Exemplo de Teste com Selenium, C#, xUnit e Relatórios usando WebDriverManager e ExtentReports

Este documento fornece um exemplo de como criar um teste parametrizado usando Selenium, C#, xUnit e gerar relatórios de execução com ExtentReports. O exemplo inclui a configuração do ambiente, o uso de WebDriverManager para gerenciar os binários do WebDriver, a implementação de testes parametrizados e a geração de relatórios de teste detalhados.

Passo 1: Configurar o Ambiente

1.1: Instalar o SDK do .NET

Certifique-se de que você tem o SDK do .NET instalado no seu sistema. Você pode baixá-lo em https://dotnet.microsoft.com/download.

1.2: Instalar o Visual Studio Code

Baixe e instale o Visual Studio Code em https://code.visualstudio.com/.

1.3: Instalar a Extensão C# para o Visual Studio Code

1. Abra o Visual Studio Code.
2. Vá para a visualização de Extensões clicando no ícone de Extensões na Barra de Atividades na lateral da janela ou pressionando Ctrl+Shift+X.
3. Procure por "C#" e instale a extensão oficial da Microsoft.

Passo 2: Configurar o Projeto

2.1: Criar um Novo Projeto de Teste

Abra um terminal no Visual Studio Code (Ctrl+` ou via o menu Terminal) e execute os seguintes comandos:

1. Crie um novo projeto de teste com xUnit executando:
   dotnet new xunit -n SeleniumParametrizedTest
2. Navegue até o diretório do projeto:
   cd SeleniumParametrizedTest

2.2: Adicionar Pacotes NuGet Necessários

Execute os seguintes comandos no terminal para adicionar os pacotes necessários ao projeto:

1. dotnet add package Selenium.WebDriver
2. dotnet add package Selenium.WebDriver.ChromeDriver

3. dotnet add package WebDriverManager
4. dotnet add package xunit
5. dotnet add package xunit.runner.visualstudio
6. dotnet add package Microsoft.NET.Test.Sdk
7. dotnet add package ExtentReports

Passo 3: Escrever o Teste Selenium com WebDriverManager, xUnit e ExtentReports

Abra o arquivo de teste no seu projeto, por exemplo, UnitTest1.cs, e substitua o conteúdo com o código abaixo.

Código de Teste Parametrizado com Relatório

```
using System;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using WebDriverManager;
using WebDriverManager.DriverConfigs.Impl;
using Xunit;
using AventStack.ExtentReports;
using AventStack.ExtentReports.Reporter;

public class GoogleSearchTests : IDisposable
{
   private readonly IWebDriver _driver;
   private readonly ExtentReports _extent;
   private readonly ExtentTest _test;

   public GoogleSearchTests()
   {
      // Usar o WebDriverManager para configurar o ChromeDriver
      new DriverManager().SetUpDriver(new ChromeConfig());
      _driver = new ChromeDriver();

      // Configurar o ExtentReports
      var htmlReporter = new ExtentSparkReporter("extent_report.html");
      _extent = new ExtentReports();
      _extent.AttachReporter(htmlReporter);

      // Criar um teste no relatório
      _test = _extent.CreateTest("GoogleSearchTests");
   }

   [Fact]
```

```
  {
    try
    {
      // Navegar para a página do Google
      _driver.Navigate().GoToUrl("https://www.google.com");
      _test.Log(Status.Pass, "Navegou para Google.com");

      // Encontrar a caixa de pesquisa pelo atributo name
      IWebElement searchBox = _driver.FindElement(By.Name("q"));
      _test.Log(Status.Pass, "Encontrou a caixa de pesquisa");

      // Realizar a pesquisa

      var searchItem = “Selenium Webdriver”;
      searchBox.SendKeys(searchTerm);
      searchBox.SendKeys(Keys.Enter);
      _test.Log(Status.Pass, $"Realizou a pesquisa por {searchTerm}");

      // Esperar um pouco para ver os resultados
      System.Threading.Thread.Sleep(3000);

      // Verificar se o título da página contém o termo de pesquisa
      Assert.Contains(searchTerm, _driver.Title, StringComparison.OrdinalIgnoreCase);
      _test.Log(Status.Pass, $"O título da página contém o termo de pesquisa
{searchTerm}");
    }
    catch (Exception e)
    {
      _test.Log(Status.Fail, e.ToString());
      throw;
    }
  }

  public void Dispose()
  {
    // Fechar o navegador
    _driver.Quit();
    _driver.Dispose();

    // Escrever o relatório
    _extent.Flush();
  }
}
```

Passo 4: Executar o Teste e Gerar Relatório

1. Navegue até o diretório do seu projeto no terminal.
2. Execute o comando para rodar os testes:
   dotnet test

O relatório será gerado no arquivo "extent_report.html".

Conclusão

Este exemplo demonstra como criar um teste parametrizado usando Selenium, C#, xUnit e gerar relatórios com o ExtentReports. A abordagem mostrada ajuda a garantir que os binários do WebDriver estejam sempre atualizados e configurados corretamente, além de permitir a execução do mesmo teste com diferentes conjuntos de dados e a geração de relatórios detalhados.