Instruções sobre Selenium

1. **Definição da ferramenta:**

Selenium é uma ferramenta para automação de testes em navegadores web, permitindo a automação de interações do navegador. Ele fornece uma interface para controlar navegadores web e realizar ações como preenchimento de formulários, cliques em botões e navegação, facilitando a criação de testes automatizados para verificar a funcionalidade e a integridade de aplicações web. O Selenium suporta várias linguagens de programação, dentre elas a linguagem Java que será o alvo do nosso estudo.

Para utilizar o Selenium é preciso saber a ordem de funcionamento e os métodos demonstrados a seguir:

2. **Inicializar o Selenium com o Webdriver:**

Para inicializar o WebDriver com o Selenium, primeiro, você precisa importar a biblioteca webdriver do Selenium e escolher um navegador. WebDriver (driver) permite que você interaja com o navegador para automatizar ações, como navegação, preenchimento de formulários e clique em elementos

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

WebDriver driver = new ChromeDriver(); // Substitua 'Chrome' pelo navegador de sua escolha
```

****É importante certificar-se que o driver está configurado para o navegador escolhido.**

3. **Navegar para uma URL:**

A ação de navegar para uma URL é fundamental para iniciar a automação de testes em uma determinada página web. Essa ação é realizada pelo método get() do WebDriver.

```
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

WebDriver driver = new ChromeDriver(); // Substitua 'Chrome' pelo navegador de sua escolha

driver.get("https://www.exemplo.com");
driver.quit();
```

4. Localizar Elementos:

Localizar elementos na página web é uma etapa crucial para interagir com eles, como preenchimento de formulários, clique em botões ou extração de informações**.**

O Selenium fornece vários métodos para localizar elementos, e os mais comuns são baseados nos atributos HTML dos elementos. Para localizar elementos é utilizado o método *findElement()* em Java, passando como argumento os ID, Name, XPath, CSS Selector e ClassName dos elementos a serem investigados.

Por exemplo:

4.1. Por ID:

WebElement elemento = driver.findElement(By.id("id_do_elemento"));

4.2. Por Name:

WebElement elemento = driver.findElement(By.name("nome_do_elemento"));

4.3. Por XPath:

WebElement elemento = driver.findElement(By.xpath("caminho_xpath"));

4.4. Por CSS Selector:

WebElement elemento = driver.findElement(By.cssSelector("seletor_css"));

4.5. Por ClassName:

WebElement elemento=driver.findElement(By.className("nome_da_classe"));

4.6. Por TagName:

WebElement elemento = driver.findElement(By.tagName("nome_da_tag"));

4.7. Por LinkText (para elementos <a>):

WebElement elemento = driver.findElement(By.linkText("texto_do_link"));

4.8. Por PartialLinkText (para elementos <a>):

WebElement elemento=driver.findElement(By.partialLinkText("parte_texto_link"));

5. Interagir com Elementos

Interagir com elementos no Selenium envolve a execução de ações específicas, como preenchimento de campos de texto, cliques em botões, seleção de opções em listas suspensas, entre outras. Abaixo estão exemplos de como interagir com elementos usando o Selenium em Java:

5.1. Preencher um Campo de Texto:

```
WebElement campoUsuario = driver.findElement(By.id("id_do_campo_usuario"));
campoUsuario.sendKeys("Nome de Usuário");
```

5.2. Clicar em um Botão:

```
WebElement botaoLogin = driver.findElement(By.id("id_do_botao_login"));
botaoLogin.click();
```

5.3. Selecionar uma Opção em uma Lista Suspensa (Dropdown):

```
WebElement listaSuspensa = driver.findElement(By.id("id_da_lista_suspensa"));
Select dropdown = new Select(listaSuspensa);
dropdown.selectByVisibleText("Opcao1");
// Ou
dropdown.selectByValue("valor_da_opcao");
// Ou
dropdown.selectByIndex(0);  // Índice da opção na lista
```

5.4. Limpar o Conteúdo de um Campo de Texto:

```
WebElement campoTexto = driver.findElement(By.id("id_do_campo_texto"));
campoTexto.clear();
```

5.5. Enviar um Formulário:

```
WebElement formulario = driver.findElement(By.id("id_do_formulario"));
formulario.submit();
```

6. Funções importantes

6.1. Extrair texto visível do elemento da página - getText()

O método getText() no Selenium é utilizado para extrair o texto visível de um elemento da página web. Esse método pode ser aplicado a elementos como botões, links, parágrafos, entre outros, para recuperar o texto que está sendo exibido na interface do usuário.

```
WebElement elementoComTexto = driver.findElement(By.id("id_do_elemento"));
```

```
// Obter o texto do elemento
String textoDoElemento = elementoComTexto.getText();
```

6.2. Navegar para uma URL específica

Direcionar o navegador para uma nova página.

```
WebDriver driver = new ChromeDriver(); // ou qualquer outro navegador
driver.navigate().to("https://www.exemplo.com");
```

6.3. Maximizar a janela do navegador

Maximizar a janela do navegador para garantir que a janela esteja no tamanho máximo disponível, proporcionando uma visão mais consistente e completa do conteúdo da página.

```
WebDriver driver = new ChromeDriver(); // ou qualquer outro navegador
driver.navigate().to("https://www.exemplo.com");
driver.manage().window().maximize();
```

6.4. Obter o título da página

Para obter o título da página atual carregada no navegador é utilizado o *getTitle()*. O título é a informação que geralmente aparece na barra de título do navegador e fornece uma descrição breve do conteúdo da página.

```
WebDriver driver = new ChromeDriver(); // ou qualquer outro navegador
driver.navigate().to("https://www.exemplo.com");

// Obter o título da página
String tituloDaPagina = driver.getTitle();

// Exibir o título no console
System.out.println("Título da Página: " + tituloDaPagina);
```

7. Finalizar o WebDriver

Finalizar ou encerrar o driver do Selenium é uma prática importante ao finalizar seus testes ou automação para liberar recursos e garantir que o navegador seja fechado adequadamente. O método quit() em Java é utilizado para fechar todas as janelas abertas pelo WebDriver e encerrar a sessão.

```
WebDriver driver = new ChromeDriver(); // ou qualquer outro navegador
driver.quit();
```