



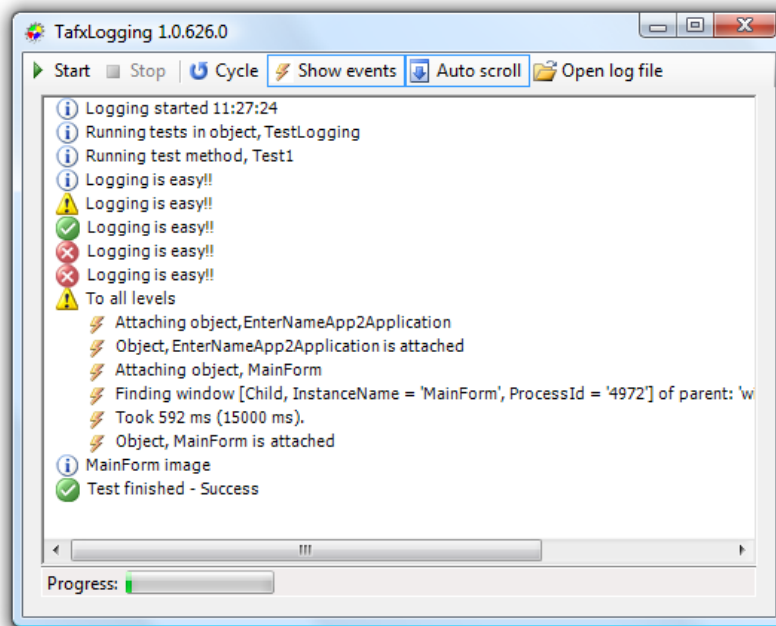
Test Automation FX

UI test automation for .NET developers

Running TAFX tests

When you create a TAFX application this will by default result in an executable showing the TAFX Test Runner window when started.

To start the test you click the **Start** button on the toolbar.



Picture 1 - The default Test Runner window shown from a TAFX test application.

To use the default Test Runner as well as starting the tests automatically without having to press the **Start** button, you can start your test executable with the command line switch – `RunTest`. This will show the Test Runner and then start the test directly.

When the tests have run to an end, the Test Runner exits with an exit code describing the outcome of the test (0 = Success, 1= Failure).

```
\TafxApplication1.exe -RunTest
```

The log from the tests is by default saved to an xml file in a folder named LogFiles. The default location for the LogFiles folder is the same location as the test application executable.

Running test with NUnit or VSTest

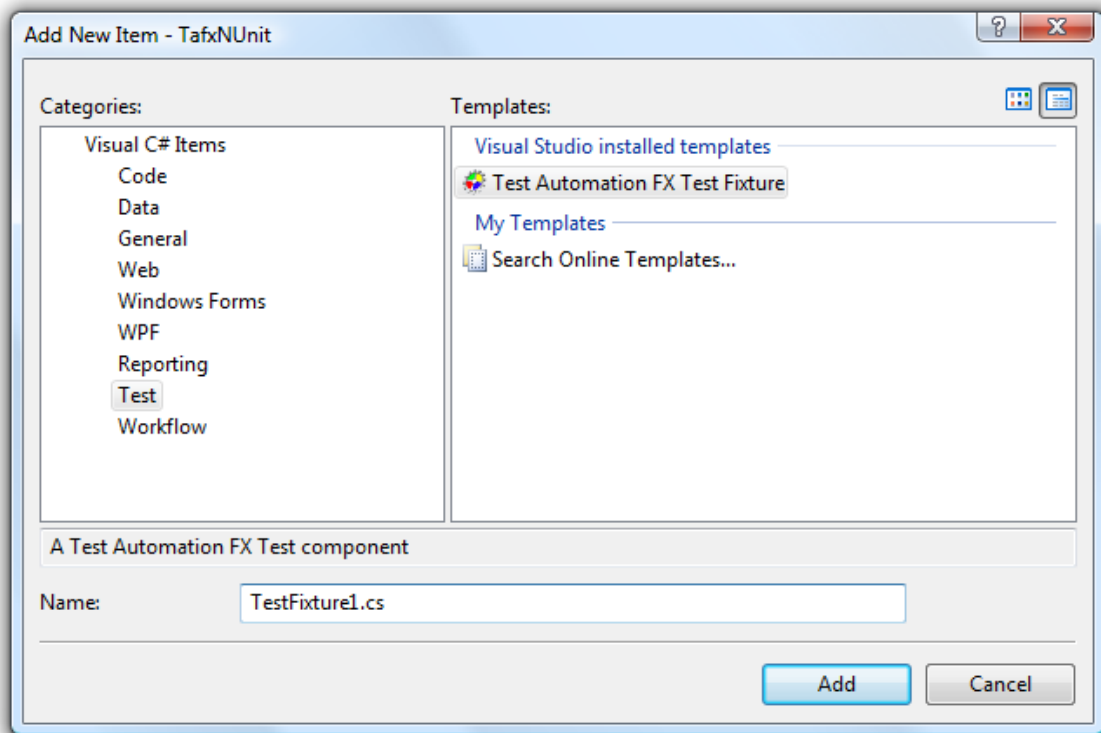
It is also easy running your tests with NUnit or the Visual Studio Unit Test framework. Just set up your unit test library as usual. Then add a 'Tafx TestFixture' item from the 'Add Item' dialog to the project.



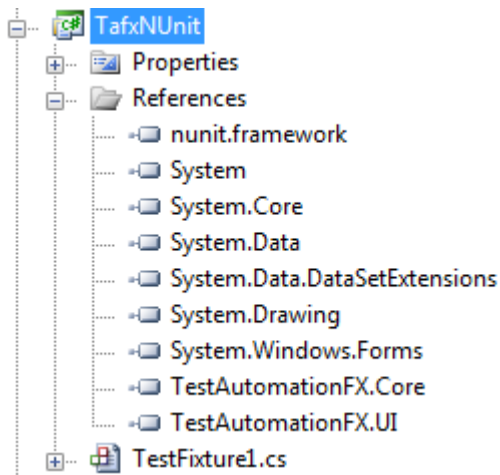


Test Automation FX

UI test automation for .NET developers



Picture 2 - To add a TAFX TestFixture to any project select the Test group in the Add New item



Picture 3. A class library project with references to NUnit and TAFX will enable UI tests to be run from any NUnit test runner.

Then you can simply change the `UITestFixture` attribute of your `TestFixture` class (derived from `UIMap`) to the one used by the unit testing framework (you can also leave the `UITestFixture` attribute if you like, but then the tests methods you record will be decorated with both the `UITest` and the `Test` attribute).





Test Automation FX

UI test automation for .NET developers

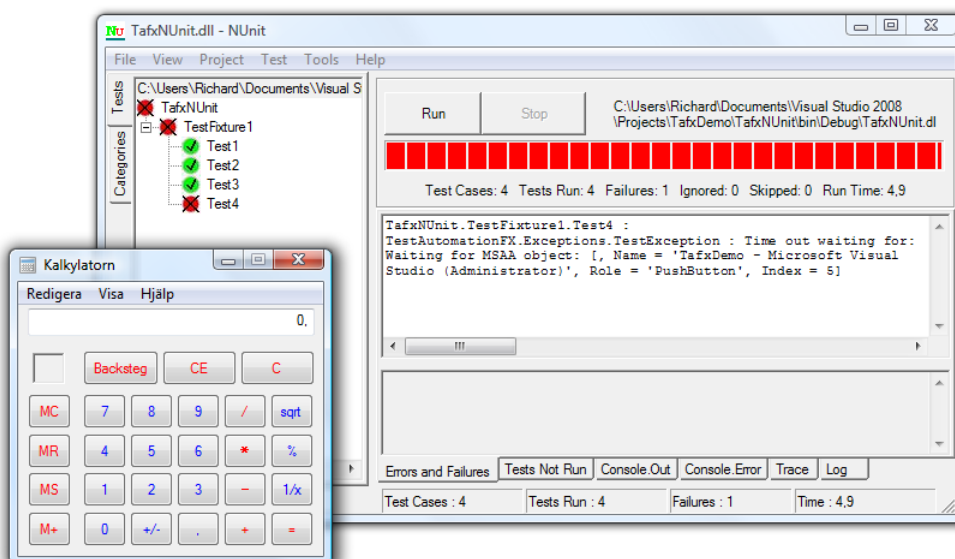
```
[SetUp]
public void Setup()
{
    ButtonC.Click();
}

[Test()]
public void Test1()
{
    Button1.Click();
    Button2.Click();
    Button3.Click();
    ButtonEquals.Click();
}

[Test()]
public void Test2()
{
    Button4.Click();
    Button2.Click();
    Button3.Click();
}
```

Picture 4. Two NUnit tests in a TAFX TestFixture that tests the calculator. The Setup attribute can be used to setup prerequisites for the tests (in this case to clear the Calculator with a click on the 'C' button).

When tests are recorded TAFX will automatically detect which Test attribute it should decorate the created method with, depending on the class attribute. I.e. if you are using the VS unit testing framework and record a test in a testfixture with a TestClass attribute, the method will be decorated with TestMethod. When you execute your unit tests the new method will be detected and run as expected. Any test exceptions will be reported to the unit test runner.



Picture 5. The NUnit test runner UI (version 2.4) running some TAFX UI tests. The test 'Test4' has thrown a TestException due to an object that cannot be found.

