

คู่มือ

พัฒนาแอปพลิเคชันแบบ Multi-Platform ด้วย



# .NET MAUI



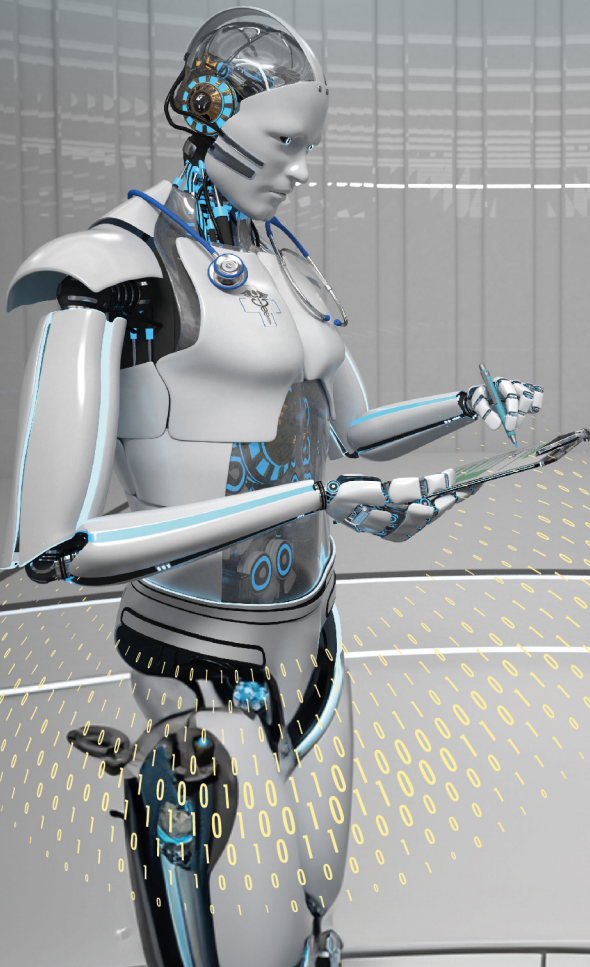
Multi-platform



One codebase



Productive



- 🔘 เรียนรู้พื้นฐานและเทคนิคการพัฒนาแอปพลิเคชัน .NET ข้ามแพลตฟอร์ม
- 🔘 รองรับการพัฒนาแอปพลิเคชันบน Android, MacOS, iOS, Windows และ Web Base
- 🔘 พัฒนาโค้ดเพียงครั้งเดียวสามารถใช้งานได้ทุกแพลตฟอร์ม
- 🔘 มีโค้ดตัวอย่างและคำอธิบายอย่างละเอียดอ่านเข้าใจง่าย
- 🔘 อธิบายเป็นขั้นตอน เหมาะสำหรับผู้เริ่มต้น



ไฟล์ตัวอย่างภายในเล่ม  
<https://serazu.com/9786164874732>

ศุภชัย สมพานิช

# สารบัญ

<b>บทที่ 1</b>	<b>การพัฒนาแอปด้วย .NET MAUI .....</b>	<b>1</b>
	การดาวน์โหลดและติดตั้งโปรแกรม Visual Studio 2022 .....	2
	สถานะของ .NET SDK .....	6
	การติดตั้งฟอนต์ Cascadia Code สำหรับเขียนโค้ดโดยเฉพาะ .....	7
	การกำหนดให้แสดงหมายเลขบรรทัด .....	9
	การติดตั้ง .NET SDK เพิ่มเติม .....	9
	การอัปเดตโปรแกรม Visual Studio .....	11
	วิธีการจัดเก็บโปรเจกต์ .NET .....	12
	การดาวน์โหลด Android Emulator เพิ่มเติมเพื่อทดสอบโปรเจกต์แบบ Android Apps .....	15
	ประเภทโปรเจกต์ .NET MAUI .....	18
	การยกเลิกฟีเจอร์ Nullable ของโปรเจกต์ ASP.NET Core Web API .....	19
<b>บทที่ 2</b>	<b>พื้นฐานการพัฒนาแอปด้วย .NET MAUI .....</b>	<b>21</b>
	พื้นฐานการสร้างโปรเจกต์ .NET MAUI .....	21
	การสร้าง Android Emulator ของ Visual Studio สำหรับทดสอบ Android Apps .....	25
	การปรับช่วงเวอร์ชันของ Android API .....	28
	การอัปเดต Android SDK ของ Visual Studio .....	30
	การทดสอบโปรเจกต์ .NET MAUI แบบ Android Apps ด้วย Android Emulator .....	30
	การทดสอบโปรเจกต์ Android Apps บนเครื่องจริง (ใช้กับ Android 8.0 ขึ้นไปเท่านั้น) .....	32
	การทดสอบโปรเจกต์ .NET MAUI แบบ Android Apps บนเครื่องจริง .....	35
	การรันโปรเจกต์ .NET MAUI ในรูปแบบ Windows Apps .....	36
	ทำความเข้าใจกับโครงสร้างโปรเจกต์ .NET MAUI .....	39
	การสร้างเหตุการณ์ (Event) ด้วยวิธีการเขียนโค้ด .....	50
	ทำความเข้าใจกับฟีเจอร์แสดงผลทันที (Hot Reload) .....	54
	วิธีการจัดเก็บโปรเจกต์ .NET MAUI .....	55
	การตรวจสอบแพลตฟอร์ม .....	57
	การตกแต่งแถบข้อความด้วยอีลีเมนต์ <Border>...</Border> .....	60
	การลงแสงเงาให้กับรูปภาพ (Image Shadow) .....	62

<b>บทที่ 3</b>	<b>ตำแหน่งและ Layout ขั้นต้นของ .NET MAUI .....</b>	<b>64</b>
	การจัดเรียงแนวตั้งด้วย VerticalStackLayout .....	64
	การจัดเรียงแนวนอนด้วย HorizontalStackLayout .....	66
	การกำหนดระยะห่างด้วยคุณสมบัติ Margin .....	67
	การกำหนดระยะห่างภายในคอนโทรลด้วยคุณสมบัติ Padding .....	69
	การจัดตำแหน่งข้อความด้วยกลุ่มคุณสมบัติ TextAlignment .....	72
	พื้นฐานการสร้างช่องรับข้อมูลด้วยอีลีเมนต์ <Entry /> .....	73
	การแสดงผลเนื้อหาด้วย ContentView .....	79
	พื้นฐานการใช้ Layout แบบตาราง Grid .....	80
	การรวมคอลัมน์ (Grid.ColumnSpan) หรือแถว (Grid.RowSpan) ใน Grid .....	83
	ความหมายของการกำหนดขนาดแบบ Auto และ * ใน Grid .....	85
	การสร้าง Layout แบบเลื่อนดูเนื้อหาด้วย ScrollView .....	88
<b>บทที่ 4</b>	<b>การใช้งานคอนโทรล (Controls) หรืออีลีเมนต์ (Element) .....</b>	<b>90</b>
	พื้นฐานการตกแต่งคุณสมบัติ (Property) หรือแอตทริบิวต์ (Attribute) ของคอนโทรล ...	90
	การสร้างปุ่มกดด้วยคอนโทรล Button .....	91
	การสร้างปุ่มกดแบบรูปภาพด้วยอีลีเมนต์ ImageButton .....	93
	การแสดงผลข้อความธรรมดาด้วย TextCell .....	96
	การสร้างช่องรับข้อมูลแบบ EntryCell .....	99
	การสร้างตัวเปิด-ปิดสถานะด้วย SwitchCell .....	101
	การสร้างตัวเลือกวันที่ด้วย DatePicker .....	104
	การสร้างตัวเลือกเวลาด้วย TimePicker .....	108
	การสร้าง Resource ส่วนกลาง .....	110
	การสร้าง Static Resources แบบกำหนดค่าเดียว .....	110
	การสร้าง Static Resources แบบกำหนดหลายค่า (Style) .....	112
	การสร้าง Dynamic Resources .....	115
	การแจ้งเตือน (DisplayAlert) .....	118
	การตรวจสอบสถานะเชื่อมต่อ Internet ด้วยคลาส Connectivity .....	124
	การสร้างตัวเลือกไฟล์ด้วยคอนโทรล FilePicker .....	127

<b>บทที่ 5</b>	<b>การผูกติดข้อมูล (Data Binding)</b> .....	<b>132</b>
	พื้นฐานการเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming - OOP).....	132
	พื้นฐานการผูกติดข้อมูล .....	133
	การผูกติดระหว่างคอนโทรล (หรืออีลีเมนต์).....	143
	โหมดการผูกติด (Data Binding Mode).....	145
<b>บทที่ 6</b>	<b>การสั่งให้เกิดการทำงานด้วยคลาส Command</b> .....	<b>149</b>
	การใช้งานคลาส Command .....	149
	การส่งพารามิเตอร์ให้กับ Command (Command Parameter).....	157
	การกำหนดค่าเริ่มต้นให้กับตัวเก็บสถานะ .....	163
<b>บทที่ 7</b>	<b>การพัฒนาแอปแบบหลายหน้าจอ</b> .....	<b>166</b>
	พื้นฐานการเพิ่มหน้าจอใหม่เข้ามาในโปรเจกต์ .....	166
	การสร้าง Resource ใน App.xaml สำหรับใช้ได้ทั้งโปรเจกต์.....	170
	การรับ-ส่งข้อมูลระหว่างหน้า .....	173
	การสร้างส่วนแสดงผลแบบแท็บ (Tab).....	176
	การสร้างเมนูสำหรับ Desktop Apps.....	180
<b>บทที่ 8</b>	<b>การสร้างรายการด้วย ListView</b> .....	<b>185</b>
	พื้นฐานการสร้างรายการแบบ list ด้วย ListView .....	185
	การปรับแต่งรายการใน ListView.....	189
	การแบ่งการทำงานแบบ MVVM .....	197
<b>บทที่ 9</b>	<b>การทำงานกับฟังก์ชันพื้นฐานของอุปกรณ์ Mobile Devices (.NET MAUI Essentials)</b> .....	<b>209</b>
	ฟังก์ชันการโทรออก .....	209
	การส่งข้อความ (SMS) .....	214
	การตรวจสอบ Hardware ด้วยคลาส DeviceInfo .....	217
	การตรวจสอบรายละเอียดแอปด้วย AppInfo.....	220
	การตรวจสอบสถานะ Battery .....	224
	การตรวจสอบสถานะการเชื่อมต่ออินเทอร์เน็ต .....	229
	การเปิดเว็บเพจด้วย Browser .....	232
	การ Copy & Paste ข้อความจากคลิปบอร์ด.....	234
	การสร้างไฟล์เก็บข้อความส่วนกลาง (Shared Preferences) .....	237

<b>บทที่ 10</b>	<b>.NET MAUI Blazor</b>	<b>241</b>
	พื้นฐานการสร้างโปรเจกต์ .NET MAUI Blazor App	241
	ทำความเข้าใจกับโครงสร้างโปรเจกต์ .NET MAUI Blazor App	245
	Layout หลักของโปรเจกต์ .NET MAUI Blazor (MainLayout.razor)	249
	ระบบเมนู (NavMenu.razor)	251
	ส่วนแสดงผล Index.razor	253
	ส่วนแสดงผล Counter.razor	256
	ส่วนแสดงผล FetchData.razor	257
<b>บทที่ 11</b>	<b>พื้นฐานการสร้าง Web API ด้วย ASP.NET Core Web API</b>	<b>265</b>
	ASP.NET Core Web API คืออะไร	265
	การสร้างโปรเจกต์ ASP.NET Core Web AP	266
	การทดสอบและหยุดรันโปรเจกต์ ASP.NET Core Web API	269
	ทำความเข้าใจกับโครงสร้างโปรเจกต์ ASP.NET Core Web API	274
	การสร้าง Web API แรกด้วย ASP.NET Core Web API	279
<b>บทที่ 12</b>	<b>การสร้าง .NET MAUI Apps แบบ CRUD เชื่อมต่อกับ ASP.NET Core Web API (Front End)</b>	<b>289</b>
	การสร้างโปรเจกต์ .NET MAUI แบบ CRUD	289
	Repository Pattern ฝั่ง Front End	297
	การกำหนดให้ Android Emulator เชื่อมต่อกับ localhost	317
	การสร้างหน้าจอแสดงรายการลูกค้า (MainPage.xaml)	320
<b>บทที่ 13</b>	<b>การสร้าง .NET MAUI Apps แบบ CRUD เชื่อมต่อกับ ASP.NET Core Web API (Back End)</b>	<b>337</b>
	ทำความเข้าใจกับโครงสร้างแบบหลายโปรเจกต์	337
	Repository Pattern ฝั่ง Back End	342
	วิธีการรันหลายโปรเจกต์	359

<b>บทที่ 14</b>	<b>พื้นฐานการพัฒนา Web Apps แบบคอนเทนเนอร์ด้วย Docker.....</b>	<b>369</b>
	การดาวน์โหลดและติดตั้ง Windows Subsystem for Linux (WSL) .....	370
	การดาวน์โหลดและติดตั้ง Docker.....	372
	การทำงานกับ Docker Image/Container ในเบื้องต้น .....	374
	การลบอิมเมจ.....	381
<b>บทที่ 15</b>	<b>การสร้าง ASP.NET Core Web API ด้วย Docker .....</b>	<b>383</b>
	การสร้างโปรเจกต์ ASP.NET Core Web API ที่มีการใช้งาน Docker .....	383
<b>บทที่ 16</b>	<b>การเผยแพร่อิมเมจแบบสาธารณะกับ Docker Registry .....</b>	<b>392</b>
	การ push อิมเมจสู่ Docker Registry .....	392
<b>บทที่ 17</b>	<b>การสร้างไฟล์ Signed APK และ Signed AAB ด้วย .NET MAUI .....</b>	<b>399</b>
	การกำหนด Target Android Framework.....	399
	วิธีการสร้างไฟล์ Key Store ให้กับ Android Apps .....	400
	การสร้างไฟล์ Signed APK และไฟล์ Signed AAB จากโปรเจกต์ .NET MAUI แบบ Android Apps.....	403
<b>บทที่ 18</b>	<b>การอัปโหลดแอปขึ้น Google Play Store.....</b>	<b>406</b>
	การสมัครบัญชีนักพัฒนา Android Apps .....	406
	ขั้นตอนการอัปโหลดไฟล์ Signed AAB ไปสู่ Google Play Store .....	409
<b>บทที่ 19</b>	<b>การสร้างไฟล์ติดตั้งของ .NET MAUI แบบ Windows Apps.....</b>	<b>412</b>
	การสร้างไฟล์ .exe จากโปรเจกต์ .NET MAUI แบบ Windows Apps.....	412
	การสร้างไฟล์ Signed MSIX สำหรับ Windows Apps.....	416





# บทที่ 1

## การพัฒนาแอปด้วย .NET MAUI

.NET MAUI เป็นความก้าวหน้าครั้งสำคัญอีกก้าวหนึ่งของแพลตฟอร์ม .NET เป็นโปรเจกต์ที่ช่วยให้นักพัฒนาสามารถสร้างแอปหลายแพลตฟอร์มในเวลาเดียวกันด้วยโครงสร้างโปรเจกต์เพียงหนึ่งเดียว ช่วยให้การดูแลรักษาโปรเจกต์ในระยะยาวได้อีกทางหนึ่งด้วย

ผลผลิตของโปรเจกต์ .NET MAUI สามารถสร้างแอปได้ 5 แบบ คือ

1. **Windows Apps** เป็นแอปประเภท Desktop บนเครื่อง PC
2. **Android Apps** เป็นแอปบนอุปกรณ์ของค่าย Google
3. **iOS Apps** เป็นแอปบนอุปกรณ์มือถือของค่าย Apple
4. **Mac Catalyst Apps** เป็นแอปประเภท Desktop บนเครื่อง MAC
5. **Tizen** เป็นแอปบนระบบปฏิบัติการของซัมซุง

เนื้อหาในหนังสือเล่มนี้เป็นการใช้งานโปรแกรม Visual Studio บนเครื่องคอมพิวเตอร์ PC ส่งผลให้ผู้เขียนสามารถนำเสนอแอปได้ 2 ประเภท คือ Windows Apps และ Android Apps

เนื้อหาที่นำเสนอในหนังสือเล่มนี้ครอบคลุมการพัฒนาแอปในรูปแบบ Front End กับ Back End กล่าวคือ

1. **งานฝั่งหน้าบ้าน (Front End)** เป็นหน้าที่ของโปรเจกต์ .NET MAUI ทำหน้าที่สร้างส่วนแสดงผลต่างๆ (User Interface - UI)
2. **งานหลังบ้าน (Back End)** เป็นหน้าที่ของโปรเจกต์ประเภท ASP.NET Core Web API ทำหน้าที่ติดต่อกับส่วนของข้อมูลเพื่อให้บริการพื้นฐานในรูปแบบ Web API ได้แก่ งานประเภทเพิ่ม, อ่าน, อัปเดต และลบข้อมูล (CRUD)

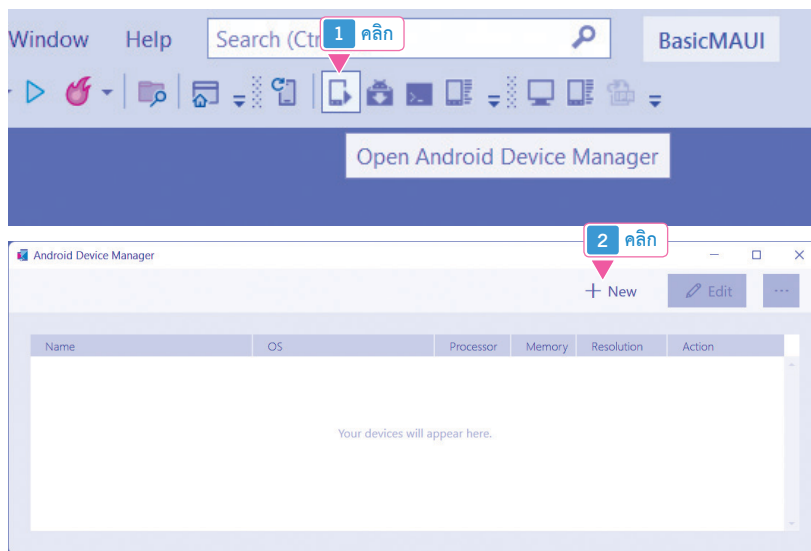


# การสร้าง Android Emulator ของ Visual Studio สำหรับทดสอบ Android Apps

ในขั้นตอนนี้เราจะสนใจโปรเจกต์ .NET MAUI เฉพาะ Android Apps เป็นลำดับแรกก่อน โดยที่เรายังไม่ต้องแก้ไขหรือเขียนโค้ดใดๆ เพิ่มเติม เราต้องการทดสอบว่า Android Apps ที่ได้มามีหน้าตาเป็นอย่างไร

การทดสอบการทำงานของโปรเจกต์ Android Apps ในขั้นต้นคือ อาศัย Android Emulator ที่มากับโปรแกรม Visual Studio มีขั้นตอนดังนี้

1. ในโปรแกรม Visual Studio ยังไม่มี Android Emulator ใดๆ ทั้งสิ้น ให้ผู้อ่านคลิกที่ปุ่ม  จากนั้นคลิกที่ปุ่ม **+ New** เริ่มต้นสร้าง Android Emulator เพื่อทดสอบโปรเจกต์ .NET MAUI ประเภท Android Apps ดังรูปที่ 2-5



รูปที่ 2-5 แสดงการเริ่มต้นสร้าง Android Emulator

2. เป็นขั้นตอนการกำหนดรายละเอียดให้กับ Android Emulator เพื่อจำลองให้มีความเหมือนมือถือเครื่องจริง ที่ต้องกำหนด ได้แก่
  - **Name:** ชื่อ Android Emulator
  - **Base Device:** ให้ผู้อ่านเลือกรุ่นมือถือที่ต้องการจำลองการทำงาน
  - **Processor:** ให้ผู้อ่านเลือก CPU แบบ x86\_x64 ตรงกับ CPU ที่ใช้งานในเครื่อง PC ของเรา

# การสร้างเหตุการณ์ (Event) ด้วยวิธีการเขียนโค้ด

สำหรับวิธีการสร้างเหตุการณ์ (Event) ให้กับอีลีเมนต์หรือคอนโทรลแต่ละตัว ผู้เขียนเลือกใช้วิธีการเขียนโค้ดผูกเหตุการณ์ด้วยวิธีการเขียนโค้ดเอง เหตุผลก็คือ ต้องการแยกส่วนแสดงผล (ไฟล์นามสกุล .xaml) กับโค้ดการทำงาน (ไฟล์นามสกุล .cs) ออกจากกันอย่างสิ้นเชิงนั่นเอง

## ตัวอย่างที่ 2-1 การสร้างเหตุการณ์ (Event) ด้วยวิธีการเขียนโค้ด

มีขั้นตอนดังนี้

1. ที่ไฟล์ส่วนแสดงผล MainPage.xaml ต้องการสร้างปุ่มกด Button แล้วสั่งให้แสดงข้อความ “คุณคลิกที่ปุ่ม Save” ในอีลีเมนต์ Label ประกอบด้วย
  - **อีลีเมนต์ Button** ตั้งชื่อว่า cmdSave (x:Name="cmdSave") กำหนดข้อความ “บันทึก” ในปุ่ม (Text="บันทึก")
  - **อีลีเมนต์ Label** ตั้งชื่อว่า lblOutput (x:Name="lblOutput") ทำหน้าที่แสดงข้อความ

### MainPage.xaml

```
<?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"
  x:Class="UsingEvent.MainPage">

  <ScrollView>
    <VerticalStackLayout>
      <Button x:Name="cmdSave" Text="บันทึก" />

      <Label x:Name="lblOutput" Text=""/>
    </VerticalStackLayout>
  </ScrollView>
</ContentPage>
```

3. ท้ายที่สุด ให้ผู้อ่านลองรันโปรเจกต์ พบว่าเราได้ช่องรับชื่อ-สกุล และที่อยู่ เมื่อคลิกที่ปุ่มบันทึกก็แสดงข้อความที่ผู้ใช้งานป้อนเข้ามา ดังรูปที่ 3-8



รูปที่ 3-8 ผลการรันตัวอย่างที่ 3-6

## การสร้าง Dynamic Resources

**Dynamic Resources** หมายถึง Resource ที่สามารถเปลี่ยนแปลงค่าได้ในขณะเขียนโค้ด ก็จะทำให้การแก้ไขคุณสมบัติต่างๆ ในคอนโทรลยืดหยุ่นมากยิ่งขึ้น

**ตัวอย่างที่ 4-11** การสร้าง Dynamic Resources ดังโค้ดต่อไปนี้

### สคริปต์ XAML ที่ 4-11 การสร้าง Dynamic Resources (MainPage.xaml)

```
<?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"
    x:Class="UsingResource.MainPage">

    <ContentPage.Resources>
        <ResourceDictionary>
            <Style x:Key="LabelStyle" TargetType="Label">
                <Setter Property="BackgroundColor" Value="Green"/>
            </Style>
        </ResourceDictionary>
    </ContentPage.Resources>

    <StackLayout Orientation="Vertical">
        <Button x:Name="cmdChange" Text="เปลี่ยนสี" />
        <Label x:Name="lblOutput"
            BackgroundColor="{DynamicResource LabelStyle}">
            Hello .NET MAUI World!!!
        </Label>
    </StackLayout>
</ContentPage>
```



# บทที่ 7

## การพัฒนาแอปแบบหลายหน้าจอ

ในการพัฒนาแอปที่มีส่วนแสดงผลตั้งแต่ 2 หน้าเป็นต้นไป ถือเป็นเรื่องพื้นฐานอีก 1 หัวข้อที่นักพัฒนาต้องทราบ เราจะมาดูกันว่า .NET MAUI มีวิธีการทำงานกับแอปที่มีหลายหน้าจออย่างไร

### พื้นฐานการเพิ่มหน้าจอใหม่เข้ามาในโปรเจกต์

เมื่อผู้อ่านสร้างโปรเจกต์ .NET MAUI ขึ้นมา ผู้อ่านจะได้หน้าหลักที่ชื่อว่า MainPage.xaml ทำหน้าที่สร้างส่วนแสดงผล และมีไฟล์ MainPage.xaml.cs ทำหน้าที่เขียนโค้ดภาษา C# ทำงานคู่กันเป็น 1 หน้าจอ อยู่ในฐานะเป็นหน้าแรกของแอป เมื่อผู้อ่านเพิ่มส่วนแสดงผลอื่นๆ เข้ามา ย่อมที่จะต้องมีการควบคุมการเปิดหน้าไป-มาระหว่างกันนั่นเอง

# การสร้างเมนูสำหรับ Desktop Apps

ระบบเมนูของ Desktop Apps กับ Mobile Apps มีความแตกต่างกันเป็นอย่างมาก เพราะว่าพื้นที่แสดงผลในหน้าจอ PC มีมากกว่าในหน้าจอ Mobile Device ในขณะที่ผู้เขียนแต่งหนังสือเล่มนี้อยู่ ระบบเมนูของ .NET MAUI รองรับเฉพาะ Desktop Apps เท่านั้น

**ตัวอย่างที่ 7-5** การสร้างเมนูสำหรับ Desktop Apps มีขั้นตอนดังนี้

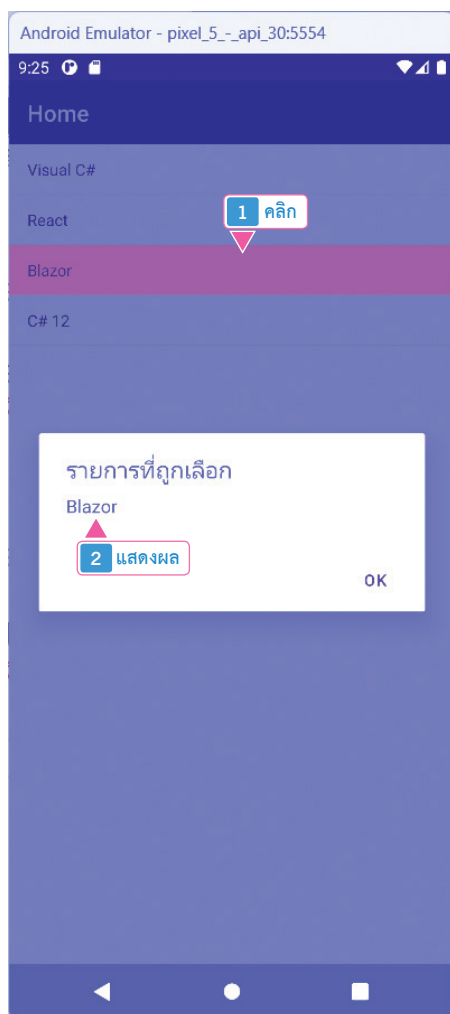
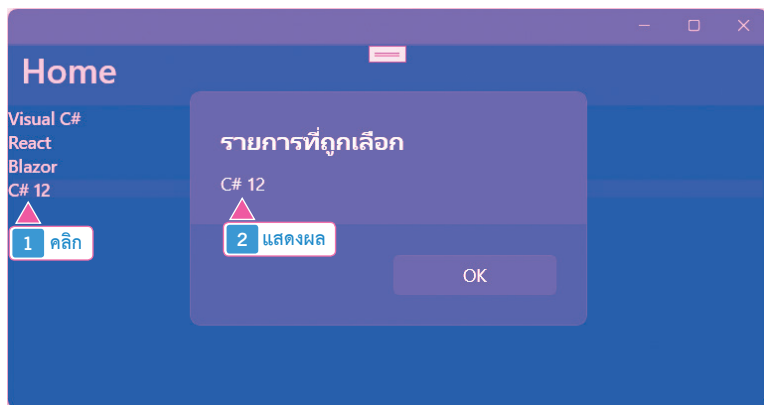
1. ที่ไฟล์ส่วนแสดงผล MainPage.xaml ต้องการสร้างเมนิวดังสคริปต์ต่อไปนี้

## สคริปต์ XAML ที่ 7-5 การสร้างเมนูสำหรับ Desktop Apps (MainPage.xaml)

```
<?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"
              x:Class="UsingMenuBarItems.MainPage">

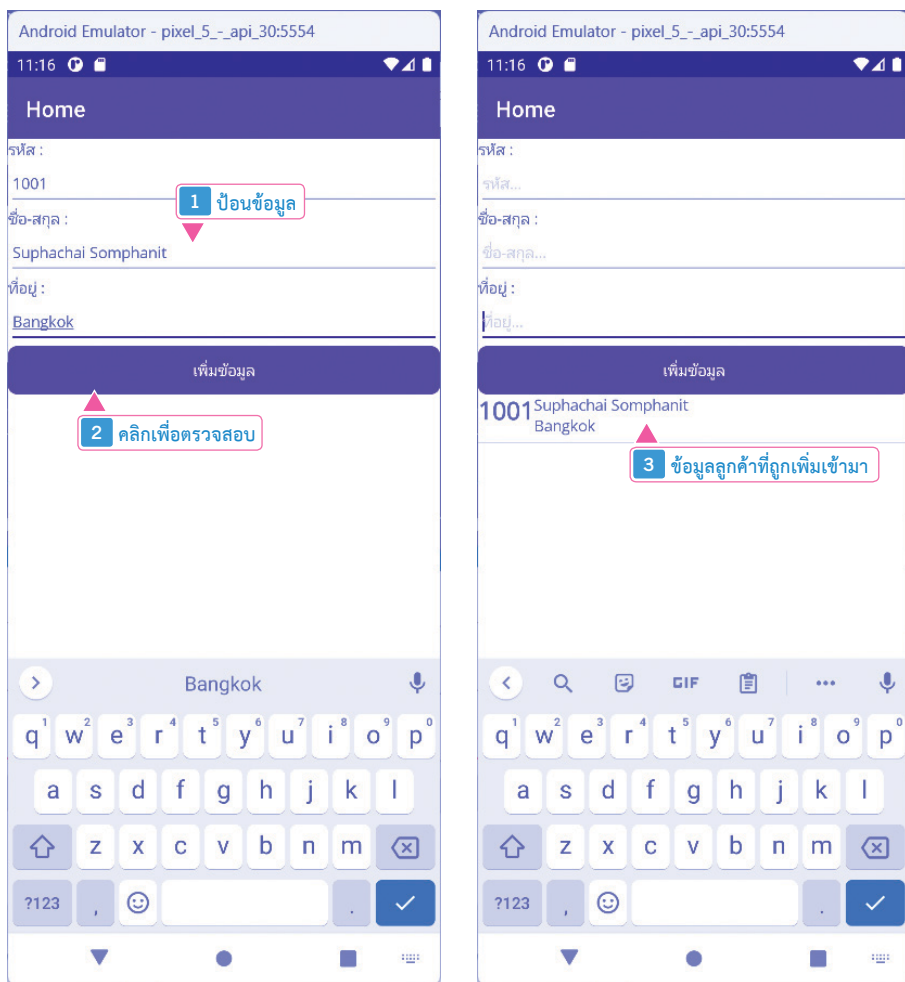
    <ContentPage.MenuBarItems>
        <MenuItem Text="File">
            <MenuFlyoutItem x:Name="mnuOpen" Text="Open..." />
        </MenuItem>
        <MenuItem Text="Edit">
            <MenuFlyoutSubItem Text="Settings">
                <MenuFlyoutItem Text="Fonts..." />
                <MenuFlyoutItem Text="System..." />
            </MenuFlyoutSubItem>
            <MenuFlyoutItem Text="About" />
        </MenuItem>
    </ContentPage.MenuBarItems>

    <StackLayout Orientation="Vertical">
        <Label x:Name="lblOutput"
              TextColor="White"
              BackgroundColor="Orange"
              FontSize="Large"/>
    </StackLayout>
</ContentPage>
```



รูปที่ 8-1 แสดงรายการ ListView บน Windows Apps และ Android Apps

7. ท้ายที่สุด ผู้อ่านสามารถเพิ่มรายชื่อลูกค้าใหม่ได้ตามที่ต้องการ ดังรูปที่ 8-10



รูปที่ 8-10 แสดงการเพิ่มข้อมูลลูกค้าใหม่





# บทที่ 9

## การทำงานกับฟังก์ชันพื้นฐานของอุปกรณ์ Mobile Devices (.NET MAUI Essentials)

ในการทำงานกับฟังก์ชันพื้นฐานต่างๆ ของอุปกรณ์ Mobile Devices ค่อนข้างสะดวกสบายเป็นอย่างมาก เพราะว่า .NET MAUI มีกลุ่มคลาสที่ทำหน้าที่ด้านนี้โดยเฉพาะ เรียกว่า **.NET MAUI Essentials**

### ฟังก์ชันการโทรออก

ฟังก์ชันการโทรออกมีเฉพาะอุปกรณ์ Mobile Device เท่านั้น คือ Android Apps กับ iOS Apps ส่วน Windows Apps ไม่มี ผู้อ่านจึงต้องแยกการทำงานของฟังก์ชันนี้ออกเป็น 2 ฝั่ง

วิธีการสั่งให้ฟังก์ชันการโทรออกทำงาน เป็นหน้าที่ของคลาสที่ชื่อว่า **PhoneDialer** ดังตัวอย่างต่อไปนี้

#### ตัวอย่างที่ 9-1 ฟังก์ชันการโทรออก

##### สคริปต์ XAML ที่ 9-1 ฟังก์ชันการโทรออก (MainPage.xaml)

```
<?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"
    x:Class="UsingMauiEssentials.MainPage">
```

# การสร้างไฟล์เก็บข้อความส่วนกลาง (Shared Preferences)

ไฟล์ข้อความ **Shared Preferences** คือ ไฟล์ที่ทำหน้าที่เก็บข้อมูลส่วนกลางไว้ใช้ประโยชน์ภายในแอปของเรา

## ตัวอย่างที่ 9-9 การสร้างไฟล์เก็บข้อความ (Shared Preferences)

ต้องการเก็บชื่อ-สกุลไว้ใน Shared Preferences ดังโค้ดต่อไปนี้

### สคริปต์ XAML ที่ 9-9 การสร้างไฟล์เก็บข้อความ (Shared Preferences) (MainPage.xaml)

```
<?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"
    x:Class="UsingMauiEssentials.MainPage">

    <StackLayout Orientation="Vertical">
        <Label Text="ชื่อ-สกุล : " />
        <Entry x:Name="etFullName" Placeholder="ป้อนชื่อ-สกุล" />
        <Button x:Name="cmdSave" Text="บันทึก" />
        <Button x:Name="cmdRemove" Text="ลบ" />
        <Label x:Name="lblOutput" Text="" />
    </StackLayout>
</ContentPage>
```

### โค้ด VC# 2022 ที่ 9-9 การสร้างไฟล์เก็บข้อความ (Shared Preferences) (MainPage.xaml.cs)

```
namespace UsingMauiEssentials;

public partial class MainPage : ContentPage
{
    public MainPage()
    {
        InitializeComponent();
        cmdSave.Clicked += CmdSave_Clicked;
        cmdRemove.Clicked += CmdRemove_Clicked;
    }
}
```



# บทที่ 12

## การสร้าง .NET MAUI Apps แบบ CRUD เชื่อมต่อกับ ASP.NET Core Web API (Front End)

หลักการพัฒนาในระบบในยุคปัจจุบันถูกแบ่งออกเป็น 2 ฝั่งอย่างชัดเจน คือ

- **งานฝั่งหน้าบ้าน (Front End)** เป็นงานที่เกี่ยวข้องกับการสร้างส่วนแสดงผลหรือหน้าจอต่างๆ ในกรณีนี้หมายถึงโปรเจกต์ประเภท .NET MAUI
- **งานฝั่งหลังบ้าน (Back End)** เป็นการสร้างบริการพื้นฐาน 4 อย่าง ประกอบด้วย การเพิ่ม (Create), การอ่าน (Read), การอัปเดต (Update) และการลบข้อมูล (Delete) ในกรณีนี้หมายถึงโปรเจกต์ประเภท ASP.NET Core Web API

การพัฒนาระบบงานในลักษณะนี้อาจจะเป็นคนเดียวทำทั้ง 2 ฝั่งก็ได้ หรือแยกกันทำก็ได้เช่นกัน ก็ต้องมีข้อตกลงกันทั้ง 2 ฝั่งว่าเราจะทำงานบนพื้นฐานของโครงสร้างข้อมูลใดหรือคลาสใด เป็นไปตามหน้าที่ของบริการนั้นๆ เนื้อหาในบทนี้จะกล่าวถึงงานฝั่งหน้าบ้าน (Front End) ก่อน เป็นหน้าที่ของโปรเจกต์ประเภท .NET MAUI

## การสร้างโปรเจกต์ .NET MAUI แบบ CRUD

**ตัวอย่างที่ 12-1** การสร้างโปรเจกต์ .NET MAUI แบบ CRUD มีขั้นตอนดังนี้

1. สร้างโปรเจกต์ .NET MAUI ตั้งชื่อว่า RestCRUD ขึ้นมาก่อน ให้ยกเลิกฟิเจอร์ Nullable ด้วย
2. ในกรณีนี้ต้องการสร้างงาน CRUD กับข้อมูลลูกค้า โดยการสร้างโฟลเดอร์ที่ชื่อว่า Models ขึ้นมาก่อน จากนั้นสร้างคลาสที่ชื่อว่า Customer ประกอบด้วย



# บทที่ 13

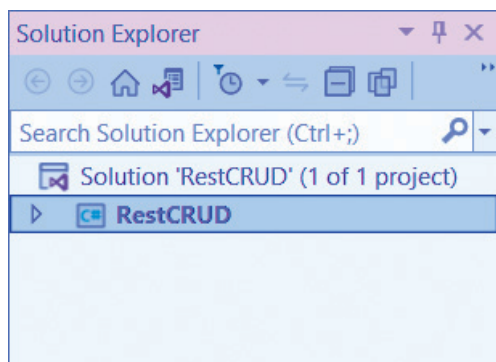
## การสร้าง .NET MAUI Apps แบบ CRUD เชื่อมต่อกับ ASP.NET Core Web API (Back End)

เนื้อหาในบทนี้เราจะมาเริ่มสร้างบริการฝั่งหลังบ้าน (Back End) ด้วยโปรเจกต์ประเภท ASP.NET Core Web API ส่งผลให้เกิดการทำงานครบถ้วนทั้ง 2 ฝั่ง

### ทำความเข้าใจกับโครงสร้างแบบหลายโปรเจกต์

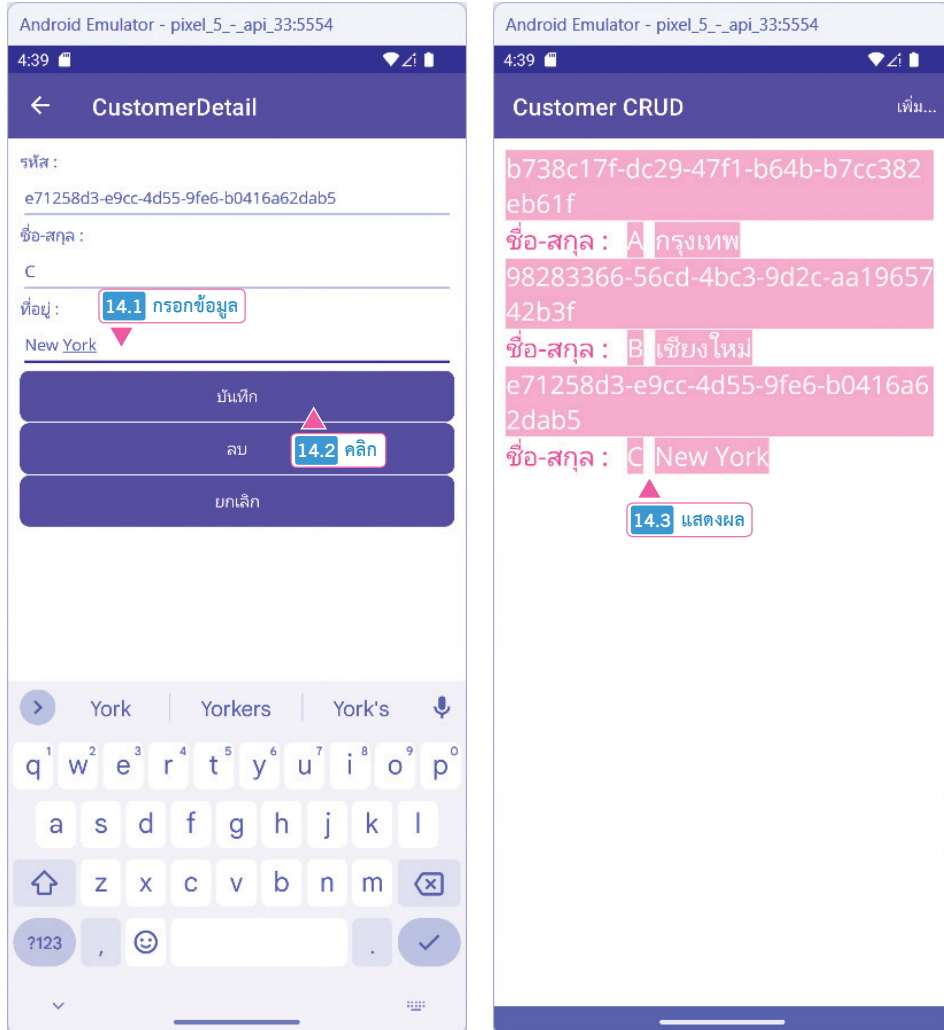
โดยปกติแล้วเมื่อผู้อ่านสร้างโปรเจกต์ขึ้นมา เราจะได้โครงสร้างโปรเจกต์แบบ 1 Solution แบบโปรเจกต์เดียว เช่น สร้างโปรเจกต์ที่ชื่อว่า RestCRUD ขึ้นมา สิ่งที่เราได้รับคือ

1. Solution ที่มีชื่อเหมือนกับโปรเจกต์ ในกรณีนี้ชื่อว่า RestCRUD
2. ใน Solution ที่เราทำงานอยู่ มี 1 โปรเจกต์ชื่อว่า RestCRUD



รูปที่ 13-1 แสดงโครงสร้างแบบ 1 Solution แบบโปรเจกต์เดียว

#### 14. ท้ายที่สุด ทดสอบเพิ่มข้อมูลลูกค้าใหม่แบบ Android Apps บ้าง ดังรูปที่ 13-26



รูปที่ 13-26 กรณีสอบเพิ่มข้อมูลลูกค้าใหม่แบบ Android Apps

#### สรุปท้ายบท

เนื้อหาของบทที่แล้วและบทนี้เป็นการพัฒนาแอปครบวงจรทั้งฝั่งหน้าบ้าน (Front End) และฝั่งหลังบ้าน (Back End) ประกอบด้วยการทำงานพื้นฐาน 4 อย่าง คือ CRUD ถือเป็นรากฐานที่จะต่อยอดในระดับถัดไป



## บทที่ 18

# การอัปโหลดแอปขึ้น Google Play Store

หลังจากที่เราพัฒนา Android Apps จนถึงขั้นตอนที่แอปของเราพร้อมใช้งาน ก็จะเข้าสู่ขั้นตอนสุดท้าย นั่นคือ การอัปโหลดแอปของเราขึ้น Google Play Store เพื่อเผยแพร่แอปของเราให้คนอื่นโหลดใช้งาน ก็คือเนื้อหาที่จะนำเสนอในบทนี้

ขั้นตอนการนำ Signed AAB ขึ้น Play Store มีการอัปเดตเปลี่ยนแปลงอยู่เสมอ ส่งผลให้ขั้นตอนต่างๆ ที่นำเสนอในบทนี้อาจจะตรงหรือไม่ตรงกับปัจจุบัน ขอให้ผู้อ่านยึดถือขั้นตอนปัจจุบันเป็นหลัก โดยที่ผู้เขียนมีข้อแนะนำในขั้นต้นอยู่ 3 อย่าง คือ

1. เตรียมไฟล์ Signed AAB ให้พร้อมก่อน โดยที่ชื่อ Package ต้องไม่มีคำว่า com.example
2. ข้อมูลที่เป็นข้อความธรรมดา ผู้อ่านต้องป้อนในช่องที่มีเครื่องหมาย \* ให้ครบตามที่ผู้อ่านเห็น
3. ไฟล์รูปภาพ ไฟล์รูปภาพแต่ละขนาดจะถูกนำไปแสดงใน Play Store ทำตามความต้องการของ Play Store ให้ครบถ้วน รวมถึงรายการอื่นๆ (ถ้ามี)

## การสมัครบัญชีนักพัฒนา Android Apps

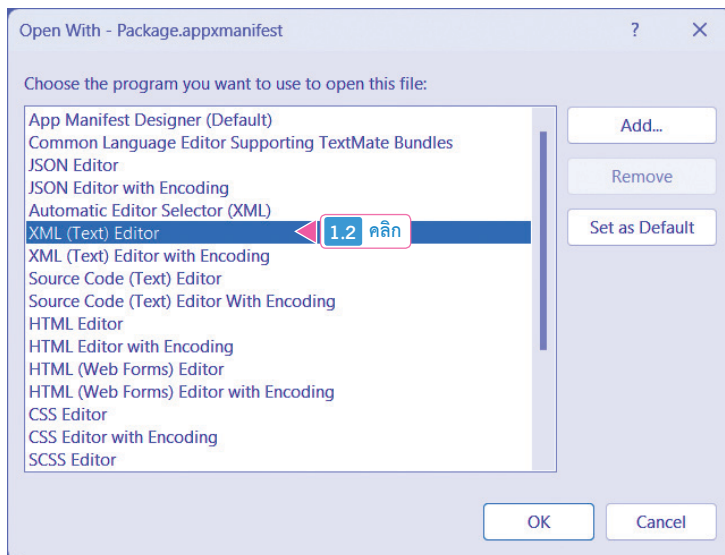
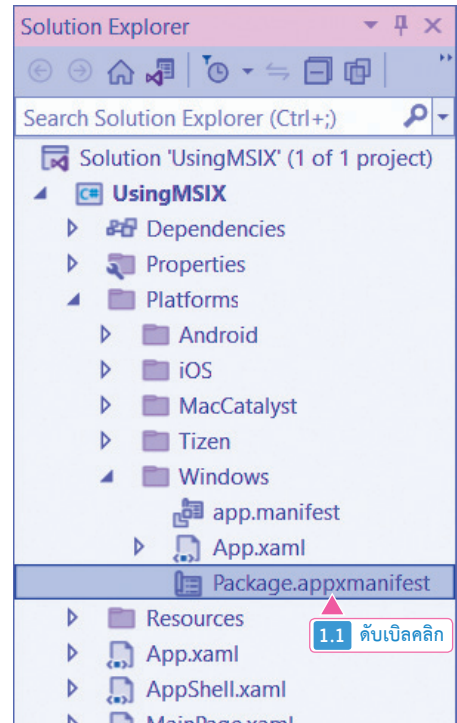
1. ผู้อ่านต้องมีอีเมลของ Gmail ก่อน สมัครฟรีได้ที่เว็บ [www.gmail.com](http://www.gmail.com) เพื่อให้ได้บัญชี Google Account มาอีกอย่างหนึ่ง
2. ให้ผู้อ่านไปที่เว็บ <https://play.google.com/apps/publish/signup/> เพื่อสมัครบัญชีนักพัฒนาของ Google เสียค่าใช้จ่ายครั้งเดียวคือ 25 เหรียญ (ค่าใช้จ่ายอาจจะเท่าหรือไม่เท่าก็ได้ ตรวจสอบตามช่วงเวลาของผู้อ่านเป็นหลัก)

# การสร้างไฟล์ Signed MSIX สำหรับ Windows Apps

วิธีการติดตั้งโปรแกรมต่างๆ ใน Windows ที่ผู้ใช้งานคุ้นเคยกันเป็นอย่างดีก็คือ การดับเบิลคลิกไฟล์ .exe หรือ .msi แล้วก็กดปุ่ม Next ไปเรื่อยๆ จนสิ้นสุดการติดตั้งโปรแกรมที่ปุ่ม Finish

การติดตั้งแอปสมัยใหม่ในยุค Windows 10/11 มีอีกรูปแบบหนึ่งเรียกว่า MSIX มีขั้นตอนดังนี้

1. ที่โฟลเดอร์ \Platforms\Windows ให้เปิดไฟล์ Package.appxmanifest ในรูปแบบ XML ก่อน ดังรูปที่ 19-6



รูปที่ 19-6 แสดงการเปิดไฟล์ Package.appxmanifest ในรูปแบบ XML