Guia Passo a Passo para Configuração de Repositório e Execução de Testes com Selenium

1. Criar um Repositório no GitHub

1. Fazer Login no GitHub:

- - Acesse https://github.com e faça login na sua conta.

2. Criar um Novo Repositório:

- - Clique no ícone de "+" no canto superior direito e selecione "New repository".
- - Preencha os detalhes do repositório:
  - - Nome do repositório: MetricConversionTests
  - - Descrição: (Opcional) Uma breve descrição do repositório.
  - - Público/Privado: Escolha a visibilidade do seu repositório.
  - - Iniciar este repositório com um README: Marque esta opção.
- - Clique em "Create repository".

2. Clonar o Repositório Usando VS Code

1. Abrir o VS Code:

- - Abra o VS Code no seu computador.

2. Abrir a Paleta de Comandos:

- - Pressione Ctrl+Shift+P (Windows/Linux) ou Cmd+Shift+P (Mac) para abrir a Paleta de Comandos.

3. Clonar Repositório:

- - Digite "Git: Clone" e selecione "Git: Clone" na lista.
- - Você será solicitado a inserir a URL do repositório. Vá para o seu repositório no GitHub, clique no botão "Code" e copie a URL do repositório.
- - Cole a URL na caixa de entrada no VS Code e pressione Enter.
- - Escolha um diretório local onde você deseja clonar o repositório.

4. Abrir o Repositório Clonado:

- - Após clonar, o VS Code irá perguntar se você deseja abrir o repositório clonado. Clique em "Open".

3. Configurar o Projeto no VS Code

1. Inicializar um Projeto .NET:

- - Abra um novo terminal no VS Code selecionando Terminal > New Terminal no menu superior.
- - Navegue até a pasta do repositório, se ainda não estiver nela.
- - Inicialize um novo projeto .NET usando os seguintes comandos:

```
dotnet new sln -n MetricConversionTests
dotnet new xunit -n MetricConversionTests
dotnet sln MetricConversionTests.sln add MetricConversionTests/MetricConversionTests.csproj
```

2. Adicionar Pacotes Necessários:

- - Adicione Selenium, xUnit e outros pacotes necessários:

```
dotnet add package Selenium.WebDriver

dotnet add package xunit

dotnet add package xunit.runner.visualstudio

dotnet add package WebDriverManager

dotnet add package Microsoft.NET.Test.Sdk

dotnet add package DotNetSeleniumExtras.WaitHelpers
```

3. No Visual Studio, adicione as pastas bin e obj no .gitignore: no painel Source Control, selecione View as a Tree, selecione as duas pastas e use o botão da direita para selecionar Add to .gitignore.

3. Criar e Adicionar Arquivos de Teste:

- - Adicione um novo arquivo de teste (por exemplo, MetricConversionTests.cs) com o código de teste necessário.

4. Código de Teste de Exemplo

Aqui está um arquivo de teste de exemplo MetricConversionTests.cs:

```
namespace MetricConversionTests;

using System;
using Xunit;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.WaitHelpers;



public class GoogleCelsiusToFahrenheitTest : IDisposable
{
   IWebDriver driver;
   WebDriverWait wait;



   public GoogleCelsiusToFahrenheitTest()
   {
      driver = new ChromeDriver();
      driver.Manage().Window.Maximize();
      wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));

   }

   [Theory]
   [InlineData("100", "212")]
   public void TestCelsiusToFahrenheitConversion(string c, string f)
```

```
    {
        // Navigate to the target webpage
        driver.Navigate().GoToUrl("https://www.metric-conversions.org/temperature/celsius-to-fahrenheit.htm#google_vignette");

        // Locate the Celsius input field
        IWebElement celsiusInputField = wait.Until(ExpectedConditions.ElementIsVisible(By.XPath("//*[@id='arg']")));

        // Input a value into the Celsius field
        celsiusInputField.Clear();
        celsiusInputField.SendKeys(c);

        // Locate the Fahrenheit result field (it may update automatically)
        IWebElement fahrenheitResultField = wait.Until(ExpectedConditions.ElementIsVisible(By.XPath("//*[@id='answerDisplay']")));

        // Verify the Fahrenheit result
        string result = fahrenheitResultField.Text;
        Assert.Contains(f, result);
    }

    public void Dispose()
    {
        driver.Quit();
    }
}
```

Substitua http://example.com/metric-conversion pela URL real da página de conversão métrica que você deseja testar.

5. Executando o Teste Localmente e Gerando Relatórios

1. Compilar o Projeto:

- - No terminal, navegue até a raiz do projeto e execute:

```
dotnet build
```

2. Executar o Teste e Gerar um Relatório XML:

- - No terminal, execute o seguinte comando para executar o teste e gerar um relatório XML:

```
dotnet test --logger "trx;LogFileName=TestResults.trx"
```

3. Converter o Relatório XML para HTML:

- - Instale a ferramenta trx2html globalmente, se ainda não tiver feito:

```
dotnet tool install --global trx2html
```

- - Converta o relatório TRX para HTML:

```
trx2html TestResults.trx TestResults.html
```

Visualizando o Relatório

1. Abrir o Relatório HTML:

- - Abra o arquivo TestResults.html no seu navegador para visualizar o relatório de teste.

Resumo

1. Criar um repositório no GitHub.

2. Clonar o repositório usando o VS Code.

3. Configurar um projeto .NET e adicionar pacotes necessários.

4. Criar e adicionar um arquivo de teste com Selenium e xUnit.

5. Executar os testes localmente e gerar um relatório XML.

6. Converter o relatório XML para HTML para fácil visualização.