BasicCommandWeb (classe)

Classe responsável por por conter comandos que executam interações em páginas web. Os métodos contidos na classe são todos públicos e para utiliza-los, não é necessário passar o WebDriver como parametro.

BasicCommandWeb (construtor)

Método construtor da classe.

Parâmetro: **`WebDriver`** driver

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
```

Retorno: **`void`** método não tem retorno.

mudarAba

Método resposanvel por mudar a aba atual do navegador recebe um **`int`** como parâmetro, as abas são lidas em um array dessa forma, são zero indexadas, ou seja, os números passados como parâmetros vão de 0 à quantidade de abas abertas no momento menos 1.

Parâmetro: **`int`** indiceAba

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.mudarAba(0);
```

Retorno: **`void`** método não tem retorno.

trocarJanela

Método responsável trocar a janela ativa do navegador que recebe uma **`String`** como parâmetro, as janelas são lidas em um array dessa forma, são zero indexadas, ou seja, os números passados vão de 0 à quantidade de janelas abertas no momento menos 1.

> **Parâmetro**: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.trocarJanela("elemento");
```

> **Retorno**: **`void`** método não tem retorno.

encontra

Método responsável por encontrar um elemento no navegador, o método recebe uma **`String`** como parâmetro e busca o elementro através dos seguintes seletores **`id`**, **`name`**, **`tagName`**, **`cssSelector`**, **`linkText`**, **`partialLinkText`**, **`xpath`**.

> **Parâmetro**: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.encontra("elemento");
```

> **Retorno**: **`WebElement`** método retorna o elemento buscado, caso o mesmo exista na página.

borda

Método responsável por localizar um elemento no navegador e destacar o mesmo com um borda vermelha para exibir qual fluxo ou ação está sendo realizado, recebe um **`WebElement`** como parâmetro.

> **Parâmetro**: **`WebElement`** elemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
WebElement menuLink = comBasico.findElement(By.id("menu-
link"));
comBasico.borda(menuLink);
```

Retorno: **`void`** método não tem retorno.

elementoExiste

Método responsável por validar se um elemento existe no *DOM* (Document Object Model), recebe uma **`String`** como parâmetro que é o seletor do elemento.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.elementoExiste("menu-link");
```

Retorno: **`void`** método não tem retorno.

elementoEstaHabilitado

Método responsável por validar se um elemento está habilitado, recebe uma **`String`** como parâmetro que é o seletor do elemento.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.elementoEstaHabilitado("menu-link");
```

Retorno: **`void`** método não tem retorno.

escrever

Método responsável por escrever em elementos que permitam a entrada de dados, recebe 2 parâmetrosdo tipo **`String`**

Parâmetro: **`String`** seletorElemento

Parâmetro: **`String`** texto

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.escrever("input-exemplo", "texto de exemplo");
```

Retorno: **`void`** método não tem retorno.

limpar

Método responsável por limpar um elemnto que recebe entrada de dados, recebe um parâmetro do tipo **`String`**

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.limpar("input-exemplo");
```

Retorno: **`void`** método não tem retorno.

clicar

Método responsável por clicar em um elemento, recebe um parâmetro do tipo **`String`**

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.clicar("exemplo");
```

Retorno: **`void`** método não tem retorno.

clicarElementos

Método responsável por clicar em todos os elementos das lista de elementos passada como parâmetro.

Parâmetro: **`List`** seletorElementos

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
List String elementos = Arrays.asList("elemento1","elemento2");
comBasico.clicarElementos(elementos);
```

Retorno: **`void`** método não tem retorno.

verificarSeRadioEstaMarcado

Método responsável por verificar se um elemento do tipo radio está selecionado recebe uma **`String`** como parâmetro.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.verificarSeRadioEstaMarcado("elemento");
```

Retorno: **`void`** método não tem retorno.

obterTexto

Método responsável por obter o texto de um elemento recebe uma **`String`** como parâmetro.

> **Parâmetro**: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterTexto("elemento");
```

> **Retorno**: **`String`** método retorna o texto obtido do elemento, caso o mesmo seja encontrado.

limparValorComBackspace

Método responsável por limpar um elemento que recebe entrada de dados mas como se fosse um usuário usando a tecla backspace recebe um **`WebElement`** como parâmetro.

> **Parâmetro**: **`WebElement`** elemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
WebElement elemento = comBasico.encontra("elemento");
comBasico.verificarSeRadioEstaMarcado(elemento);
```

> **Retorno**: **`void`** método não tem retorno.

limlimparCampopar

Método responsável por limpar um elemnto que recebe entrada de dados, recebe um parâmetro do tipo **`String`**

> **Parâmetro**: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.limpar("input-exemplo");
```

Retorno: **`void`** método não tem retorno.

pegarValorCss

Método responsável por pegar o valor de uma propriedade CSS de um determinado elemento, recebe 2 parâmetros do tipo **`String`**.

Parâmetro: **`String`** seletorElemento

Parâmetro: **`String`** elementoCss

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.pegarValorCss("elemento-exemplo", "color");
```

Retorno: **`String`** método retorna o valor do atrituto CSS buscado.

obterValorDoElementoAttribute

Método responsável por pegar o valor de uma propriedade de um determinado elemento, recebe 2 parâmetros do tipo **`String`**.

Parâmetro: **`String`** seletorElemento

Parâmetro: **`String`** attribute

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterValorDoElementoAttribute("elemento-exemplo",
"value");
```

Retorno: **`String`** método retorna o valor do atrituto buscado.

verificarSeOcheckBoxEstaMarcado

Método responsável por verificar se um checkbox está marcado recebe um parâmetro do tipo **`String`**.

> **Parâmetro**: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.verificarSeOcheckBoxEstaMarcado("elemento-exemplo");
```

> **Retorno**: **`boolean`** método retorna **true** caso o elemento esteja marcado e **false** caso negativo.

abrirUrl

Método responsável por abrir uma determinada url, recebe um parâmetro do tipo **`String`**.

> **Parâmetro**: **`String`** url

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.abrirUrl("https://www.google.com.br");
```

> **Retorno**: **`void`** método não tem retorno.

navegarUrl

Método responsável por navegar até uma determinada url, recebe um parâmetro do tipo **`String`**.

> **Parâmetro**: **`String`** url

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.navegarUrl("https://www.google.com.br");
```

Retorno: **`void`** método não tem retorno.

validarTituloDoBrowser

Método responsável por validar o titulo browser, recebe um parâmetro do tipo **`String`**.

Parâmetro: **`String`** tituloPagina

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.validarTituloDoBrowser("Google");
```

Retorno: **`void`** método não tem retorno.

validaTexto

Método responsável por validar o o texto de um determinado elemento, recebe um parâmetro do tipo **`String`** e uma parâmetro do tipo **`WebElement`**.

Parâmetro: **`WebElement`** elemento

Parâmetro: **`String`** texto

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
WebElement elemento = comBasico.encontra("elemento-exemplo");
comBasico.validaTexto(elemento, "texto-exemplo");
```

Retorno: **`void`** método não tem retorno.

validarUrlAtual

Método responsável por validar a url atual, recebe um parâmetro do tipo **`String`**.

Parâmetro: **`String`** url

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.validarUrlAtual("https://www.google.com.br");
```

Retorno: **`void`** método não tem retorno.

selecionarComboPorTextoVisivel

Método responsável por selecionar uma opção com combobox através do texto visivel, recebe 2 parâmetro do tipo **`String`**.

Parâmetro: **`String`** seletorElemento

Parâmetro: **`String`** valor

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.selecionarComboPorTextoVisivel("elemento-exemplo",
"texto-exemplo");
```

Retorno: **`void`** método não tem retorno.

obterTextoDaPrimeiraPosicaoDoCombo

Método responsável por retornar o texto de primeira opção de um combobox através do texto visivel, recebe um parâmetro do tipo **`String`**.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterTextoDaPrimeiraPosicaoDoCombo("elemento-
exemplo");
```

Retorno: **`String`** método retorna o texto da primeira opção de um combobox.

obterQuantidadeOpcoesCombo

---

método reponsável por retornar a quantidade de opções de um combobox, recebe uma **`String`** como parâmetro.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterQuantidadeOpcoesCombo("elemento-exemplo");
```

Retorno: **`int`** método retorna a quantidade de opções do combobox

verificarSeExisteOpcaoNoSelect

---

método reponsável por verificar se existe uma opção no combobox, recebe 2 parâmetros do tipo **`String`**

Parâmetro: **`String`** seletorElemento

Parâmetro: **`String`** texto

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.verificarSeExisteOpcaoNoSelect("elemento-exemplo",
"texto-exemplo");
```

Retorno: **`void`** método não tem retorno

desmarcarComboPorTextoVisivel

método responsável por desmarcar um combobox através do texto que está visivel, o mesmo recebe 2 parâmetros do tipo **`String`**

Parâmetro: **`String`** seletorElemento

Parâmetro: **`String`** valor

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.desmarcarComboPorTextoVisivel("elemento-exemplo",
"texto-exemplo");
```

Retorno: **`void`** método não tem retorno.

obterTextosCombo

método reponsável por retornar uma lista com o texto das opções presentes em um combobox, recebe uma **`String`** como parâmetro.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterTextosCombo("elemento-exemplo");
```

Retorno: **`List`** método uma lista do tipo **`String`** com o texto das opções do combobox

envia

método responsável por dar um submit num formulário, recebe um parâmetro do tipo **`String`**.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.envia("elemento-exemplo");
```

Retorno: **`void`** método não tem retorno.

obterTodasAsOpcoesSelecionadasNoCombo

---

método responsável por obter todas as opções selecionadas no combobox, recebe um parâmetro do tipo **`String`**.

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterTodasAsOpcoesSelecionadasNoCombo("elemento-
exemplo");
```

Retorno: **`List`** método retorna uma lista do tipo **`String`** com todas as opções selecionadas num combobox.

entrarFrame

---

método Responsável por alterar o contexto da página para o contexto do iframe, recebe uma parâmetro do tipo **`String`**

Parâmetro: **`String`** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.entrarFrame("elemento-exemplo");
```

Retorno: `void` método não tem retorno.

sairFrame

método Responsável por alterar o contexto do iframe para o contexto da página, não recebe parâmetro.

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.sairFrame();
```

Retorno: `void` método não tem retorno.

aceitarAlerta

método responsável por aceitar um window.alert, não recebe parâmetro.

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.aceitarAlerta();
```

Retorno: `void` método não tem retorno.

obterTextoDoAlerta

método responsável por obter o texto de um window.alert, não recebe parâmetro.

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.obterTextoDoAlerta();
```

Retorno: `String` método retorna o texto do window.alert.

negarAlerta

método responsável por negar um window.alert, não recebe parâmetro.

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.negarAlerta();
```

Retorno: **`void`** método não tem retorno.

escreverNoAlerta

método responsável por inserir um determinado texto em um window.alert, recebe um parâmetro do tipo **`String`**.

Parâmetro: **`String`** conteudo

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.escreverNoAlerta("texto-exemplo");
```

Retorno: **`void`** método não tem retorno.

esperarElemento

método responsável por esperar de forma explicita até que um elemento passe a existir recebe um parâmetro do tipo **`String`** e um parâmetro do tipo **`int`**</code.

Parâmetro: **`String`** seletorElemento

Parâmetro: **`int`** tempo

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.esperarElemento("elemento-exemplo", 10);
```

Retorno: **void** método não tem retorno.

esperarCarregar

---

método responsável por esperar de forma explicita até que um elemento seja carregado recebe 2 parâmetros do tipo **String**.

Parâmetro: **String** seletorElemento

Parâmetro: **String** texto

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.esperarCarregar("elemento-exemplo", "texto-exemplo");
```

Retorno: **void** método não tem retorno.

esperarSerInvisivel

---

método responsável por esperar de forma explicita até que um elemento esteja invisivel recebe um parâmetros do tipo **String** e um parâmetro do tipo int.

Parâmetro: **String** seletorElemento

Parâmetro: **int** tempo

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.esperarSerInvisivel("elemento-exemplo", "texto-
exemplo");
```

Retorno: **void** método não tem retorno.

esperarElementoSerVisivel

método responsável por esperar de forma explicita até que um elemento esteja visivel recebe um parâmetros do tipo **String** e um parâmetro do tipo int.

Parâmetro: **String** seletorElemento

Parâmetro: **int** tempo

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.esperarElementoSerVisivel("elemento-exemplo", "texto-exemplo");
```

Retorno: **void** método não tem retorno.

navega

método responsável por navegar até uma determinada página, recebe um parâmetro do tipo **String**

Parâmetro: **String** url

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.navega("https://www.google.com.br");
```

Retorno: **void** método não tem retorno.

navega

método responsável por encerrar o navegador, não recebe parâmetro.

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.encerra();
```

Retorno: **void** método não tem retorno.

navega

---

método responsável por encerrar o navegador, não recebe parâmetro.

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.encerra();
```

Retorno: **void** método não tem retorno.

clicarPorTexto

---

método responsável por clicar em um elemento através do texto, tag e seletor de um elemento, recebe 3 parâmetros do tipo **String**.

Parâmetro: **String** seletorElemento

Parâmetro: **String** tag

Parâmetro: **String** texto

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.clicarPorTexto("exemplo-teste", "h1", "texto-
exemplo");
```

Retorno: **void** método não tem retorno.

rolarParaBaixo

---

método responsável por mover o scroll para baixo até um determinado ponto da página, recebe um parâmetro do tipo **int**

Parâmetro: **int** posicao

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.rolarParaBaixo(100);
```

Retorno: **void** método não tem retorno.

rolarParaCima

---

método responsável por mover o scroll para cima até um determinado ponto da página, recebe um parâmetro do tipo **int**

Parâmetro: **int** posicao

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.rolarParaCima(100);
```

Retorno: **void** método não tem retorno.

moverParaOelemento

---

método responsável por mover o scroll da página até um determinado elemento na página usando javascript, recebe um parâmetro do tipo **By**

Parâmetro: **By** seletorElemento

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.moverParaOelemento("exemplo-teste");
```

Retorno: **void** método não tem retorno.

escrever

---

método responsável por escrever em um elemento que recebe entrada de dados usando javascript, recebe 2 parâmetros um do tipo **By** e um do tipo **String**.

Parâmetro: **By** seletorElemento

Parâmetro: **String** texto

Ex:

```
BasicCommandWeb comBasico = new BasicCommandWeb(driver);
comBasico.escrever("exemplo-teste", "texto-exemplo");
```

Retorno: **void** método não tem retorno.

TestLink (classe)

Classe responsável pela comunicação via rest com o TestLink.

TestLink (construtor)

Responsável por criar um objeto que se conecta ao TestLink

Ex:

```
TestLink testLink = new TestLink();
```

Retorno: **void** método não tem retorno.

obtemProjetoPorNome

Responsável por obter o projeto com base no nome.

Parâmetro: **String** nomeProjeto

Ex:

```
testLink.obtemProjetoPorNome("TESTES INDRA 2018");
```

Retorno: **TestProject** Método retorna um objeto do tipo TestProject

obtemSuitesDeTestes

Obtem as suites de testes com base no Id de um projeto encontrado.

Parâmetro: **TestProject** projeto

Ex:

```
testLink.obtemSuitesDeTestes(projeto);
```

Retorno: **TestSuite[]** Retorna um array de suites de testes.

obtemTestesCases

Obtem os Test Cases a partir de uma determinada Suite de Testes

Parâmetro: **TestSuite** teste

Ex:

```
testLink.obtemTestesCases(testSuite);
```

Retorno: **TestCase[]** um array com os casos de testes.

obtemStepsTratarParaconteudoGherkin

Apos receber os steps este método trata os textos para que possam ser escritos no formato scenarios

Parâmetro: **List** steps

Ex:

```
testLink.obtemStepsTratarParaconteudoGherkin(listaSteps);
```

Retorno: **void** método não possui retorno.

escreveFeatures

Este método criará os arquivos ".feature" já na pasta "src/test/resources". Não deve ser usado de forma direta. Deve ser usado dentro de outro método capaz de buscar os casos de Testes do TestLink.

Ex:

```
escreveFeature();
```

Retorno: **void** método não possui retorno.

gerarTodosOsscenariossDoProjeto

Método responsável por acessar um projeto e obter todos os casos de testes, passando por todos os nível (suites, test cases, etc).

Parâmetro: **String** nomeDoProjeto

Ex:

```
testLink.gerarTodosOsscenariossDoProjeto(projeto);
```

Retorno: **void** Embora não possua retorno, este método gerará os arquivos ".feature" nas suas respectivas pastas..

gerarTodosOsscenariossDoProjeto

Método responsável por acessar um projeto e obter todos os casos de testes com base numa suite especifica, passando por todos os nível (suites, test cases, etc).

Parâmetro: **String** nomeDoProjeto

Parâmetro: **String** suiteEspecificada

Ex:

```
testLink.gerarscenariosAPartirDeUmaSuite(projeto);
```

Retorno: **void** Embora não possua retorno, este método gerará os arquivos ".feature" nas suas respectivas pastas.