

# Xamarin 行動開發 Android課程

Steven Chang  
講師 張朝銘



# Agenda

- Xamarin簡介
- 開發工具安裝及環境介紹
- 基礎架構及專案結構
- 基本控制項介紹
- 多頁面巡覽控制
- 使用WebService
- 使用WebAPI+Json.Net
- 基礎ListView使用
- 客製化ListView
- Sqlite資料處理
- GCM推播服務
- 程式佈署、安裝及偵錯

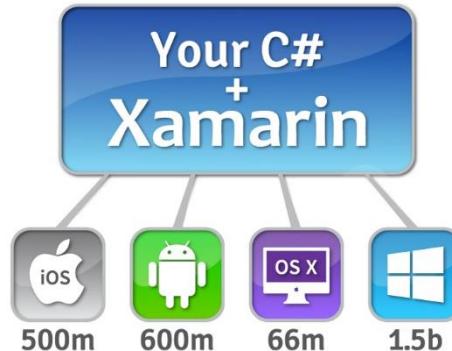


thinkpower

# 甚麼是Xamarin?

是一種使用C#語言進行開發

可以運作在30億台設備上



# Xamarin的運作方式

- 在iOS以AOT的方式編譯
- 在Android以JIT方式編譯
- 編譯後產生的文件為原生平台的ARM機器碼
- Android 和 iOS皆為1對1的方式對應至原生SDK

## Native Performance



Xamarin.iOS does full Ahead Of Time (AOT) compilation to produce an ARM binary for Apple's App Store.



Xamarin.Android takes advantage of Just In Time (JIT) compilation on the Android device.

# Xamarin的好處

- 透過良好的設計可共享約70%的程式碼
- 任何原生可做的都可用C#達成
- 即時與iOS或Android同一日推出更新
- 直接編譯為原生機器碼，效能如原生相同
- 與VisualStudio整合，讓原C#開發人員快速熟悉介面



# 如何學習？



我這兒有本秘笈，你要不要？



# 開發工具安裝及環境介紹

# Xamarin 安裝- Android

## ■ Windows

- Windows7或更新的作業系統
- Java SDK
- Android SDK
- Xamarin.Android
- Xamarin Plug-in for Visual Studio(2010以上)
- Xamarin Studio

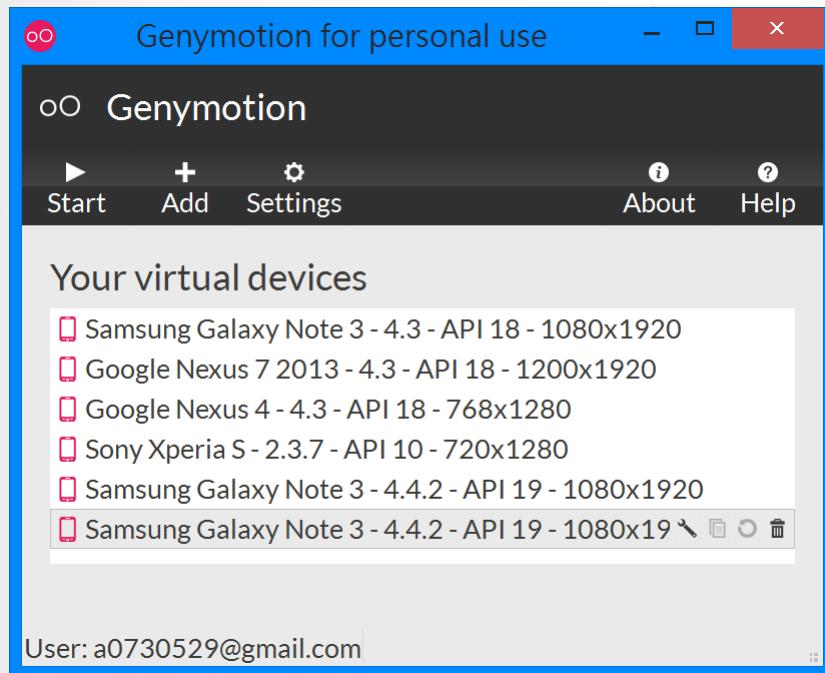
## ■ MAC

- OS X Lion 或更新的作業系統
- Java SDK
- Android SDK
- Xamarin Studio for MAC

# 使用模擬器-Genymotion

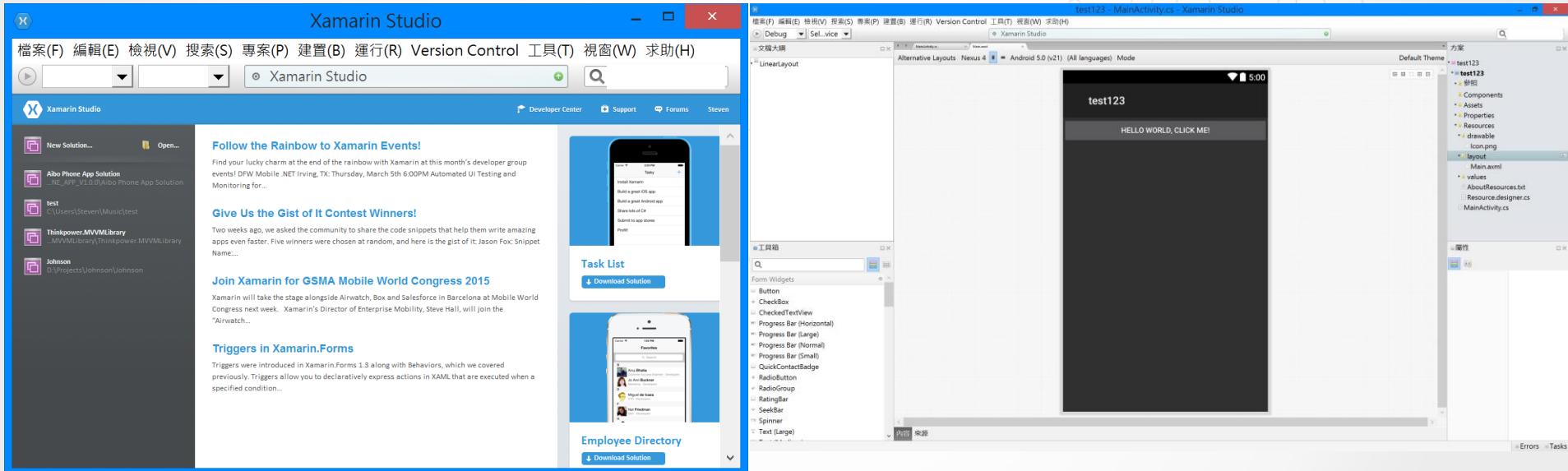
10

- 可免費使用，但須註冊帳號
- 速度比原生模擬器快，可模擬多款手機
- <https://www.genymotion.com/>



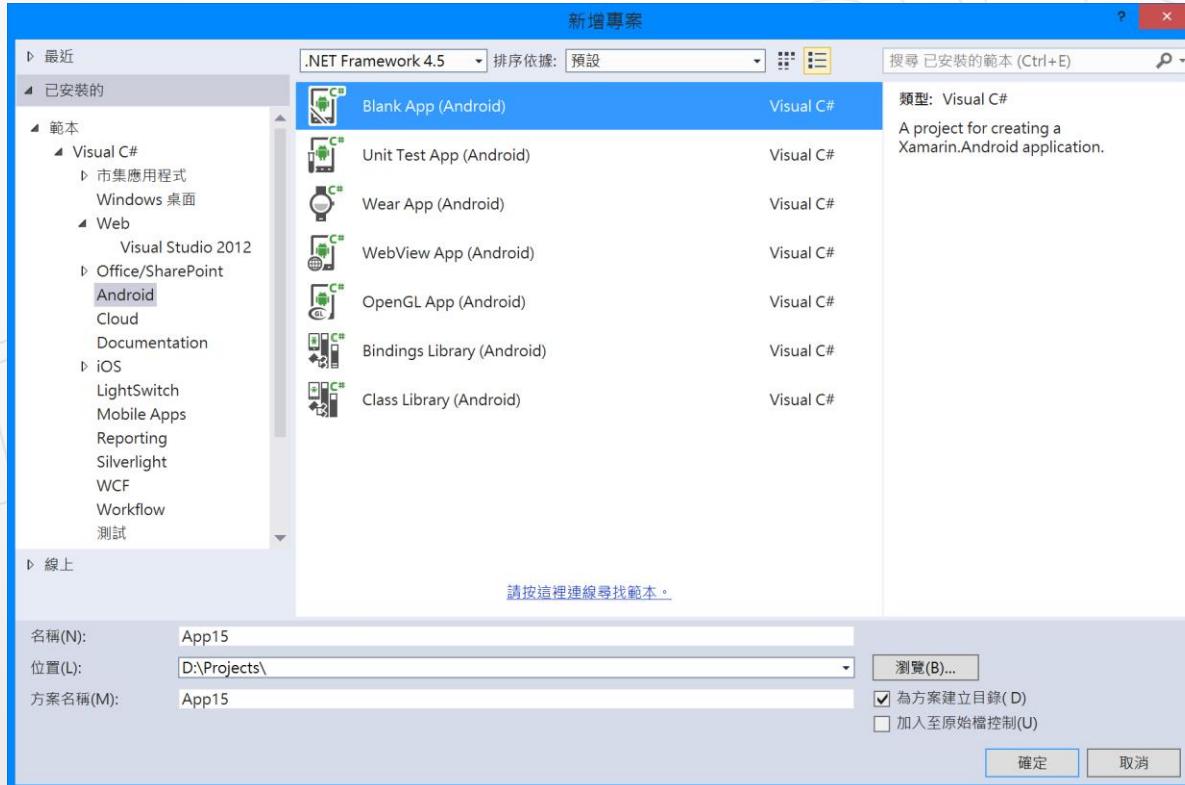
# 使用Xamarin Studio

- 不需要安裝Visual Studio即可進行開發
- 介面與Visual Studio類似，並可與Git整合



# 使用Visual Studio

- 支援2010以上(最新到2015)
- 可整合TFS、Git
- 需登入試用帳號或付費帳號才能使用



# 下載及試用

- 於Xamarin官網中可下載一鍵安裝檔  
<http://xamarin.com/platform>
- 安裝程式會自動偵測下載所有需要下載的檔案
- 若要在Visual Studio中使用則必須註冊一組Xamarin帳號並啟動試用
- Visual Studio中選擇 工具→Xamarin Account 進行登出、登入及檢視帳號資訊



# 註冊試用帳號

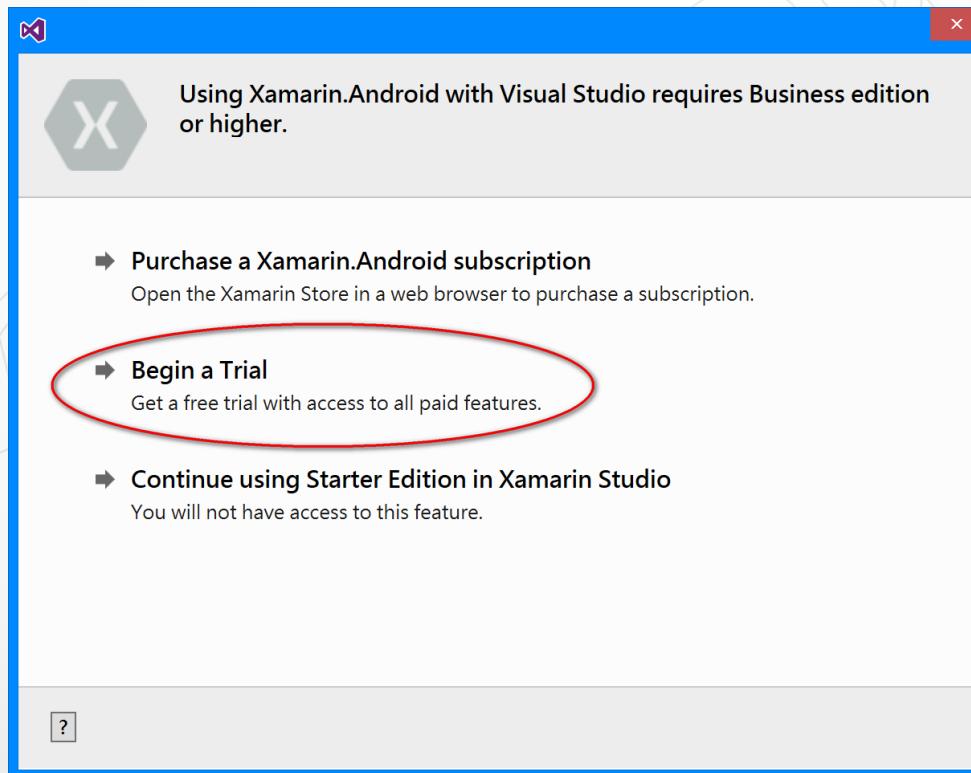
- 於以下網址註冊Xamarin帳號  
<https://store.xamarin.com/account/register>
- Email不會進行認證可隨意填寫
- 試用帳號於APP開啟時會顯示試用提示畫面
- 試用所編譯出來的APP在安裝後僅能使用24小時
- 試用帳號可使用30天



## 基礎架構及專案結構

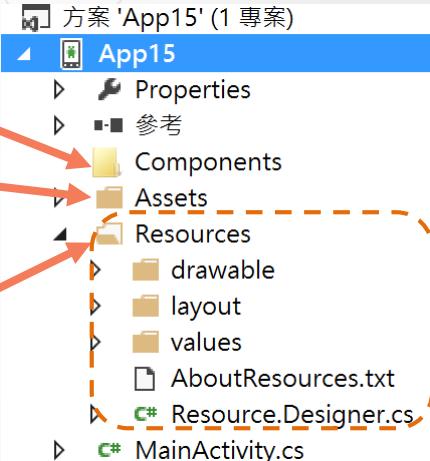
# 新增Android專案

- 開啟Visual Studio並新增專案，選擇Android→Blank App
- 開啟專案後會跳出驗證畫面，選擇啟用試用帳號

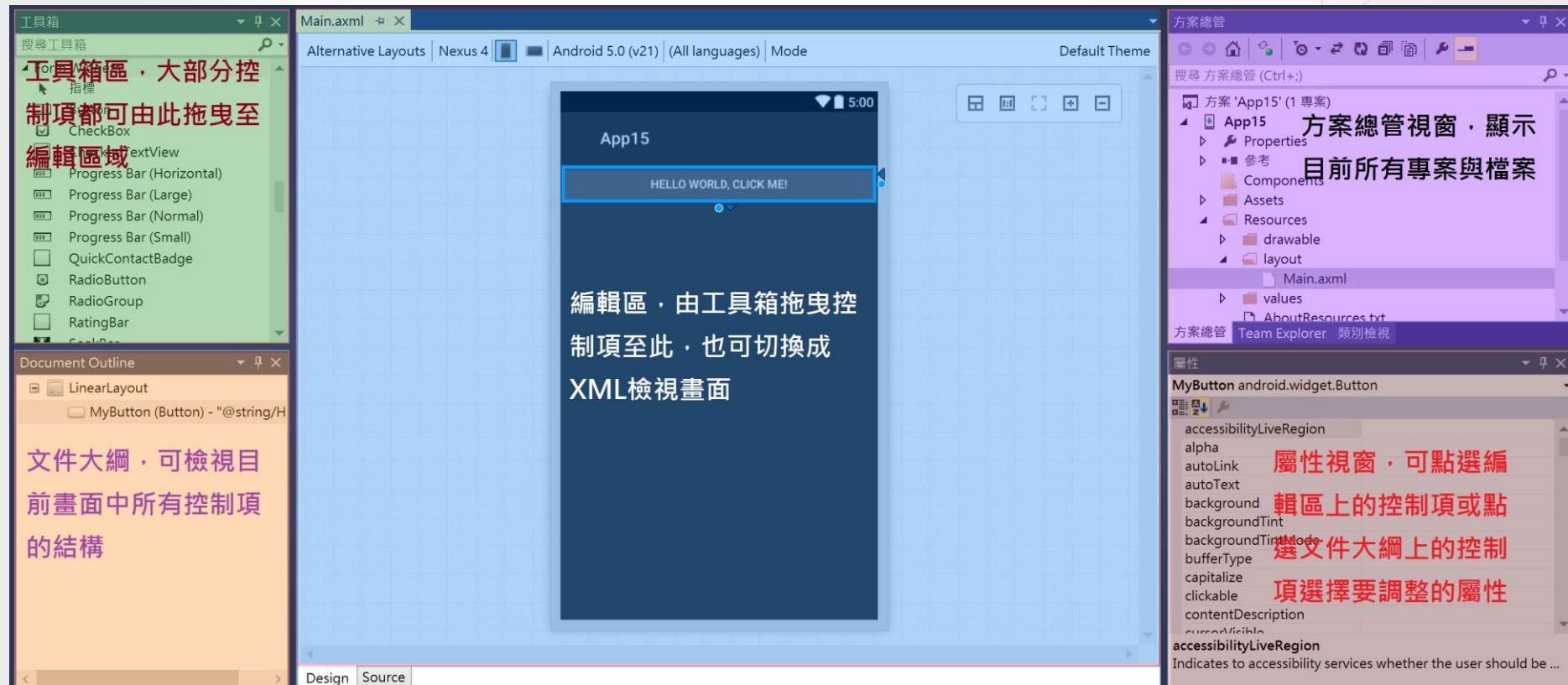


# 專案結構

- Components:  
放置下載的Xamarin元件
- Assets:  
資料內的檔案會被封裝進Android的封裝檔中，  
放置文字檔、Apk檔等應用程式所需用的檔案
- Resources:  
放置如字串資源檔(values)、圖片(drawable)、Xml  
定義的畫面(layout)，在該資料夾下的檔案都會自動  
編譯至Resource.Designer.cs中產生靜態類別Resource



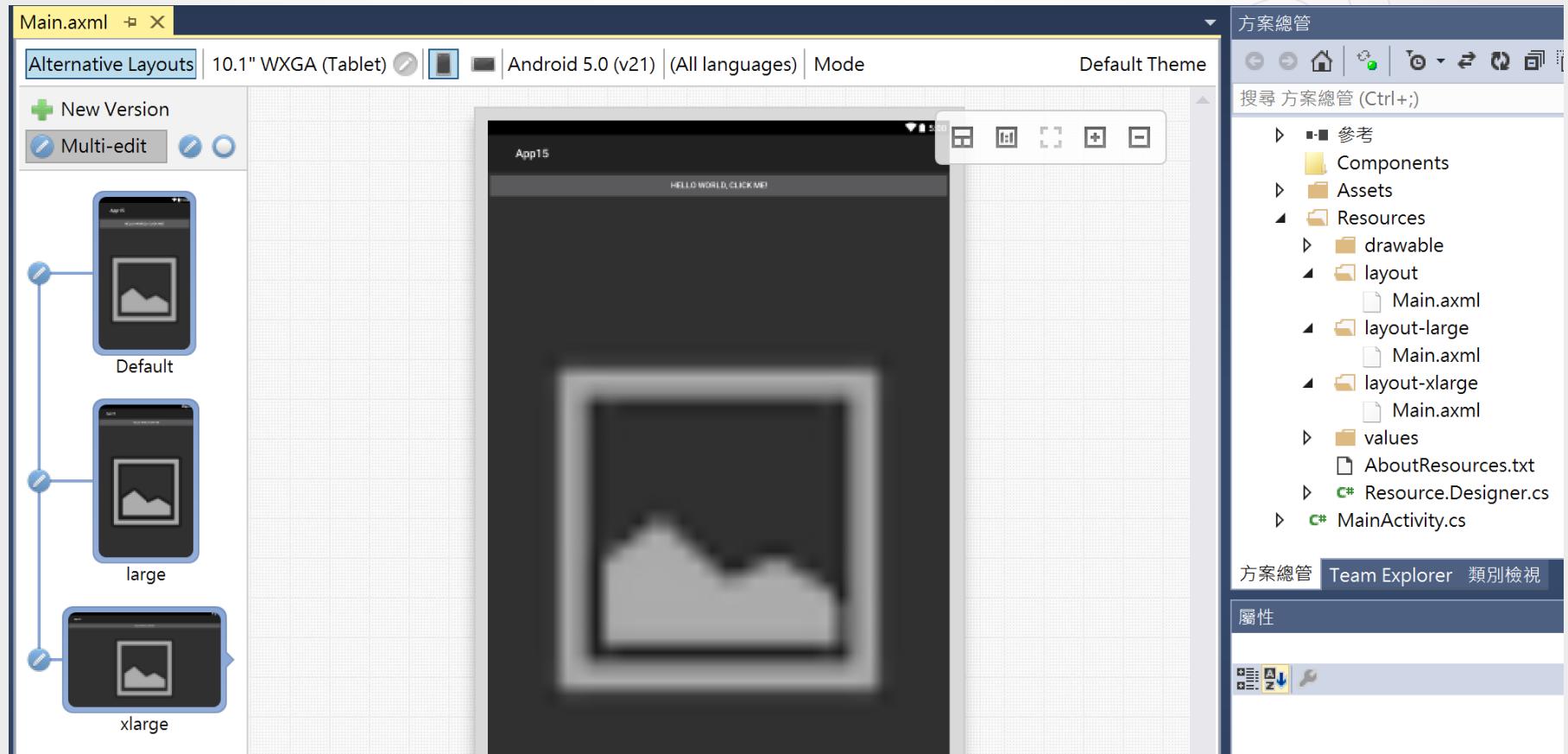
# 可視化編輯畫面



# 多畫面編輯支援

19

在編輯區內，可開啟多畫面編輯的模式，並設定一次修改多筆來調整畫面



# Android架構

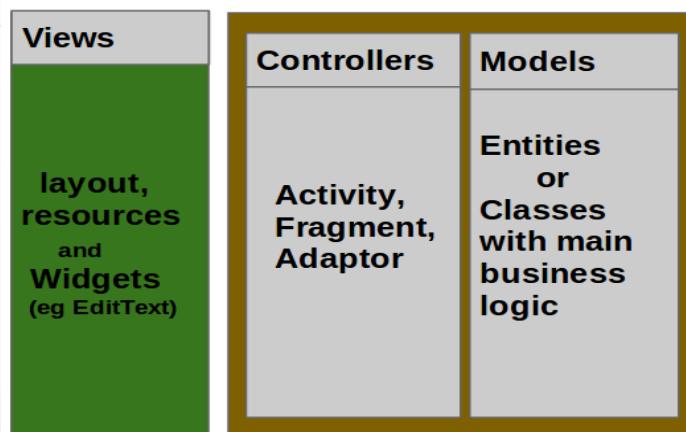
採用了MVC框架：

## ■ View

檢視層，一般在Android中使用XML文件進行描述，文件放置在Resources→layout的目錄下，在Xamarin中文件檔案為.axml。

## ■ Controller

控制器層，在Android中通常由Activity負責，每個Activity通常會指定展示某一個View，除此之外還有Adapter、Fragment等。



# 指定起始Activity

- Android程式啟動後，會尋找設定為 MainLauncher = true 的Activity 做為主要應用程式起始的Activity

```
[Activity(Label = "L01", MainLauncher = true, Icon = "@drawable/icon")]
0 個參考
```

- 在Activity的OnCreate這個方法內，使用SetContentView()方法指定要顯示的View，可由前述提過的靜態類別Resource找到layout

```
protected override void OnCreate(Bundle bundle)
{
    base.OnCreate(bundle);

    // Set our view from the "main" layout resource
    SetContentView(Resource.Layout.Main);
```

- 使用FindViewById<T>()方法，並由靜態類別Resource找到畫面上控制項的ID，就可取得該控制項的類別物件。

```
// Get our button from the layout resource,
// and attach an event to it
Button button = FindViewById<Button>(Resource.Id.MyButton);
```

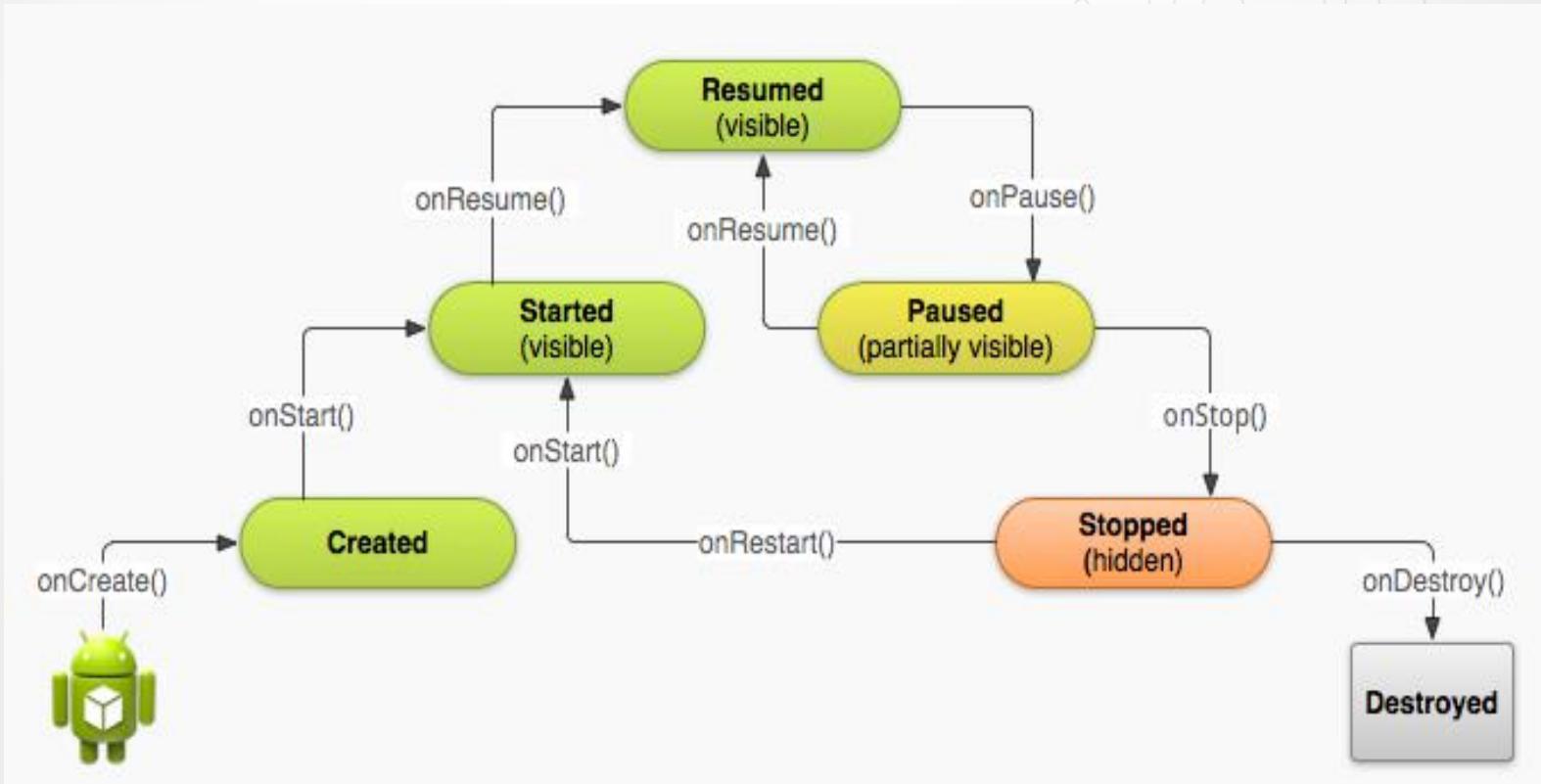
# 實作自己的Application類別

- Android最初的起始類別為Application，在應用程式啟動時或所有的Activity都結束時它都還會存活著。
- 複寫Application類別可以讓每個Activity共用Application中建立的屬性、方法甚至事件。
- 複寫Application類別時需時做對應的建構子，並複寫OnCreate方法。

```
[Application]  
3 個參考  
public class MyApp:Application  
{  
    2 個參考  
    public string Value { get; set; }  
  
    0 個參考  
    public MyApp(IntPtr javaReference, JniHandleOwnership transfer)  
        : base(javaReference, transfer)  
    {}  
    1 個參考  
    public override void OnCreate()  
    {  
        Value = "i am from Application's Value";  
        base.OnCreate();  
    }  
}
```

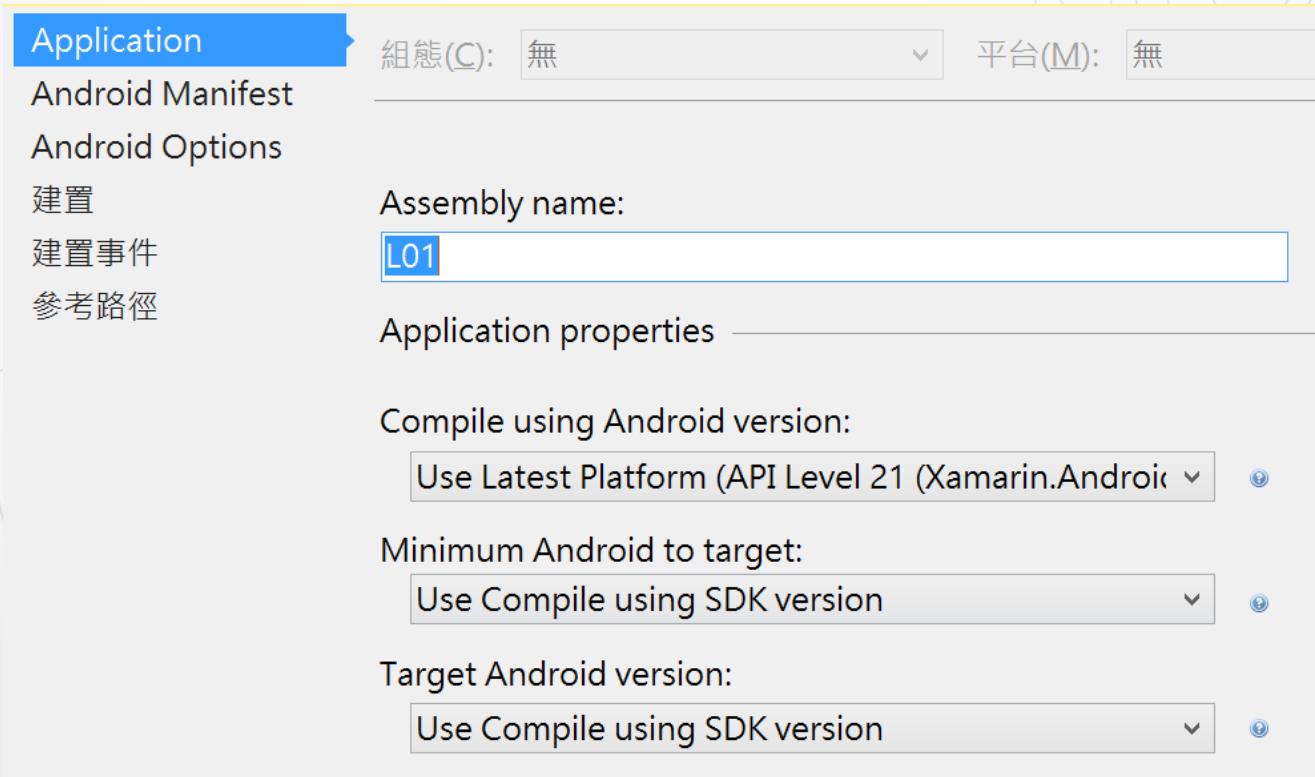
# 生命週期

如同ASP.NET網頁一般有Page\_Load等生命週期，Activity也會有自己的生命週期。



# 設定版本

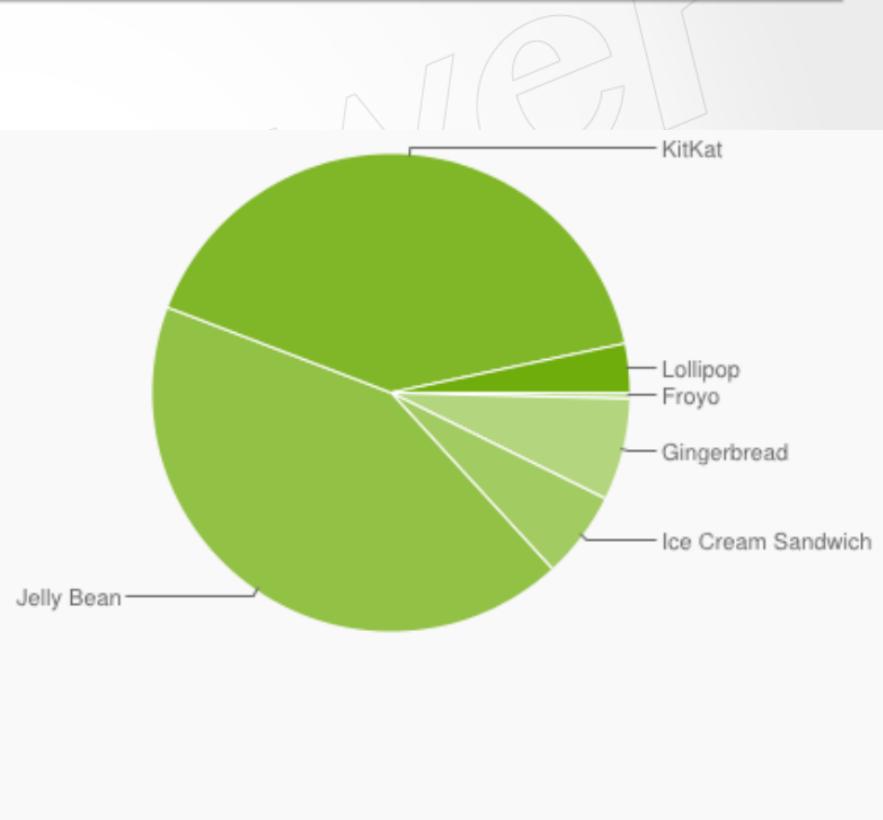
點選專案並開啟專案屬性，在Application頁籤內可選擇Android的編譯版本，依照目標裝置的版本，想要支援的最低版本等來進行選擇。



# Android Versions

<http://developer.android.com/about/dashboards/index.html>

Version	Codename	API	Distribution
2.2	Froyo	8	0.4%
2.3.3 - 2.3.7	Gingerbread	10	6.9%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	5.9%
4.1.x	Jelly Bean	16	17.3%
4.2.x		17	19.4%
4.3		18	5.9%
4.4	KitKat	19	40.9%
5.0	Lollipop	21	3.3%





# 基本控制項介紹

# XML檔案結構

- @+id/[id名稱]:告訴Android parser, 為物件建立一個resource id
- layout\_width、layout\_height:指定控制項的大小，fill\_parent表示充滿外部容器，wrap\_content表示符合控制項內容，也可用dp定義大小
- text:@string/Hello:Button的text亦即顯示文字由字串資源檔內的Hello讀取
- LinearLayout:畫面控制項位置依據由此Layout來決定。

The screenshot shows the code editor for an Android XML layout file named Main.axml. The code is as follows:

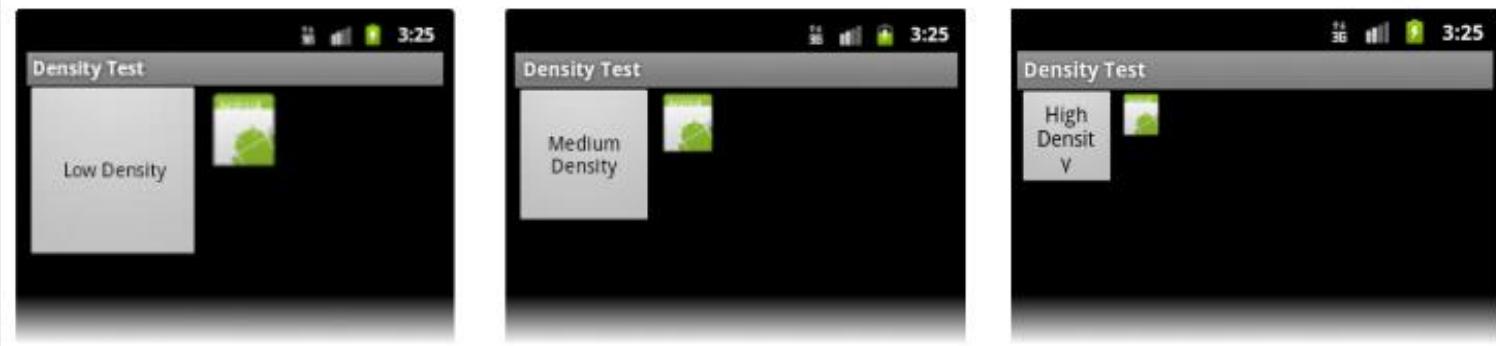
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
        android:id="@+id/MyButton"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/Hello" />
</LinearLayout>
```

The code is highlighted with syntax coloring. The first five lines of the LinearLayout block and the entire Button block are enclosed in a blue rounded rectangle, likely indicating a selected or copied section of the code.

# 長度單位-dp

Android的控制項若需指定特定長度時，建議使用的單位為dp(device independent pixels-設備獨立象素，或dip)，表示使用此單位時不會因為裝置的DPI值(銀幕像素密度)而影響到。

使用px為單位時：



使用dp為單位時：



# LinerLayout

LinerLayout為堆疊式的控制項排序，可水平或者垂直放置控制項

orientation="vertical"



orientation="horizontal"



# RelativeLayout

RelativeLayout為指定相對位置的方式，控制項可指定相對於RelativeLayout或相對於Layout內的其他控制項的哪個位置。

`layout_alignParentBottom`



`layout_above= "@id/控制項A"`



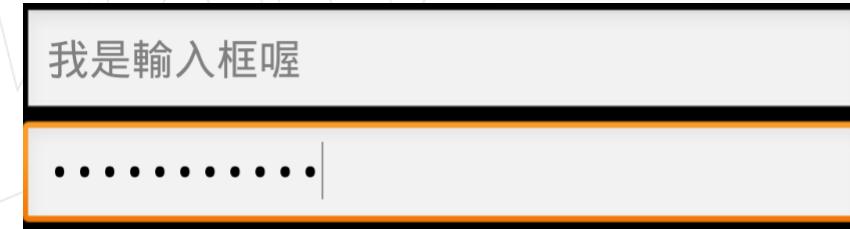
# 基本控制項

## ■ TextView 我的電話是0912345678

- 自動連結判斷:autolink
- 文字大小、顏色:textsize, textcolor
- 文字:text

## ■ EditText

- 提示:hint
- 輸入類型:inputType



## ■ Button

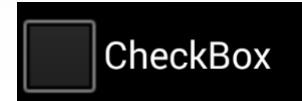
- 點擊事件:Click



# 基本控制項

## ■ CheckBox

- 是否勾選:Checked
- 事件:CheckedChange



## ■ RadioGroup&RadioButton

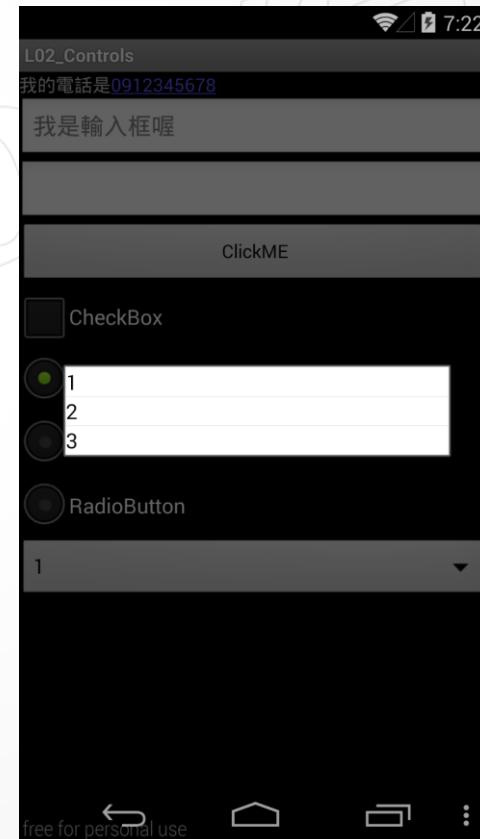
- RadioButton與CheckBox皆繼承CompoundButton，同樣有Checked屬性與CheckedChange事件
- RadioGroup繼承了LinearLayout，多個RadioButton需放在同個RadioGroup中才會有單選效果



# 基本控制項

## ■ Spinner

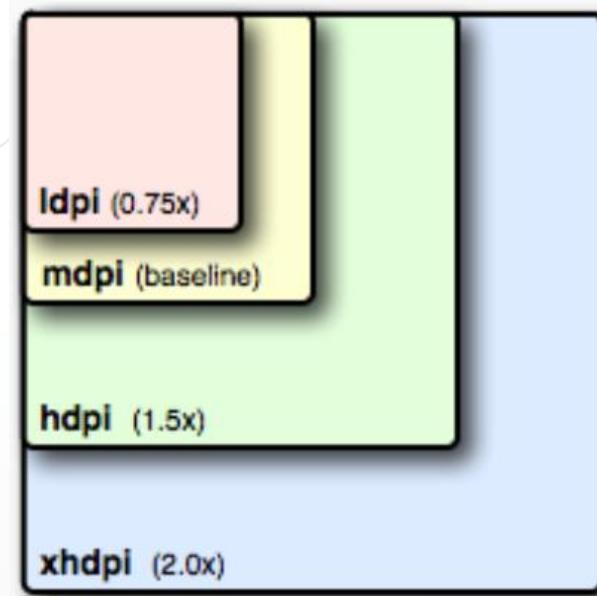
- 需設定屬性Adapter給予資料來源



# 基本控制項

## ■ ImageView

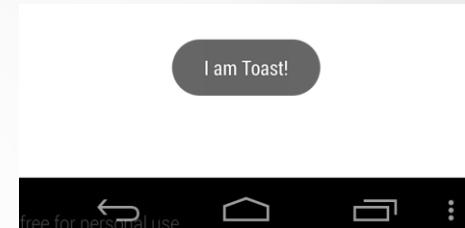
- 圖片存放在**drawable**中，可依照不同解析度銀幕給予不同的圖片
- 如沒有給予不同解析度大小對應的圖片時，系統會自動放大縮小圖片



# 基本控制項

## ■ Toast

- 可設定出現的時間(2000ms or 3500ms)



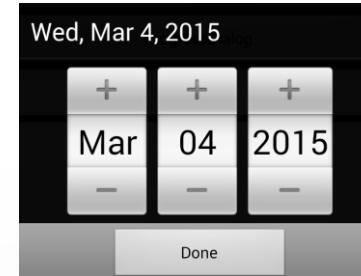
## ■ AlertDialog

- 最多可三個按鈕



## ■ DatePicker

- 顯示格式因系統語系而變化



## ■ ProgressDialog

- 可控制是否可取消畫面



# 基本控制項

## ■ AutoCompleteText

- 需設定屬性Adapter給予資料
- 提示字數:Threshold

I love Xamarin!

I don't understand

i

## ■ WebView

- 載入網址:LoadUrl
- 開啟Javascript:  
Settings.JavaScriptEnabled



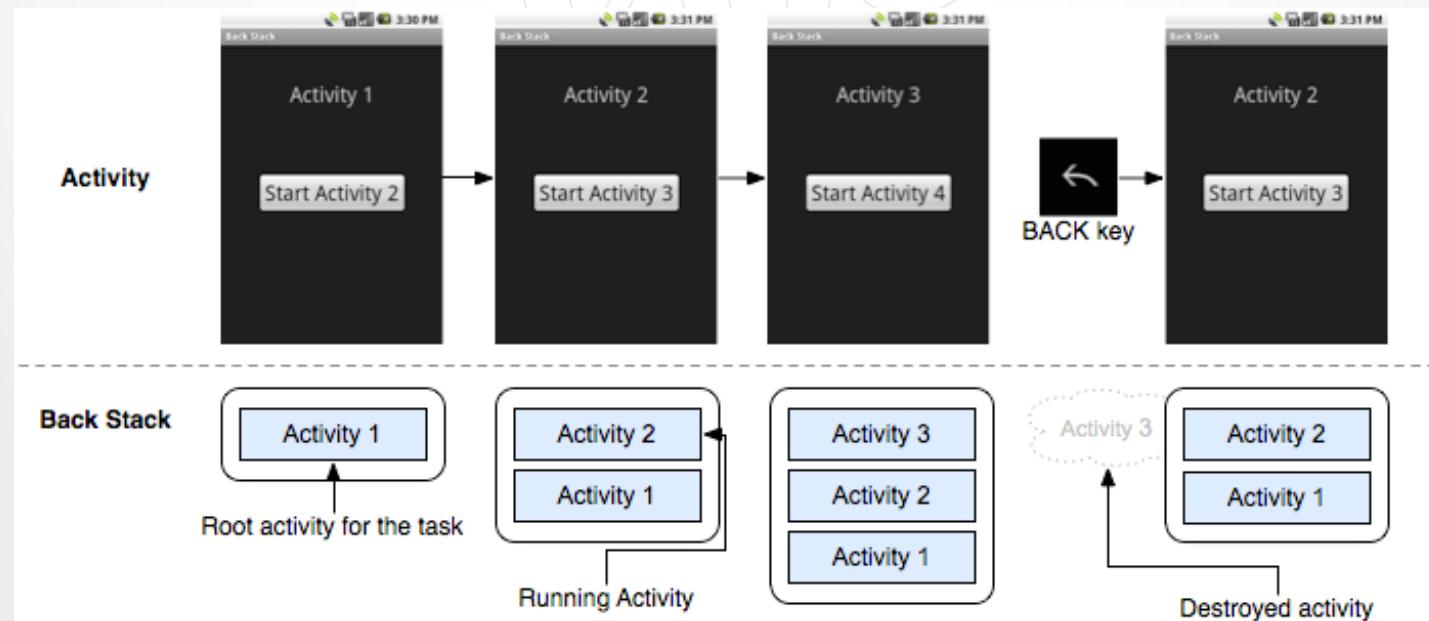


## 多頁面巡覽控制

# Activity切換運作

多頁面巡覽其實就是切換不同的Activity，並呈現該Activity所指定的View。當Activity A切換到Activity B時，系統會將Activity A放到Activity Stack(寧列)中，你可以想像成是B的畫面蓋在了A的上面。

當使用者按下手機上的Back鍵時，B會被關掉銷毀，因此畫面又會回到了A。



# 啟動另一個Activity

- 使用StartActivity(type)方法
  - 傳入下一個Activity的Type
- 建立Intent物件，同樣使用StartActivity(Intent)方法
  - Intent即為意圖的意思，代表告知應用程式接下來要執行的任務，在此我們為Intent設定了想要開啟的Activity。
  - Intent物件可以放入想傳遞到下一個Activity的參數，由下個Activity的Intent內取回參數。
  - 使用Intent時我們可以對目標的Activity標示Flag，例如目標的Activity不放入佇列中，或者從佇列中開啟已存在的Activity等。

# Intent的Activity Flag

Activity Flag有非常非常多種，常用的如下列所述：

- **ActivityFlags.ClearTop**

當目標Activity存在於Stack中，啟動該Activity並將其之上的Activity結束

- **ActivityFlags.NoHistory**

啟動目標的Activity不會被放置在Stack內

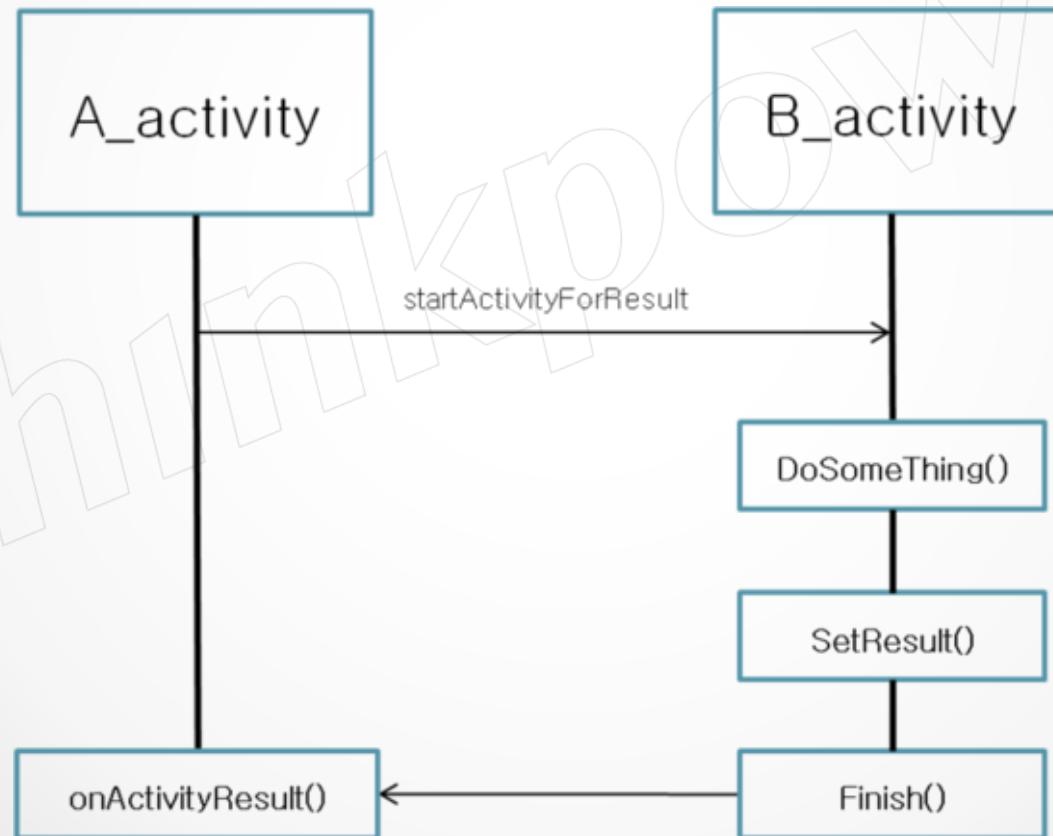
- **ActivityFlags.SingleTop**

當目標Activity已在Stack的TOP時，不會再被重新建立。

# 回傳結果

41

除了前述的參數往下一個Activity傳遞之外，也能將其結果回傳到上一個Activity中。



# 回傳結果

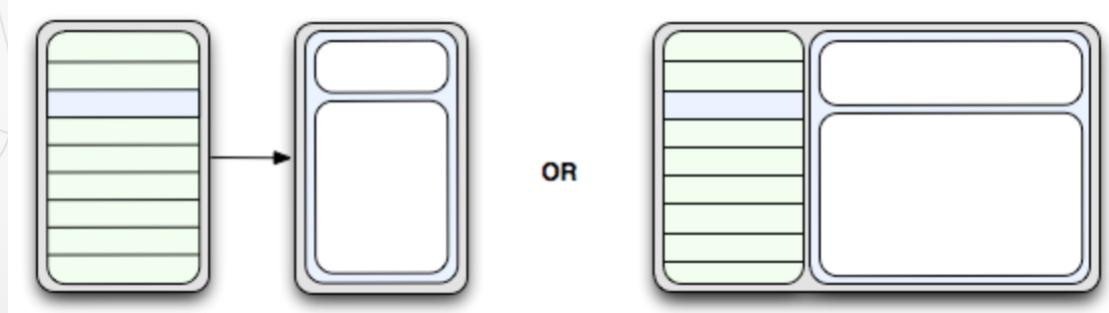
42

- 在A中使用`StartActivityForResult()`方法啟動下一個Activity B。
- 在B中建立Intent物件，將回傳的值放入Intent內。
- 使用`SetResult()`方法將Intent物件放入，並關閉Activity B。
- 於A中複寫`OnActivityResult`方法，即可取得傳回來的Intent。
- `StartActivityForResult()`可自訂`RequestCode`，用於在結果回傳時做辨識。
- `SetResult()`可指定回傳的狀態是取消或OK，再由A來判定。

# 使用Fragment

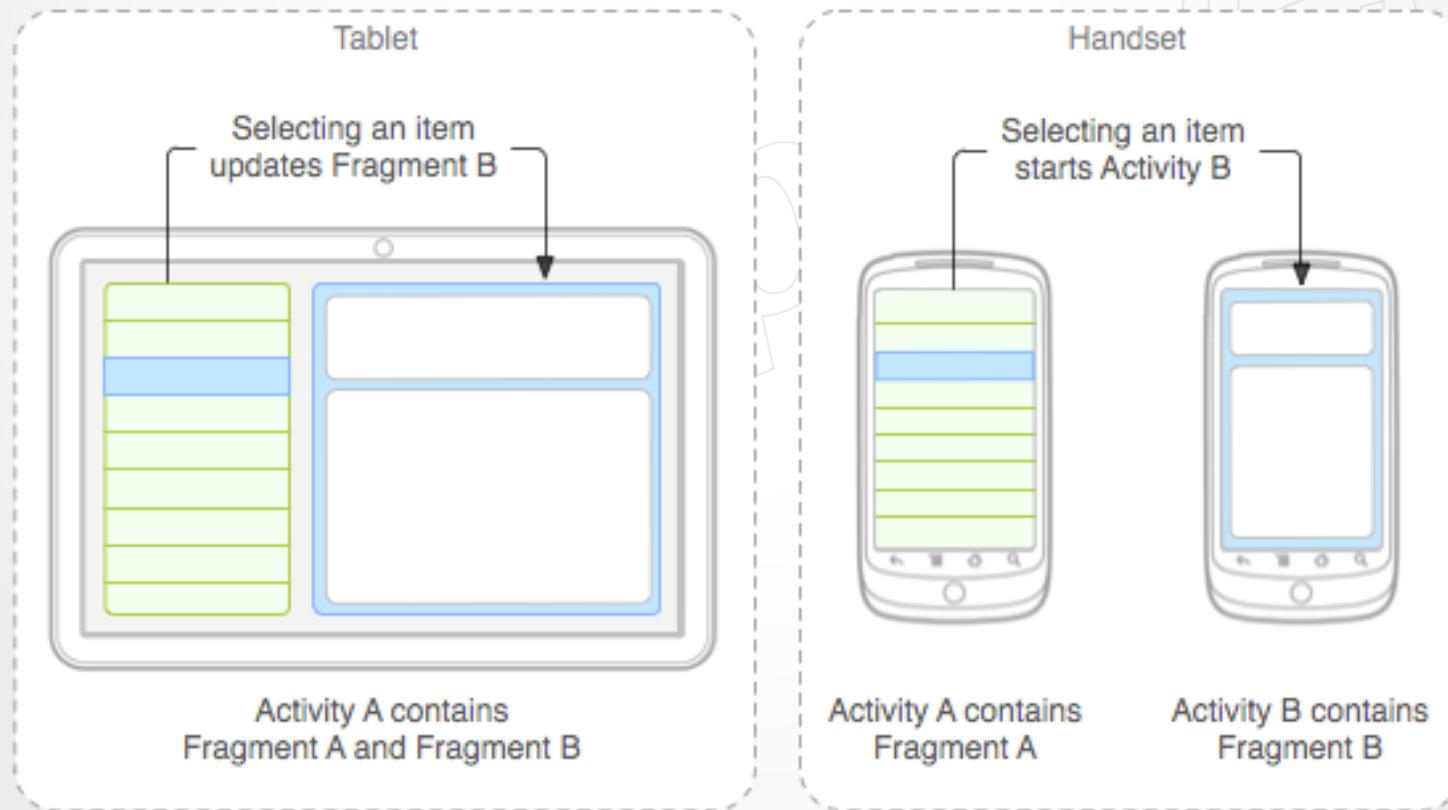
至此我們所使用的都是Activity，當我們遇到下面所示的畫面，左邊為手機，右邊為平板畫面，使用Activity則必須要做三種不同的畫面，而且在平板中每當切換右方內容時又必須再進到下一個Activity。

在Android3.0後，出現了Fargment，你可以把原本切換Activity的方式當作是網頁整個PostBack回Service再重新Response回來，而使用Fragment就像在使用Ajax一樣，只需更換畫面上個某一區塊即可。



# 使用Fragment

Fragment是依存在Activity下的，一定要有Activity才會有Fragment，使用了Fragment後可以更靈活的使用View。



# 使用Fragment

使用Fragment必須要在Android3.0(API 11)以上的系統，但若是希望在此之前的系統也能使用Fargment，就必須在專案安裝Support Library V4套件，可從Nuget或Xamarin的Components Store中抓取。

The screenshot shows the Xamarin Component Store interface. At the top, there are navigation links for 'Xamarin Component Store' and 'Android Support Library v4'. On the right, there are buttons for 'Suggest a Component' and 'Submit a Component'. Below the header, the main content area displays the following information:

- Android Support Library v4** (version 21.0.3.0)
- Xamarin Inc.**
- Provide backward-compatible versions of Android framework APIs.**
- 13 ratings** (4 stars)
- Compatible with** (Android icon)
- Add to App** button

On the left side, there are sidebar filters for Publisher (Xamarin Inc.), Category (Libraries), Price (Free), and Versions (dropdown set to 21.0.3.0). Below these filters, it shows 'Published On' (December 16, 2014) and 'Tags' (android).

The main content area also contains a detailed description of the library and its benefits.

*Portions of this page are modifications based on work created and shared by the Android Open Source Project and used according to terms described in the Creative Commons 2.5 Attribution License.*

# 動態使用Fragment

- 使用FragmentManager物件呼叫BeginTransaction()方法
- 使用Replace()、Add()或Attach()方法將目標Fragment物件傳入，並指定要替換掉的區塊(通常會是FrameLayout，以圖層堆疊概念的Layout)
- 可使用AddToBackStack()方法將Fragment加入堆疊佇列中，這裡的佇列跟Activity Stack無關
- 使用SetTransition()方法可以設定Fragment切換時要展示的動畫效果
- 最後要使用Commit()方法才會進行Fragment的替換動作

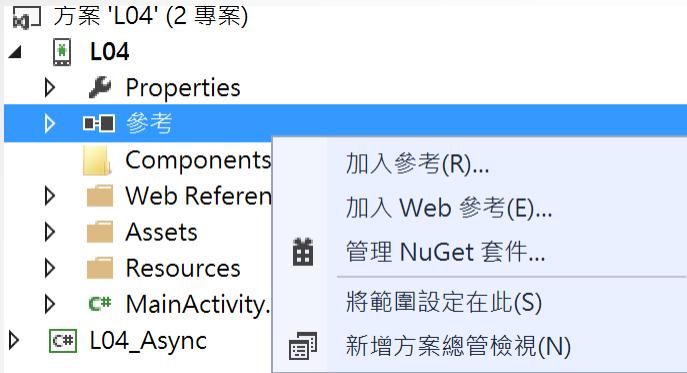
```
FragmentManager.BeginTransaction().AddToBackStack(null).SetTransition(FragmentTransit.FragmentOpen)
    .Replace(Resource.Id.frameLayout, new SecondFragment()).Commit();
```



# 使用WebService

# 加入WebService服務參考

## ■ 於專案的參考按下滑鼠右鍵，選擇加入Web參考



## ■ 輸入WebService位置後加入參考



# 呼叫WebService提供的方法

## ■ 建立由工具幫我們所新增的Proxy 類別

```
◀ { } L05_WebService.net.azurewebsites.testmyws
  ▷ BasicHttpBinding_IService1
  ▷ BasicHttpsBinding_IService1
  ▷ GetDataCompletedEventArgs
  ▷ GetDataCompletedEventHandler
```



```
//建立WebService Proxy類別
BasicHttpBinding_IService1 service = new BasicHttpBinding_IService1();
```

## ■ 呼叫WebService提供的方法

```
:service.GetData(123, true);
```

至此步驟已完成了WebService呼叫，但若該方法會花費很久的時間才 Response結果回來，會導致APP看起來像是當掉一樣停住，因此要使用由Proxy類別自動產生的非同步方法

# 使用非同步方式(EAP)

- 使用相同的方法名稱，但後方加了Async後綴詞

```
service.GetDataAsync(123, true);
```

- 該方法會再對應到一個相同方法名但後綴詞為Completed的事件，表示該Async方法執行完成後會觸發的事件。

```
service.GetDataCompleted += (obj, arg) =>
{
};

};
```

使用此方式可讓APP不會有停頓的效果，因為它不是執行在UI執行緒上，此種由一個[MethodName]Async加上完成事件[MethodName]Completed的非同步架構又叫做事件架構非同步模式(簡稱EAP)，但呼叫的WebService方法如果很多，就會變得很亂，因要註冊許多完成事件

# 使用非同步方式(TAP)

在.NET 4.5新增了async與await關鍵字，搭配Task Parallel Library 使用，可以快速開發非同步程式。此種方式又稱為以工作為基礎的非同步模式(TAP)，在Xamarin行動開發上非常廣泛使用。

在EAP的模式中，建立一個TaskCompletionSource<T>的類別，利用此類別產生Task<T>回傳結果並將原本的程式包裝起來。

```
private Task<string> GetData(int value)
{
    TaskCompletionSource<string> task = new TaskCompletionSource<string>();
    service.GetDataCompleted += (sender, e) =>
    {
        task.SetResult(e.Result);
    };
    service.GetDataAsync(value, true);
    return task.Task;
}
```

button.Click += async (sender, e) =>



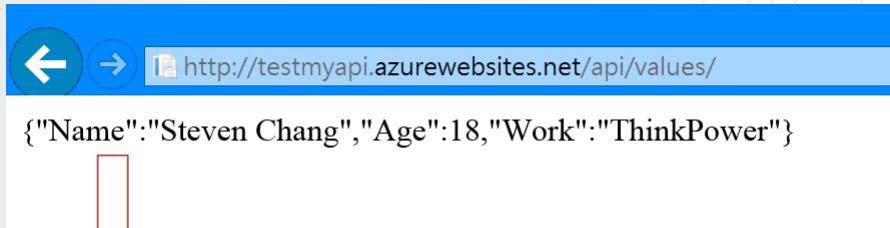
var result = await this.GetData(999);



# 使用WebAPI+Json.Net

# 呼叫WebAPI取得Json字串

- 建立出 WebClient 類別
- 使用 DownloadStringTaskAsync() 非同步方法取得 Json 字串



```
WebClient request = new WebClient();
var result = await request.DownloadStringTaskAsync(new Uri("http://testmyapi.azurewebsites.net/api/values/"));
```

# 將Json字符串轉換成物件

- 手動建立Class，屬性名稱就是Json的欄位名稱
- 使用線上工具建立Class-<http://json2csharp.com/>

json2csharp  
generate c# classes from json

```
{"Name": "Steven Chang", "Age": 18, "Work": "ThinkPower"}|
```

Generate

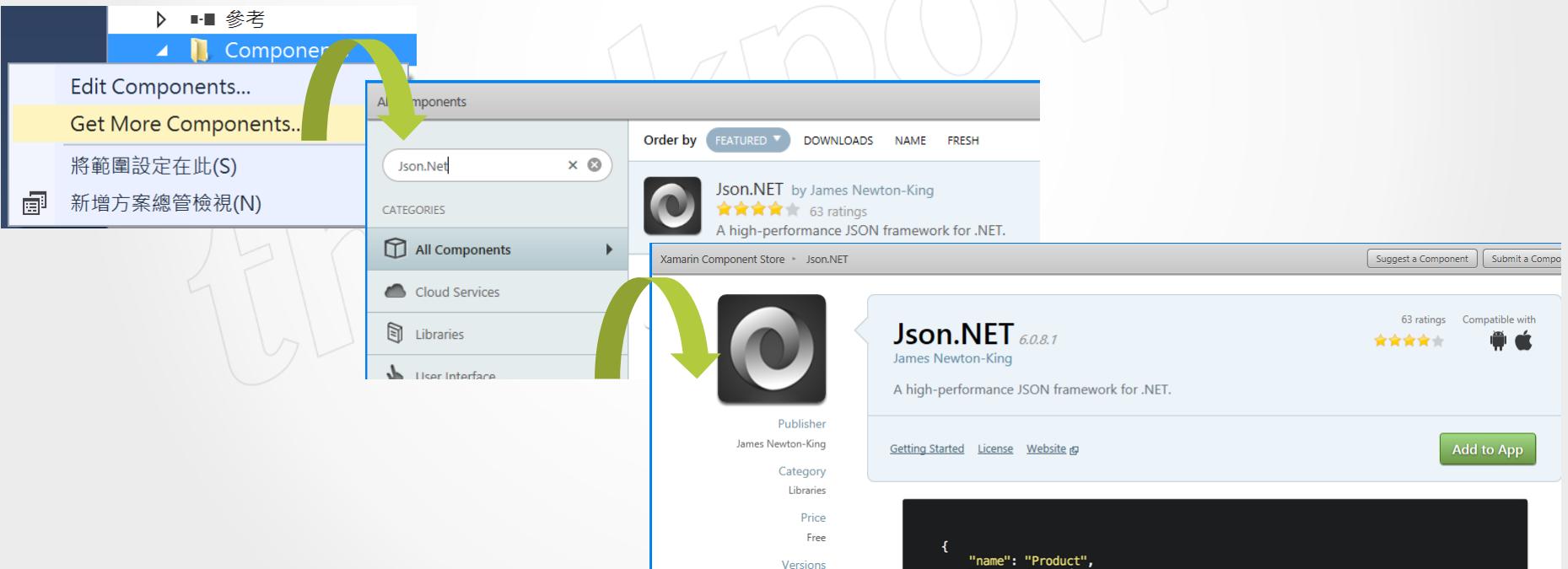


```
public class RootObject
{
    public string Name { get; set; }
    public int Age { get; set; }
    public string Work { get; set; }
}
```

# 使用Json.NET

## ■ 於Xamarin Components Store或者NuGet下載Json.Net

- 於專案中的Components 點滑鼠右鍵選擇GetMore Components
- 搜尋Json.Net點選並按下Add To App



# 使用Json.NET

56

Json.NET可快速將物件轉換成Json字串，也可將Json字串反序列化回物件。

- 使用`JsonConvert`類別下的靜態方法`DeserializeObject<T>()`可將Json反序列化為物件
- 反之使用`JsonConvert`類別下的`SerializeObject()`則將物件序列化成Json

```
result "{\"Name\":\"Steven Chang\",\"Age\":18,\"Work\":\"ThinkPower\"}"
```

```
var obj = Newtonsoft.Json.JsonConvert.DeserializeObject<MyClass>(result);
```

監看式 1	
名稱	值
obj	{L05.MyClass}
Age	18
Name	"Steven Chang"
Work	"ThinkPower"



# 基礎 ListView 使用

# 快速使用簡單的ListView

ListView為APP上非常常用到的一個元件，用於展示多筆列表資料。

- 準備要展現在ListView上的資料
- 使用ArrayAdapter，它有點概念類似於ListView的控制器(Controller)，ListView則為View
- ListView中的每一列(Row)都是一個View，因此要為這個View指定一個Layout，Android有提供幾種預設的Layout  
[http://developer.xamarin.com/guides/android/user\\_interface/working\\_with\\_listviews\\_and\\_adapters/part\\_3 - customizing\\_a\\_listview's\\_appearance/](http://developer.xamarin.com/guides/android/user_interface/working_with_listviews_and_adapters/part_3 - customizing_a_listview's_appearance/)
- 將ArratAdapter指定給ListView上的Adapter屬性。

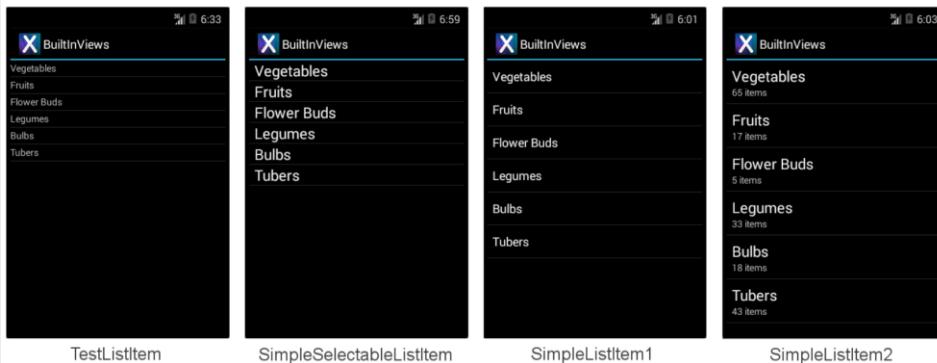
# 快速使用簡單的ListView

## ■ 準備資料

```
string[] items;
using (var stream = Assets.Open("data.txt"))
{
    using (StreamReader reader = new StreamReader(stream))
    {
        items = reader.ReadToEnd().Split(',');
    }
}
```

L06  
data.txt  
獅子,老虎,長頸鹿,草尼瑪(?)

## ■ 建立ArrayAdapter，放入資料並指定Row的layout



## ■ 指定Adapter

```
listView.Adapter = adapter;
```

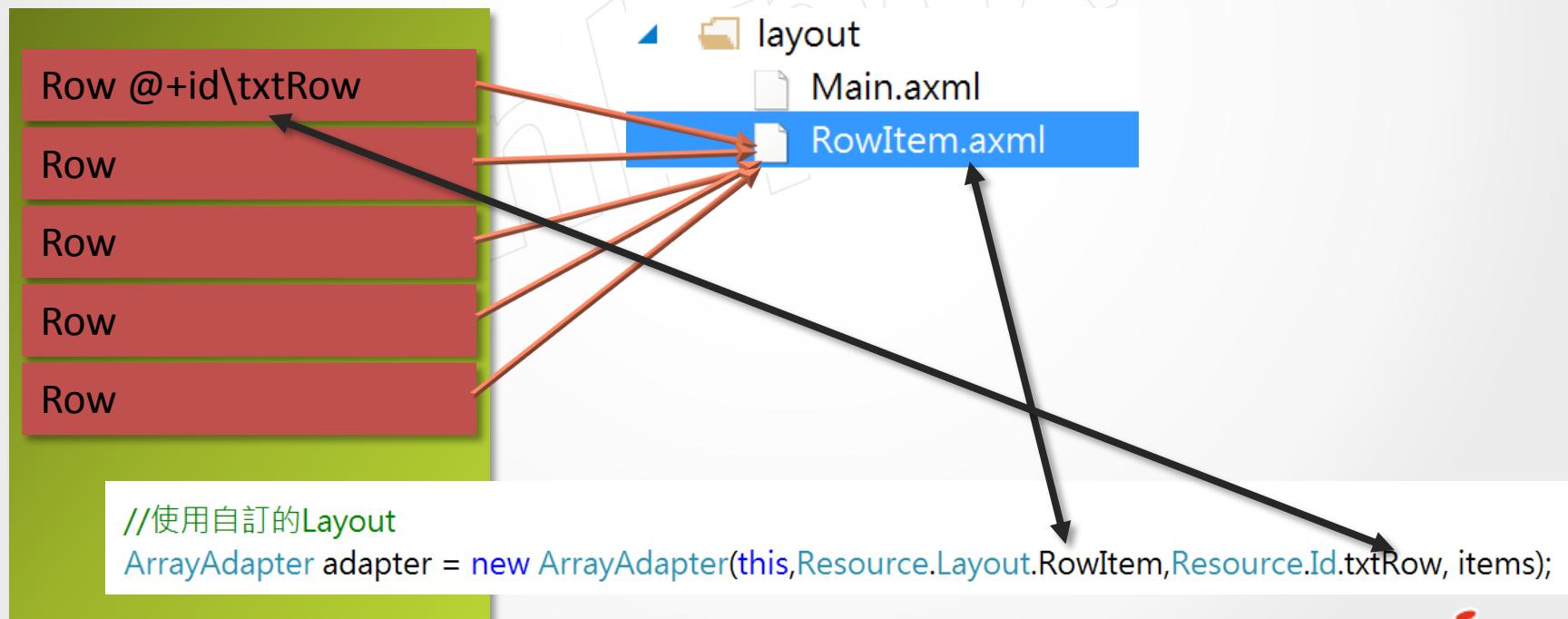
//建立Adapter並給予資料來源，以及指定Layout

```
ArrayAdapter adapter = new ArrayAdapter(this, Android.Resource.Layout.SimpleListItem1, items);
```

# 更換成自己的Layout

60

使用ArrayAdapter時除了Android提供預設的Layout，也可更換成自己所設計的Layout。在ArrayAdapter中指定自己建立的Layout與文字控制項的ID即可。



# ListView點擊事件

- ListView提供了ItemClick的事件，註冊此事件後當任一個Row被點下時即會觸發。
- 事件參數中會提供Row的索引值，透過它便可找回對應的資料。

```
//註冊click事件
listView.ItemClick += (sender, e) =>
{
    int index = e.Position;//點擊的索引位置
    Toast.MakeText(this, items[index], ToastLength.Short).Show();
};
```



# 客製化ListView

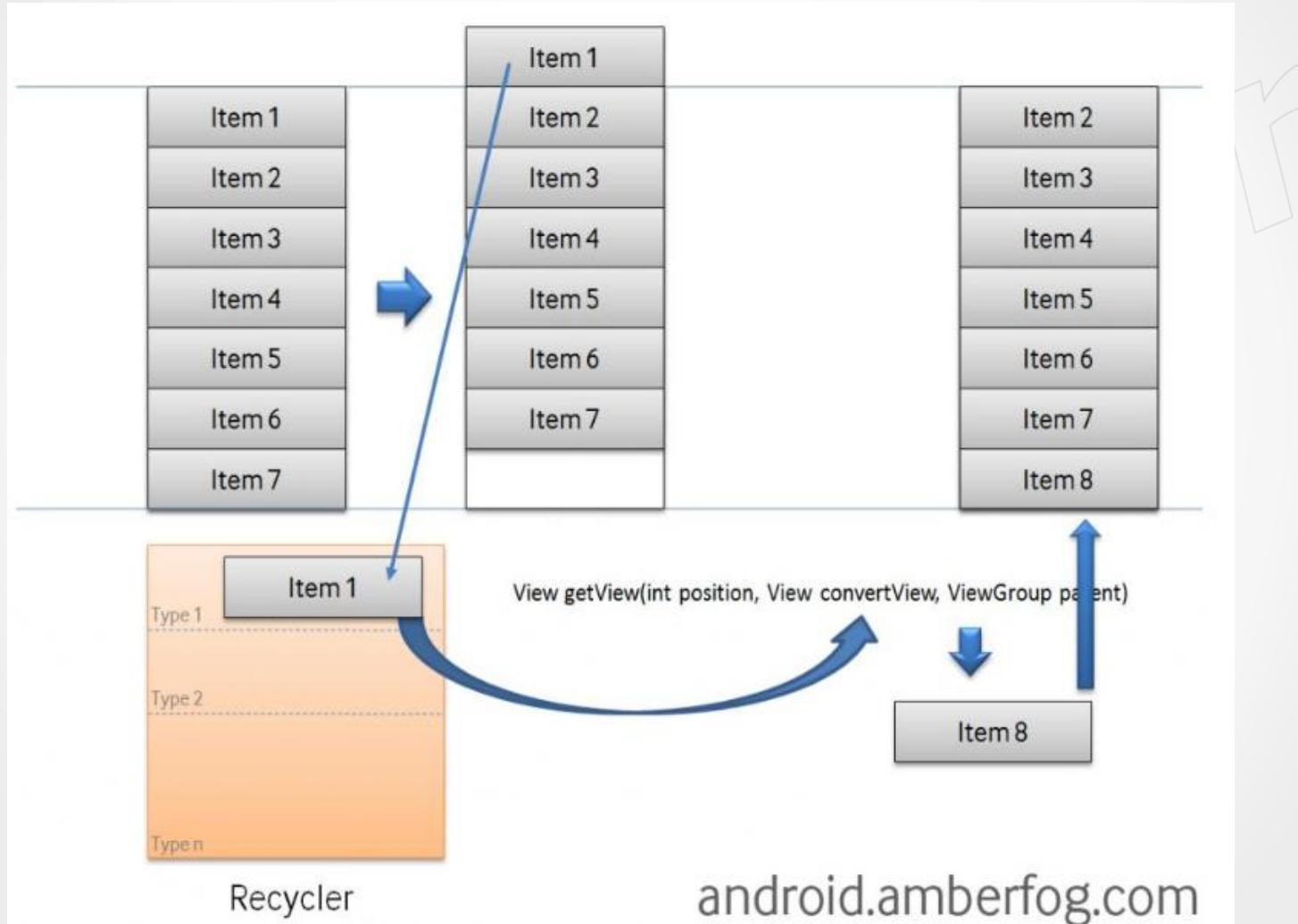
# ListView運作原理

在做客製化的ListView前，首先要先了解ListView的運作方式。

- ListView的每一個Row都是一個View
- 假設資料100筆，當目前畫面只顯示10筆資料(10個Row)，則在記憶體中也只會建立10筆View
- 當ListView往下或上滑動時，進到視窗外的Row的View會被放置進回收區域(Recycler)
- 當新Row因為滑動而出現時，該筆Row的View會先看回收區內是否有不使用的View並取回重複使用

# ListView 運作原理

64



# 實作Adapter

客製化的ListView就是要讓Row使用自己所做的Layout，前章節中的ArrayAdapter只能指定一個TextView來顯示文字，因此若是複雜一點的Layout，就必須自行實作出Adapter

- 繼承BaseAdapter<T>抽象類別，T為資料集合的型別
- 實作抽象方法或屬性-this、Count、GetItemId()、GetView()

```
public class MyAdapters : BaseAdapter<Item>
{
    1 個參考
    public override Item this[int position]
    {
        get { throw new NotImplementedException(); }
    }

    1 個參考
    public override int Count
    {
        get { throw new NotImplementedException(); }
    }

    1 個參考
    public override long GetItemId(int position)
    {
        throw new NotImplementedException();
    }

    1 個參考
    public override View GetView(int position, View convertView, ViewGroup parent)
    {
        throw new NotImplementedException();
    }
}
```

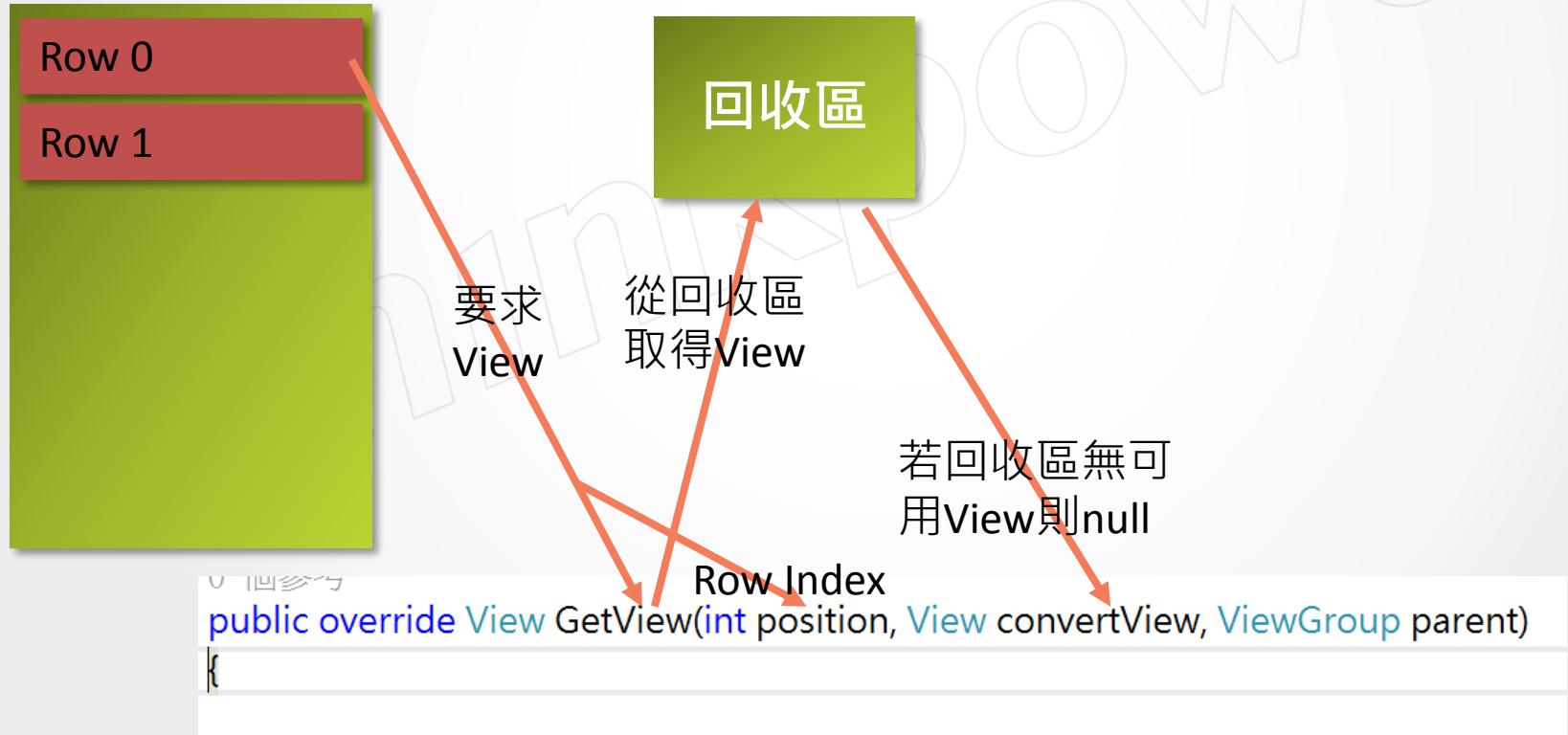
# 將Layout變成View物件

- 在Activity下可取得型別為LayoutInflater的屬性名:LayoutInflater
- 使用LayoutInflater的Inflate()方法，傳入layout的ID，及要加入的View群組(如不需要則可為null)，會回傳這個layout產生的View物件
- 透過取得的View物件，使用FindViewById<T>()方法，就可以找到這個layout下所有的控制項。

```
View view=this.LayoutInflater.Inflate(Resource.Layout.RowItem, null);
// view.FindViewById<.....>
```

# 實作GetView()

實作BaseAdapter<T>的抽象方法GetView()會要求傳回View型別的物件，該方法就是ListView在畫面上每一個Row要輸出View時，會到此方法中取得View，該方法參數提供了Row的索引值，以及從回收區取回的View(若沒有回收可用則傳入null)



# 實作GetView()

- 判斷傳進的參數convertView是否為null值。
- 若為null值則使用LayoutInflater建立出View
- 由參數position取得對應的資料
- 使用FindViewByID< T >修改View中的資料。

```
public override View GetView(int position, View convertView, ViewGroup parent)
{
    if (convertView == null)//NULL代表VIEW未建立
    {
        convertView = Context.LayoutInflater.Inflate(Resource.Layout.RowItem, null);//
    }
    var item = Items[position];
    convertView.FindViewById<TextView>(Resource.Id.textView1).Text = item.Text;/
    var image=convertView.FindViewById<ImageView>(Resource.Id.image);
    var drawable = Context.Resources.GetDrawable(Context.Resources.GetIdentifier(
    image.SetImageDrawable(drawable);
    return convertView;
}
```

# ListView使用注意

- ListView的高度儘量設定成match\_parent(or fill\_parent) , 避免因ListView為了計算Row的高度而重複多次跑GetView()
- 若客製化的Row內有會強制Focus的控制項，如Button、CheckBox等，則要先將其focusable設為false，避免Row的Click事件失效
- Row中如有Button控制項需要註冊事件時，要判斷該Row使用的View是否由回收區收回的，避免Button的事件重複註冊
- 更新資料時應對資料做更新而非畫面上的值，再使用Adapter的NotifyDataSetChanged()讓資料更新到畫面上



# Sqlite資料處理

# Sqlite簡述

- SQLite是遵守ACID的關聯式資料庫管理系統，是行動裝置上一種輕量級的資料庫系統
- 使用了標準的SQL語言來進行操作，支援了大部分的SQL語法
- 完整支援Unicode

# Sqlite型別

型別名稱	對應的SQLite型別
INT INTEGER TINYINT SMALLINT MEDIUMINT BIGINT UNSIGNED BIG INT INT2 INT8	INTEGER
CHARACTER(20) VARCHAR(255) VARYING CHARACTER(255) NCHAR(55) NATIVE CHARACTER(70) NVARCHAR(100) TEXT CLOB	TEXT

# Sqlite型別

型別名稱	對應的SQLite型別
BLOB <i>no datatype specified</i>	NONE
REAL DOUBLE DOUBLE PRECISION FLOAT	REAL
NUMERIC DECIMAL(10,5) BOOLEAN DATE DATETIME	NUMERIC

# Sqlite-使用ADO.Net

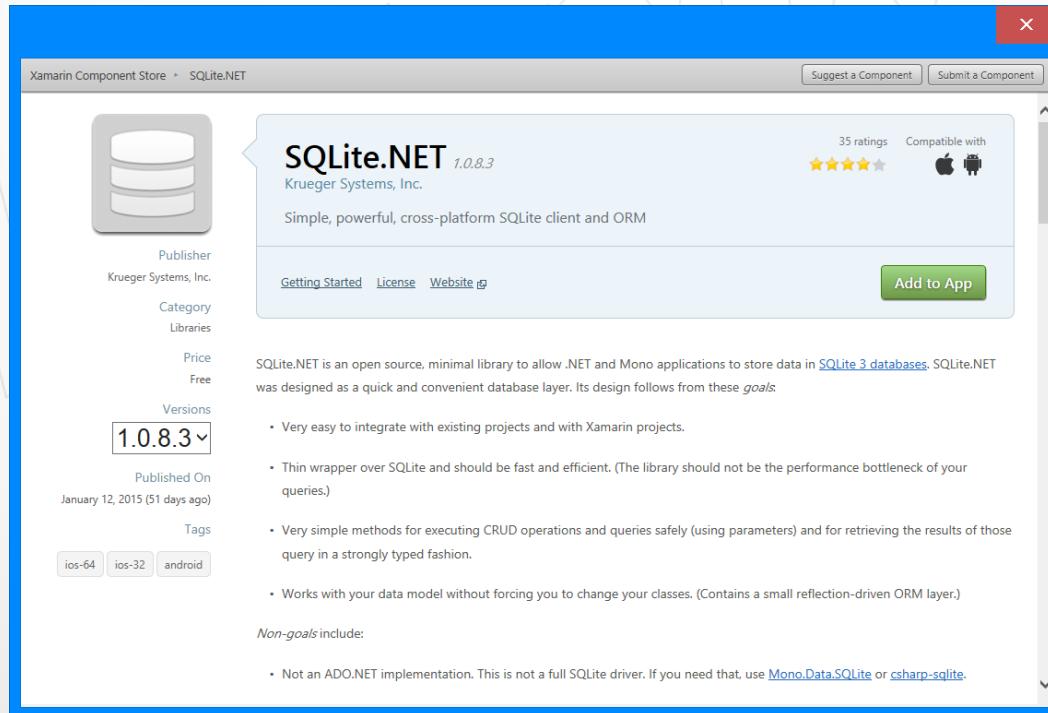
- 於專案參考中將System.Data與System.Data.SQLite加入參考
- 建立連線字串，並在字串內指定資料庫的位置  
(Data Source=資料庫路徑;Version=3)
- 使用SqliteConnection建立對資料庫連線的物件
- 使用SqliteConnection.CreateFile()建立資料庫檔案
- 使用SqliteCommand類別對資料庫進行指令操作

```
using (var conn = new SqliteConnection(_ConnectionString))
{
    conn.Open();
    using (var cmd = conn.CreateCommand())
    {
        cmd.CommandType = CommandType.Text;
        cmd.CommandText = "Select * from.....";
        using (var reader = cmd.ExecuteReader())
        {
            _Data = new List<Info>();
            while (reader.Read())

```

# Sqlite-使用Sqlite.NET

- Sqlite.NET為ORM ( Object Relational Mapping ) 導向的設計方式，不需寫Sql指令使用Linq即可對資料庫進行操作
- 於Xamarin的Components Store中取得Sqlite.NET元件



# Sqlite.NET常用方法

方法名稱	說明
CreateTable<T>()	使用指定的泛型型別T建立資料表
DropTable<T>()	使用指定的泛型型別T刪除資料表
Insert(object obj)	新增一筆資料至資料表中
InsertAll(IEnumerable objects)	新增多筆資料至資料表中
Delete(object objectToDelete)	刪除一筆資料
DeleteAll<T>()	將資料表的資料全部刪除
Table<T>()	取得指定的資料表資料
Execute(string query, params object[] args)	使用SQL指令對資料庫下指令
ExecuteScalar<T>(string query, params object[] args)	使用SQL指令對資料庫查詢資料

# 使用Sqlite

- 使用SQLiteConnection並傳入連線字串建立連線物件
- 使用CreateTable<T>方法，會依照T的類別屬性建立對應的Table欄位
- 使用Table<T>則可用Linq查詢該Table資料
- 使用Delete(T object)、Insert(T object)、Update(T object)可刪除、新增及更新資料。

```
string dbName = "L08_SqliteNet.db";
_DBPath = Path.Combine(FilesDir.Path, dbName);
_Conn = new SQLiteConnection(_DBPath);
_Conn.CreateTable<Info>();
```



## GCM推播服務

# GCM簡介

79

- GCM全名為Google Cloud Messaging，是由Google提供的服務
- 可讓 Android app 的開發者從自己的伺服器傳送訊息到安裝在 Android 設備的 app
- 若 client 四周內都沒有連上 GCM server，則 GCM 上訊息將被丟棄
- 一個 client 在 GCM 上訊息，最多可保留 100 則
- 建議一次最多發送100 則訊息至 client APP

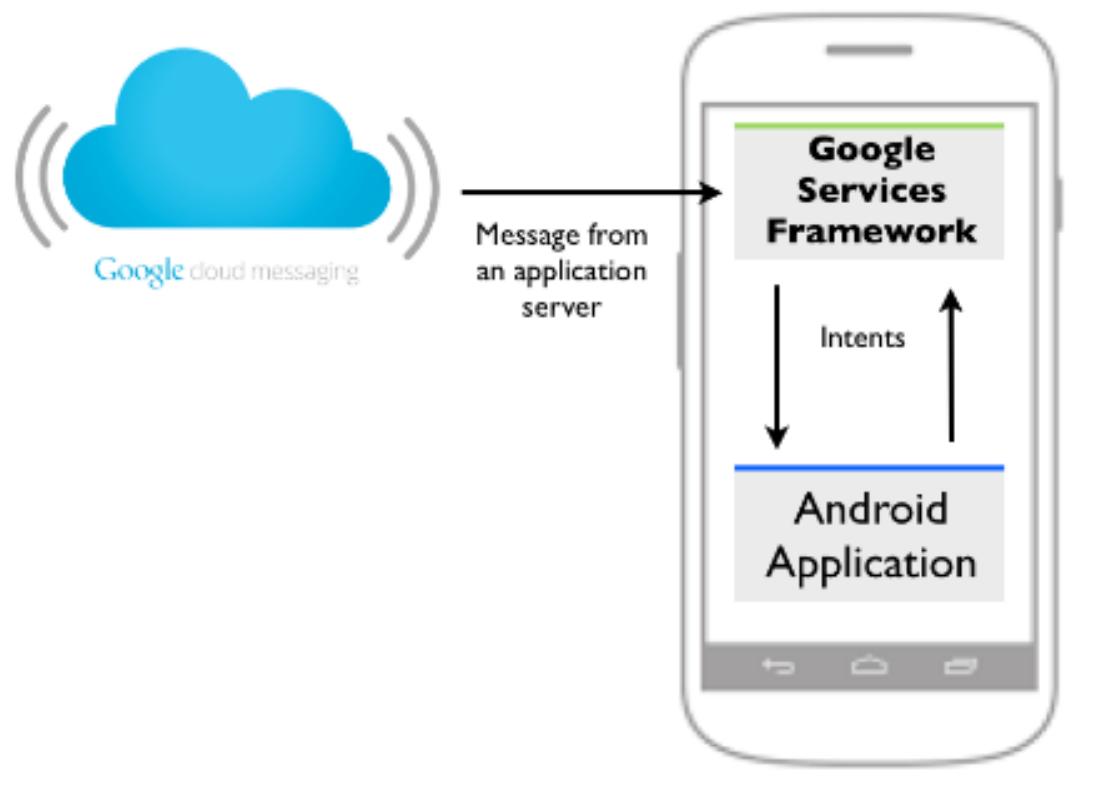
# GCM流程

80

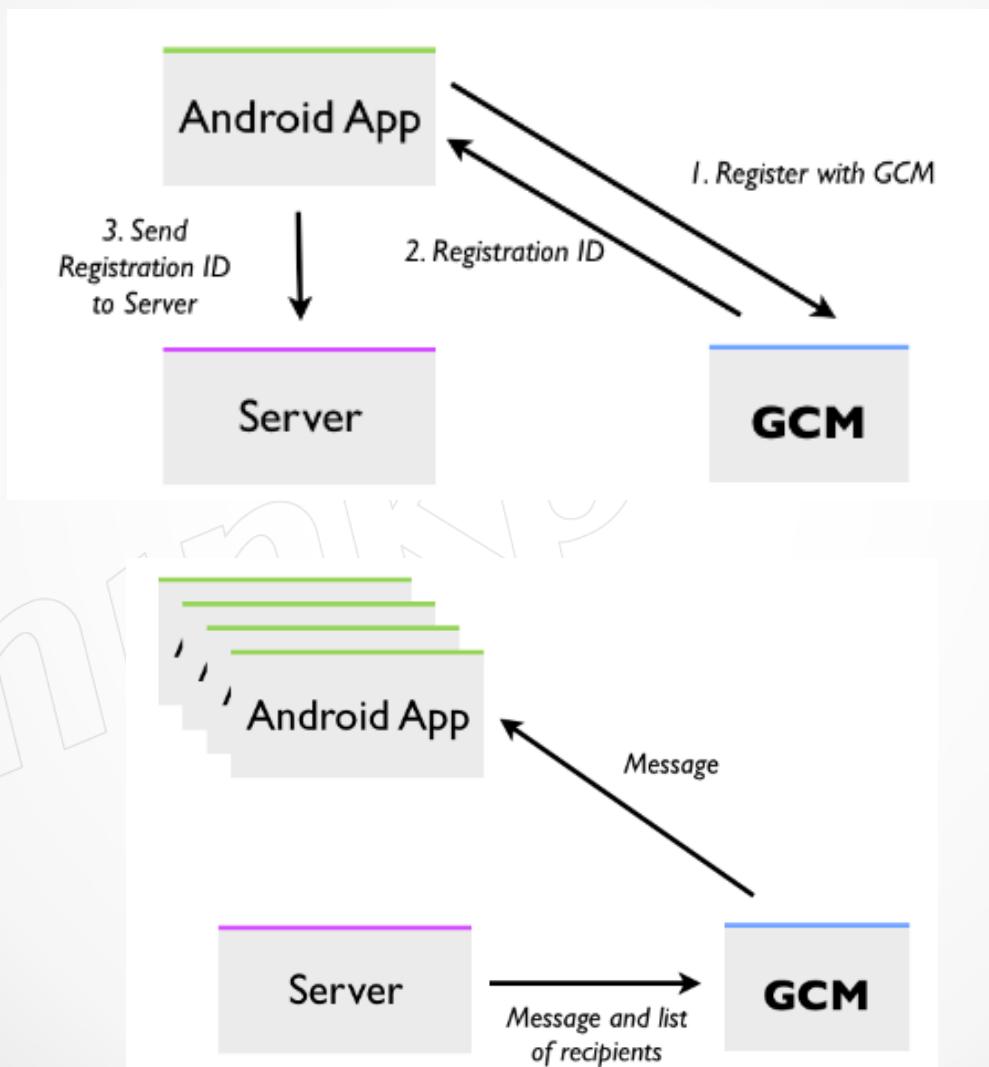
- Client(安裝APP的一方) 端對 GCM server 註冊取得 regId，regId 長度最大為4K，以最小的文字編碼單一字1byte表示最長可達4096長度。
- Client(APP) 將 取得的regId 傳給要推送訊息的 server 存起來
- Server(推送端) 取得 client(APP) regId，並將 regId 與訊息以 JSON 格式傳送至 GCM server
- GCM server 將訊息傳送至 client(APP)

# GCM流程

APP端是由手機內已安裝的Google Services Framework負責接收由GCM發出的推播訊息



# GCM流程



# 啟用GCM服務

■ 要使用GCM服務，需要有Google帳號並開啟GCM API。

<https://code.google.com/apis/console>

■ 在網站內建立新的專案



專案的用途是將不同的資源加以歸類分組。

首先，請建立您的第一項專案。

建立專案



## 新增專案

專案名稱 ?

MyTest



專案 ID ?

sylvan-ocean-87510



[顯示進階選項...](#)



I agree that my use of any [services and related APIs](#) is subject to my compliance with the applicable [Terms of Service](#).

建立

取消

# 啟用GCM服務

## ■ 啟用API服務

The screenshot shows the Google Cloud Deployment Manager API page. It lists two services with their respective daily request limits:

- Google Cloud DNS API**: 50,000 個要求/天 (Limits tab)
- Google Cloud Messaging for Android**: 無 (Limits tab)

A red dashed circle highlights the "關..." (Disable) button for the Google Cloud Messaging for Android service.

API	Google Cloud Deployment Manager API	10,000 回要求/天	關...
Google Cloud DNS API	50,000 個要求/天	關...	
Google Cloud Messaging for Android	無	關...	

## ■ 建立憑證，選擇伺服器金鑰

The screenshot shows the Google Developers Console interface for the project "MyTest". The left sidebar shows the "憑證" (Credentials) option selected under "API 和驗證". A modal window titled "建立新的金鑰" (Create New Credential) is open, explaining the purpose of credentials and providing four options: "伺服器金鑰" (Server Key), "瀏覽器金鑰" (Browser Key), "Android 金鑰" (Android Key), and "iOS 金鑰" (iOS Key).

建立新的金鑰

Google Developers Console 中的 API 規定所有要求都必須包含專案的專屬識別碼。這樣一來，Google Developers Console 才能將要求連結至相對應的專案，以便監控流量、執行配額限制及處理帳單。

伺服器金鑰  
瀏覽器金鑰  
Android 金鑰  
iOS 金鑰

# 啟用GCM服務

## ■ 如不須過慮IP則直接建立

建立伺服器金鑰並設定允許使用的 IP 位址

請將這個金鑰妥善保存在您的伺服器中，避免外洩。

每個 API 要求都是由您所控管裝置上執行的軟體所產生。系統會使用每個要求的 userIP 中所提供的位址，實行使用者限制。如果要求中缺少 userIp 參數，系統會改用您的裝置詳情

接受這些伺服器 IP 位址發出的要求

每行一個 IP 位址或子網路。範例：192.168.0.1、172.16.0.0/16、2001:db8::1 或 2001:db8::1/128

伺服器應用程式的金鑰

建立

取消

API 金鑰

AlzaSyDMlSkD8yDV\_0RfO5OpSAwdiZztRCI0qxk

IP 位址

任何允許使用的 IP 位址

啟用日期

2015年3月5日 上午10:47:00

啟用者

stevenghost.ghost@gmail.com (您本人)

編輯允許使用的 IP 位址

重新產生金鑰

刪除

# 啟用GCM服務

## ■ 取得專案編號與API 金鑰

專案 ID : polar-equinox-87510 專案編號: 841039486856

專案資訊主頁

伺服器應用程式的金鑰

API 金鑰	AlzaSyDMISkD8yDV_0RfO50pSAwdiZztRCI0qxk
IP 位址	任何允許使用的 IP 位址
啟用日期	2015年3月5日 上午10:47:00
啟用者	stevenghost.ghost@gmail.com (您本人)

編輯允許使用的 IP 位址 重新產生金鑰 刪除

# 指定Key與專案碼

## ■ 在APP端註冊時會用到專案編號

```
string senders = "31490604079";//放置你的API 專案碼
Intent intent = new Intent("com.google.android.c2dm.intent.REGISTER");
intent.SetPackage("com.google.android.gsf");
intent.PutExtra("app", PendingIntent.GetBroadcast(this, 0, new Intent(), 0));
intent.PutExtra("sender", senders);
this.StartService(intent);
```

## ■ 在Server端發送推播給GCM時，需使用到API金鑰

```
var devices = new[] { reg };
push.RegisterGcmService(new GcmPushChannelSettings("AIzaSyAz_3agZ7V7VbMycgahSRRg9XVcHApOx6o"));
push.QueueNotification(new GcmNotification().ForDeviceRegistrationId(devices)// (Devices)
```

# 撰寫 BroadcastReceiver 程式

- BroadcastReceiver(廣播)通常用來通知系統，比如收到簡訊、電池量變化等，都可用廣播系統接收，我們可用它來接收由GCM傳來的 Registration ID與推播訊息。

```
namespace L09_GCM
{
    //用來接收廣播事件的BroadcastReceiver
    [BroadcastReceiver(Permission= "com.google.android.c2dm.permission.SEND")]
    [IntentFilter(new string[] { "com.google.android.c2dm.intent.RECEIVE" }, Categories = new string[] { "thinkpc" })
    [IntentFilter(new string[] { "com.google.android.c2dm.intent.REGISTRATION" }, Categories = new string[] { "thinkpc" })
    [IntentFilter(new string[] { "com.google.android.gcm.intent.RETRY" }, Categories = new string[] { "thinkpc" })
    0 個參考
    public class MyGCMBroadcastReceiver : BroadcastReceiver
    {
        const string TAG = "PushHandlerBroadcastReceiver";
        0 個參考
        public override void OnReceive(Context context, Intent intent)
        {
            MyIntentService.RunIntentInService(context, intent);
            SetResult(Result.Ok, null, null);
        }
    }

    [Service]
    2 個參考
    public class MyIntentService : IntentService
    {
        public MyIntentService() : base("MyIntentService")
    }

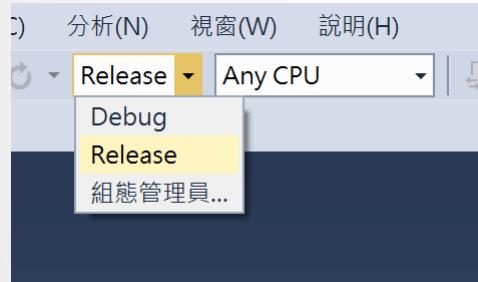
    protected override void OnHandleIntent(Intent intent)
    {
        string action = intent.Action;
        if (action == "com.google.android.c2dm.intent.RECEIVE")
        {
            string registrationId = intent.GetStringExtra("registration_id");
            string message = intent.GetStringExtra("message");
            // Handle registration ID and message
        }
        else if (action == "com.google.android.c2dm.intent.REGISTRATION")
        {
            string registrationId = intent.GetStringExtra("registration_id");
            // Handle registration ID
        }
        else if (action == "com.google.android.gcm.intent.RETRY")
        {
            string error = intent.GetStringExtra("error");
            // Handle retry error
        }
    }
}
```



# 程式佈署、安裝及偵錯

# 封裝程式

- 將偵錯模式更改為Release模式
- 選擇建置→Export Android Package
- Apk檔會產生再bin\Release目錄下
- 結尾為-Signed.apk的才是正確的安裝檔

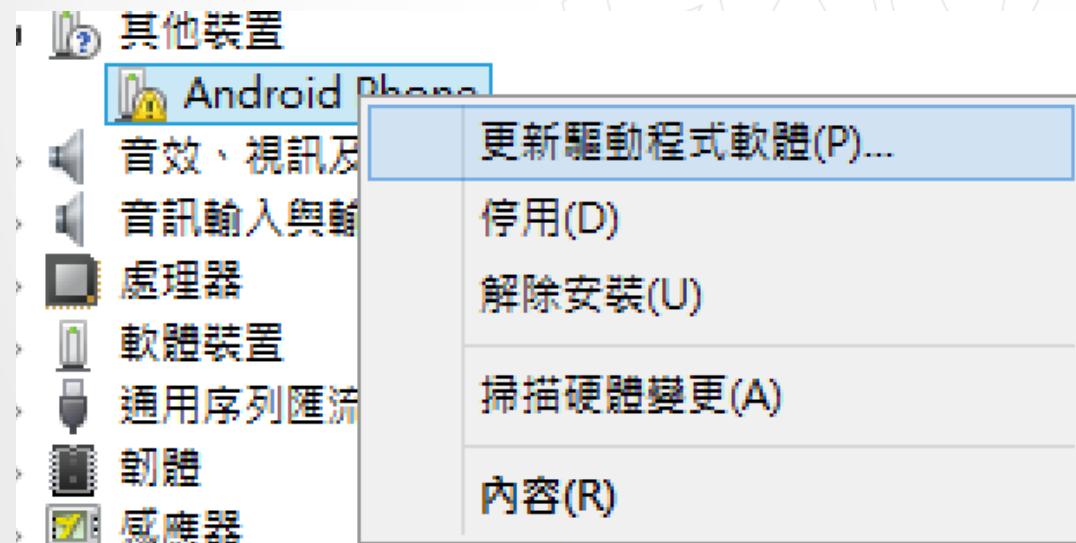


名稱	修改日期	類型
L01.dll	2015/3/5 下午 07:03	應用程式擴充
L01.dll.mdb	2015/3/5 下午 07:03	MDB 檔案
L01.L01.apk	2015/3/5 下午 07:03	APK 檔案
L01.L01-Signed.apk	2015/3/5 下午 07:03	APK 檔案

# 驅動程式安裝

- 需安裝實機的Usb驅動程式。
- 若找不到驅動程式也可安裝SDK提供的ADB驅動：

開啟電腦中的裝置管理員，找到未安裝驅動程式的地方選擇更新驅動程式。



# 驅動程式安裝

選擇瀏覽電腦上的驅動程式軟體 → 讓我從電腦上的裝置驅動程式清單中挑選 → 顯示所有裝置

您要如何搜尋驅動程式軟體？

→ 自動搜尋更新的驅動程式軟體(S)

除非您在裝置安裝設定中停用此功能，否則 Windows 搜尋是否有裝置適用的最新驅動程式軟體。

→ 瀏覽電腦上的驅動程式軟體(R)

手動尋找並安裝驅動程式軟體。

在您的電腦上瀏覽驅動程式軟體

在此位置搜尋驅動程式軟體:

`\ppData\Local\Android\android-sdk\extras\google\usb_driver\`

包含子資料夾①

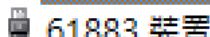
→ 讓我從電腦上的裝置驅動程式清單中挑選(L)

此清單會顯示已安裝並且與裝置相容的驅動程式軟體，以及與裝置驅動程式軟體。

請從下列清單中選取您裝置的類型。

一般硬體類型(H):

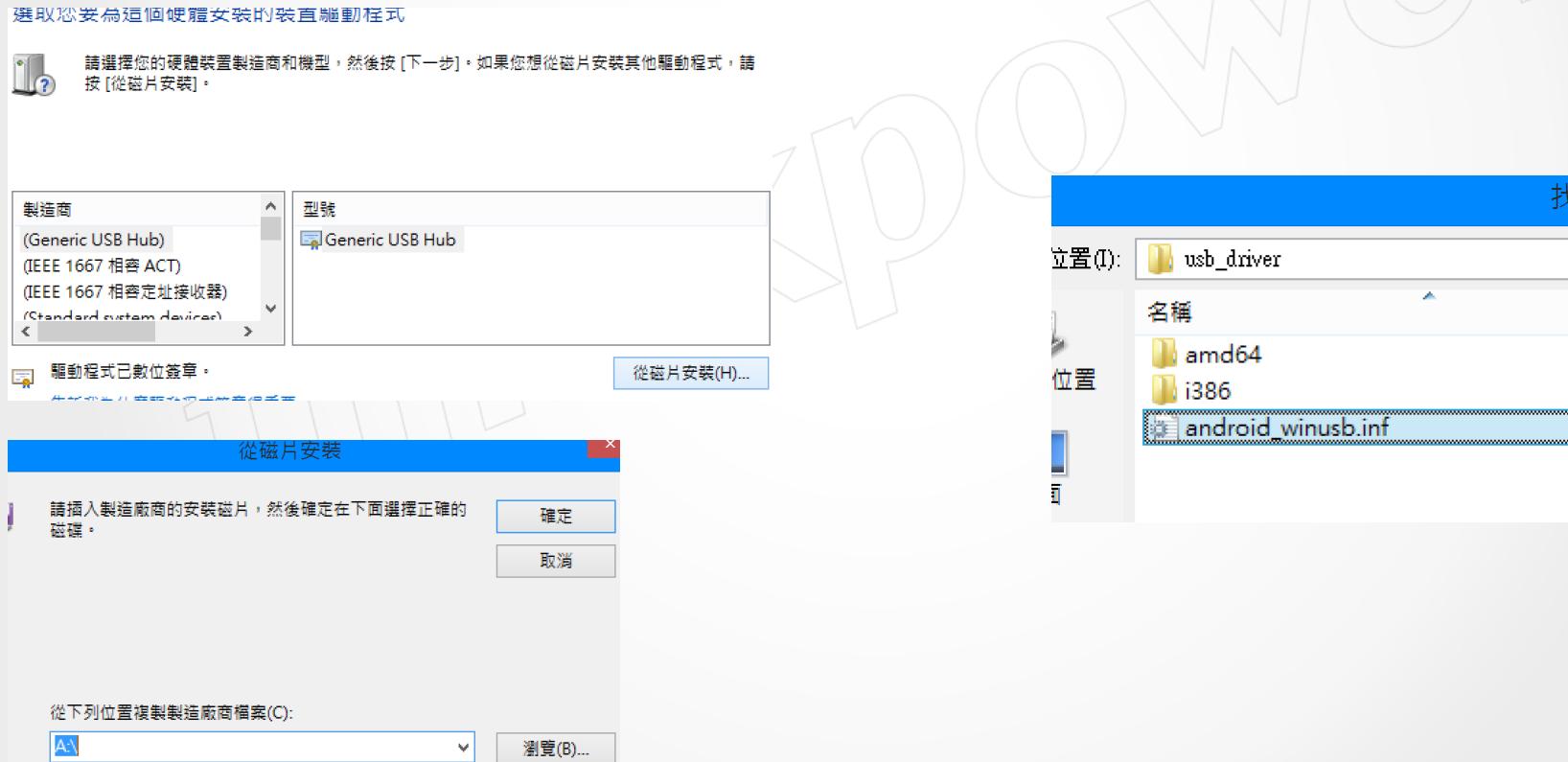
顯示所有裝置



61883 裝置

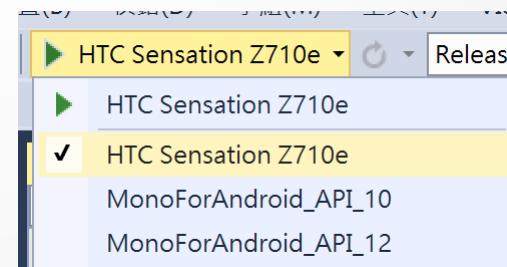
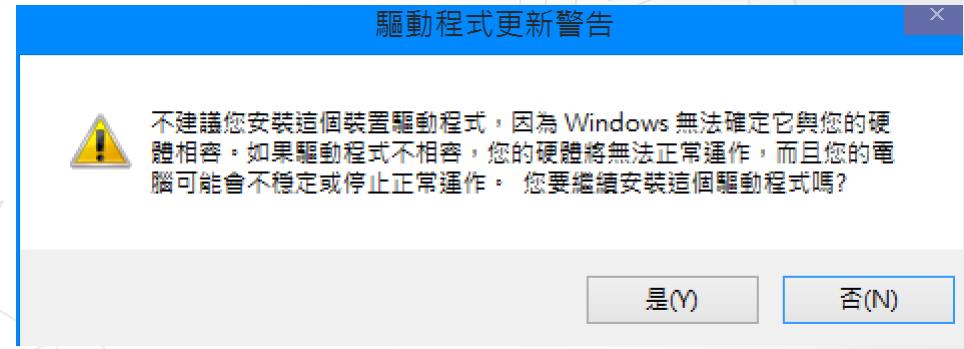
# 驅動程式安裝

點選從磁片安裝，選擇瀏覽，到Android SDK安裝路徑下找到extras\google\usb\_driver，並選擇android\_winusb.inf



# 驅動程式安裝

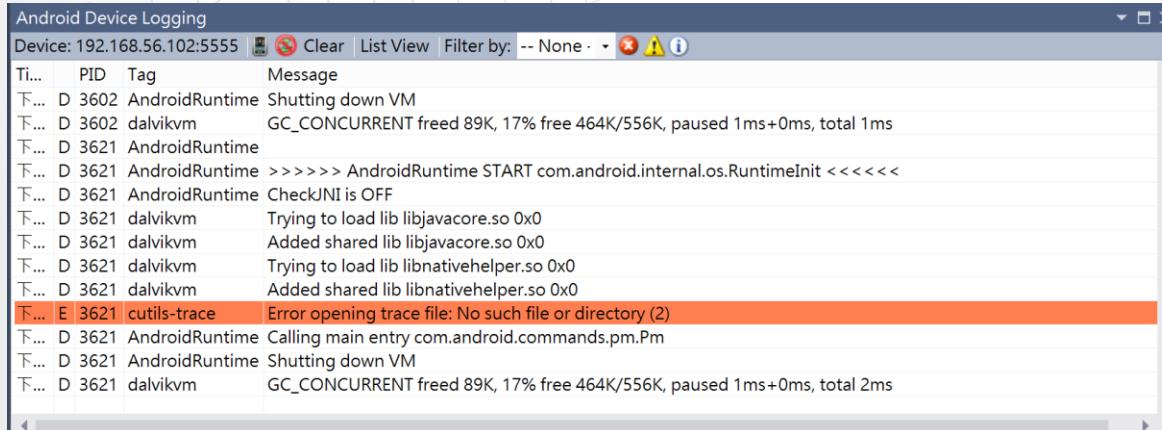
選擇Android ADB Interface開始安裝驅動程式，跳出警告按是即可，安裝完成後就可以接上實機選擇裝置。



# LOG偵錯

- 在程式碼中可用SDK提供的Log功能寫入Console log
- 在Visual Studio開啟Android Device Logging可查看看應用程式LOG

```
Android.Util.Log.WriteLine(Android.Util.LogPriority.Debug, "Tag", "Test message");
```



# 開啟偵錯模式

- 接上實機後，需要開啟手機上的偵錯模式
- 若沒看到開發者選項，可到關於手機中的軟體版本，連續點擊至被隱藏的開發者選項打開為止。

