**Primeira Aula – Testes Web (Parte 1)**

***Pré-requisitos***

Java instalado. JDK instalado.

***Referenciando Biblioteca do Selenium na Aplicação***Com a aplicação Java criada, é preciso baixar a última versão estável do server standalone do Selenium.

* Depois disso, criar um diretório no projeto, chamado “lib”.
* Em seguida, colar a biblioteca do selenium dentro desse diretório.
* Por último, clicar com o botão direito na biblioteca > build path > Add to build path.

Dessa forma, a biblioteca estará referenciada no projeto e pronto para uso.

***Criando Uma Classe Principal no Projeto***

Criar uma classe no projeto, pressionando CTRL+N e escolhendo a opção “Classe”. Em seguida, basta dar um nome e finalizar.

Depois que a classe for criada, é preciso criar um método principal dentro dela. Basta escrever “main”, pressionar CTRL+SPACE e teclar enter.

***Criando Uma Classe/Método de Testes no Projeto***

Criar uma classe no projeto, pressionando CTRL+N e escolhendo a opção “Classe”. Em seguida, basta dar um nome e finalizar. Pode-se criar um pacote diferente para a classe também.

Na tela que se abrir, dentro da “public class”, escrever @Test e, na linha seguinte, declarar o método. Exemplo: public void inserirUsuario(){}

Ao final do método, fechar o driver de testes com “driver.close();”

***Colocando o Driver do Navegador no Projeto***

Para que o selenium atue sobre um navegador, é preciso referenciar o driver do navegador no projeto.

Para isso, baixe o driver mais atual do navegador desejado (no caso, usarei o Google Chrome), crie um folder no projeto com o nome que desejar e jogue o “.exe” do driver do navegador dentro desse folder.

Em seguida, clique com o botão direito sobre o “.exe” e copie seu caminho. Dentro da “main” do projeto, escrever a seguinte linha de código:

System.setProperty("webdriver.chrome.driver", "AQUI COLAR O CAMINHO");

***Instanciando o Driver do Navegador***

Tomando como base o Google Chrome: WebDriver driver = new ChromeDriver();

***Passando a Página que Deve ser Aberta ao Iniciar os Testes***

driver.get("URL DA PÁGINA");

***Referenciando o Framework de Testes no Projeto***

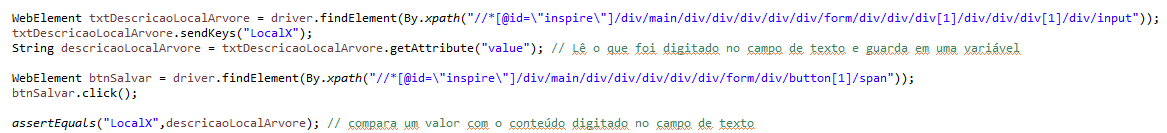
Clicar com botão direito no projeto > build path > Libraries > JUnit e selecionar a versão mais atual.

***Mapeando uma Caixa de Texto Pelo Nome e Escrever Nela***

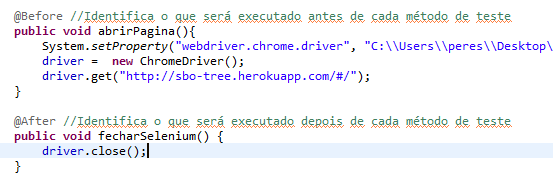
Vale lembrar que, para obter o nome do elemento, basta utilizar a ferramenta de inspeção do navegador e analisar o atributo “name” do elemento. Pode ser utilizado também o ID, a Classe e o Xpath do elemento.



***Comparando se um Valor Esperado Realmente Foi Informado***



***Executando Métodos Antes e Depois de Cada Teste***



***Utilizando Planilha Como Fonte de Dados***

Primeiramente, é preciso baixar os seguintes arquivos .jar (ou versões mais atuais deles):

* dom4j-1.6.1.jar;
* poi-3.10-FINAL-20140208.jar;
* poi-ooxml-3.10-FINAL-20140208.jar;
* poi-ooxml-schemas-3.10-FINAL-20140208.jar;
* xmlbeans-2.3.0.jar;

Existem pacotes que já trazem todos esses arquivos.

Depois de baixar, adicioná-los em um folder do projeto e fazer o “build path” deles.

Em seguida, fazer a importação da planilha .xlsx desejada, através do seguinte código:



Depois, criar a variável que receberá essa planilha:



Por último, criar a variável que apontará para qual “aba” da planilha será utilizada:



Fazer todos os “imports” solicitados pela IDE.

Com tudo configurado e declarado, criar uma variável para receber o conteúdo da célula em questão. No caso, considerando uma string, o seguinte código:



Note que o getRow está passando “i” como parâmetro, pois esse código está em um laço de repetição que toda vez que rodar irá para a linha seguinte da planilha. Para que todas as linhas da planilha sejam percorridas pelo laço, usar o seguinte código:



Depois, basta seguir o processo normal de inserção de registros.

***Pegando Texto de Elementos Inputs e Listboxes***

**Input**: basta pegar o xPath do input e colocar .getAttribute(“value”):

driver.findElement(By.*xpath*("xPathExpression")).getAttribute("value");

**Listbox**: depois de mapear o Listbox para a inserção do dado (sendKeys), é preciso mapear novamente a div exibida com o valor informado, para então depois pegar o texto via .getAttribute(“innerText”);

driver.findElement(By.*xpath*("xPathExpression")).getAttribute("innerText");

***Removendo históricos duplicados do resumo para não exibir mais de uma vez***

Antes de exibir o resumo, basta colocar a seguinte linha:

resumo = **new** ArrayList<>(**new** LinkedHashSet<String>(resumo));

Depois, a exibição pode ser feita normalmente através do “for”.

**for** (**int** i = 0; i < resumo.size(); i++) {

System.***out***.println(resumo.get(i));

}

***Fazendo Scroll up e Scroll down na tela***

Primeiramente, é preciso instanciar o objeto responsável por fazer scroll:

JavascriptExecutor Scrool = (JavascriptExecutor) driver;

Após fazer os imports necessários, basta executar os seguintes comandos:

Para scroll down:

Scrool.executeScript("window.scrollBy(0,300)", "");

Para scroll up:

Scrool.executeScript("window.scrollBy(0,-300)", "");