**Pedro William Andrade Carvalho**

Teste de Software

Atividade 1

**1. Entendendo o Problema**

**Resumo do Problema**

Problema: O autor da pergunta tem um método assíncrono DoSomethingAsync() em uma interface que retorna Task. Ao testar com o Moq, o teste falha quando o código sob teste aguarda a chamada a DoSomethingAsync().

[TestMethod()]

public async Task MyAsyncTest()

{

Mock<ISomeInterface> mock = new Mock<ISomeInterface>();

mock.Setup(arg => arg.DoSomethingAsync()).Callback(() => { <my code here> });

...

}

Solução Aceita: A solução aceita configura o mock para retornar uma Task que é completada imediatamente, o que permite que o teste passe sem erros.

**2. Solução Aceita**

A solução aceita para esse problema é usar Returns(Task.CompletedTask) para métodos que retornam Task não genérico:

mock.Setup(arg => arg.DoSomethingAsync())

.Returns(Task.CompletedTask);

**Motivo da Aceitação:**

Simplicidade: Usar Returns(Task.CompletedTask) é a forma mais direta e correta de configurar o retorno de um método assíncrono que não precisa retornar valores. Task.CompletedTask representa uma Task que já foi completada, o que é adequado para métodos que não precisam retornar dados.

Conformidade: Task.CompletedTask é a forma recomendada para métodos que retornam Task, e é suportada por todas as versões modernas do .NET.

**3. Outras Respostas Não Aceitas**

**Reposta 1**

mock.Setup(arg => arg.DoSomethingAsync())

.Returns(Task.FromResult(default(object)));

Motivo da Não Aceitação: Task.FromResult(default(object)) é utilizado para métodos que retornam um valor (Task), não apenas Task. Para métodos que retornam Task, usar Task.CompletedTask é mais apropriado e claro.

**Resposta 2**

mock.Setup(arg => arg.DoSomethingAsync())

.Callback(() => { /\* Código aqui \*/ })

.Returns(Task.FromResult(0));

Motivo da Não Aceitação: Embora essa abordagem funcione, adicionar Returns(Task.FromResult(0)) pode ser confuso porque Task.FromResult(0) retorna uma Task de um tipo específico (Task), e o método esperado é Task. Isso não se alinha bem com o tipo de retorno esperado e pode levar a confusões.

**Resposta 3**

mock.SetupAsync(arg => arg.DoSomethingAsync());

mock.SetupAsync(arg => arg.DoSomethingAsync()).Callback(() => { /\* Código aqui \*/ });

mock.SetupAsync(arg => arg.DoSomethingAsync()).Throws(new InvalidOperationException());

Motivo da Não Aceitação: Esta solução sugere o uso de uma biblioteca externa (Talentsoft.Moq.SetupAsync) que adiciona métodos de extensão para SetupAsync. Enquanto isso pode simplificar a configuração, não é uma solução padrão e pode não ser amplamente adotada ou suportada.

**Resposta 4**

public static class MoqExtensions

{

public static IReturnsResult<TMock> ReturnsAsync<TMock>(this IReturns<TMock, Task> mock)

where TMock : class

{

return mock.Returns(Task.CompletedTask);

}

public static IReturnsResult<TMock> ReturnsAsync<TMock>(this IReturns<TMock, ValueTask> mock)

where TMock : class

{

return mock.Returns(ValueTask.CompletedTask);

}

}

Motivo da Não Aceitação: Embora fornecer extensões adicionais possa ser útil, a solução padrão (Task.CompletedTask ou Returns(Task.CompletedTask)) é mais direta e evita a necessidade de adicionar código auxiliar. Muitas vezes, soluções padrão e simples são preferidas para evitar muita complexidade desnecessária.

**4. Como testar**

Clone o repositório no [Github](https://github.com/pewilliam/Teste_Software_2024_Carvalho_Pedro).

Em seguida vá até a pasta clonada através do terminal e utilize o comando:

dotnet test

**5. Conclusão**

Utilizei de exemplo prático para ilustrar o problema descrito pelo autor da pergunta e reproduzi uma solução aceita e elegante.