

TestApp by Unicus

Start för testutvecklare

Tutorial för att sätta upp miljö och komma igång med att köra egna tester

Författare: Joakim Odermalm

Dokumentversion: 1.0, 190510, Skapad

Innehåll

Sätt upp utvecklingsmiljö.....	2
Installera TestApp by Unicus.....	2
Installera Visual Studio.....	2
Skapa ett testprojekt.....	2
Kompilera.....	3
Köra test.....	4
Lägg till biblioteksfilen i TestApp.....	4
Skapa ett test.....	4
Köra.....	4

Sätt upp utvecklingsmiljö

Installera TestApp by Unicus

Installationsfilen finns (för närvarande) på <http://joakim.dreamhosters.com/testapp/>

Gå dit och klicka på **Install**.

Under installationen behöver man godkänna att en .NET-runtime installeras. Eventuellt visas också en stor farlig varning liknande denna:

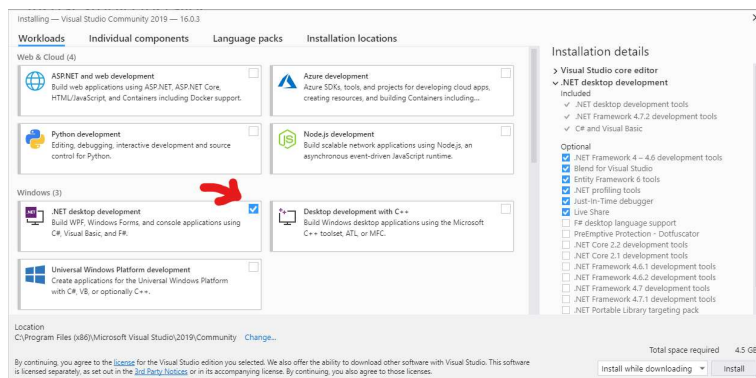


För att komma förbi den, klicka på **More info** och sedan **Run anyway**.

Installera Visual Studio

Ladda ner **Visual Studio Community** här: <https://visualstudio.microsoft.com/downloads/>

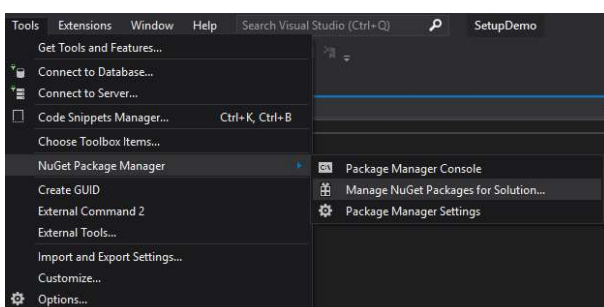
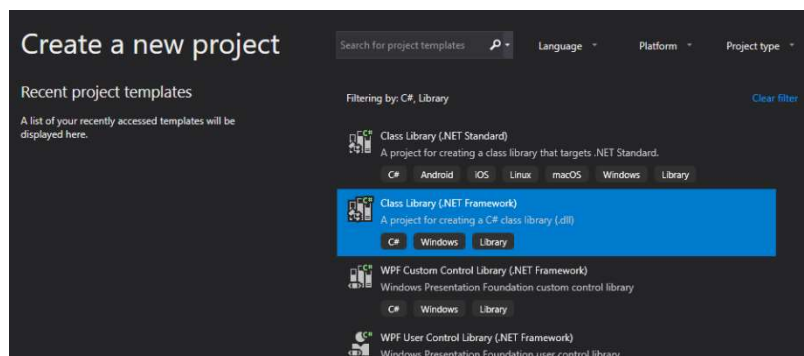
Under **Workloads**, välj **.Net desktop development ...**



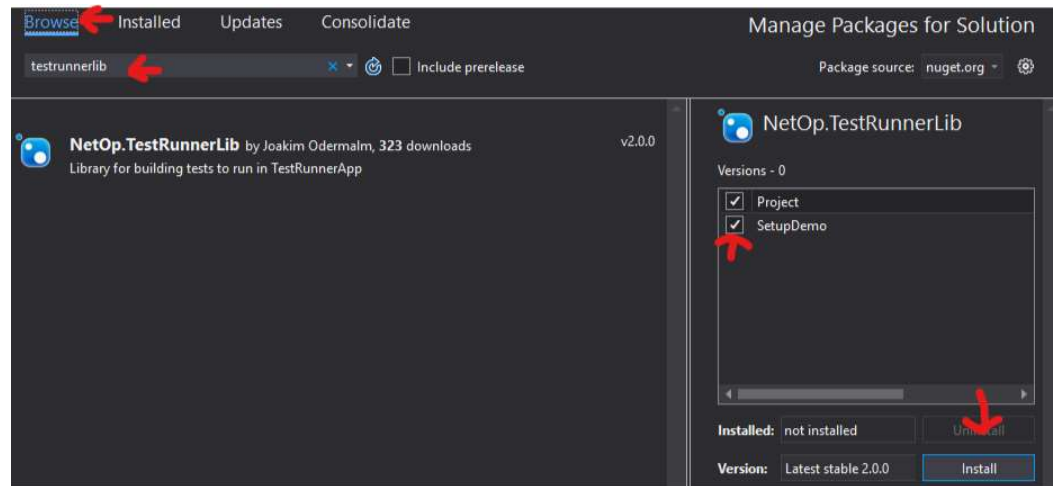
Skapa ett testprojekt

När VS startar, välj **Create a new project** och skapa ett **Class Library (.NET Framework)**.

När projektet öppnats, gå till **Tools -> NuGet Package Manager -> Manage NuGet Packages for Solution...**

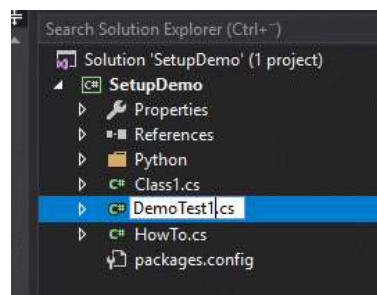


Välj **Browse**, sök efter **TestRunnerLib**, kryssa för ditt projekt och **Install**.



Nu har du en fil i ditt projekt som heter **HowTo.cs** och som kan användas som mall för ditt första test.

Gör en kopia (copy + paste) av HowTo.cs och döp om den till något lämpligt.



Döp om klassen MyWebTest till samma du döpte filen till ...

```
// Selenium
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;

References
public class DemoTest1 : IWebTest
{
    1 reference
    public TestResult Test(WebDriverType
    {
        int testStep = 0;

        try
```

... och ta bort klassen MyGeneralTest helt.

```
public class MyGeneralTest : ITest
{
    1 reference
    public TestKind Kind { get; } = TestKind.Other;

    1 reference
    public TestResult Test(string[] testdata)
    {
        int testStep = 0;

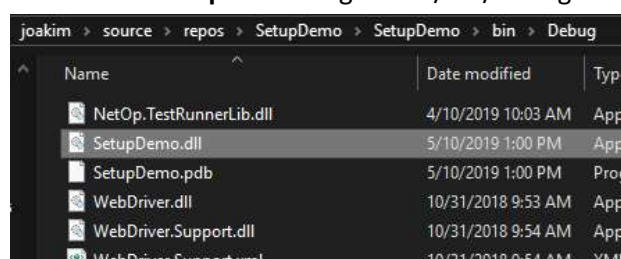
        try
        {
            // Test something
            testStep++;
            string testReturn = "Something my test returns";
            bool MyParameterTest =
                testReturn.Contains(testdata[1], StringComparison.Ordinal);

            // Return result
            if (MyParameterTest)
                return new TestResult(Outcome.Pass);
            else
                return new TestResult(Outcome.Fail, testReturn);

            // Exception while testing
            catch (Exception e)
            {
                return new TestResult(Outcome.Warning, testStep, e);
            }
        }
    }
} // class
```

Kompilera

När du författat ditt test, tryck **Ctrl-B** för att kompilera det. Filen som produceras hittar du enklast genom att högerklicka på projektet och välja **Open Folder in File Explorer** och gå ner i **/bin/Debug**.



Köra test

Lägg till biblioteksfilen i TestApp

Innan du kan köra testet måste den kompilerade filen kopieras till /Tests under testappens programkatalog. För instruktioner om det, läs kapitlet **Filer** i *Introduktion för användare* som ligger under Gemensamt Unicus Sverige\TestApp by Unicus.

Metoden som beskrivs under *Tips och tricks* rekommenderas hjärtligt.

Skapa ett test

Som sista steg behöver du nu skapa ett test i testappen som kör ditt skript.

Fyll i ett ID och ett namn så du vet vad det är.

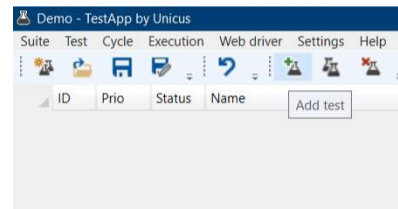
De viktiga fälten är sedan de längst till höger:

Assembly är namnet på biblioteksfilen (samma som projektet).

Namespace är det namespace skript-klassen ligger i. I det här fallet är detta mest organisatoriskt, välj ett samlingsnamn för den samling tester du jobbar med; en websida, en kund...

Type är typen / klassen för det specifika testet.

Testdata som skriptet får som input lägger du också in här.



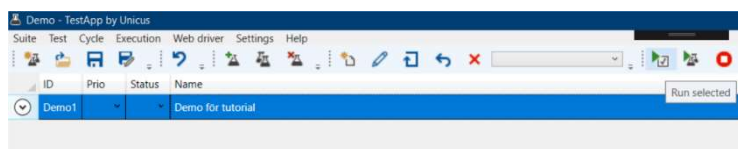
Add new test

ID Demo1	Objective	Assembly SetupDemo						
Name Demo för tutorial		Namespace DemoTests						
Status ▼	Pre-conditions	Type DemoTest1						
Priority ▼								
Component	Execution steps	Add test data						
Labels		<table><thead><tr><th>Index</th><th>Test data</th></tr></thead><tbody><tr><td>1</td><td>http://www.demo.net</td></tr><tr><td>2</td><td>Lite testdata</td></tr></tbody></table>	Index	Test data	1	http://www.demo.net	2	Lite testdata
Index	Test data							
1	http://www.demo.net							
2	Lite testdata							

```
1 namespace DemoTests
2 {
3     // TestApp
4     using TestRunnerLib;
5     // Selenium
6     using OpenQA.Selenium;
7     using OpenQA.Selenium.Support.UI;
8
9     // References
10    public class DemoTest1 : IWebTest
11    {
12        // 1 reference
13        public TestResult Test(WebDriverType webDriverType, string testdata)
14        {
15            int testStep = 0;
16
17            try
18            {
19                using (IWebDriver driver = WebDriver.Get(webDriverType))
20                {
21                    // Go somewhere
22                    testStep++;
23                    driver.Navigate().GoToUrl(testdata[1]);
24
25                    // Test something & return result
26                    testStep++;
27                    if (driver.Title.Contains(testdata[2], StringComparison.OrdinalIgnoreCase))
28                        return new TestResult(Outcome.Pass, $"Page {testdata[2]} found");
29                }
30            }
31            catch { }
32
33            return new TestResult(Outcome.Fail, "Test failed");
34        }
35    }
36 }
```

Köra

Tuta och kör:



F5 funkar också.