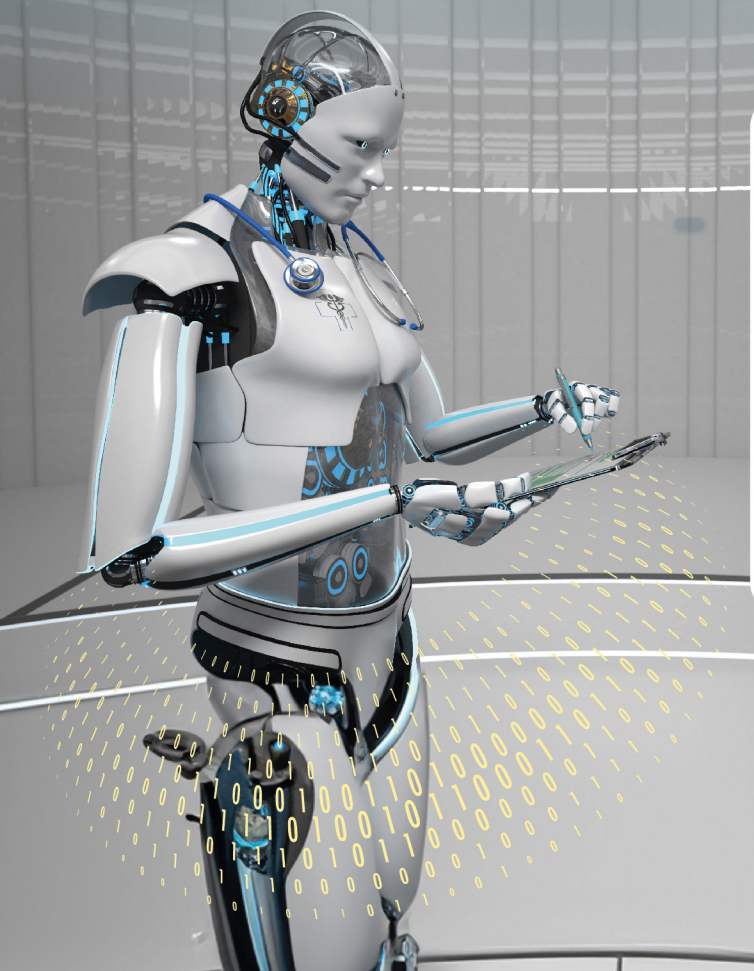
คู่มือ
พัฒนาแอปพลิเคชันแบบ Multi-Platform ด้วย

.NET MAUI

Multi-platform One codebase Productive



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

ไฟล์ตัวอย่างภายในเล่ม
https://serazu.com/
9786164874732

ศุภชัย สมพานิช

สารบัญ


.NET MAUI

บทที่ 1

การพัฒนาแอปด้วย .NET MAUI

.NET MAUI เป็นความก้าวหน้าครั้งสำคัญอีกก้าวหนึ่งของแพลตฟอร์ม .NET เป็นโปรเจ็กต์ที่ช่วยให้นักพัฒนาสามารถสร้างแอปหลายแพลตฟอร์มในเวลาเดียวกันด้วยโครงสร้างโปรเจ็กต์เพียงหนึ่งเดียว ช่วยให้ลดการดูแลรักษาโปรเจ็กต์ในระยะยาวได้อีกทางหนึ่งด้วย

ผลผลิตของโปรเจ็กต์ .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 บนเครื่องคอม PC ส่งผลให้ผู้เขียนสามารถนำเสนอแอปได้ 2 ประเภท คือ Windows Apps และ Android Apps

เนื้อหาที่นำเสนอในหนังสือเล่มนี้ครอบคลุมการพัฒนาแอปในรูปแบบ Front End กับ Back End กล่าวคือ

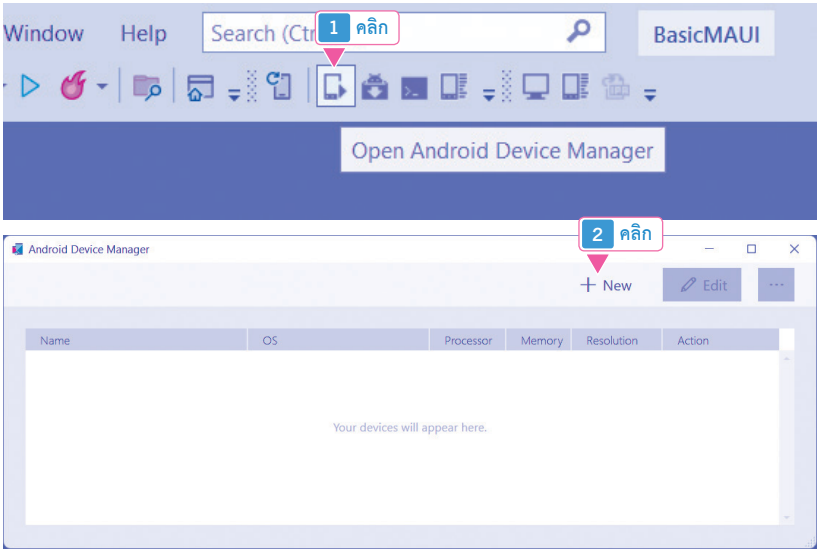
1. **งานฝั่งหน้าบ้าน (Front End)** เป็นหน้าที่ของโปรเจ็กต์ .NET MAUI ทำหน้าที่สร้างส่วนแสดงผลต่างๆ (User Interface – UI)
2. **งานหลังบ้าน (Back End)** เป็นหน้าที่ของโปรเจ็กต์ประเภท ASP.NET Core Web API ทำหน้าที่ติดต่อกับส่วนของข้อมูลเพื่อให้บริการพื้นฐานในรูปแบบ Web API ได้แก่ งานประเภทเพิ่ม, อ่าน, อัพเดต และลบข้อมูล (CRUD)

การสร้าง Android Emulator ของ Visual Studio สำหรับทดสอบ Android Apps

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

การทดสอบการทำงานของโปรเจ็กต์ 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 เพื่อทดสอบโปรเจ็กต์ .NET MAUI ประเภท Android Apps ดังรูปที่ 2-5



รูปที่ 2-5 แสดงการเริ่มต้นสร้าง Android Emulator

2. เป็นขั้นตอนการกำหนดรายละเอียดให้กับ Android Emulator เพื่อจำลองให้มีการทำงานเหมือนกับมือถือเครื่องจริง 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 ให้ผู้อ่านลองรันโปรเจ็กต์ พบว่าเราได้ช่องรับชื่อ-สกุล และที่อยู่ เมื่อคลิกที่ปุ่มบันทึกก็จะแสดงข้อความที่ผู้ใช้งานป้อนเข้ามา 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

พื้นฐานการเพิ่มหน้าจอใหม่เข้ามาในโปรเจ็กต์

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

การสร้างเมนูสำหรับ Desktop Apps

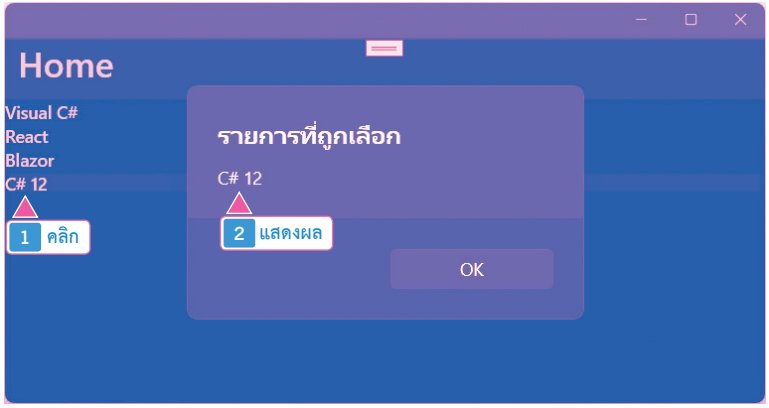
ระบบเมนูของ Desktop Apps กับ Mobile Apps มีความแตกต่างกันเป็นอย่างมาก เพราะว่าพื้นที่แสดงผลในหน้าจอ PC มีมากกว่าในหน้าจอ Mobile Device ในขณะที่ผู้เขียนแต่งหนังสือเล่มนี้อยู่ ระบบเมนูของ .NET MAUI รองรับเฉพาะ Desktop Apps เท่านั้น

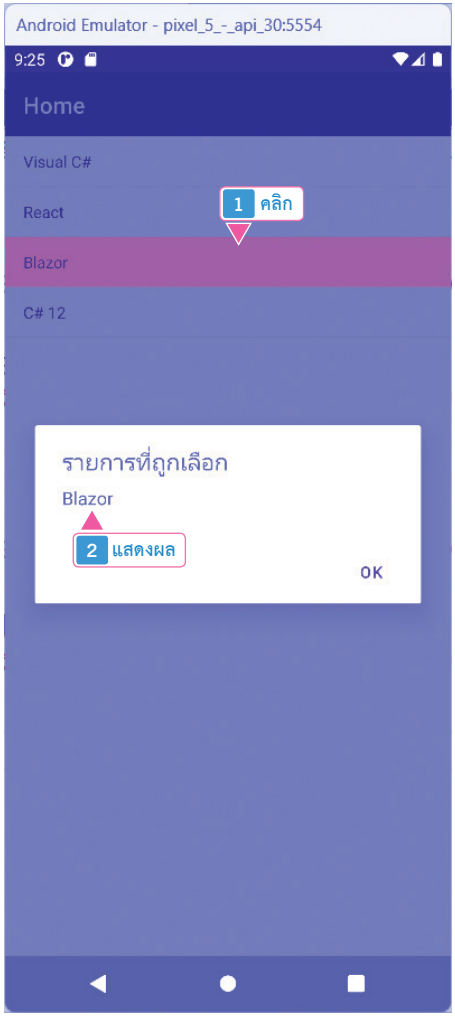
ตัวอย่างที่ 7-5 การสร้างเมนูสำหรับ Desktop Apps มีขั้นตอนดังนี้

1. ที่ไฟล์ส่วนแสดงผล MainPage.xaml ต้องการสร้างเมนูดังสคริปต์ต่อไปนี้

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

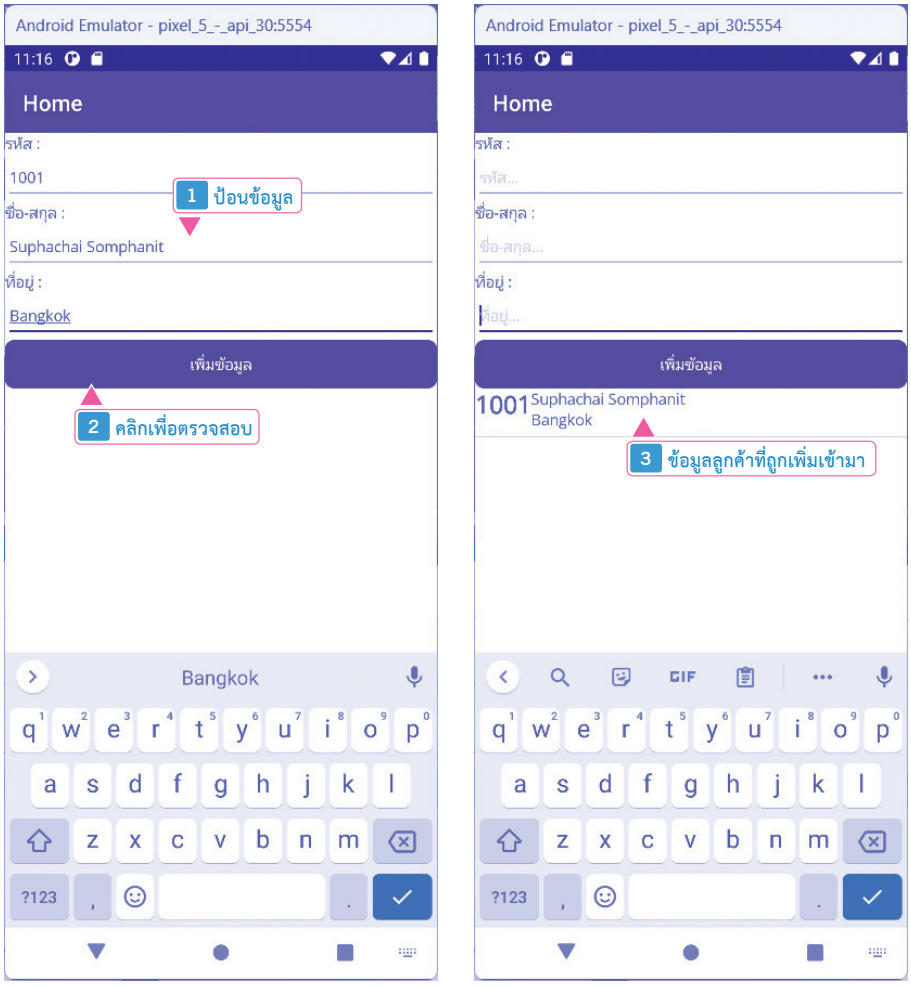
```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="UsingMenuBarItems.MainPage">

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

หลักการพัฒนาระบบในยุคปัจจุบันถูกแบ่งออกเป็น 2 ฝั่งอย่างชัดเจน คือ

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

การพัฒนาระบบงานในลักษณะนี้อาจจะเป็นคนเดียวทำทั้ง 2 ฝั่งก็ได้ หรือแยกกันทำก็ได้ เช่นกัน ก็ต้องมีข้อตกลงกันทั้ง 2 ฝั่งว่าเราจะทำงานบนพื้นฐานของโครงสร้างข้อมูลใดหรือคลาสใด เป็นไปตามหน้าที่ของบริการนั้นๆ เนื้อหาในบทนี้จะกล่าวถึงงานฝั่งหน้าบ้าน (Front End) ก่อน เป็นหน้าที่ของโปรเจ็กต์ประเภท .NET MAUI

การสร้างโปรเจ็กต์ .NET MAUI แบบ CRUD

ตัวอย่างที่ 12-1 การสร้างโปรเจ็กต์ .NET MAUI แบบ CRUD มีขั้นตอนดังนี้

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

บทที่ 13

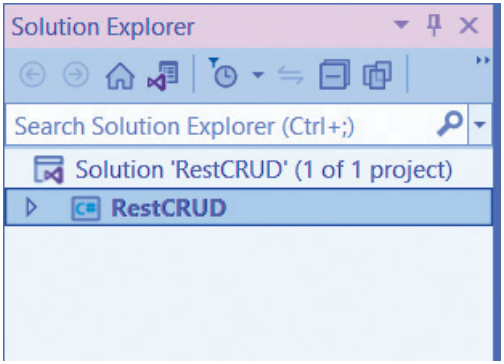
การสร้าง .NET MAUI Apps แบบ CRUD เชื่อมต่อกับ ASP.NET Core Web API (Back End)

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

ทำความรู้จักกับโครงสร้างแบบหลายโปรเจ็กต์

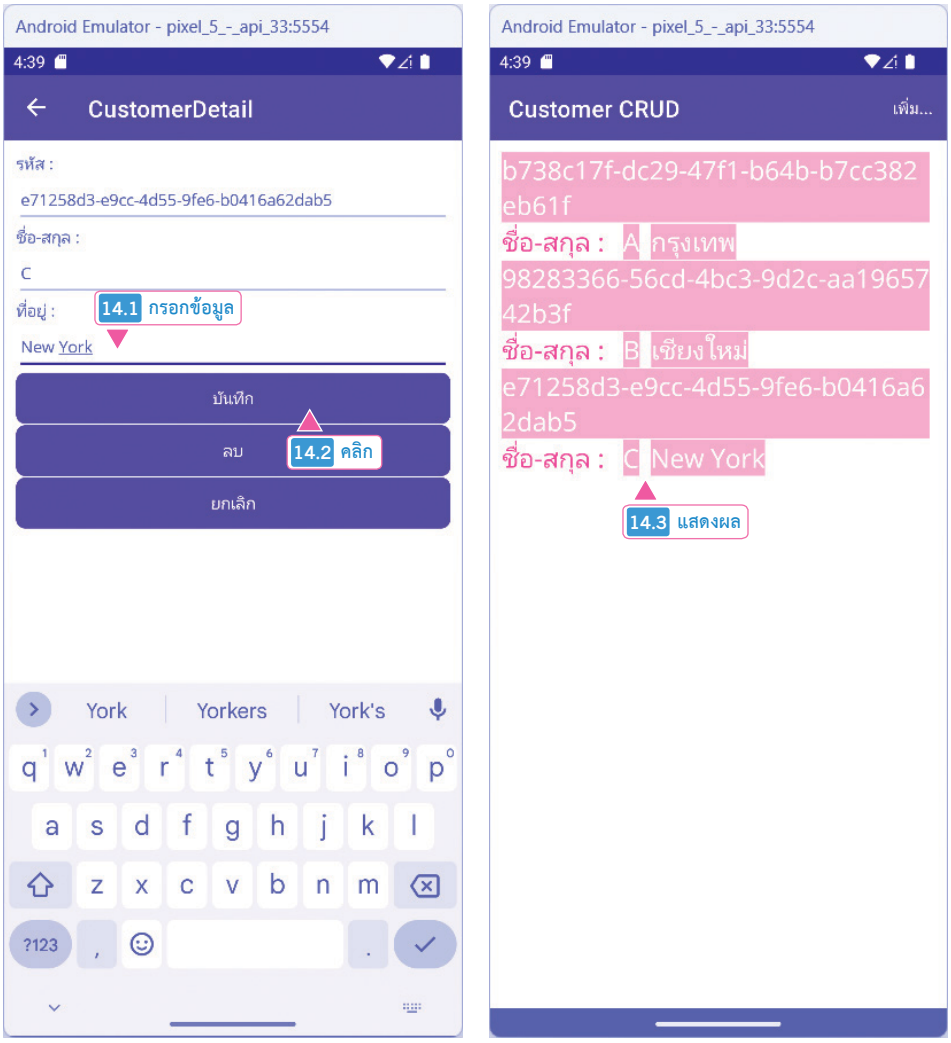
โดยปกติแล้วเมื่อผู้อ่านสร้างโปรเจ็กต์ขึ้นมา เราจะได้โครงสร้างโปรเจ็กต์แบบ 1 Solution แบบโปรเจ็กต์เดียว เช่น สร้างโปรเจ็กต์ที่ชื่อว่า RestCRUD ขึ้นมา สิ่งที่เราได้รับคือ

1. Solution ที่มีชื่อเหมือนกับโปรเจ็กต์ ในกรณีนี้ชื่อว่า RestCRUD
2. ใน Solution ที่เราทำงานอยู่ มี 1 โปรเจ็กต์ชื่อว่า RestCRUD



รูปที่ 13-1 แสดงโครงสร้างแบบ 1 Solution แบบโปรเจ็กต์เดียว

14. ท้ายที่สุด ทดสอบเพิ่มข้อมูลลูกค้าใหม่แบบ Android Apps บ้าง ดังรูปที่ 13-26



รูปที่ 13-26 กรณีเพิ่มข้อมูลลูกค้าใหม่แบบ Android Apps

สรุปท้ายบท

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

บทที่ 18

การอัพโหลดแอปขึ้น Google Play Store

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

ขั้นตอนการนำ Signed AAB ขึ้น Play Store มีการอัพเดตเปลี่ยนแปลงอยู่เสมอ 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

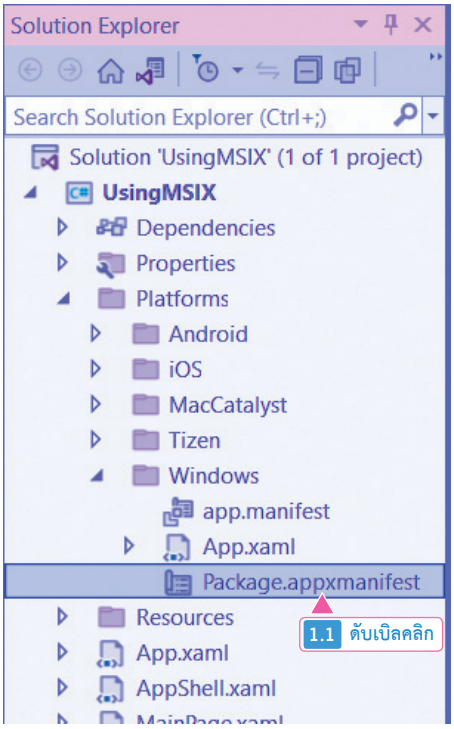
1. ผู้อ่านต้องมีอีเมลของ Gmail ก่อน สมัครฟรีได้ที่เว็บ www.gmail.com เพื่อให้ได้บัญชี Google Account มาอีกอย่างหนึ่ง
2. ให้ผู้อ่านไปที่เว็บ https://play.google.com/apps/publish/signup/ เพื่อสมัครบัญชีนักพัฒนาของ Google เสียค่าใช้จ่ายครั้งเดียวคือ 25 เหรียญ (ค่าใช้จ่ายอาจจะเท่าหรือไม่เท่าก็ได้ ตรวจสอบตามช่วงเวลาของผู้อ่านเป็นหลัก)

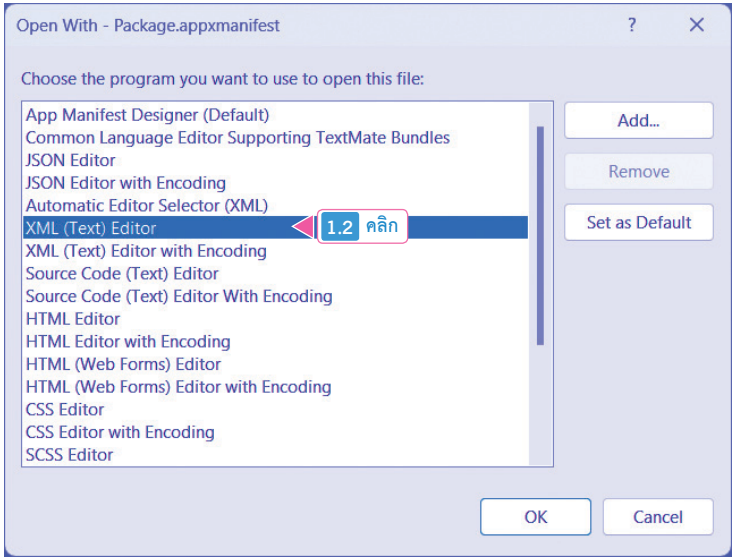
การสร้างไฟล์ Signed MSIX สำหรับ Windows Apps

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