# UI 測試教學

## 環境設定

1. 開啟 Windows 開發人員模式

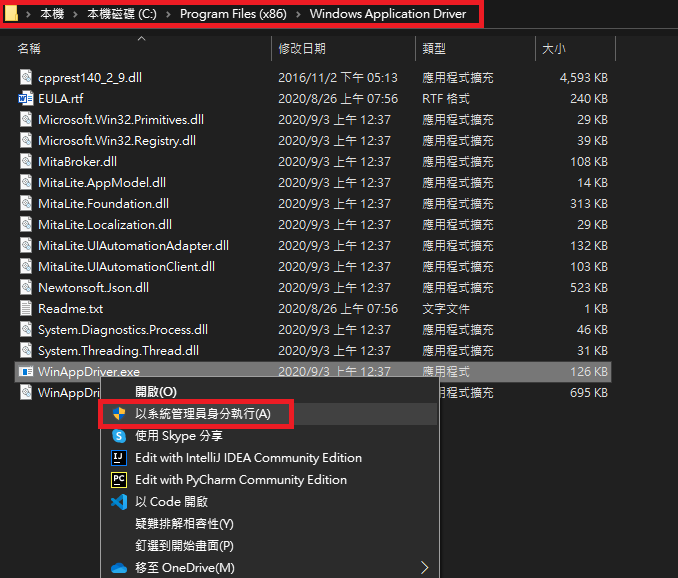
* 設定 -> 更新與安全性 -> 開發人員專用 -> 開發人員模式



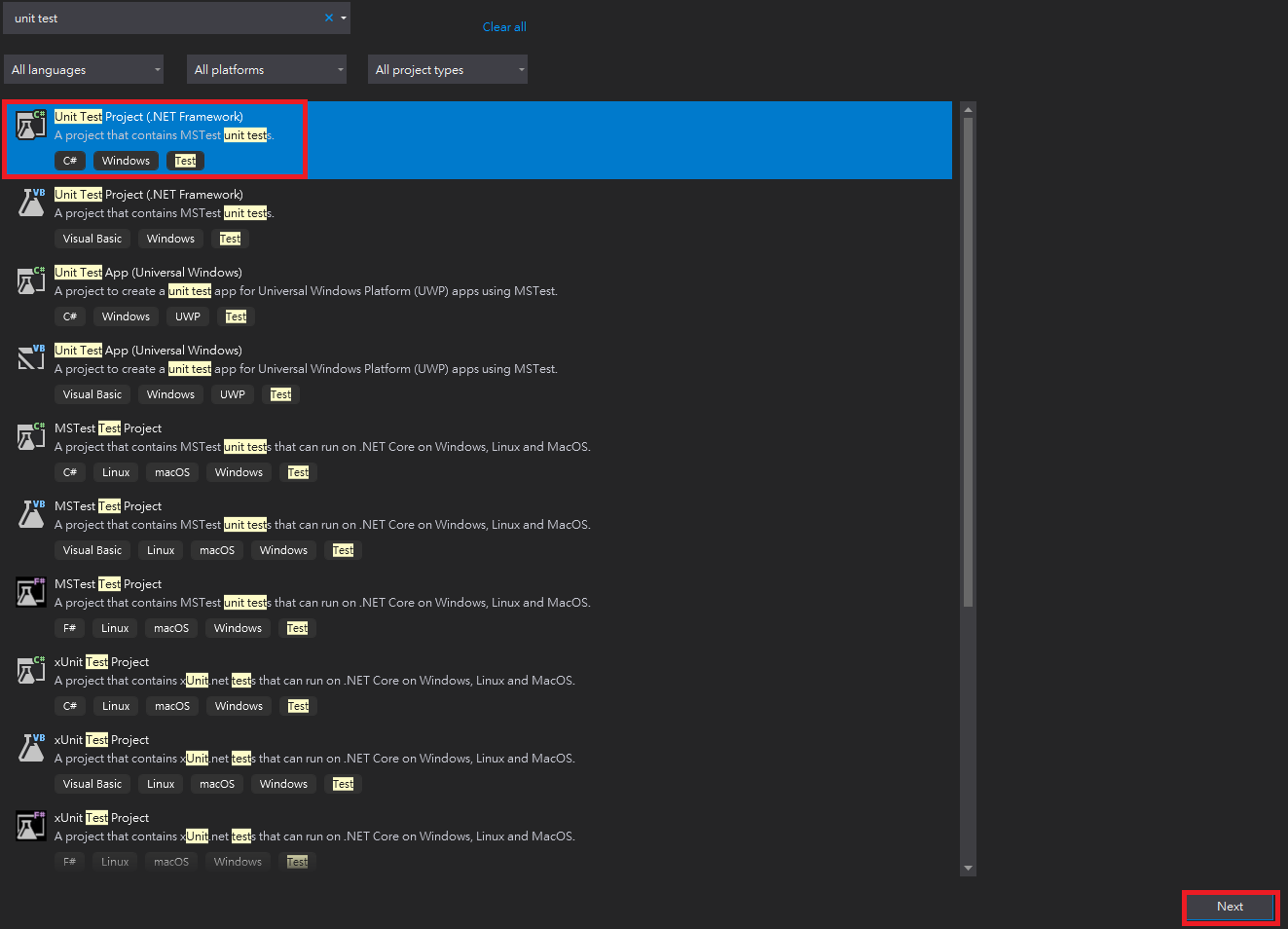


1. Winappdriver
   * 下載 [winappdriver](https://github.com/microsoft/WinAppDriver/releases/download/v1.2.1/WindowsApplicationDriver_1.2.1.msi) 並安裝
   * 進行測試前，請先以系統管理員身分開啟 winappdriver

(預設路徑為: *C:\Program Files (x86)\Windows Application Driver\WinAppDriver.exe*)

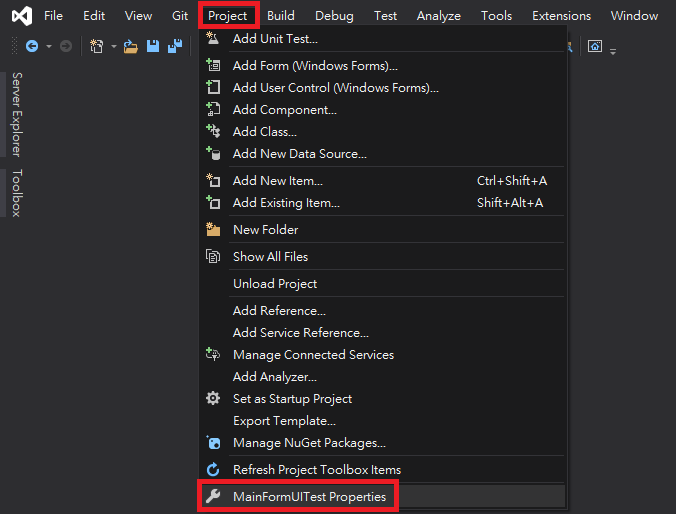


## 建立測試專案

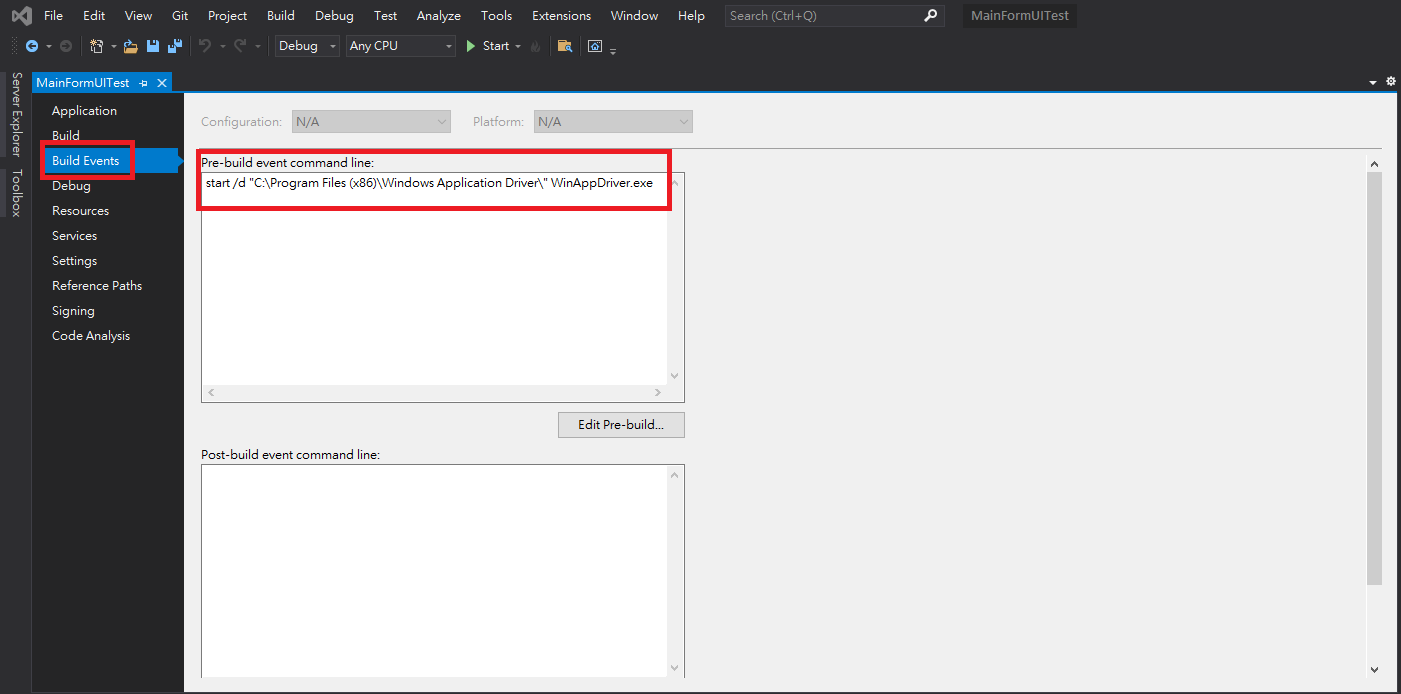


## 設定 VS pre-build

* 開啟 VS2019
* project -> project properties -> Build Event -> Pre-build event command line

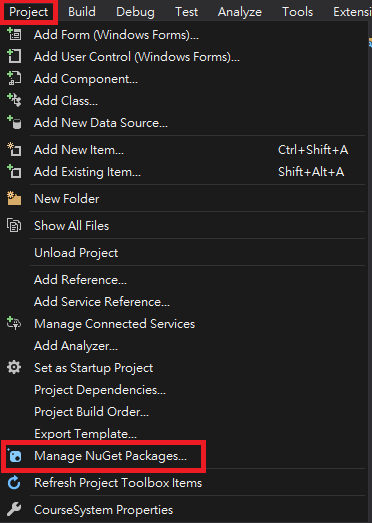


* 輸入 *start /elevate /d "C:\Program Files (x86)\Windows Application Driver\" WinAppDriver.exe* 後，每次執行測試前將會自動開啟 winappdriver

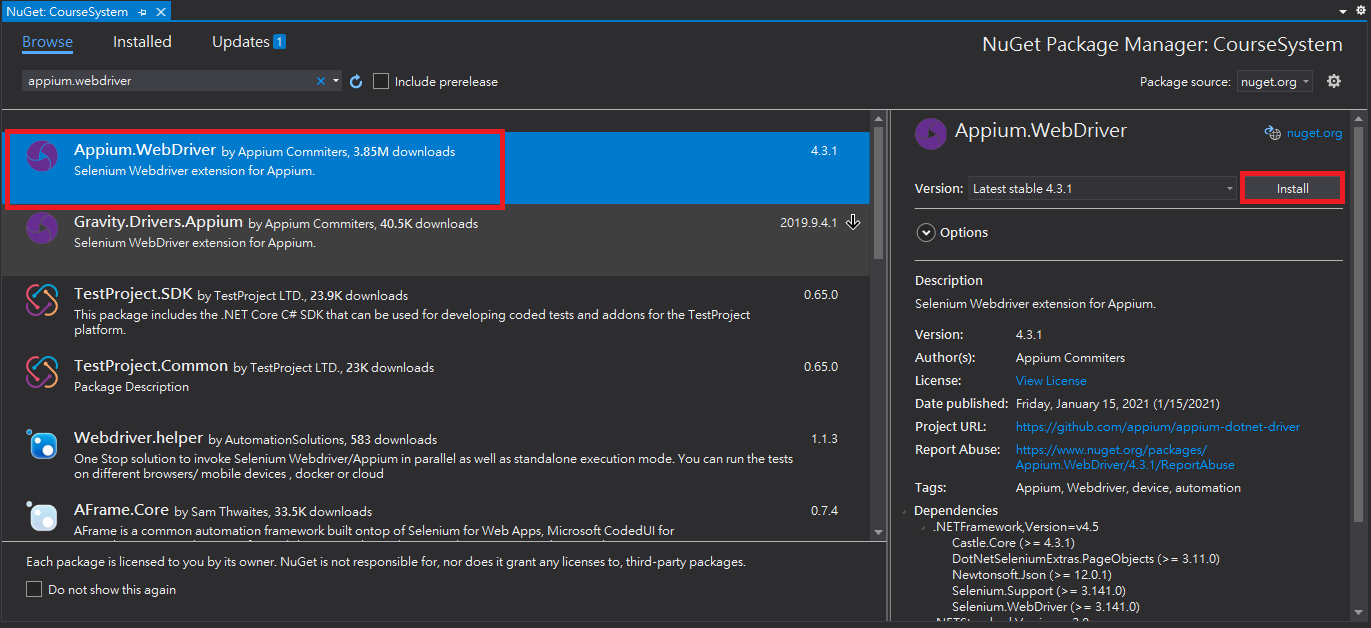


## 安裝 Appium

* project -> Manage NuGet Packages…

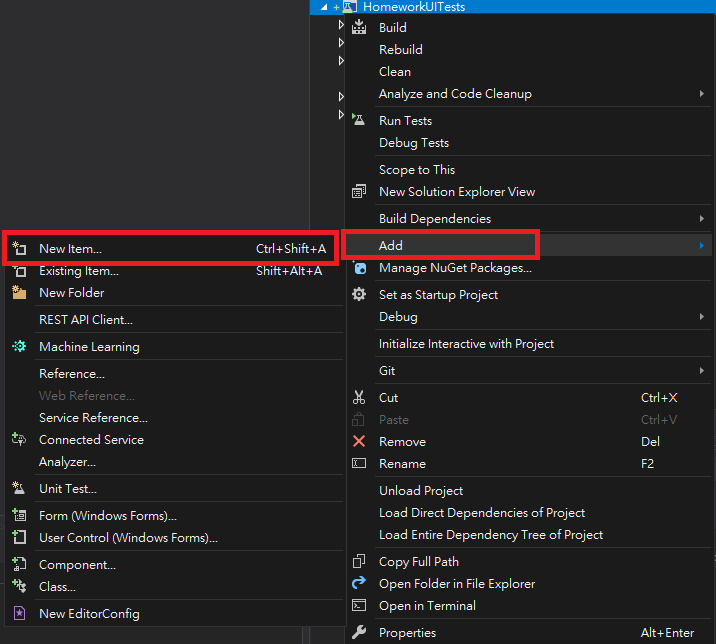


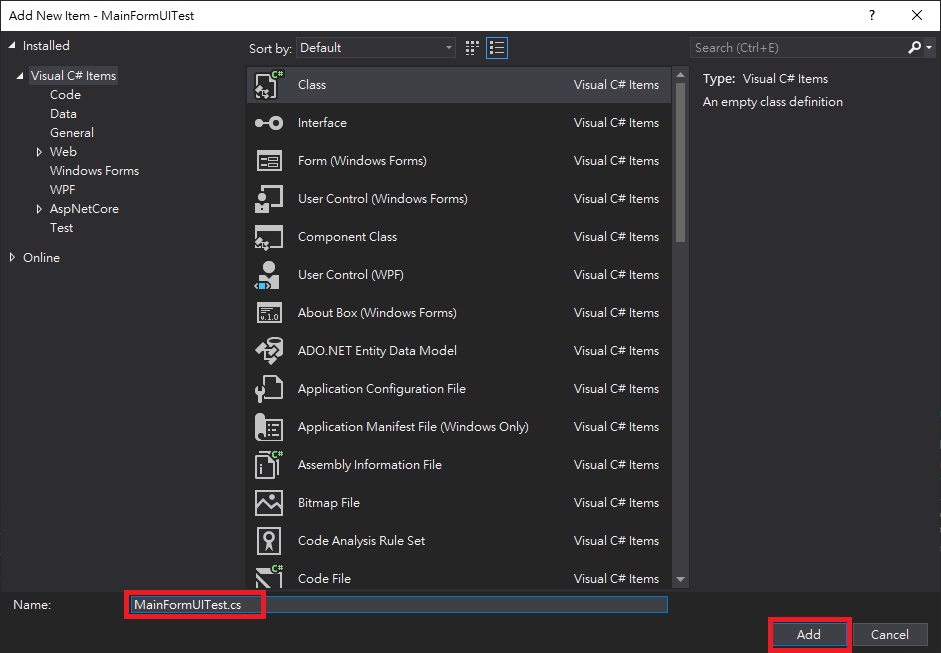
* install appium.webdriver



## 加入測試檔案

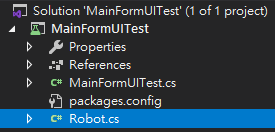
* add -> new item -> class



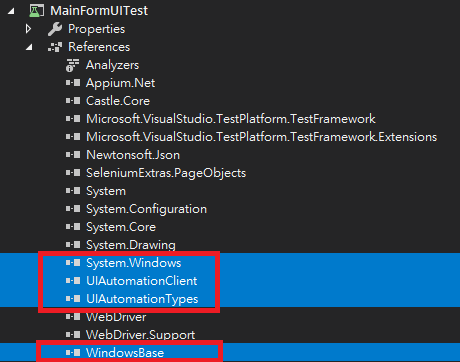


## 引入 Robot.cs

* 確認 [Robot.cs](https://drive.google.com/file/d/1VhzBamFc0yXk4TswLS8iGSL8lvvzgglH/view?usp=sharing) 的 namespace 是一樣的



* 加入以下參考: ***System.Windows, UIAutomationClient, UIAutomationTypes, WindowsBase***



## 修改 MainFormUITest.cs

/// <summary>

/// Summary description for MainFormUITest

/// </summary>

[TestClass()]

public class MainFormUITest

{

private Robot \_robot;

private const string APP\_NAME = "Microsoft.WindowsCalculator\_8wekyb3d8bbwe!App";

private const string CALCULATOR\_TITLE = "小算盤";

private const string EXPECTED\_VALUE = "顯示是 444";

private const string RESULT\_CONTROL\_NAME = "CalculatorResults";

/// <summary>

/// Launches the Calculator

/// </summary>

[TestInitialize()]

public void Initialize()

{

}

/// <summary>

/// Closes the launched program

/// </summary>

[TestCleanup()]

public void Cleanup()

{

}

}

## 開啟小算盤

/// <summary>

/// Launches the Calculator

/// </summary>

[TestInitialize()]

public void Initialize()

{

\_robot = new Robot(APP\_NAME, CALCULATOR\_TITLE);

}

## 關閉小算盤

/// <summary>

/// Closes the launched program

/// </summary>

[TestCleanup()]

public void Cleanup()

{

\_robot.CleanUp();

}

## 撰寫點擊按鈕的程式

/// <summary>

/// Runs the script: 123 + 321 =

/// </summary>

private void RunScriptAdd()

{

\_robot.ClickButton("清除");

\_robot.ClickButton("一");

\_robot.ClickButton("二");

\_robot.ClickButton("三");

\_robot.ClickButton("加");

\_robot.ClickButton("三");

\_robot.ClickButton("二");

\_robot.ClickButton("一");

\_robot.ClickButton("等於");

}

## 確認結果

/// <summary>

/// Tests that the result of 123 + 321 should be 444

/// </summary>

[TestMethod]

public void TestAdd()

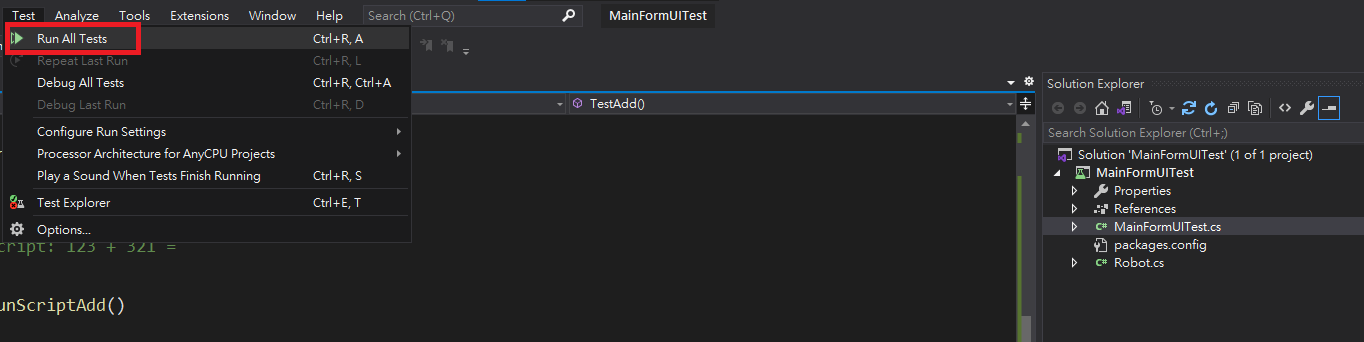
{

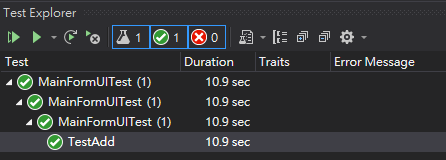
RunScriptAdd();

\_robot.AssertText(RESULT\_CONTROL\_NAME, EXPECTED\_VALUE);

}

## 執行





## Appendix

1. 測試其他程式
   * 如果要測試其他的程式，只要將程式的檔案路徑傳入 Robot.Initialize 中即可
   * 以下以作業程式作為範例，取得程式路徑程式碼

private string targetAppPath;

private const string START\_UP\_FORM = "StartUpForm";

// init

[TestInitialize]

public void Initialize()

{

var projectName = "CourseSystem";

string solutionPath = Path.GetFullPath(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "..\\..\\..\\"));

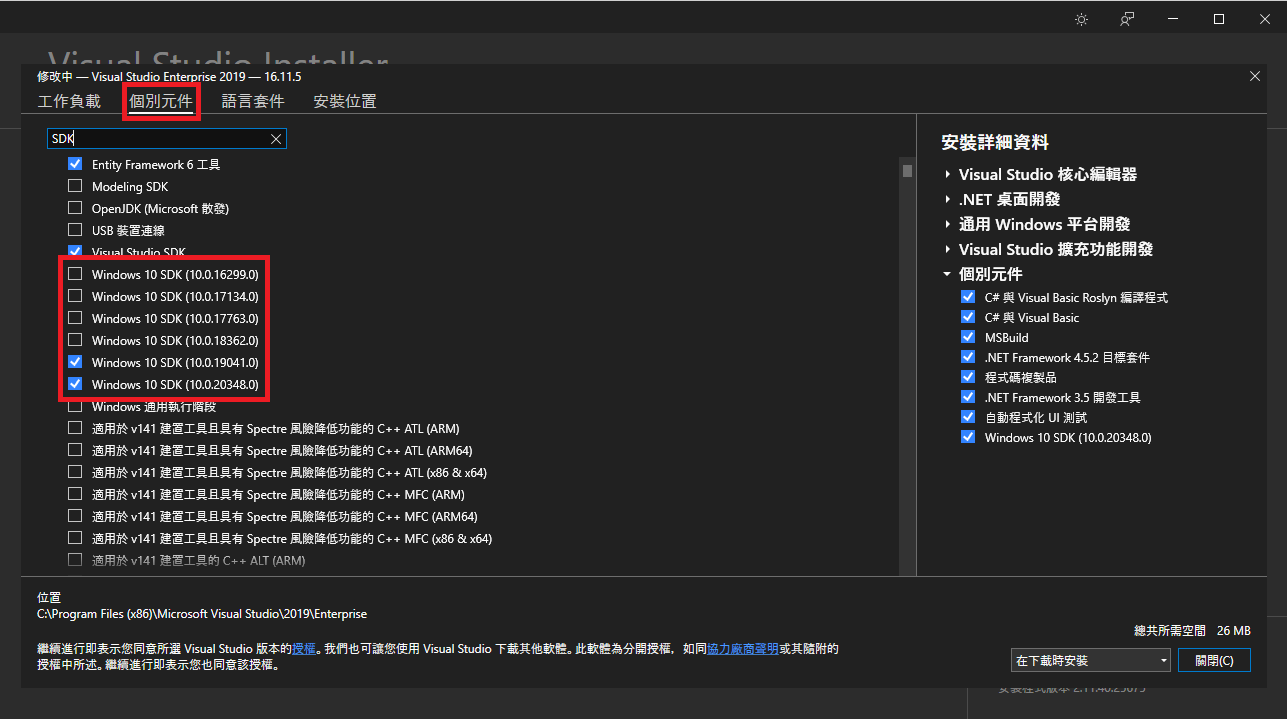
targetAppPath = Path.Combine(solutionPath, projectName, "bin", "Debug", "CourseSystem.exe");

\_robot = new Robot(targetAppPath, START\_UP\_FORM);

}

1. 利用 Windows SDK inspect.exe 協助測試

* 安裝 VS 時，就會安裝 Windows SDK，因此不須額外安裝，如果要安裝其他版本的 Windows SDK，可使用 VS installer



* inspect 程式路徑 *C:\Program Files (x86)\Windows Kits\10\bin\{ SDK 版本 }\x64\inspect.exe*
* 執行程式時，可使用 inspect 來查看元件名稱或結構
* 以下圖為例，當滑鼠點擊 DataGridView 中的視窗程式設計課程時，DataGridView 的結構會顯示在左方，而欄位的資訊顯示在右方。

