O que é um teste unitário?

No desenvolvimento de software, os testes unitários testam basicamente uma parte individual ou unidade de código (principalmente métodos) a fim de verificarem se a unidade em teste funciona como esperado pelo programador ou não.

Normalmente os testes são escritos para testar métodos de uma classe, funções ou outros pequenos pedaços de um programa, por forma a avaliar cada "unidade" individualmente.

De uma forma geral os testes são escritos em forma de funções destinadas a avaliar e determinar se um determinado valor retornado, após a realização do teste é igual ao valor esperado pelo programador, ou não. O objetivo do teste será sempre isolar as unidades de código, verificar se estão a ser executadas corretamente e por fim informar o programador do resultado do teste.

Por que precisamos de teste de unidade

Porque não haveríamos de precisar? Um dos maiores e mais valiosos benefícios de usar testes unitários no desenvolvimento de projetos é o fato de nos dar alguma confiança de que o código efetivamente funciona de acordo com o esperado, de acordo com o que pensamos antes de o escrever.

Pesquisem um pouco sobre TDD.

Resumidamente: O princípio básico do TDD é primeiro a criação do teste e depois a construção do código para atendê-lo, desta forma esta técnica garante que a aplicação só tenha código suficiente para suprir o negócio, e nada mais. Esta técnica também valoriza o desenvolvimento ágil, uma vez que defende a eficiência no desenvolvimento de requisitos. Mas não é simplesmente sair desenvolvendo testes, tem todo um passo a passo recomendado para que isso ocorra.

Tem um livro muito bom Chamado:

Test-Driven Development: Teste e Design no Mundo Real com .NET

Lá o autor explica muito bem sobre os melhores caminhos ao TDD, Baby Steps, entre outras coisas muito interessante. Bom, mas isso fica para outra conversa...

Voltando somente nos Testes Unitários.

Frameworks de Teste .NET

No caso específico da plataforma .NET, merecem destaque como alternativas para a implementação de testes unitários:

- O Visual Studio Unit Testing Framework (MS Test), parte integrante da própria IDE de desenvolvimento da Microsoft;
- O NUnit, framework open source derivado do projeto JUnit (um dos primeiros frameworks para testes unitários na plataforma Java);
- O xUnit.net, também de código aberto.

Vamos focar no Nunit. Creio que todos os Testes Unitários do Admin. estão sob este framework.

Explicando alguns conceitos:

Atributo TestFixture

O atribut o TestFixture é uma indicação de que uma classe contém métodos de teste. Quando você menciona esse atributo para uma classe em seu projeto, a aplicação Test Runner irá procurar por métodos de testes.

```
using NUnit.Framework;
using System.Text;

namespace UNitTestando
{
    [TestFixture]
    public class Program
    {
    }
}
```

Atributo Test

O atributo de Test indica que um método dentro de um TestFixture deve ser executado pela aplicação Test Runner. O método deve ser público e retornar void.

```
using NUnit.Framework;
using System.Text;

namespace UNitTestando
{
[TestFixture]
  public class Program
  {
    [Test]
    public void Test()
    {
      ...
    }
  }
}
```

Classe Assert

A classe Assert é usada para confirmar se os casos de testes estão produzindo o resultado esperado ou não usando métodos auxiliares como AreEqual () ou AreNotEqual ().

Atributo ExpectedException

Você pode fortalecer o seu código para tratar várias exceções usando blocos try/catch. Mas em algumas circunstâncias você quer ter certeza de que uma exceção vai ocorrer. Para contornar este problema você pode usar o atributo ExpectedException conforme abaixo:

```
[TestFixture]
public class Program
{
[Test]
  [ExpectedException(typeof(DivideByZeroException))]
  public void Test()
  {
    int i=10,j=0,x;
    x = i / j;
  }
}
```

MOQ e Setup

Existe na língua portuguesa o verbo **mocar** que significa enganar, atraiçoar ou ainda esconder (*popular*), mas na área de software **mocar objetos** ou **Mock Objects** significa objetos que imitam objetos reais para realização de testes de software.

Os objetos Mocks são geralmente usados em testes de unidade.

Assim, os objetos Mock são criados para testar o comportamento de algum outro objeto (real); com isso estamos mocando, ou seja, simulando ou fingindo o objeto real e fazendo algumas operações de forma controlada de modo que o resultado retornado (teste) é sempre válido.