Aula 32

Prof: Henrique Augusto Maltauro

Codificar Back-end de Aplicações Web

NUnit

- [TestCase]

O atributo de configuração [TestCase] irá demarcar um método de uma classe demarcada com [TestFixture], definindo o método como um caso de teste, determinando situações de valores para os parâmetros do método, que serão utilizados para executar os testes.

NUnit

- [TestCase]

```
[TestFixture]
public class ClasseDeTeste
{
    [TestCase(1, "ABC")]
    [TestCase(2, "DEF")]
    [TestCase(3, "GHI")]
    public void MetodoDeTeste(int numero, string texto)
    {
        //Bloco de código do método
    }
}
```

NUnit

- [Random]

O atributo de configuração [Random] irá demarcar os parâmetros de um método demarcado com [Test], determinando uma faixa de valores aleatórios, que serão utilizados uma quantidade determinada de vezes, para executar os testes.

NUnit

- [Random]

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste([Random(1, 5, 10)] int numero)
    {
        //Bloco de código do método
    }
}
```

NUnit

- [Ignore]

O atributo de configuração [Ignore] irá demarcar uma classe demarcada com [TestFixture] ou um método demarcado com [Test], determinando que aquela classe ou método serão ignorados no processo de testes.

NUnit

- [Ignore]

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    [Ignore]
    public void MetodoDeTeste()
    {
        //Bloco de código do método
    }
}
```

NUnit

- Assert

O Assert é uma classe do NUnit, e é uma das peças centrais do processo de testes. Essa classe possui diversos métodos estáticos, que irão validar se o teste foi ou não bem sucedido.

Nas versões iniciais do NUnit, havia um método para cada validação diferente. Este modelo ainda funciona, mas como os novos recursos não são mais adicionados a esse modelo, o método That deve ser usado para ter acesso total aos recursos novos do NUnit.

NUnit

- Assert (That)

O método That faz parte do modelo de validação baseado em restrição. Ele vai servir para validar se o teste foi ou não bem sucedido.

Ele recebe apenas dois parâmetros. O primeiro é o valor que será validado. O segundo é uma expressão do tipo IResolveConstraint, que é o que vai validar se o valor está de acordo segundo uma restrição.

NUnit

- IResolveConstraint

O IResolveConstraint é uma interface do NUnit, que vai agrupar as expressões que irão definir uma restrição. As principais classes que implementam essa interface e fornecem métodos estáticos para definir essas restrições são:

- Is
- Has
- Does
- Throws

NUnit

- IResolveConstraint (Is)

A classe Is fornece diversos métodos estáticos para definir restrições, entre os quais os principais são:

- Null
- Not
- True
- False
- Positive
- Negative
- Zero
- NaN
- Empty
- EqualTo
- GreaterThan
- GreaterThanOrEqualTo
- LessThan
- LessThanOrEqualTo
- TypeOf
- AnyOf

NUnit

- IResolveConstraint (Is Null)

O método Null vai retornar uma restrição para validar se o valor recebido é nulo.

NUnit

- IResolveConstraint (Is Null)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        string? texto = null;
        Assert.That(texto, Is.Null);
    }
}
```

NUnit

- IResolveConstraint (Is True)

O método True vai retornar uma restrição para validar se o valor recebido é verdadeiro.

NUnit

- IResolveConstraint (Is True)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        bool boleano = true;
        Assert.That(boleano, Is.True);
    }
}
```

NUnit

- IResolveConstraint (Is False)

O método False vai retornar uma restrição para validar se o valor recebido é falso.

NUnit

- IResolveConstraint (Is False)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        bool boleano = false;
        Assert.That(boleano, Is.False);
    }
}
```

NUnit

- IResolveConstraint (Is Positive)

O método Positive vai retornar uma restrição para validar se o valor numérico recebido é positivo.

NUnit

- IResolveConstraint (Is Positive)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 1234;
        Assert.That(numero, Is.Positive);
    }
}
```

NUnit

- IResolveConstraint (Is Negative)

O método Negative vai retornar uma restrição para validar se o valor numérico recebido é negativo.

NUnit

- IResolveConstraint (Is Negative)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = -1234;
        Assert.That(numero, Is.Negative);
    }
}
```

NUnit

- IResolveConstraint (Is Zero)

O método Zero vai retornar uma restrição para validar se o valor numérico recebido é igual a zero.

NUnit

- IResolveConstraint (Is Zero)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 0;
        Assert.That(numero, Is.Zero);
    }
}
```

NUnit

- IResolveConstraint (Is NaN)

O método NaN *(Not a Number)* vai retornar uma restrição para validar se o valor recebido não é um número.

NUnit

- IResolveConstraint (Is NaN)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        double numero = Double.NaN;
        Assert.That(numero, Is.NaN);
    }
}
```

NUnit

- IResolveConstraint (Is Empty)

O método Empty vai retornar uma restrição para validar se o valor recebido está vazio.

NUnit

- IResolveConstraint (Is Empty)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        string texto = "";
        Assert.That(texto, Is.Empty);
    }
}
```

NUnit

- IResolveConstraint (Is EqualTo)

O método EqualTo vai receber um objeto como parâmetro e retornar uma restrição para validar se o valor recebido é igual ao parâmetro definido.

NUnit

- IResolveConstraint (Is EqualTo)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        string texto = "Texto";
        Assert.That(texto, Is.EqualTo("Texto"));
    }
}
```

NUnit

- IResolveConstraint (Is GreaterThan)

O método GreaterThan vai receber um objeto como parâmetro e retornar uma restrição para validar se o valor recebido é maior que o parâmetro definido.

NUnit

- IResolveConstraint (Is GreaterThan)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 1234;
        Assert.That(numero, Is.GreaterThan(12));
    }
}
```

NUnit

- IResolveConstraint (Is GreaterThanOrEqual)

O método GreaterThanOrEqual vai receber um objeto como parâmetro e retornar uma restrição para validar se o valor recebido é maior ou igual ao parâmetro definido.

NUnit

- IResolveConstraint (Is GreaterThanOrEqual)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 1234;
        Assert.That(numero, Is.GreaterThanOrEqual(1234));
    }
}
```

NUnit

- IResolveConstraint (Is LessThan)

O método LessThan vai receber um objeto como parâmetro e retornar uma restrição para validar se o valor recebido é menor que o parâmetro definido.

NUnit

- IResolveConstraint (Is LessThan)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 12;
        Assert.That(numero, Is.LessThan(1234));
    }
}
```

NUnit

- IResolveConstraint (Is LessThanOrEqual)

O método LessThanOrEqual vai receber um objeto como parâmetro e retornar uma restrição para validar se o valor recebido é menor ou igual ao parâmetro definido.

NUnit

- IResolveConstraint (Is LessThanOrEqual)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 1234;
        Assert.That(numero, Is.LessThanOrEqual(1234));
    }
}
```

NUnit

- IResolveConstraint (Is TypeOf)

O método TypeOf vai receber um parâmetro de tipo e retornar uma restrição para validar se o valor recebido é do mesmo tipo que o tipo definido no parâmetro.

NUnit

- IResolveConstraint (Is TypeOf)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 1234;
        Assert.That(numero, Is.TypeOf<int>());
    }
}
```

NUnit

- IResolveConstraint (Is AnyOf)

O método AnyOf vai receber um array de objetos como parâmetro e retornar uma restrição para validar se o valor recebido é igual a algum dos valores definidos no array de objetos.

NUnit

- IResolveConstraint (Is AnyOf)

```
[TestFixture]
public class ClasseDeTeste
{
    [Test]
    public void MetodoDeTeste()
    {
        int numero = 12;
        Assert.That(numero, Is.AnyOf(4, 12, 98));
    }
}
```