Automated tests often mirror the structure of the tested code. For example, we recommend creating at least one test project for every custom project that you have in your Kentico solution. You can have any number of test projects in the solution.

The following steps describe how to create test projects using the Kentico **CMS.Tests** library:

1. Open your Kentico solution in Visual Studio (using the **WebSite.sln** or **WebApp.sln** file).
2. Add a new project to the solution:
   - **Important**: The name of the project must end with the **.Tests** suffix.
   - Select the **Class Library** project template if you plan to use the NUnit framework.
   - Select the **Unit Test Project** template if you plan to use the Microsoft unit test framework (in the New Project dialog box, expand the **Installed** node, choose the language that you want to use for your test project, and then choose **Test**).

   Microsoft unit test framework projects in Visual Studio 2017

   After creating a Unit Test Project in Visual Studio 2017, you need to manually update the *UnitTestFramework* assembly to use the Visual Studio Program Files instead of the default MSTest NuGet package.

   - Uninstall the default **MSTest.TestFramework** NuGet package from the test project (right-click the project in the *Solution Explorer* and select *Manage NuGet Packages*).
   - Add a new reference to the **VisualStudio.QualityTools.UnitTestFramework** assembly on the **Assemblies -> Extensions** tab in the **Reference Manager** dialog.

3. Right-click the test project in the **Solution Explorer** and select **Manage NuGet Packages**.
4. Search for the **Kentico.Libraries.Tests** package.
5. Install the *Kentico.Libraries.Tests* version that matches the hotfix version of your Kentico instance (with all dependencies).

   Maintaining Kentico.Libraries.Tests

   If you upgrade or hotfix the related Kentico project to a newer version, you also need to update the *Kentico.Libraries.Tests* NuGet package to a matching version in your test projects.

6. Add references from the test project to all projects whose code you wish to test, and any other required project.

Now you can write test classes inside the new test project. You can use the Visual Studio Test Explorer to execute your tests.

NUnit test class example

```
using CMS.Tests;
using NUnit.Framework;

namespace UnitTestExample
{
        [TestFixture]
        public class MyTests : UnitTests
        {
                [SetUp]
                public void MyTestSetUp()
                {
                        // Setup executed before each test in this test class
                }

                [Test]
                public void MyTest()
                {
                        // Test execution
                }
        }
}
```

Note: To run NUnit tests, you need to install the NUnit3TestAdapter NuGet package (recommended) or the NUnit 3 Test Adapter Visual Studio extension.

For additional information about writing tests, see the following pages:

- Faking Info and Provider objects in unit tests
- Creating integration tests with a connection string
- Creating isolated integration tests