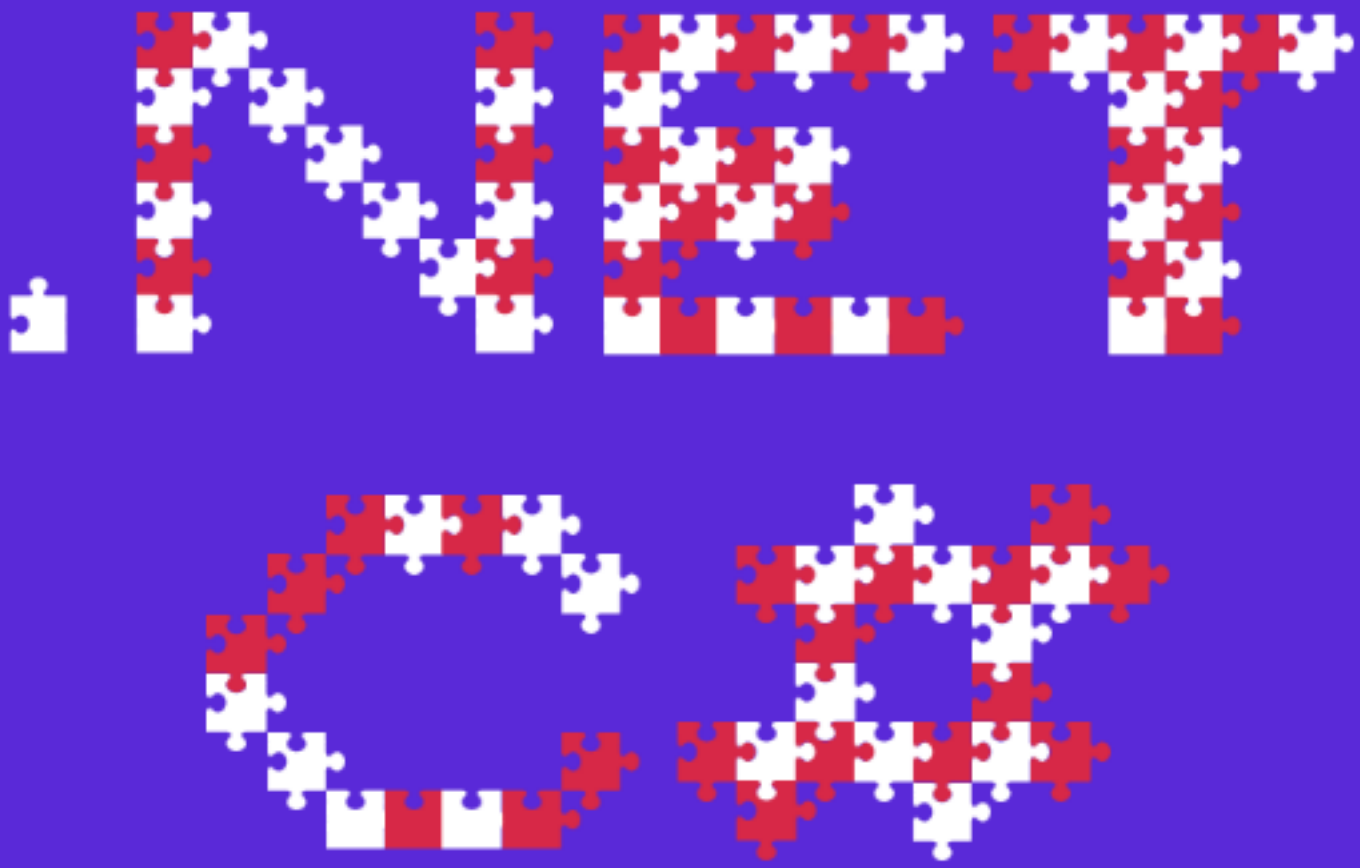
使用 Prism 進行 MAUI 專案開發

動手練習系列



Vulcan Lee 著

使用 Prism 進行 .NET MAUI 專案開發

動手練習系列叢書

Vulcan Lee

這本書的網址是 http://leanpub.com/Maui-Prism-Book

此版本發布於 2022-08-12

Contents

1. 關於本書

.NET MAUI[1] 多平臺應用程式 UI (是 Multi-platform App UI 的縮寫) 是一種跨平臺架構，可用於使用 C# 和 XAML 建立原生行動和傳統型應用程式。.NET MAUI 的專案採用了單一專案方式，便可以產生出 Android, iOS, macOS Catalyst, WinUI3 等不同作業系統平台上可以執行的應用程式，而這個 .NET MAUI 工具也可以視為 Xamarin.Forms 的繼承者，也就是說，.NET MAUI 是由 Xamarin.Forms 進化而來的全新開發跨平台工具。

之前在網路上看到微軟 James Montemagno 寫的 .NET MAUI - Workshop[2] 文章與專案程式碼與 Learn .NET MAUI - Full Course for Beginners | Build cross-platform apps in C#[3] Youtube 實際專案開發教學影片，讓我深深地覺得這是一份相當好的教學教材內容，它包含了靜態的文字說明與各個練習階段所開發出來的專案原始碼，有提供了長達四個小時的開發過程說明教學影片，讓任何想要理解與學習使用 .NET MAUI 工具來進行跨平台開發的人，可以有個很好的進入點。

在 James 所寫的文章與影片中，採用的是 .NET MAUI Shell[4] 與 Community-Toolkit.Mvvm[5] 這個工具來進行開發行動裝置應用專案的開發，其中前者 .NET MAUI Shell 在官方文章中有指出：.NET MAUI Shell 提供大部分應用程式所需的基本功能，以減少應用程式開發的複雜度，包括：描述應用程式視覺階層的單一位置、常見的導覽使用者體驗、URI 型流覽配置，允許流覽至應用程式中的任何頁面、整合式搜尋處理常式。對於第一次接觸 .NET MAUI 的開發者，可以透過 Shell 提供的相關 UI 設計機制與功能，快速開發出想要的應用程式。

[1] https://docs.microsoft.com/en-us/dotnet/maui/what-is-maui?WT.mc_id=DT-MVP-5002220
[2] https://github.com/dotnet-presentations/dotnet-maui-workshop
[3] https://www.youtube.com/watch?v=DuNLR_NJv8U
[4] https://docs.microsoft.com/en-us/dotnet/maui/fundamentals/shell?WT.mc_id=DT-MVP-5002220
[5] https://docs.microsoft.com/zh-tw/dotnet/communitytoolkit/mvvm?WT.mc_id=DT-MVP-5002220

而對於 CommunityToolkit.Mvvm 而言，則扮演了另外一個重要的角色，這是因為 .NET MAUI 這個專案開發，採用的 MVVM Model-View-ViewModel 的設計模式來進行開發，第一個 Model 通常為一個 .NET C# 內的 POCO 類別，而 View 則是為採用 XAML (XAML Extensible Application Markup Language 發音為 /□zæm□l/ 是一種宣告式的標記語言) 來進行要顯示在畫面上的 UI 內容宣告與定義，而對於這個頁面上的相關商業邏輯部分，則是會寫在一個 .NET C# 類別內，該類別也稱之為 ViewModel (該類別需要實作 INotifyPropertyChanged[6] 介面)。

如此，View 與 ViewModel 之間便可以透過 Data Binding 資料綁定 (繫結) 的方式整合再一起，讓 UI 與商業邏輯程式碼部分可以做到鬆散耦合的設計，形成關注點分離的方式，讓這個專案可以好維護與方便進行測試；由於要自行手作寫出具有資料綁定規範的程式碼，說實在的有些繁瑣，因此，CommunityToolkit.Mvvm 這個套件扮演了重要的角色，他透過原始碼產生器的功能，自動產生出許多原本需要開發者自行設計的程式碼，並且滿足資料綁定程式碼設計的要求，簡化整體程式碼設計過程與數量，讓整體原始碼看起來相當的清爽與簡潔。

因此，James 充分的發揮 .NET MAUI Shell 與 CommunityToolkit.Mvvm 的強大功能與特色，設計出這樣的動手實作練習教材，透過自己動手練習與開發，實際完成一個使用 .NET MAUI 開發出來的專案，間接地也學會如何使用 .NET MAUI 來開發跨平台專案的技能。

為什麼會有這本書的產生呢？這是因為作者早年在進行 Xamarin.Forms 專案開發與教學課程的時候，就習慣使用 Prism[7] 這個開發框架類別庫，Prism 是一個開發框架，讓所開發的專案可以採用與開發出具有鬆散耦合、可維護性與可測試性的能力，這些類型的專案包含了 WPF, Xamarin.Forms, .NET MAUI，這個套件是相當的好用與方便，可以快速地進行各種類型的專案開發，並且對於上面提到的一個資料綁定設計上的問題，那就是要自己手作寫出很多相同的程式碼問題，之前是使用了 PropertyChanged.Fody[8] 這個套件來

[6] https://docs.microsoft.com/zh-tw/dotnet/api/system.componentmodel.inotifypropertychanged?WT.mc_id=DT-MVP-5002220

[7] https://prismlibrary.com/docs/index.html

[8] https://github.com/Fody/PropertyChanged

解決此一問題，該套件使用的手法與 CommunityToolkit.Mvvm 做法不同，並不是使用原始碼產生器來產生出許多原始碼來解決問題，而是透過注入程式碼的方式，讓有實作 INotifyPropertyChanged 介面的類別，可以有正常運作的程式碼存在。

因此，作者想要使用 .NET MAUI + Prism.Maui + PropertyChanged.Fody 這三項工具，同樣的可以設計出如同 James 文章或者影片中的跨平台應用程式結果，所以，便會有這本書的誕生，因為，作者想要推廣 .NET MAUI 這個開發工具，希望有更多從事 .NET C# 開發者，可以有機會學習到這麼優秀的開發工具。

這本 “使用 Prism 進行 .NET MAUI 專案開發” 這是一本對於初學者與入門者，想要學習如何使用 .NET MAUI 來進行跨平台應用程式開發的必須要閱讀過的一本書籍。在這本電子書中，將從無到有的說明如何開發出跨平台的應用程式，將會從如何安裝與設定開發環境開始，因為這是一本定位為入門等級的書籍，因此，在書中將會有著操作畫面截圖，可以讓讀者清楚的知道每個步驟是如何設計出來的。接下來將會說明與使用 Prism.Maui 這個開發框架來進行開發 .NET MAUI 的應用程式。

本書內容屬於動手實作系列，也就是說，期望讀者可以按照這本書上的說明與操作步驟，逐一進行開發與設計，並且試圖邊做邊解決問題，當然，在這本書中會列出更多的技術參考來源連結，可以針對這些文章來強化對於 .NET MAUI 這個 UI 開發工具的相關專業知識。

1.1 這本書能提供什麼

在這本書裡面，將會提供 9 章的內容，分別是

- .NET MAUI 開發環境的安裝與設定

  當想要採用 .NET MAUI 來進行跨平台架構開發之前，需要將相關的 SDK 與開發工具安裝到自己的電腦上，在一開始的內容將會說明如何進行這些軟體的安裝與設定，此時，需要安裝與設定 Visual Studio 2022 與 Prism.Maui 的專案開發範本，如此，便

可以參考接下來的文章內容，跟著上面講解的程式碼與 XAML 標記宣告語言，親自動手寫入到練習專案內；一旦完成這個專案開發練習，相信也已經學會與具備 .NET MAUI 開發的能力。

- 建立專案與核心服務

  首先需要先來建立起一個 .NET MAUI 開發專案，由於 Prism.Maui 這是一個開發框架，因此需要有很多的程式碼必須事先設計在 .NET MAUI 專案內，這樣的事情是滿麻煩的，不過，Prism.Maui 提供了專案建立與開發的範本，透過這個專案範本所產生出來的 .NET MAUI 專案，就已經是可以進行與使用 Prism 相關功能了。一旦專案建立好了之後，便開始進行這個專案的結構設計與要用到的資料模型 Model 與讀取遠端 JSON 資料的服務設計，有了這些基礎商業邏輯程式碼，便可以開始準備進行 MAUI 程式碼的設計。

- 顯示單頁面的集合清單資料

  第一個要來學習如何建立一個新的頁面，也就是 MVVM 中的 View，接著要來建立一個新的 C# 類別，也就是 MVVM 中的 ViewModel，該類別必須要符合資料綁定的規範，也就是要實作 INotifyPropertyChanged 介面。不要急著進行其他程式碼或者 UI 標記的設計，養成習慣，先把 View 與 ViewModel 註冊到 DI / IoC 相依性注入容器內，因為之後要進行測試與執行過程中，才不會看到莫名的錯誤。最後就是要把 ViewModel 內的 C# 程式碼設計出來，把 View 內的 XAML 標記宣告語言定義出來，其中透過 XAML 延伸標記 `{Binding ...}` 用法，使用資料綁定語法，將 View 內的某個控制項內的屬性與 ViewModel 類別內的某個屬性進行綁定起來，讓 View 與 ViewModel 可以正常協同運作起來。

- MVVM 與 Data Binding 資料綁定

  大家都知道，只要有使用 XAML 這種標記宣告語言來設計畫面 UI 的專案，必定會使用到 MVVM 設計模式，而 MVVM 設計模式的內部運作核心那就是 Data Binding 資料綁定機制，能夠讓資料綁定機制正常運作的背後推手則是 PropertyChanged 事件，因此，在這個章節內，將會說明屬性變更通知的設計原由，以及為什麼直接自行手作 INotifyPropertyChanged 介面是相當不親民的，緊接著說明採用 PropertyChanged.Fody 之後，可以達到甚麼樣的好處與如何使用這個套件來進行資

料綁定的設計。

- 頁面導航 Navigation 與參數傳

  一旦猴子清單頁面設計好了，便可以將所有從網路上讀取到的猴子資訊，顯示在螢幕上，當使用者點選任何一隻猴子之後，將會再另外一個新的頁面，顯示出這隻猴子的詳細資訊出來，而在這個章節將會說明如何在這個專案內，建立第二個新的 View 與 ViewModel，並且如何在猴子清單頁面有互動的時候，可以切換到這裡新的頁面上，這樣的過程在 .NET MAUI 內稱之為 導航 Navigation[9]，同時，也要學習如何將使用者點選的猴子物件，傳遞到新的頁面上，並且在新的頁面上，要如何才能夠讀取到這個剛剛傳遞進來的猴子物件。

- 平台特性 Platform Feature 設計

  在這個練習專案中，有些需求需要使用到行動裝置硬體上的功能，例如，前面有使用到判斷該行動硬體裝置的網路是否有正常連線到 Internet 上，而在這裡還有兩個需求，那就是要讀取這台行動裝置上 GPS 晶片上的現在經度與維度位置，取得該 GPS 位址之後，便可以做到這樣的設計需求：找出離你最近的猴子是哪一個種類與使用地圖應用程式，顯示出該猴子存在於哪個地方。

- 其他集合檢視 CollectionView 下拉更新設計

  在之前所設計的猴子清單頁面內，想要取得所有猴子清單的資訊，需要使用者點選螢幕右下角的按鈕，這樣便會觸發該按鈕所綁定在 ViewModel 上的命令物件內的委派方法，如此就可以取得在網路上最新的猴子清單，並且可以顯示在螢幕上了，可是，這樣的設計似乎不太友善，在這裡將會設計一個手勢操作，那就是當使用者使用手指從螢幕的上方往下滑動的時候，便可以觸發一個事件或者命令，如此，就可以執行之前設計好的讀取與更新猴子清單的程式碼了。

- UI 美化之集中管理設計與主題佈景 Theme

  前面的所有設計與操作過程，都事先已完成整體應用程式的功能需求為主來進行開發，現在，這個專案已經開發到最後了，將會針對這兩個關於猴子的頁面上的 UI 來學習如何進行集中管理這樣的技能與美化整體頁面工作，想要做到集中管理與設計 UI 視覺效果的目的是在於可以減少剪下、貼上的設計動作數量，讓整體 XAML 標記

[9] https://docs.microsoft.com/zh-tw/dotnet/maui/user-interface/pages/navigationpage?WT.mc_id=DT-MVP-5002220

更加的清爽與簡潔，並且，一旦有需要變更 UI 顏色與任何配置效果的時候，可以集中管理設計地方進行一次性的修正，這樣，便可以完成相關頁面與 UI 控制項的視覺效果變更；另外，在這裡還會說明如何設計符合暗色與亮色主題佈景環境下應用程式顯示效果。

1.2 誰適合閱讀這本書

對於任何想要學習 .NET MAUI 這個開發技術，可以透過本書的開發練習說明，逐步了解到如何真正的設計出一個 .NET MAUI 實用專案。

讀者必須具備 .NET / C# 的開發經驗、對於開發工具而言，本書是在 Windows 10 作業系統下，使用任何 Visual Studio 2022 的版本，就可以進行開發。

1.3 練習專案原始碼

本電子書中的練習專案原始碼，可以從 hthttps://github.com/vulcanlee/Maui-Prism[10] 取得

1.4 關於作者

作者從 2012 年開始進行 .NET / C# 教育訓練課程開發與授課之後，對於許多開發出來的課程，並不是隨便敷衍的設計出來，這其中我會不斷地針對已經開發出來的課程來進行修正、改版、追加內容，因此，相關的課程是不斷的在變化與強化的。

要完成這樣的一系列課程是一種自我挑戰，讓自己可以看得更高、看得更遠、讓抄襲者永遠無法模仿，因為，這些課程加入了許多巧思與技能在裡面，內行看門道、外行看熱鬧。

[10] https://github.com/vulcanlee/Maui-Prism

我所設計的教學課程，為了要能夠讓入門者、想要精通者都可以喜歡我設計的課程，將會把課程應該具有的特色，所提供的內容將會涵蓋到更多的層面、更多範例碼、更多觀念介紹、更多的動畫來理解艱澀技術應用、更多挑戰應用、更多的日常遇到問題與解決方法、更完整的開發設計指引。

因此，很期望到時候能夠與大家一起來進行這個課程的交流與互動，也希望大家可以從這些課程學到更多知識與經驗，應用在日常開發專案上。

若對於這些課程有興趣，或者想要企業內訓者，可以到 Xamarin Blazor 實驗室[11] 粉絲團來私訊給我

課程名稱: 初探 .NET 平行程式設計 (非同步程式設計系列之 1 / 6)

這年頭手機都多核心了我們寫的程式還跑在單核上嗎？

一台電腦8核16緒但我們的程式就是跑不快？

平行程式設計是近年來一個很務實的議題，之前 SkillTree 有開過較進階的「勇闖非同步程式設計」，收到許多開發人員的好評但有開發人員反應希望能夠開設更初階一點的入門主題，於是本活動來了！這是專門為了「沒實際摸過平行運算、非同步的開發人員」所設計的，讓你短時間掌握平行程式設計基本概念與觀念。

課程大綱

- 使用同步程式設計來解決問題
- 了解同步程式設計的瓶頸與使用非同步程式設計要解決的問題
- 為什麼要有平行程式設計與微軟提出的解決方案

[11] https://www.facebook.com/vulcanlabtw

- 將待解決問題採用平行、並行非同步程式設計 - 使用執行緒與執行緒集區
- 了解平行與並行計算的不同
- 介紹什麼是非同步程式設計
- 使用 TPL Task Parallel Library 工作平行類別庫來解決問題
- 使用 Timer，Background，委派來進行非同步程式設計
- 使用資料平行處理程式設計來解決問題
- 使用 PLINQ 來解決問題

等級

入門

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 的知識與使用技術

想要參加

此課程為 Skill Tree 專屬課程，想要參加者，請隨時注意 Skill Tree 的最新公告

參考網址: https://skilltree.my/Events/2022/1/9/Parallel-programming-in-dotnet-for-beginners-Batch-2[12]

課程名稱: 由 Parallel.For 來看多執行緒程式設計 (非同步程式設計系列之 2 / 6)

在多執行緒程式設計領域中，有 TPL , ThreadPool , Parallel.For , PLINQ 等等技術，其目的在降低使用複雜度，提供高階程式設計模型，開發者可以很容易地使用這些功能。但坊

[12] https://skilltree.my/Events/2022/1/9/Parallel-programming-in-dotnet-for-beginners-Batch-2

間流竄許多種各式各樣的奇技淫巧，有些是聽從前輩的建議，有些是自身特定情境中的經驗，這些招式與看法不能說錯，但總是片片斷斷無法有系統的理解背後的原理與限制，所以 SkillTree 舉辦了本活動針對 Parallel.For 做深入的探討，藉由因循漸進的案例讓您充分了解這技術的奧妙。

本課程是平行程式設計的初階，不是程式學習的初階，您必須具備 C# 開發經驗、了解泛型與委派的使用方式，並且具備基本電腦架構運作知識。

本課程的緣由來自臉書討論版上的一連長串討論內容，有興趣的人可以查看 討論串列[13] 螢幕截圖，若你對於這片串列上所提出的問題，或者有人提出的解答，存在著疑問或者更多的問題，那麼，你一定要來報名參加這個課程，因為，你所有的問題都可以從這個課程中獲得解答。

課程大綱

- Paraller.For 效能驗證
- Paraller.For 與 Thread 兩者的差異與極限
- Paraller.For 與 Task(TPL) 相互的優缺點評比
- Task 實踐與原理探討

等級

入門

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 的知識與使用技術

[13] https://vulcanfiles.blob.core.windows.net/\protect__xunadd_text_character:nN{\textdollar}{S}web/Share/2022/Multi-Thread-Parallel-For.png

想要參加

此課程為 Skill Tree 專屬課程，想要參加者，請隨時注意 Skill Tree 的最新公告

參 考 網 址: https://skilltree.my/Events/2022/1/16/NET-CSharp-Parallel-Programming-Batch-2[14]

課程名稱: 精準解析 .NET Thread 執行緒 (非同步程式設計系列之 3 / 6)

在 .NET 要建立一個執行緒，將需要指定一個委派方法，而一個執行緒 Thread 代表一個正在同步執行程式碼，若想要同時執行多個委派方法，則需要建立多個執行緒，而一台電腦能夠同時處理執行緒的數量，將會取決於這台電腦上的 CPU 的能力。

身為一個 .NET / C# 程式設計師，想要提升自我能力，使其可以進行平行程式設計技能，就需要具備多執行緒開發技術。透過多執行緒設計出來的程式碼，將可以同時執行多個程式碼，並會有助於整體應用程式的執行效能提升，充分發揮這台電腦 CPU 的執行效能。

然而，如何進行多執行緒的程式設計，將會需要學習 .NET 中的 Thread 物件的使用與操作，當完成此課程之後，你將會具有多執行緒程式設計的能力，並且了解到多執行緒程式設計上會遇到的問題與瓶頸。

課程大綱

- 執行緒　　Thread - 基本認識
- 產生　　Creation
- 啟動　　Start
- 傳入參數　　Parameter

[14] https://skilltree.my/Events/2022/1/16/NET-CSharp-Parallel-Programming-Batch-2

- 結束 Wait / Join
- 傳回值 Return Value
- 優先權 Priority
- 前景與背景 Foreground / Background
- 取消 Cancellation
- 異常與除錯 Exception

等級

中階

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 的知識與使用技術
- 了解平行、並行、同步、非同步等知識

想要參加

此課程為 Skill Tree 專屬課程，想要參加者，請隨時注意 Skill Tree 的最新公告

參考網址: https://skilltree.my/Events/2022/7/17/analyzing-dot-net-thread[15]

課程名稱: 精準解析 .NET Task 工作 (非同步程式設計系列之 4 / 6)

以往想要進行平行或非同步程式設計 (平行計算是一種非同步計算，前者屬於透過 CPU 來做到同時執行的需求，後者大多表示要進行 I/O 或者網路呼叫的時候，所要進行的處理作業)，往往需要透過多執行緒來完成，可是要能夠充分駕馭執行緒來完成上述設計需

[15] https://skilltree.my/Events/2022/7/17/analyzing-dot-net-thread

求，對於絕大多數的程式設計師而言，將不是一件簡單的工作；有鑑於此，微軟在 .NET Framework 4.0 之後，推出了工作平行類別庫 Task Parallel Library (TPL)，而在 .NET BCL 中的許多 API，也都改寫成為使用了 TAP 以工作為基礎的非同步模式 Task-based Asynchronous Pattern 的 API，取代以往 APM 與 EAP 的程式設計做法；這樣的改變將會讓 C# 程式設計師可以享受到許多 TPL 類別庫所帶來好處。

使用 TPL 中來執行的工作 Task 物件，通常是從執行緒集區上取得執行緒來以非同步方式進行執行，透過將複雜的執行緒操作封裝到工作物件內，讓程式設計師可以更加輕鬆與容易地來進平行與非同步的需求設計。

當完成此活動之後，你將會具備使用 Task 物件來進行非同步程式設計能力，讓設計出來的應用程式專案不在執行時候發生卡卡現象，充分享受到非同步程式設計所帶來的好處，當然，對於日後要精通 C# 5.0 所提供的 async 與 await 技術，將是不可或缺與必備的知識。

課程大綱

- 什麼是「工作」(Task)
- Thread 與 Task 的異同說明
- Task 的使用情境
  - 產生
  - 啟動或傳入參數
  - 等候結束或傳回值
  - 繼續
  - 狀態
  - 取消
  - 進度
  - 異常與除錯

等級

中階

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 的知識與使用技術
- 了解平行、並行、同步、非同步等知識

想要參加

此課程為 Skill Tree 專屬課程，想要參加者，請隨時注意 Skill Tree 的最新公告

參考網址: https://skilltree.my/Events/2022/7/10/analyzing-dot-net-task[16]

課程名稱: 精通與使用 async 與 await (非同步程式設計系列之 5 / 6)

在以往想要在 .NET / C# 下進行非同步程式設計的時候，可以套用不同設計模式來做到，這包括了：Asynchronous Programming Model (APM) / Event-based Asynchronous Pattern (EAP) / Task-based Asynchronous Pattern (TAP)；然而這些設計模式還是存有潛在許多問題無法解決，使用起來總是卡卡不順。

當 .NET Framework 4.5 與 C# 5.0 推出之後，對於 .NET 開發者而言，日後面對於非同步程式設計需求變得更加簡單雨容易，而且也方便除錯與開發，這些神奇的事情將會來自於可以透過 async 關鍵字將同步方法轉換為非同步方法，如此，在此非同步方法內便允許使用 await 關鍵字。一旦使用 await 關鍵字要進行非同步作業呼叫的時候，將不會造成當前執行緒被封鎖直到非同步作業結束，而是立即歸還當前執行緒，這樣的設計方式大幅提升整體系統的運作效能與 UI 流暢回應能力。

[16] https://skilltree.my/Events/2022/7/10/analyzing-dot-net-task

在本課程中，將會介紹使用 async 修飾符和 await 關鍵字，如何在 C# 中進行非同步程式設計的作法。這個課程將會提供相關 async await 的相關知識與觀念，透過不同範例程式碼來理解使用與應用技巧，透過動畫來理解艱澀難懂的技術與知識應用、學會更多日常遇到問題與解決方法、與完整的開發設計指引與口訣。

課程大綱

- 初次探索 async await 運作方式
- 編譯器對 async 做了哪些事情
- 了解非同步工作、方法差異
- async await 的開發指引與設計口訣
- 洞悉與活用 async await
- 更多 await 的相關問題

等級

高階

需求

不建議對於 .NET C# 非同步程式設計沒有經驗人參加此課程

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 的知識與使用技術
- 了解平行、並行、同步、非同步等知識
- 具有 Thread 執行緒類別使用與程式設計經驗
- 具有使用 TPL Task 工作類別的使用與程式設計經驗

想要參加

此課程為 Skill Tree 專屬課程，想要參加者，請隨時注意 Skill Tree 的最新公告

課程名稱: Blazor 全端開發，新手村一日脫逃術

在 Visual Studio 2022 正式版即將推出的時候你有沒有發現 Blazor 經常出現在文件內？Blazor 是微軟全新的全端解決方案，它可以讓 C# 開發者只需要會 C# 就可以達到非常棒的網頁開發能力，它與 Web forms, Silverlight 的理念類似，讓開發人員只需要會 C# 就可以完成 Web APP，現在競爭激烈，如何讓開發的經驗可以共通 Blazor 就是一個絕佳的選擇，畢竟僅需要會 .NET / C# 就可以開發 Web App 是件相當誘人因素，尤其讓許多桌面應用程式的開發者頭痛的 JavaScript 在這樣的開發方式下，就變成不是必須的考量了。

不相信 Blazor 的能力，或擔心使用 Blazor 造成技術門檻過高？我們準備了一個講者的真實經驗，你會發現使用 Blazor 是可以降低技術門檻的，也讓團隊補人變的更容易

對於身為 .NET C# 開發者而言，想要成為一個全端網站工程師，將不再是夢想，因為透過 Blazor 框架，不需要會 JavaScript ，便可以輕鬆、容易、快速地完成網站專案開發；Blazor 相較於其他前端網頁開發技術，其學習曲線不會十分陡峭，對於 .NET C# 不太有經驗的人，也是可以輕鬆上手的，現在就讓 SkillTree 一起帶你逃離新手村！

課程大綱

- 網站開發基本知識回顧
- 了解網頁之 HTML & CSS 轉譯過程、介紹三種網頁開發架構的比較與分析
- Blazor 雙開發框架解析
- Blazor Server
  - 運作原理
  - 專案結構
- Blazor WebAssembly
  - 運作原理
  - 專案結構

    - 如何挑選 Blazor 架構來進行開發
- Blazor 預設範本專案解析
    - Counter 元件
    - FetchData 元件
    - Razor 語法
    - Dependency Injection 觀念與用法
- Blazor 新手必學實戰技能
    - 第一個 Blazor
    - C# 程式碼設計方法
    - 單向資料綁定與重新轉譯
    - 互動與事件設計
    - 元件生命週期事件
    - 元件參數傳遞與回應事件
    - C#/JavaScript 互相呼叫
    - 表單欄位輸入與驗證檢查
    - 頁面間的的導航切換’

等級

入門

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 與事件 Event 的知識與使用技術
- 使用過 LINQ, 非同步開發方式
- 孰悉 HTML, CSS 語言用法
- 原則上不需要了解與使用到 JavaScript

想要參加

此課程為 Skill Tree 專屬課程，想要參加者，請隨時注意 Skill Tree 的最新公告

參考網址: https://skilltree.my/Events/2021/12/11/Blazor-for-Beginners[17]

課程名稱: 精通 Xamarin.Forms 跨平台開發

採用 Prism 開發框架與 MVVM 設計模式來進行 Xamarin.Forms 跨平台專案開發，製作出可以在 iOS, Android 下執行的 App 應用程式

課程大綱

等級

入門

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 與事件 Event 的知識與使用技術

想要參加

請與作者私訊

---

[17] https://skilltree.my/Events/2021/12/11/Blazor-for-Beginners

課程名稱: 精通 .NET MAUI 跨平台開發

採用 Prism.Maui 開發框架與 MVVM 設計模式來進行 Xamarin.Forms 跨平台專案開發，製作出可以在 iOS, Android 下執行的 App 應用程式

課程大綱

等級

入門

需求

- 具備 C# 程式語言開發經驗
- 具有委派 Delegate 與事件 Event 的知識與使用技術

想要參加

請與作者私訊

2. .NET MAUI 開發環境的安裝與設定

微軟在今年 2022 年推出了 .NET 多平臺應用程式 UI (.NET MAUI[1])，這是一種跨平臺架構，可透過 C# 和 XAML[2] 建立原生行動和桌面應用程式，而其中 MAUI 是 Multi-Platform App UI 的縮寫。

其實，有使用過 Xamarin.Forms[3] 開發過跨平台應用程式的開發者對於 MAUI 應該不會很陌生，因為，他就是 Xamarin.Forms 的下一代進化版本工具，所以，當然是可以透過 MAUI 來開發出 Windows / iOS / Android 的應用程式，除了這三種平台，還可以開發出 macOS 的應用程式。

在這章中將要來了解 MAUI 這個 UI Toolkit 的實際產生過程與專案架構，然而，在這個時間點，想要開發 MAUI 應用程式，必須安裝 Visual Studio 2022 17.3 以上版本。

2.1 安裝 Hyper-V 管理員

為了要能夠在本機電腦上跑 Android 模擬器，因此，建議需要安裝 Hyper-V 管理員工具，底下將會說明如何安裝這個工具。

在 Windows 11 工具列上，點選 [開始] 圖示

請在最上方的文字輸入盒內，輸入 [控制台]

[1] https://docs.microsoft.com/zh-tw/dotnet/maui/what-is-maui?WT.mc_id=DT-MVP-5002220
[2] https://docs.microsoft.com/zh-tw/dotnet/desktop/wpf/xaml?WT.mc_id=DT-MVP-5002220
[3] https://docs.microsoft.com/zh-tw/xamarin/xamarin-forms?WT.mc_id=DT-MVP-5002220

搜尋控制台

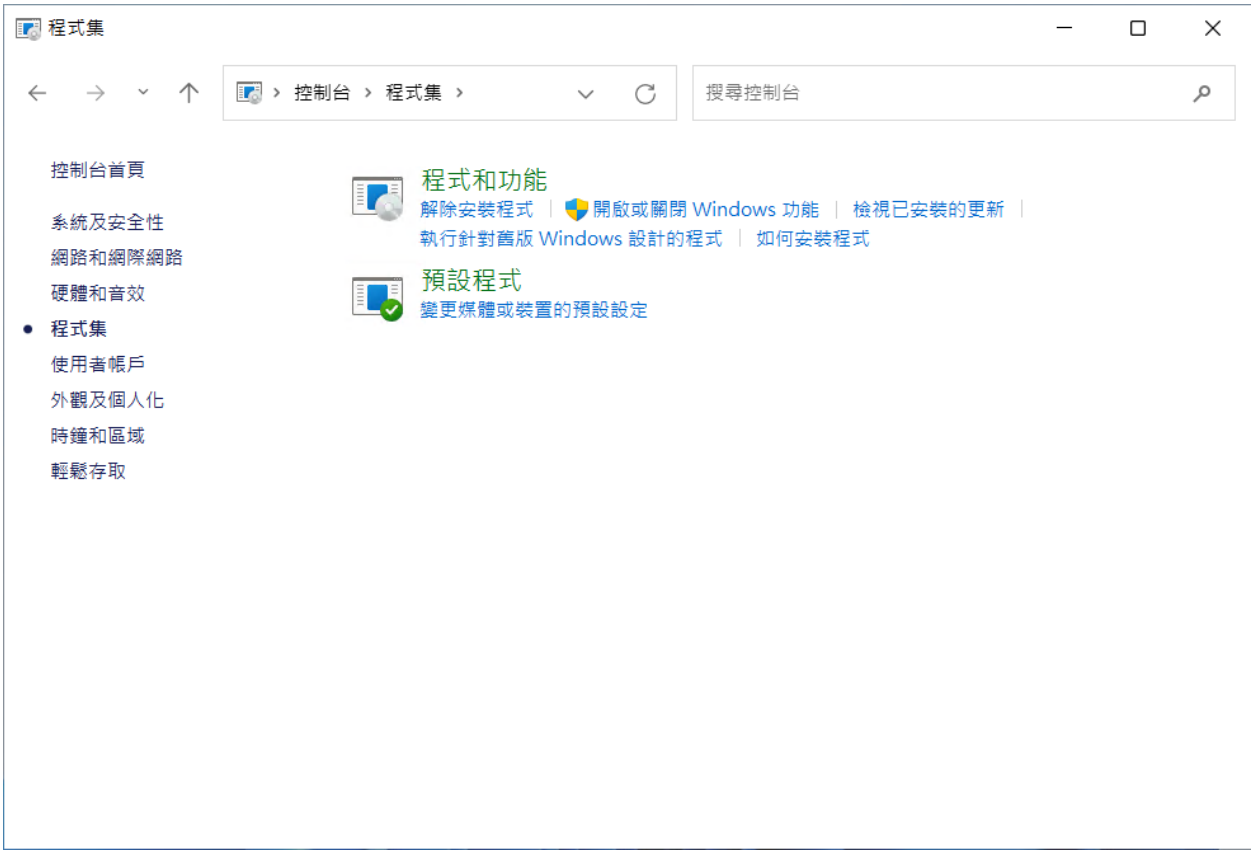
點選 [控制台] 圖示，啟動這個應用程式

找到綠色文字的 [程式集] 並點選這個文字

控制台

看到 [程式集] 視窗內容後

找到並且點選 [開啟或關閉 Windows 功能] 文字

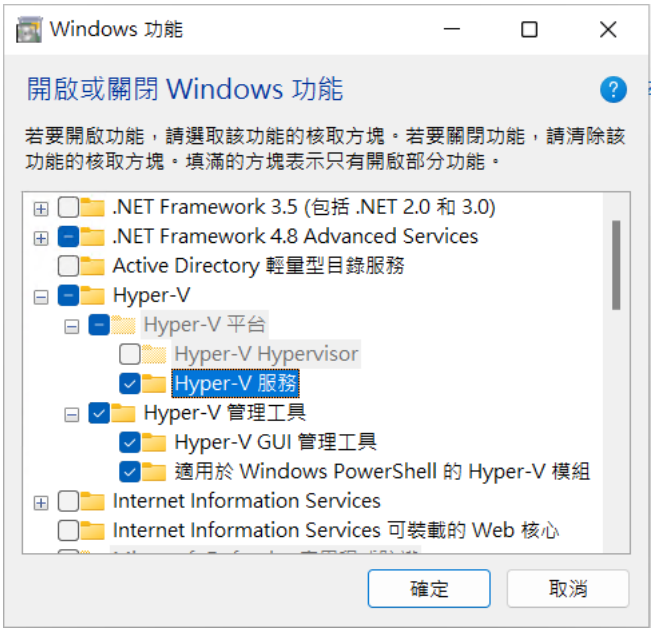


程式集

現在將會顯示出 [Windows 功能] 對話窗

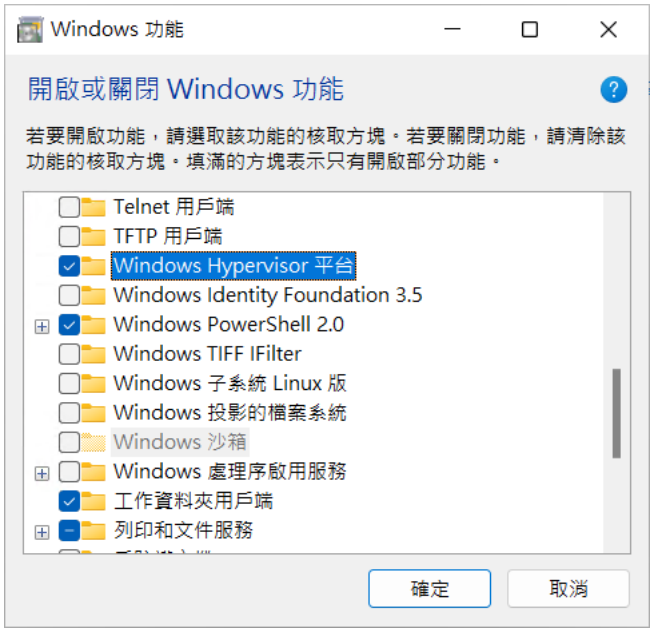
在此對話窗內，找到 Hyper-V 節點

勾選該節點內的所有子節點



Windows 功能 - Hyper-V

接著找到並且勾選這個 [Windows Hypervisor 平台] 節點



Windows 功能 - Windows Hypervisor 平台

最後，點選右下方的 [確定] 按鈕，開始進行安裝這兩個功能

一旦安裝完成之後，請在 [Windows 功能] 對話窗的右下方

找到並且點選 [立即重新啟動] 這個按鈕

Windows 功能 - Windows Hypervisor 平台

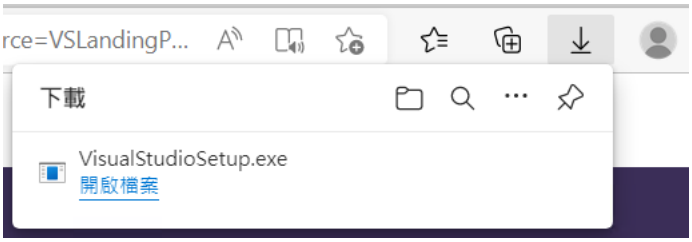
當作業系統重新開機完成後，便可以進行底下的 Visual Studio 2022 安裝工作

2.2 安裝 Visual Studio 2022 開發工具

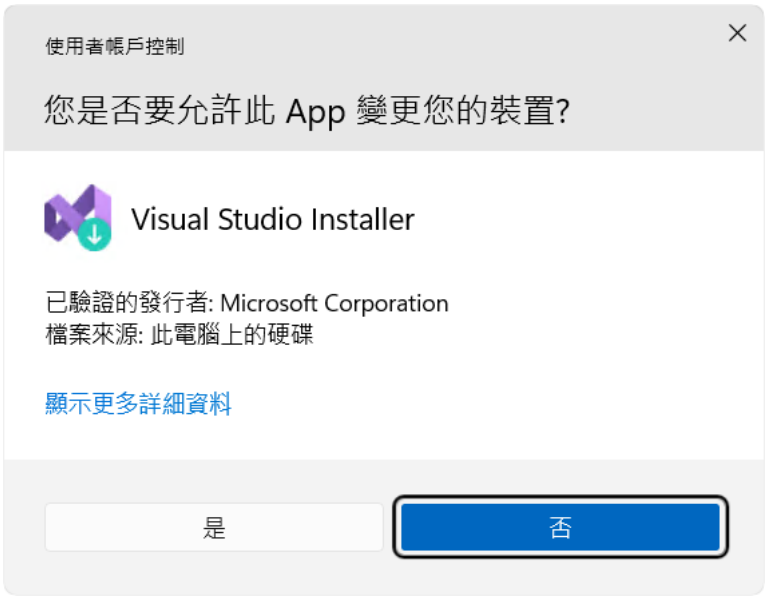
因此，先透過瀏覽器開啟 Visual Studio 2022[4] 下載網址

- 點選網頁上的 [下載 Visual Studio] 下拉選單按鈕
- 請下載 [Visual Studio Community] 這個版本即可，當然，下載其他的版本也是沒有問題的，其中 [Visual Studio Community] 版本是免費且功能完整的 IDE，適用於學生、開放原始碼參與者以及個別開發人員。
- 一旦 VisualStudioSetup.exe 安裝檔案下載完成後，請直接開啟這個檔案

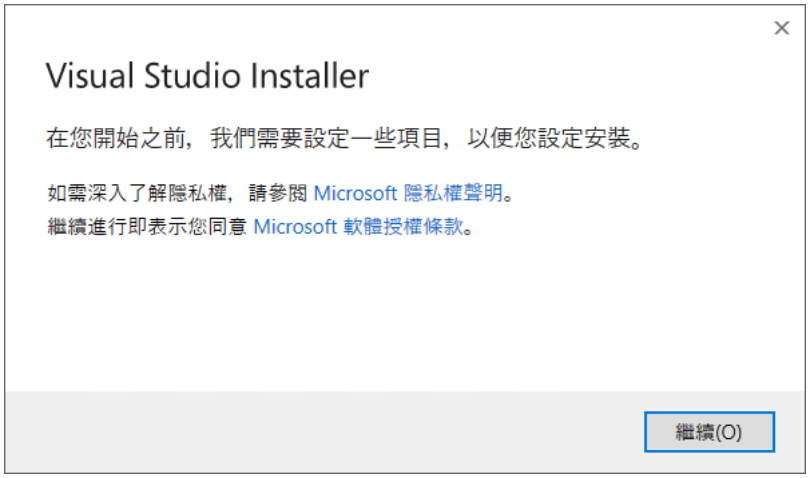
[4] https://visualstudio.microsoft.com/zh-hant/vs?WT.mc_id=DT-MVP-5002220



- 在出現 [使用者帳戶控制] 對話窗時候，點選 [是] 按鈕，接受允許此 App 變更您的裝置。

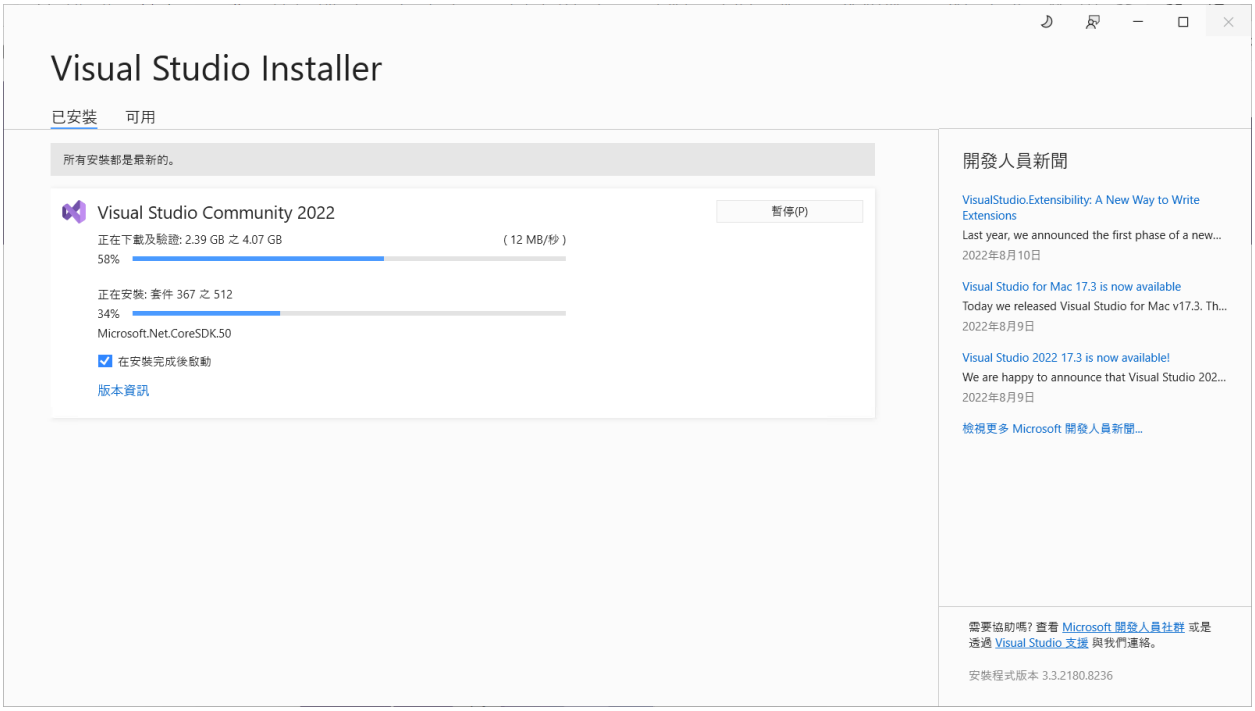


- 看到 [Visual Studio Installer] 對話窗後，點選該對話窗右下方的 [繼續] 按鈕

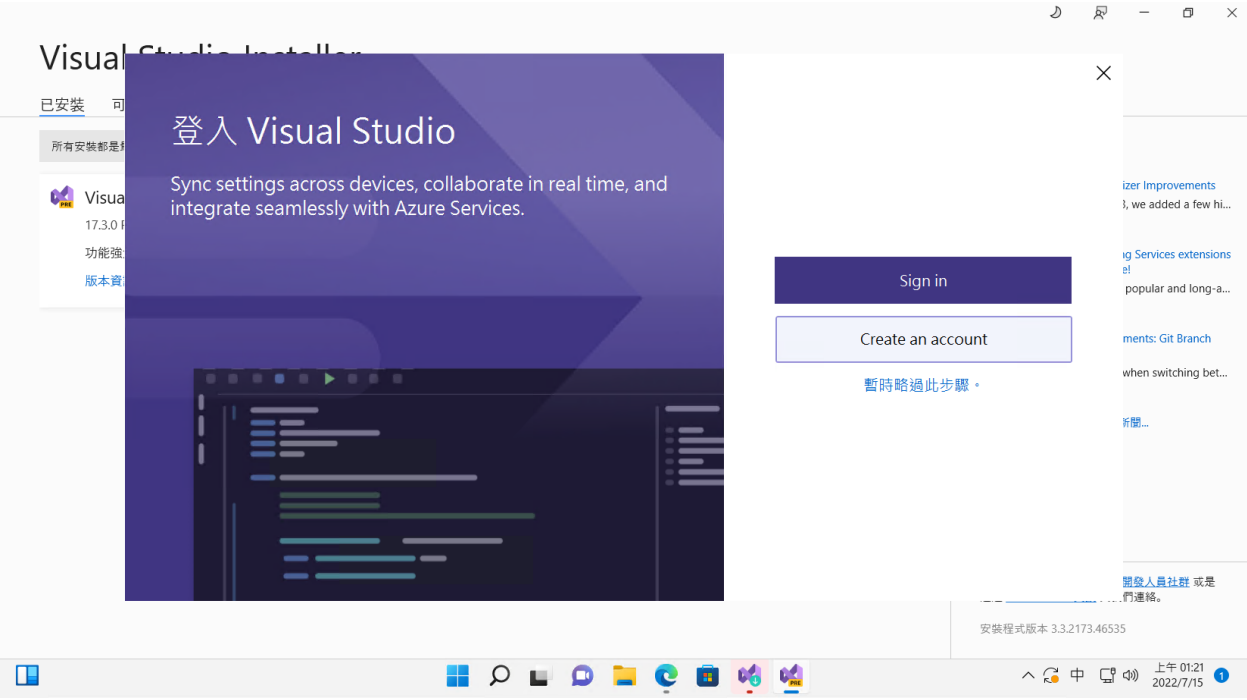


- 稍微等候一下，將會看到 Visual Studio Community 2022 安裝程式畫面
- 請勾選 [NET Multi-Platform App UI 開發] 這個選項，其中這個選項的說明內容為：使用 C# 與 .NET MAUI，從單一程式碼基底建置 Android, iOS, Windows 和 Mac 應用程式。
- 請在右方 [安裝詳細資料] 區域的最下方，找到並且勾選 [Xamarin] 選項

- 完成後，點選右下方的 [安裝] 按鈕
- 現在要稍微等候一段時間，因為安裝程式正在下載與安裝所需要用到的相關檔案



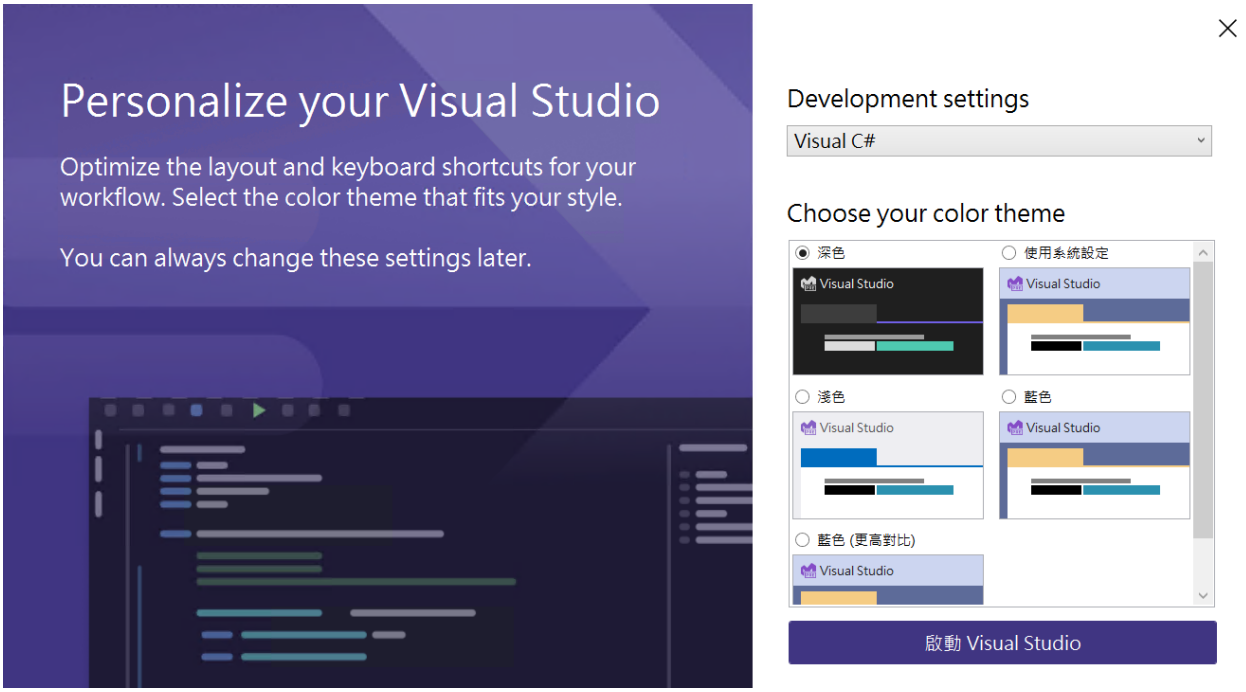
- 當看到底下的畫面，那就表示 Visual Studio 2022 已經安裝完成了



- 在此，點選右邊的 [暫時略過此步驟] 文字
- 建議對於 [Development settings] 的下拉選單控制項，選擇 [Visual C#]
- 對於 [Choose your color theme] 區段，可以依據自己喜好，選擇合適顏色配置主題佈景，在此，我選擇預設值

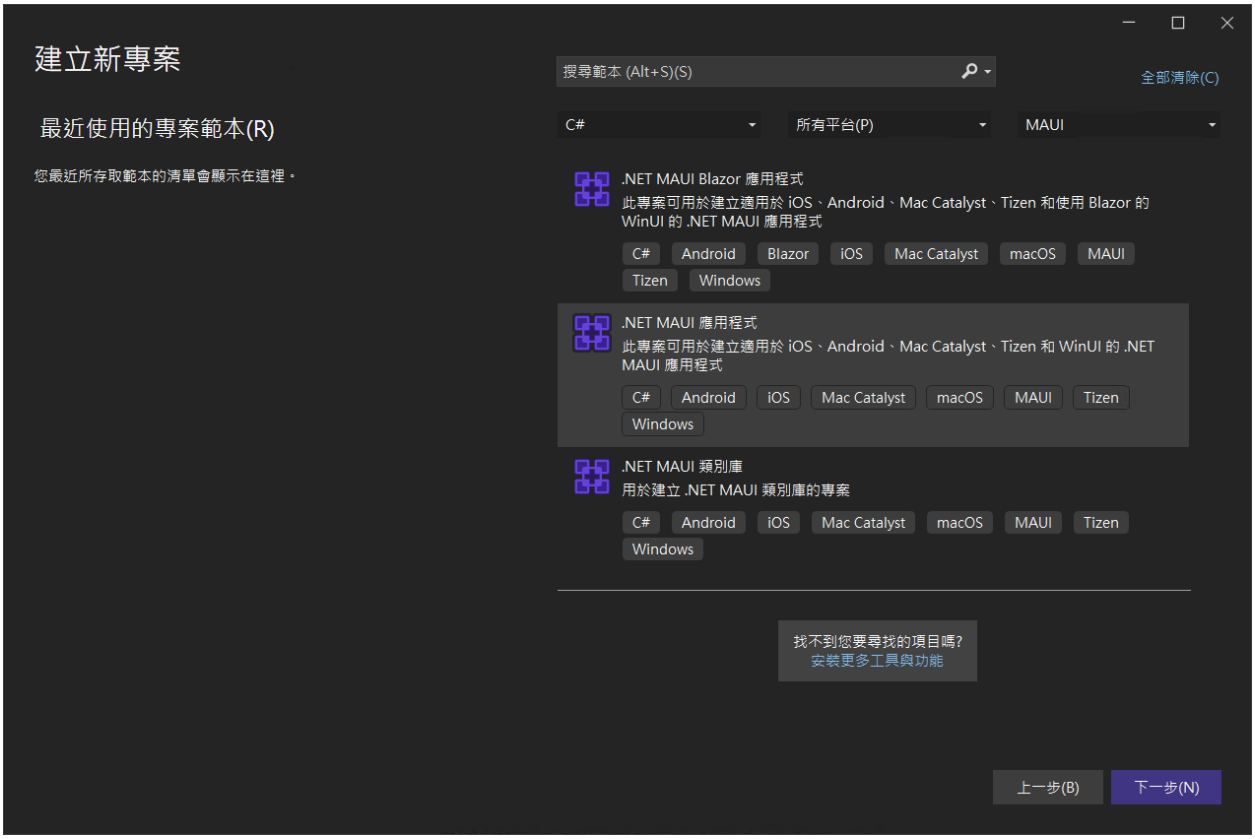
2.3 啟動、測試和設定開發環境

- 請點選右下方的 [啟動 Visual Studio] 按鈕

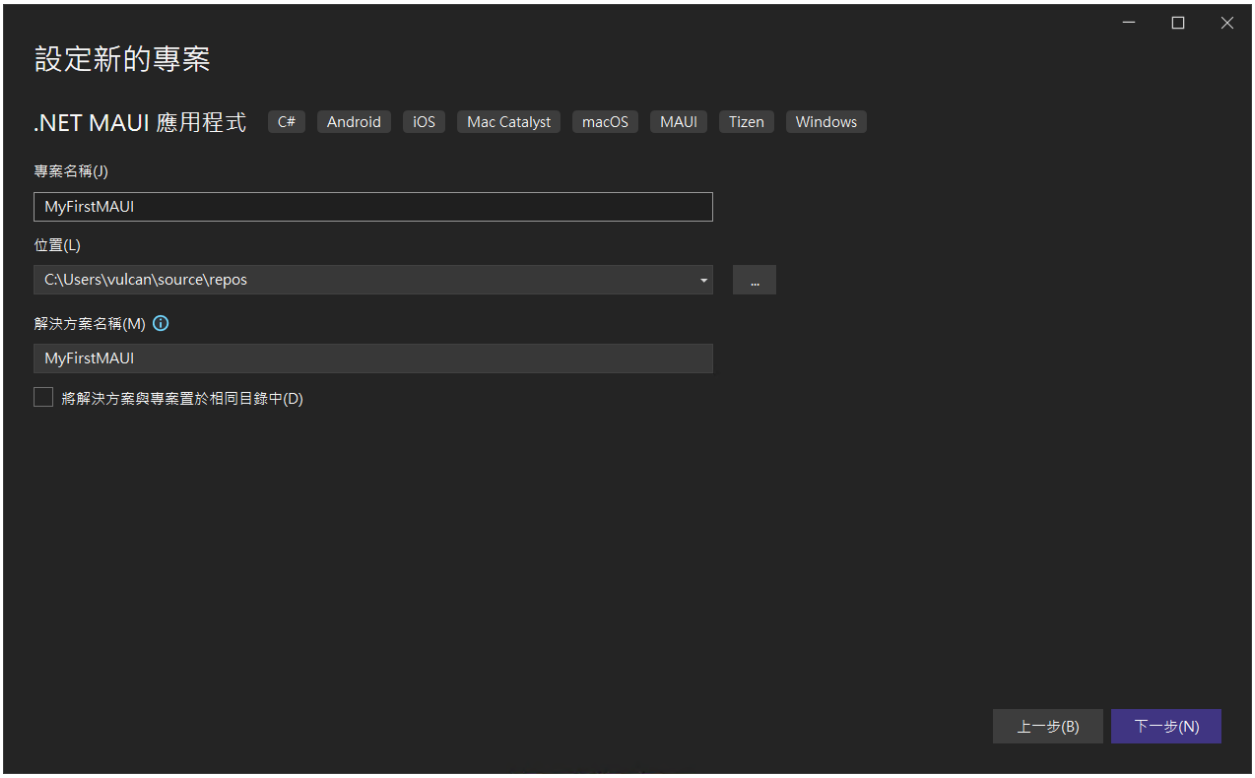


- 一旦 Visual Studio 2022 啟動成功後，就會看到 Visual Studio 2022 對話窗
- 請點選右下方的 [建立新的專案] 表示透過程式碼 Scaffolding 選擇專案範本以開始使用

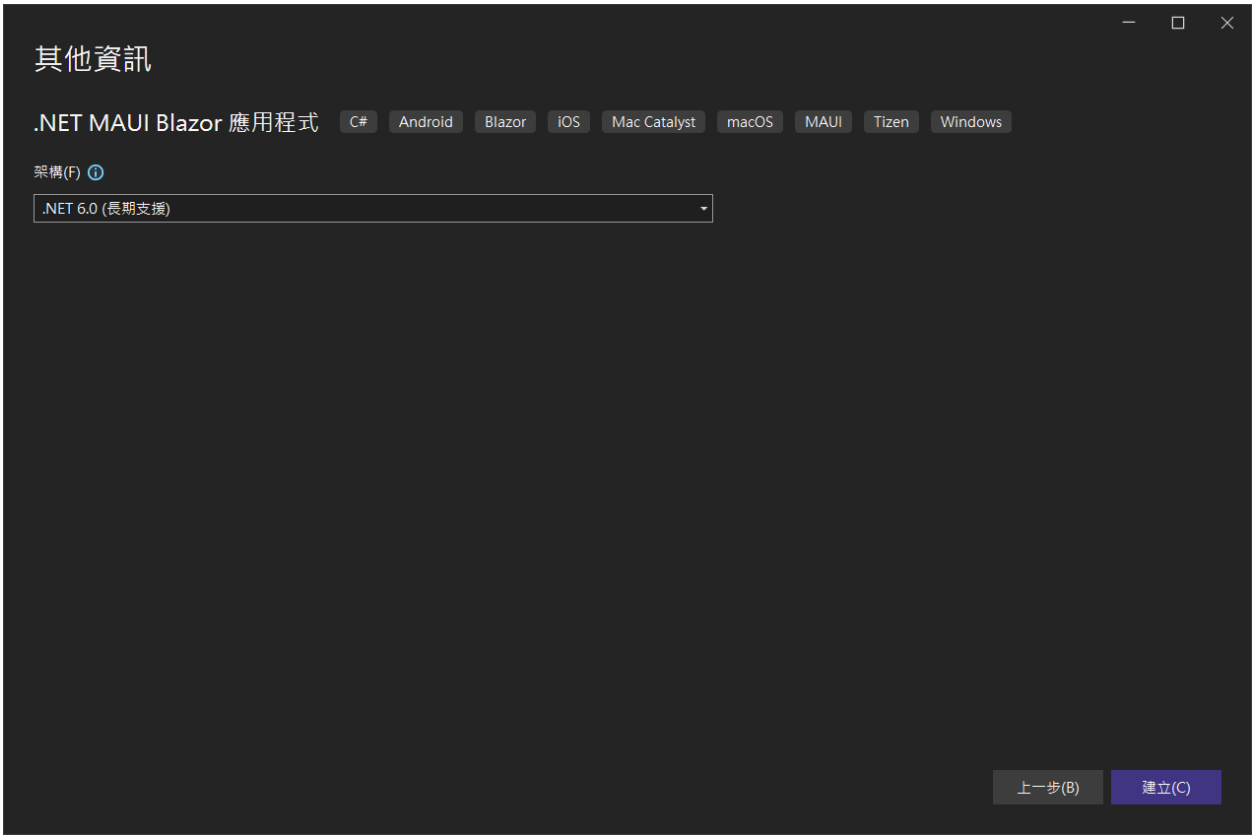
- 當出現 [建立新專案] 對話窗
- 在中間最上方有三個下拉選單控制項
- 切換 [所有語言] 下拉選單控制項為 [C#]
- 切換 [所有專案類型] 下拉選單控制項為 [MAUI]
- 此時，在中間區域將會看到有三種專案範本可以選擇
- 請點選中間那個 [.NET MAUI 應用程式] 此專案可用於建立適用於 iOS、Android、Mac Catalyst、Tizen 和 WinUI 的 .NET MAUI 應用程式
- 最後，點選右下方的 [下一步] 按鈕



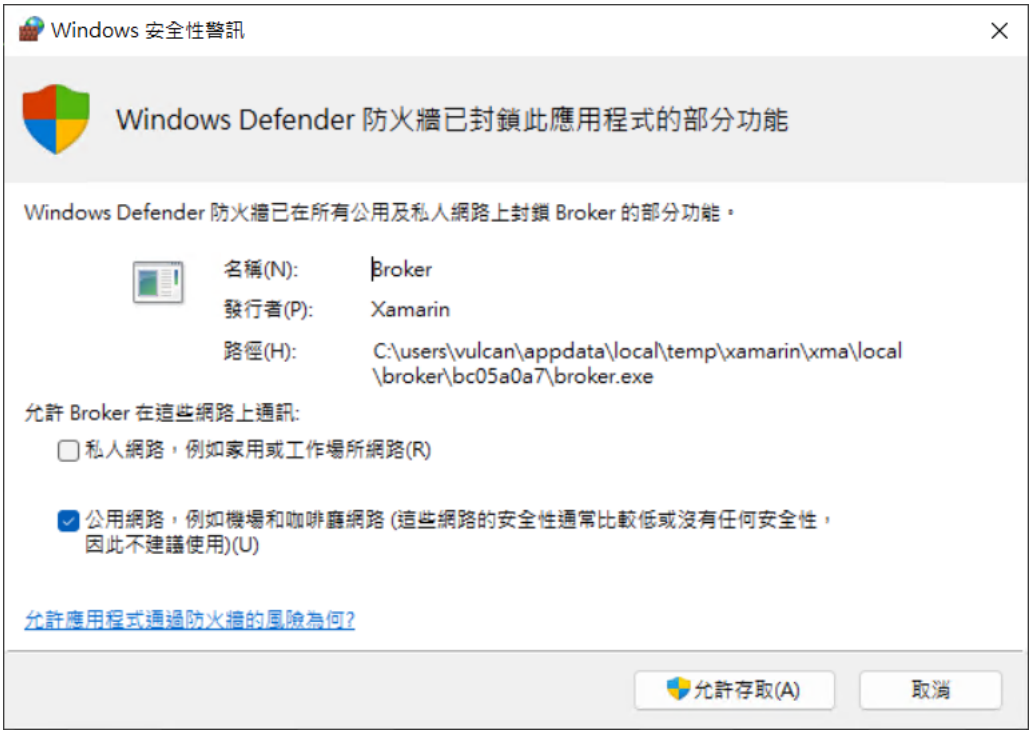
- 在 [設定新的專案] 對話窗出現後
- 在 [專案名稱] 欄位內輸入 `MyFirstMAUI`
- 點選右下方的 [下一步] 按鈕



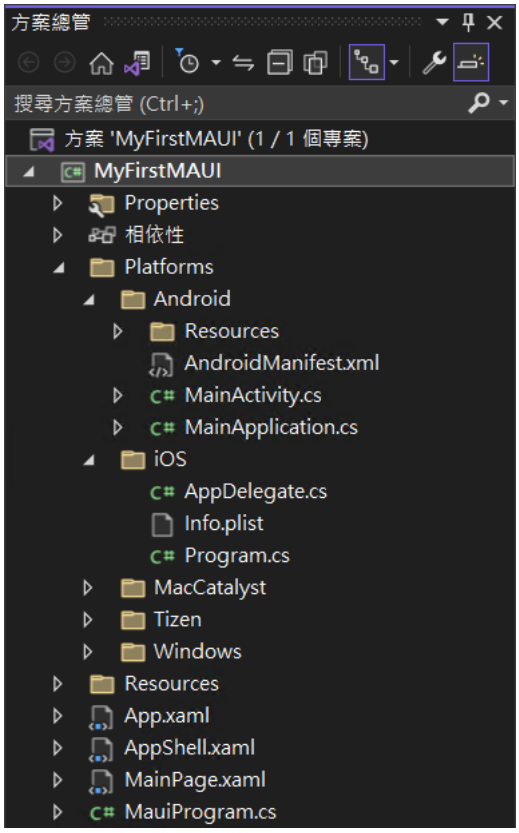
- 看到 [其他資訊] 對話窗，點選右下方的 [建立] 按鈕



- 稍微等候 Visual Studio 建立這個專案
- 現在會看到 [Windows 安全性警告] 對話窗出現，提示 Windows Defender 防火牆已封鎖此應用程式的部分功能
- 點選右下方的 [允許存取] 按鈕



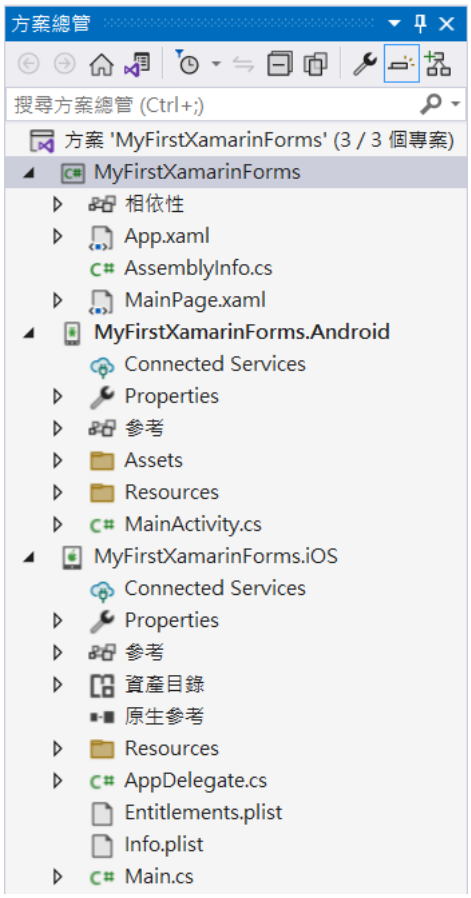
- 底下是建立好的 MAUI 整體方案的結構



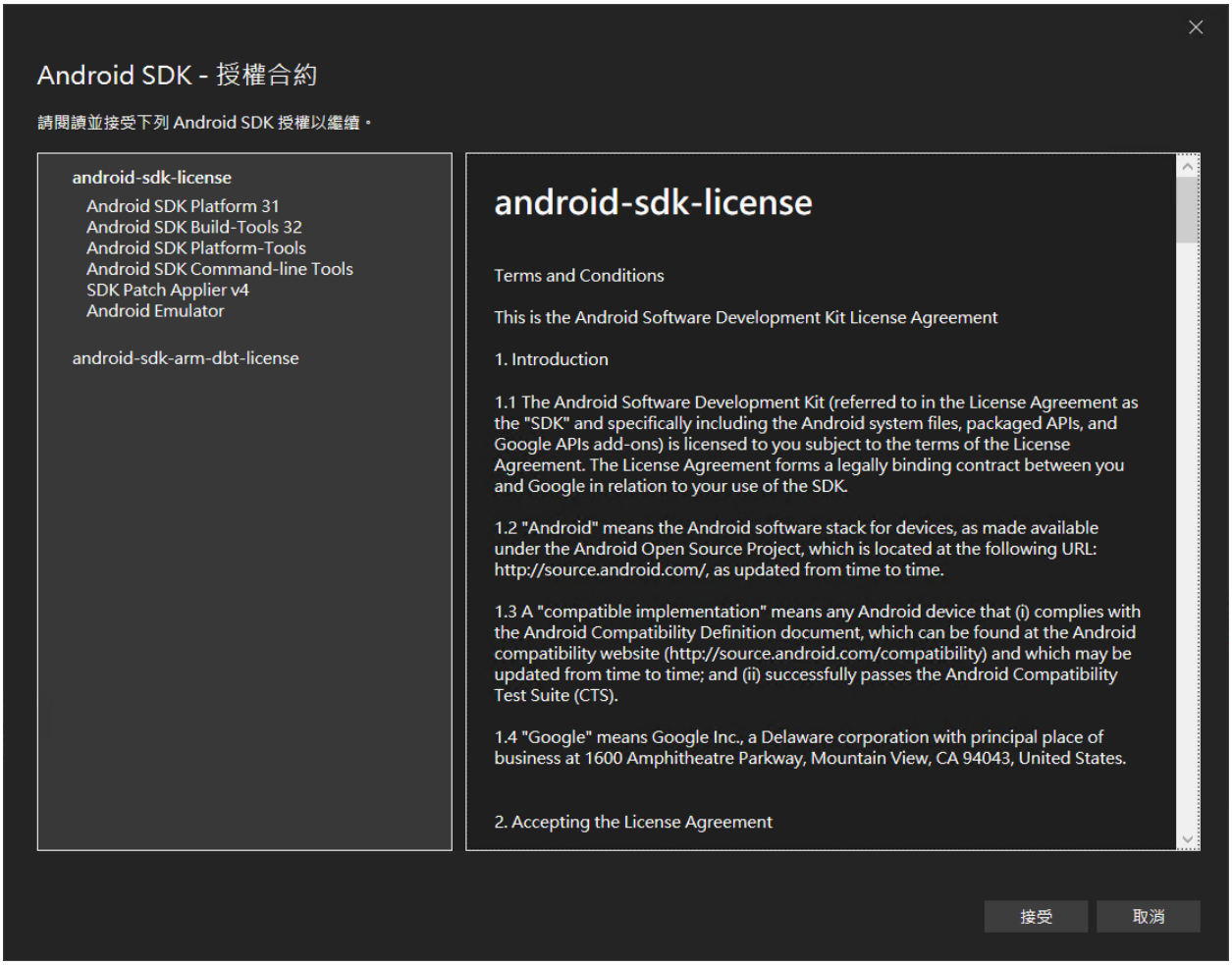
請另外開啟一個新的 Visual Studio，並且建立一個 Xamarin.Forms 的專案，底下將會是新建立一個 Xamarin.Forms 專案的操作說明的結果

[建立新的專案] > [行動應用程式 (Xamarin.Forms)] > [下一步] > [專案名稱] > `MyFirstXamarinForms` > [建立] > [空白] > [建立]

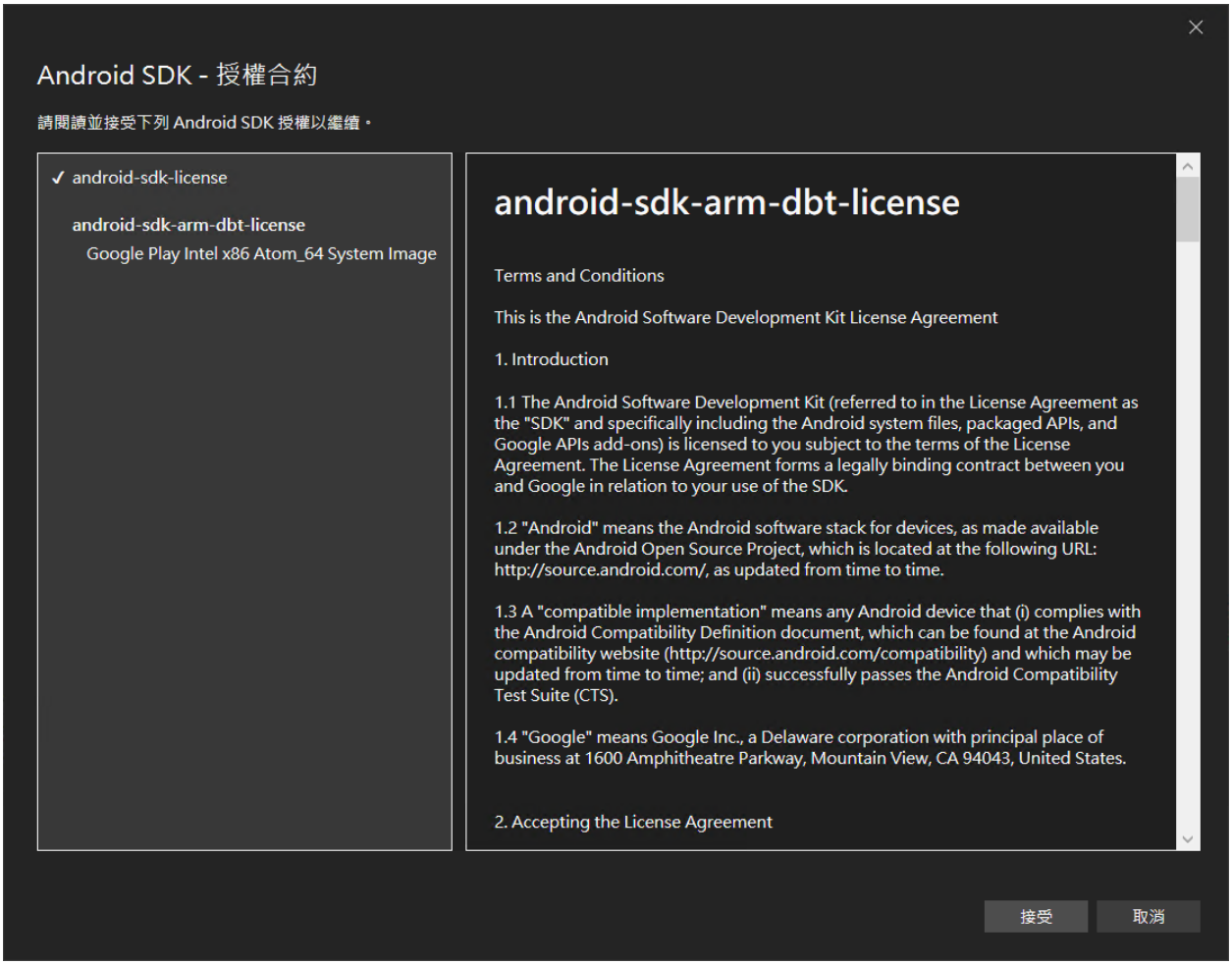
底下是建立好的 Xamarin.Forms 方案結構



若此時在畫面上看到一個對話窗 [Android SDK - 授權合約] ，請點選該對話窗右下方 [接受] 按鈕



若緊接著又在畫面上看到一個對話窗 [Android SDK - 授權合約] ，請點選該對話窗右下方 [接受] 按鈕



從這兩個方案總管的畫面可以看出

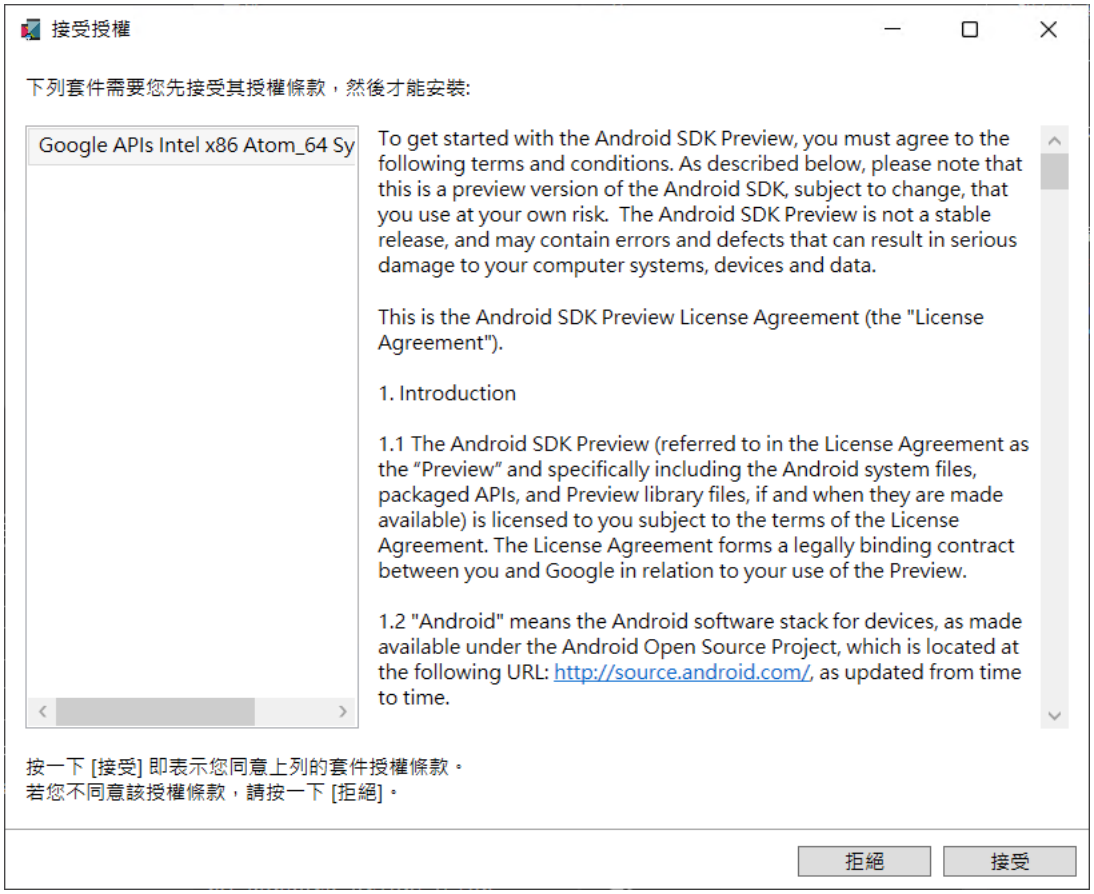
- MAUI 方案內，僅有一個專案存在，也就是說，採用 MAUI 方式來開發多個跨平台需求，僅需要一個專案就可以了，而在 Xamarin.Forms 專案內，則會看到一個 Xamarin.Forms 的共用專案，和多個原生專案，在這個例子中，將會看到有 Android 與 iOS 兩個專案，因此，可以同時開發出這兩個平台的跨平台專案
- 資源可以共用，對於像是圖片的資源，可以放在該專案根目錄下的 [Resources] 資料夾內，這樣每個各別平台專案，是可以共用這個圖片資源的

2.4 建立 Android 模擬器與實際體驗執行過程

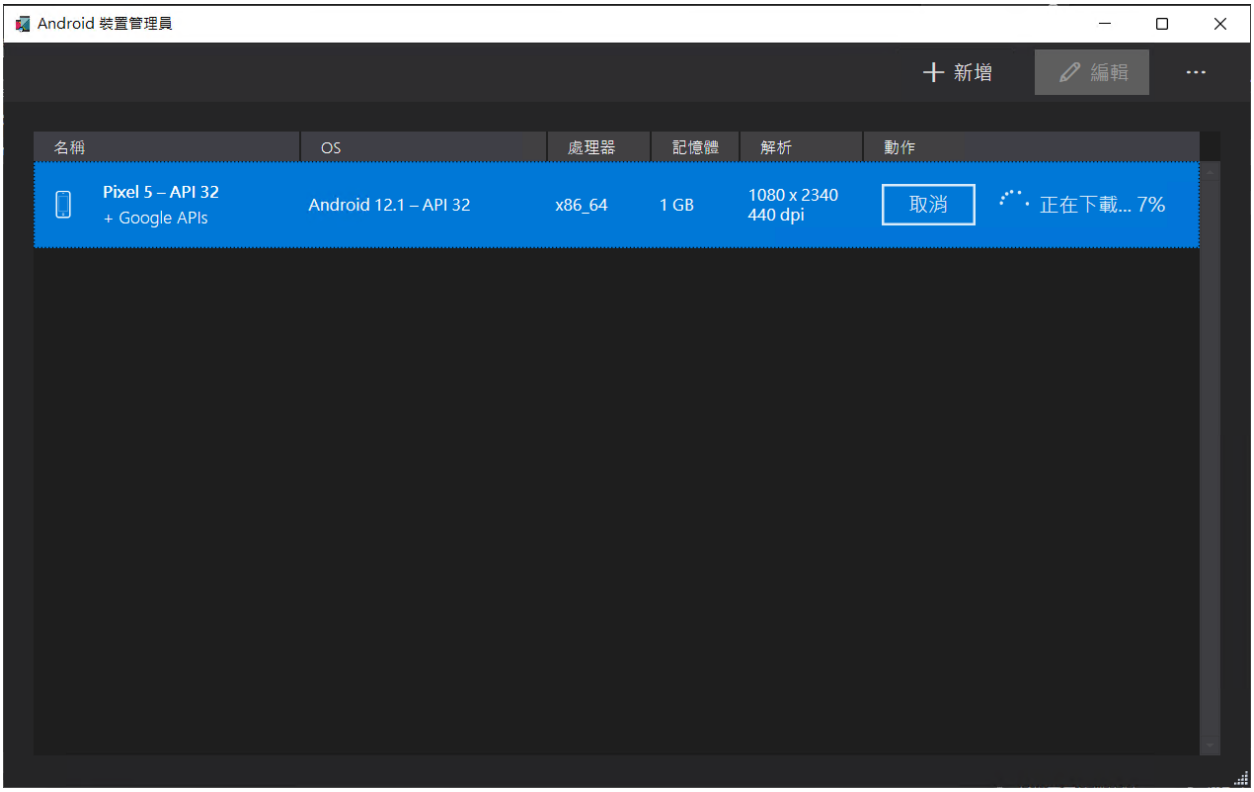
- 從功能表中，點選 [工具] > [Android] > [Android 裝置管理員] 選項
- 在出現 [使用者帳戶控制] 對話窗時候，點選 [是] 按鈕，接受允許此 App 變更您的裝置。

- 當 [Android 裝置管理員] 對話窗出現後
- 點選右上角的 [+ 新增] 按鈕
- 在 [新增裝置] 對話窗內，點選右下角的 [建立] 按鈕，接受預設建議的模擬器

- 現在，將會看到 [接受授權] 這個對話窗畫面
- 點選右下角的 [接受] 按鈕，以便繼續安裝 Android SDK 檔案



- 此時，[Android 裝置管理員] 便會開始下載剛剛指定的模擬器檔案，接著會進行解壓縮過程，與安裝到這台電腦上，因此，需要等候一段時間



- 依但該模擬器建立完成後
- 請點選該模擬器項目最右方的 [啟動] 按鈕

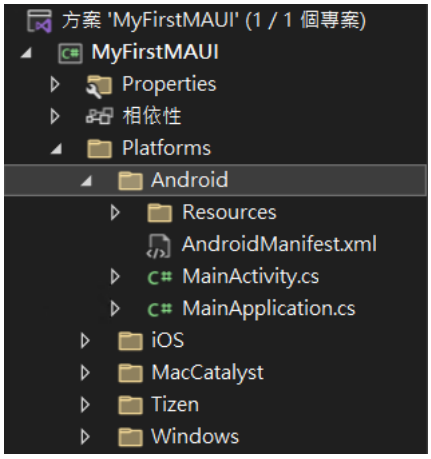
2.5 了解 MAUI 專案的結構

總結來說，MAUI 就是 Xamarin.Forms 的進化版本，只不過最大的特色，就是把原先分別存在於不同專案中的原生專案，全部都整合到一個專案內，所以，接下來看看，對於每個元生平的進入點程式碼長的怎麼樣。

在這個 MAUI 專案內，將會看到 [Platforms] 資料夾，該資料夾的下方有其他資料夾，這些資料夾的名稱分別是每個平台的名稱，現在要分別進入到這些資料夾內來查看

2.5.1 Android 平台的進入點程式碼

首先看到的是 Android 平台的進入點程式碼



這個 [MainApplication.cs] 檔案，是當 Android 平台應用程式一啟動之後，就會優先要執行的程式碼，其中，在最下方有個 `MauiProgram.CreateMauiApp()` 表示式，將會回傳一個 MauiApp 型別物件，這個靜態方法將會在最後面來了解，而每個專屬平台的啟動程式，都會需要呼叫這個方法，準備開始執行共用程式碼的部分，也就是之前所謂的 Xamarin.Forms 中的程式碼。

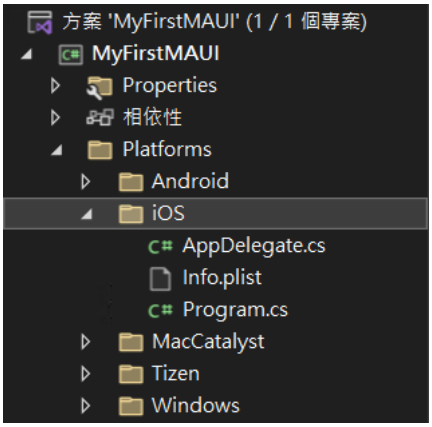
```
using Android.App;
using Android.Runtime;

namespace MyFirstMAUI;

[Application]
public class MainApplication : MauiApplication
{
        public MainApplication(IntPtr handle, JniHandleOwnership ownership)
                : base(handle, ownership)
        {
        }

        protected override MauiApp CreateMauiApp() => MauiProgram.CreateMauiApp();
}
```

這個 [MainActivity.cs] 會是 Android App 啟動之後，第一個畫面要執行的 Activity ，這裡的內容與做法，將會與 Xamarin.Androi 中相同。

```
using Android.App;
using Android.Content.PM;
using Android.OS;

namespace MyFirstMAUI;

[Activity(Theme = "@style/Maui.SplashTheme", MainLauncher = true, ConfigurationChang\
es = ConfigChanges.ScreenSize | ConfigChanges.Orientation | ConfigChanges.UiMode | C\
onfigChanges.ScreenLayout | ConfigChanges.SmallestScreenSize | ConfigChanges.Density\
)]
public class MainActivity : MauiAppCompatActivity
{
}
```

2.5.2 iOS 平台的進入點程式碼

接下來要來看到的是 iOS 平台的進入點程式碼



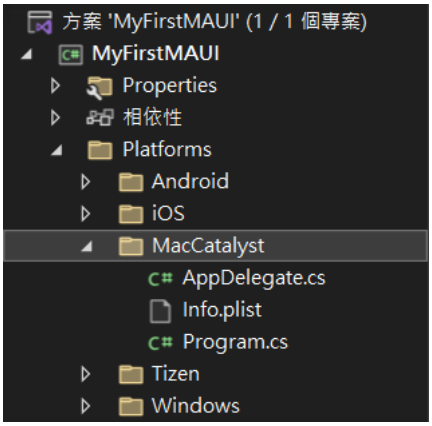
這個 [Program.cs] 檔案內，僅有一個 Main 方法，這是 iOS 應用程式第一個要執行的方法

```
using ObjCRuntime;
using UIKit;

namespace MyFirstMAUI;

public class Program
{
        // This is the main entry point of the application.
        static void Main(string[] args)
        {
                // if you want to use a different Application Delegate class from "AppDeleg
                // you can specify it here.
                UIApplication.Main(args, null, typeof(AppDelegate));
        }
}
```

而對於 [AppDelegate.cs] 檔案而言，這裡的用法也是與 Xamarin.Forms 中的 Xamarin.iOS 用法相同，更詳盡的使用方式，也可以參考 iOS SDK 文件。在這個檔案中，同樣的將會透過 `MauiProgram.CreateMauiApp()` 取得要執行共用程式碼的物件

```
using Foundation;

namespace MyFirstMAUI;

[Register("AppDelegate")]
public class AppDelegate : MauiUIApplicationDelegate
{
        protected override MauiApp CreateMauiApp() => MauiProgram.CreateMauiApp();
}
```

2.5.3 MacCatalyst 平台的進入點程式碼



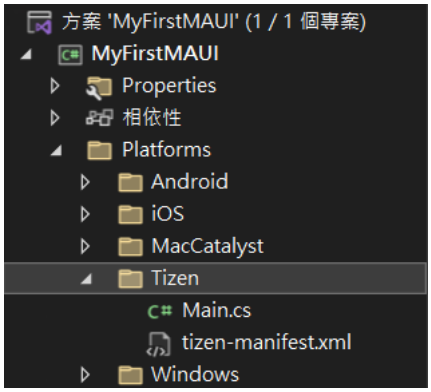
與 iOS 專屬程式碼相同，這個 [Program.cs] 檔案內，僅有一個 Main 方法，這是 MacCatalyst 應用程式第一個要執行的方法

```
using ObjCRuntime;
using UIKit;

namespace MyFirstMAUI;

public class Program
{
        // This is the main entry point of the application.
        static void Main(string[] args)
        {
                // if you want to use a different Application Delegate class from "AppDeleg
                // you can specify it here.
                UIApplication.Main(args, null, typeof(AppDelegate));
        }
}
```

而對於 [AppDelegate.cs] 檔案而言，這裡的用法也是與 iOS 用法相同，在這個檔案中，同樣的將會透過 `MauiProgram.CreateMauiApp()` 取得要執行共用程式碼的物件

```
using Foundation;

namespace MyFirstMAUI;

[Register("AppDelegate")]
public class AppDelegate : MauiUIApplicationDelegate
{
        protected override MauiApp CreateMauiApp() => MauiProgram.CreateMauiApp();
}
```

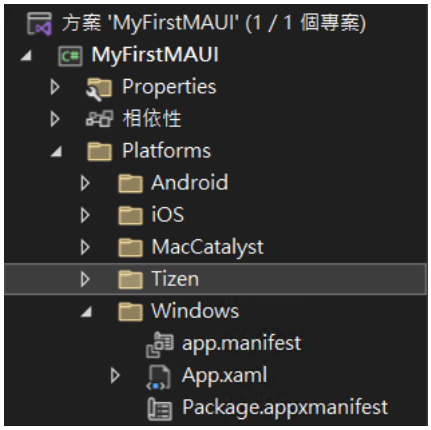
2.5.4 Tizen 平台的進入點程式碼



在 Tizen 平台下，將會是從 [Main.cs] 這個檔案開始進行執行，同樣的這裡也會透過 `MauiProgram.CreateMauiApp()` 取得要執行共用程式碼的物件

```
using System;
using Microsoft.Maui;
using Microsoft.Maui.Hosting;

namespace MyFirstMAUI;

class Program : MauiApplication
{
        protected override MauiApp CreateMauiApp() => MauiProgram.CreateMauiApp();

        static void Main(string[] args)
        {
                var app = new Program();
                app.Run(args);
        }
}
```

2.5.5 Windows 平台的進入點程式碼



最後則是 Windows 平台，這裡將會使用 WinUI 來執行一個 Windows 應用程式，底下將會是 [App.xaml] 內容

```
<maui:MauiWinUIApplication
    x:Class="MyFirstMAUI.WinUI.App"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:maui="using:Microsoft.Maui"
    xmlns:local="using:MyFirstMAUI.WinUI">

</maui:MauiWinUIApplication>
```

而在這個 [App.xaml] 檔案內 code behind 程式碼，將要打開這個 [App.xaml.cs] 節點，就會看到

```
using Microsoft.UI.Xaml;

// To learn more about WinUI, the WinUI project structure,
// and more about our project templates, see: http://aka.ms/winui-project-info.

namespace MyFirstMAUI.WinUI;

/// <summary>
/// Provides application-specific behavior to supplement the default Application cla\
ss.
/// </summary>
public partial class App : MauiWinUIApplication
{
        /// <summary>
        /// Initializes the singleton application object.  This is the first line of autho
ed code
        /// executed, and as such is the logical equivalent of main() or WinMain().
        /// </summary>
        public App()
        {
                this.InitializeComponent();
        }

        protected override MauiApp CreateMauiApp() => MauiProgram.CreateMauiApp();
}
```

2.5.6 MauiProgram.cs

在上面的五個專屬平台，都將會透過 [MauiProgram] 這個靜態類別內的 [CreateMauiApp()] 方法，來取得 [MauiApp] 這個物件

```
namespace MyFirstMAUI;

public static class MauiProgram
{
        public static MauiApp CreateMauiApp()
        {
                var builder = MauiApp.CreateBuilder();
                builder
                        .UseMauiApp<App>()
                        .ConfigureFonts(fonts =>
                        {
                                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                                fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
                        });

                return builder.Build();
        }
}
```

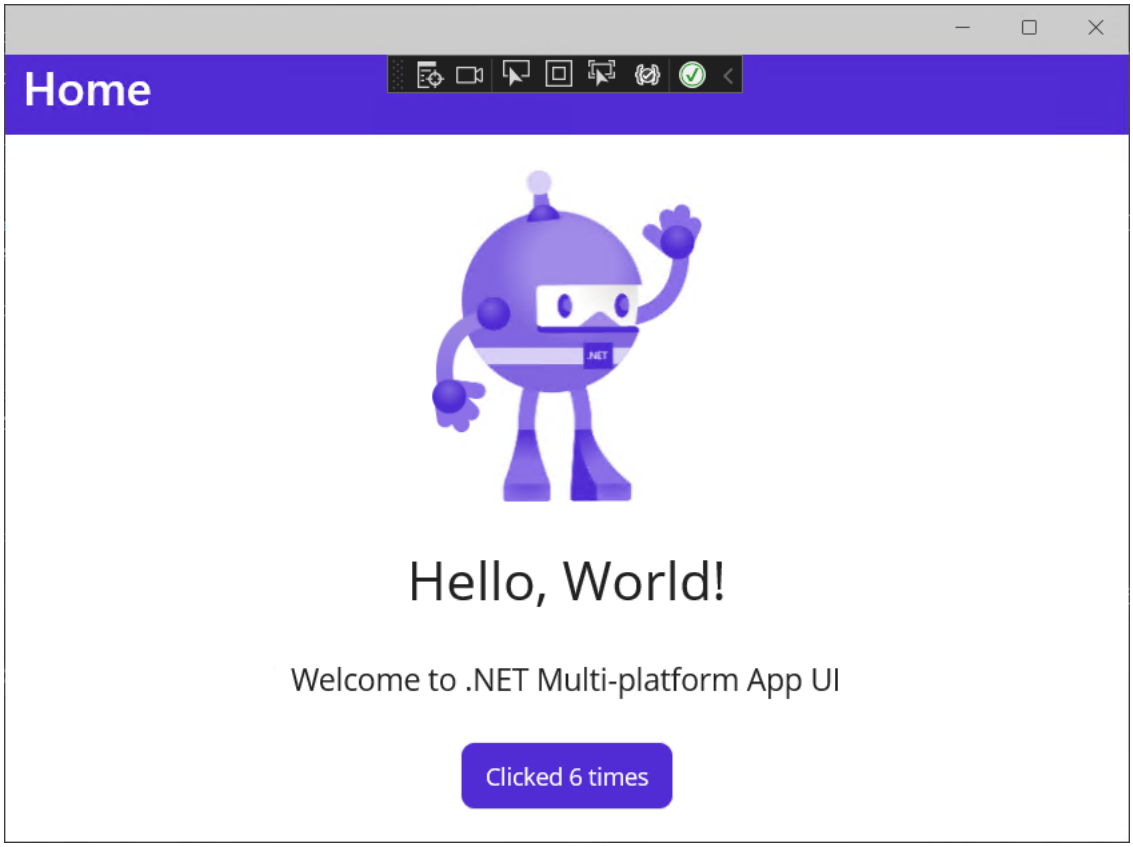
2.6 開始執行 Maui 應用程式

首先，測試一下在 Windows 平台下的執行結果

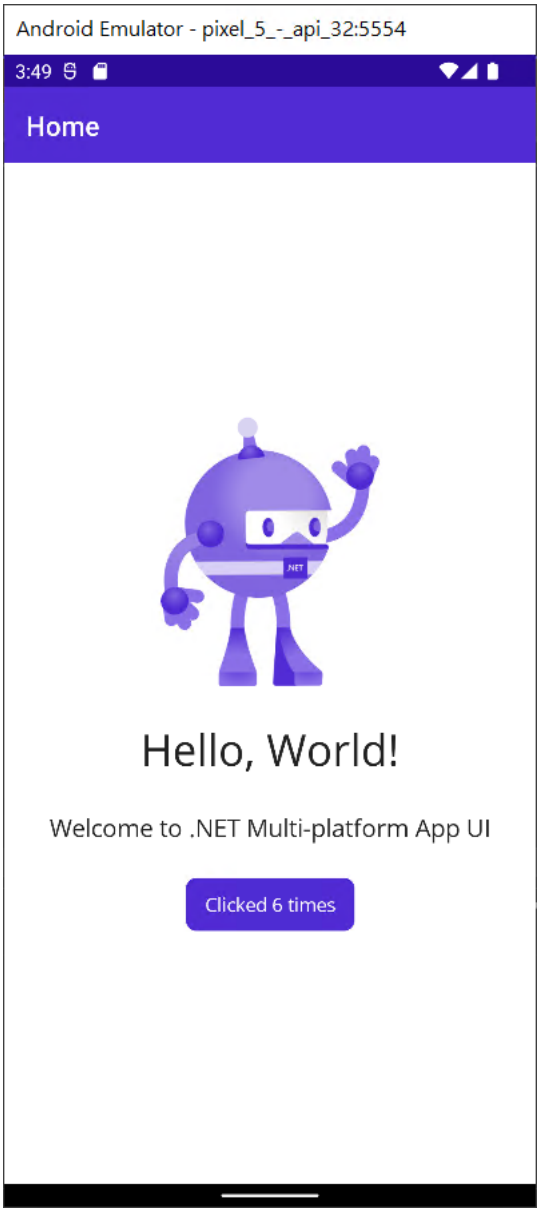
- 在最上方工具列中間區域，會看到一個綠色實體三角形，請下拉這個下拉選單控制項，從這裡個清單中，選擇 [Windows Machine] 這個選項

- 不過，將會看到底下的警告對話視窗 [啟用 Windows 開發人員模式]，這裡提到必須正確設定此裝置，才能為 Windows 開發此類型的應用程式。若未安裝，則無法在將應用程式提交至 Windows 市集之前安裝並測試應用程式。

- 點選該對話窗上藍色的文字 [是用於開發人員的設定]
- 一旦設定可以在開發人員模式下運行
- 將會看到這個 Windows App 出現在畫面上



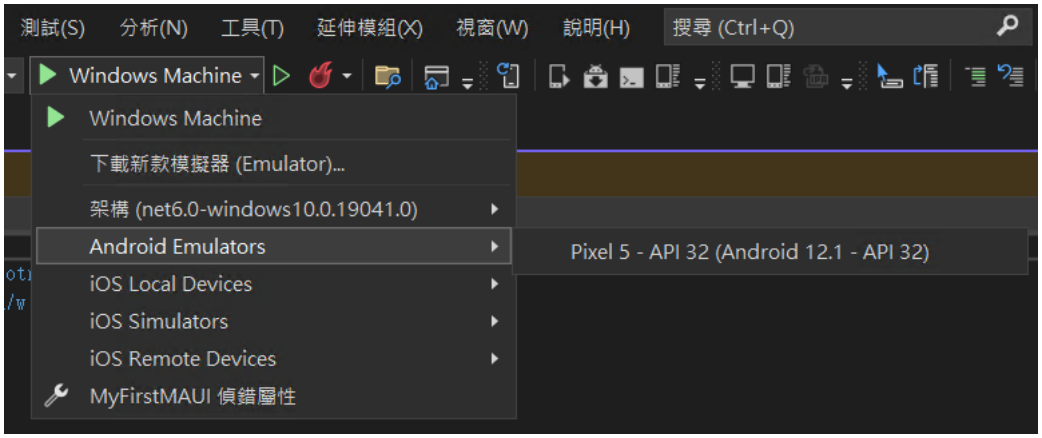
- 停止執行這個應用程式
- 在最上方工具列中間區域，會看到一個綠色實體三角形，請下拉這個下拉選單控制項，從這裡個清單中，選擇 [Android Emulators] > [Pixel 5 - API 32 (Android 12.1 - API 32)] 這個選項
- 此時將會出現一個 [Android SDK - 授權合約] 對話窗
- 請點選該對話窗右下方的 [接受] 按鈕
- 在出現 [使用者帳戶控制] 對話窗時候，點選 [是] 按鈕，接受允許此 App 變更您的裝置。
- 等候下載與安裝 Android SDK
- 完成後
- 請再次重新執行這個 Android 應用程式
- 此時，將會在模擬器中，看到這個 MAUI 應用程式了



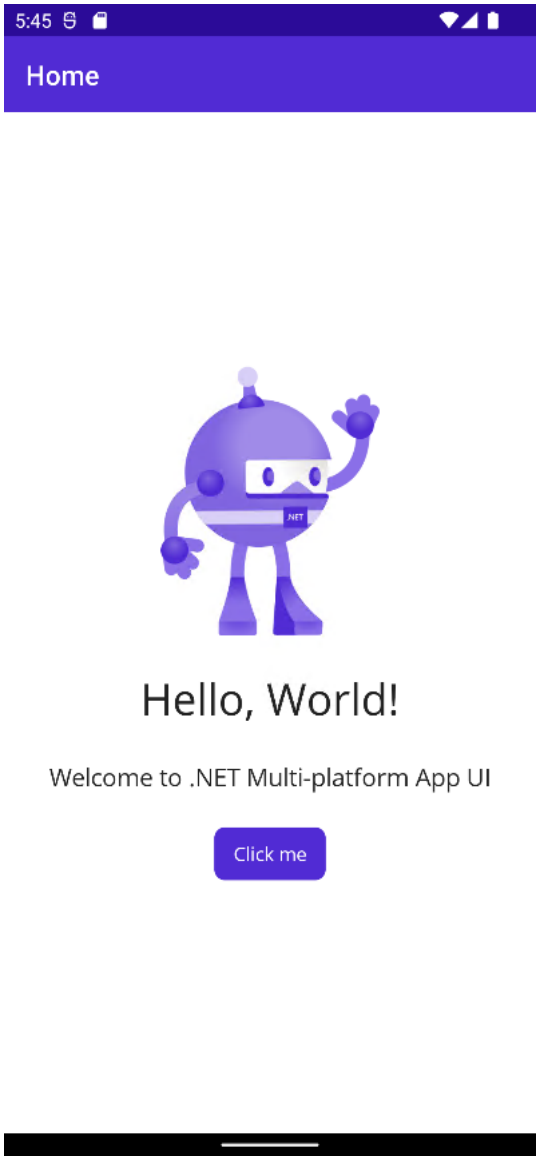
接下來，要來測試一下在 Android 平台下的模擬器執行結果

- 在最上方工具列中間區域，會看到一個綠色實體三角形，請下拉這個下拉選單控制項，從這裡個清單中，選擇 [Android Emulators] 這個選項
- [Android Emulators] 這個選項旁將會看到剛剛建立起來的 Android 模擬器選項 [Pixel 5 - API 32 (Android 12.1 - API 32)]

- 請點選 [Pixel 5 - API 32 (Android 12.1 - API 32)] 這個選項



- 稍待一會，將會看到這個 MAUI 應用程式已經成功在 Android 模擬器下執行起來了。



2.7 安裝 Prism Template 來建立 MAUI 應用程式

在這個時間點，想要使用 Prism.Maui 來進行開發 MAUI 應用程式，最快與最有效的方式那就是安裝 [Prism Template for MAUI] 這個工具，其可以快速地建立起一個使用 Prism 開發框架來開發的 MAUI 專案。

首先，要先安裝 Prism Template for MAUI ，想要做到這樣的需求，請先使用管理者權限來開啟命令提示字元視窗，接著，在該視窗內輸入

dotnet new –install Prism.Templates::8.1.97

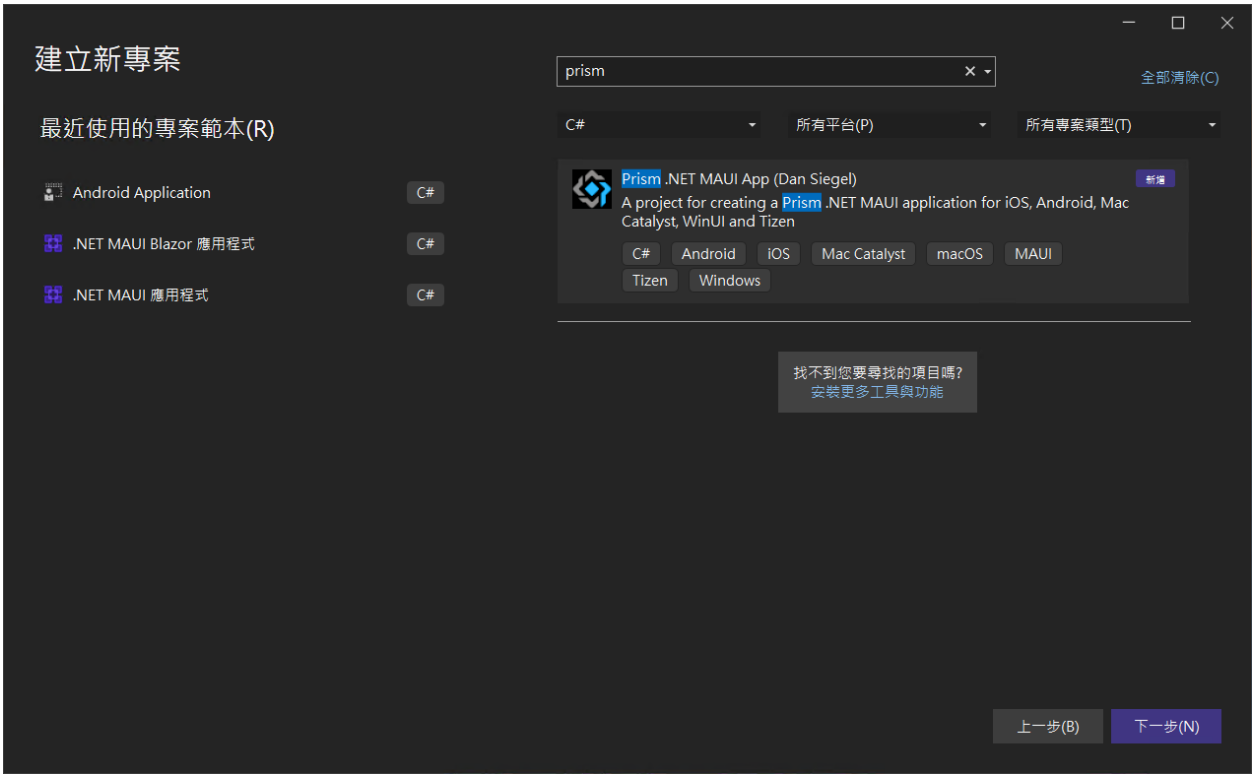
就可以把這個專案範本安裝到 Visual Studio 內，底下是下達這個命令之後的執行結果內容。

```
C:\Windows\system32>dotnet new --install Prism.Templates::8.1.97

歡迎使用 .NET 6.0!
---------------------
SDK 版本: 6.0.400-preview.22330.6

遙測
---------
.NET 工具會收集使用資料，協助我們改進您的體驗。資料會由 Microsoft 收集，並分享給社群使用。您可以選
LI_TELEMETRY_OPTOUT 環境變數設定為 '1' 或 'true' 即可。

閱讀更多有關 .NET CLI 工具遙測的內容: https://aka.ms/dotnet-cli-telemetry

----------------
已安裝 ASP.NET Core HTTPS 開發憑證。
若要信任憑證，請執行 'dotnet dev-certs https --trust' (僅限 Windows 與 macOS )。
深入了解 HTTPS: https://aka.ms/dotnet-https
----------------
撰寫第一個應用程式: https://aka.ms/dotnet-hello-world
了解全新功能: https://aka.ms/dotnet-whats-new
探索文件: https://aka.ms/dotnet-docs
於 GitHub 回報問題和尋找來源: https://github.com/dotnet/core
Use 'dotnet --help' 查看可用的命令或瀏覽: https://aka.ms/dotnet-cli
--------------------------------------------------------------------------------\
--
將安裝下列範本套件:
   Prism.Templates::8.1.97

成功: Prism.Templates::8.1.97 已安裝下列範本:
範本名稱                            簡短名稱          語言   標記
------------------------------  --------------  ----  --------------------------\
```

```
---------------------------------
Prism .NET MAUI App               prism-maui       [C#]  MAUI/Android/iOS/macOS/Mac C\
atalyst/Windows/Tizen
Prism Blank App (Uno Platform)    uno-blank        [C#]  Prism/Xamarin/Uno Platform/W\
ebAssembly/iOS/Android/WinUI/UWP
Prism Blank App (WPF)             wpf-core-blank   [C#]  Desktop
Prism Blank App (Xamarin.Forms)   xf-blank         [C#]  Prism/Xamarin/Xamarin.Forms
Prism Full App (WPF)              wpf-core-full    [C#]  Desktop
Prism Module (WPF)                wpf-module-core  [C#]  Desktop
Prism Module (Xamarin)            xf-module        [C#]  Prism/Xamarin/Xamarin.Forms
```

2.8 建立一個可以支援 Prism 開發框架的 MAUI 專案

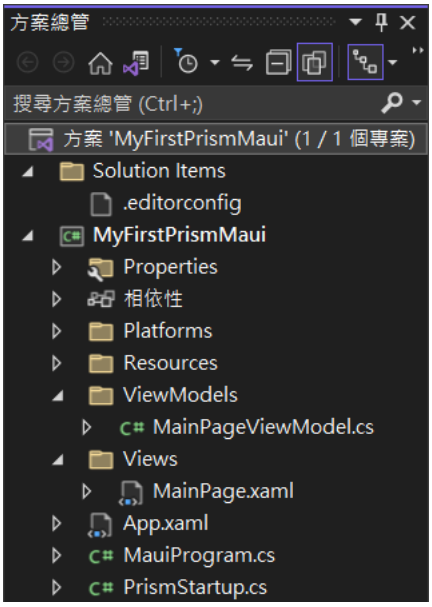
- 開啟 Visual Studio 2022 版本
- 點選螢幕右下角的 [建立新的專案] 按鈕
- 在最上方的 [搜尋範本] 文字輸入盒內
- 輸入 `prism` 找出可用的專案範本



- 現在可以從 [建立新專案] 對話窗內出現了 [Prism .NET MAUI App (Dan Siegel)] 這個專案範本
- 選擇這個專案範本
- 點選右下角的 [下一步] 按鈕
- 當出現了 [設定新的專案] 對話窗
- 在 [專案名稱] 欄位內，輸入 `MyFirstPrismMaui`
- 點選右下角的 [建立] 按鈕

現在使用 Prism 開發框架的 MAUI 專案已經成功建立了

底下是建立好的整個專案結構



首先打開 [MauiProgram.cs] 這個檔案，將會看到底下的內容

```
namespace MyFirstPrismMaui;

public static class MauiProgram
{
    public static MauiApp CreateMauiApp()
    {
        var builder = MauiApp.CreateBuilder();
        builder
            .UsePrismApp<App>(PrismStartup.Configure)
            .ConfigureFonts(fonts =>
            {
                fonts.AddFont("OpenSans-Regular.ttf", "OpenSansRegular");
                fonts.AddFont("OpenSans-Semibold.ttf", "OpenSansSemibold");
            });

        return builder.Build();
    }
}
```

在這裡所看到的是標準 MAUI 專案的啟動內容，不過，在此使用了

UsePrismApp<App>(PrismStartup.Configure)

敘述，指定要使用 Prism 作為這個專案的開發框架

因此，在這個專案根目錄下，找到並且打開 [PrismStartup.cs] 這個檔案，底下是這個檔案的內容

```
using MyFirstPrismMaui.Views;

namespace MyFirstPrismMaui;

internal static class PrismStartup
{
    public static void Configure(PrismAppBuilder builder)
    {
        builder.RegisterTypes(RegisterTypes)
                .OnAppStart("NavigationPage/MainPage");
    }

    private static void RegisterTypes(IContainerRegistry containerRegistry)
    {
        containerRegistry.RegisterForNavigation<MainPage>()
                    .RegisterInstance(SemanticScreenReader.Default);
    }
}
```

在這裡將會看到 Prism 在這個專案中的設定程式碼，首先，對於 DI / IoC Container 相依性注入容器，並不是使用微軟內建的原件，而是 Prism 自己的，因此，想要注入的相關服務，要在這裡來進行註冊，當然，要使用到的頁面，也需要在這裡來宣告。

打開 [App.xaml] 與 [App.xaml.cs] 檔案，看到的是很清爽的內容

```
<?xml version = "1.0" encoding = "UTF-8" ?>
<Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:local="clr-namespace:PrismApp1"
             x:Class="MyFirstPrismMaui.App">
  <Application.Resources>
    <ResourceDictionary>
      <ResourceDictionary.MergedDictionaries>
        <ResourceDictionary Source="Resources/Styles/Colors.xaml" />
        <ResourceDictionary Source="Resources/Styles/Styles.xaml" />
      </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
  </Application.Resources>
</Application>
```

```
namespace MyFirstPrismMaui;

public partial class App : Application
{
    public App()
    {
        InitializeComponent();
    }
}
```

現在來看看第一個頁面，[Views] 資料夾 > [MainPage.xaml] 檔案，請打開這個檔案，其內容如下

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             Title="{Binding Title}"
             x:Class="MyFirstPrismMaui.Views.MainPage">

  <ScrollView>
    <VerticalStackLayout
            Spacing="25"
            Padding="30,0"
            VerticalOptions="Center">

      <Image Source="prism.png"
             SemanticProperties.Description="Cute dot net bot waving hi to you!"
             HeightRequest="150"
             HorizontalOptions="Center" />

      <Label Text="Hello, World!"
             SemanticProperties.HeadingLevel="Level1"
             FontSize="32"
             HorizontalOptions="Center" />

      <Label Text="Welcome to Prism for .NET MAUI"
             SemanticProperties.HeadingLevel="Level2"
             SemanticProperties.Description="Welcome to Prism for dot net Multi plat\
form App U I"
             FontSize="18"
             HorizontalOptions="Center" />

      <Button Text="{Binding Text}"
              SemanticProperties.Hint="Counts the number of times you click"
              Command="{Binding CountCommand}"
              HorizontalOptions="Center" />

    </VerticalStackLayout>
  </ScrollView>

</ContentPage>
```

這個頁面是個相當簡單的 XAML 頁面，現在要來看看 ViewModel 的內容，是否有甚麼變化。

打開 [ViewModels] 資料夾 > [MainPageViewModel.cs] 檔案，其內容如下

```
namespace MyFirstPrismMaui.ViewModels;

public class MainPageViewModel : BindableBase
{
    private ISemanticScreenReader _screenReader { get; }
    private int _count;

    public MainPageViewModel(ISemanticScreenReader screenReader)
    {
        _screenReader = screenReader;
        CountCommand = new DelegateCommand(OnCountCommandExecuted);
    }

    public string Title => "Main Page";

    private string _text = "Click me";
    public string Text
    {
        get => _text;
        set => SetProperty(ref _text, value);
    }

    public DelegateCommand CountCommand { get; }

    private void OnCountCommandExecuted()
    {
        _count++;
        if (_count == 1)
            Text = "Clicked 1 time";
        else if (_count > 1)
            Text = $"Clicked {_count} times";

        _screenReader.Announce(Text);
    }
}
```

這裡還是與 Xamarin.Forms 時期的時候，用法都相同，在這裡繼承了 [BindableBase] 這個類別，接著就可以方便使用資料綁定的功能了

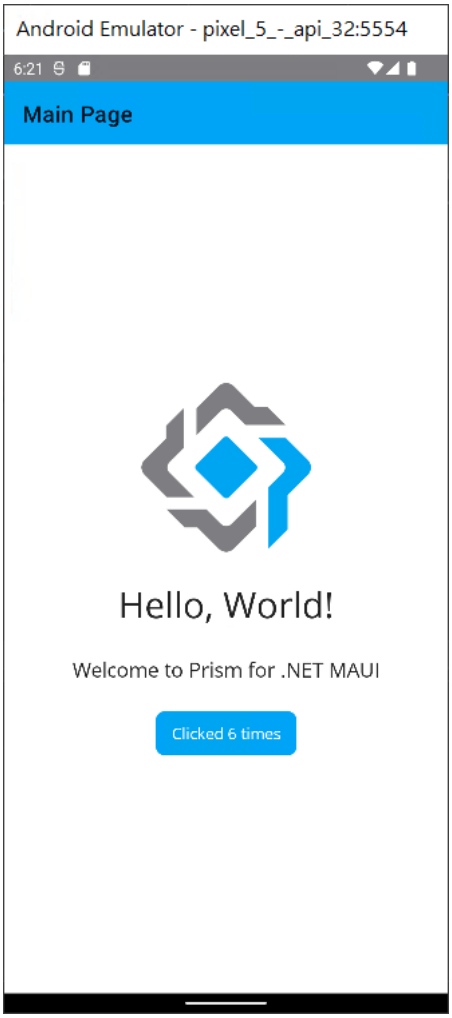
只不過，看到建構式內，注入了 [ISemanticScreenReader] 介面實作物件，說實在的，我還不知道這是甚麼服務功能，因此，我在 [OnCountCommandExecuted] 命令委派方法內，將 `_screenReader.Announce(Text);` 敘述註解起來，結果執行後，與沒有結果的似乎沒有差異，等到日後理解這個介面的用法後，會再寫篇文章來說明。

點選中間上方工具列的 [Windows Machine] 這個工具列按鈕旁的下拉選單三角形

從彈出功能表中，找到 [Android Emulators] 內的任何一個模擬器

接者，開始執行這個專案，讓他可以在 Android 模擬器出現

底下是執行後的結果


Android Emulator - pixel_5_-_api_32:5554
6:21
Main Page
Hello, World!
Welcome to Prism for .NET MAUI
Clicked 6 times

3. 版權頁

使用 Prism 進行 .NET MAUI 專案開發

檔案格式： EPUB3、PDF、MOBI

版本： 1.0
日期： 2022.08

作者： Vulcan Lee 李進興