UNIT TEST ATTRIBUTES CHEAT SHEET

ASTAQC

Installation

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

Test Execution Workflow

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
namespace MSTestUnitTests
{
  // A class that contains MSTest unit tests. (Required)
  [TestClass]
  public class YourUnitTests
  {
    [AssemblyInitialize]
    public static void AssemblyInit(TestContext context)
    {
      // Executes once before the test run. (Optional)
    }
    [ClassInitialize]
    public static void TestFixtureSetup(TestContext context)
    {
      // Executes once for the test class. (Optional)
    }

    [TestInitialize]
    public void Setup()
    {
      // Runs before each test. (Optional)
    }
    [AssemblyCleanup]
    public static void AssemblyCleanup()
    {
      // Executes once after the test run. (Optional)
    }

    [ClassCleanup]
    public static void TestFixtureTearDown()
    {
      // Runs once after all tests in this class are executed. (Optional)
      // Not guaranteed that it executes instantly after all tests from the class.
    }

    [TestCleanup]
    public void TearDown()
    {
      // Runs after each test. (Optional)
    }
    // Mark that this is a unit test method. (Required)
    [TestMethod]
    public void YouTestMethod()
    {
      // Your test code goes here.
    }
  }
}
```

Attributes

NUnit	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 void GivenDataFibonacciReturnsResultsOk(int number, int result)
{
  var fib = new Fib();
  var actual = fib.Fibonacci(number);
  Assert.AreEqual(result, actual);
}
```

Data Driven Test CSV

```
[DataSource("Microsoft.VisualStudio.TestTools.DataSource.CSV", "TestsData.csv",
"TestsData#csv", DataAccessMethod.Sequential)]
[TestMethod]
public void DataDrivenTest()
{
  int valueA = Convert.ToInt32(this.TestContext.DataRow["valueA"]);
  int valueB = Convert.ToInt32(this.TestContext.DataRow["valueB"]);
  int expected = Convert.ToInt32(this.TestContext.DataRow["expectedResult"]);
}
```

Data Driven Test Dynamic Data

```
[DataTestMethod]
[DynamicData(nameof(GetData), DynamicDataSourceType.Method)]
public void TestAddDynamicDataMethod(int a, int b, int expected)
{
  var actual = _calculator.Add(a, b);
  Assert.AreEqual(expected, actual);
}
public static IEnumerable<object[]> GetData()
{
  yield return new object[] { 1, 1, 2 };
  yield return new object[] { 12, 30, 42 };
  yield return new object[] { 14, 1, 15 };
}
```

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, typeof(Falcon9Rocket)); // Tests whether the specified object is an instance of the expected type
Assert.IsNotInstanceOfType(_actualRocket, typeof(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<ArgumentNullException>(() => new Regex(null)); // Tests whether the code specified by delegate throws exact given exception of type T
```

Execute Tests in Parallel

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