Test automation is an important part of every development project and is essential for agile projects. Automated tests help you improve productivity, reduce the testing phase of the development cycle, and increase code predictability and quality.

When developing custom functionality for Kentico projects, we recommend that you create projects with automated tests to cover the given code.

Kentico provides the **CMS.Tests** library, which simplifies writing of automated tests for code that uses the Kentico API. The library contains base classes for unit and integration tests, as well as helper classes for faking *Info* and *Provider* objects. You can obtain the library by installing the **Kentico.Libraries.Tests** NuGet package into your test projects.

Supported testing frameworks

All of the functionality in the *CMS.Tests* library is fully compatible with both the NUnit framework and **Microsoft unit test** frameworks. The examples in this documentation are written for the NUnit framework.

Types of tests

You can create the following general types of tests (categorized according to the way the tests work with database data):

Unit tests

Unit tests are able to run without external resources, such as a database. Try to write most of your tests as unit tests, since they are used to examine relatively small pieces of code. Unit tests execute much faster than the other types of automated tests.

Use fake *Info* and *Provider* objects in your unit tests to avoid accessing the database. For more information, see Faking Info and Provider objects in unit tests.

Integration tests

Integration tests can access a database provided by a connection string. Use integration tests when you need to read the data from the database (unless you are absolutely sure you clean up everything properly after the test). Integration tests are significantly slower than unit tests.

See: Creating integration tests with a connection string

Isolated integration tests

Isolated integration tests automatically create their own database before the test execution and clean up the database after the test is finished. Use isolated integration tests for complex testing that requires writing to the database, for example if the cleanup after an integration test is difficult. Isolated integration tests are the slowest of the three types of automated tests.

See: Creating isolated integration tests

API differences in test code

By default, all Kentico API executed within the body of tests that inherit from the *CMS.Tests* base classes runs without additional logging operations. For example, this includes logging of staging tasks, web farm synchronization tasks, etc. Such operations are typically not relevant for the results of tests and unnecessarily reduce test performance.

If you wish to run these additional operations within your tests, you need to explicitly enable them for blocks of code using the corresponding properties of **CMSActionContext**.

Example

```
using CMS.Base;

...

using (new CMSActionContext { LogSynchronization = true, LogWebFarmTasks = true })
{
        // Test execution
}
```