DSL para acelerar o desenvolvimento de automação de testes web.

Público-alvo/usuários: desenvolvedores de testes automatizados web utilizando selenium e java.

Grupo: Victor Hugo Freire Ramalho

O problema

Os testes automatizados web por vezes é muito verboso e **inelegível**, de difícil compreensão que leva a criação de vários métodos e padrões de projeto para torná-lo mais “legível” e menos verboso. Essa situação fica ainda pior quando visualizamos projetos em português.

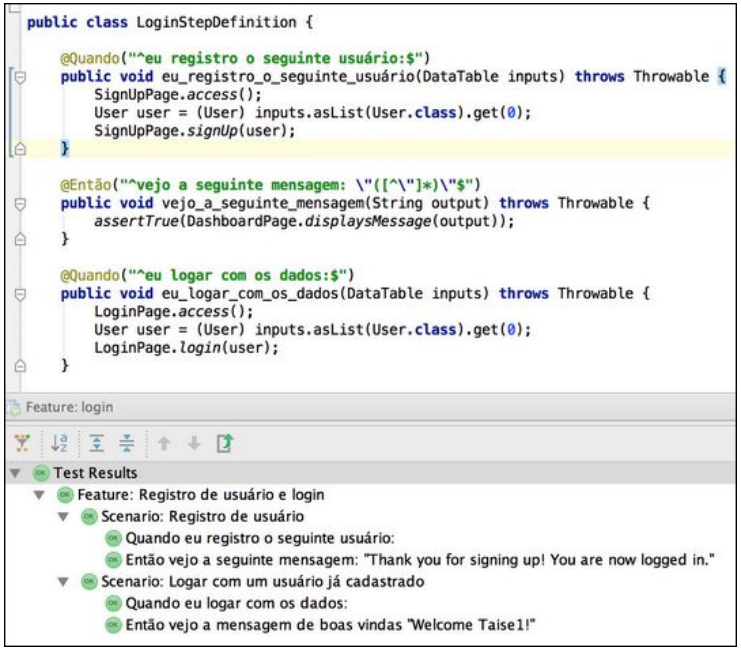
```
public class PerfilPage extends AbstractTest {

	public void acessarPerfil(String usuario) throws InterruptedException {
		WebElement inputBusca = driver.findElement(By.xpath("//input[@placeholder='Busca']"));
		wait.until(ExpectedConditions.elementToBeClickable(inputBusca));
		try {
			inputBusca.click();
		}catch(ElementClickInterceptedException e ) {
			Thread.sleep(3000);
			actions.click(inputBusca).perform();
		}
		inputBusca.clear();
		inputBusca.sendKeys(usuario);

		wait.until(ExpectedConditions.elementToBeClickable(By.xpath("//a[@href='/"+usuario+"/'][.//span[contains(text(), '"+usuario+"')]]")));
		actions.click(driver.findElement(By.xpath("//a[@href='/"+usuario+"/'][.//span[contains(text(), '"+usuario+"')]]"))).perform();

	}
```

Soluções sem usar a DSL proposta



```
public class LoginStepDefinition {

    @Quando("^eu registro o seguinte usuário:$")
    public void eu_registro_o_seguinte_usuário(DataTable inputs) throws Throwable {
        SignUpPage.access();
        User user = (User) inputs.asList(User.class).get(0);
        SignUpPage.signUp(user);
    }

    @Então("^vejo a seguinte mensagem: \"([^\"]*)\"$")
    public void vejo_a_seguinte_mensagem(String output) throws Throwable {
        assertTrue(DashboardPage.displaysMessage(output));
    }

    @Quando("^eu logar com os dados:$")
    public void eu_logar_com_os_dados(DataTable inputs) throws Throwable {
        LoginPage.access();
        User user = (User) inputs.asList(User.class).get(0);
        LoginPage.login(user);
    }
```

Feature: login

Test Results
- Feature: Registro de usuário e login
  - Scenario: Registro de usuário
    - Quando eu registro o seguinte usuário:
    - Então vejo a seguinte mensagem: "Thank you for signing up! You are now logged in."
  - Scenario: Logar com um usuário já cadastrado
    - Quando eu logar com os dados:
    - Então vejo a mensagem de boas vindas "Welcome Taise1!"

Perceba: o desenvovedor não deixou de escrever os testes verbosos e inelegíveis, apenas criou uma referência para eles.

Como seria a DSL

Quando navegar pela URL “http://google.com”
E clicar em botaoMenu
E clicar em botaoOpcoes
E clicar em botaoAcervo
E clicar em botaoPesquisa

E escrever no campo de texto campoPesquisa “artigos”

E clicar em botaoPesquisar
E esperar artigoVictor **ficar visivel**
Entao verifico que “artigoVictor” **existe**

Então, gera o seguinte código:

```
public class Teste {

    protected WebDriver driver;

    @Test
    public void presencaArtigo() throws InterruptedException {
        driver.manage().window().maximize();
        driver.get("http://google.com");
        WebElement botaoMenu = driver.findElement(By.id("botaoMenu"));
        botaoMenu.click();
        WebElement botaoOpcoes = driver.findElement(By.id("botaoOpcoes"));
        botaoOpcoes.click();
        WebElement botaoAcervo = driver.findElement(By.id("botaoAcervo"));
        botaoAcervo.click();
        WebElement botaoPesquisa = driver.findElement(By.id("botaoPesquisa"));
        botaoPesquisa.click();

        WebElement campoPesquisa = driver.findElement(By.id("campoPesquisa"));
        campoPesquisa.sendKeys("artigos");
        WebElement botaoPesquisar = driver.findElement(By.id("botaoPesquisar"));
        botaoPesquisar.click();

        assertTrue(driver.getPageSource().contains("artigoVictor"));
    }
}
```

Vantagens/desvantagens da DSL

Vantagens:

- O código fica mais limpo, mais legível e menos verboso.
- Acelera de forma exponencial a escrita dos testes.
- O desenvolvimento fica mais lógico.
- Uso de linguagem natural para a escrita dos testes.

Desvantagens:

- Pode gerar códigos desnecessários
- Pode precisar de manutenção no código gerado visto que cada teste possui particularidades (colocar um wait a mais, fazer mais uma espera específica etc).