



CAPACITAÇÃO

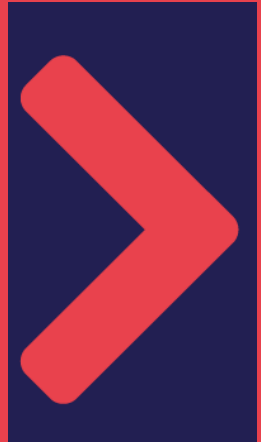
AUTOMAÇÃO WEB

<db>

Agenda:

- O que é Automação de Testes
- Pirâmide de Testes
- Frameworks de Testes Automatizados
- Locators
- Design Pattern
- Esperas (Wait)
- Relatórios de Execução
- Frames e Iframes
- Alerts





O que é Automação de Testes?

- Dentro do projeto de teste de software, existe um processo que é responsável pela automatização desses testes. Essa automatização nada mais é do que a **escrita de uma programa para executar testes** de outro sistema.
- A principal **vantagem da automatização de testes é a economia de tempo e recursos** durante a execução dos testes. Além da possibilidade de executar os mesmos testes repetidas vezes. E, dessa forma, uma vez automatizados os testes, um grande número de casos de testes podem ser validados rapidamente.
- Entretanto, a automação dos testes pode ser cara, e, por isso, diversas vezes é utilizada em conjunto com técnicas manuais. No entanto, **a longo prazo**, o processo de automação de testes gera um grande corte nos custos destinados aos testes.

> Porquê Automatizar ?

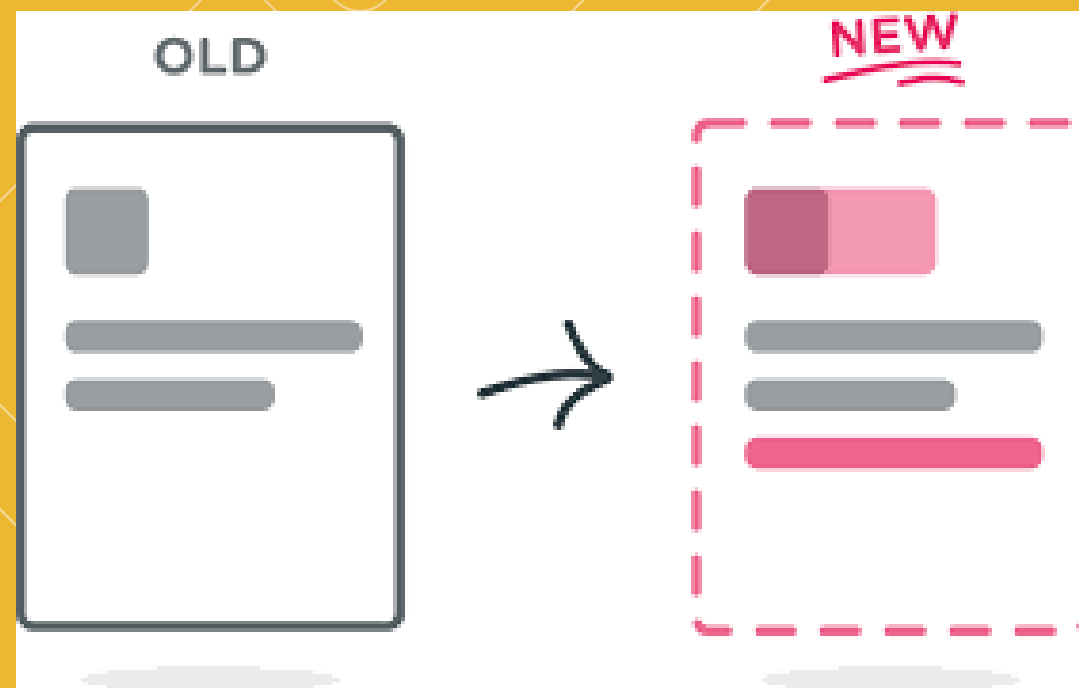


> O que autoomatizar? Por quê?

TESTES DE REGRESSÃO

Esses testes são desenvolvidos, geralmente, de forma vagarosa e precisam ser executados repetidas vezes. Tudo isso os torna fortes candidatos à automação.

- **Testes Unitários**
- **Evitar retrabalho em excesso**
- **Prover Feedback constante desde o início**
- **Funcionalidades críticas para o Cliente**



> Quando Automatizar?

Como identificar quando o teste automatizado pode agregar valor



Estabilidade

Quando os requisitos estiverem estabilizados, com poucas alterações.



Esforço

Quanto mais instável o requisito, maior o esforço da automação.



Mudança

Toda mudança que impacte na funcionalidade possivelmente impactará em uma manutenção nos scripts de automação.

> Quando Automatizar?

- Relacionada a qualidade do produto final.
- Estudar viabilidade , ou seja, conseguiremos obter ganho? Obter ganho de tempo? Reduzir custos manter a qualidade?
- Analisar a maturidade do time no processo de teste.
- Grau de reutilização dos testes automatizados.
- Conhecimento do que é esperado do sistema a ser testado.
- Tempo disponível para automatizar.
- Quão frequente são as mudanças no sistema?
- É viável automatizar a funcionalidade?

> Automação de Testes é **BALA DE PRATA?**

Assim como o bote, a automatização de testes é a ferramenta que ajuda na travessia.....

Se mal implementada, causa prejuízo.

Automação não resolve questões culturais dos times de desenvolvimento



Automação de testes x Semi Automação de testes x Automação de processos

- Automação de testes contém **validações assertivas** e traz feedback da saúde do sistema.
- Teste Automatizado **sem validação** é automação de **Processo**.
- Automação com **interação humana** são testes semi-automatizados.





Pirâmide de Testes

> O que é a Pirâmide de Testes?

Metáfora que agrupa os testes em “baldes” de diferentes **granularidade**.

Martin Fowler, Lisa Crispin

Maior integração

\$\$\$\$

Maior isolamento

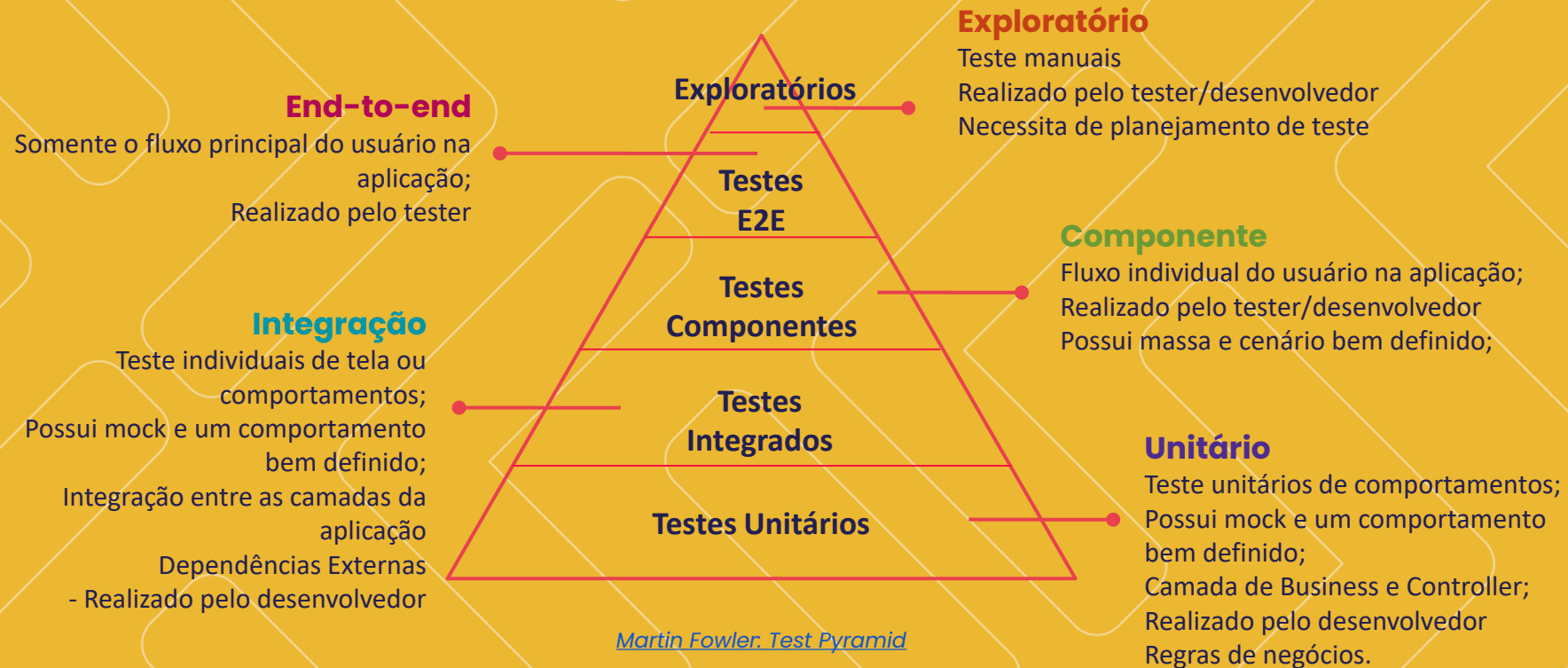
\$

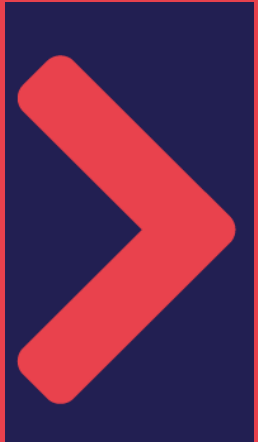


Mais devagar
Escopo Maior
Menos casos escritos

Mais rápido
Escopo Menor
Mais casos de testes

> Camadas da Pirâmide de Testes





Frameworks de Testes

> O que é um framework?

ESTRUTURA
DE CÓDIGO
GENÉRICA

ESTRUTURA

ABSTRAÇÃO

CONJUNTO
DE CLASSES

FUNCIONALIDADES
EM COMUM

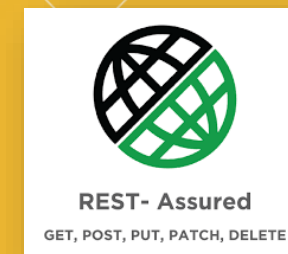
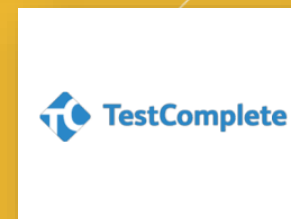
ADICIONAR
TRECHOS DE
CÓDIGO
GENÉRICOS

PROVÊ
FUNCIONALIDADES

FACILITAÇÃO

REUTILIZAÇÃO
CÓDIGO

> Frameworks de Testes



> Selenium



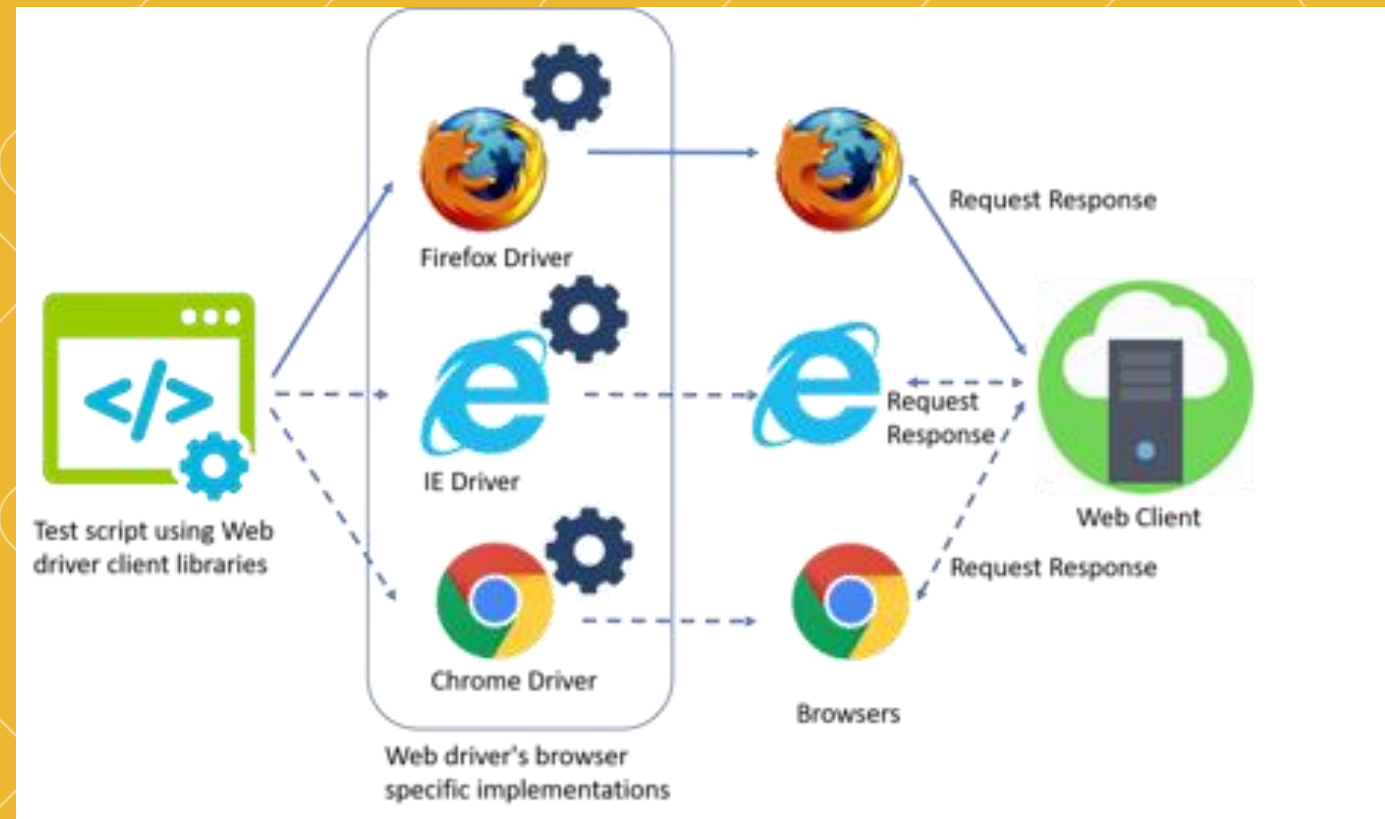
- É um framework utilizado para **testes de interfaces** utilizando Browsers como Chrome, Firefox, Edge e etc.
- Suporta diversas linguagens de programação, como por exemplo C#, Java, Python , Ruby.
- Do Framework do Selenium , temos a **API** que se comunica com os Browsers que é o **Selenium WebDriver**.

> Selenium WebDriver

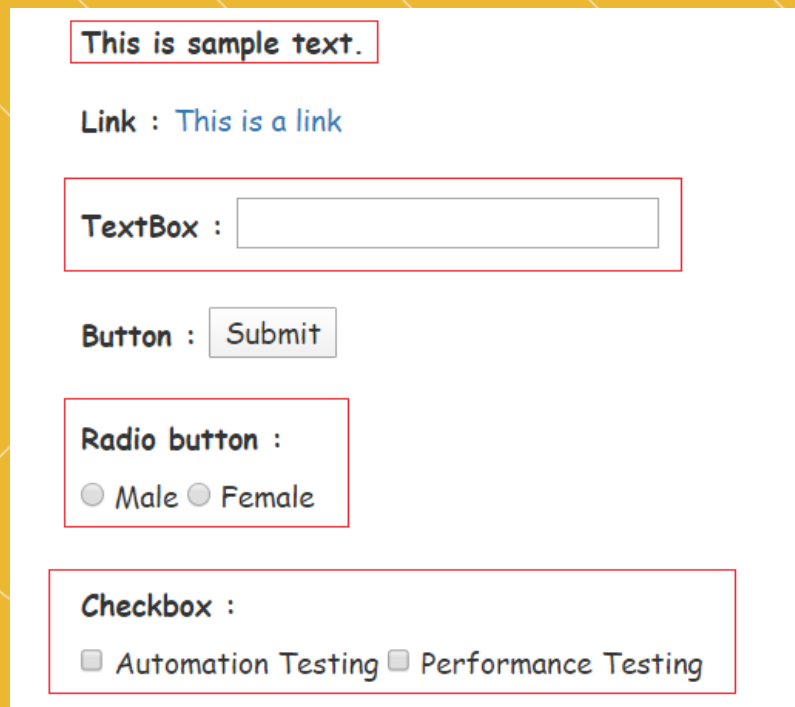


- O Webdriver é a classe base do Selenium.
- Responsável por **diversas funcionalidades** como abrir um browser, fechar um browser, abrir uma aba, fechar uma, entre outras.
- **Webdriver** é uma API orientada a objetos compactos que pode e interage diretamente com o aplicativo de testes. Utiliza a compatibilidade nativa do navegador para automação sem o uso de qualquer entidade periférica.

> Selenium WebDriver



> WebElement? O que é?



This is sample text.

Link : [This is a link](#)

TextBox :

Button :

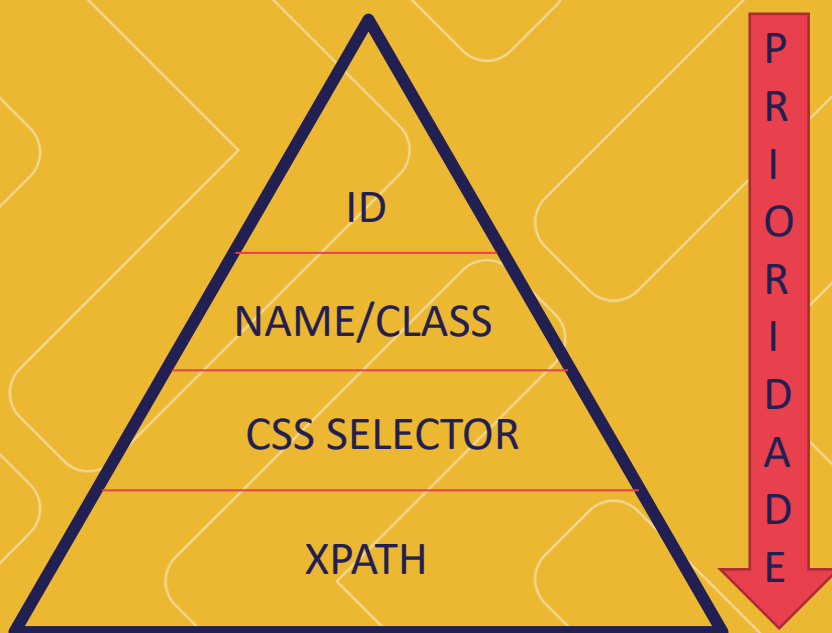
Radio button :
☐ Male ☐ Female

Checkbox :
☐ Automation Testing ☐ Performance Testing

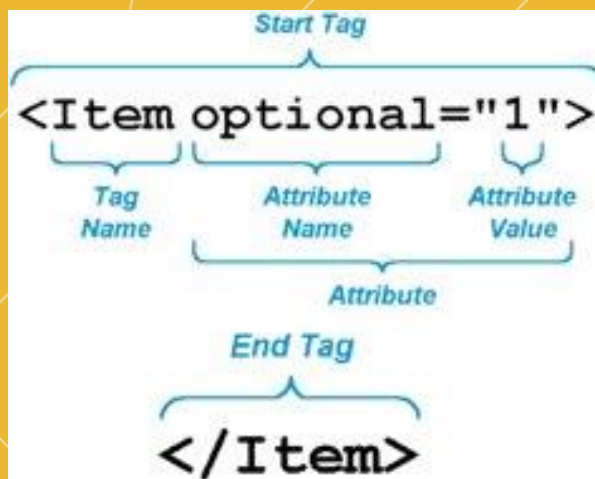
- O **WebElement** é uma classe do **Selenium** representando um elemento **HTML**. As principais operações de interação em uma página são realizadas através desta interface; por exemplo: interações com caixas de texto, botões, links, entre outras.
- **WebElement** representa um **elemento HTML**. Os documentos **HTML** são compostos por *elementos HTML*. Elementos HTML são escritos com uma marca de **início**, com uma marca de **fim**, com o **conteúdo** entre: `<tagname> conteúdo </ tagname>`.
- O **elemento HTML** é tudo a partir da marca de início para a marca de fim: `<P> Meu primeiro parágrafo HTML </ p>` elementos HTML podem ser aninhadas (elementos podem conter elementos). Todos os documentos HTML consistem em elementos HTML aninhadas.

> WebElement – Mapeando Elementos

HIERARQUIA DE MAPEAMENTO



ESTRUTURA HTML



COMPOSIÇÃO DE UM XPATH

```
Xpath=//tagname[@attribute='value']
```

- `//`: Select current node.
- **Tagname**: Tagname of the particular node.
- `@`: Select attribute.
- **Attribute**: Attribute name of the node.
- **Value**: Value of the attribute.

Outras formas de uso do Xpath:

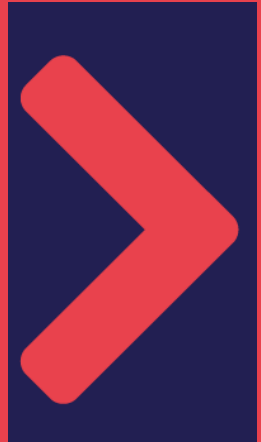
<https://www.guru99.com/xpath-selenium.html>

<https://www.swtestacademy.com/xpath-selenium/>



> WebElement – Mapeando Elementos

Localizador	Descrição	Referência
id	Mapeia um único elemento pela sua id	<code><div id="corpo"></code>
name	Mapeia elementos através do name	<code><form name="formulario"></code>
class	Mapeia pela classe de um elemento	<code><div class="produtos"></code>
cssSelector	Seletor CSS livre	Ex: <code>#corpo > .produtos</code>
XPath	Usa XPath para mapear	Ex: <code>//div[@id="corpo"]</code>



Design Patterns



Page Objects

- É um **Design Pattern** usado para o desenvolvimento de projetos de automação, conhecido também como **POM (Page Object Model)**.
- Nos permite criar um repositório de objetos com elementos contidos numa página web.
- Sob este modelo, para **cada página**, deve haver uma classe correspondente. Esta classe obtém e classifica os **WebElements** da página e também pode conter métodos que executam operações nesses **WebElements**.
- O nome dos métodos dessas classes podem ser dados de acordo com a tarefa que eles estão executando. Se iremos fazer login em um site de compras, o nome do método **POM** pode ser por exemplo: AutenticarUsuário, FazerLogin.

> Page Objects

this API is about
the application

`selectAlbumWithTitle()`
`getArtist()`
`updateRating(5)`

Page Objects

Album
Page

Album List
Page

this API is
about HTML

`findElementsWithClass('album')`
`findElementsWithClass('title-field')`
`getText()`
`click()`
`findElementsWithClass('ratings-field')`
`setText(5)`

HTML Wrapper

title: Whiteout
artist: In the Country
rating:

title: Ouro Negro
artist: Moacir Santos
rating:

PAGES

WebPage1
LoginPage

WebPage2
HomePage

TestCase1

RealizarLoginTest

TestBase

TESTS



Page Objects

Ao adotar o PageObject temos alguns benefícios:

- Reaproveitamento de Código.
- Código mais limpo.
- Facilidade na manutenção.
- Maior independência dos testes.



Page Objects

E também precisamos seguir algumas regras do padrão:

- Prover uma interface que seja fácil de programar.
- Encapsular os mecanismos necessários para cada página e manipular os dados no controle da interface gráfica.
- Não devem ser construídos para cada página, e sim para elementos relevantes da página.
- Um **PageObject** deve retornar sempre um **PageObject** ao navegar pelas páginas. Ou seja, se você navegar para outra página, o **PageObject** inicial deve retornar outro **PageObject** para a nova página.
- Sua responsabilidade é providenciar acesso ao estado das partes internas da página.



Esperas (Waits)

> WAITS – Esperas Do Selenium

- No Selenium WebDriver existem dois tipos de esperas:
 - **Implícita e Explícita.**
- **Implícita:**
 - A espera implícita basicamente diz ao WebDriver **um tempo máximo pelo qual ele deve aguardar**, quando estiver tentando encontrar um elemento.
 - Ex: `driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);`
- **Explícita:**
 - A espera explícita diz ao WebDriver **uma condição** (ou tempo) para que ele aguarde antes de prosseguir com a ação.
 - Ex:
`WebDriverWait wait = new WebDriverWait(driver, 10);`
`wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("botaoOk")));`





Estrutura de Projeto

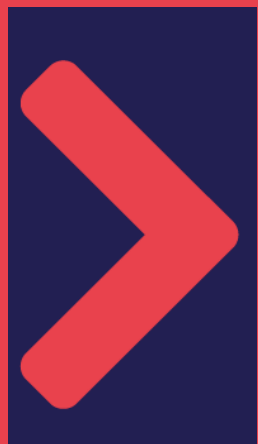


Diretórios Projeto

- Dentro de projetos construídos com JAVA, sempre teremos duas camadas : **MAIN e TEST**
- **Main:**
 - Aqui nesta camada ficará todas classes abstratas e de reutilização nos nossos testes, assim como, classes utilitárias e arquivos de resources.
- **Test :**
 - Aqui nesta camada ficará todas as classes referentes ao nossos casos de testes, como por exemplo, as page objects, ações, asserções , classes de testes e as testes suites(runner de testes, um determinado conjunto de testes).

> DIRETÓRIOS PROJETO

```
>Projeto
  >src
    >main
      >java
        >Famework
        >Utils
      >resources
    >test
      >java
        >PageObjects
        >Tasks
        >TestCases
        >TestSuites
        >Validations
```



Massa de Dados

> Tipos de Massa de Dados

- Existem diversas formas de obtermos input de dados para os nossos testes, veremos algumas formas abaixo:
 - **Datapool:**
 - “Piscina de dados”, modularizar os dados para o teste, os dados são separados por vírgulas e por linhas, no caso temos a variável e o dado na linha abaixo, assim podemos utilizar a informação a qualquer momento do nosso teste, basta ler o arquivo de extensão .CSV.
 - Ex: usuario,senha
teste, 123
 - **Dataprovider :**
 - Como o nome já diz, provedor de dados, normalmente esse provedor de dados serve para gerar dados para o programados utilizar, tanto dados fakes como pré-criados. A biblioteca do JavaFaker por exemplo é um Dataprovider. Bibliotecas que geram dados como CPF válidos por exemplo, também são Dataproviders.

> Tipos de Massa de Dados

➤ Dataclass:

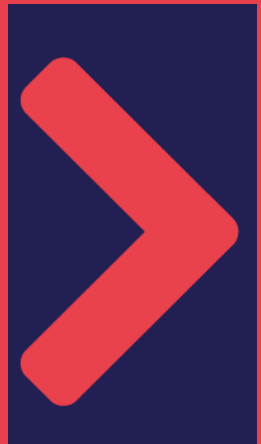
- É uma estrutura simples basicamente com dados, também chamado de registro e sem comportamentos.
- Ex:

```
public static Object [][] nomeMetodo(){  
    return new Object [][] {"xxx","yyy"};  
}
```

➤ Propriedades :

- Normalmente usado no contexto da plataforma Java, é um arquivo com extensão **.properties**, utilizado para armazenar **um mapa de conjunto com o par chave-valor** associados. Com as propriedades podemos trabalhar com input e output de dados para serem utilizados a qualquer momento dentro da chamada de código.
- Ex:

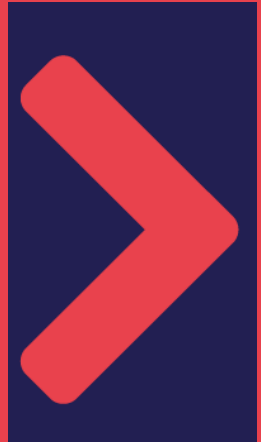
```
1 url.index=meuendereco  
2 nome=meunome  
3 sobrenome=meusobrenome
```



Relatórios de Execução

> Relatório de Execução

- Todo projeto de automação deve conter um relatório de execução dos scripts executados como output.
- No relatório devem conter evidências das **jornadas construídas e suas validações**, o famoso by step.
- Existem diferentes frameworks de geração de relatórios de execução, nós iremos aprender sobre o framework **ExtentReport** , conhecido mundialmente e amplamente usado no mercado.
- Abaixo a documentação completa sobre o framework:
 - <https://www.extentreports.com/docs/versions/5/java/>



Frames/Iframes e Alerts

> Alerts

- Um Alerta no Selenium é uma pequena caixa de mensagem que aparece na tela para dar ao usuário alguma informação ou notificação. Notifica o usuário com algumas informações ou erros específicos, pede permissão para realizar certas tarefas e também fornece mensagens de aviso.
- Com o WebDriver é possível interagir com o pop-up através dos métodos abaixo:
 - `getText()` – irá obter o texto do alerta.
 - `Accept()` – irá aceitar o pop-up OK.
 - `Dismiss()` – irá descartar o pop-up CANCEL.
 - `SendKeys()` - irá enviar caracteres para o pop-up (Mensagem).

> Alerts



```
//Store the alert in a variable
Alert alert = driver.switchTo().alert();

//Store the alert in a variable for reuse
String text = alert.getText();

//Press the Cancel button
alert.dismiss();

//Type your message
alert.sendKeys("Selenium");

//Press the OK button
alert.accept();
```



Frames/Iframes

- Frame, em HTML, nada mais é que uma sub-janela, uma moldura, um pedaço em um contexto maior.
- Normalmente os Frames possuem dois elementos básicos:
 - Uma página chamada frameset, que contém em seu código tags que especificam a divisão das frames dentro da janela do browser;
 - As páginas internas em si, chamadas frames, carregadas de acordo com as instruções contidas no código da frameset.



Frames/Iframes

- Um **iFrame** é um código em HTML que permite que um elemento seja exibido dentro de outro elemento. Somente possível em páginas HTML, permite incorporar documentos, vídeos e mídia interativa em uma página.

Ao fazer isso, você pode exibir uma **página secundária da web** na sua página principal.

Você pode adicioná-lo como forma de contextualizar um determinado tópico para os leitores.

Dá para inserir o elemento usando a tag **<iframe>** em um documento HTML.

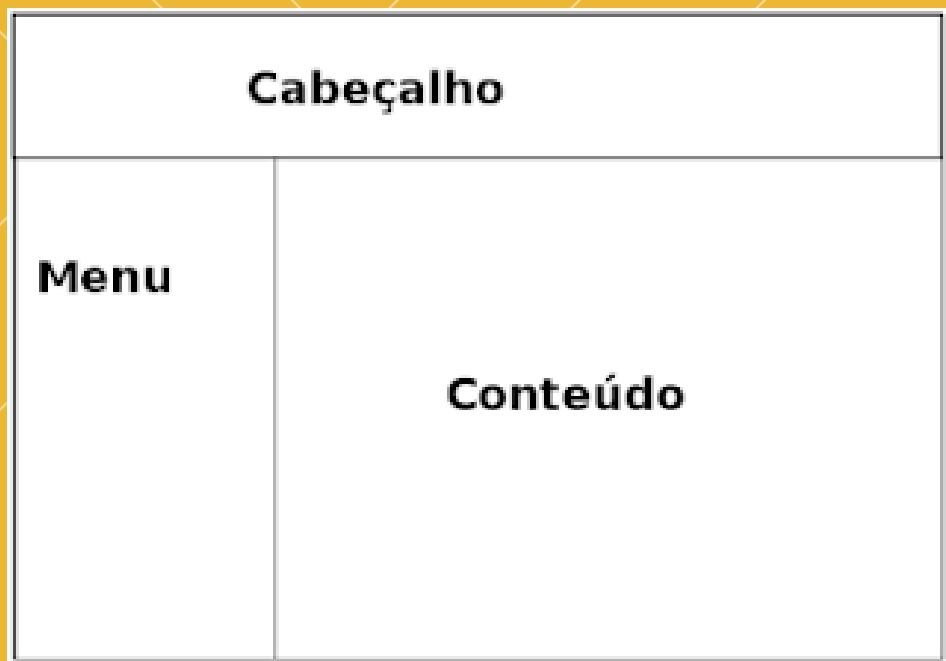
A tag `<iframe>...</iframe>` é usada para que seja inserido o vídeo dentro do conteúdo.

```
<iframe src="https://www.youtube.com/embed/dXBohfjc4WA" width="680" height="480"
allowfullscreen></iframe>
```

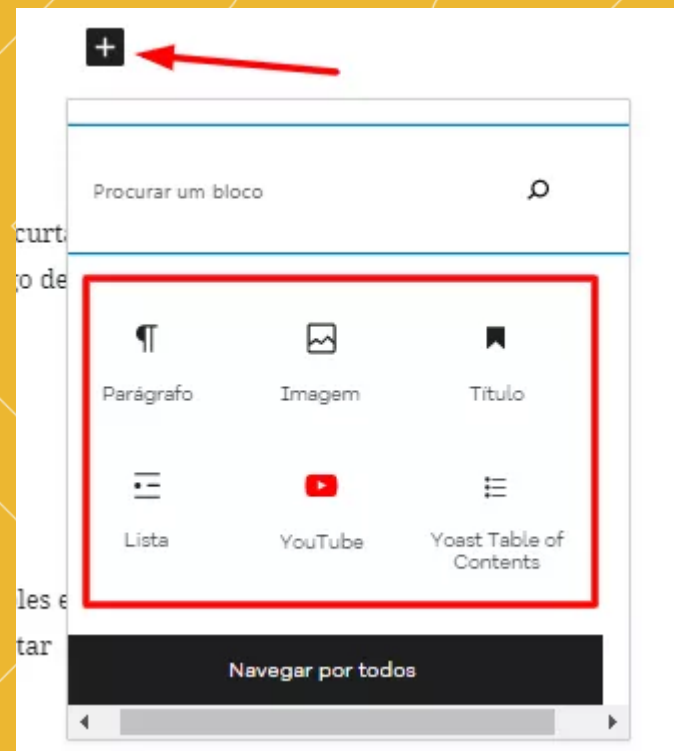


Frames/Iframes

Exemplo Frame



Exemplo Iframe



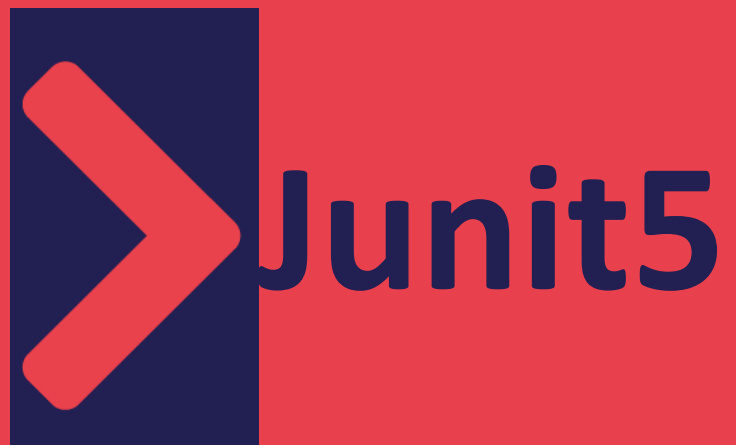


Troca de Contexto

- Normalmente quando trabalhamos com os dois tipos de elementos ALERTS e Frames/Iframes precisamos realizar uma troca de contexto, tendo em vista que o **Alert** é uma ação que ocorre sobre o contexto atual em que o Driver esta atuando e para podermos interagir precisamos realizar essa troca através do **SWITCHTO()**. Assim como para o Frame/Iframe também necessitamos realizar esta troca de contexto para acessar os seus elementos dentro de páginas HTML.
- Para realizarmos a troca de contexto com os Alertas, usamos o **switchTo().alert**.
- Para realizarmos a troca de contexto com os Frames/Iframes , usamos o **switchTo().frame** , com este tipo de elemento podemos mapear através de INDEX, NAME ou ID.

```
driver.switchTo().frame("Frame_ID");  
driver.switchTo().frame(1);
```

- Quando trabalhamos com elementos do tipo Frame/Iframe sempre precisamos voltar ao context original para conseguirmos acessar um novo Frame/Iframe , para isso sempre precisamos realizar o **switchTo().defaultContent()**.



> Anotações Importantes

- @BeforeAll - Configurações que acontecem antes de todos os testes.
- @BeforeEach - Configurações que acontecem antes de cada teste.
- @AfterAll - Configurações que acontecem depois de todos os testes.
- @AfterEach - Configurações que acontecem depois de cada teste.
- @Test - Identifica ao compilador que aquele método é um teste.
- @ParametrizedTest - Identifica que aquele método vai receber um parâmetro vindo de algum Datapool ou Dataprovider.
- @CsvFileSource - Procura um arquivo Csv para receber os parâmetros no método de teste parametrizado.
- @MethodSource - Procura um método de uma classe.

MUITO OBRIGADO!

Entre em contato:

dbsqualidade@db.tec.br

