



Quick Start Guide

for Windows

Examples with Visual Studio & NUnit

Some Useful Definitions

- **AUT:** Application Under Test
- **AUT Launch File:** the configuration file used by the Maveryx Test Automation Framework to run the related AUT
- **Keyword-driven testing:** a codeless approach to write test cases for non-programmers
- **Data-driven testing:** a methodological approach to separate test cases from test data

Requirements

To work with Maveryx, your system shall meet the following minimum requirements:

- Windows 7 or later
- Java Runtime Environment ver. 1.8.0_211
(<https://www.java.com/en/download/>)
- .NET Runtime Environment ver. 4.6 or later
(<https://www.microsoft.com/en-us/download/details.aspx?id=48130>)

Summary

- Install and configure Maveryx
- Get license key
- Run the Demo Solution
- Create and run your first test

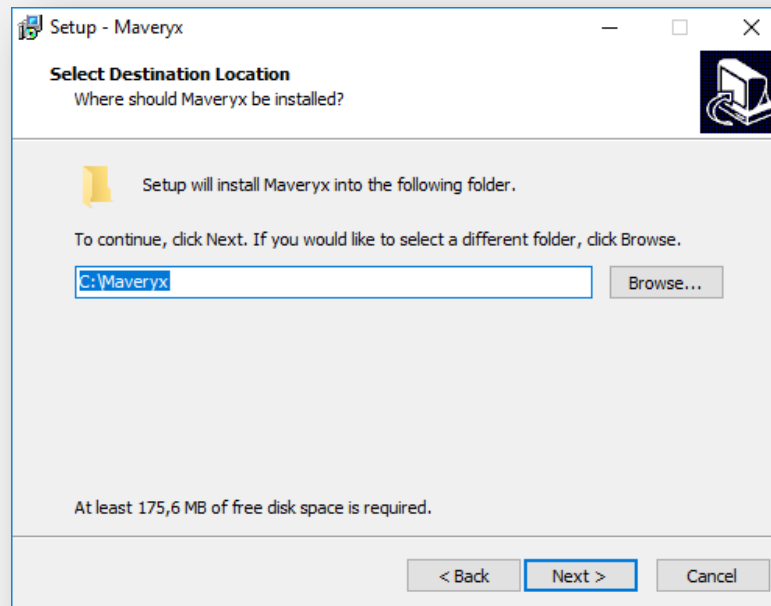
Cap I

- **Install and configure Maveryx**
- Get license key
- Run the Demo Solution
- Create and run your first test

Installation (1)

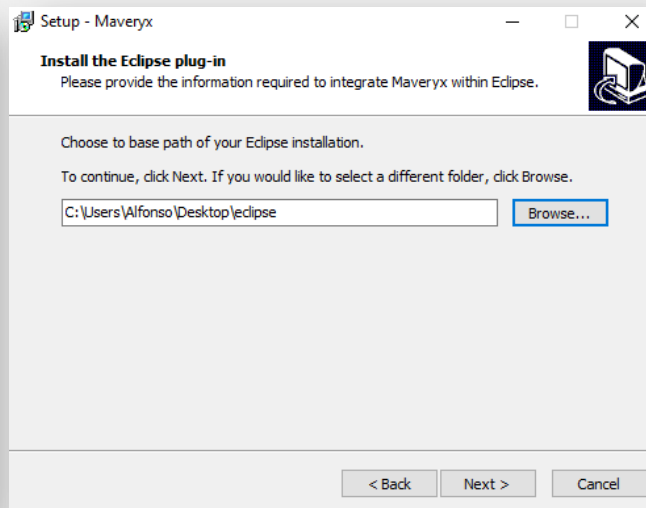
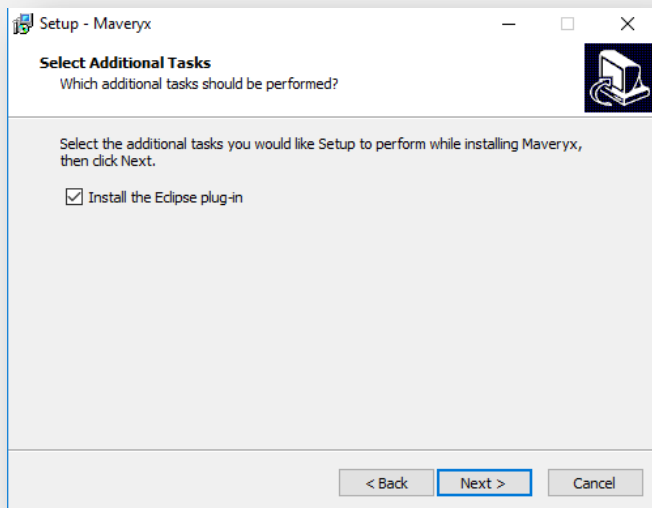
To install the Maveryx Test Automation Framework on your system, run **Maveryx_Win_2.X.y_Trial.exe** and follow the steps of the setup wizard.

Choose the directory into which you want to install the Maveryx software. You must have write permissions to this directory.



Installation (2)

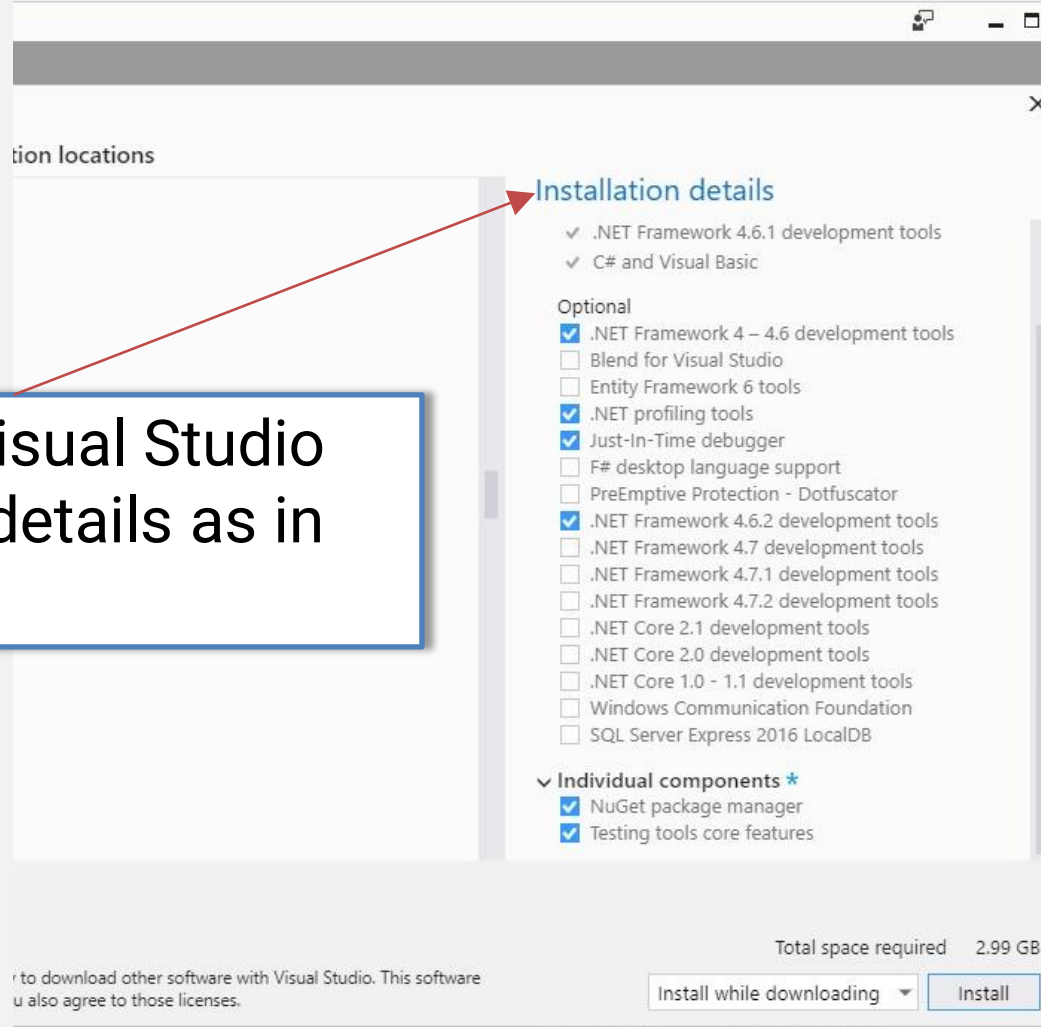
You can skip this step but, in case, you can also **Install the Eclipse plug-in** by clicking on the checkbox and selection the Eclipse path.



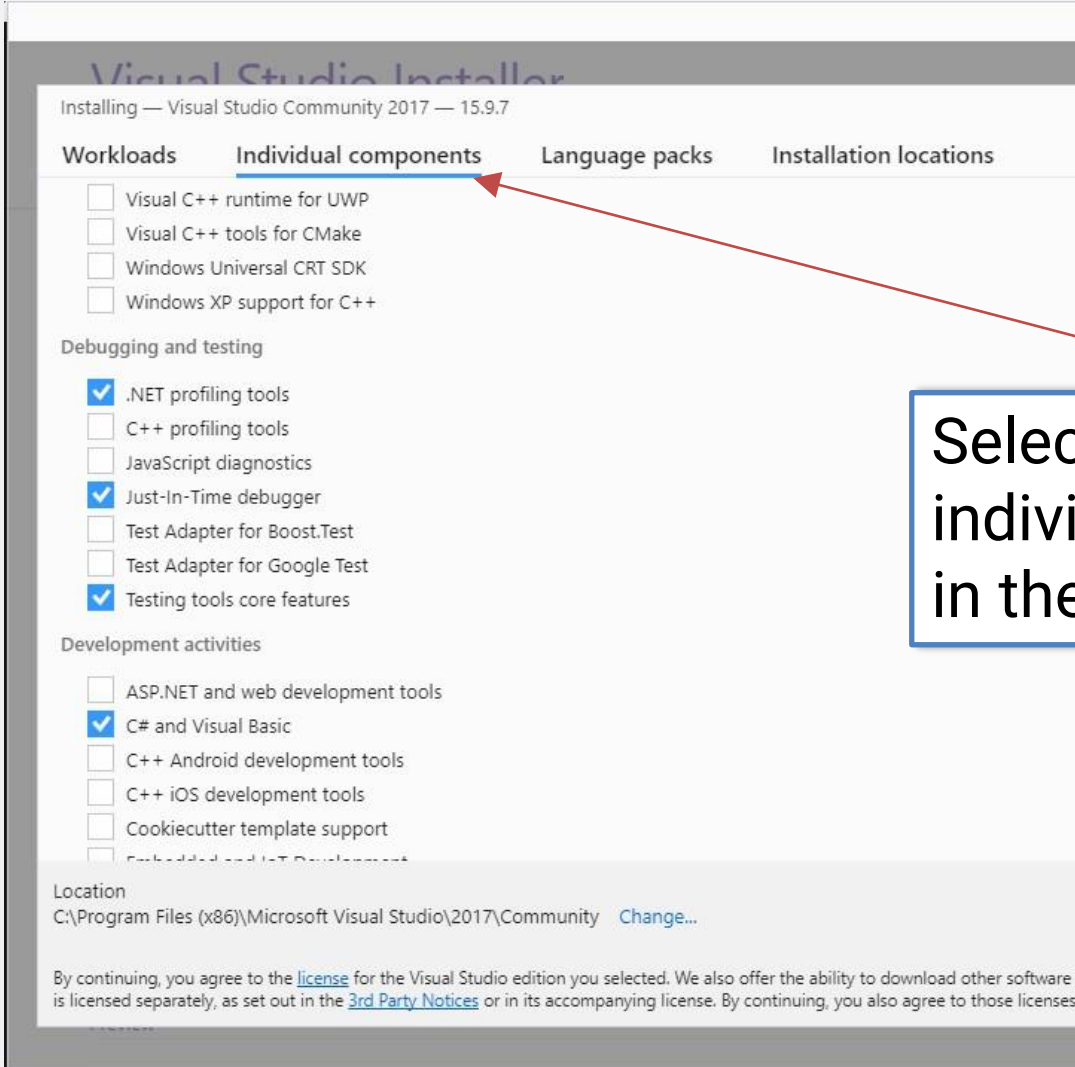
If you want to install the Maveryx Eclipse Plugin later, copy the files in **MAVERYX_HOME\tools\EclipsePlugin** folder into the **/dropins** directory of your Eclipse installation.

Setting up Visual Studio (1)

Select the Visual Studio installation details as in the example



Setting up Visual Studio (2)



Visual Studio Installer

Installing — Visual Studio Community 2017 — 15.9.7

Workloads Individual components Language packs Installation locations

☐ Visual C++ runtime for UWP

☐ Visual C++ tools for CMake

☐ Windows Universal CRT SDK

☐ Windows XP support for C++

Debugging and testing

☒ .NET profiling tools

☐ C++ profiling tools

☐ JavaScript diagnostics

☒ Just-In-Time debugger

☐ Test Adapter for Boost.Test

☐ Test Adapter for Google Test

☒ Testing tools core features

Development activities

☐ ASP.NET and web development tools

☒ C# and Visual Basic

☐ C++ Android development tools

☐ C++ iOS development tools

☐ Cookiecutter template support

☐ Embedded and IoT Development

Location

C:\Program Files (x86)\Microsoft Visual Studio\2017\Community [Change...](#)

By continuing, you agree to the [license](#) for the Visual Studio edition you selected. We also offer the ability to download other software which is licensed separately, as set out in the [3rd Party Notices](#) or in its accompanying license. By continuing, you also agree to those licenses.

Select the Visual Studio individual components as in the example

Cap II

- Install and configure Maveryx
- **Get license key**
- Run the Demo Solution
- Create and run your first test

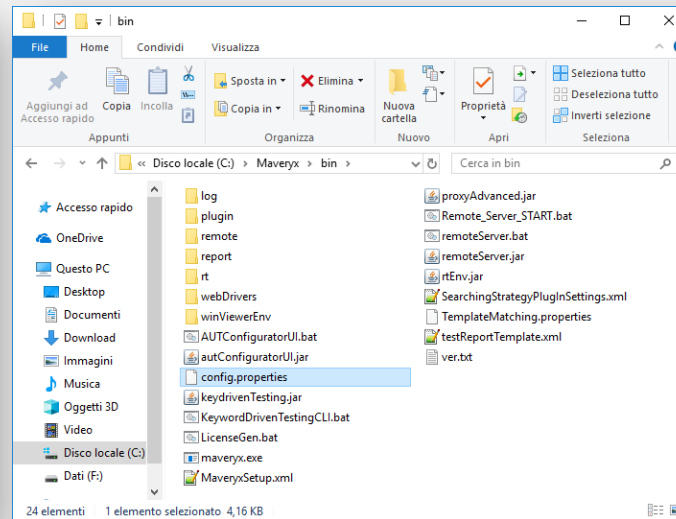
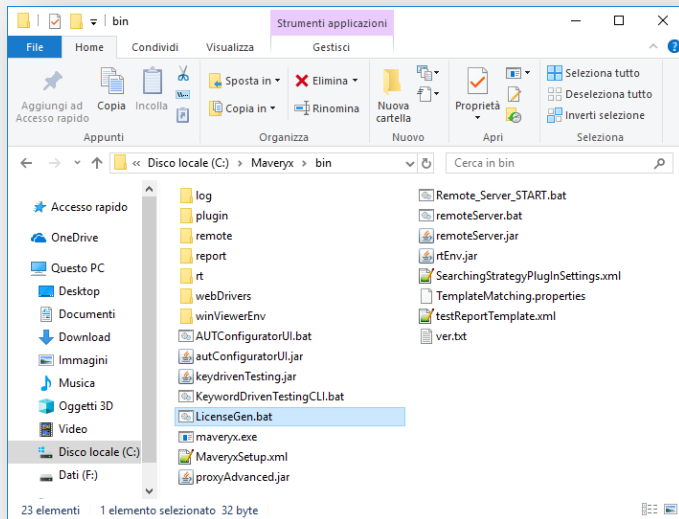
Maveryx License

- Maveryx uses a **Node-Locked** license type. A node-locked license for Maveryx lets you run the application on a specific machine or workstation. This license type is considered a single-user license, although it's bound to the machine, not the user.
- Trial versions of Maveryx (which is always licensed as Node-Locked) have a time-limited license. After it expires, you can no longer use the product.
- After a commercial license for Maveryx expires, you can continue using the product. However, you will not be able to get updates for the product and technical assistance from the Maveryx Support team.

Collecting Node Data

To generate a license open **MAVERYX_HOME/bin/** folder and run the **LicenseGen.bat** file.

This utility will automatically collect all hardware and software information needed to generate a valid Maveryx license by saving them into the **config.properties** file in **MAVERYX_HOME/bin/** folder.

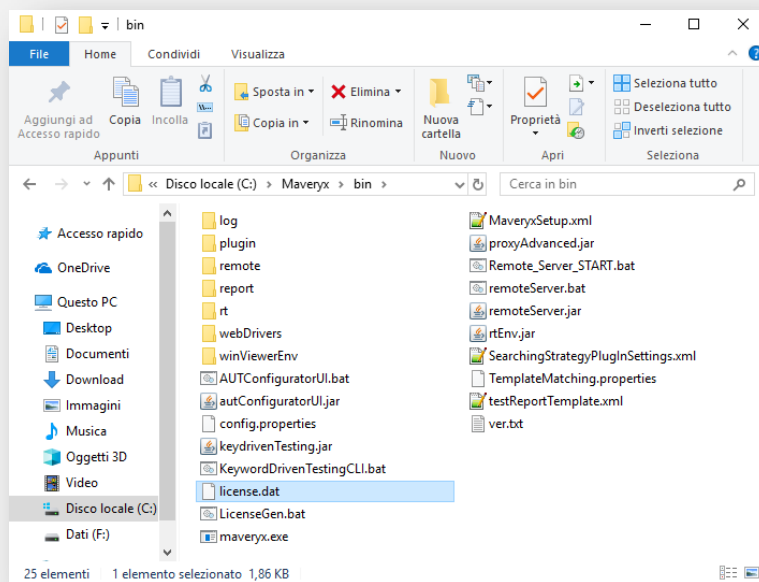


License Key

When you will have the **config.properties** file, send it by email to license.manager@maveryx.com with the subject "**License Key Request**".

In reply to your email, you will receive your license key file (**license.dat**) as attachment.

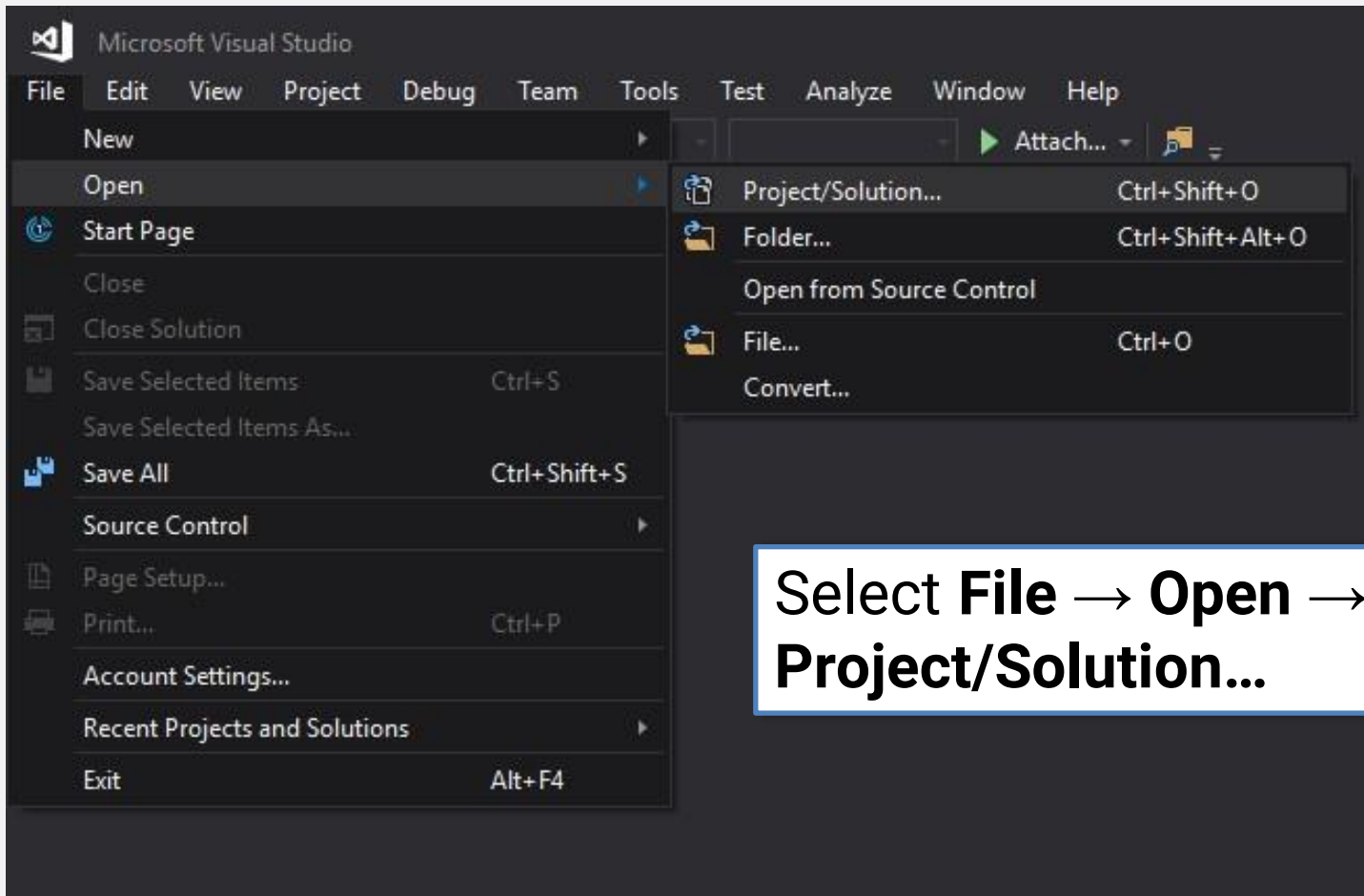
Save the **license.dat** file into the **MAVERYX_HOME/bin/** folder.



Cap III

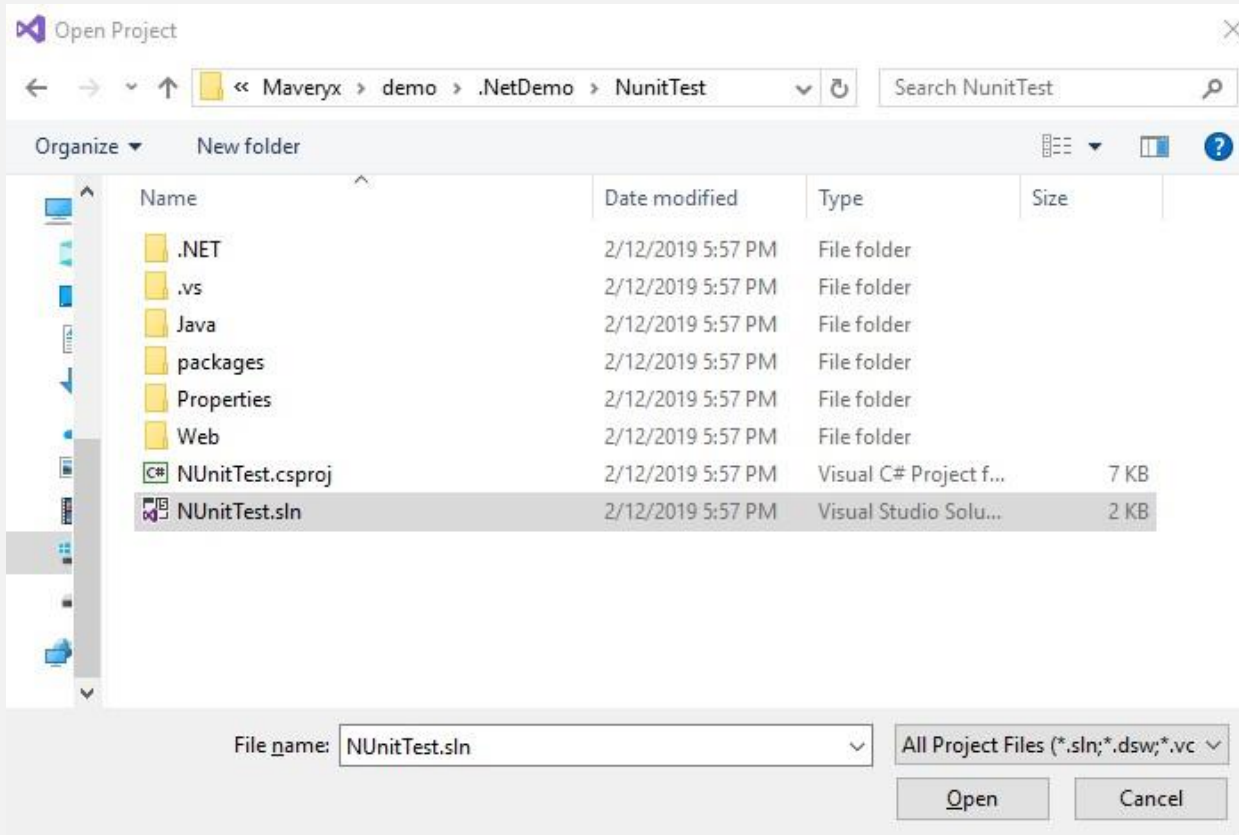
- Install and configure Maveryx
- Get license key
- **Run the Demo Solution**
- Create and run your first test

Open the Demo Solution (1)

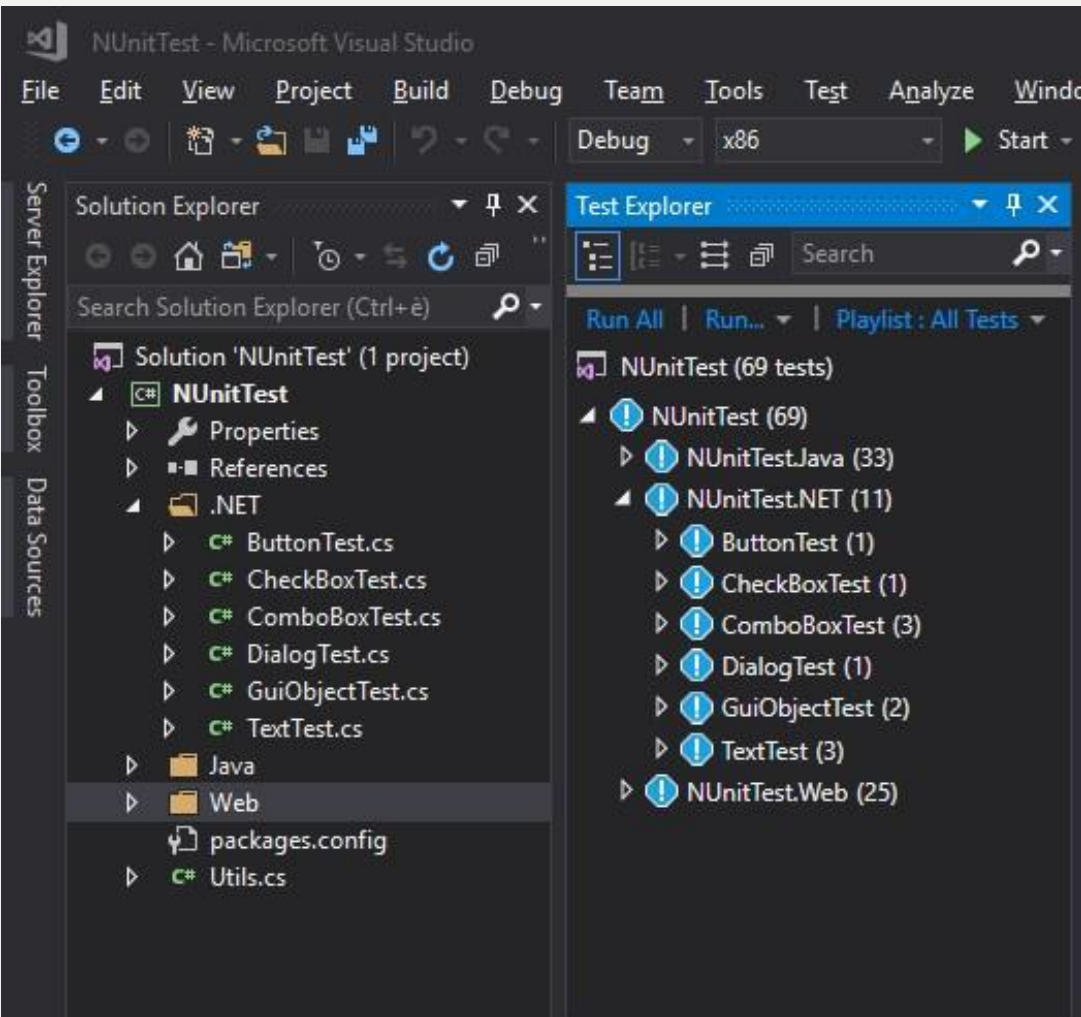


Open the Demo Solution (2)

The solution file NUnitTest.sln is located into the **MAVERYX_HOME\demo\.NetDemo\NunitTest** folder.



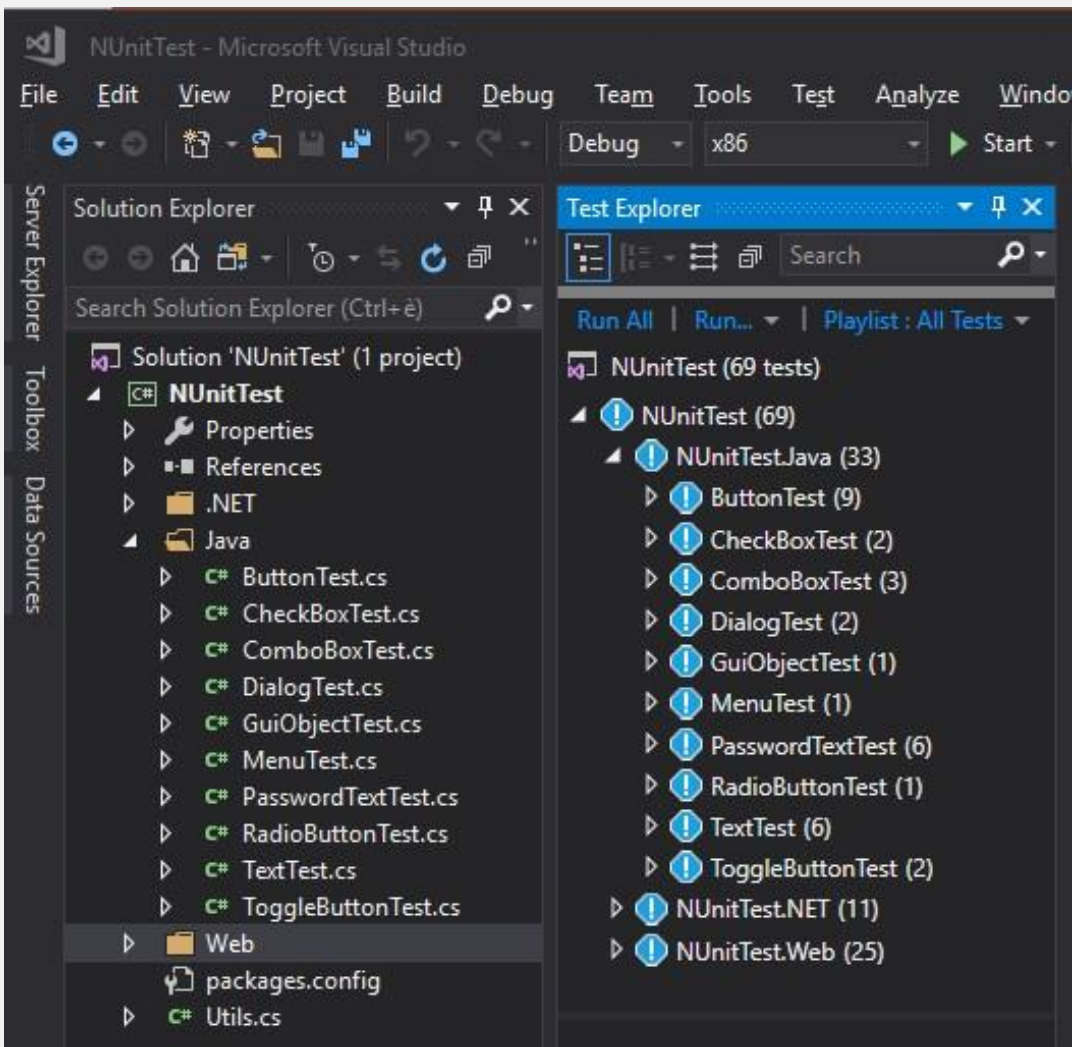
The Demo Solution (1)



.NET

The built-in Demo project has many "ready to be executed" examples for .NET Desktop Applications.

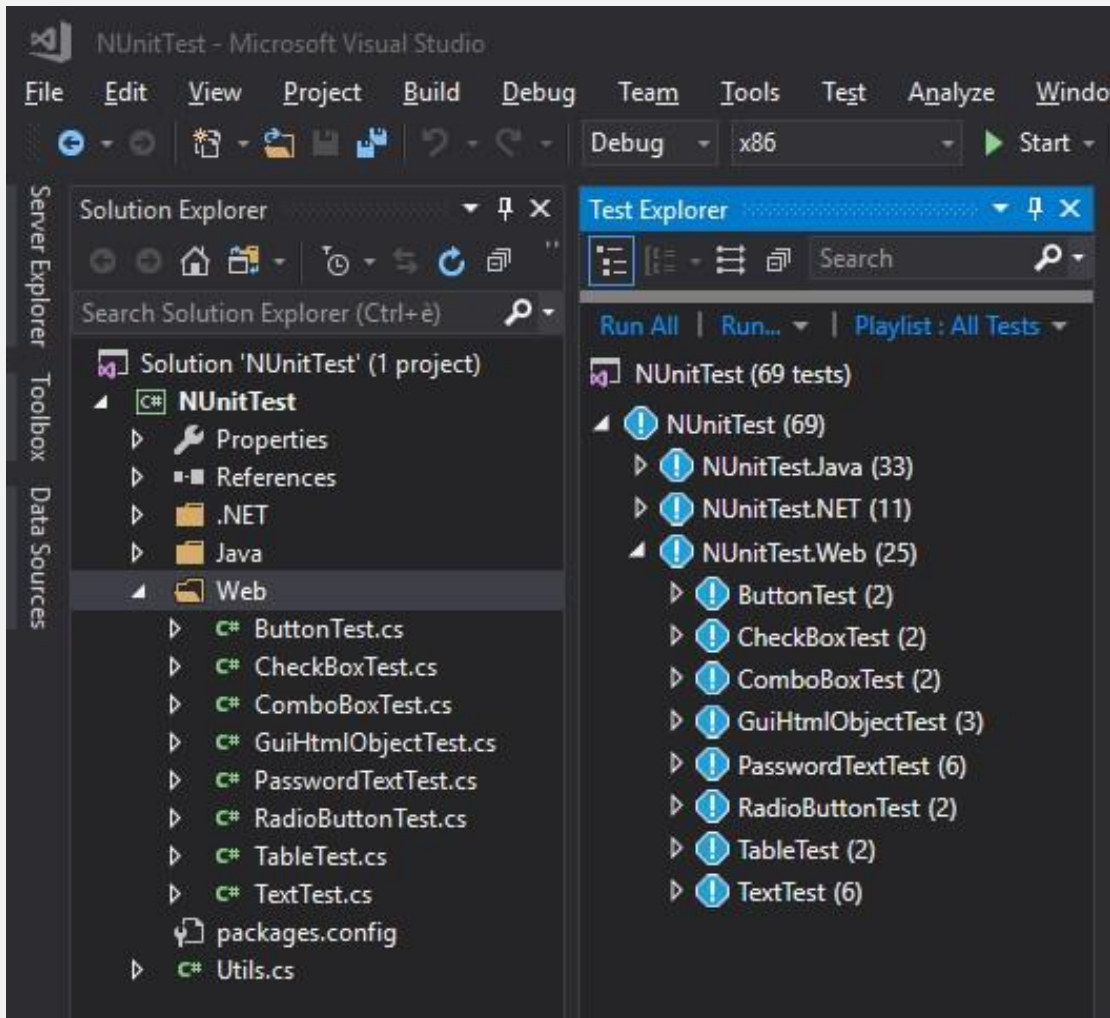
The Demo Solution (2)



Java

The built-in Demo Solution has many "ready to be executed" examples for Java Desktop Applications.

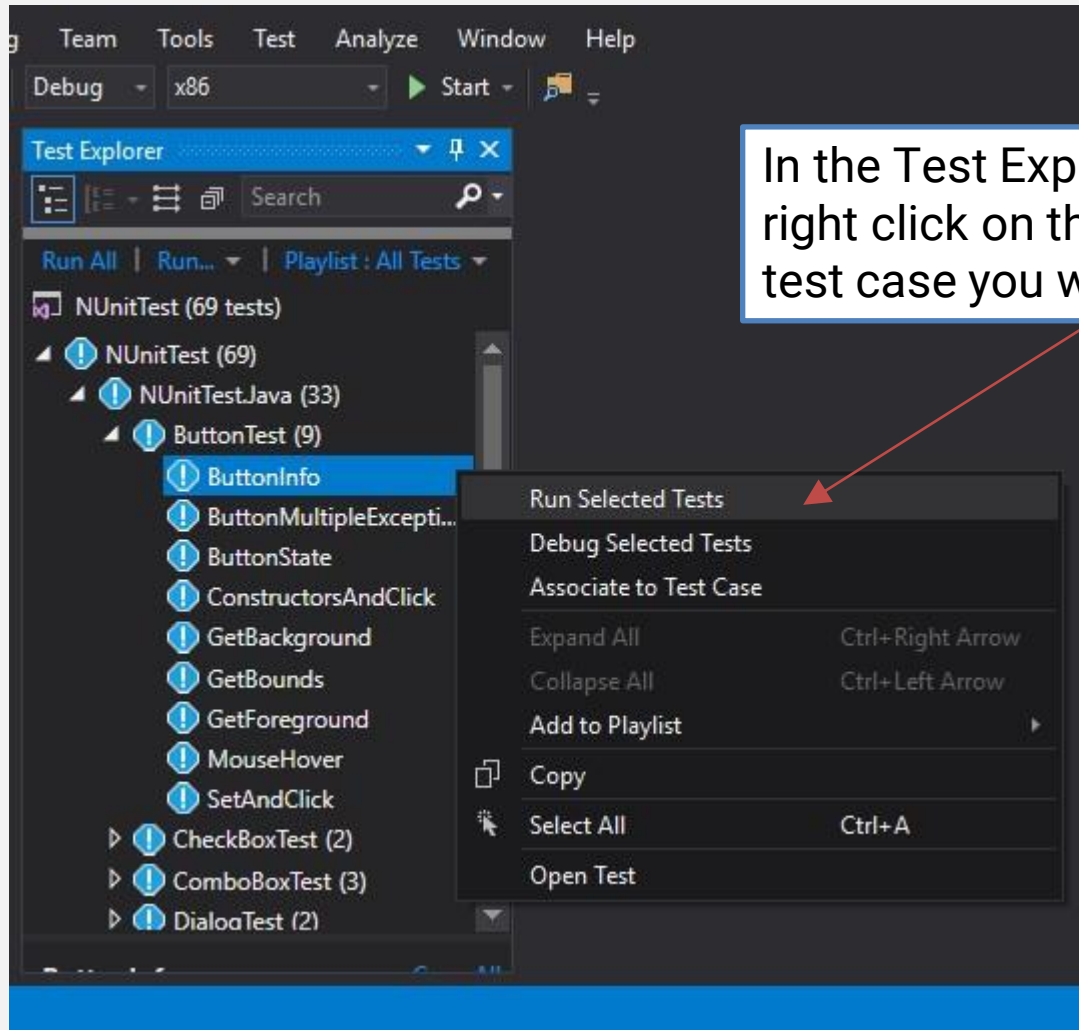
The Demo Solution (3)



Web

The built-in Demo project has many "ready to be executed" examples for Web Applications.

Run a Test Script



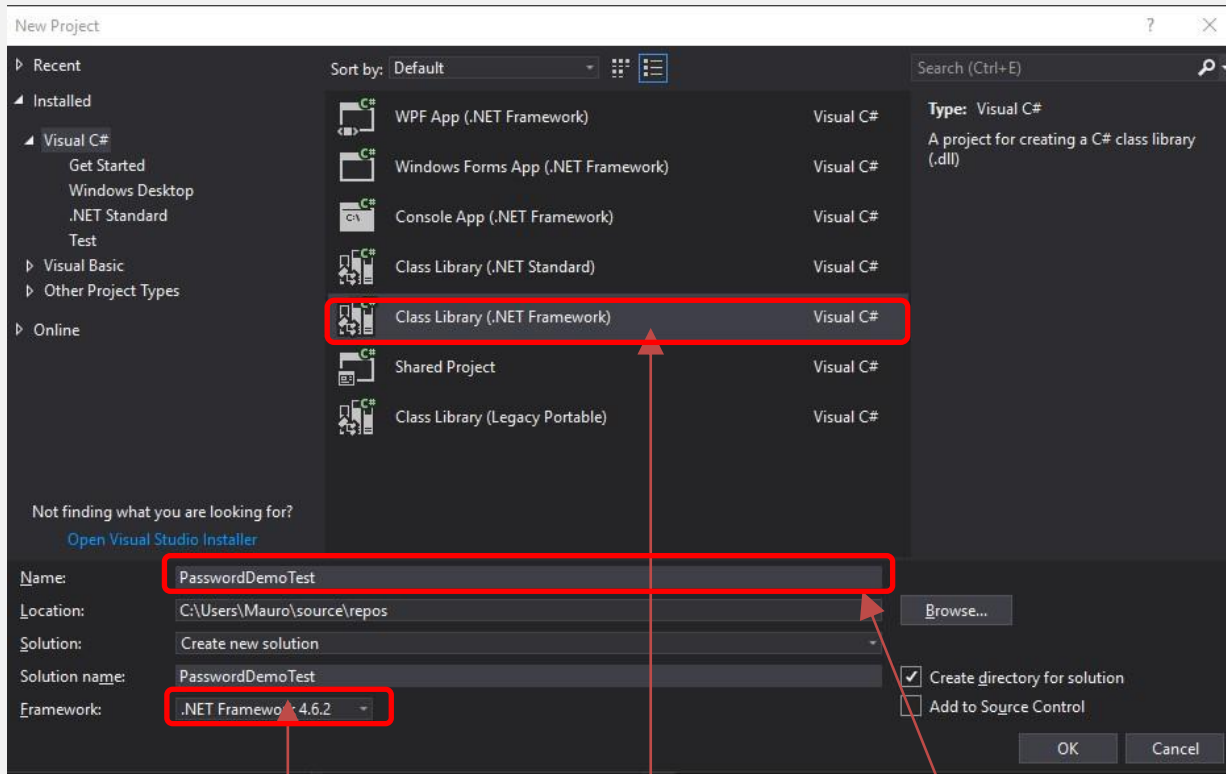
Cap IV

- Install and configure Maveryx
- Get license key
- Run the Demo Solution
- **Create and run your first test**

Create and run your first test

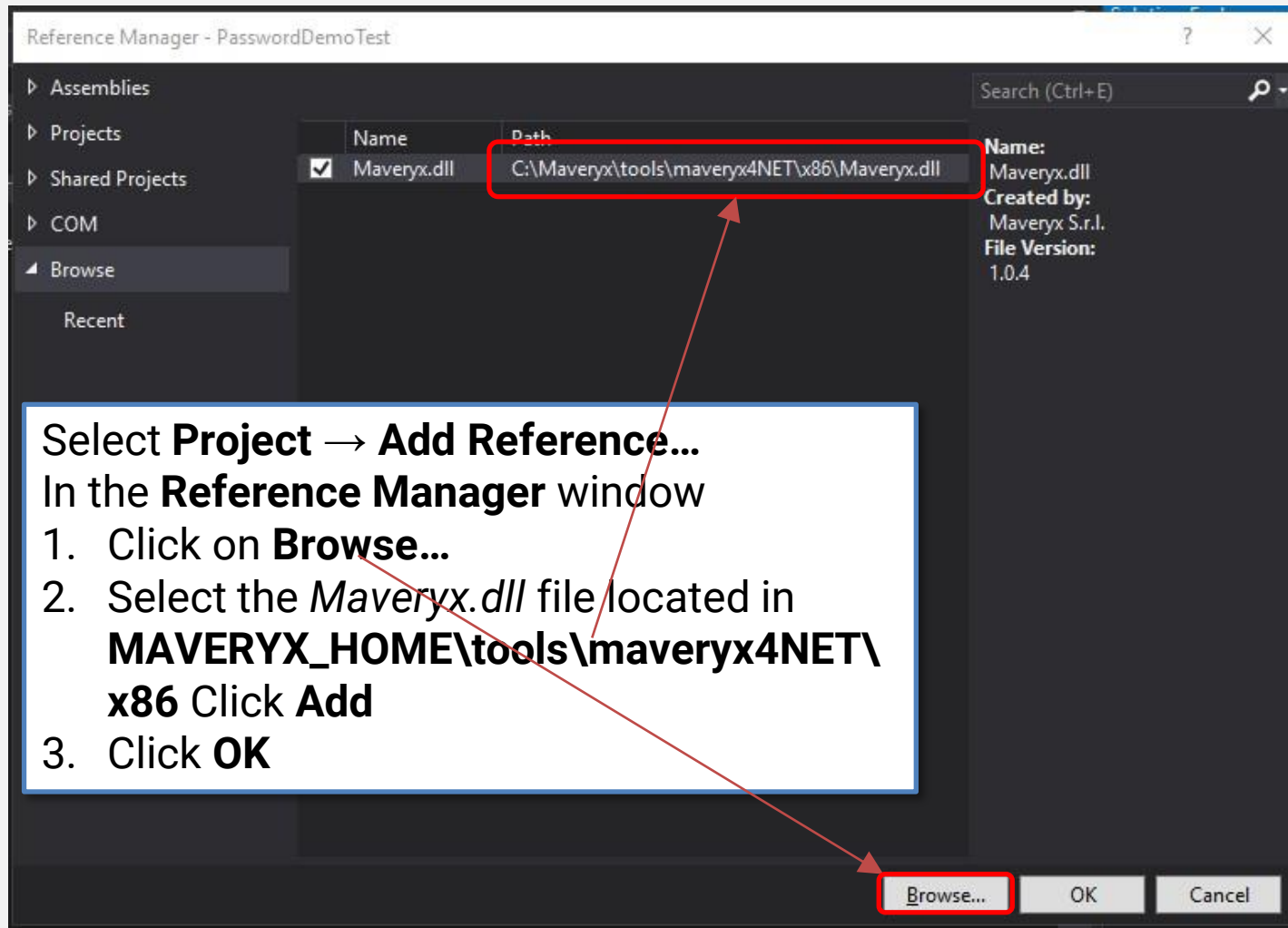
- 1. Create a new Maveryx Test Project**
2. Create a new Maveryx Test Class
3. Create the AUT Launch file
4. Write the test case
5. Run the test

Create New Test Project (1)

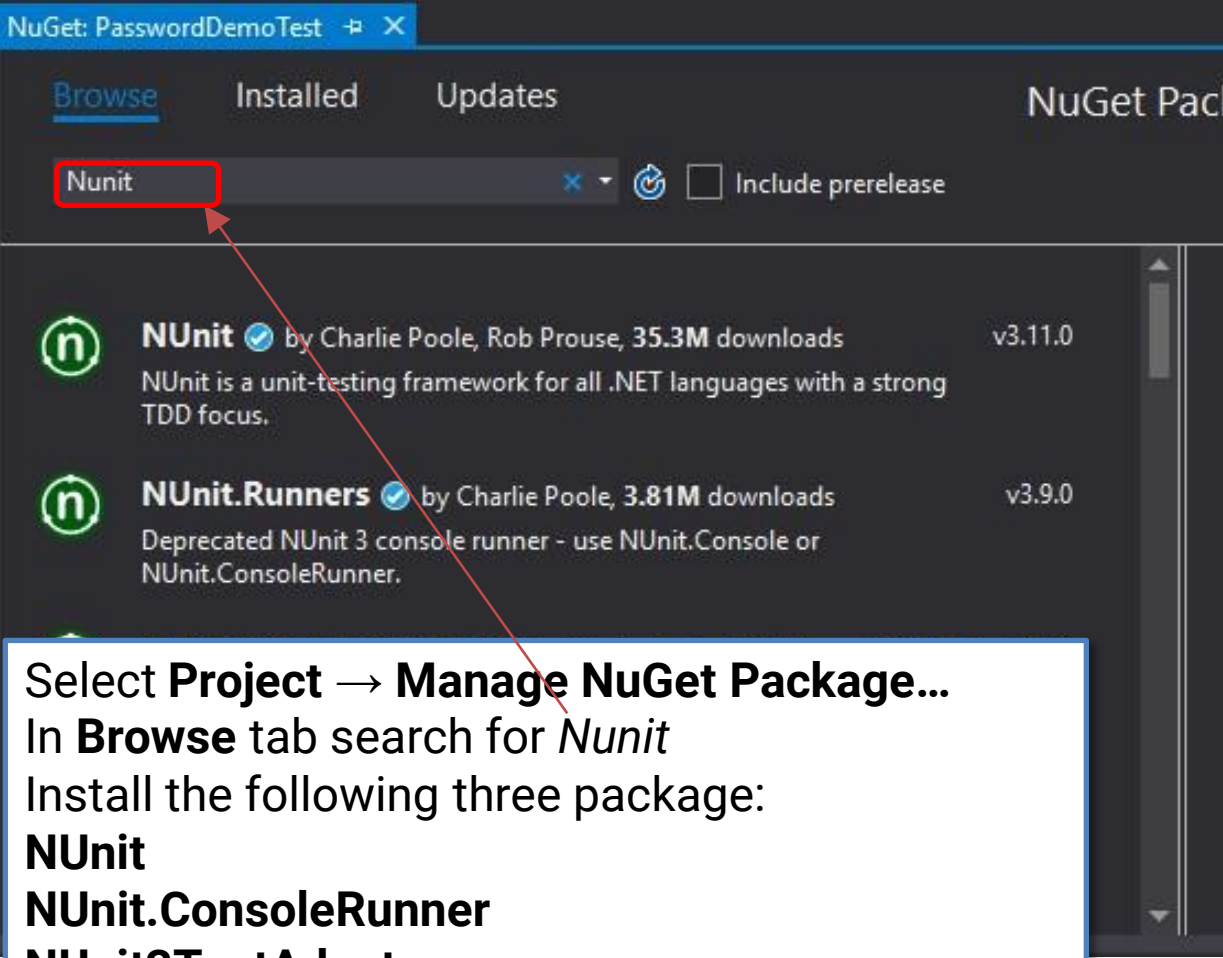


1. Select **File** → **New** → **Project...**
In the **New Project** window
 1. Select **Class Library(.NET Framework) Visual C#**
 2. Enter the Name and Solution name (e.g. "PasswordDemoTest")
 3. In the **framework** section make sure that .NET Framework 4.6.2 is selected
2. Click **OK**

Create New Test Project (2)



Create New Test Project (3)



NuGet: PasswordDemoTest

Browse Installed Updates NuGet Pack

Nunit

Include prerelease

NUnit by Charlie Poole, Rob Prouse, 35.3M downloads v3.11.0
NUnit is a unit-testing framework for all .NET languages with a strong TDD focus.

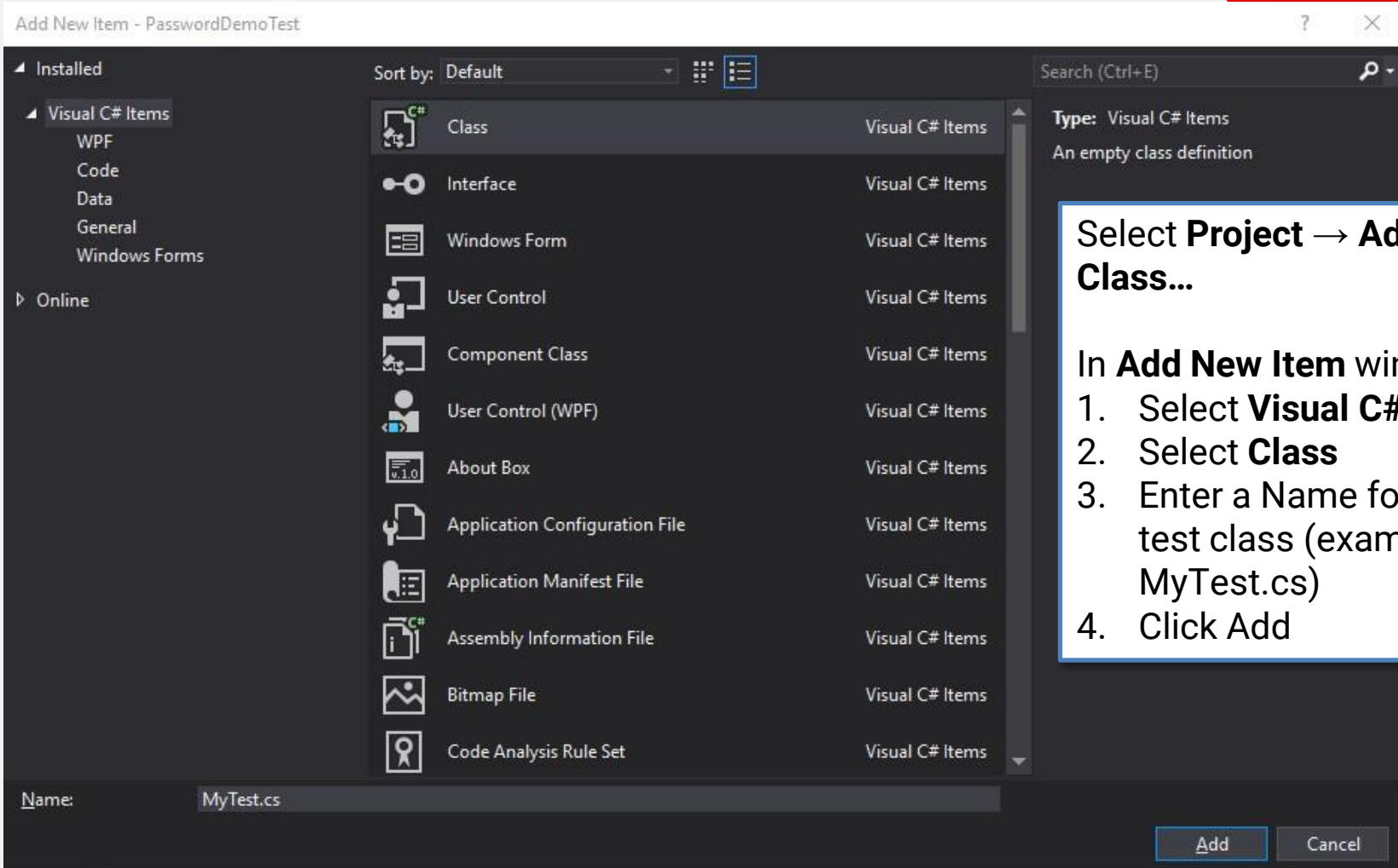
NUnit.Runners by Charlie Poole, 3.81M downloads v3.9.0
Deprecated NUnit 3 console runner - use NUnit.Console or NUnit.ConsoleRunner.

Select **Project** → **Manage NuGet Package...**
In **Browse** tab search for *Nunit*
Install the following three package:
NUnit
NUnit.ConsoleRunner
NUnit3TestAdapter

Create and run your first test

1. Create a new Maveryx Test Project
- 2. Create a new Maveryx Test Class**
3. Create the AUT Launch file
4. Write the test case
5. Run the test

Create New Test Class (1)



Create New Test Class (2)

```
using com.maveryx.bootstrap;
using NUnit.Framework;

namespace NUnitTest.Java
{
    [TestFixture]
    public class TestClass
    {
        string _autPath= "C:/Maveryx/demo/AUT/PasswordDemo.xml";

        [SetUp]
        public void SetUp()
        {
            Bootstrap4Net.Start();

            Bootstrap4Net.StartApplication(_autPath);
        }

        [TearDown]
        public void TearDown()
        {
            Bootstrap4Net.Stop();
        }

        [Test]
        public void Test001()
        {
            //write here yout test script
        }
    }
}
```

Path to the AUT xml file

Start the Maveryx

Start the AUT

Stop the Maveryx
Framework

Create and run your first test

1. Create a new Maveryx Test Project
2. Create a new Maveryx Test Class
- 3. Create the AUT Launch file**
4. Write the test case
5. Run the test

Java AUT Launch File

To execute a Java Application-Under-Test it is necessary to create the related AUT launch file.

```
<?xml version="1.0" encoding="UTF-8"?>
<AUT_DATA>
  <SERVER_URL></SERVER_URL>

  <WORKING_DIR>./src/resources/AUT/java</WORKING_DIR> <!-- change this path to your working directory -->

  <APPLICATION_NAME>ButtonDemo</APPLICATION_NAME>

  <AUT_ARGUMENTS></AUT_ARGUMENTS>

  <VM_ARGUMENTS></VM_ARGUMENTS>

  <DESCRIPTION>
    Push-Button testing
  </DESCRIPTION>

  <JRE_PATH>${java.home}</JRE_PATH> <!-- change this path to your JRE home -->

  <MAIN_CLASS>com.sun.demo.ButtonDemo</MAIN_CLASS>

  <!-- on UNIX-like and MAC OS X systems change the path separator ';' to ':' -->
  <CLASSPATH>
    <LIB>
      <PATH>examples.jar</PATH> <!-- change this path to your Maveryx installation directory /demo -->
    </LIB>
    <!-- do not change the data below! (except for path separator on UNIX-like and MAC OS X systems) -->
  </CLASSPATH>
</AUT_DATA>
```

MFC & .Net AUT Launch File

To execute a MFC or .NET Application-Under-Test it is necessary to create the related **AUT launch file**.

```
<?xml version="1.0" encoding="UTF-8"?>
<AUT_DATA>
  <EXECUTABLE_PATH>\src\resources\AUT\windows\Notepad Enhanced.exe</EXECUTABLE_PATH>
  <APPLICATION_NAME>Notepad Enhanced</APPLICATION_NAME>
  <TOOLKIT>WIN</TOOLKIT>
  <TIMEOUT>1000</TIMEOUT>
  <DELTA_CHECK>1000</DELTA_CHECK>
  <AUT_ARGUMENTS></AUT_ARGUMENTS>
</AUT_DATA>
```

Set the absolute or relative path to your AUT executable file

Web AUT Launch File

To execute a Web Application-Under-Test it is necessary to create the related **AUT launch file**.

```
<?xml version="1.0" encoding="UTF-8"?>
<AUT_DATA>
  <EXECUTABLE_PATH>C:/Program Files (x86)/Google/Chrome/Application/chrome.exe</EXECUTABLE_PATH>
  <APPLICATION_NAME>CHROME</APPLICATION_NAME>
  <TOOLKIT>WEB</TOOLKIT>
  <AUT_ARGUMENTS>file:///./src/resources/AUT/web/index.html</AUT_ARGUMENTS>
</AUT_DATA>
```

Set the URL of the AUT

Set the path of the browser **you** want to use for your tests

Create and run your first test

1. Create a new Maveryx Test Project
2. Create a new Maveryx Test Class
3. Create the AUT Launch file
- 4. Write the test case**
5. Run the test

Example

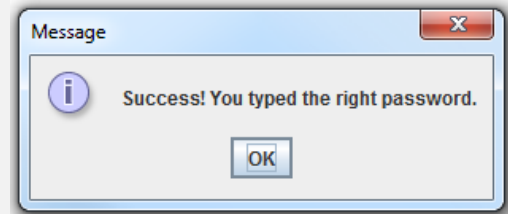
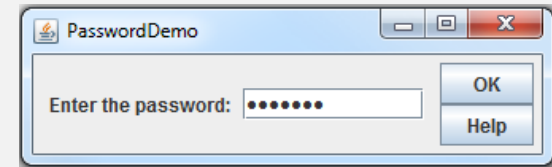
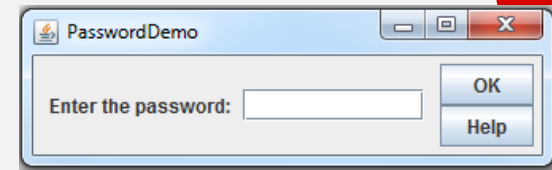
```
[Test]
public void Test001()
{
    var t = new GuiPasswordText("Enter the password");
    //check wheter the text field is editable
    Assert.True(t.IsEditable());
    //enter the password
    t.SetText("bugaboo");

    var ok = new GuiButton("OK");
    //check wheter the push button is enabled
    Assert.True(ok.IsEnabled());
    //click the OK button in the main frame
    ok.Click();

    var dialog = new GuiDialog("Message");
    var message = new GuiLabel("Success!", dialog);

    var expectedMessage = "Success! You typed the right password.";
    //check wheter the message dialog contains the excepted message
    Assert.True(message.GetActualId().Equals(expectedMessage));

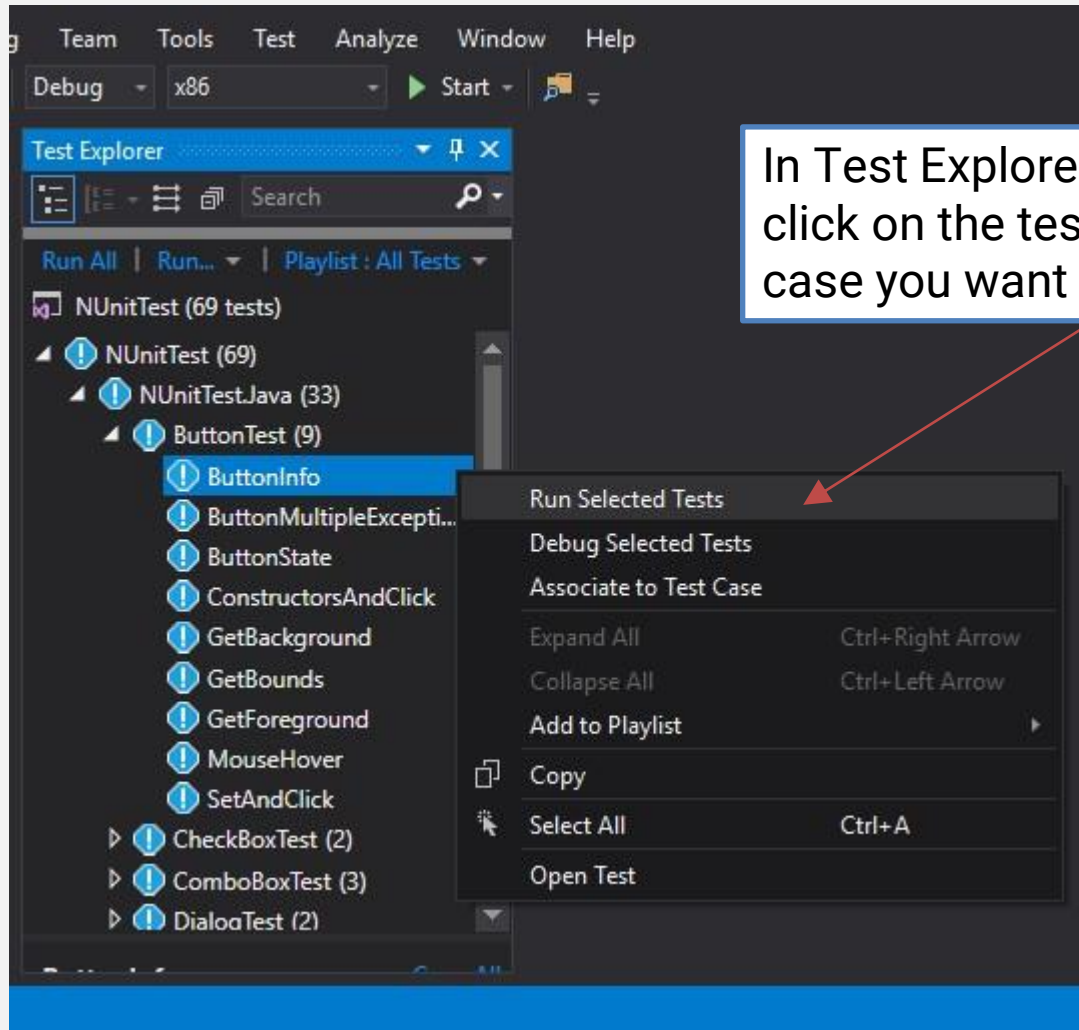
    //close the dialog
    dialog.Close();
}
```



Create and run your first test

1. Create a new Maveryx Test Project
2. Create a new Maveryx Test Class
3. Create the AUT Launch file
4. Write the test case
- 5. Run the test**

Run the Test Script



In Test Explorer window, right click on the test class or test case you want to run

THANK YOU

sales@maveryx.com

info@maveryx.com

+39 333 30 72 597

+39 351 87 85 706

