|  |
| --- |
|  |
| |  | | --- | |  |     **Material do curso de automação – básico** |
| Aula 3 |
|  |
|  |
|  |

**Quadro de Revisão do Documento**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versão** | **Data** | **Autor** | **Descrição** |
| 1.0 | 31/06/2019 | Felipe Amorim | Criação do Documento |

**Lista de Distribuição**

|  |  |  |
| --- | --- | --- |
| **Nome** | **Responsabilidade** | **E-mail** |
| Paulo Luis | N/A | pluis@artit.com.br |
| Roberto Junior Tempesta | N/A | rjtempesta@artit.com.br |
| Murilo Antunes Alves | N/A | maalves@artit.com.br |
| Renato Oliveira | N/A | rvoliveira@artit.com.br |
| Denis Augusto L. de Castro | N/A | dlcastro@artit.com.br |
| Cesar Augusto Cordeiro | N/A | ccordeiro@artit.com.br |
| Estevan Luis Gregori | N/A | egregori@artit.com.br |
| Rubens Gonçalves | N/A | rngoncalves@artit.com.br |

**Sumário**

[Objetivo do Documento 4](#_Toc15538466)

[1. Documentos de Referência 4](#_Toc15538467)

[2. Definições, Acrônimos e Abreviaturas. 4](#_Toc15538468)

[3. Conteúdos 4](#_Toc15538469)

[3.1 Selenium. 4](#_Toc15538470)

[3.1.1 WebDriver. 4](#_Toc15538471)

[3.1.2 Ações do navegador. 6](#_Toc15538472)

[3.1.3 Localizando elementos. 7](#_Toc15538473)

[3.1.4 Ações nos elementos. 9](#_Toc15538474)

[3.1.5 Exceptions. 10](#_Toc15538475)

# Objetivo do Documento

Transmitir o material passado durante a primeira aula do curso de automação de testes – básico.

# Documentos de Referência

N/A

# Definições, Acrônimos e Abreviaturas.

N/A.

# Conteúdos

## Selenium.

### WebDriver.

O selenium é um plugin de comunicação com os navegadores web, para funcionar é preciso um executável do navegador em formato webdriver.

Exemplo:

-chromedriver

-geckodriver (firefox)

-iedriver

etc..

O selenium chama o WebDriver e armazena os valores da sua sessão em um objeto tipo WebDriver, esse objeto é similar a uma matrix 2xn que irá ter todas as informações do navegador e sessão

Exemplo:

'chave da matriz' - 'parametro vindo do webdriver'

-versão do webdriver : 74.256485

-id da sessão : 151344561548

-data de instancia : 31/07/2019/08:10:31:351

etc...

Logo, ao instanciar um navegador, estamos abrindo uma sessão controlada do mesmo navegador utilizado pelo usuário.

Para conseguir abrir uma nova sessão deve se utilizar blocos de código disponíveis na documentação de cada webdriver, disponibilizado pela empresa responsável

Exemplo:

chromedriver -> documentação do google: http://chromedriver.chromium.org/getting-started

iedriver -> documentação da microsoft: https://www.microsoft.com/en-us/download/details.aspx?id=44069

geckodriver -> documentação da firefox: https://github.com/mozilla/geckodriver

OBS: Nesse bloco de código, ao menos uma linha será comum à todos, que é a função 'System.setProperty', este método java não se limita ao selenium, ele serve para definir parâmetros e variáveis globais, como por exemplo qual a encoding de compilação do java, exemplo: UTF-8

Exemplo de implementação do chromedriver

protected static WebDriver navegador = null;

...

...{

System.setProperty("webdriver.chrome.driver", "src/test/resources/drivers/chromedriver.exe");

navegador = new ChromeDriver();

navegador.manage().window().maximize();

}

...

A linha setProperty está definindo um parâmetro global em que, o java consegue acessar o executável do chromedriver, localizado à partir da source do projeto, apontando até o arquivo

OBS: Caso o executável esteja dentro do projeto, é necessário apontar apenas à partir do 'src', porém, nada impede de utilizar um executável fora do projeto, porém, é necesário apontar o seu caminho por completo.

As linhas subsequêntes estão respectivamente criando uma sessão nova do webdriver e maximizando a tela.

OBS: Os parâmetros chamados após o 'new ChromeDriver();' são nativos do selenium, portanto, compartilhados entre todos os webdrivers, como são muitos, é necessário consultar a documentação do selenium para utilizar suas outras funcionalidades.

### Ações do navegador.

Tendo a sessão do navegador criada, podemos enfim executar ações.

OBS: Ao chamar essas linhas, o webdriver é aberto, porém, existem inclusive métodos para executar o webdriver em plano de fundo, a 'interface' exibida ao usuário é apenas um espelho da execução

Existem algumas interações que realizamos à nivel webdriver, as mais comuns são:

.get(String) | .navigate(String)

Descrição: Realiza a navegação para uma determinada página na web

Parâmetro: String que aceita apenas urls válidas

Exemplo: https://www.google.com - válido; google.com - inválido

Retorno: void

Exemplo: navegador.get("https://www.google.com");

.quit()

Descrição: Finaliza a sessão do webdriver e fecha o navegador

Parâmetro: void

Retorno: void

Exemplo: navegador.quit();

OBS: Esse método não transforma o objeto webdriver 'navegador' em null novamente, apenas a sessão dele

.close()

Descrição: Fecha o navegador e mantém a sessão

Parâmetro: void

Retorno: void

Exemplo: navegador.close();

OBS: Esse método não transforma o objeto webdriver 'navegador' em null novamente

OBS2: Não recomendado, por ser passivel à erros de ações quando se utiliza multiplas sessões

### Localizando elementos.

Para conseguir efetuar uma ação, é necessário localizar o elemento alvo (Com qual elemento da web estamos querendo interagir), para isso iremos chamar o objeto navegador e a partir dele, localizar o elemento

Exemplo:

navegador.findElement(By.xpath(MEUXPATH));

OBS: existem n localizadores, porém, todos podem ser capturados pelo xpath (assim como foi tratado na aula 2), então neste curso será utilizado apenas o By.xpath()

Essa linha de código, apresentada no exemplo acima, representa um novo tipo de objeto: WebElement, que possui as suas próprias interações

OBS: Esse objeto não precisa ser instanciado para realizar ações, apenas caso o usuário precise guardar as informações daquele elemento

Exemplo: navegador.findElement(By.xpath(MEUXPATH)); é igual a: WebElement meuElemento = navegador.findElement(By.xpath(MEUXPATH));, com a única diferença que a variável 'meuElemento' irá armazenar os valores daquele elemento, enquanto a não instância do elemento pode ser utilizada apenas para disparar uma ação.

O localizador, pode ser traduzido para um objeto, também nativo do selenium, chamado By, este objeto apenas

By teste;

Descrição: indica qual é o tipo de localizador que o usuário está utilizando

Parâmetro: Tipo de localizador (Objetos nativos do selenium)

Exemplo: By.xpath; By.id; By.className; etc...

Retorno: void

Exemplo:

By teste = By.xpath(MEUXPATH);

By teste = By.id(IDDOELEMENTO);

By teste = By.className(CLASSEDOELEMENTO);

Existem duas funções para localizar o elemento, sendo elas:

.findElement(By)

Descrição: Realiza a busca por elementos na web

Parâmetro: Objeto tipo By (nativo do selenium) que representa o tipo de localizador, no caso, xpath

Exemplo: By meuTipo = By.xpath(MEUXPATH);

Retorno: WebElement (O primeiro elemento no index do localizador, por isso, como mencionado na aula 2, o localizador deve retornar apenas um elemento)

Exemplo: navegador.findElement(meuTipo);

.findElements(By)

Descrição: Realiza a busca por elementos na web

Parâmetro: Objeto tipo By (nativo do selenium) que representa o tipo de localizador, no caso, xpath

Exemplo: By meuTipo = By.xpath(MEUXPATH);

Retorno: List<WebElement> (Retorna a lista completa de elementos que o localizador encontrar)

Exemplo: Caso o index do localizador seja 5, será retornado um List<WebElement> de dimensão 4 (0 a 4, ou seja, 5 elementos)

Exemplo: navegador.findElements(meuTipo);

### Ações nos elementos.

Após ter o elemento desejado localizado, podemos disparar as ações para o webdriver

OBS: Não necessariamente as ações representam a utilização do mouse, o mesmo não se move

Exemplo: Caso um clique seja disparado duas vezes em um campo texto, o mesmo não sofre seleção.

Similaridade: Jquery -> .click();

.click()

Descrição: Realiza um clique no elemento localizado

Parâmetro: void

Retorno: void

Exemplo: navegador.findElements(meuTipo).click();

.sendKeys(String)

Descrição: Realiza um clique no elemento localizado

Parâmetro: String contendo o texto para ser enviado

Retorno: void

Exemplo: navegador.findElements(meuTipo).sendKeys("Exemplo de input");

OBS: Esse método não apaga o campo input antes de enviar o texto, apenas envia

.clear()

Descrição: Limpa o conteúdo de input do elemento localizado

Parâmetro: void

Retorno: void

Exemplo: navegador.findElements(meuTipo).clear();

.getText()

Descrição: Retorna o texto do elemento localizado

Parâmetro: void

Retorno: String

Exemplo:

navegador.findElements(meuTipo).getText();

String textoDoElemento = navegador.findElements(meuTipo).getText();

System.out.println(navegador.findElements(meuTipo).getText()); -> output: <textoDoElemento>

.getAttribute()

Descrição: Retorna o atributo almejado do elemento localizado em formato de string

Parâmetro: String com o nome do atributo almejado

Exemplo: "Value"

Retorno: String

Exemplo:

navegador.findElements(meuTipo).getAttribute(ATRIBUTO);

String atributoDoElemento = navegador.findElements(meuTipo).getAttribute(ATRIBUTO);

System.out.println(navegador.findElements(meuTipo).getAttribute(ATRIBUTO);); -> output: <atributoDoElemento>

### Exceptions.

O selenium pode retornar várias exceptions nativas, uma para cada respectivo problema, porém, assim como no java temos o termo 'Exception' que captura todas as demais exceptions nativas do java, essa não irá capturar a do selenium, caso essa tratativa seja desejada, utiliza-se 'WebDriverException'