MSTest Unit Testing Framework VB.NET Cheat Sheet

AUTOMATE THE PLANET

Test Execution Workflow

```
Imports Microsoft.VisualStudio.TestTools.UnitTesting
Namespace MSTestUnitTests
    ' A class that contains MSTest unit tests. (Required)
    <TestClass>
    Public Class YourUnitTests
        <AssemblyInitialize>
        Public Shared Sub AssemblyInit(ByVal context As TestContext)
            ' Executes once before the test run. (Optional)
        End Sub
        <ClassInitialize>
        Public Shared Sub TestFixtureSetup(ByVal context As TestContext)
            ' Executes once for the test class. (Optional)
        End Sub
        <TestInitialize>
        Public Sub Setup()
            ' Runs before each test. (Optional)
        End Sub
        <AssemblyCleanup>
        Public Shared Sub AssemblyCleanup()
            ' Executes once after the test run. (Optional)
        End Sub
        <ClassCleanup>
        Public Shared Sub TestFixtureTearDown()
            ' Runs once after all tests in this class are executed. (Optional)
            ' Not guaranteed that it executes instantly after all tests from the class.
        End Sub
        <TestCleanup>
        Public Sub TearDown()
            ' Runs after each test. (Optional)
        End Sub
            ' Mark that this is a unit test method. (Required)
        <TestMethod>
        Public Sub YouTestMethod()
            ' Your test code goes here.
        End Sub
    End Class
End Namespace
```

Installation

```
Install-Package MSTest.TestFramework
Install-Package MSTest.TestAdapter
Install-Package Microsoft.NET.Test.Sdk
```

Execute Tests in Parallel

```
<?xml version="1.0" encoding="utf-8"?>
<RunSettings>
  <MSTest>
    <Parallelize>
      <Workers>8</Workers>
      <Scope>MethodLevel</Scope>
    </Parallelize>
  </MSTest>
</RunSettings>
```

Data Driven Test CSV

```
<DataSource(
"Microsoft.VisualStudio.TestTools.DataSource.CSV",
"TestsData.csv", "TestsData#csv",
DataAccessMethod.Sequential)>
<TestMethod>
Public Sub DataDrivenTest()
    Dim valueA As Integer =
Convert.ToInt32(Me.TestContext.DataRow("valueA"))
    Dim valueB As Integer =
Convert.ToInt32(Me.TestContext.DataRow("valueB"))
    Dim expected As Integer =
Convert.ToInt32(Me.TestContext.DataRow("expectedResult"))
End Sub
```

Data Driven Test Dynamic Data

```
<DataTestMethod>
<DynamicData(NameOf(GetData),
DynamicDataSourceType.Method)>
Public Sub TestAddDynamicDataMethod(ByVal a As Integer,
ByVal b As Integer, ByVal expected As Integer)
    Dim actual = _calculator.Add(a, b)
    Assert.AreEqual(expected, actual)
End Sub
Public Shared Iterator Function GetData() As
IEnumerable(Of Object())
    Yield New Object() {1, 1, 2}
    Yield New Object() {12, 30, 42}
    Yield New Object() {14, 1, 15}
End Function
```

Attributes

NUNIT 3.X	MSTEST V2.X	XUNIT.NET 2.X	COMMENTS
<Test>	<TestMethod>	<Fact>	Marks a test method.
<TestFixture>	<TestClass>	n/a	Marks a test class.
<SetUp>	<TestInitialize>	Constructor	Triggered before every test case.
<TearDown>	<TestCleanup>	IDisposable.Dispose	Triggered after every test case.
<OneTimeSetUp>	<ClassInitialize>	IClassFixture<T>	One-time triggered method before test cases start.
<OneTimeTearDown>	<ClassCleanup>	IClassFixture<T>	One-time triggered method after test cases end.
<Ignore("reason")>	<Ignore>	<Fact(Skip="reason")>	Ignores a test case.
<Property>	<TestProperty>	<Trait>	Sets arbitrary metadata on a test.
<Theory>	<DataRow>	<Theory>	Configures a data-driven test.
<Category("")>	<TestCategory("")>	<Trait("Category", "")>	Categorizes the test cases or classes.

Data Driven Test Attributes

```
<DataRow(0, 0)>
<DataRow(1, 1)>
<DataRow(2, 1)>
<DataRow(80, 23416728348467685)>
<DataTestMethod>
Public Sub GivenDataFibonacciReturnsResultsOk(ByVal number As Integer, ByVal
result As Integer)
    Dim fib = New Fib()
    Dim actual = fib.Fibonacci(number)
    Assert.AreEqual(result, actual)
End Sub
```

Assertions

```
Assert.AreEqual(28, _actualFuel) ' Tests whether the specified values are equal.
Assert.AreNotEqual(28, _actualFuel) ' Tests whether the specified values are unequal. Same as AreEqual for numeric values.
Assert.AreSame(_expectedRocket, _actualRocket) ' Tests whether the specified objects both refer to the same object
Assert.AreNotSame(_expectedRocket, _actualRocket) ' Tests whether the specified objects refer to different objects
Assert.IsTrue(_isThereEnoughFuel) ' Tests whether the specified condition is true
Assert.IsFalse(_isThereEnoughFuel) ' Tests whether the specified condition is false
Assert.IsNull(_actualRocket) ' Tests whether the specified object is null
Assert.IsNotNull(_actualRocket) ' Tests whether the specified object is non-null
Assert.IsInstanceOfType(_actualRocket, GetType(Falcon9Rocket)) ' Tests whether the specified object is an instance of the expected type
Assert.IsNotInstanceOfType(_actualRocket, GetType(Falcon9Rocket)) ' Tests whether the specified object is not an instance of type
StringAssert.Contains(_expectedBellatrixTitle, "Bellatrix") ' Tests whether the specified string contains the specified substring
StringAssert.StartsWith(_expectedBellatrixTitle, "Bellatrix") ' Tests whether the specified string begins with the specified substring
StringAssert.Matches("(281)388-0388", "(?d{3})?-? *d{3}-? *-?d{4}") ' Tests whether the specified string matches a regular expression
StringAssert.DoesNotMatch("281)388-0388", "(?d{3})?-? *d{3}-? *-?d{4}") ' Tests whether the specified string does not match a regular expression
CollectionAssert.AreEqual(_expectedRockets, _actualRockets) ' Tests whether the specified collections have the same elements in the same order
and quantity.
CollectionAssert.AreNotEqual(_expectedRockets, _actualRockets) ' Tests whether the specified collections does not have the same elements or the
elements are in a different order and quantity.
CollectionAssert.AreEquivalent(_expectedRockets, _actualRockets) ' Tests whether two collections contain the same elements.
CollectionAssert.AreNotEquivalent(_expectedRockets, _actualRockets) ' Tests whether two collections contain different elements.
CollectionAssert.AllItemsAreInstancesOfType(_expectedRockets, _actualRockets) ' Tests whether all elements in the specified collection are
instances of the expected type
CollectionAssert.AllItemsAreNotNull(_expectedRockets) ' Tests whether all items in the specified collection are non-null
CollectionAssert.AllItemsAreUnique(_expectedRockets) ' Tests whether all items in the specified collection are unique
CollectionAssert.Contains(_actualRockets, falcon9) ' Tests whether the specified collection contains the specified element
CollectionAssert.DoesNotContain(_actualRockets, falcon9) ' Tests whether the specified collection does not contain the specified element
CollectionAssert.IsSubsetOf(_expectedRockets, _actualRockets) ' Tests whether one collection is a subset of another collection
CollectionAssert.IsNotSubsetOf(_expectedRockets, _actualRockets) ' Tests whether one collection is not a subset of another collection
Assert.ThrowsException(Of ArgumentNullException)(Function() New Regex(Nothing)) ' Tests whether the code specified by delegate throws exact
given exception of type T
```