INFOPRESS

คู่มือ Coding และพัฒนาแอปพลิเคชันด้วย C#

ฉบับสมบูรณ์

Desktop Web Cloud Mobile Games IoT AI

- เรียนรู้พื้นฐานและเทคนิคการพัฒนาแอปพลิเคชันด้วยภาษา C#
- อธิบายตั้งแต่พื้นฐานจนสามารถนำไปพัฒนาแอปพลิเคชันใช้งานได้จริง
- มีโค้ดตัวอย่างและคำอธิบายอย่างละเอียด อ่านเข้าใจง่าย
- อธิบายเป็นขั้นตอน เหมาะสำหรับนักเรียน นักศึกษา และผู้เริ่มต้น

ไฟล์ตัวอย่างภายในเล่ม
https://serazu.com/9786164874848

ศุภชัย สมพานิช

IDC PREMIER

มีเพียง “**ความรู้**” เท่านั้นที่มนุษย์ใช้พลิก “**โลก**”
และเปลี่ยน “**ชีวิต**” เราจึงสร้างสรรค์ และส่งมอบ “**ความรู้**”
ในรูปแบบที่ดีกว่า เพื่อให้คนไทย “**เรียนรู้**” ได้ตลอดชีวิต

Only “**Knowledge**” can help human
change “**The World**” and “**Their Lives**”.
With this truth, it drives us to deliver
“**Knowledge**” for Thai being able to
“**Learn**” better everyday.

Think Beyond
INFOPRESS
Read comics
Little HEART

คู่มือ Coding และพัฒนาแอปพลิเคชันด้วย C# ฉบับสมบูรณ์

Author
ศุภชัย สมพานิช
Editorial
กิตินันท์ พลสวัสดิ์ kitinan_p@idcpremier.com
Production Manager
วรพล ณธิกุล
Graphic Designer
ชวนันท์ รัตนะ
Page Layout
ศรัณย์ คมขำ
Proofreader
สุนทรี บรรลือศักดิ์
Publishing Coordinators
สุรีย์รัตน์ จิอู๋, ฉัตรดนัย ศรีสวัสดิ์
Product Specialist
ระพีพัฒน์ โทจำปา

PUBLISHED AND DISTRIBUTED BY

บริษัท ไอดีซี พรีเมียร์ จำกัด
200 หมู่ 4 ชั้น 19 ห้อง 1901 อาคารจัสมินอินเตอร์เนชั่นแนล ทาวเวอร์ ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี 11120
โทรศัพท์ 0-2962-1081 (อัตโนมัติ 10 คู่สาย)
โทรสาร 0-2962-1084
สมาชิกสัมพันธ์
โทรศัพท์ 0-2962-1081-3 ต่อ 121
โทรสาร 0-2962-1084
ร้านค้าและตัวแทนจำหน่าย
โทรศัพท์ 0-2962-1081-3 ต่อ 112-114
โทรสาร 0-2962-1084

สร้างสรรค์โดย

พิมพ์ครั้งที่ 1 ตุลาคม 2566

ข้อมูลทางบรรณานุกรม
ศุภชัย สมพานิช
คู่มือ Coding และพัฒนาแอปพลิเคชันด้วย C# ฉบับสมบูรณ์
นนทบุรี : ไอดีซีฯ, 2566
472 หน้า
1. การเขียนโปรแกรมโดยใช้ภาษาโปรแกรมเฉพาะชนิด I ชื่อเรื่อง
005.262
Barcode 885-916-101-059-3

ราคา 480 บาท

พิมพ์ที่ **บริษัท ดี.เค.ปริ้นติ้ง จำกัด**
441/56 หมู่ 2 ต.บางบ่อ อ.บางบ่อ จ.สมุทรปราการ 10560
โทรศัพท์ 0-2115-9105 โทรสาร 0-2115-9044

คำนำ

หนังสือเล่มนี้นำเสนอพื้นฐานการเขียนโปรแกรมด้วยภาษา C# เป็นภาษาหลักของแพลตฟอร์ม .NET ของไมโครซอฟท์ คุณผู้อ่านที่เป็นมือใหม่ สามารถเริ่มต้นศึกษาการเขียนโปรแกรมด้วยภาษา C# เป็นภาษาแรกได้อีกด้วย เพราะว่าตัวภาษา C# ค่อนข้างง่าย ไวยากรณ์ไม่ซับซ้อนแต่อย่างใด สามารถนำไปพัฒนาแอปครบถ้วนตั้งแต่ Desktop Apps, Web Apps และ Mobile Apps ในรูปแบบโปรเจ็กต์ต่างๆ ที่ .NET รองรับ

คุณผู้อ่านท่านใดที่ติดปัญหาหรือต้องการสอบถาม, ให้ข้อแนะนำ, ติติง รวมถึงข้อผิดพลาดต่างๆ ที่อาจจะเกิดขึ้นได้ สามารถติดต่อกับผู้เขียนได้ 5 ช่องทางคือ

1. แฟนเพจใน Facebook ชื่อว่า Thaivb.NET (https://www.facebook.com/thaivb.net)
2. กลุ่มพูดคุยเรื่องเขียนโปรแกรมใน Facebook ชื่อว่า Thaivb.NET (https://www.facebook.com/groups/Thaivb.NET)
3. ช่องใน YouTube ชื่อว่า Thaivb.NET (https://www.youtube.com/@thaivb)
4. กลุ่มใน LINE ชื่อว่า Thaivb.NET
5. แฟนเพจใน Blockdit ชื่อว่า Thaivb.NET (https://www.blockdit.com/thaivb.net)

ศุภชัย สมพานิช

ตุลาคม 2023


C#11

สารบัญ

บทที่ 1

ก้าวแรกก่อนเข้าสู่โลกของ .NET

แพลตฟอร์ม .NET (.NET Platform) ยังคงได้รับความนิยมต่อเนื่องมาอย่างยาวนานจนถึงยุคปัจจุบัน หนังสือเล่มนี้นำเสนอการเขียนโปรแกรมด้วยภาษา C# ถือเป็นภาษาหลักของแพลตฟอร์ม .NET เป็นอีก 1 ทางเลือกที่ผู้อ่านสามารถเริ่มต้นหัดเขียนโปรแกรมได้อีกด้วย

ทำความรู้จักกับแพลตฟอร์ม .NET (.NET Platform)

แพลตฟอร์ม .NET (.NET Platform) เป็น Framework ที่ไมโครซอฟท์สร้างขึ้นมา รองรับการเขียนโปรแกรมมากมายหลายภาษา เพื่อสร้างแอปพลิเคชันในรูปแบบพื้นฐาน 3 อย่าง ได้แก่

1. **Desktop Apps** หมายถึง แอปพลิเคชันที่รันบนระบบปฏิบัติการ Windows
2. **Web Apps** หมายถึง การสร้างเว็บแอปหรือสร้างเว็บไซต์ขึ้นมา
3. **Mobile Apps** หมายถึง การพัฒนาแอปสำหรับรันบน Android และ iOS

เราสามารถแบ่งยุคสมัยของ .NET ได้ 3 ช่วงคือ

1. **.NET Framework (เวอร์ชัน 1.0 ถึง 4.x)** เป็นยุคเริ่มต้นของแพลตฟอร์ม .NET นักพัฒนาสามารถพัฒนาโปรเจ็กต์ประเภทต่างๆ บนระบบปฏิบัติการ Windows เท่านั้น เรียกว่า ยุค .NET Framework
2. **.NET Core (เวอร์ชัน 1.0 ถึง 3.x)** ไมโครซอฟท์ปรับปรุงให้แพลตฟอร์ม .NET สามารถพัฒนาโปรเจ็กต์ในรูปแบบ Cross-Platform บน Windows, macOS และ Linux เรียกว่า ยุค .NET Core

3. **.NET (เวอร์ชัน 5.0 และเวอร์ชันใหม่กว่า)** ไมโครซอฟท์ยุบ .NET Framework และ .NET Core เข้าด้วยกัน ตั้งชื่อใหม่ว่า **.NET** ไม่มีอะไรต่อท้าย เริ่มต้นที่ .NET 5.0 และกำหนดให้มีการออก .NET เวอร์ชันใหม่ปีละ 1 รุ่นในเดือนพฤศจิกายนของทุกปี เรียกว่า ยุค .NET

ขอบเขตการนำเสนอของหนังสือเล่มนี้

เนื้อหาที่จะนำเสนอในหนังสือเล่มนี้ ครอบคลุมการพัฒนาโปรแกรมบน Desktop Apps และ Web Apps โดยอาศัยภาษา C# ในระดับเบื้องต้นเท่านั้น แต่ก็มีเนื้อหาสาระเพียงพอที่จะนำไปต่อยอดในลำดับต่อไป ประกอบด้วยเนื้อหา 5 ส่วนใหญ่ๆ คือ

1. ศึกษาไวยากรณ์ภาษา C# อัพเดตเนื้อหาเริ่มต้นที่ C# 11 ที่มากับ .NET 7 เป็นต้นไป
2. พัฒนาแอปทำงานกับระบบฐานข้อมูล SQL Server ด้วย Entity Framework Core เรียกสั้นๆ ว่า EF Core
3. แนะนำการพัฒนา Web Apps ด้วย ASP.NET Core MVC
4. แนะนำการพัฒนา Web API ด้วย ASP.NET Core Web API
5. แนะนำการพัฒนาแอปแบบ Container ของแพลตฟอร์ม .NET

ทำความรู้จักกับประเภทโปรเจ็กต์ที่น่าสนใจในขั้นต้น

โปรแกรม Visual Studio รองรับการพัฒนาแอปในแพลตฟอร์มต่างๆ มากมาย ผู้อ่านสามารถแยกติดตั้งแต่ละประเภทอย่างชัดเจน ที่น่าสนใจในขั้นต้นมีดังนี้

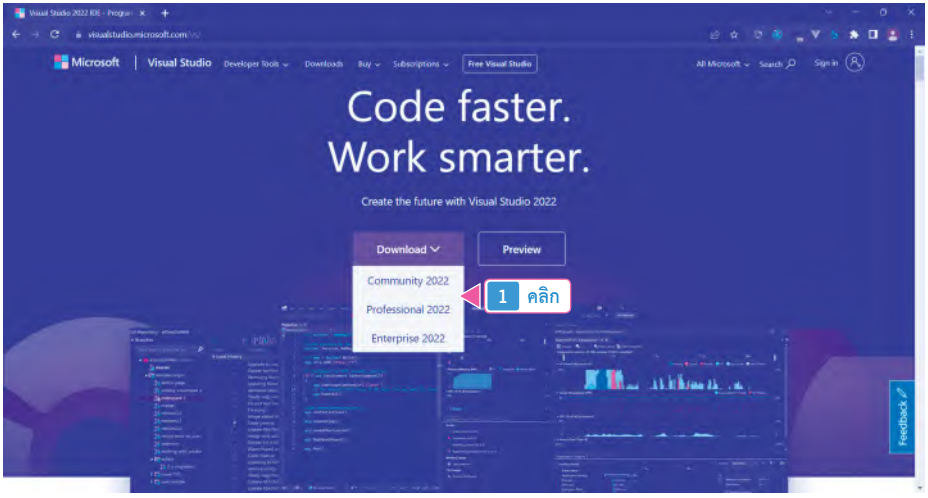
1. **ASP.NET and web development** การพัฒนา Web Apps หรือสร้างเว็บไซต์ในโลกของ .NET/.NET อยู่ในความรับผิดชอบของโปรเจ็กต์ประเภท ASP.NET มีโปรเจ็กต์ที่น่าสนใจ คือ ASP.NET CORE MVC และ ASP.NET Web API
2. **Python development** ใช้สำหรับเขียนโปรแกรมภาษาไพธอน
3. **Azure development** เป็นโปรเจ็กต์ที่รองรับการใช้บริการต่างๆ บน Cloud ของไมโครซอฟท์ เรียกว่า **Azure**
4. **Node.js development** เป็นแพลตฟอร์มที่ใช้สำหรับสร้าง Web Apps หรือสร้างเว็บไซต์โดยอาศัยภาษา JavaScript ทั้งฝั่ง Client และฝั่ง Server เพียงภาษาเดียว เรียกว่า **MEAN Stack**

5. **.NET desktop development** เป็นโปรเจ็กต์ที่ใช้สร้างแอปที่ติดตั้งบน Windows ที่เราคุ้นเคยกันเป็นอย่างดีคือ ไฟล์นามสกุล .exe หรือ .msi โดยอาศัยภาษา VB หรือ C# นั่นเอง เช่น โปรเจ็กต์ประเภท Windows Forms Application, WinUI และ WPF เป็นต้น
6. **Universal Windows Platform development** เป็นโปรเจ็กต์ที่ใช้สำหรับสร้างแอปที่ติดตั้งบน Windows 10
7. **Desktop development with C++** เป็นโปรเจ็กต์ที่ใช้สร้างแอปที่ติดตั้งบน Windows เช่นกัน แต่ใช้ภาษา C++
8. **.NET Multi-platform App UI development** การพัฒนา Mobile Apps ในโลกของ .NET เป็นหน้าที่ของโปรเจ็กต์ที่เรียกว่า .NET MAUI กับ .NET MAUI Blazor สามารถพัฒนาได้ทั้ง iOS/Android Apps และ Desktop Apps ในโปรเจ็กต์เดียวกันอีกด้วย

การดาวน์โหลดและติดตั้งโปรแกรม Visual Studio 2022

การพัฒนาโปรแกรมในโลกของ .NET มีหลายวิธี หนังสือเล่มนี้อาศัยโปรแกรมที่ชื่อว่า Visual Studio 2022 มี 2 แบบคือ

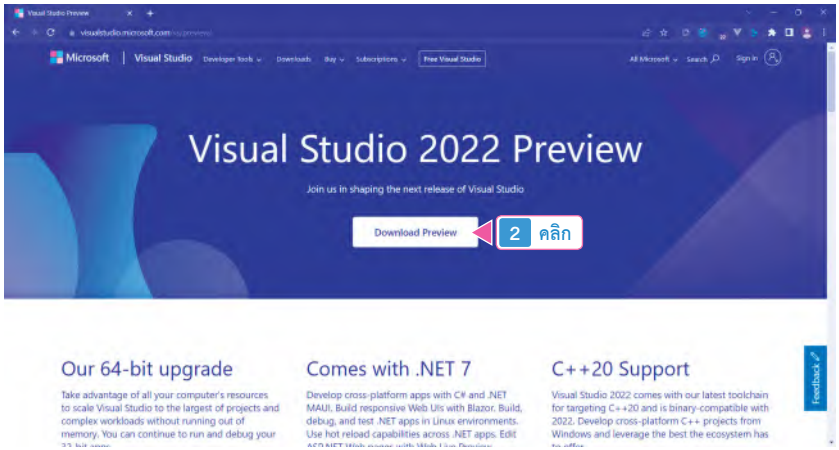
1. **โปรแกรม Visual Studio เวอร์ชันปกติ** เป็นเวอร์ชันที่เหมาะกับการนำไปใช้ในงาน Production จริง สามารถดาวน์โหลดมาใช้งานได้ฟรีที่ https://visualstudio.microsoft.com/vs/



รูปที่ 1-1 แสดงเว็บสำหรับดาวน์โหลดโปรแกรม Visual Studio เวอร์ชันปกติ

2. **โปรแกรม Visual Studio เวอร์ชัน Preview** เป็นเวอร์ชันที่เหมาะสมกับกรณีที่ต้องการทดสอบฟีเจอร์ใหม่ๆ ก่อนใคร มีความเสี่ยงกับบั๊กและ Error ต่างๆ มากกว่าเวอร์ชันปกติ ไม่แนะนำให้ใช้เวอร์ชันนี้กับงาน Production จริง ใช้แค่ในระดับศึกษาเท่านั้น สามารถโหลดมาใช้งานฟรีได้ที่ https://visualstudio.microsoft.com/vs/preview/

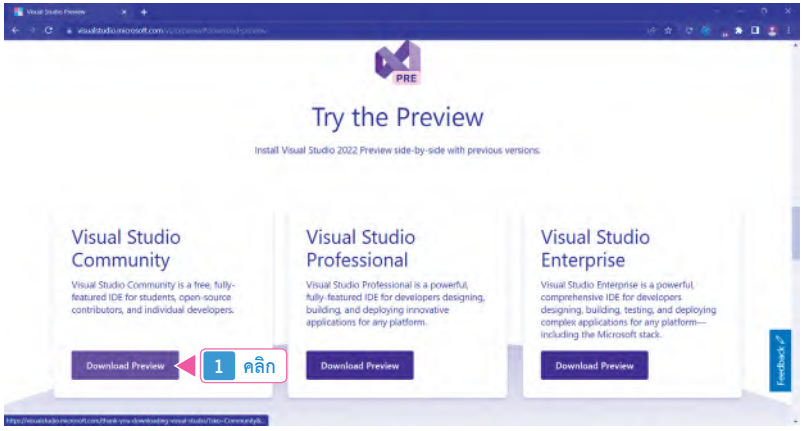
ฟีเจอร์ต่างๆ ที่ถูกนำมาทดสอบในเวอร์ชันนี้ เมื่อผ่านการทดสอบแล้วไมโครซอฟท์จะนำไปใช้งานกับเวอร์ชันปกติในลำดับถัดไป ดังรูปที่ 1-2



รูปที่ 1-2 แสดงเว็บสำหรับดาวน์โหลดโปรแกรม Visual Studio 2022 เวอร์ชัน Preview

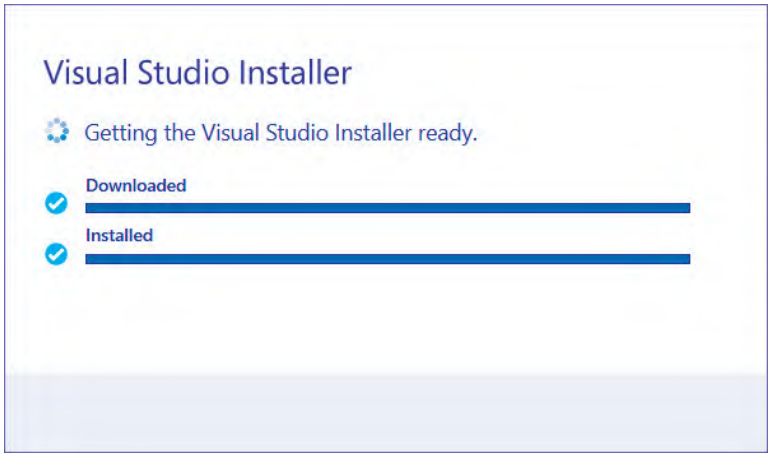
ผู้อ่านสามารถติดตั้งโปรแกรม Visual Studio 2022 ทั้ง 2 เวอร์ชันได้ในเครื่องเดียวกัน ต่างคนต่างอยู่ โดยมีวิธีการติดตั้งเหมือนกันดังนี้

1. ให้ผู้อ่านเลือกดาวน์โหลดตัวติดตั้งโปรแกรม Visual Studio 2022 Community Edition มาก่อน ดังรูปที่ 1-3



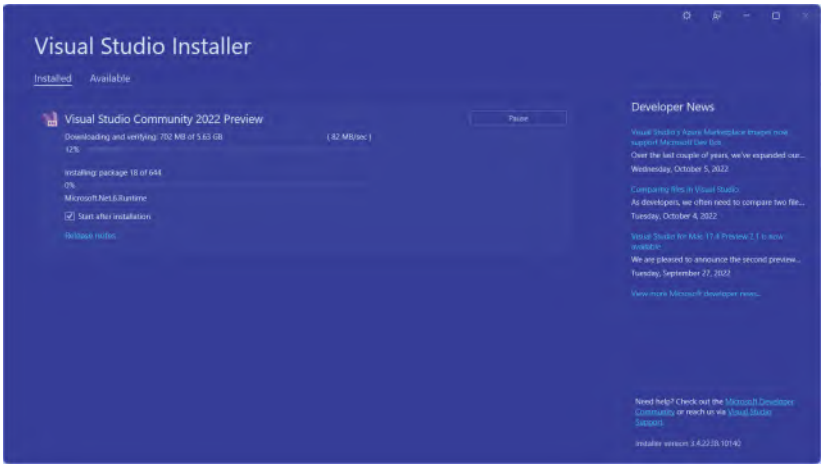
รูปที่ 1-3 แสดงการดาวน์โหลดตัวติดตั้งโปรแกรม Visual Studio 2022

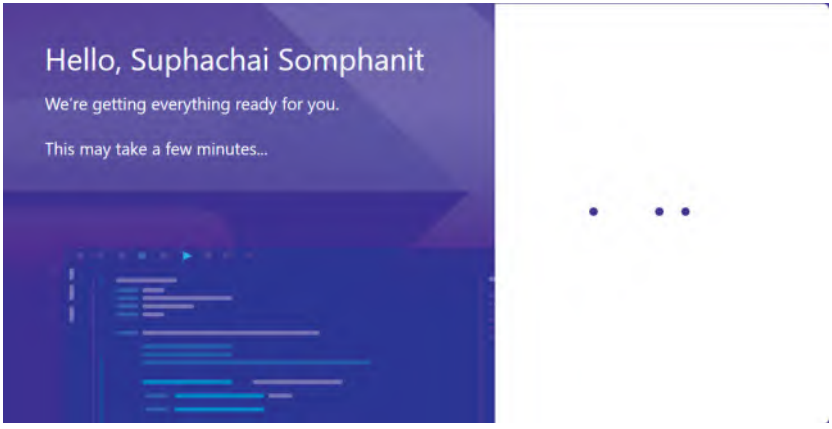
2. ให้ผู้อ่านดับเบิลคลิกไฟล์ที่ดาวน์โหลดมา เพื่อเข้าสู่ขั้นตอนการติดตั้งโปรแกรม Visual Studio 2022 ผู้อ่านสามารถเลือกประเภทโปรเจ็กต์ที่สนใจได้อย่างอิสระ ขอบเขตการนำเสนอเนื้อหาของหนังสือเล่มนี้ ประกอบด้วย
   - ASP.NET and web development สำหรับพัฒนาโปรเจ็กต์ประเภท Web Apps
   - .NET desktop development สำหรับพัฒนาโปรเจ็กต์ที่รันบนระบบปฏิบัติการ Windows

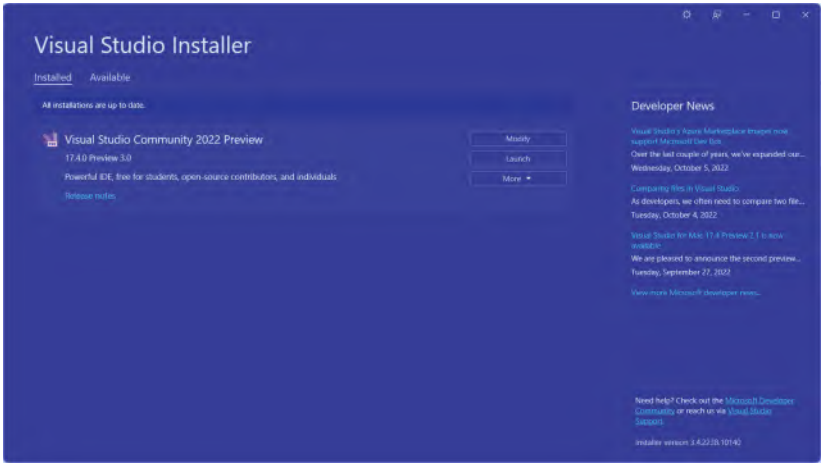


รูปที่ 1-4 แสดงหน้าจอเลือกติดตั้งประเภทโปรเจ็กต์

3. ท้ายที่สุด ให้ผู้อ่านรอการดาวน์โหลดและติดตั้งรายการต่างๆ ตามที่ผู้อ่านเลือกจนเสร็จสมบูรณ์ ถือว่าโปรแกรม Visual Studio 2022 พร้อมใช้งานแล้ว ดังรูปที่ 1-5







รูปที่ 1-5 แสดงการติดตั้งโปรแกรม Visual Studio เสร็จสมบูรณ์

การติดตั้ง .NET SDK เพิ่มเติม

เนื้อหาหนังสือเล่มนี้ต้องการ .NET SDK เวอร์ชัน 7 ขึ้นไป ผู้อ่านสามารถติดตั้ง .NET SDK หลายเวอร์ชันในเวลาเดียวกัน ส่งผลให้โปรแกรม Visual Studio 2022 ของผู้อ่านสามารถสร้างโปรเจ็กต์ได้ตามเวอร์ชัน .NET SDK ที่ถูกติดตั้งในเครื่องของคุณ มีขั้นตอนดังนี้

1. ให้ผู้อ่านไปที่เว็บไซต์ https://dotnet.microsoft.com/en-us/download/dotnet เพื่อดาวน์โหลด .NET 7 SDK เพิ่มเติม (ในช่วงเวลาของผู้อ่านจะมีเวอร์ชันใหม่กว่าเสมอ) ดังรูปที่ 1-6

รูปที่ 1-6 แสดงการดาวน์โหลด .NET 7 SDK

จากรูปที่ 1-6 ผู้เขียนใช้งาน Windows 10/11 แบบ 64 บิต จึงเลือก .NET 7 SDK แบบ x64 (หรือเลือกเวอร์ชันที่ใหม่กว่าตามที่ต้องการ) ผู้อ่านต้องตรวจสอบด้วยว่า .NET SDK ที่ดาวน์โหลดมาติดตั้งเองเพิ่มเติม ต้องการโปรแกรม Visaul Studio เวอร์ชันใด ตรวจสอบได้ที่หัวข้อ Visual Studio support

Note

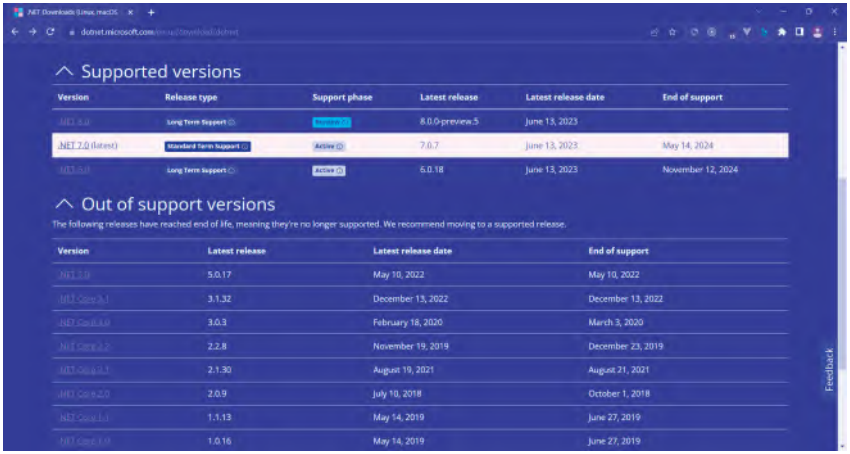
เมื่อมีการอัพเดต .NET SDK เวอร์ชันใหม่ ก็จะมีการอัพเดตโปรแกรม Visual Studio ควบคู่มาด้วยเสมอ เพื่อให้สามารถใช้งาน .NET SDK เวอร์ชันใหม่ได้นั่นเอง โดยปกติแล้วโปรแกรม Visual Studio เวอร์ชัน Preview มักจะได้สิทธิ์ทดสอบ .NET SDK เวอร์ชันใหม่ก่อนเวอร์ชันปกติ

2. ท้ายที่สุด ให้ผู้อ่านดับเบิลคลิกไฟล์ที่ดาวน์โหลดมา เพื่อเข้าสู่ขั้นตอนการติดตั้ง .NET 7 SDK ใช้ค่าเริ่มต้นที่มากับตัวติดตั้งทั้งหมด ส่งผลให้โปรแกรม Visual Studio 2022 สามารถพัฒนาโปรเจ็กต์บน .NET SDK ได้หลายเวอร์ชัน

สถานะของ .NET SDK

หลังจากที่ไมโครซอฟท์รวม .NET Framework และ .NET Core เข้าด้วยกันกลายเป็นยุค .NET ในปัจจุบัน มีนโยบายออก .NET SDK เวอร์ชันใหม่ปีละ 1 รุ่น มีสถานะแตกต่างกันกล่าวคือ

- **.NET SDK เวอร์ชันเลขคี่** ได้แก่ เวอร์ชัน 5, 7, ... มีระยะเวลาสนับสนุนอัพเดตแพตช์แก้ไข 18 เดือน
- **.NET SDK เวอร์ชันเลขคู่** ได้แก่ เวอร์ชัน 6, 8, ... มีระยะเวลาสนับสนุนอัพเดตแพตช์แก้ไข 3 ปี เป็นเวอร์ชันที่เรียกว่า Long-term support เรียกย่อๆ ว่า LTS การพัฒนาโปรเจ็กต์ในระดับ Production จริง ผู้เขียนแนะนำให้เลือกใช้เวอร์ชัน LTS เหมาะสมที่สุด



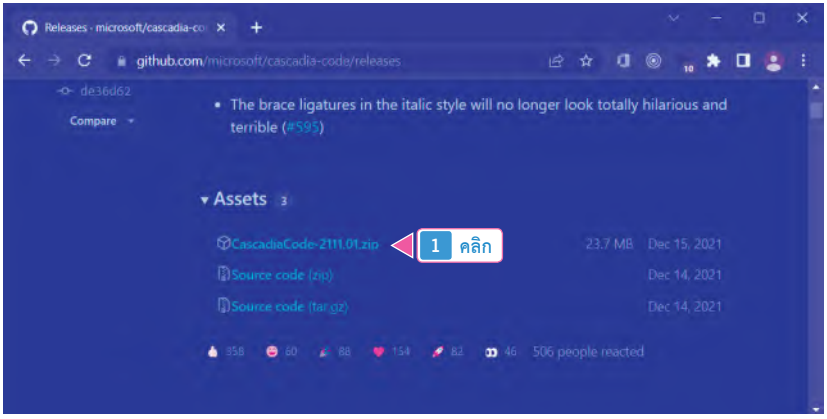
รูปที่ 1-7 แสดงรายละเอียดของ .NET SDK แต่ละรุ่น

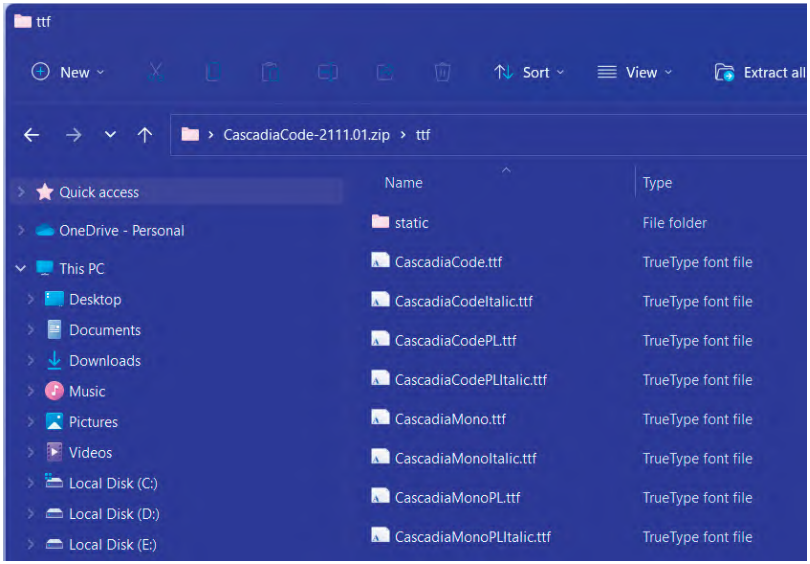
.NET SDK เวอร์ชันใดก็ตามที่ใกล้หมดระยะเวลา Support เรียกว่า ระยะ Maintenance ถ้าโปรเจ็กต์ของเราใช้เวอร์ชันที่มีสถานะนี้ ควรที่จะอัพเดตไปใช้เวอร์ชันที่ใหม่กว่าให้เร็วที่สุด

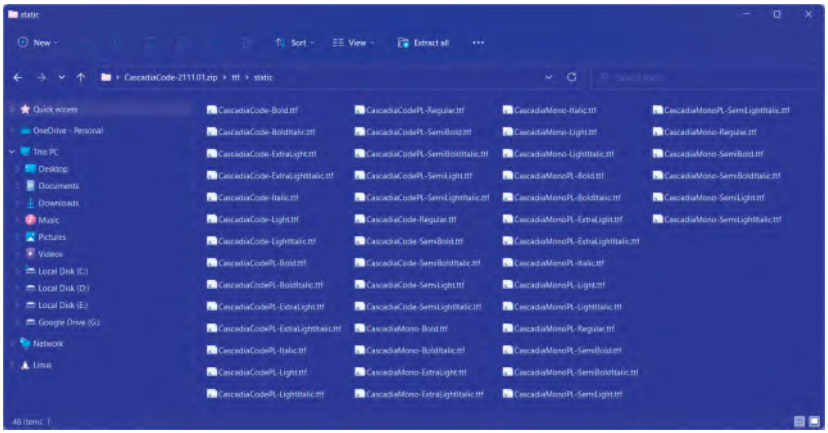
ส่วน .NET SDK เวอร์ชันเก่าที่หมดช่วงเวลาระยะเวลา Support แล้ว ก็จะถูกจัดให้อยู่ในกลุ่ม Out of support versions เป็นเวอร์ชันที่ไม่แนะนำให้ใช้งานในระดับ Production จริงแล้ว

การติดตั้งฟอนต์ Cascadia Code สำหรับเขียนโค้ดโดยเฉพาะ

ไมโครซอฟท์สร้างฟอนต์สำหรับเขียนโค้ดโดยเฉพาะขึ้นมา ชื่อว่า Cascadia Code สามารถดาวน์โหลดมาใช้งานได้ที่เว็บไซต์ https://github.com/microsoft/cascadia-code/releases ดังรูปที่ 1-8

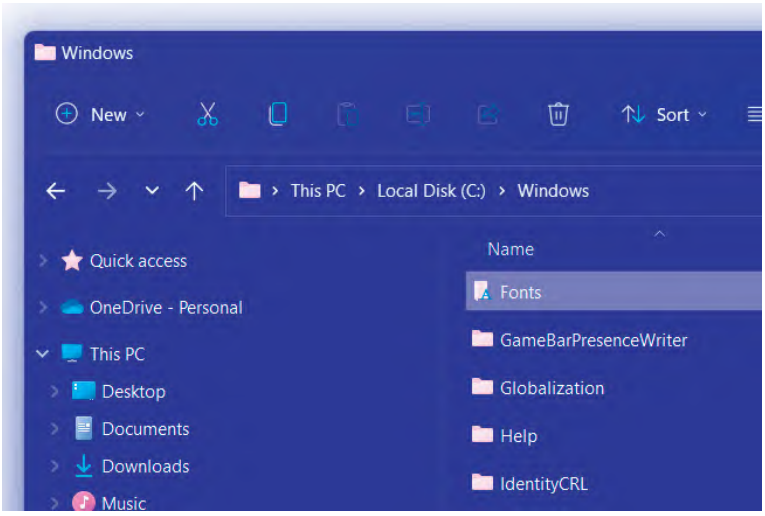






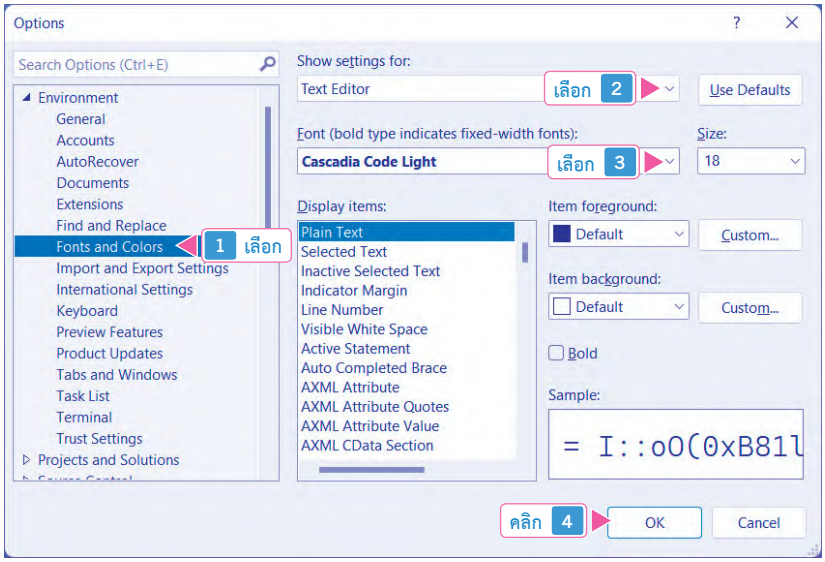
รูปที่ 1-8 แสดงเว็บไซต์ที่ให้ดาวน์โหลดฟอนต์เขียนโค้ด

จากรูปที่ 1-8 ให้แตกไฟล์ zip ที่ได้มา เราสนใจฟอนต์ที่มีนามสกุล .ttf เท่านั้น จะอยู่ในโฟลเดอร์ที่ชื่อว่า static ด้วย จากนั้นให้ผู้อ่าน Copy ฟอนต์ที่ได้มาทั้งหมดไปวางไว้ในพาธ C:\Windows\Fonts



รูปที่ 1-9 แสดงโฟลเดอร์ที่จัดเก็บรายการฟอนต์ในระบบ Windows 10/11

ให้ผู้อ่านเปิดโปรแกรม Visual Studio ขึ้นมา คลิกเมนู Tools > Options ที่หัวข้อ Environment > Fonts and Colors เลือกใช้ฟอนต์ที่ชื่อว่า Cascadia Code Light

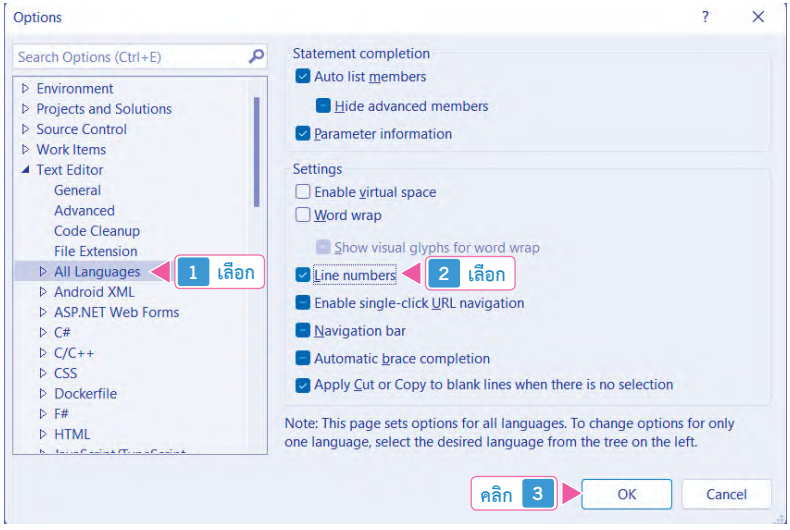


รูปที่ 1-10 แสดงการกำหนดให้โปรแกรม Visual Studio ใช้ฟอนต์ Cascadia Code Light

ข้อแตกต่างของฟอนต์ Cascadia Code กับฟอนต์อื่นๆ ที่น่าสนใจคือ สามารถแสดงสัญลักษณ์พิเศษทางภาษาได้ เช่น เครื่องหมายหัวลูกศร เกิดมาจากการพิมพ์เครื่องหมาย = ติดกับเครื่องหมายมากกว่า >

การกำหนดให้แสดงหมายเลขบรรทัด

ส่วนการกำหนดให้ VS 2022 แสดงหมายเลขกำกับโค้ดในแต่ละบรรทัด โดยการคลิกเมนู Tools > Options... ให้ผู้อ่านไปที่หัวข้อ Text Editor > All Languages คลิกเลือก Line numbers ดังรูปที่ 1-11

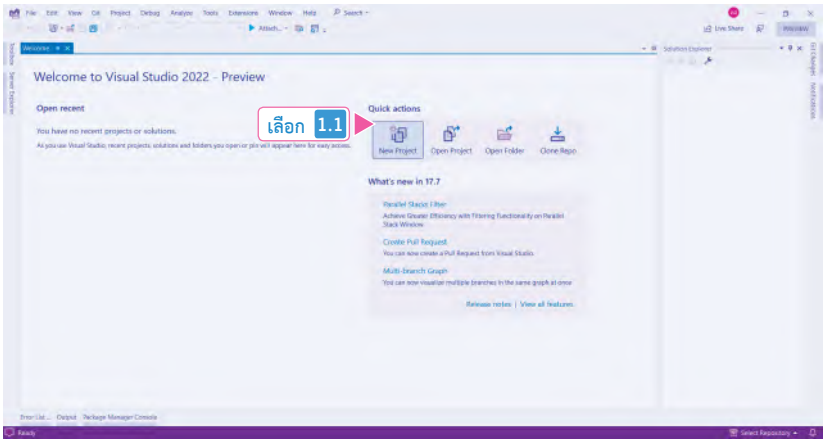


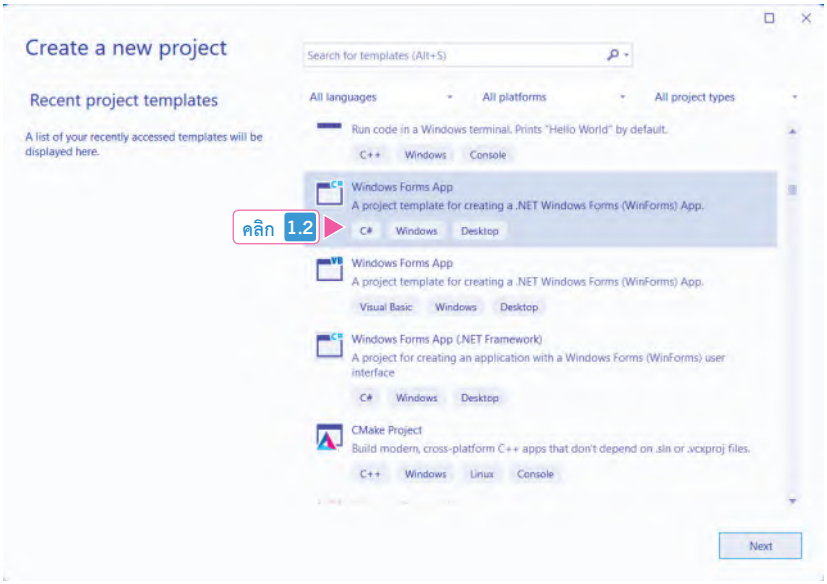
รูปที่ 1-11 แสดงการกำหนดให้แสดงหมายเลขกำกับโค้ดแต่ละบรรทัด

การสร้างโปรเจ็กต์ Windows Forms App

โปรเจ็กต์แบบ Windows Forms App ถือเป็นโปรเจ็กต์ที่เหมาะสมกับการเริ่มต้นเขียนโปรแกรมมากที่สุด มีโครงสร้างโปรเจ็กต์ค่อนข้างง่าย โดยมีขั้นตอนดังนี้

1. ที่หน้าจอเริ่มต้น ให้ผู้อ่านเลือก New Project จากนั้นเลือกสร้างโปรเจ็กต์ประเภท Windows Forms App ดังรูปที่ 1-12

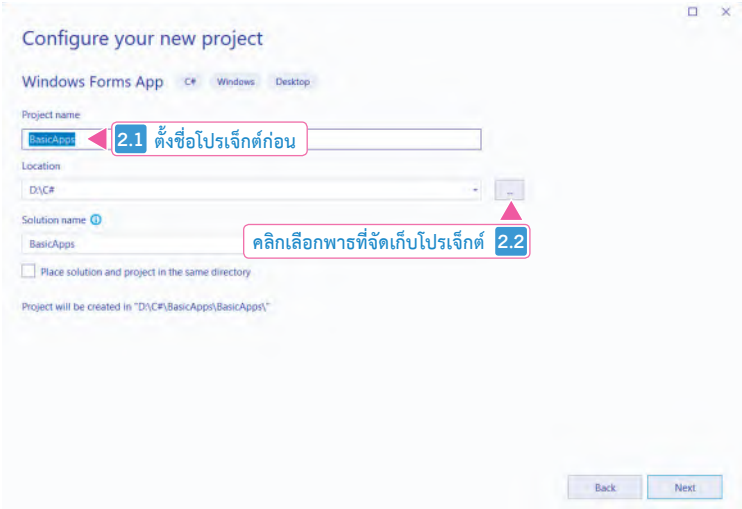




รูปที่ 1-12 แสดงการสร้างโปรเจ็กต์ Windows Forms App ของภาษา C#

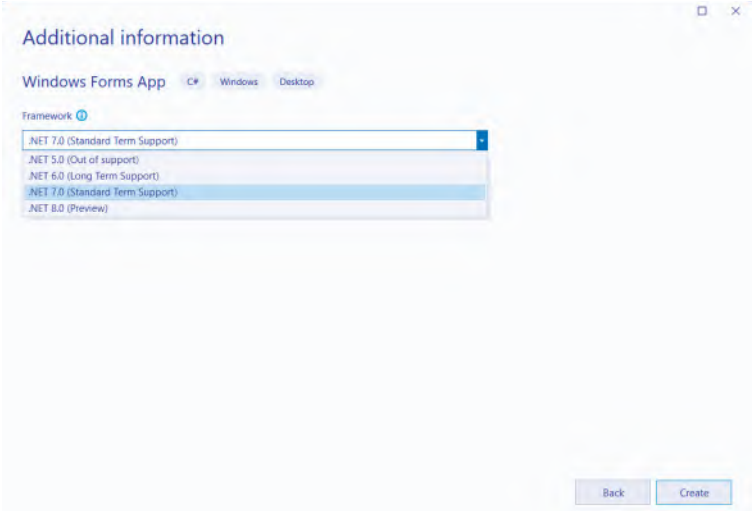
2. กำหนดรายละเอียดให้กับโปรเจ็กต์ที่จะสร้างขึ้นมา โดยที่

   - **ช่อง Project name** หมายถึง ชื่อโปรเจ็กต์ ในกรณีนี้ตั้งชื่อว่า BasicApps
   - **ช่อง Location** หมายถึง พาธที่ใช้จัดเก็บโปรเจ็กต์ปัจจุบัน โดยการคลิกปุ่ม ... เลือกโฟลเดอร์ตามที่ผู้อ่านต้องการ ในกรณีนี้เก็บโปรเจ็กต์ไว้ที่พาธ D:\C# ดังรูปที่ 1-13

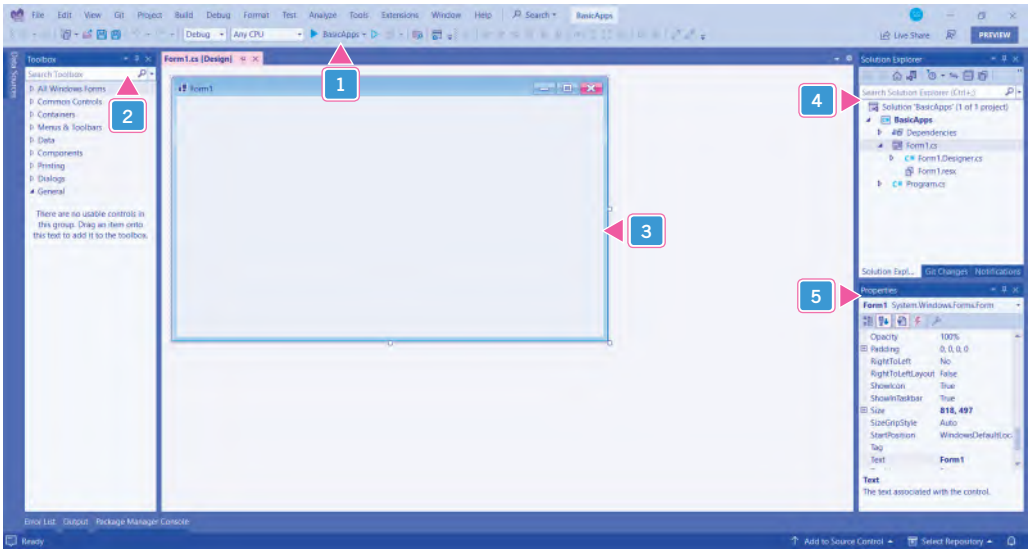


รูปที่ 1-13 แสดงการกำหนดรายละเอียดให้กับโปรเจ็กต์ที่จะสร้างขึ้นมา

3. ขั้นตอนนี้เป็นการกำหนดว่า โปรเจ็กต์ Windows Forms App ที่จะสร้างขึ้นมา ผู้อ่านต้องการใช้ .NET SDK เวอร์ชันใด รายการที่ปรากฏขึ้นมาในขั้นตอนนี้ เป็นไปตามรายการ .NET SDK ที่ผู้อ่านติดตั้งในเครื่อง เนื้อหาในหนังสือเล่มนี้ต้องการ .NET SDK 7 ขึ้นไป ดังรูปที่ 1-14



รูปที่ 1-14 การกำหนดเวอร์ชันของโปรเจ็กต์

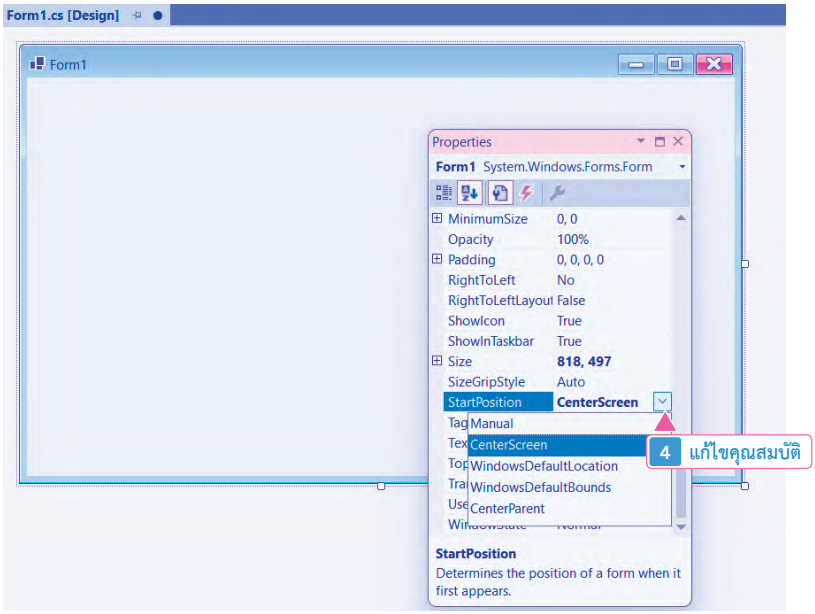


รูปที่ 1-15 แสดงสภาพแวดล้อมของ VS 2022 แบบ Windows Forms Application

จากรูปที่ 1-15 รายละเอียดแต่ละหน้าต่าง มีดังนี้

- **หมายเลข 1** แถบเครื่องมือสำหรับทดสอบโปรเจ็กต์ปัจจุบัน
- **หมายเลข 2** แถบคอนโทรลสำเร็จรูป ทำหน้าที่ออกแบบส่วนแสดงผล (User Interface เรียกย่อๆ ว่า **UI**) ภายในโปรเจ็กต์ของผู้อ่าน
- **หมายเลข 3** ส่วนแสดงผลจำลอง ผู้อ่านสามารถออกแบบส่วนแสดงผลให้กับหน้าจอต่างๆ ได้ที่นี่ โดยการลากคอนโทรลต่างๆ มาวาดตามที่ผู้อ่านต้องการ
- **หมายเลข 4** เรียกว่า หน้าต่าง **Solution Explorer** ทำหน้าที่แสดงโครงสร้างโปรเจ็กต์ปัจจุบันของผู้อ่านว่า ประกอบไปด้วยไฟล์อะไรบ้าง
- **หมายเลข 5** เรียกว่า **หน้าต่างคุณสมบัติ (Properties)** ทำหน้าที่แสดงคุณสมบัติต่างๆ ของคอนโทรลที่กำลังถูกโฟกัสในส่วนแสดงผลจำลองว่า มีคุณสมบัติอะไรบ้าง ผู้อ่านสามารถแก้ไขคุณสมบัติต่างๆ ได้ตามที่ผู้อ่านต้องการเช่นกัน เช่น ตั้งชื่อคอนโทรลได้ที่คุณสมบัติ Name, กำหนดสีพื้นหลังได้ที่คุณสมบัติ **BackColor** เป็นต้น

4. ต้องการกำหนดให้ Form1 ปรากฏอยู่กึ่งกลางหน้าจอ โดยการโฟกัสที่ Form1 ก่อน ที่คุณสมบัติ **StartPosition** ทำหน้าที่กำหนดตำแหน่งของ Form1 ผู้เขียนเลือกแบบ **CenterScreen** หมายถึง ให้ปรากฏขึ้นมากึ่งกลางหน้าจอ ดังรูปที่ 1-16

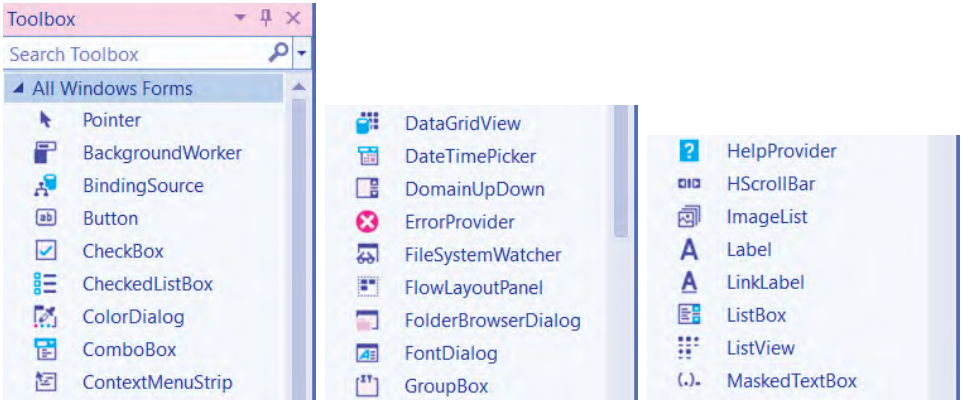


รูปที่ 1-16 แสดงการแก้ไขตำแหน่งของฟอร์มให้อยู่กึ่งกลางหน้าจอ

จากรูปที่ 1-16 ณ จุดนี้ เราได้โปรเจ็กต์ตามที่ต้องการแล้ว

การออกแบบส่วนแสดงผลด้วยคอนโทรล (User Interface - UI)

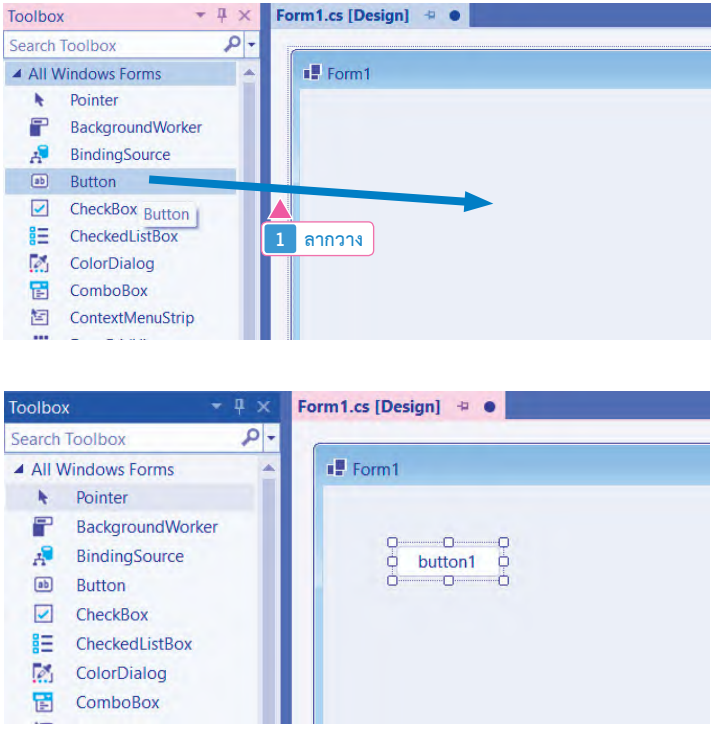
ไม่ว่าผู้อ่านจะสร้างโปรเจ็กต์ประเภทใดก็ตามใน Visual Studio ไมโครซอฟท์กำหนดให้สภาพแวดล้อมของตัวโปรแกรม Visual Studio มีลักษณะคล้ายกันหรือใกล้เคียงกันมากที่สุดเท่าที่จะเป็นไปได้ เพื่อให้นักพัฒนาคุ้นเคยกับตัวโปรแกรมได้ไม่ยาก



รูปที่ 1-17 แสดงแถบ Toolbox ของโปรเจ็กต์แบบ Windows Forms App

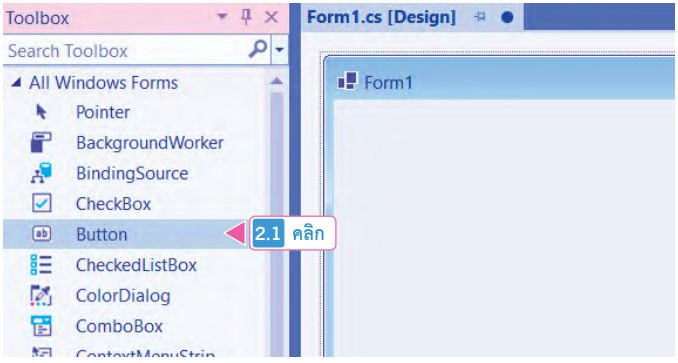
จากรูปที่ 1-17 แถบคอนโทรล (Control) อยู่ด้านซ้ายมือ ทำหน้าที่ออกแบบส่วนแสดงผล (**User Interface** เรียกย่อๆ ว่า **UI**) ในขั้นต้นนี้ ผู้อ่านสามารถใช้งานคอนโทรลนี้ได้ 2 วิธี

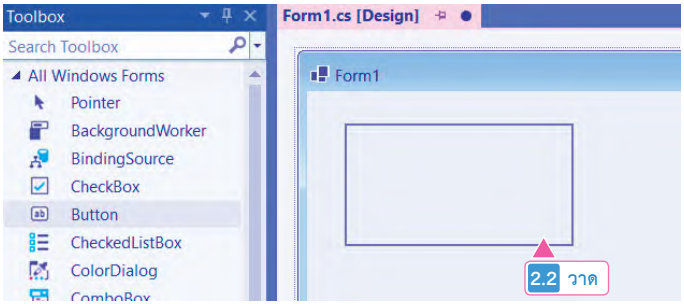
1. ดับเบิลคลิกที่ตัวคอนโทรล เพื่อกำหนดให้ Visual Studio ใช้งานคอนโทรลดังกล่าว ในหน้าจอออกแบบโดยอัตโนมัติ เช่น ผู้เขียนต้องการใช้ปุ่มกด button1 ดังรูปที่ 1-18



รูปที่ 1-18 แสดงการใช้ปุ่มกดด้วยวิธีการดับเบิลคลิกในแถบ Toolbox

2. คลิกเลือกคอนโทรลที่ต้องการใช้งาน แล้วนำไปวาดในส่วนของฟอร์ม (Form) ดังรูปที่ 1-19



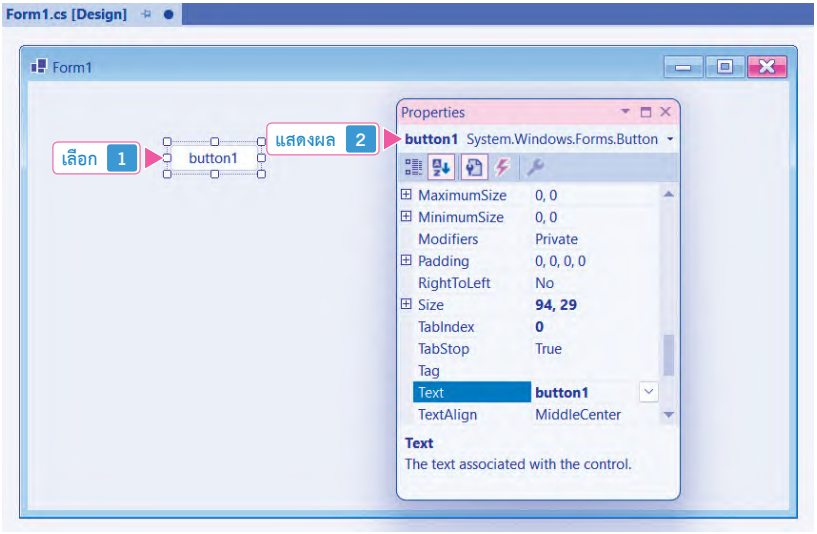


รูปที่ 1-19 แสดงการใช้ปุ่มกด Button ด้วยวิธีวาด

จากรูปที่ 1-19 เราได้ส่วนแสดงผลง่ายๆ มาแล้ว 1 หน้าจอ มีปุ่มกด Button เพียง 1 ปุ่ม ผู้อ่านสามารถออกแบบส่วนแสดงผลได้อย่างอิสระตามที่ต้องการ

การปรับแต่งคอนโทรลขั้นต้นด้วยวิธีแก้ไขคุณสมบัติ (Property)

คอนโทรลต่างๆ ที่ผู้อ่านนำมาใช้งาน ผู้อ่านสามารถปรับแต่งหรือแก้ไขได้ในหน้าต่างคุณสมบัติ (Properties) กล่าวคือ เมื่อผู้อ่านโฟกัสที่คอนโทรลตัวใดก็ตาม หน้าต่างคุณสมบัติ (Properties) จะเปลี่ยนไปโดยอัตโนมัติ



รูปที่ 1-20 แสดงรายการคุณสมบัติของปุ่มกด Button

จากรูปที่ 1-20 ส่วนแสดงผลปัจจุบันของผู้เขียนมีเพียง 1 ปุ่ม เมื่อถูกโฟกัสแล้ว หน้าต่างคุณสมบัติก็จะแสดงรายการคุณสมบัติต่างๆ ของคอนโทรล Button ให้เราโดยอัตโนมัติ

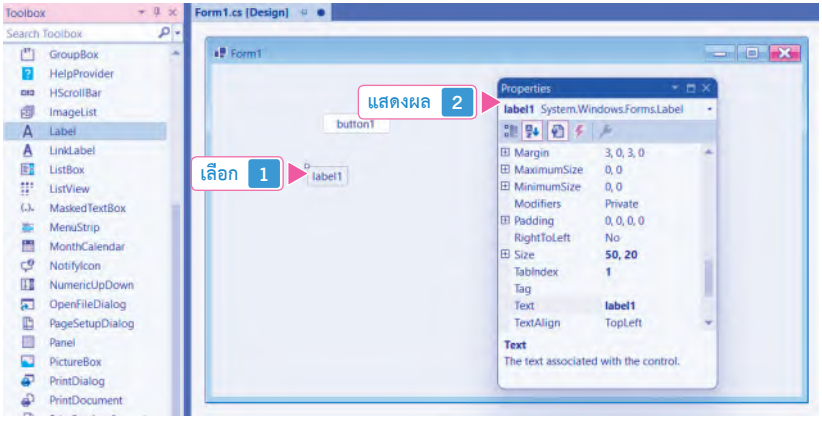
รายการคุณสมบัติของคอนโทรล Button ที่น่าสนใจ แสดงดังตารางต่อไปนี้

คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับตัวคอนโทรล ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงในปุ่มกด
BackColor	กำหนดสีพื้นหลังให้กับปุ่มกด
ForeColor	กำหนดสีตัวอักษรที่อยู่ในปุ่มกด

Note

คุณสมบัติใดก็ตามที่ถูกแก้ไข จะแสดงด้วยตัวอักษรตัวหนา ส่วนตัวอักษรปกติ หมายถึง เป็นค่าเริ่มต้นที่โปรแกรม Visual Studio ตั้งไว้ โดยที่ผู้อ่านไม่จำเป็นต้องไปแก้ไขให้ครบทุกคุณสมบัติ แก้ไขเฉพาะเท่าที่จำเป็น

ผู้เขียนลองใช้คอนโทรล Label ทำหน้าที่แสดงข้อความ เมื่อคอนโทรล Label ได้รับโฟกัส พบว่าที่หน้าต่างคุณสมบัติ ก็จะแสดงรายการคุณสมบัติของคอนโทรล Label ให้เราโดยอัตโนมัติ



รูปที่ 1-21 แสดงรายการคุณสมบัติของโทรล Label

รายการคุณสมบัติที่น่าสนใจ แสดงดังตารางต่อไปนี้

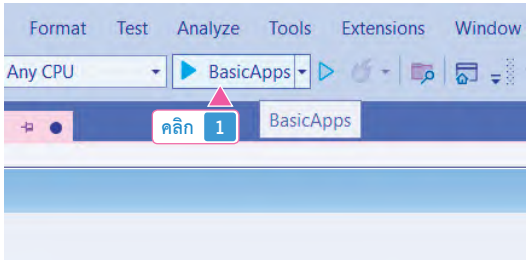
คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับตัวคอนโทรล ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงใน Label

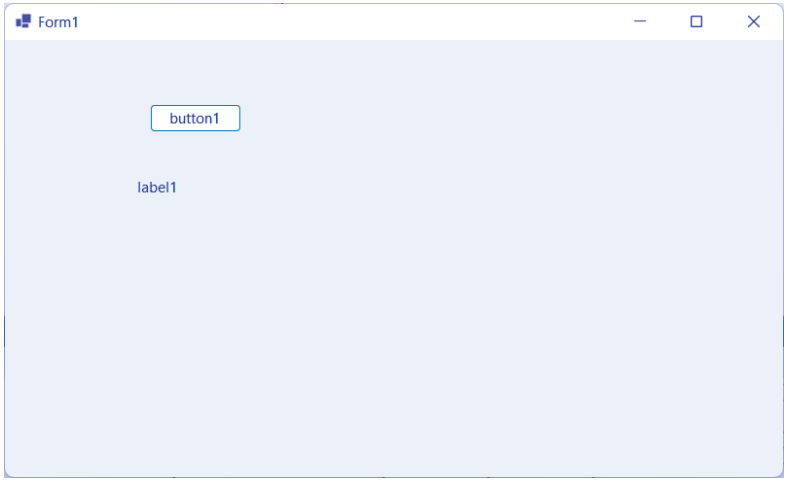
คุณสมบัติ	หน้าที่
BackColor	กำหนดสีพื้นหลังให้กับคอนโทรล Label
ForeColor	กำหนดสีตัวอักษรที่อยู่ในคอนโทรล Label
AutoSize	กำหนดขนาดของคอนโทรล Label มี 2 แบบคือ • **True** หมายถึง กำหนดให้ Label มีขนาดเท่าที่ข้อความแสดงอยู่ • **False** หมายถึง ผู้อ่านสามารถกำหนดขนาดของ Label ได้อย่างอิสระ

คอนโทรลแต่ละตัวก็จะมีรายการคุณสมบัติทั้งที่เหมือนกัน และต่างกันไปตามหน้าที่ของตัวมันเอง

การทดสอบโปรเจ็กต์

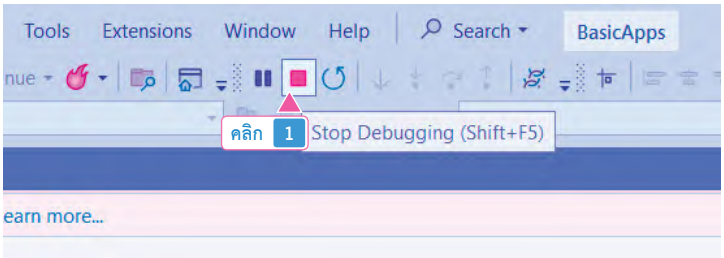
หลังจากที่ผู้อ่านออกแบบและเขียนโค้ดแล้ว ก็จะเข้าสู่ขั้นตอนการทดสอบโปรเจ็กต์ว่า มีการทำงานหรือมีผลการทำงานเป็นไปตามที่เราต้องการหรือไม่ โดยการคลิกปุ่ม BasicApps เพื่อเริ่มต้นทดสอบรันโปรเจ็กต์





รูปที่ 1-22 แสดงการเริ่มทดสอบโปรเจ็กต์

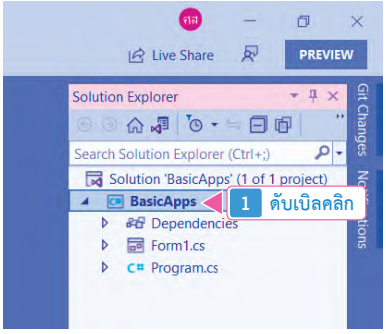
ในกรณีที่ผู้อ่านต้องการหยุดรันโปรเจ็กต์ปัจจุบัน ให้ผู้อ่านคลิกปุ่ม ■ ดังรูปที่ 1-23



รูปที่ 1-23 กรณีต้องการหยุดรันโปรเจ็กต์ปัจจุบัน

การยกเลิกฟีเจอร์ Nullable

การสร้างโปรเจ็กต์ทุกประเภทตั้งแต่ .NET 6 เป็นต้นไป ผู้เขียนแนะนำให้ผู้อ่านยกเลิกฟีเจอร์ Nullable เป็นลำดับแรกก่อนเสมอ เพราะว่ายังไม่ได้ใช้ประโยชน์อะไรกับฟีเจอร์นี้ โดยการดับเบิลคลิกที่ชื่อโปรเจ็กต์ก่อน ดังรูปที่ 1-24



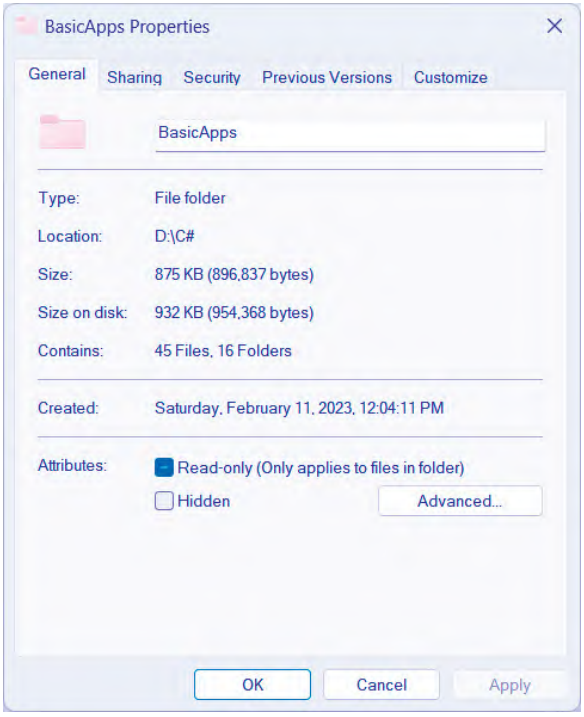
รูปที่ 1-24 แสดงการยกเลิกฟีเจอร์ Nullable

จากรูปที่ 1-24 ให้แก้ไขค่าเดิม enable เป็นค่าใหม่ disable เพื่อยกเลิกการใช้งานฟีเจอร์ Nullable

วิธีการจัดเก็บโปรเจ็กต์ .NET

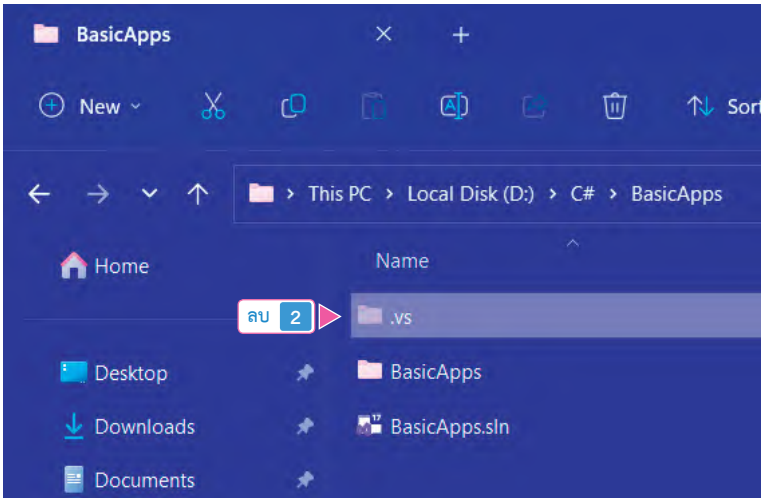
เป้าหมายของวิธีนี้คือ ต้องการลดขนาดไฟล์โปรเจ็กต์ให้มีขนาดเล็กลง โปรเจ็กต์ทุกประเภทของ .NET สามารถใช้วิธีการจัดเก็บโปรเจ็กต์แบบนี้ได้ มีขั้นตอนดังนี้

1. ให้ผู้อ่านตรวจสอบขนาดโปรเจ็กต์ก่อนปรับปรุง โดยการคลิกขวาแล้วเลือกคำสั่ง Properties พบว่าขนาดโปรเจ็กต์ 875 KB แสดงดังรูปที่ 1-25



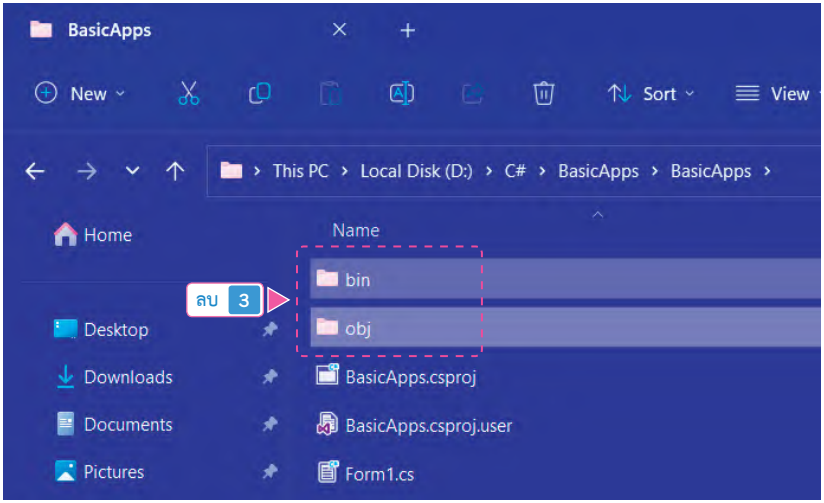
รูปที่ 1-25 แสดงขนาดโปรเจ็กต์ก่อนปรับปรุง

2. การปรับปรุงขนาดโปรเจ็กต์ .NET ทำได้โดยการเข้าไปในโฟลเดอร์ย่อยระดับแรกก่อน ผู้อ่านสามารถลบโฟลเดอร์ .vs ออกได้เลย ดังรูปที่ 1-26



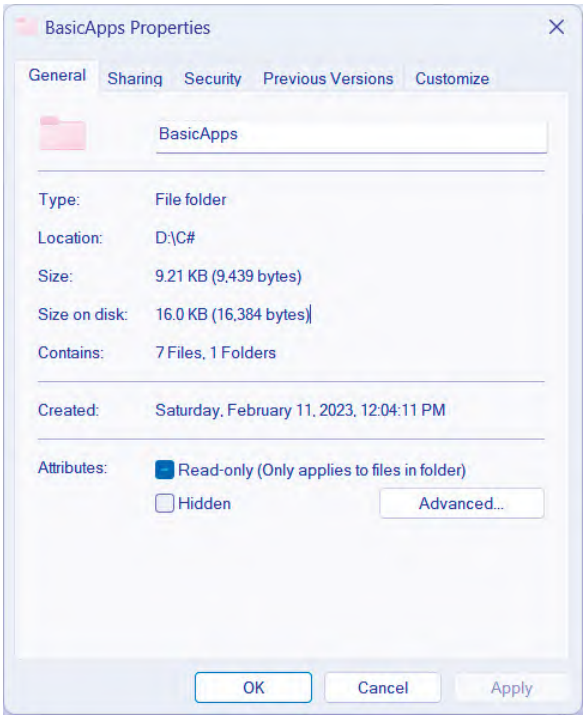
รูปที่ 1-26 แสดงการลบโฟลเดอร์ .vs

3. เข้าไปในโฟลเดอร์ย่อยอีก 1 ระดับ ให้ลบโฟลเดอร์ bin และ obj ออก ถือว่าเสร็จสิ้นแล้ว ดังรูปที่ 1-27



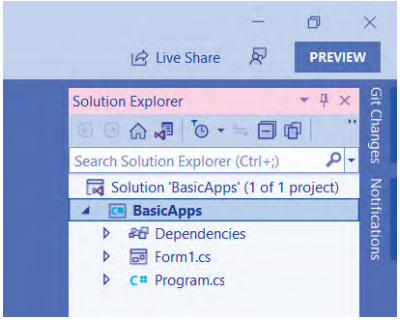
รูปที่ 1-27 แสดงการลบโฟลเดอร์ย่อย bin และ obj

4. ให้ลองตรวจสอบขนาดโปรเจ็กต์อีกครั้ง พบว่าขนาดโปรเจ็กต์จะลดลงแล้ว ผู้อ่านสามารถนำโปรเจ็กต์ .NET ไปจัดเก็บได้แล้ว ดังรูปที่ 1-28



รูปที่ 1-28 แสดงขนาดโปรเจ็กต์หลังปรับปรุงแล้ว

5. การเปิดโปรเจ็กต์ที่มีการปรับปรุงแบบนี้ ต้องรอให้เครื่องหมายอัศเจรีย์ (!) ในหน้าต่าง Solution Explorer หายไปก่อน ก็จะใช้งานโปรเจ็กต์ได้ตามปกติ



รูปที่ 1-29 แสดงการเปิดโปรเจ็กต์ที่มีการปรับปรุงขนาด

สรุปท้ายบท

เนื้อหาในบทนี้ เป็นการเตรียมความพร้อมก่อนเข้าสู่การเขียนโปรแกรมด้วยภาษา C# บนแพลตฟอร์มของ .NET ในบทถัดไป

NOTE

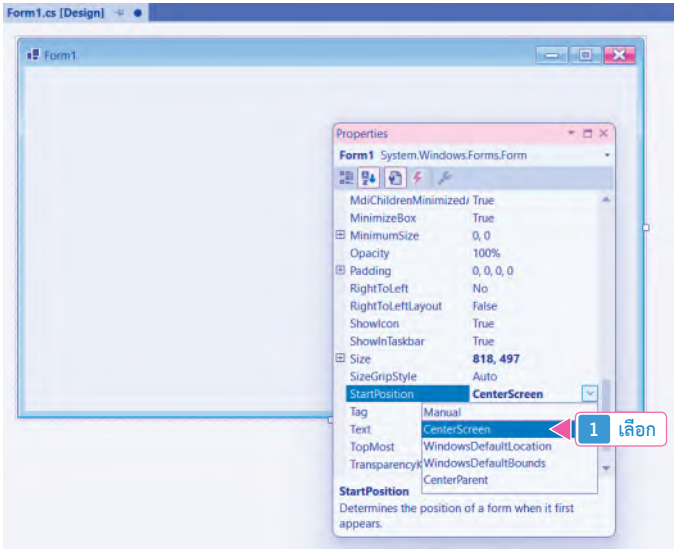
บทที่ 2

พื้นฐานการทำงานกับโปรเจ็กต์ประเภท Windows Forms App (.NET)

ฟอร์ม (Form) ทำหน้าที่เหมือนกับเป็นตัวบรรจุ (Container) คอนโทรลต่างๆ เพื่อสร้างส่วนแสดงผลตามที่ผู้อ่านออกแบบไว้ เนื้อหาในบทนี้ เราจะมาดูกันว่ามีอะไรน่าสนใจในขั้นต้นบ้าง

ทำความรู้จักกับฟอร์ม (Form)

เมื่อผู้อ่านสร้างโปรเจ็กต์แบบ Windows Forms Application ก็จะได้ฟอร์ม (Form) ที่มีชื่อว่า Form1 เป็นหน้าจอเริ่มต้น เรามักจะกำหนดให้แบบฟอร์มอยู่กึ่งกลางหน้าจอ (หรือกึ่งกลางส่วนแสดงผลเสมอ) โดยการกำหนดคุณสมบัติ **StartPosition** เท่ากับ **CenterScreen** ดังรูปที่ 2-1



รูปที่ 2-1 แสดงรายการคุณสมบัติของฟอร์ม

จากรูปที่ 2-1 เมื่อผู้อ่านโฟกัสที่ Form1 หน้าต่างคุณสมบัติก็จะแสดงรายการคุณสมบัติของ Form1 เช่นกัน

คุณสมบัติของ Form ที่สำคัญ

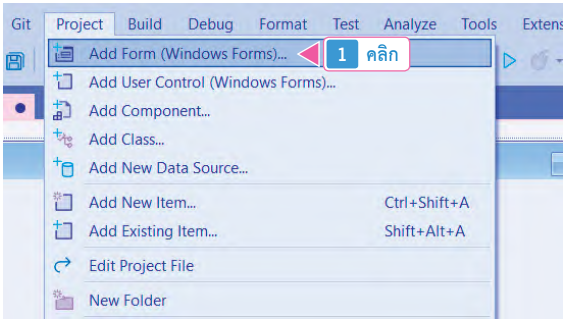
คุณสมบัติของ Form ที่สำคัญ แสดงดังตารางต่อไปนี้

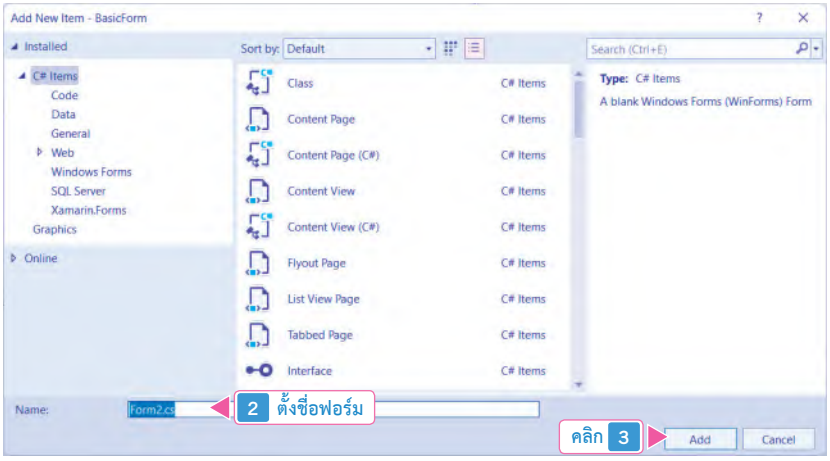
คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับฟอร์ม ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงในแถบ Title Bar
BackColor	กำหนดสีพื้นหลังให้กับฟอร์ม
ForeColor	กำหนดสีตัวอักษรที่อยู่ในฟอร์ม
FormBorderStyle	กำหนดรูปแบบเส้นขอบของฟอร์ม โดยที่ • **None** หมายถึง ไม่มีแถบไตเติลบาร์ • **FixedSingle** หมายถึง ปรับขนาดฟอร์มไม่ได้ แต่มีปุ่ม Minimize, Maximize และปุ่ม Close • **Fixed3D** หมายถึง กำหนดให้มีเส้นขอบแบบ FixedSingle แต่มีแนวลึกด้วย • **FixedDialog** หมายถึง กำหนดให้เป็นแบบ FixedSingle แต่ใช้ในรูปแบบไดอะล็อกโต้ตอบกับผู้ใช้งาน • **Sizable** หมายถึง แบบฟอร์มปรับขนาดได้ • **FixedToolWindow** หมายถึง แบบฟอร์มปรับขนาดไม่ได้ และไม่มีปุ่ม Minimize และปุ่ม Maximize • **SizableToolWindow** หมายถึง แบบฟอร์มปรับขนาดได้ และไม่มีปุ่ม Minimize และปุ่ม Maximize
MaximizeBox	กำหนดให้แบบฟอร์มแสดงปุ่มขยายขนาดฟอร์ม โดยที่ • **True** หมายถึง ให้มีปุ่ม Maximize มุมขวา-บน • **False** หมายถึง ซ่อนปุ่ม Maximize
MinimizeBox	กำหนดให้แบบฟอร์มแสดงปุ่มย่อขนาดฟอร์ม โดยที่ • **True** หมายถึง ให้มีปุ่ม Minimize มุมขวา-บน • **False** หมายถึง ซ่อนปุ่ม Minimize
IsMdiContainer	กำหนดให้ฟอร์มทำหน้าที่บรรจุฟอร์มอื่นๆ แบบหลายหน้าต่างได้หรือไม่ โดยที่ • **True** หมายถึง กำหนดให้ฟอร์มเป็นตัวบรรจุแบบ MDI • **False** หมายถึง กำหนดให้เป็นแบบฟอร์มปกติ

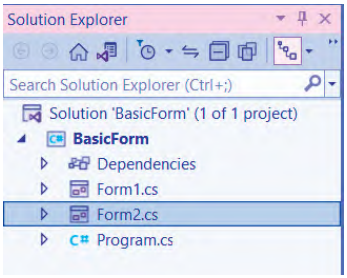
คุณสมบัติ	หน้าที่
WindowState	กำหนดขนาดของฟอร์ม โดยที่ • **Normal** หมายถึง กำหนดให้ฟอร์มมีขนาดตามที่เราออกแบบ • **Minimized** หมายถึง กำหนดให้ฟอร์มมีขนาดย่อลงในแถบ Taskbar • **Maximized** หมายถึง กำหนดให้ฟอร์มมีขนาดเต็มหน้าจอส่วนแสดงผล

การเพิ่มฟอร์มใหม่เข้ามาในโปรเจ็กต์

ในการพัฒนาแอปขึ้นมาใช้งาน ย่อมที่จะต้องประกอบไปด้วยส่วนแสดงผลตั้งแต่ 2 หน้าจอขึ้นไป วิธีการเพิ่มหน้าจอใหม่เข้ามา ให้ผู้อ่านคลิกเมนู Project > Add Form (Windows Forms)... ในกรณีนี้ เราต้องการเพิ่มหน้าจอใหม่ประเภท Windows Forms Application ตั้งชื่อว่า Form2 ดังรูปที่ 2-2

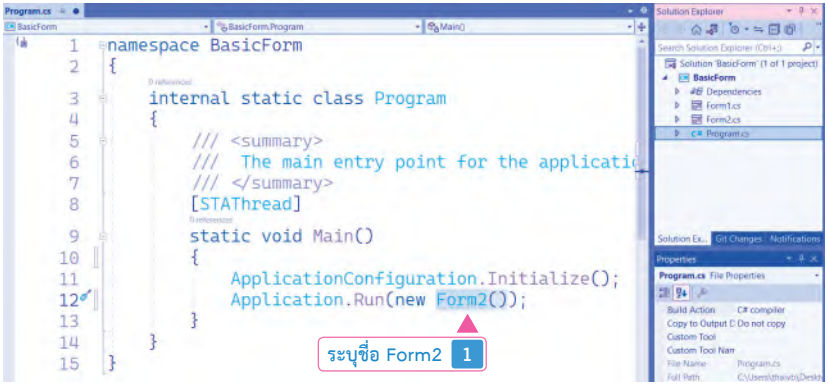






รูปที่ 2-2 แสดงการเพิ่มฟอร์มที่ชื่อว่า Form2 เข้ามาในโปรเจ็กต์

จากรูปที่ 2-2 ณ จุดนี้โปรเจ็กต์ของเรามี 2 ฟอร์ม โดยปกติแล้วจะกำหนดให้ Form1 รันเป็นลำดับแรกเสมอ ผู้อ่านสามารถแก้ไขได้ที่คลาส Program เช่น ต้องการให้ Form2 ปรากฏเป็นลำดับแรก ดังรูปที่ 2-3

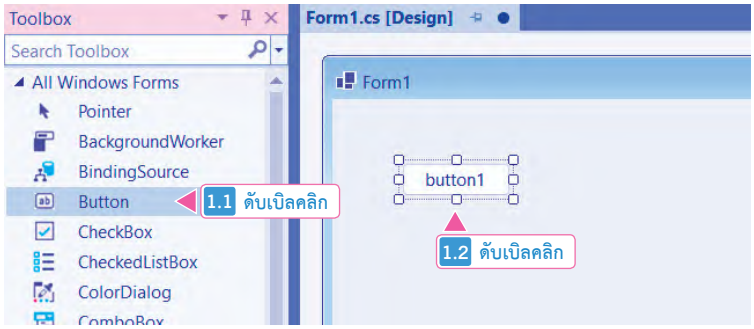


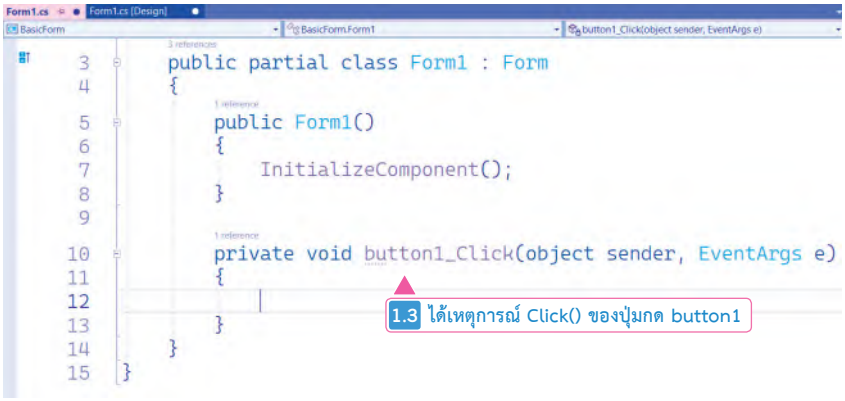
รูปที่ 2-3 แสดงการกำหนดให้ Form2 ปรากฏขึ้นมาเป็นลำดับแรก

วิธีการสั่งให้เปิดฟอร์มใหม่โดยการเขียนโค้ด

สำหรับวิธีการสั่งให้เปิดฟอร์มใหม่ที่เราเพิ่มเข้ามา มีขั้นตอนดังนี้

1. ใน Form1 เพิ่มปุ่มกด button1 เข้ามาก่อน ให้ผู้อ่านดับเบิลคลิกปุ่มกด button1 เพื่อสร้างเหตุการณ์ Click()





รูปที่ 2-4 แสดงโค้ดที่ใช้สำหรับเปิด Form2

2. จากนั้นเขียนโค้ดดังต่อไปนี้

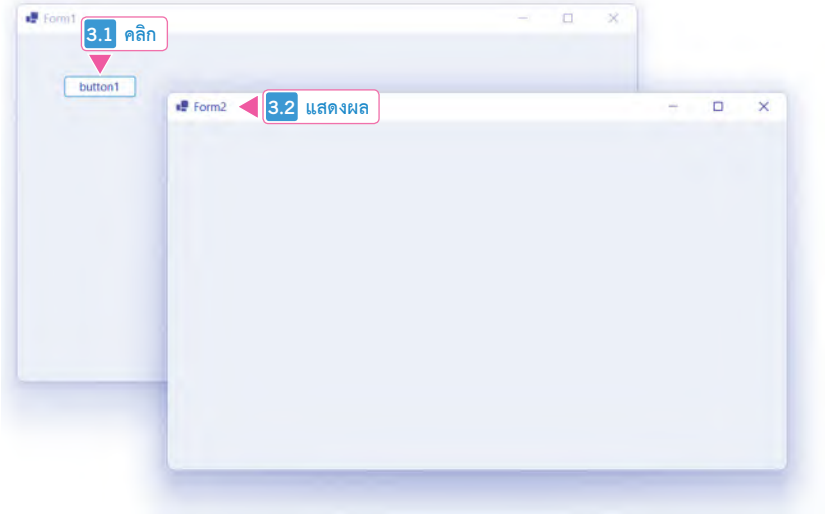
Form1.cs

```
namespace BasicForm
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form2 frm = new Form2();
            frm.Show();
        }
    }
}
```

การทำงานของโค้ดข้างต้นคือ ในเหตุการณ์ Click() ของปุ่มกด Button ที่ชื่อว่า button1 ให้ทำ

- สร้างตัวแปรออบเจ็กต์ Form2 ที่ชื่อว่า frm (Form2 frm = new Form2()) ขึ้นมาก่อน
- สั่งให้ Form2 ปรากฏขึ้นมา ผ่านทางตัวแปร frm ร่วมกับเมธอด Show() (frm.Show())

3. ให้ผู้อ่านทดสอบรันโปรเจ็กต์ พบว่าเราสามารถเปิด Form2 ได้แล้ว ดังรูปที่ 2-5



รูปที่ 2-5 Form2 ที่ถูกเปิดจากปุ่มกด button1 ใน Form1

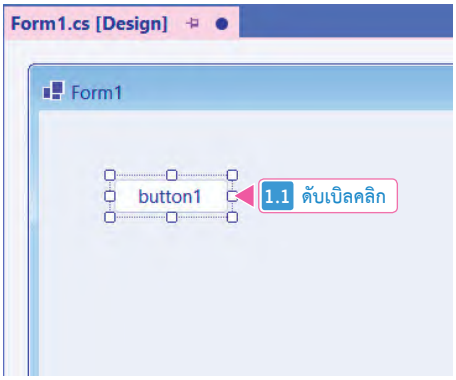
หลักการเขียนโปรแกรมแบบ Event Driven Programming

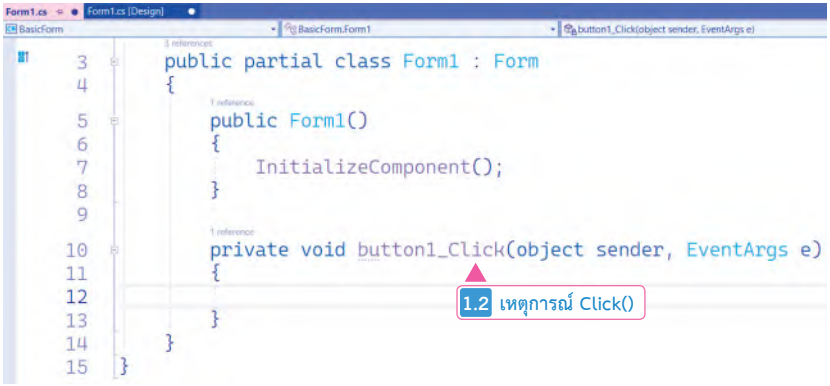
หลักการทำงานพื้นฐานของโปรเจ็กต์ประเภท Windows Forms Application เรียกว่า **การเขียนโปรแกรมตอบสนองเหตุการณ์ที่เกิดขึ้น (Event Driven Programming)** กล่าวคือ ผู้อ่านเป็นผู้กำหนดเองว่า เมื่อเกิดเหตุการณ์ Click() ให้ทำอะไร, เมื่อเกิดเหตุการณ์ Load() ให้ทำอะไร เป็นต้น

หมายความว่า การกระทำต่างๆ ที่ผู้ใช้งานกระทำต่อโปรแกรม ผู้อ่านเป็นผู้กำหนดขึ้นมาเองทั้งสิ้นว่า "ให้ทำอะไร" ประเด็นที่ตามมาคือ วิธีการสร้างเหตุการณ์ให้กับคอนโทรลแต่ละตัว ทำอย่างไร ลำดับโค้ดมาก่อนไม่ได้หมายความว่าโค้ดทำงานก่อน โค้ดทำงานตามเหตุการณ์ที่เกิดขึ้น

สมมติว่าผู้เขียนต้องการสร้างเหตุการณ์ Click() ให้กับปุ่มกด Button การสร้างเหตุการณ์ให้กับปุ่มกด button1 นี้ มี 2 วิธี

1. ดับเบิลคลิกที่คอนโทรล button1 เพื่อสั่งให้ VS 2022 สร้างเหตุการณ์ประจำตัวของคอนโทรล Button ขึ้นมา คอนโทรล Button ถูกสร้างขึ้นมาเพื่อทำหน้าที่รองรับการคลิกจากผู้ใช้งาน ส่งผลให้เหตุการณ์ประจำตัวของคอนโทรล Button ก็คือ เหตุการณ์ Click()

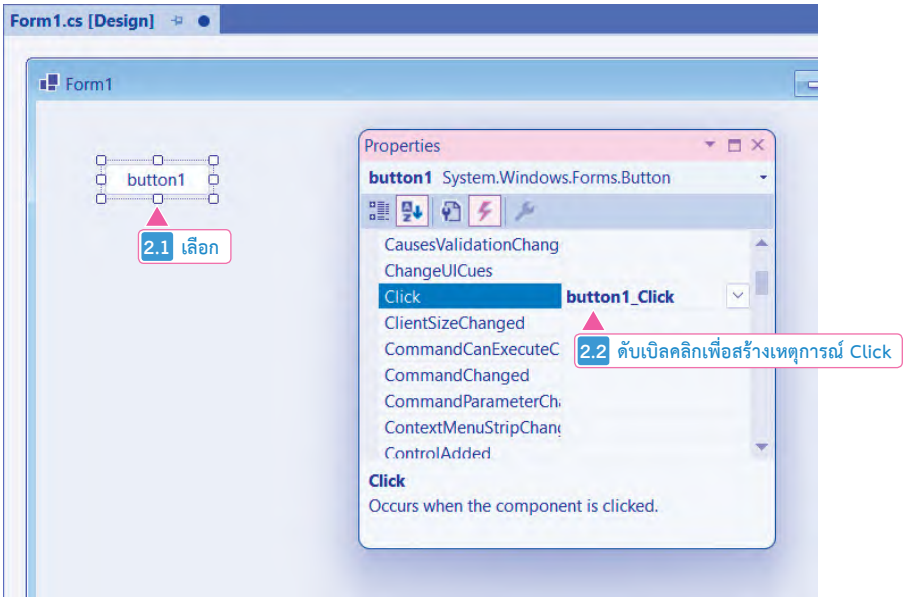




รูปที่ 2-6 แสดงการสร้างเหตุการณ์ Click() ด้วยวิธีดับเบิลคลิก

คอนโทรลแต่ละตัวมีเหตุการณ์ประจำตัวแตกต่างกัน ขึ้นอยู่กับว่า คอนโทรลตัวนั้นๆ ถูกสร้างขึ้นมาเพื่อ "ทำหน้าที่อะไร"

2. การสร้างเหตุการณ์อีกวิธีหนึ่งคือ ให้ผู้อ่านโฟกัสที่คอนโทรล button1 ก่อน จากนั้นคลิกปุ่ม ⚡ ในหน้าต่าง Properties เพื่อเลือกสร้างเหตุการณ์ที่เราต้องการ จะเห็นได้ว่า ปุ่มกด Button รองรับหลายเหตุการณ์ ให้ผู้อ่านดับเบิลคลิกที่ Click เพื่อสร้างเหตุการณ์ Click() ขึ้นมาสำหรับปุ่มกด button1 ปุ่มนี้



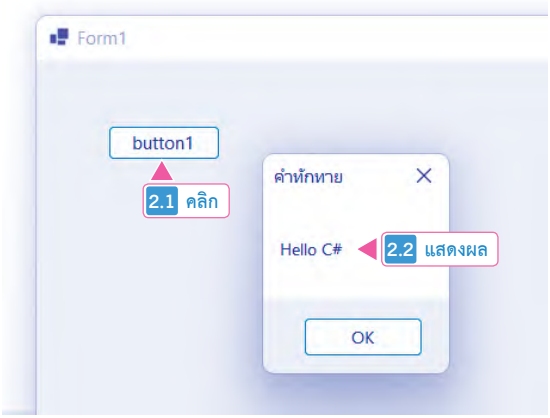
รูปที่ 2-7 แสดงเหตุการณ์ Click() ที่ได้จากวิธีเลือกสร้างเหตุการณ์

ผู้เขียนลองตั้งโจทย์ง่ายๆ ว่า ต้องการแสดงข้อความ "Hello C#" เมื่อผู้ใช้งานคลิกปุ่ม button1 ดังโค้ดต่อไปนี้

Form1.cs

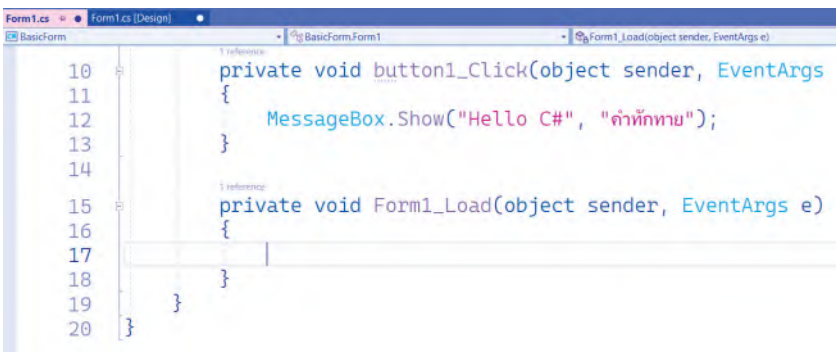
```
namespace BasicForm
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Hello C#", "คำทักทาย");
        }
    }
}
```

จากโค้ดข้างต้น เป็นการสั่งให้แสดงข้อความ "Hello C#" โดยอาศัยคลาส MessageBox เข้ามาทำหน้าที่สร้างไดอะล็อกแสดงข้อความ ดังรูปที่ 2-8



รูปที่ 2-8 แสดงข้อความที่ได้

3. ในกรณีที่ผู้อ่านดับเบิลคลิกบริเวณพื้นที่ว่างๆ ของ Form1 ส่งผลให้ C# ก็จะสร้างเหตุการณ์ Load() ขึ้นมา เหตุการณ์ประจำตัวของ Form ก็คือ เหตุการณ์ Load() เพื่อให้ผู้อ่านกำหนดว่า เมื่อ Form1 ถูกโหลดขึ้นมาแล้วให้ทำอะไร



```
10          private void button1_Click(object sender, EventArgs
11          {
12              MessageBox.Show("Hello C#", "คำทักทาย");
13          }
14
15          private void Form1_Load(object sender, EventArgs e)
16          {
17
18          }
19      }
20  }
```

รูปที่ 2-9 แสดงเหตุการณ์ Form1_Load()

พื้นฐานการใช้งานฟอร์มแบบ MDI และสร้างระบบเมนู (Menu)

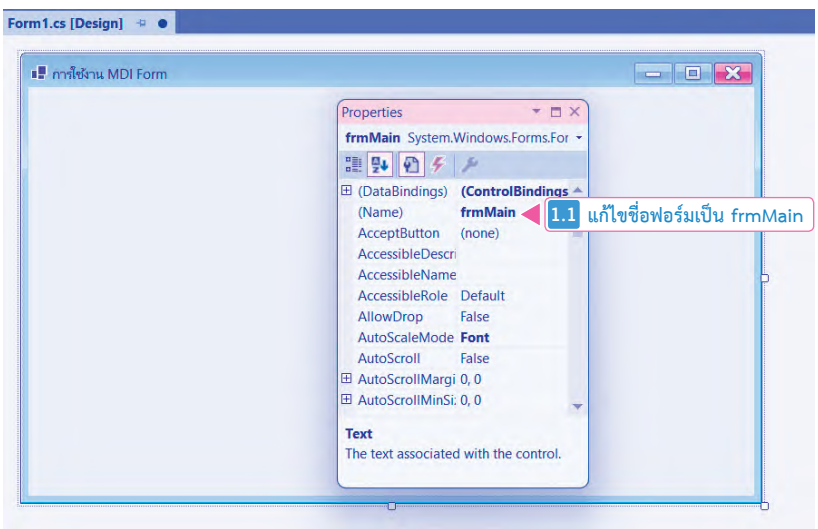
ฟอร์ม (Form) ทำหน้าที่เป็นตัวบรรจุ (Container) คอนโทรลต่างๆ ที่ผู้อ่านเรียกใช้งานอยู่ โดยที่ในแอปพลิเคชันชนิด Windows Application 1 ตัว จะต้องประกอบไปด้วยฟอร์มอย่างน้อยที่สุด 1 ฟอร์ม เพื่อทำหน้าที่เป็นส่วนแสดงผลให้กับผู้ใช้ (เรียกว่า User Interface – UI) รูปร่างหน้าตาโปรแกรมของผู้อ่านจะเป็นอย่างไร ก็จะขึ้นอยู่กับการออกแบบฟอร์มของผู้อ่านนั่นเอง

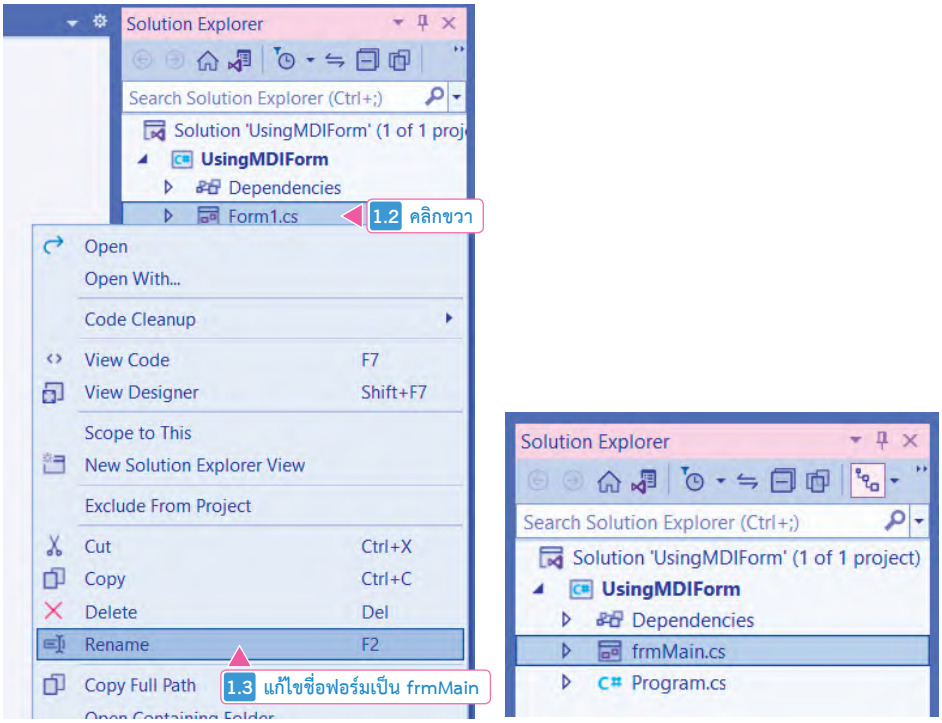
เราสามารถแบ่งลักษณะของฟอร์มได้ 2 ลักษณะคือ

- **การใช้งานฟอร์มแบบ SDI (Single Document Interfaces)** หมายถึง แอปพลิเคชันที่สามารถเปิดเอกสารได้เพียงครั้งละ 1 เอกสาร หรือแสดงผลส่วนติดต่อกับผู้ใช้ได้ครั้งละ 1 หน้าจอ
- **การใช้งานฟอร์มแบบ MDI (Multiple Document Interfaces)** หมายถึง การใช้งานฟอร์มที่สามารถแสดงส่วนแสดงผล หรือส่วนติดต่อกับผู้ใช้ได้มากกว่า 1 หน้าจอในเวลาเดียวกัน โดยที่จะมี 1 ฟอร์ม ทำหน้าที่เป็นฟอร์มแม่ เรียกว่า **Parent Form** ทำหน้าที่บรรจุฟอร์มลูกที่เรียกว่า **Child Form**

ตัวอย่างที่ 2-1 พื้นฐานการใช้งานฟอร์มแบบ MDI มีขั้นตอนดังนี้

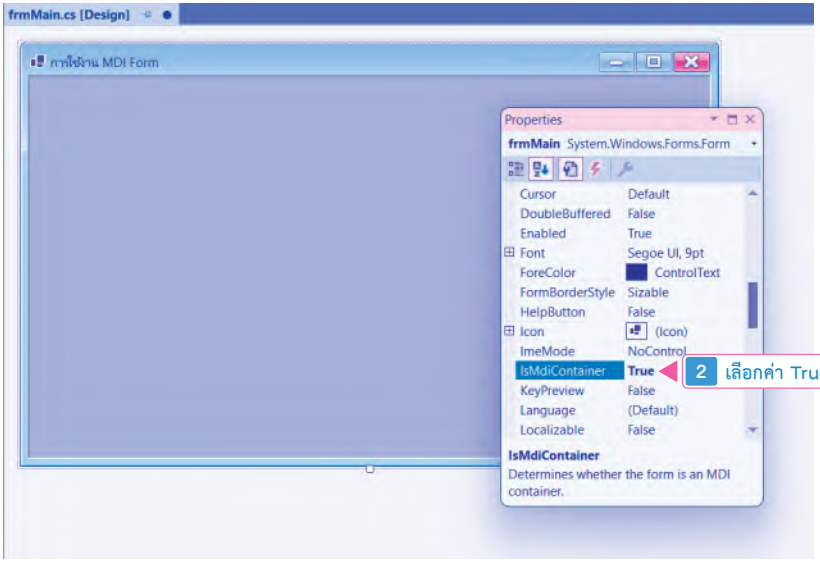
1. สร้างโปรเจ็กต์ที่ชื่อว่า UsingMDIForm เปลี่ยนชื่อฟอร์ม Form1 เป็น frmMain ก่อนเพื่อทำหน้าที่เป็นฟอร์มหลักของโปรเจ็กต์ ดังรูปที่ 2-10





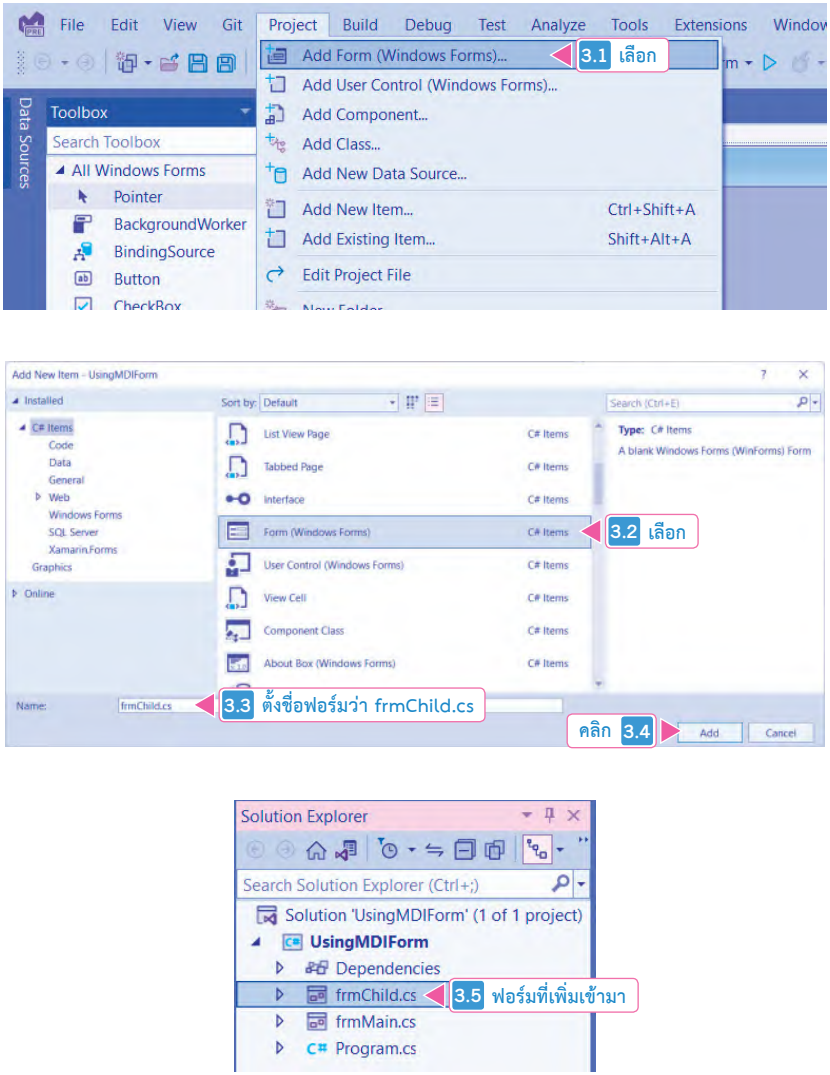
รูปที่ 2-10 แสดงการเปลี่ยนชื่อฟอร์มเป็น frmMain

2. กำหนดให้ฟอร์มนี้เป็นฟอร์มหลัก MDI ทำหน้าที่บรรจุฟอร์มลูกต่างๆ โดยการกำหนดที่คุณสมบัติ **IsMdiContainer** = True ดังรูปที่ 2-11



รูปที่ 2-11 แสดงการกำหนดให้ฟอร์มนี้ทำหน้าที่เป็นฟอร์มหลัก MDI

3. ให้ผู้อ่านคลิกเมนู Project > Add Form (Windows Forms)... เพื่อเพิ่มฟอร์มใหม่เข้ามาในโปรเจ็กต์ ตั้งชื่อว่า frmChild ทำหน้าที่เป็นฟอร์มลูก ดังรูปที่ 2-12



รูปที่ 2-12 แสดงฟอร์มลูกที่ถูกเพิ่มเข้ามาในโปรเจ็กต์