C# Unit Test Template Classes

If you are using MSTest as your testing framework, and want to leverage the test functionality from OGA.Testing.Lib, here are the minimal classes and structure you will need.

For a cheat-sheet on MSTest framework usage, see this: Unit Test Cheat Sheet

Dependencies

Aside from the libraries that are automatically included in an MSTest project, these will be required:

- NLog
- OGA.SharedKernel
- OGA.Common.NET or OGA.Common.NETCore
- OGA.Testing.Lib

Assembly Test Class

The template assembly test class provides common methods like logging.
Create a similar class in your test project, so that logging is automatically enabled.

The latest class definition is here: TestTemplate_Assembly.cs
Here's the pasted class content:

```
[TestClass]
public class TestTemplate_Assembly : OGA.Testing.Lib.TestAssembly_Base
{
    #region Test Assembly Setup / Teardown

    /// <summary>
    /// This initializer calls the base assembly initializer.
    /// </summary>
    /// <param name="context"></param>
```

```
    [AssemblyInitialize]
    static public void TestAssembly_Initialize(TestContext context)
    {
        TestAssemblyBase_Initialize(context);
    }

    /// <summary>
    /// This cleanup method calls the base assembly cleanup.
    /// </summary>
    [AssemblyCleanup]
    static public void TestAssembly_Cleanup()
    {
        TestAssemblyBase_Cleanup();
    }

    #endregion
}
```

It’s simple enough, to just copy the above class as your test project’s assembly test class.

Each Test Class

Each top-level test class in your project will require a few elements, in order to access common function.
This is because the MSTest framework doesn’t readily recognize test attributes in base classes.
So, that means several methods must be included in your top-level test class.

The below template test class shows what elements are needed in each of your top-level test classes.

For each test class you create, ensure it includes the methods and properties of the below class template:

The latest class definition is here: TestTemplate_Tests.cs

```
/// <summary>
/// Base template for unit tests.
/// Provides an example for usage of OGA.Testing.Lib, when making a unit test.
/// NOTE: There is a good bit of ceremony required in a top-level test class.
///       This is because the MSTest framework doesn't handle derived test classes well.
```

```
/// NOTE: For proper logging, be sure to include a test class that derives from TestAssembly_Base,
///       or make sure its setup and cleanup methods are called.
/// </summary>
[TestCategory(Test_Types.Unit_Tests)]
[TestClass]
public class TestTemplate_Tests : Test_Base_abstract
{
    #region Setup

    /// <summary>
    /// This will perform any test setup before the first class tests start.
    /// This exists, because MSTest won't call the class setup method in a base class.
    /// Be sure this method exists in your top-level test class,
    ///     and that it calls the corresponding test class setup method of the base.
    /// </summary>
    [ClassInitialize]
    static public void TestClass_Setup(TestContext context)
    {
        TestClassBase_Setup(context);
    }
    /// <summary>
    /// This will cleanup resources after all class tests have completed.
    /// This exists, because MSTest won't call the class cleanup method in a base class.
    /// Be sure this method exists in your top-level test class,
    ///    and that it calls the corresponding test class cleanup method of the base.
    /// </summary>
    [ClassCleanup]
    static public void TestClass_Cleanup()
    {
        TestClassBase_Cleanup();
    }

    /// <summary>
    /// Called before each test runs.
    /// Be sure this method exists in your top-level test class, and that it calls the corresponding test setup method
of the base.
    /// </summary>
    [TestInitialize]
    override public void Setup()
    {
```

```
        //// Push the TestContext instance that we received at the start of the current test,
        ////    into the common property of the test base class...
        //Test_Base.TestContext = TestContext;

        base.Setup();

        // Runs before each test. (Optional)
    }

    /// <summary>
    /// Called after each test runs.
    /// Be sure this method exists in your top-level test class,
    ///    and that it calls the corresponding test cleanup method of the base.
    /// </summary>
    [TestCleanup]
    override public void TearDown()
    {
        // Runs after each test. (Optional)

        base.TearDown();
    }

    #endregion


    #region Test Methods

    [TestMethod]
    public async Task Test_1_1_1()
    {
        OGA.SharedKernel.Logging_Base.Logger_Ref?.Debug(
            $"{TestContext.ManagedType}:{TestContext.TestName} - " +
            "Test started.");

        if (1 != 1)
            Assert.Fail("Wrong Value");
    }

    #endregion
}
```

It’s simple enough to copy the properties and methods from the above class, into each of your top-level test classes.

Revision #2
Created 11 May 2025 07:06:23 by glwhite
Updated 11 May 2025 07:09:58 by glwhite