

Automação de testes com Selenium WebDriver Ambiente

Antônio Moraes Trindade

<https://about.me/amtrindade>



Família Selenium

O que é o Selenium?

- Selenium é uma ferramenta para automação web. Segundo os próprios criadores não apenas para testes, mas para automatizar atividades repetitivas.
- Selenium é núcleo de inúmeras ferramentas de automação para diversos browsers.

O que é o Selenium?

- Selenium também pode ser considerada uma API ou framework de teste.
- Os módulos do Selenium podem ser divididos em: Selenium IDE, Selenium RC (Remote Control), Selenium Webdriver e Selenium GRID.
- Muito mais sobre isso: www.seleniumhq.com

O que é o Selenium IDE?

- O Selenium IDE é uma ferramenta Open source para construção de testes automatizados funcionais de páginas web;



Selenium IDE

Vantagens:

- Facilidade de utilização;
- Rápida curva de aprendizagem;
- Facilidade na criação de suíte de testes;
- Record and play;
- Não é necessário muito conhecimento em programação;
- Facilidade de instalação;

Desvantagens

- Funciona apenas no Firefox;
- Pouco ou praticamente nenhum reuso de código;
- Dificuldade de manutenção dos scripts;



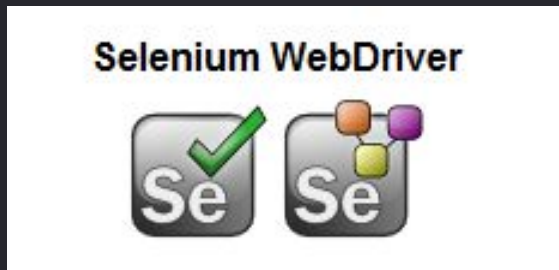
Selenium WebDriver

Selenium WebDriver 4

- **Selenium 1 + WebDriver = Selenium 4**
- **WebDriver:** Projeto criado pelo Google para testes funcionais, mas com algumas melhorias em relação ao Selenium 1.0:
 - Uma API (*Application Programming Interface*) voltada para desenvolvedores;
 - Consistência entre browsers
 - Corrigir funcionalidades mal suportadas pelo Selenium 1.0

WebDriver e suas melhorias

- Dependência do server removida;
- Driver para os browsers independentes;
- Arquitetura em PageObjects;
- Sintaxe mais familiar para desenvolvedores;
- Maior velocidade na execução do teste.





Java + Eclipse

Linguagem de programação + IDE



Maven

- Gerenciador das dependências do projeto Java.
- Responsável por gerenciar dependências, controlar versão de artefatos, gerar relatórios de produtividade, garantir execução de testes, manter nível de qualidade do código dentre outras.

JUnit

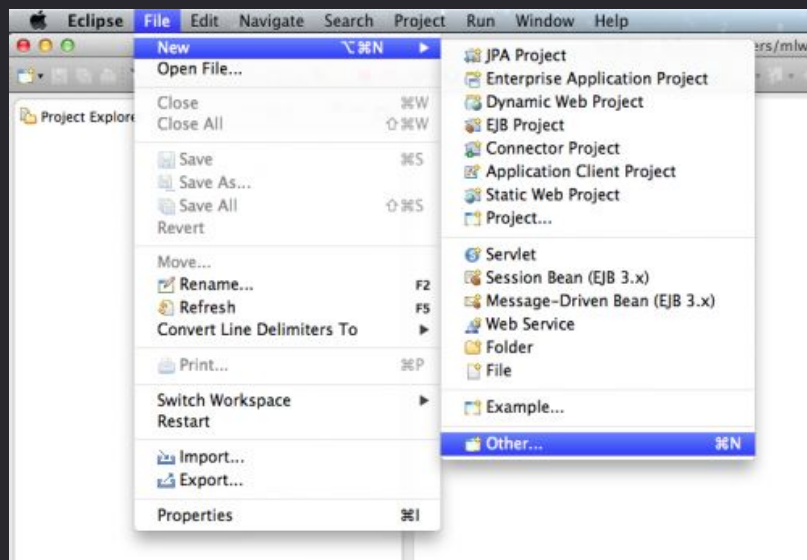
A horizontal line with a diamond-shaped arrow pointing to the right, passing behind the JUnit logo.

JUnit

- Framework para testes unitários e funcionais.
- Responsável por facilitar a criação de código para automação de testes com apresentação de resultados.
- Funciona como player de execução e checkpoint de validações.

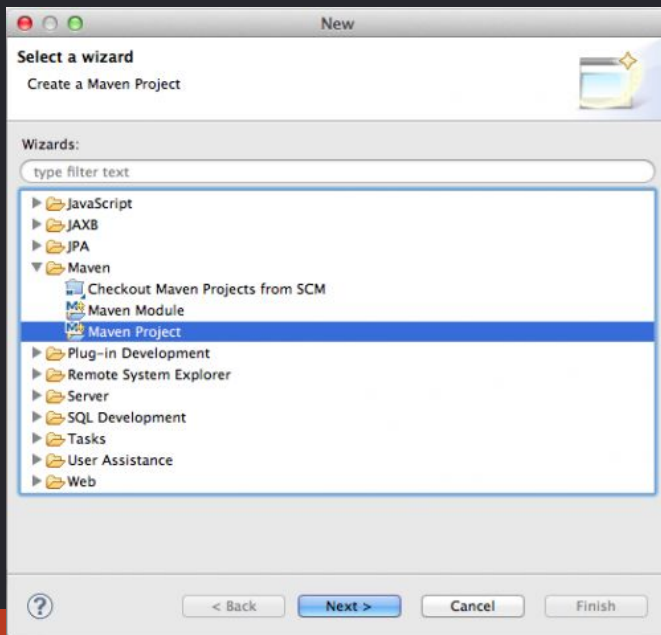
Criar o projeto Maven Java

No Eclipse IDE acesse **File > New > Other...** e um wizard será exibido.



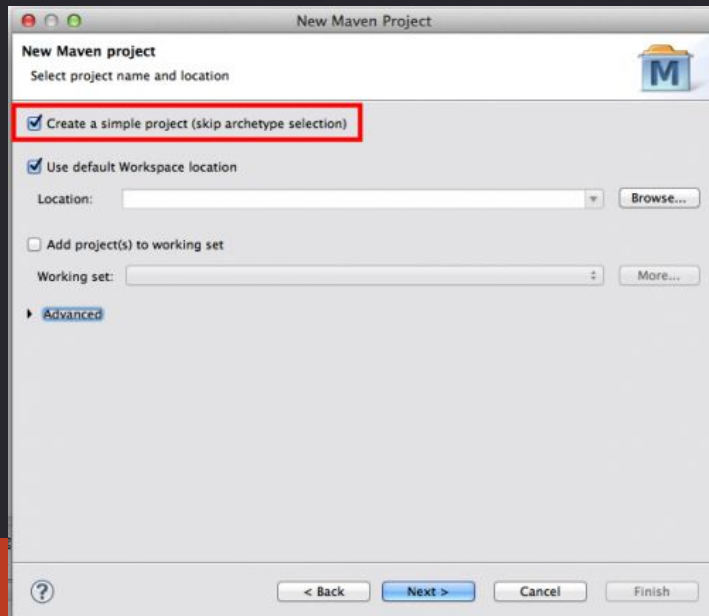
Criar o projeto Maven Java

Selecione a pasta <Maven>, abra e escolha a opção <Maven Project>



Criar o projeto Maven Java

Marque a opção <Create a simple Project (skip archetype selection)> e acione <Next>



Criar o projeto Maven Java

Agora você precisa informar ao projeto alguns parâmetros de configuração como **<Group Id>**, **<Artifact Id>** e **<Name>** e acionar o **<Finish>**

New Maven Project

Configure project

Artifact

Group Id: com.organization.name

Artifact Id: myProject

Version: 0.0.1-SNAPSHOT

Packaging: war

Name: My Project

Description: This project is for giving a demo of Maven capabilities.

Parent Project

Group Id:

Artifact Id:

Version:

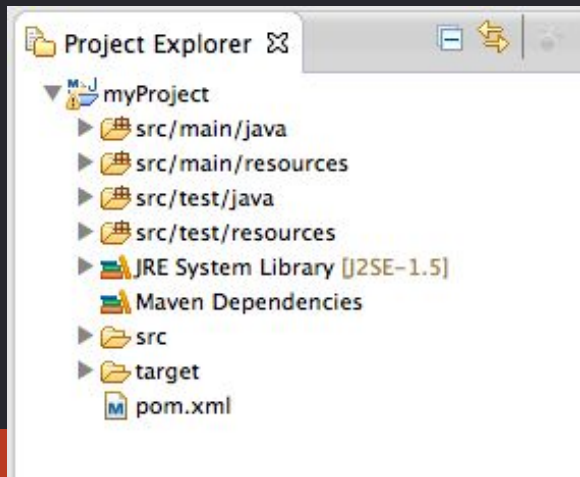
Browse... Clear

Advanced

< Back Next > Cancel Finish

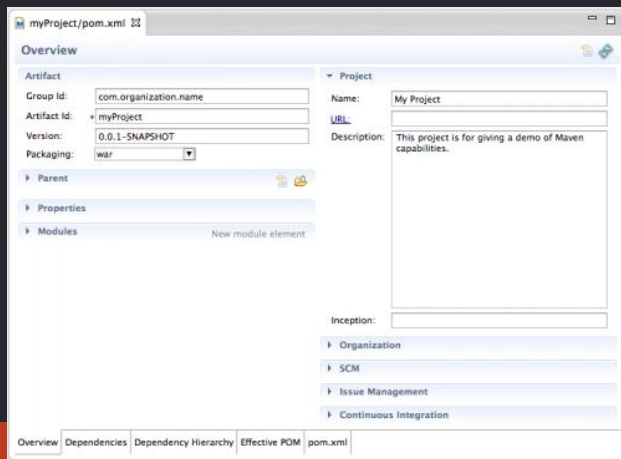
Criar o projeto Maven Java

Você receberá uma notificação que seu projeto foi criado com a mesma estrutura abaixo. O código de teste deverá ser criado em **src/test/java** e **src/test/resource** para as configurações.



Criar o projeto Maven Java

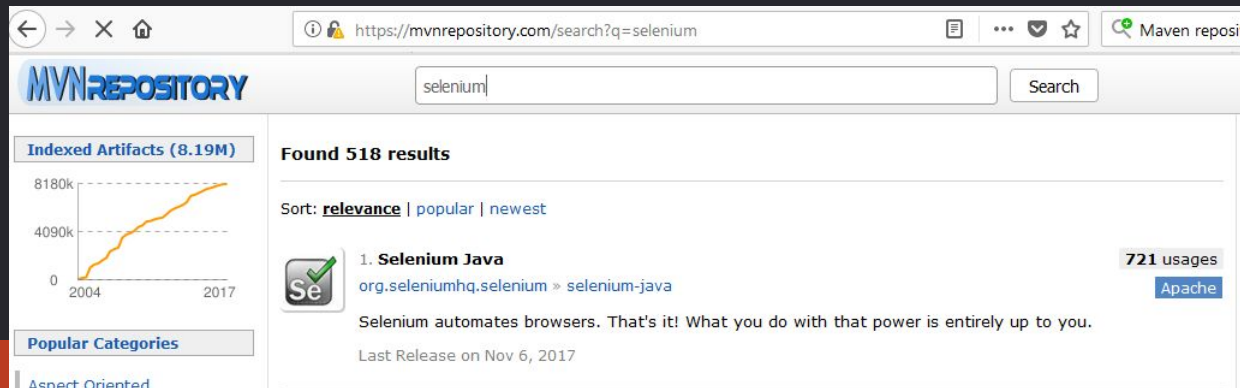
Abra o arquivo **pom.xml** para visualizar a estrutura de um projeto Maven com as suas dependências. Você pode adicionar dependências ao projeto diretamente no XML ou via guia dependencies.



Configurar o WebDriver

Todas as dependências podem ser consultadas no repositório Maven e adicionadas ao projeto.

Pesquise por **Selenium** e copie a configuração Maven da versão mais atualizada para as dependências do pom.xml do seu projeto.



The screenshot shows a web browser window with the URL `https://mvnrepository.com/search?q=selenium`. The page displays the Maven Repository logo and a search bar containing the text "selenium". Below the search bar, there is a section titled "Indexed Artifacts (8.19M)" with a line graph showing the growth of artifacts from 2004 to 2017. The graph shows a steady increase, starting near 0 in 2004 and reaching approximately 8180k by 2017. Below the graph, there is a section titled "Popular Categories" with a link to "Aspect Oriented".

The main content area shows "Found 518 results" and a sort dropdown menu set to "relevance". The first result is "1. Selenium Java" with the URL `org.seleniumhq.selenium » selenium-java`. It includes a description: "Selenium automates browsers. That's it! What you do with that power is entirely up to you." and a note "Last Release on Nov 6, 2017". To the right of the result, there is a badge indicating "721 usages" and a link to "Apache".

Configurar o WebDriver

Copie a configuração do site e adicione ao arquivo **pom.xml** na guia **pom.xml**



Selenium Java » 3.7.1
Selenium automates browsers. That's it! What you do with that power is entirely up to you.

License: [Apache 2.0](#)

Categories: [Web Testing](#)

HomePage: <http://www.seleniumhq.org/>

Date: (Nov 06, 2017)

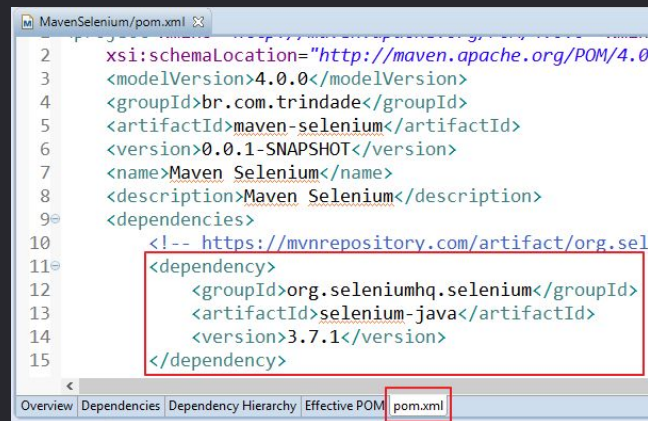
Files: [pom \(3 KB\)](#) | [jar \(293 bytes\)](#) | [View All](#)

Repositories: [Central](#) | [Sonatype Releases](#)

Used By: **721 artifacts**

Maven | Gradle | SBT | Ivy | Grape | Leiningen | Buildr

```
<!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
<dependency>
  <groupId>org.seleniumhq.selenium</groupId>
  <artifactId>selenium-java</artifactId>
  <version>3.7.1</version>
</dependency>
```



```
2  <xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3  </xsi:schemaLocation>
4  <groupId>br.com.trindade</groupId>
5  <artifactId>maven-selenium</artifactId>
6  <version>0.0.1-SNAPSHOT</version>
7  <name>Maven Selenium</name>
8  <description>Maven Selenium</description>
9  <dependencies>
10 <!-- https://mvnrepository.com/artifact/org.seleniumhq.selenium/selenium-java -->
11 <dependency>
12   <groupId>org.seleniumhq.selenium</groupId>
13   <artifactId>selenium-java</artifactId>
14   <version>3.7.1</version>
15 </dependency>
```

Configurar o JUnit

Faça o mesmo para o JUnit:

- Pesquise no Maven Repository por 'JUnit'
- Copie a configuração Maven disponibilizada para o JUnit em sua última versão.
- Adicione ao arquivo pom.xml do projeto nas **<dependencies>**

```
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>test</scope>
</dependency>
```

Projeto criado!
Estão prontos pra
codar???

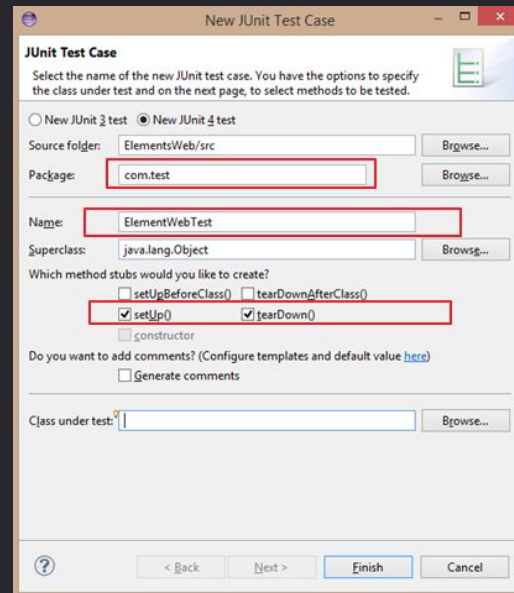


Criar a classe de testes

Na pasta src/test/java clique com o botão direito e selecione New > JUnit Test Case.

Coloque o nome da classe, começando com letra maiúscula e finalizando com o sufixo Test:

Por exemplo: WebElementsTest



Criar a classe de testes

Se as opções setUp e tearDown foram marcadas, a classe criada deve estar semelhante a esta.

O JUnit facilita criando os métodos de pré e pós condições dos testes da classe, realiza o import static do Junit Assert, e cria um método de teste de exemplo com uma falha.

```
1 package com.teste;  
2  
3 import static org.junit.Assert.*;  
4  
5  
6  
7  
8  
9 public class ElementWebTest {  
10  
11     @Before  
12     public void setUp() throws Exception {  
13     }  
14  
15     @After  
16     public void tearDown() throws Exception {  
17     }  
18  
19     @Test  
20     public void test() {  
21         fail("Not yet implemented");  
22     }  
23  
24 }  
25
```


Configurar o setUp

No setUp vamos configurar o driver do browser que iremos utilizar, definir o tempo implícito de espera e indicar qual a URL iremos acessar.

Configurar o tearDown

No método de pós configuração do teste, a única ação que faremos por agora é fechar a instância do browser que está em execução:

Ex. : `driver.quit();`

```
1 package com.test;
2
3 import static org.junit.Assert.assertEquals;
19
20
21 public class ElementWebTest {
22     public WebDriver driver;
23
24     @Before
25     public void setUp() throws Exception {
26
27         System.setProperty("webdriver.chrome.driver",
28             "F:\\\\Drivers\\chromedriver.exe");
29         driver = new ChromeDriver();
30
31         driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
32         driver.get("http://www.treinoautomacao.hol.es/elementsweb.html");
33     }
34
35     @After
36     public void tearDown() throws Exception {
37         driver.quit();
38     }
39 }
```

Prática

Criar nosso primeiro teste



Validar o nome inserido no TextField

Objetivo do teste é inserir seu nome no campo de texto e validar se realmente é o seu nome que está escrito no componente:

- Identificar o componente TextField, pode ser pela propriedade **name**.
- Escrever seu nome no componente, utilizando a função **sendKeys('seu nome')**.
- Faça a validação com um `assertEquals` comparando o resultado esperado com o resultado atual.

Validar o nome inserido no TextField

Implementado o código:

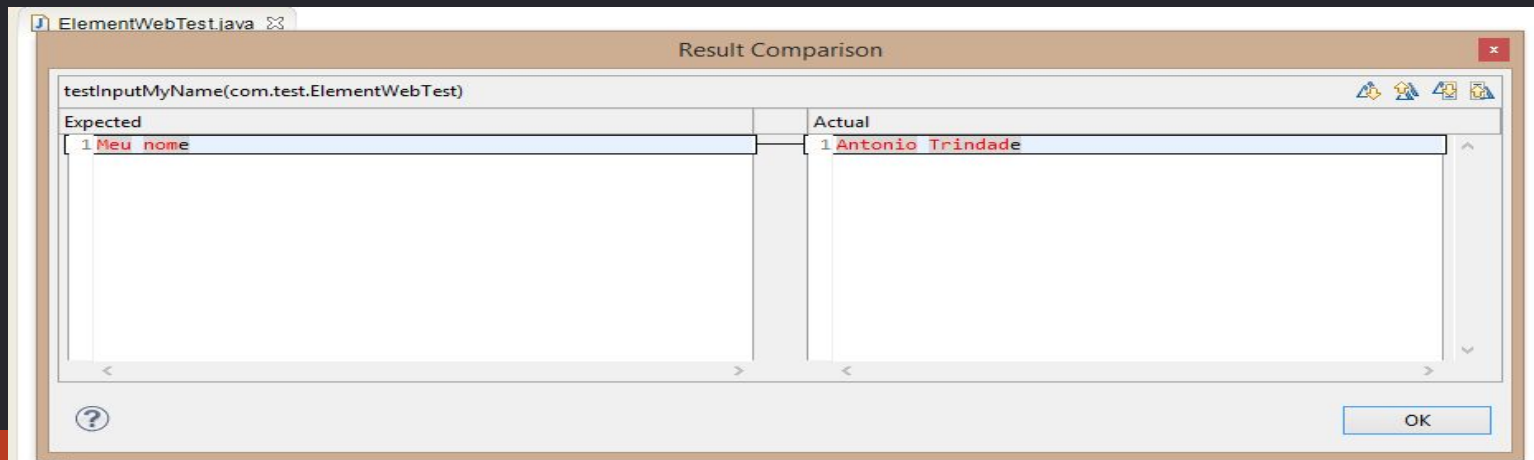
- Para executar o teste posicione o mouse sobre o nome do teste que quer executar, clique com o botão direito e **Run As**

> JUnit Test

```
@Test
public void testInputMyName() {
    driver.findElement(By.name("textbox1")).sendKeys("Meu nome próprio");
```

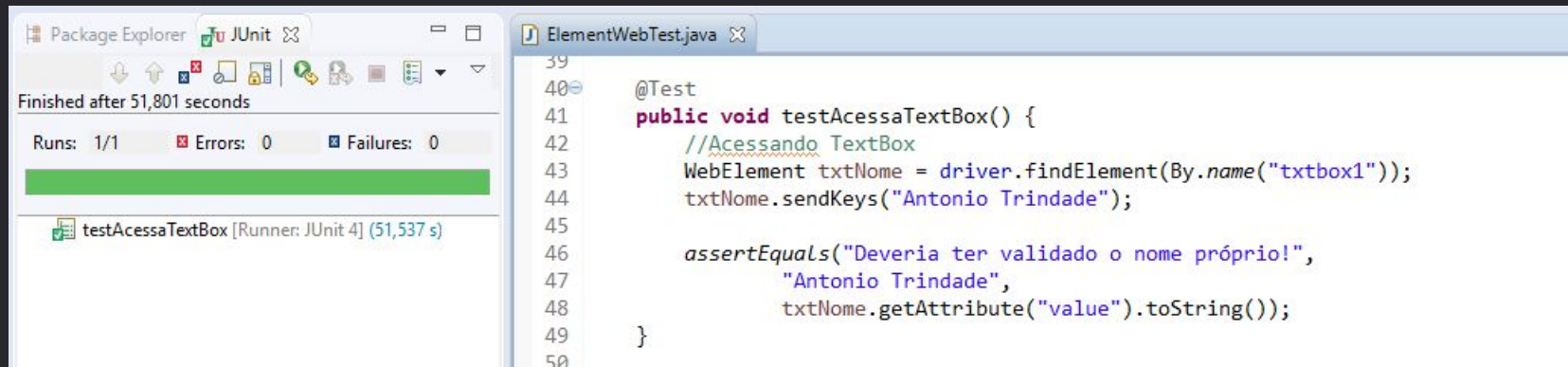
Validar o nome inserido no TextField

Ao executar com o resultado esperado diferente do que informamos, o teste gera uma falha, que pode ser visualizada com um clique duplo sobre ela.



Validar o nome inserido no TextField

Objetivo validado, o teste está sendo executado com sucesso, validando o resultado esperado com o resultado atual. GREEN!

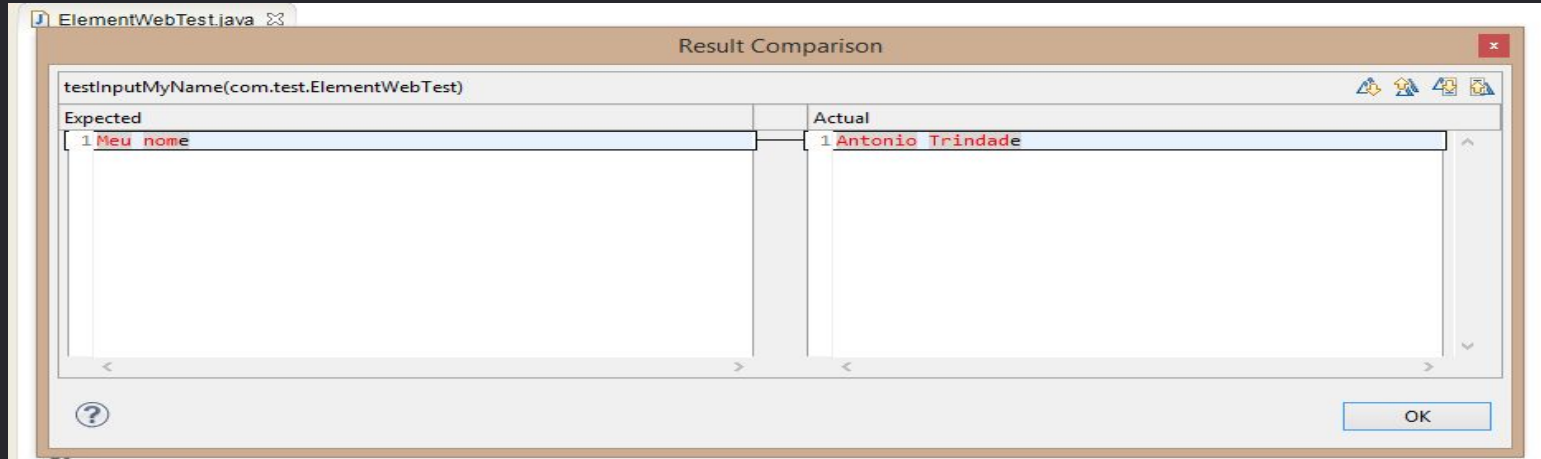


The screenshot displays an IDE interface. On the left, the 'JUnit' tab in the Package Explorer shows a successful test run: 'testAcessaTextBox [Runner: JUnit 4] (51,537 s)'. The status bar indicates 'Finished after 51,801 seconds', 'Runs: 1/1', 'Errors: 0', and 'Failures: 0'. On the right, the 'ElementWebTest.java' file is open, showing the following code:

```
39
40
41 @Test
42 public void testAcessaTextBox() {
43     //Acessando TextBox
44     WebElement txtNome = driver.findElement(By.name("textbox1"));
45     txtNome.sendKeys("Antonio Trindade");
46
47     assertEquals("Deveria ter validado o nome próprio!",
48                 "Antonio Trindade",
49                 txtNome.getAttribute("value").toString());
50 }
```

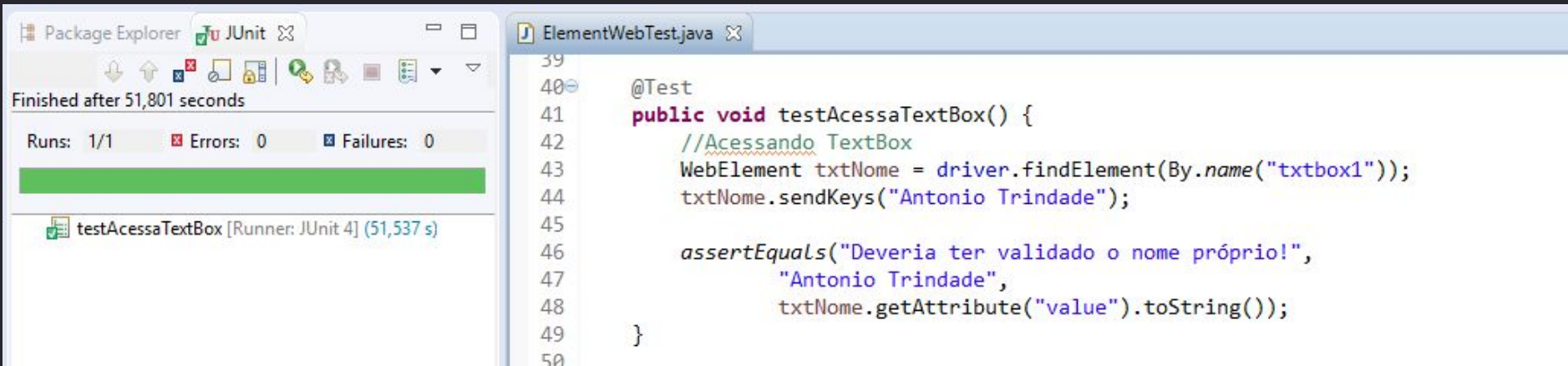

Validar o nome inserido no TextField

Ao executar com o resultado esperado diferente do que informamos o teste gera uma falha, que pode ser visualizada com um clique duplo sobre ela.



Validar o nome inserido no TextField

Objetivo validado, o teste está sendo executado com sucesso, validando o resultado esperado com o resultado atual. GREEN!



The screenshot displays an IDE interface with two main panels. The left panel shows the 'JUnit' test runner results, indicating a successful execution. The right panel shows the source code for 'ElementWebTest.java'.

JUnit Test Results:

- Package Explorer: JUnit
- Finished after 51,801 seconds
- Runs: 1/1
- Errors: 0
- Failures: 0
- testAcessaTextBox [Runner: JUnit 4] (51,537 s)

Source Code (ElementWebTest.java):

```
39
40 @Test
41 public void testAcessaTextBox() {
42     //Acessando TextBox
43     WebElement txtNome = driver.findElement(By.name("textbox1"));
44     txtNome.sendKeys("Antonio Trindade");
45
46     assertEquals("Deveria ter validado o nome próprio!",
47                 "Antonio Trindade",
48                 txtNome.getAttribute("value").toString());
49 }
50
```

Refatorar o teste com WebElements

Com o teste executado e funcionando, é hora de refatorar e deixar ele mais legível e elegante utilizando **WebElements**.

WebElements é um tipo, como um Integer ou String no Java, com ele podemos dar nome aos elementos e componentes que estamos interagindo no teste e usa-los como variáveis no teste.

Refatorar o teste com WebElements

```
WebElement elemento = driver.findElement(By.id("txt-nome"));
```

Chamamos o método **findElement()** do objeto driver (proveniente da interface WebDriver), e passamos como parâmetro de busca um id específico (txt-nome). Dessa forma, o Selenium WebDriver é capaz de varrer toda a estrutura do HTML da página em teste até encontrar o elemento cujo o id seja igual a txt-nome, armazenando-o na variável elemento.

Refatorar o teste com WebElements

Prática: Altere o teste colocando o TextField mapeado em uma variável do tipo WebElement.



Locators



Locators e suas formas

O WebDriver nos possibilita localizar um elemento web de várias formas:

- `By.className` -> `driver.findElement(By.className("name"));`
- `By.cssSelector` ->
`driver.findElement(By.cssSelector("#button"));`
- `By.partialLinkText` ->
`driver.findElement(By.partialLinkText("very"));`
- `By.tagName` -> `driver.findElement(By.tagName("td"));`

Locators e suas formas

- `By.id -> driver.findElement(By.id("id"));`
- `By.linkText -> driver.findElement(By.linkText("acesse"));`
- `By.name -> driver.findElement(By.name("name"));`
- `By.xpath -> driver.findElement(By.xpath("td/span[2]"));`

Locators priorizados por performance

Dê preferência na utilização dos locators, sempre priorizando a velocidade de identificação dos elementos e a velocidade de execução dos testes:

- Por uma propriedade do elemento: “ID” ou “name” ou “link”
- CSS selector
- XPath pelo atributo ou propriedade
- XPath por posição
- DOM

Locators e mais referências

Mais referências para continuar seus estudos a respeito:

Site com documentação oficial do Selenium:

http://www.seleniumhq.org/docs/02_selenium_ide.jsp#locating-elements

Referências interessantes:

<http://www.seleniumeasy.com/selenium-tutorials/selenium-locators>

<https://medium.com/@brunobatista101/aprenda-por-definitivo-a-usar-css-selector-adeus-xpath-1f3956763c2>

Vamos praticar?



RadioButton e CheckBox

Utilizando a lista de elementos, elabore dois novos testes interagindo com os seguintes componentes:

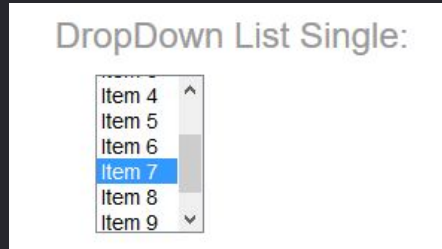
- RadioButton: Clique na 3ª opção e valide que está checked.
- CheckBox: Clique na 3ª e na 4ª opção e valide que estão

```
List<WebElement> elementsRadio = driver.findElements(By.name("radioGroup1"));

for (WebElement e : elementsRadio) {
    System.out.println(e.getAttribute("value").toString());
}
```

DropDown List Single

Elabore um teste que selecione o 7º elemento da lista e valide que este está selecionado.



```
//Dropdown List
WebElement dropdownlist = driver.findElement(By.name("dropdownlist"));
Select listboxelements = new Select(dropdownlist);
```

DropDown List Multi Select

Elabore um teste que selecione o “Item 5”, “Item 8” e “Item 9” simultaneamente validando que os 3 valores estão selecionados.



iFrames

iFrames (inline frames) é um recurso muito utilizado em websites. Consiste na inserção de páginas web dentro de páginas web.

Não confunda com Frames. Frames são divisões da mesma página em seções, já iFrame não, são páginas dentro de páginas.

iFrames

Para navegar entre os iFrames com o WebDriver é necessário direcionar o foco do driver para iFrame que gostaríamos de interagir da seguinte forma:

```
driver.switchTo().frame(1);  
driver.switchTo().defaultContent();  
driver.switchTo().frame("iframe_b");
```


iFrames

Utilizando as funções de navegação entre os iFrames, realize um teste que:

- Escreva o seu nome no campo de text e valide o resultado.

Popups: Alert, Confirm e Prompt

Elabore um teste que faça as validações nos 3 botões existentes na página de WebElements :
assertEquals() deve ser utilizado quando o alerta estiver na tela.

Para confirmar a mensagem utilize o comando accept()

Utilizando POP UP:

Simple Alert Box Message

Alert

Simple Confirm Box Message

Confirm

Simple Prompt Box Message

Prompt

Desafio Calculadora



Desafio Calculadora

Acesse a calculadora no site de treinamento e elabore uma classe de teste com os 5 casos abaixo:

CT1 – Soma

CT2 – Subtração

CT3 – Multiplicação

CT4 – Divisão

CT5 – Divisão por zero

Atenção: Os valores devem ser armazenados em variáveis, e o assert final deve validar o resultado apresentado com a operação realizada pelo script de teste.

