

## **Prova: Python e Selenium**

### **Parte 1: Questões Teóricas (5 questões)**

#### **1. Explique a diferença entre Selenium IDE e Selenium WebDriver. (2 pontos)**

##### **Selenium IDE (Integrated Development Environment):**

- **Funcionalidade Principal:** O Selenium IDE é uma ferramenta de gravação e reprodução (record and playback). Ela permite que o usuário grave suas ações no navegador e depois reproduza essas ações automaticamente. Não requer habilidades de programação para criar scripts, tornando-a ideal para iniciantes ou para testes rápidos.
- **Propósito:** É usado para automação simples de testes em navegadores, criando scripts sem necessidade de programação avançada. Ideal para criar testes rápidos e protótipos, mas com limitações em testes mais complexos.
- **Exemplo de Uso:** Criar um script básico para testar funcionalidades de uma página web sem escrever código manualmente.

##### **Selenium WebDriver:**

- **Funcionalidade Principal:** O Selenium WebDriver é uma biblioteca de automação de testes programática. Ele permite que o usuário escreva scripts de teste utilizando várias linguagens de programação, como Java, Python, C#, etc. WebDriver interage diretamente com o navegador sem a necessidade de uma interface gráfica de usuário (GUI), o que torna a automação mais robusta e flexível.
- **Propósito:** É utilizado para automação de testes complexos e altamente personalizáveis. WebDriver oferece mais controle sobre a interação com o navegador e pode ser integrado em frameworks de testes mais avançados, tornando-o ideal para testes de regressão, integração e outros cenários mais elaborados.
- **Exemplo de Uso:** Criar testes automatizados complexos que envolvem interação dinâmica com a página, como preenchimento de formulários, navegação entre páginas e validação de conteúdos.

**2. Quais são os principais tipos de localizadores (locators) usados no Selenium WebDriver para encontrar elementos na página? Explique dois deles. (2 pontos)**

- **ID:** Localiza elementos na página usando o atributo `id` do elemento HTML. É um dos métodos mais rápidos e confiáveis, pois o atributo `id` deve ser único dentro de uma página, garantindo precisão na identificação do elemento. Este localizador é ideal para elementos com identificadores exclusivos.
- **Class Name:** Identifica elementos com base no atributo `class` do elemento HTML. É útil para localizar elementos que compartilham um estilo ou funcionalidade específica. No entanto, como várias classes podem ter o mesmo nome em uma página, esse localizador pode retornar múltiplos elementos, exigindo atenção ao contexto do uso.

Outros tipos de localizadores incluem: Name, Tag Name, Link Text, Partial Link Text, CSS Selector, e XPath, cada um com finalidades e níveis de precisão diferentes.

**3. O que é um WebElement no Selenium? Dê um exemplo de como interagir com um WebElement usando Python. (2 pontos)**

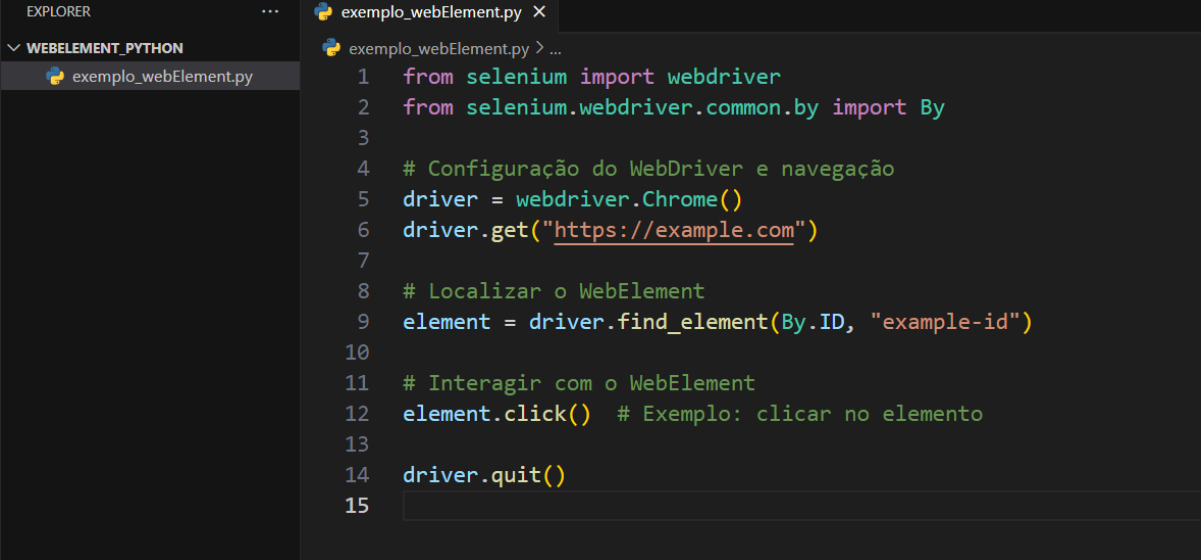
Um WebElement no Selenium representa um elemento HTML individual presente em uma página da web. Ele é um objeto retornado pelo Selenium WebDriver ao localizar um elemento específico na página, permitindo que você interaja com ele, como clicar, enviar texto, ou obter atributos e propriedades.

O **WebElement** é uma abstração usada para manipular elementos como botões, campos de texto, links, etc.

Ele permite interações comuns, como:

- `click()` - clicar em um botão ou link.
- `send_keys()` - enviar texto para campos de entrada.
- `get_attribute()` - obter o valor de um atributo de um elemento.
- `text` - recuperar o texto exibido dentro do elemento.

## Exemplo de Interação com um WebElement em Python: (por ID, "example-id")



```
EXPLORER
  WEBELEMENT_PYTHON
    exemplo_webElement.py

exemplo_webElement.py X
exemplo_webElement.py > ...
1  from selenium import webdriver
2  from selenium.webdriver.common.by import By
3
4  # Configuração do WebDriver e navegação
5  driver = webdriver.Chrome()
6  driver.get("https://example.com")
7
8  # Localizar o WebElement
9  element = driver.find_element(By.ID, "example-id")
10
11 # Interagir com o WebElement
12 element.click() # Exemplo: clicar no elemento
13
14 driver.quit()
15
```

**4. No Selenium WebDriver, o que acontece se você tentar interagir com um elemento que ainda não está visível ou carregado na página? Qual comando você usaria para resolver isso? (2 pontos)**

No Selenium WebDriver, se você tentar interagir com um elemento que ainda não está visível ou não foi completamente carregado na página, ocorrerá uma exceção, como:

- `ElementNotInteractableException`: O elemento está na DOM, mas não está visível ou habilitado para interação.
- `NoSuchElementException`: O elemento ainda não foi carregado na DOM ou não pode ser encontrado pelo localizador.

### Solução:

Para resolver esse problema, é necessário esperar até que o elemento esteja visível ou carregado antes de interagir com ele. O Selenium oferece um mecanismo chamado `WebDriverWait` em conjunto com `ExpectedConditions` para lidar com esses cenários.

Comando sugerido:

O comando `WebDriverWait` permite esperar até que uma determinada condição seja satisfeita, como a visibilidade de um elemento. Ele ajuda a evitar exceções causadas por tentativas de interagir com elementos não prontos.

- Exemplo de condição:
  - `visibility_of_element_located`: Espera até que o elemento esteja visível na página.
  - `element_to_be_clickable`: Espera até que o elemento esteja visível e habilitado para clique.

Esse método reduz a dependência de esperas estáticas (`time.sleep()`), tornando o script mais dinâmico e robusto.

## **5. Cite duas limitações do Selenium IDE que podem levar à escolha do Selenium WebDriver em projetos maiores. (2 pontos)**

### **1. Falta de Suporte para Testes Complexos:**

O Selenium IDE é uma ferramenta de gravação e reprodução simples, projetada para automação básica. Ele não suporta cenários avançados como:

- Interação com múltiplas janelas ou abas.
- Testes condicionais ou loops complexos.
- Integração com frameworks de teste ou sistemas de controle de versão.

Para testes mais robustos e escaláveis, o Selenium WebDriver permite escrita de scripts personalizados em linguagens como Python, Java ou C#, possibilitando lógica avançada e modularização.

### **2. Execução Limitada em Ambientes Diferentes:**

O Selenium IDE é restrito a navegadores compatíveis com sua extensão (Chrome e Firefox). Ele não suporta execução em vários ambientes como

dispositivos móveis ou diferentes tipos de navegadores.

Já o Selenium WebDriver oferece suporte multiplataforma e pode ser integrado a ferramentas como Selenium Grid para execução paralela em diversos navegadores e sistemas operacionais.

Essas limitações tornam o Selenium IDE adequado para prototipagem e testes simples, enquanto o Selenium WebDriver é mais indicado para automação de testes em projetos maiores e mais complexos.