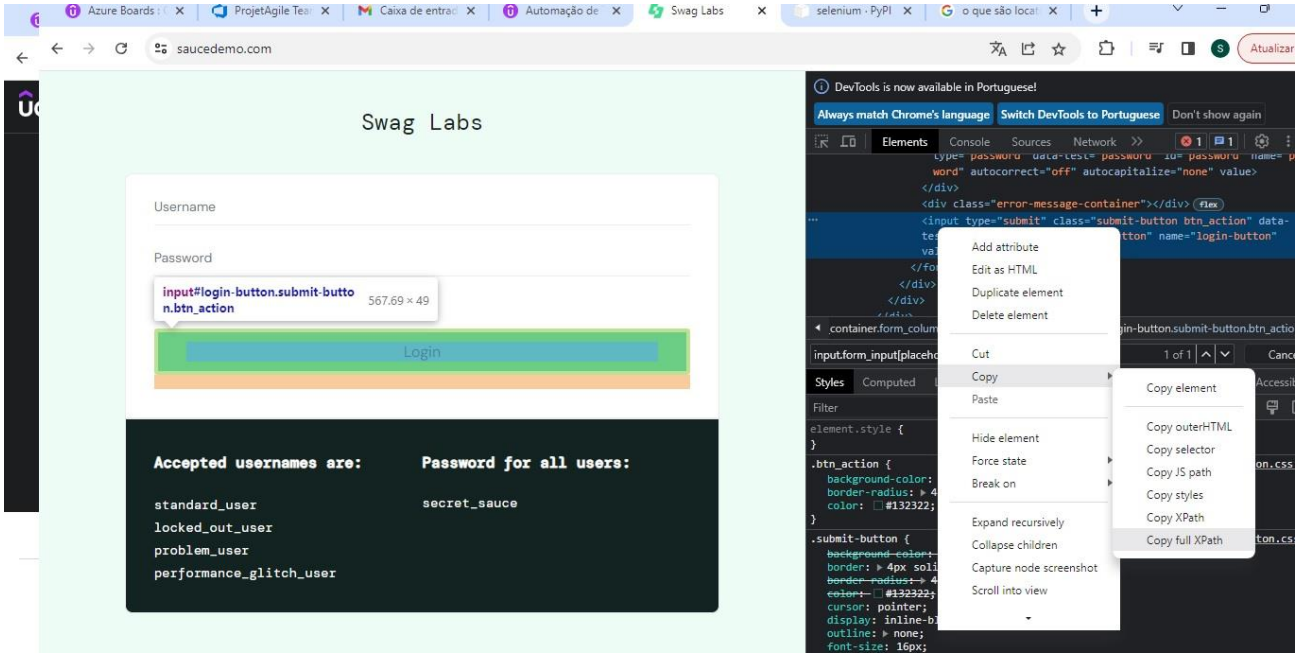
ou seja:

`input.form_input` = me dê todos os elementos onde a `<tag>` é `input` e possui `form_input` dentro do atributo `class`



Xpath é uma forma de navegar pela página HTML (em XML) por meio do DOM. É o endereço do elemento na página.

Existem dois tipos de Xpath, o absoluto e o relativo, iremos ver os dois.



No site https://www.saucedemo.com/ ainda inspecionando, selecione um elemento (no exemplo é o botão de login), o código ficará selecionado, então clique com o botão direito em cima do código em seguida em copy e em seguida clique em Copy full Xpath, ele irá copiar o caminho total do elemento, assim:

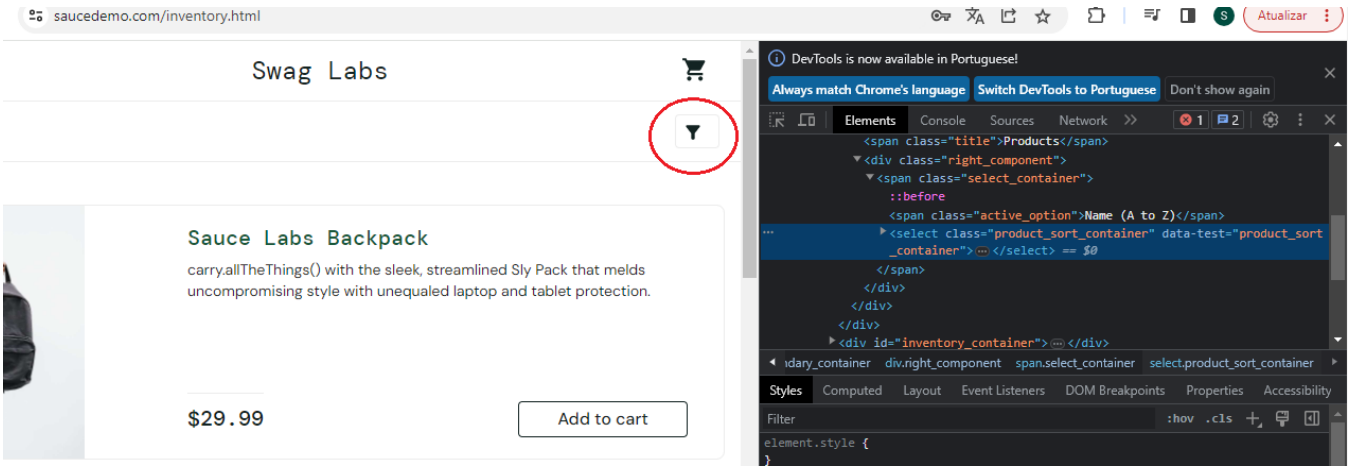
/html/body/div/div/div[2]/div[1]/div/div/form/input

Agora repita o processo mas clique em Copy Xpath, ele irá copiar o caminho parcial do elemento, assim:

//*[@id="login-button"]

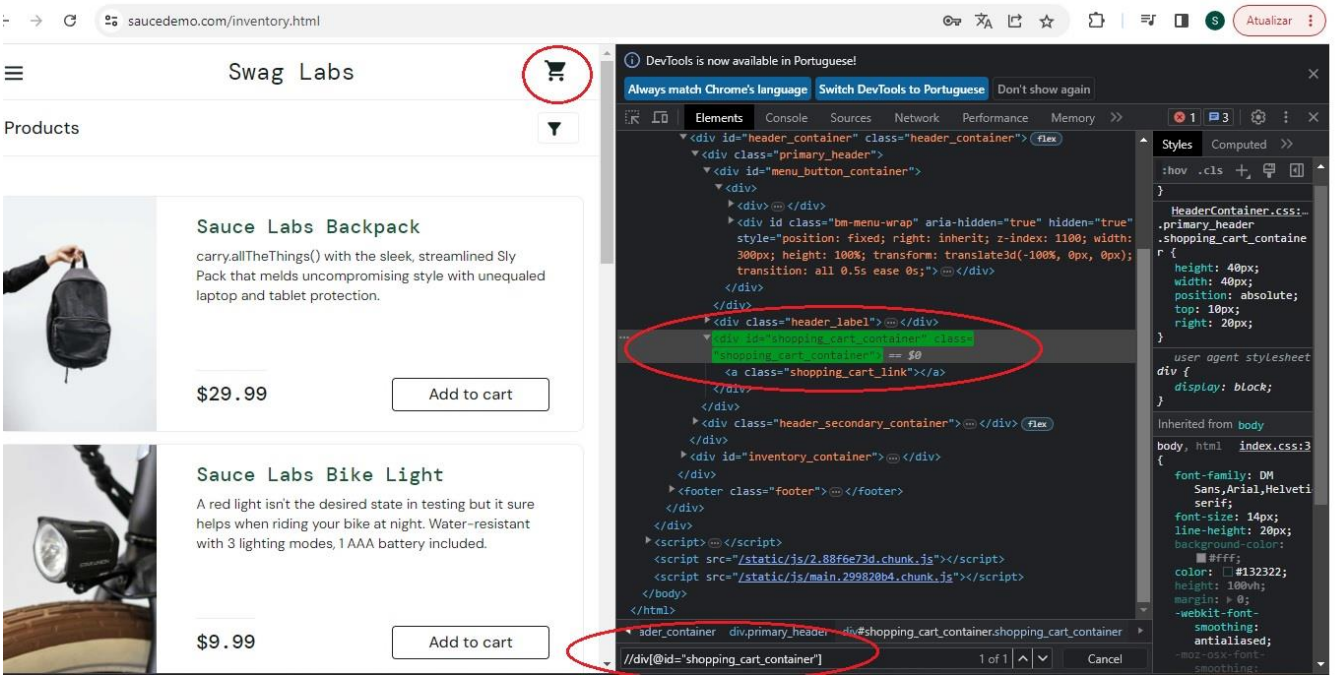
Iremos utilizar a forma relativa Copy Xpath.

Faça login na página https://www.saucedemo.com/ e cliwue com o botão direito do mouse e depois em inspecionar novamente. Dessa vez vamos inspecionar o filtro da página:



Repare que se trata de um Select, se fizer uma busca na página por //select[@class="product_sort_container"] ele irá encontrar o elemento. Outra maneira também seria essa: //*[@class="product_sort_container"]

Agora vamos selecionar outro elemento na página, pode ser o carrinho de compras



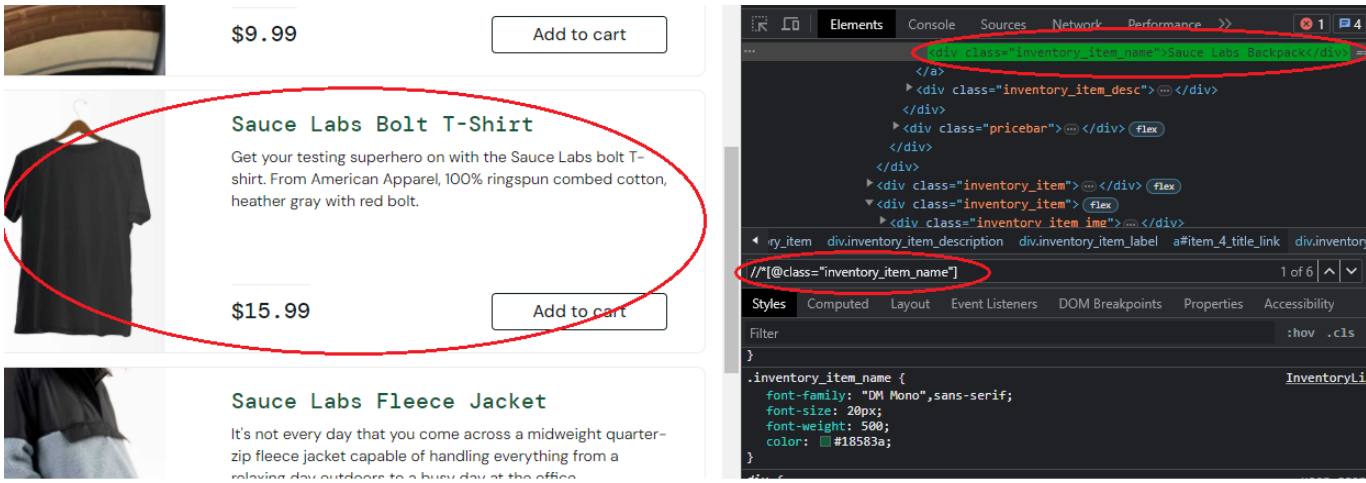
Basta dar um CTRL+ F e digitar a sequência //div[@id="shopping_cart_container"]

Existem várias formar de mapear o mesmo elemento, no exemplo acima ainda poderiamos ser mais específicos adicionando a linha de baixo da tag div que foi encontrada(observe a figura acima), no caso ficaria da seguinte forma:

//div[@id="shopping_cart_container"]/a[@class="shopping_cart_link"

Observação: Portanto Xpath tem ums relação de uma casa sem número de identificação, pra você identificar a casa pra alguém você informa as caracterìsticas da mesma, de forma a identificar a casa. O mesmo ocorre com um elemento sem ID, você navega pelo código e vai criando a identificação.

O próximo elemento é um produto de compra da página, pode ser uma camiseta como mostrado no exemplo:



Observe que esse elemento utiliza uma classe //*[@class="inventory_item_name"] que também é utilizada por mais 6 elementos (observe a imagem acima), nesse caso não serve, nesse caso temos que utilizar outra sequência de comandos, utilizando 'contains' para identificar o elemento:

//*[contains(text(),"Get your testing")]

Acima o comando contains onde se utiliza um trecho da descrição da camiseta para identificar o elemento

Agora utilizando o comando abaixo:

//*[contains(@class, "inventory")]

È feita uma busca no código e traz todas as classes que contém a palavra 'inventory' (com o uso do contains)

Na sequência vamos utilizar o 'And' para unir duas buscas, assim:

//*[contains(@class, "inventory") and contains(text(), "Get your")]

Dessa forma ele vai buscar a classe que usa o texto 'inventory' e também achar o texto que começa com 'Get your'

Ao utilizar o 'Or', ele vai trazer ou um ou outro:

//*[contains(@class, "inventory") or contains(text(), "Get your")]

Observação: Se usarmos a barra | pipe funcionará como um Or visto anteriormente.