Exemplo de Teste Parametrizado, com Page Object, usando Selenium, C# e xUnit

Este documento fornece um exemplo de como criar um teste parametrizado com Page Object, usando Selenium, C# e xUnit. Um teste parametrizado permite executar o mesmo teste com diferentes conjuntos de dados, tornando os testes mais robustos e eficientes.

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

Passo 3: Escrever o Teste Selenium com WebDriverManager e xUnit

1. Abra o arquivo de teste no seu projeto, por exemplo, UnitTest1.cs.
2. Atualize o código para usar o WebDriverManager para gerenciar os binários do WebDriver automaticamente.

Classe de Página (Page Object)

```
using OpenQA.Selenium;

public class GoogleSearchPage
{
   private readonly IWebDriver _driver;
   private readonly By _searchBox = By.Name("q");

   public GoogleSearchPage(IWebDriver driver)
   {
      _driver = driver;
   }

   public void NavigateTo()
   {
      _driver.Navigate().GoToUrl("https://www.google.com");
   }

   public void EnterSearchTerm(string term)
   {
      _driver.FindElement(_searchBox).SendKeys(term);
      _driver.FindElement(_searchBox).SendKeys(Keys.Enter);
   }

   public string GetTitle()
   {
      return _driver.Title;
   }
}
```

Teste Parametrizado com xUnit

```
using System;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
```

```
using Xunit;

public class GoogleSearchTests : IDisposable
{
    private readonly IWebDriver _driver;
    private readonly GoogleSearchPage _googleSearchPage;

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

    [Theory]
    [InlineData("Selenium WebDriver")]
    [InlineData("xUnit testing")]
    [InlineData("C# tutorials")]
    public void TestGoogleSearch(string searchTerm)
    {
        // Navegar para a página do Google
        _googleSearchPage.NavigateTo();

        // Inserir o termo de pesquisa
        _googleSearchPage.EnterSearchTerm(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, _googleSearchPage.GetTitle(),
StringComparison.OrdinalIgnoreCase);
    }

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

Passo 4: Executar o Teste

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

Conclusão

Este exemplo demonstra como criar um teste parametrizado, com Page Object, usando Selenium, C# e xUnit. Os testes parametrizados permitem a execução do mesmo teste com diferentes conjuntos de dados, aumentando a eficiência e cobertura dos testes automatizados.