Professional Visual C# 2015

- อ่านเข้าใจง่าย อธิบายอย่างเป็นขั้นตอน มีตัวอย่างประกอบทุกหัวข้อ
- รองรับการพัฒนาทั้ง PC, Mobile และ Web Site
- เหมาะสำหรับนักเรียน นักศึกษา และผู้ที่สนใจพัฒนาแอพพลิเคชัน

ดาวน์โหลดโค้ด
http://www.infopress.co.th/devbook/CodeVC2015.zip

ศุภชัย สมพานิช

IDC
PREMIER

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

Think Beyond

Read Comics

Professional Visual C# 2015

ผู้แต่ง	ศุภชัย สมพานิช
บรรณาธิการ	กิตินันท์ พลสวัสดิ์ kitinan_p@idcpremier.com
ออกแบบปก	วสันต์ พึ่งพูลผล, ยุทธนา เกิดประดิษฐ์
ออกแบบและจัดรูปเล่ม	สิริลักษณ์ วาระเลิศ, จิตราภรณ์ เหมะจันทร
พิสูจน์อักษร	มนฤดี ศรีอุทโยภาส, ธนภรณ์ ก๊กใหญ่
ประสานงานการผลิต	สุพัตรา อาจปรุ, ฉัตรชนก แก้วจันทร์

บริษัท ไอดีซี พรีเมียร์ จำกัด จัดตั้งขึ้นเพื่อเผยแพร่ความรู้ที่มีคุณภาพสู่ผู้อ่านชาวไทย เรายินดีรับงานเขียนของนักวิชาการและนักเขียนทุกท่าน ท่านผู้สนใจกรุณาติดต่อผ่านทางอีเมลที่ infopress@idcpremier.com หรือทางโทรศัพท์หมายเลข 0-2962-108 (อัตโนมัติ 10 คู่สาย) โทรสาร 0-2962-1084

ข้อมูลทางบรรณานุกรม

ศุภชัย สมพานิช
Professional Visual C# 2015
นนทบุรี : ไอดีซีฯ, 2559
440 หน้า
1. การเขียนโปรแกรมโดยใช้ภาษา
โปรแกรมเฉพาะชนิด
I ชื่อเรื่อง
005.262

ISBN 885-916-100-435-6

พิมพ์ครั้งที่ 1 กันยายน 2559
2 4 6 8 10 9 7 5 3 1

จัดพิมพ์และจัดจำหน่ายโดย

บริษัท ไอดีซี พรีเมียร์ จำกัด
200 หมู่ 4 ชั้น 19 ห้อง 1901 อาคารจัสมินอินเตอร์เนชั่นแนลทาวเวอร์
ถ.แจ้งวัฒนะ อ.ปากเกร็ด จ.นนทบุรี 11120
โทรศัพท์ 0-2962-1081 (อัตโนมัติ 10 คู่สาย) โทรสาร 0-2962-1084

ราคา 350 บาท

บรรณาธิการ

ปัจจุบันโปรแกรมหรือแอพพลิเคชันไม่ได้มีแต่เพียงแค่บนเครื่องคอมพิวเตอร์ตั้งโต๊ะหรือโน้ตบุ๊ค แต่ได้มีการพัฒนาไปบนเครื่องมืออิเล็กทรอนิกส์ต่าง ๆ เป็นจำนวนมาก โดยเฉพาะบนโทรศัพท์เคลื่อนที่ หรือที่เรียกกันว่า สมาร์ทโฟน และแท็บเล็ต

จึงเป็นที่มาของ Visual Studio 2015 ที่ไมโครซอฟท์ตั้งใจพัฒนาขึ้นมาเพื่อรองรับการพัฒนาแอพพลิเคชันในทุก ๆ ช่องทาง ทั้งการพัฒนาแอพพลิเคชันสำหรับ PC (หรือ Notebook), การพัฒนาเว็บไซต์ (ไม่ขึ้นกับแพลตฟอร์ม), การพัฒนาแอพพลิเคชันสำหรับมือถือและแท็บเล็ต

เนื้อหาในหนังสือเล่มนี้จึงเรียบเรียงขึ้นบนแนวความคิดที่ว่า “เมื่อผู้อ่านอ่านหนังสือเล่มนี้ ต้องสามารถพัฒนาแอพพลิเคชันบน Desktop, Web, มือถือ และแท็บเล็ต ด้วย Visual C# 2015 ได้” ซึ่งทางสำนักพิมพ์เชื่อว่าผู้อ่านสามารถทำได้จริง เพราะผู้เขียนได้อธิบายทุกขั้นตอนอย่างละเอียด

ทั้งนี้ทางสำนักพิมพ์หวังเป็นอย่างยิ่งว่า หนังสือเล่มนี้จะสามารถช่วยให้ท่านผู้อ่านทุกท่านพัฒนาแอพพลิเคชันได้จริง และหากหนังสือเล่มนี้มีข้อผิดพลาดประการใด ต้องขออภัยมา ณ โอกาสนี้ด้วย ทางสำนักพิมพ์พร้อมน้อมรับทุกคำติชม เพื่อนำไปใช้ในการปรับปรุงในครั้งต่อ ๆ ไป

กิตินันท์ พลสวัสดิ์

บรรณาธิการ

คำนำ

หนังสือ Professional Visual C# 2015 เล่มนี้ เป็นการนำเสนอเนื้อหาพื้นฐานการเขียนโปรแกรมด้วยภาษา C# โดยใช้รูปแบบเรียงลำดับเนื้อหาตั้งแต่เบื้องต้นไปจนกระทั่งถึงเนื้อหาในระดับที่มีความซับซ้อนมากขึ้นเรื่อย ๆ ครอบคลุมทั้ง Windows Forms Apps (Desktop Apps), การพัฒนาร่วมกับระบบฐานข้อมูล, การพัฒนา Web Apps ด้วย ASP.NET MVC และการพัฒนา Universal Apps บน Windows 10

สำหรับคุณผู้อ่านที่ยังไม่เคยศึกษาการเขียนโปรแกรมด้วยภาษาใด ๆ มาก่อนเลย ขอให้ศึกษาตามเนื้อหาในหนังสือเล่มนี้ ส่วนคุณผู้อ่านที่เคยศึกษาภาษาอื่น ๆ มาบ้างแล้ว สามารถเลือกดูบทต่าง ๆ ได้ตามที่ต้องการ

ในกรณีที่คุณผู้อ่านติดปัญหาหรือมีข้อสงสัยเกี่ยวกับเนื้อหาบทใดก็ตาม สามารถโพสต์คำถามได้ที่แฟนเพจของผู้เขียนที่ https://www.facebook.com/thaivb.net/

ในบางกรณีผู้เขียนอาจจะตอบคำถามในรูปแบบของคลิปวิดีโออีกด้วย ขอให้คุณผู้อ่านกดติดตามช่องใน Youtube โดยค้นหาช่องที่ชื่อว่า “Thaivb.NET”

ศุภชัย สมพานิช

กันยายน 2559

CONTENTS

บทที่ 1 เตรียมความพร้อมก่อนเริ่มต้นเขียนโปรแกรมด้วยภาษา Visual C# 2015 1

การดาวน์โหลดและติดตั้ง Visual Studio 2015 2
การตั้งค่า Visual Studio 2015 เบื้องต้น 3
- การกำหนดให้แสดงไดอะล็อกเลือกโปรเจ็กต์ (New Project Dialog) 3
- การกำหนดเลขบรรทัด 4

การสร้างโปรเจ็กต์ใหม่ 5
- ทำความรู้จักกับโปรเจ็กต์ประเภทอื่นๆ ของ Visual Studio 2015 7

ทำความรู้จักกับสภาพแวดล้อม (IDE) ของ VS ในขั้นต้น 10
มือใหม่ต้องทราบก่อนเขียนโปรแกรมในโลกของ .NET 11
- การออกแบบส่วนแสดงผลด้วยคอนโทรล (User Interface) 11
- การปรับแต่งคอนโทรลขั้นต้นด้วยวิธีแก้ไขคุณสมบัติ (Property) 12
- ทำความรู้จักกับฟอร์ม (Form) 14
- การเพิ่มฟอร์มใหม่เข้ามาในโปรเจ็กต์ 15
- เริ่มต้นสร้างแอพพลิเคชันแรก 16
- การทดสอบโปรเจ็กต์ 17
- วิธีการสั่งให้เปิดฟอร์มใหม่โดยการเขียนโค้ด 18

หลักการทำงานของภาษา C# 19
สรุปท้ายบท 21

บทที่ 2 พื้นฐานการเขียนโปรแกรมด้วยภาษา Visual C# 23

การประกาศตัวแปรขึ้นมาใช้เก็บข้อมูล 23
- กฎการตั้งชื่อตัวแปรในภาษา C# 23
- คำสงวน (Reserved Words) ของภาษา C# 24

ขอบเขตของตัวแปร (Variable Scope) 24
- ตัวแปรระดับ Local และระดับฟอร์ม 26
- ตัวแปรระดับ Block 28

ชนิดข้อมูล (Type) 29
- ขอบเขตการจัดเก็บข้อมูลแต่ละชนิดข้อมูล และขนาดหน่วยความจำที่ใช้ 29
- การกำหนดชนิดข้อมูลโดยอัตโนมัติ (Type Inference) 32
- การใช้งานข้อมูลชนิดตัวเลข 34
- การใช้งานเลขจำนวนเต็มชนิด BigInteger 35

การจัดรูปแบบของตัวเลขทศนิยม 38
พื้นฐานการทำงานกับข้อมูลชนิดข้อความ string 40
วิธีการตรวจสอบจำนวนตัวอักษรใน string 40
การต่อข้อความเข้าด้วยกัน 41
การแปลงตัวเลขที่อยู่ในฐานะข้อความ string ให้กลายเป็นข้อมูลชนิดตัวเลข 42
การแทนที่ข้อความด้วยเมธอด Replace() 43
การแบ่งข้อความด้วยฟังก์ชัน Split() 43
การใช้งานข้อมูลชนิดตัวอักษร char 44
การแปลงชนิดข้อความ string เป็นชนิดข้อมูลอาร์เรย์ของ char 44
ทำงานกับข้อมูลชนิดวันที่และเวลา (DateTime) 45
การดักจับข้อผิดพลาดด้วยคำสั่ง try catch และคำสั่ง when 47
สรุปท้ายบท 50

บทที่ 3 การออกแบบส่วนแสดงผลด้วยคอนโทรล (User Interface-UI) 51

ทำความรู้จักกับแถบคอนโทรล 52
แสดงข้อความด้วยคอนโทรล Label 53
คุณสมบัติที่น่าสนใจของคอนโทรล Label 53
รับข้อความด้วยคอนโทรล TextBox 54
คุณสมบัติที่น่าสนใจของคอนโทรล TextBox 54
สร้างปุ่มกดด้วยคอนโทรล Button 55
คุณสมบัติที่น่าสนใจของคอนโทรล Button 55
การสร้างไฮเปอร์ลิงค์ด้วย LinkLabel 58
คุณสมบัติที่น่าสนใจของคอนโทรล LinkLabel 60
การแสดงรูปภาพด้วยคอนโทรล PictureBox 61
การกำหนดรูปภาพในขณะออกแบบ 62
การเขียนโค้ดกำหนดรูปภาพที่ต้องการแสดง 62
คุณสมบัติที่น่าสนใจของคอนโทรล PictureBox 64
การสร้างตัวเลือกแบบเลือกได้ 1 อย่าง ด้วยคอนโทรล RadioButton 64
การสร้างตัวเลือกแบบหลายตัวเลือกด้วยคอนโทรล CheckBox 66
การสร้างส่วนแสดงผลแบบ list ด้วย ComboBox และ ListBox 68
การเพิ่ม ถอด และล้างรายการใน ListBox 72
การสร้างส่วนแสดงผลแบบรายการด้วยคอนโทรล ListView 76
การสร้างรายการแบบมีตัวเลือก CheckBox ด้วยคอนโทรล CheckedListBox 80
การกำหนดรูปแบบข้อมูลที่รับเข้ามาด้วย MaskedTextBox 82
การสร้างตัวเลือกแบบตัวเลขด้วย NumericUpDowns 84

การสร้างตัวเลือกวันที่ด้วย DateTimePicker 85
คุณสมบัติที่น่าสนใจของคอนโทรล DateTimePicker 87
การสร้างปฏิทินด้วยคอนโทรล MonthCalendar 87
คุณสมบัติที่น่าสนใจของคอนโทรล MonthCalendar 89
การแสดงข้อมูลแบบลำดับชั้นด้วยคอนโทรล TreeView 90
คุณสมบัติที่น่าสนใจของคอนโทรล TreeView 93
ทำงานกับไฟล์ข้อความด้วยคอนโทรล RichTextBox, OpenFileDialog และ SaveFileDialog 94
คุณสมบัติที่น่าสนใจของคอนโทรล RichTextBox 94
คุณสมบัติที่น่าสนใจของคอนโทรล OpenFileDialog 94
คุณสมบัติที่น่าสนใจของคอนโทรล SaveFileDialog 95
การแสดงคำอธิบายด้วย ToolTip 99
การสร้างส่วนแสดงผลแบบแท็บด้วยคอนโทรล TabControl 100
การจัดกลุ่มด้วยคอนโทรล GroupBox 101
การใช้งานตัวบรรจุ Panel 101
การแสดงความคืบหน้าในการทำงานด้วยคอนโทรล ProgressBar 102
การสร้างตัวจับเวลาด้วย Timer 103
สรุปท้ายบท 104

บทที่ 4 การตรวจสอบและควบคุม 105

ตัวดำเนินการด้านคณิตศาสตร์ 105
ตัวดำเนินการด้านเปรียบเทียบ 106
การตรวจสอบเงื่อนไขด้วยคำสั่ง if 107
การตรวจสอบเงื่อนไขด้วยคำสั่ง if...else 108
การตรวจสอบเงื่อนไขด้วยคำสั่ง if ... else if 109
การตรวจสอบเงื่อนไขด้วยคำสั่ง switch case 110
ตัวดำเนินการ “และ” (&&) และตัวดำเนินการ “หรือ” (||) 112
การตรวจสอบเงื่อนไขร่วมกับตัวดำเนินการด้านตรรกะ 113
การวนลูปด้วยคำสั่ง for 115
การวนลูปด้วยคำสั่ง while 117
การวนลูปแบบ do...while 118
สรุปท้ายบท 119

บทที่ 5 การใช้งานฟังก์ชันและพารามิเตอร์ 121

รูปแบบฟังก์ชันในภาษา VC# 2015 121
การสร้างฟังก์ชันที่มีชื่อเหมือนกันด้วยการทำ Overload Method 124

ลักษณะของพารามิเตอร์ในเมธอด 127
พารามิเตอร์ชนิดรับเข้าและส่งออกในเวลาเดียวกัน 129
การกำหนดลักษณะเพิ่มเติมของพารามิเตอร์ 132
การใช้งานพารามิเตอร์แบบ Optional 132
การส่งพารามิเตอร์แบบ Params 134
การใช้งานพารามิเตอร์แบบอาร์เรย์ 135
สรุปท้ายบท 136

บทที่ 6 พื้นฐานการเขียนโปรแกรมเชิงวัตถุ 137

การเขียนโปรแกรมเชิงวัตถุ (OOP) คืออะไร 137
พื้นฐานการสร้างคลาสขึ้นมาใช้งานเอง 138
การสร้างคลาสด้วยวิธีเขียนโค้ดแบบย่อ 142
การกำหนดค่าเริ่มต้นให้กับคุณสมบัติ (Auto Properties) 142
การสร้างคุณสมบัติที่อ่านค่าได้เพียงอย่างเดียว 144
ทำความรู้จักกับชนิดข้อมูลแบบ Anonymous Type 144
การจัดประเภทคลาสด้วยระบบเนมสเปซ (Namespaces) 147
วิธีการเรียกใช้เนมสเปซ 148
แบบที่ 1 อาศัยคำสั่ง using 148
แบบที่ 2 อาศัยการระบุชื่อเต็ม 149
แบบที่ 3 อาศัยการตั้งชื่อ Alias 150
แบบที่ 4 อาศัยคลาส global 150
ค่า null คืออะไร 150
การตรวจสอบค่า null กับตัวแปรประเภทออบเจ็กต์ด้วยเครื่องหมาย ? และ ?? 152
ทำความรู้จักกับ Constructor, การระบุสมาชิกด้วยคำสั่ง this และการกำหนดค่าเริ่มต้น
ด้วย Object Initializer 156
การสร้างคลาสแบบ Abstract Class 161
การใช้งาน Abstract Class ในฐานะ Dynamic Type 164
สรุปท้ายบท 166

บทที่ 7 อาร์เรย์และสตรัคเจอร์ (Array and Structure) 167

พื้นฐานการใช้งานตัวแปรชุดแบบอาร์เรย์ (Array) 167
การสร้างอาร์เรย์ด้วยคลาส ArrayList 171
การถอดค่าซ้ำกันในอาร์เรย์ 173
การหาค่ามากสุดและน้อยสุดในอาร์เรย์ 174
การวนลูปในอาร์เรย์แบบหลายมิติด้วยลูปแบบ foreach 178
พื้นฐานการใช้โครงสร้างแบบ struct 179

การสร้างคุณสมบัติและเมธอดใน struct 180
ข้อแตกต่างระหว่าง struct กับ Class 183
สรุปท้ายบท 185

บทที่ 8 การเขียนโปรแกรมแบบ Generic 187

ทำความรู้จักกับการเขียนโปรแกรมแบบ Generic ภายใน 30 นาที 187
ทำไม .NET จึงต้องมี Generic 191
พารามิเตอร์ T มีชนิดข้อมูลอะไรในขณะออกแบบ (Design-Time) 192
การสร้างคลาสแบบ Generic 193
การสร้างรายการ list แบบ Generic (Generic List) 195
การทำงานกับ Generic List ที่น่าสนใจ 196
การใช้งานโครงสร้างข้อมูลแบบ Key-Value ด้วย Dictionary 198
ทำงานกับโครงสร้างข้อมูลแบบเข้าก่อน-ออกก่อน ด้วย Queue (First-In First-Out หรือ FIFO) 202
ทำงานกับโครงสร้างข้อมูลแบบเข้าหลัง-ออกก่อน ด้วย Stack (Last-In First-Out หรือ LIFO) 205
สรุปท้ายบท 207

บทที่ 9 พื้นฐานการใช้งานระบบฐานข้อมูล SQL Server 2014 Express Edition 209

การเตรียมสภาพแวดล้อมให้พร้อมใช้งาน 209
การดาวน์โหลดและติดตั้ง SQL Server 2014 Express Edition 211
ขั้นตอนการติดตั้งฐานข้อมูล SQL Server 2014 Express Edition 212
พื้นฐานการใช้งานฐานข้อมูล SQL Server 2014 Express Edition 214
- การสร้างฐานข้อมูล 214
- การสร้างตาราง 216
- การกำหนด Primary Key 216

การสร้างความสัมพันธ์ระหว่างตารางในฐานข้อมูล SQL Server 2014 217
การ Backup และ Restore ฐานข้อมูล SQL Server 2014 219
- การ Backup ฐานข้อมูล 219
- การ Restore ฐานข้อมูล 221

การยกเลิกการป้องกันการแก้ไขโครงสร้างตาราง 222
การเพิ่มฐานข้อมูล Northwind เข้าไปใน SQL Server 2014 Express Edition 223
สรุปท้ายบท 224

บทที่ 10 พื้นฐานการทำงานกับฐานข้อมูล SQL Server ด้วย Entity Framework 225

พื้นฐานการใช้งาน Entity Framework เชื่อมต่อกับฐานข้อมูล SQL Server 225
การเลือกดูข้อมูลบางฟิลด์โดยอาศัย Anonymous Type 231
การใช้งาน Anonymous Type ในฐานะแหล่งข้อมูลร่วมกับ Entity Framework 236
พื้นฐานการค้นหาข้อมูลด้วย Entity Framework 239
พื้นฐานการกรองข้อมูลด้วย Entity Framework 241
สรุปท้ายบท 244

บทที่ 11 การจัดการข้อมูลในฐานข้อมูลด้วย Entity Framework . 245

พื้นฐานการเพิ่มและแก้ไขข้อมูลโดยการทำ Transaction 245
การเพิ่มและลบข้อมูลแบบ 1 to Many (Master-Details) 257
การทำแบบฟอร์มป้อนข้อมูล (Data Entry) โดยอาศัย Entity Framework 262
การแก้ไขข้อมูลระหว่างฟอร์ม 275
สรุปท้ายบท 282

บทที่ 12 พื้นฐานการพัฒนา Web Application ด้วย ASP.NET MVC 5 283

HTML5 โครงสร้างพื้นฐานของเว็บเพจ 284
วิธีการเข้ารหัสหน้าเว็บเพจให้สามารถแสดงผลภาษาไทย 284
พื้นฐานการสร้างโปรเจ็กต์แบบ ASP.NET MVC 5 285
การทดสอบรันโปรเจ็กต์ 286
หลักการทำงานของ ASP.NET MVC 5 288
การกำหนดส่วนแสดงผลหน้าแรกของ ASP.NET MVC 5 289
พื้นฐานการแสดงผลด้วยเมธอด View() 291
สรุปท้ายบท 293

บทที่ 13 การใช้งาน Controller 295

การสั่งให้แสดงผลไฟล์ cshtml ด้วย Controller 295
การสร้างเมธอดใน Controller เพื่อแสดงผลไฟล์ cshtml 298
พื้นฐานการสร้างลิงค์ในส่วนแสดงผลด้วย Html.ActionLink() 300
การแสดงข้อความจาก Controller โดยตรง 302
การส่งค่าให้กับเมธอดแบบมีพารามิเตอร์ 303
การส่งค่าให้กับพารามิเตอร์แบบ 2 ตัว 306
การสร้าง Controller ใหม่ 308

การตั้งชื่อเมธอดด้วยแอตทริบิวต์ ActionName 312
สรุปท้ายบท 314

บทที่ 14 การทำงานของ ASP.NET MVC ร่วมกับระบบฐานข้อมูลด้วย Entity Framework 315

การแสดงข้อมูลแบบตารางด้วย Entity Framework 315
การสร้างคอลัมน์แบบเรียงลำดับข้อมูลได้ 324
การสร้างส่วนแสดงผลแบบค้นหาข้อมูล 331
สรุปท้ายบท 337

บทที่ 15 การเตรียมสภาพแวดล้อมสำหรับพัฒนา Universal Apps 339

การลงทะเบียนขอ Microsoft account 339
การขอสิทธิ์ชั่วคราวในการพัฒนา Windows 10 Universal Apps 340
เริ่มต้นสร้างแอพฯ แรกให้กับ Windows 10 341
การทดสอบรันโปรเจ็กต์ของ Windows 10 Apps 343
- การรันโปรเจ็กต์สำหรับหน้าจอใหญ่ (PC, Notebook และแท็บเล็ต) 343
- การรันโปรเจ็กต์สำหรับหน้าจอเล็ก (มือถือที่ติดตั้ง Windows 10 for phones) 344

ข้อผิดพลาดของการทดสอบโปรเจ็กต์แบบ Windows 10 for phones 346
การอ้างอิงคอนโทรลหรืออิลิเมนต์ 347
วิธีการเขียนอิลิเมนต์ของ XAML 347
การกำหนดเพจเริ่มต้น 348
การจัดตำแหน่งคอนโทรลด้วยคุณสมบัติด้าน Alignment 349
การจัดระยะห่างด้วยคุณสมบัติ Margin 350
การ Copy คอนโทรลด้วยปุ่ม Alt บนคีย์บอร์ด 351
การยกเลิกการแก้ไขคุณสมบัติ 352
การเลือกโฟกัสคอนโทรลที่สนใจอยู่ 353
พื้นฐานการกำหนดสีพื้นหลังให้กับส่วนแสดงผล 354
พื้นฐานการใช้คุณสมบัติที่เกี่ยวกับสี 355
- การลงสีเดียวแบบ Solid color brush 356
- การลงสีแบบไล่โทนสี Gradient 356

ขั้นตอนการเพิ่มเพจใหม่เข้ามาในโปรเจ็กต์ 358
การสร้างเหตุการณ์ในสคริปต์ XAML 360
สรุปท้ายบท 363

บทที่ 16 การสร้างส่วนแสดงผลแบบรายการด้วยกลุ่มคอนโทรลประเภท list 365

การแสดงรายการด้วยคอนโทรล ComboBox 365
การเพิ่มรายการในคอนโทรล ComboBox ด้วยอิลิเมนต์ <ComboBox.Items>... </ComboBox.Items> 368
การสร้างส่วนแสดงผลแบบรายการด้วยคอนโทรล ListBox 369
การแสดงรูปภาพในรายการของคอนโทรล ListBox 373
การเพิ่มและลบรายการในคอนโทรล ListBox 378
สรุปท้ายบท 380

บทที่ 17 การเขียนโปรแกรมแบบขัดจังหวะกับระบบเมนูและไดอะล็อก 381

การแจ้งเตือนแบบ Toast Notification 381
แนวความคิดของการเขียนโปรแกรมแบบขัดจังหวะได้ (Asynchronous Programming) 385
การสร้างกรอบข้อความด้วยคลาส MessageDialog 386
การใช้ MessageDialog แบบมีการตรวจสอบปุ่มที่ถูกเลือก 388
การสร้าง MessageDialog ก่อนออกจาก App 390
การสร้างเมนูจากการคลิกขวา (ContextMenu) 391
การแสดงข้อมูลแบบ Popup ด้วยคอนโทรล Popup 393
พื้นฐานการสร้างแถบเมนูแบบ AppBar 395
การสร้างเมนูให้กับแถบ AppBar 397
สรุปท้ายบท 400

บทที่ 18 การพัฒนา Windows Store Apps แบบหลายหน้าจอ 401

การสร้างส่วนแสดงผลแบบเปลี่ยนหน้าได้ 401
การรับ-ส่งข้อมูลระหว่างเพจ 404
การรักษาสถานะข้อมูลด้วยระบบแคช (Cache) 408
การสร้างส่วนแสดงผลนำเสนอข้อมูลด้วยคอนโทรล FlipView 410
สรุปท้ายเล่ม 420

CHAPTER 01

เตรียมความพร้อมก่อนเริ่มต้นเขียนโปรแกรมด้วยภาษา Visual C# 2015

โลกของการเขียนโปรแกรมไม่ได้มีแต่เพียงแค่พัฒนาโปรแกรมบน PC เพียงอย่างเดียวเท่านั้น แต่ยังรวมไปถึงมือถือและแท็บเล็ตอีกด้วย

ไมโครซอฟท์พัฒนา Visual Studio 2015 (เรียกสั้นๆ ว่า VS 2015) ขึ้นมา เพื่อรองรับการพัฒนาโปรแกรม 3 ช่องทาง คือ

1. พัฒนาโปรแกรมสำหรับ PC (หรือ Notebook)
2. พัฒนาโปรแกรมบนเว็บ (ไม่ขึ้นกับแพลตฟอร์ม)
3. พัฒนาโปรแกรมสำหรับมือถือและแท็บเล็ต (iOS, Android และ Windows Apps)

ในปัจจุบันไมโครซอฟท์พัฒนา Windows 10 ขึ้นมา ทำให้โลกของโปรแกรมบน PC, มือถือและแท็บเล็ตกลายเป็นโลกเดียวกัน ส่งผลให้นักพัฒนาไม่ต้องเสียเวลาเรียนรู้ซ้ำซ้อนหลายขั้นตอนอีกต่อไป

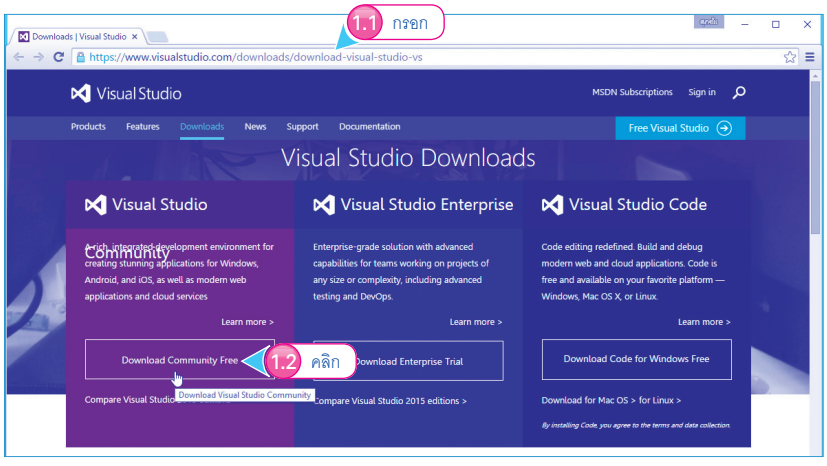
เนื้อหาที่จะนำเสนอในหนังสือเล่มนี้ครอบคลุมการพัฒนาโปรแกรมบน Desktop, Web, มือถือและแท็บเล็ต โดยอาศัยภาษา Visual C# 2015 (เรียกสั้นๆ ว่า VC# 2015) ที่มีเนื้อหาสาระเพียงพอที่จะต่อยอดเป็นโปรแกรมเมอร์มืออาชีพในลำดับต่อไปได้ไม่ยาก

การดาวน์โหลดและติดตั้ง Visual Studio 2015

ผู้อ่านสามารถเขียนโปรแกรมในตระกูล .NET ได้โดยการใช้โปรแกรมที่ชื่อว่า Visual Studio ซึ่งมีขั้นตอนดังนี้

1. ให้ผู้อ่านไปที่เว็บไซต์ https://www.visualstudio.com/downloads/download-visual-studio-vs เพื่อดาวน์โหลดโปรแกรม Visual Studio เวอร์ชันล่าสุดเท่าที่มีอยู่ในปัจจุบัน ดังรูปที่ 1-1

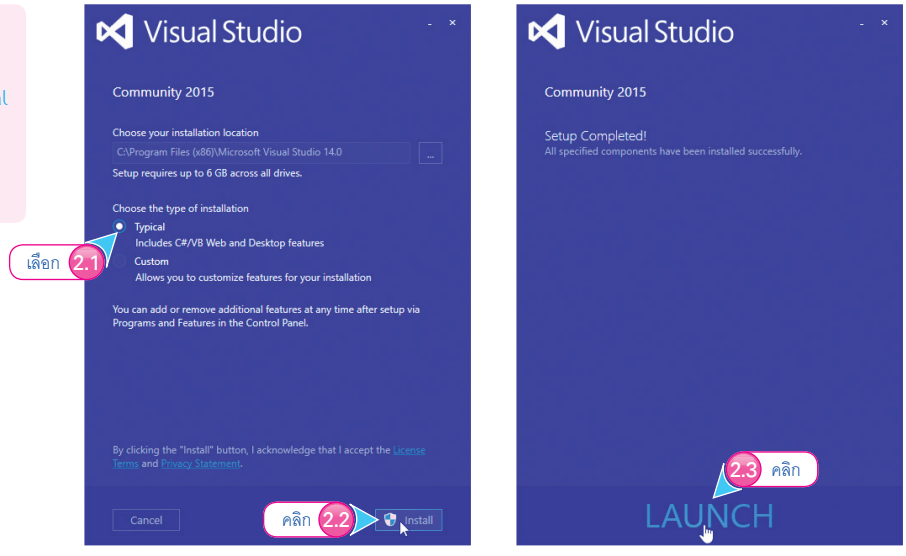
รูปที่ 1-1 แสดงหน้าเว็บเพจที่ให้ดาวน์โหลด Visual Studio



จากรูปที่ 1-1 ขอให้ผู้อ่านดาวน์โหลดเวอร์ชัน Community Edition มาใช้งานได้ฟรี

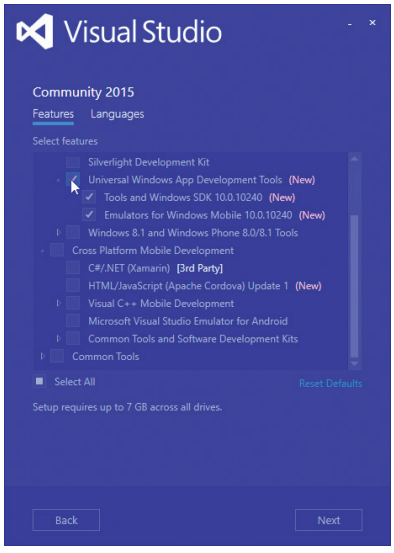
2. การติดตั้ง Visual Studio แบบ Typical เป็นการติดตั้งแบบปกติ (ไม่รวมโปรเจ็กต์ประเภท Windows 10 Apps)

รูปที่ 1-2 แสดงการติดตั้งโปรแกรม Visual Studio 2015 Community Edition



3. ในกรณีที่ผู้อ่านต้องการพัฒนา Windows 10 Apps ด้วย ขอให้ติดตั้ง Visual Studio 2015 บน Windows 10 และเลือกติดตั้งแบบ Custom

รูปที่ 1-3
กรณีติดตั้ง
Visual Studio
แบบ Custom



จากรูปที่ 1-3 ที่รายการ Universal Windows App Development Tools ให้ผู้อ่านเลือกติดตั้งเพิ่มเติม คือตัว Windows 10 SDK กับ Emulators สำหรับรันมือถือ Windows 10

การตั้งค่า Visual Studio 2015 เบื้องต้น

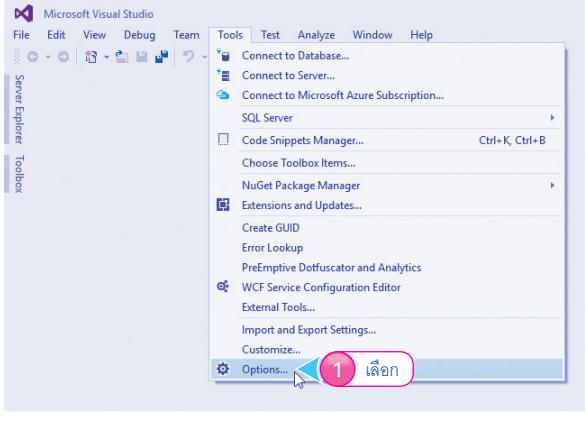
ก่อนเริ่มต้นเรียนรู้การพัฒนาแอพพลิเคชันด้วย Visual Studio 2015 ผู้เขียนอยากจะแนะนำให้ผู้อ่านรู้จักการตั้งค่าต่าง ๆ ที่ช่วยให้การพัฒนาแอพพลิเคชันต่าง ๆ ง่ายขึ้นดังนี้

การกำหนดให้แสดงไดอะล็อกเลือกโปรเจ็กต์ (New Project Dialog)

เพื่อเป็นการอำนวยความสะดวกในการสร้างโปรเจ็กต์ใหม่ทุกครั้ง ผู้อ่านสามารถกำหนดให้ VS แสดงไดอะล็อกเลือกสร้างโปรเจ็กต์ทุกครั้งเมื่อมีการเปิด Visual Studio 2015 ขึ้นมาโดยมีขั้นตอนดังนี้

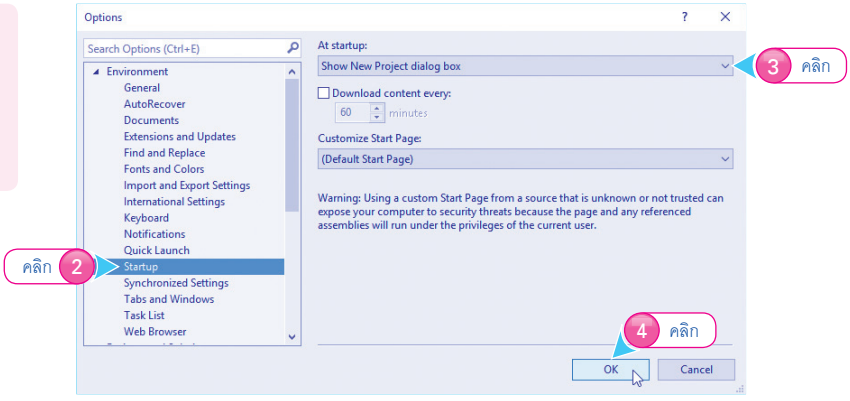
1. คลิกเมนู Tools > Options...

รูปที่ 1-4
แสดงการเลือก
รายการ Show
New Project
dialog box



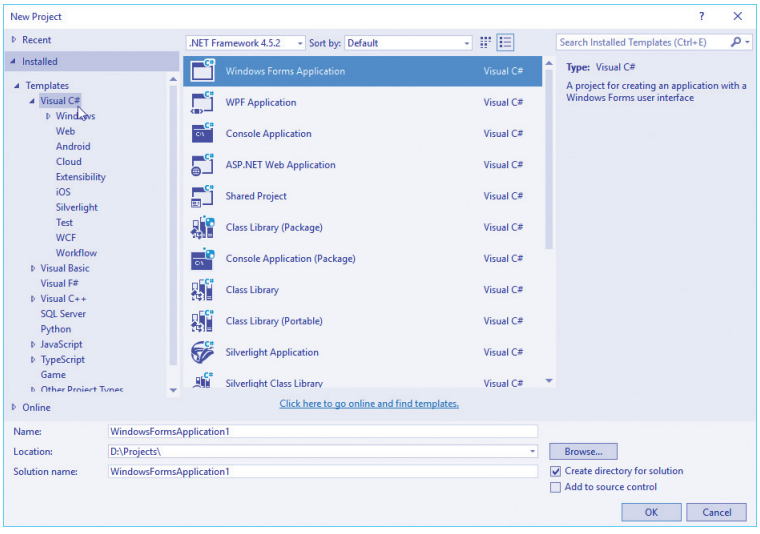
2. จะปรากฏไดอะล็อก Options ให้คลิกเลือก Environment > Startup
3. เลือกรายการ Show New Project dialog box
4. คลิกปุ่ม OK

รูปที่ 1-4 (ต่อ) แสดงการเลือกรายการ Show New Project dialog box



5. จะปรากฏไดอะล็อกสำหรับสร้างโปรเจ็กต์ใหม่ดังรูป

รูปที่ 1-5 แสดงไดอะล็อกเลือกสร้างโปรเจ็กต์ใหม่



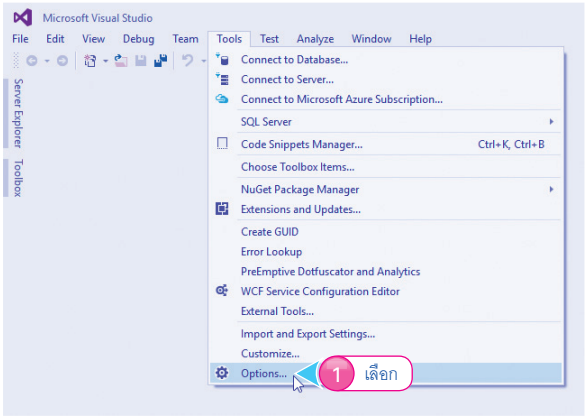
จากรูปที่ 1-5 ผู้อ่านสามารถเลือกสร้างโปรเจ็กต์ใน VS ได้ตามที่ต้องการได้จากไดอะล็อกนี้

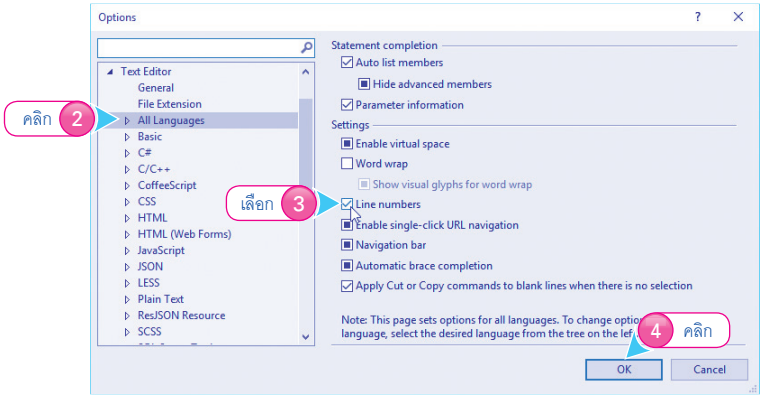
การกำหนดเลขบรรทัด

การกำหนดให้ Visual Studio 2015 แสดงหมายเลขกำกับโค้ดในแต่ละบรรทัด ผู้อ่านสามารถทำได้โดยมีขั้นตอนดังนี้

1. คลิกเมนู Tools > Options
2. จะปรากฏไดอะล็อก Options ให้คลิกเลือก Text Editor > All Languages
3. คลิกเลือก Line numbers
4. คลิกปุ่ม OK

รูปที่ 1-6 รูปที่ 1-6 แสดงการกำหนดให้แสดงหมายเลขกำกับโค้ดแต่ละบรรทัด





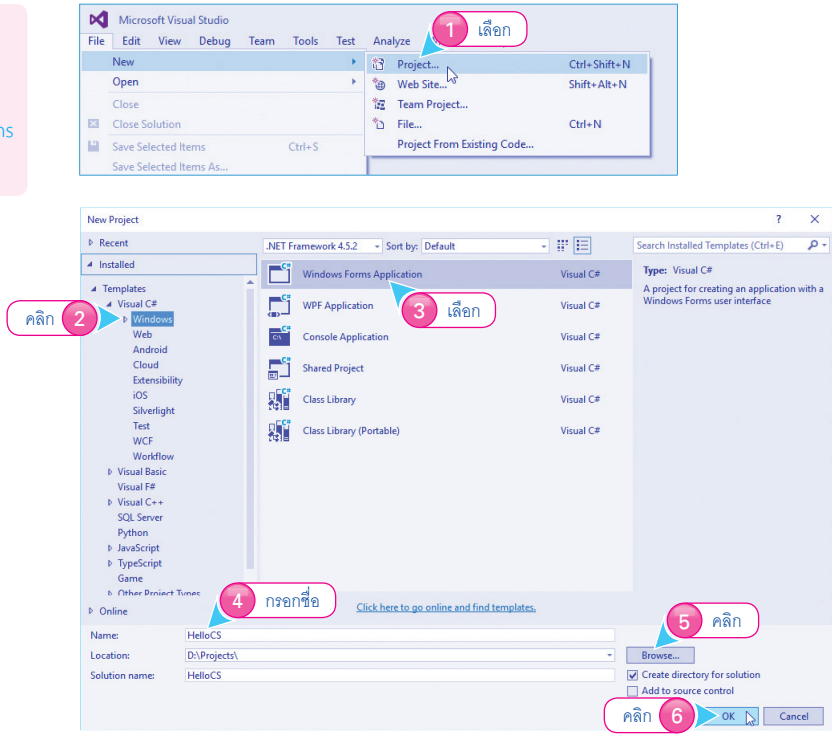
การสร้างโปรเจ็กต์ใหม่

การสร้างโปรเจ็กต์ใหม่ถือเป็นจุดเริ่มต้นของการพัฒนาแอพพลิเคชันต่าง ๆ ในที่นี้จะเริ่มต้นจาก Windows Forms Application ซึ่งเป็นโปรเจ็กต์หลักของ Visual C# 2015 โดยมีขั้นตอนดังนี้

1. ให้ผู้อ่านคลิกเมนู File > New > Project...
2. เลือกเมนู Visual C# > Windows
3. เลือกสร้างโปรเจ็กต์แบบ Windows Forms Application ซึ่งเป็นโปรเจ็กต์ที่ใช้สร้างโปรแกรมสำหรับทำงานบน Windows
4. กรอกชื่อโปรเจ็กต์

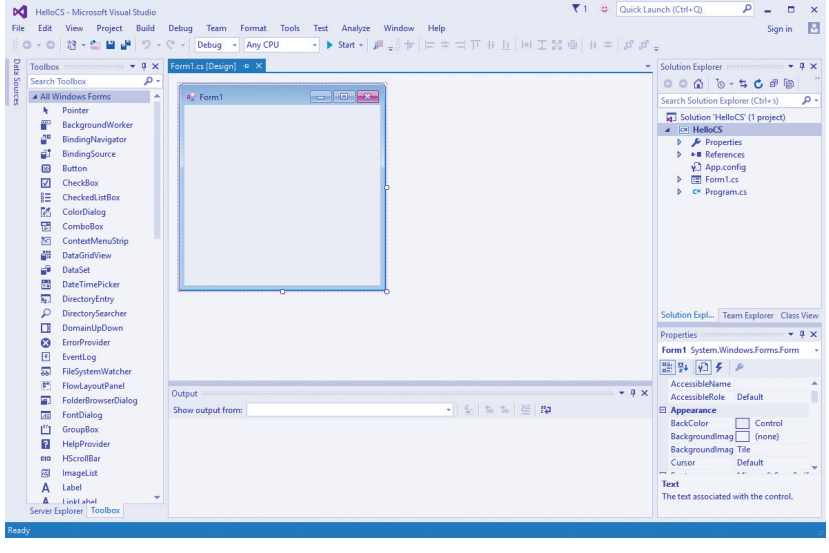
5. เลือกตำแหน่งที่อยู่ของโปรเจ็กต์
6. คลิกปุ่ม OK

รูปที่ 1-7 แสดงการสร้างโปรเจ็กต์แบบ Windows Forms Application



7. จะปรากฏหน้า Windows Forms Application สำหรับสร้างโปรเจ็กต์ดังรูป

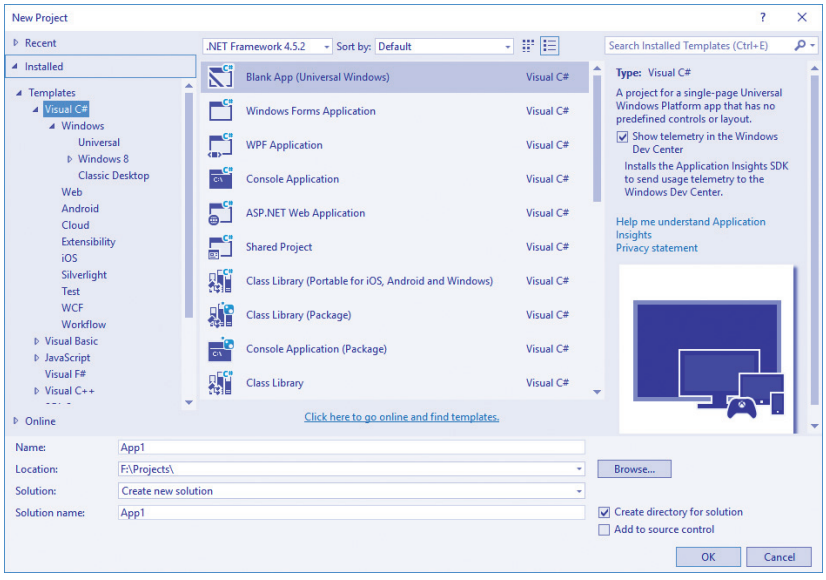
รูปที่ 1-8 แสดงสภาพแวดล้อมของ VS แบบ Windows Forms Application



ทำความรู้จักกับโปรเจ็กต์ประเภทอื่นๆ ของ Visual Studio 2015

นอกเหนือไปจากโปรเจ็กต์ประเภท Windows Forms Application แล้ว ตัวโปรแกรม Visual Studio 2015 ยังรองรับการพัฒนาแอพฯ ประเภทต่าง ๆ อีกมากมาย

รูปที่ 1-9 แสดงไดอะล็อกสร้างโปรเจ็กต์ประเภทต่างๆ ของ Visual Studio 2015



โปรเจ็กต์ประเภท Windows

โปรเจ็กต์ประเภท Windows ยังถูกแบ่งออกเป็น 3 ประเภท คือ

- Universal เป็นการพัฒนาแอพฯ ที่รันบนระบบปฏิบัติการ Windows 10 โดยที่ผู้อ่านสามารถสร้างโปรเจ็กต์ขึ้นมาเพียง 1 โปรเจ็กต์ แอพฯ ที่ได้มาจะรันได้ทั้ง PC, แท็บเล็ตและมือถือ
- Windows 8 เป็นการพัฒนาแอพฯ ที่รันบนระบบปฏิบัติการ Windows 8/8.1 เท่านั้น
- Classic Desktop เป็นการพัฒนาแอพฯ แบบเดิมที่เราคุ้นเคยกันเป็นอย่างดี แอพฯ ที่ได้มาก็คือไฟล์ exe สามารถติดตั้งในระบบปฏิบัติการ Windows ต่าง ๆ ที่มีการติดตั้ง .NET Framework ตามเวอร์ชันที่เราเป็นผู้สร้างโปรเจ็กต์ขึ้นมา

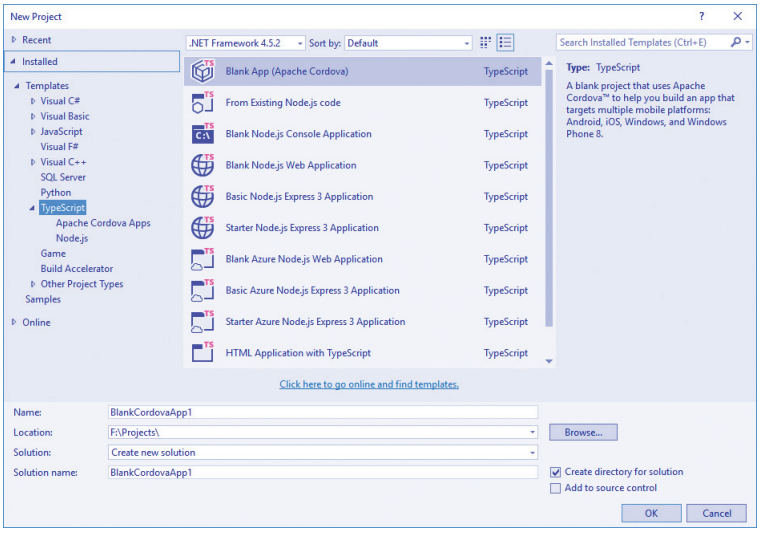
โปรเจ็กต์ประเภท Web

ถ้าคุณต้องการสร้างเว็บไซต์ต่าง ๆ เช่น เว็บแสดงรูปภาพ, เว็บบอร์ด, เว็บข่าวสาร ฯลฯ แสดงผลในบราวเซอร์ เราถือว่าเป็นโปรเจ็กต์ประเภท Web อยู่ในความรับผิดชอบของภาษา ASP.NET

โปรเจ็กต์ประเภท JavaScript

เดิมทีเดียวภาษา JavaScript ถูกนำมาใช้พัฒนาด้าน Web Application เพียงอย่างเดียวเท่านั้น แต่ในปัจจุบันภาษา JavaScript ถูกนำมาพัฒนาแอพฯ ได้ทั้งแบบ Web Apps กับ Mobile Apps อีกด้วย

รูปที่ 1-10 แสดงโปรเจ็กต์ประเภท JavaScript



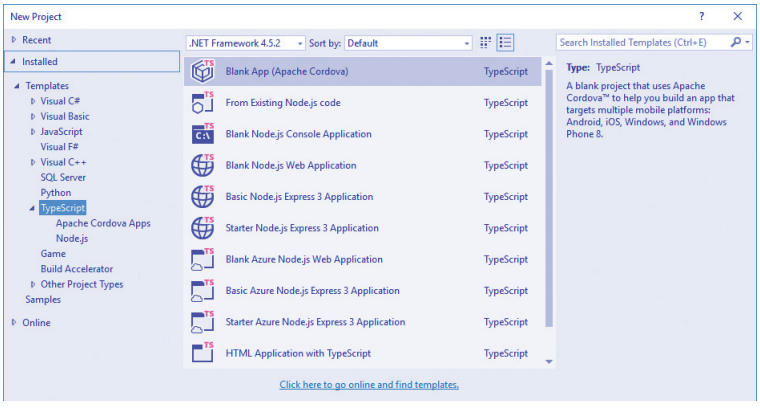
จากรูปที่ 1-10 ภาษา JavaScript ใน Visual Studio 2015 ในขั้นต้นสามารถสร้างโปรเจ็กต์ได้ 3 แบบ คือ

- **Apache Cordova Apps** เป็นการใช้งานภาษา JavaScript ร่วมกับภาษา HTML5 เพื่อสร้างแอพฯ ที่สามารถรันบนมือถือโดยอาศัย Cordova (อาจจะเรียกว่า PhoneGap ก็ได้)
- **Node.js** เป็นการพัฒนา Web Apps ที่รันได้ทั้งฝั่ง Server/Client โดยอาศัยภาษา JavaScript เพียงภาษาเดียว
- **Windows – Universal** เป็นการพัฒนาแอพฯ ที่รันบน Windows 10 (PC, แท็บเล็ต และมือถือ) โดยอาศัยภาษา HTML5 ร่วมกับภาษา JavaScript

โปรเจ็กต์ประเภท TypeScript

ภาษา TypeScript เป็นภาษาใหม่ที่ไมโครซอฟท์สร้างขึ้นมา โดยมีหลักการที่ว่าทุกสิ่งทุกอย่างของภาษา JavaScript คือภาษา TypeScript และเพิ่มเติมความสามารถอื่นๆ ในตัวภาษา TypeScript อีกด้วย เช่น การกำหนดประเภทข้อมูล, การสร้างคลาสขึ้นมาใช้งานเองได้ เป็นต้น เป็นภาษาสคริปต์ที่กำลังได้รับความนิยมเป็นอย่างมาก

รูปที่ 1-11 แสดงโปรเจ็กต์ประเภท TypeScript



จากรูปที่ 1-11 ภาษา TypeScript ใน Visual Studio 2015 ในขั้นต้นสามารถสร้างโปรเจ็กต์ได้ 2 แบบ คือ

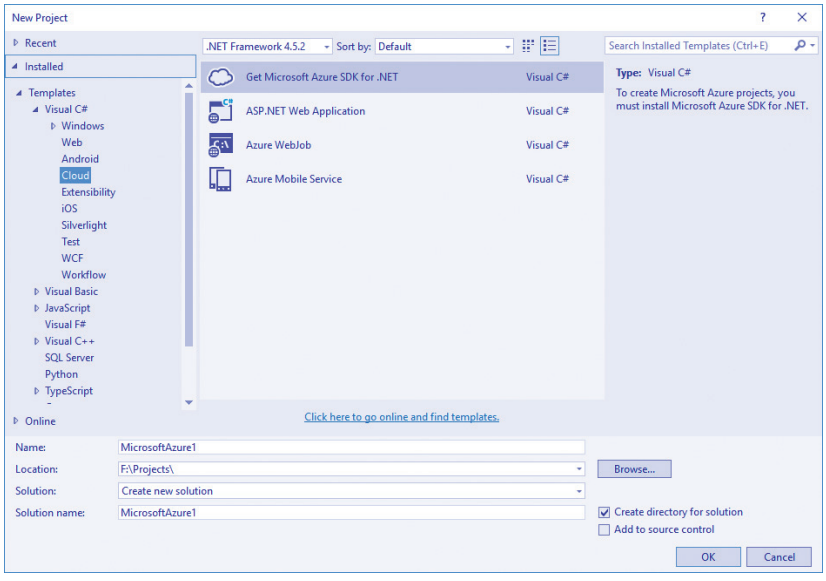
- **Apache Cordova Apps** เป็นการใช้งานภาษา TypeScript ร่วมกับภาษา HTML5 เพื่อสร้างแอพฯ ที่สามารถรันบนมือถือโดยอาศัย Cordova (อาจจะเรียกว่า PhoneGap ก็ได้)
- **Node.js เป็นการพัฒนา Web Apps** ที่รันได้ทั้งฝั่ง Server/Client โดยอาศัยภาษา TypeScript เพียงภาษาเดียว

โปรเจ็กต์ประเภท Cloud

ในปัจจุบันการขอใช้บริการต่าง ๆ บนก้อนเมฆ (Cloud) ได้รับความนิยมเป็นอย่างมาก บริการ Cloud ของไมโครซอฟท์ชื่อว่า Azure เราสามารถสร้างโปรเจ็กต์ได้ 2 รูปแบบใหญ่ ๆ คือ

- **โปรเจ็กต์ประเภท ASP.NET Web Application** เป็นการพัฒนา Web Apps ด้วยภาษา ASP.NET เพื่อขอใช้บริการต่าง ๆ ใน Azure
- **โปรเจ็กต์ประเภท Azure Mobile Service** เป็นการสร้างบริการที่เชื่อมต่อกับ Azure ให้บริการด้านต่าง ๆ โดยอาศัยโครงสร้างของ Web API สำหรับอุปกรณ์ประเภท Mobile Devices ต่าง ๆ

รูปที่ 1-12 แสดงโปรเจ็กต์ประเภท Cloud

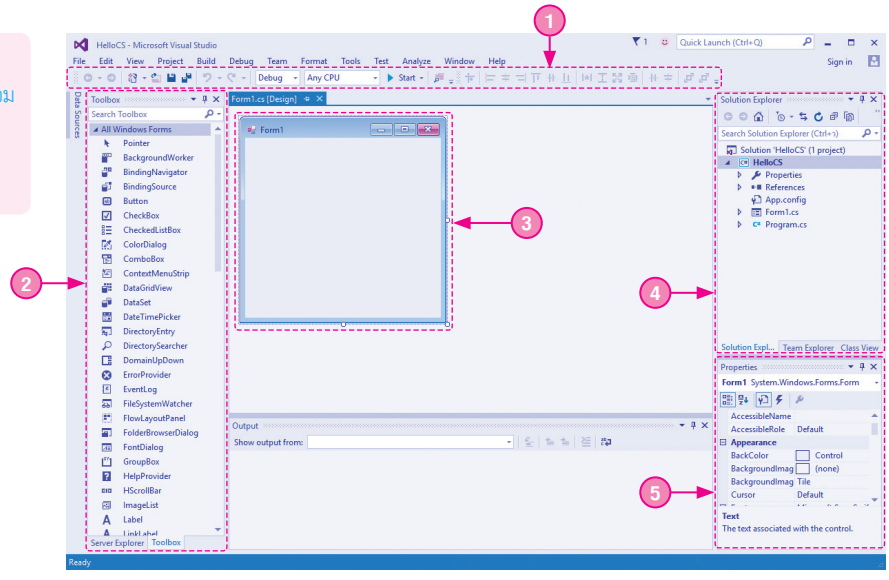


ประเภทโปรเจ็กต์ที่ผู้เขียนนำเสนอรายละเอียดในขั้นต้นนี้ เป็นการติดตั้งเพิ่มเติมแยกต่างหาก เพื่อแสดงให้เห็นถึงความสามารถของตัวโปรแกรม Visual Studio 2015 ที่มีอยู่มากมาย

ทำความรู้จักกับสภาพแวดล้อม (IDE) ของ VS ในขั้นต้น

VS 2015 รองรับการพัฒนาแอพฯ หลายแพลตฟอร์ม (PC, มือถือ และแท็บเล็ต) โดยที่ผู้อ่านสามารถเลือกสร้างโปรเจ็กต์ได้ตามความต้องการของผู้อ่าน สภาพแวดล้อมของโปรเจ็กต์แต่ละประเภทมีลักษณะคล้ายกัน เพื่อให้นักพัฒนาคุ้นเคยนั่นเอง

ผู้เขียนขอยกตัวอย่างสภาพแวดล้อมของโปรเจ็กต์ประเภท Windows Forms Application มาอธิบายดังรูปที่ 1-13



รูปที่ 1-13 แสดงสภาพแวดล้อม IDE ของ VS แบบ Windows Forms Application

- หมายเลข 1 แถบเครื่องมือสำหรับทดสอบโปรเจ็กต์ปัจจุบัน
- หมายเลข 2 แถบคอนโทรลสำเร็จรูป ทำหน้าที่ออกแบบส่วนแสดงผล (User Interface เรียกย่อ ๆ ว่า UI) ภายในโปรเจ็กต์ของผู้อ่าน
- หมายเลข 3 ส่วนแสดงผลจำลอง ผู้อ่านสามารถออกแบบส่วนแสดงผลให้กับหน้าจอต่าง ๆ ได้ที่นี่ โดยการลากคอนโทรลต่าง ๆ มาวาดตามที่ต้องการ
- หมายเลข 4 เรียกว่า หน้าต่าง Solution Explorer ทำหน้าที่แสดงโครงสร้างโปรเจ็กต์ปัจจุบันของผู้อ่านว่าประกอบไปด้วยไฟล์อะไรบ้าง
- หมายเลข 5 เรียกว่า หน้าต่างคุณสมบัติ (Properties) ทำหน้าที่แสดงคุณสมบัติต่าง ๆ ของคอนโทรลที่กำลังถูกโฟกัสในส่วนแสดงผลจำลองว่ามีคุณสมบัติอะไรบ้าง ผู้อ่านสามารถแก้ไขคุณสมบัติต่าง ๆ ได้ตามที่ต้องการเช่นกัน เช่น ตั้งชื่อคอนโทรลได้ที่คุณสมบัติ Name, กำหนดสีพื้นหลังได้ที่คุณสมบัติ BackColor เป็นต้น

มือใหม่ต้องทราบก่อนเขียนโปรแกรมในโลกของ .NET

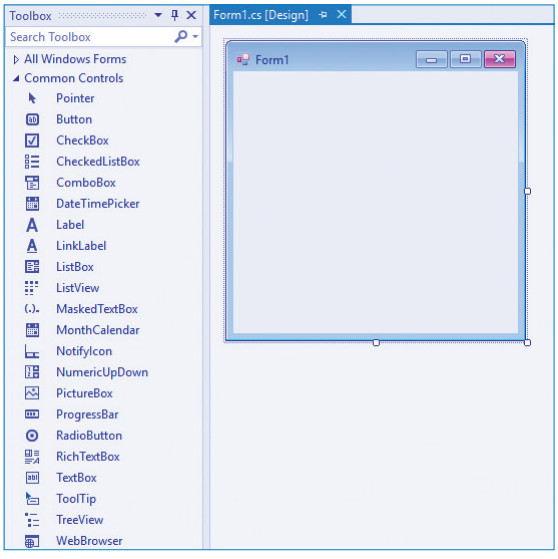
ก่อนที่จะเข้าสู่รายละเอียดของการเขียนโปรแกรมในโลกของ .NET Frawework ผู้เขียนจะขอนำเสนอหลักการพื้นฐานในภาพกว้างที่มือใหม่ต้องทราบในขั้นต้นเสียก่อน เพื่อปูพื้นฐานก่อนที่จะเข้าสู่รายละเอียดเนื้อหาส่วนต่าง ๆ ในแต่ละบทต่อไป

การออกแบบส่วนแสดงผลด้วยคอนโทรล (User Interface)

ไม่ว่าผู้อ่านจะสร้างโปรเจ็กต์ประเภทใดก็ตามใน Visual Studio 2015 ไมโครซอฟท์กำหนดให้สภาพแวดล้อมของตัวโปรแกรม Visual Studio 2015 มีลักษณะคล้ายกัน ใกล้เคียงกันมากที่สุดเท่าที่จะเป็นไปได้ เพื่อให้นักพัฒนาคุ้นเคยกับตัวโปรแกรมได้ไม่ยาก

แม้ว่าภาพประกอบผู้เขียนจะใช้โปรเจ็กต์ประเภท Windows Forms Application ก็จริงอยู่ แต่สามารถเทียบเคียงได้กับโปรเจ็กต์ประเภทอื่น ๆ ได้อีกด้วย ซึ่งจะเป็นเนื้อหาในภาคต่าง ๆ ในลำดับต่อไป

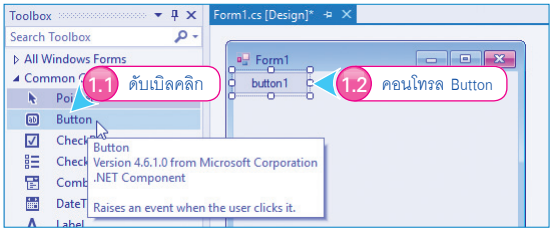
ก่อนอื่นขอให้ผู้อ่านสร้างโปรเจ็กต์แบบ Windows Forms Application ว่าง ๆ ขึ้นมา 1 โปรเจ็กต์ ดังรูปที่ 1-14



รูปที่ 1-14 แสดงแถบ Toolbox

จากรูปที่ 1-14 แถบคอนโทรล (Control) อยู่ด้านซ้ายมือ ทำหน้าที่ออกแบบส่วนแสดงผล (User Interface เรียกย่อ ๆ ว่า UI) ในขั้นต้นนี้ผู้อ่านสามารถใช้งานคอนโทรลได้ 2 วิธี

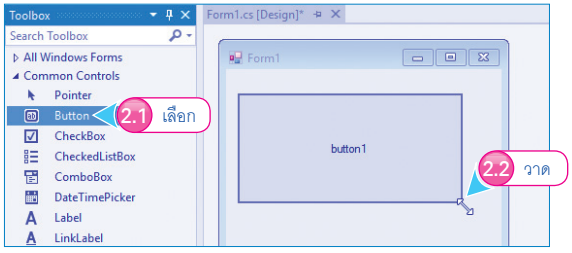
1. ดับเบิลคลิกที่ตัวคอนโทรลเพื่อกำหนดให้ Visual Studio ใช้งานคอนโทรลดังกล่าวในหน้าจอออกแบบโดยอัตโนมัติ เช่น ผู้เขียนต้องการใช้ปุ่มกด Button1 ดังรูปที่ 1-15

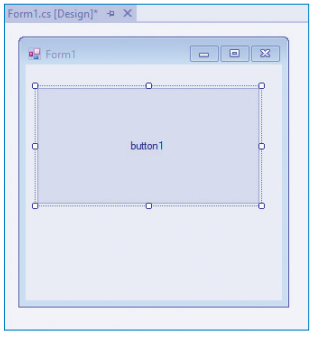


รูปที่ 1-15 แสดงการใช้ปุ่มกดด้วยวิธีการดับเบิลคลิกในแถบ Toolbox

2. คลิกเลือกคอนโทรลที่ต้องการใช้งานแล้วนำไปวาดในส่วนของฟอร์ม (Form) ดังรูปที่ 1-16

รูปที่ 1-16 แสดงการใช้ปุ่มกดด้วยวิธีวาด



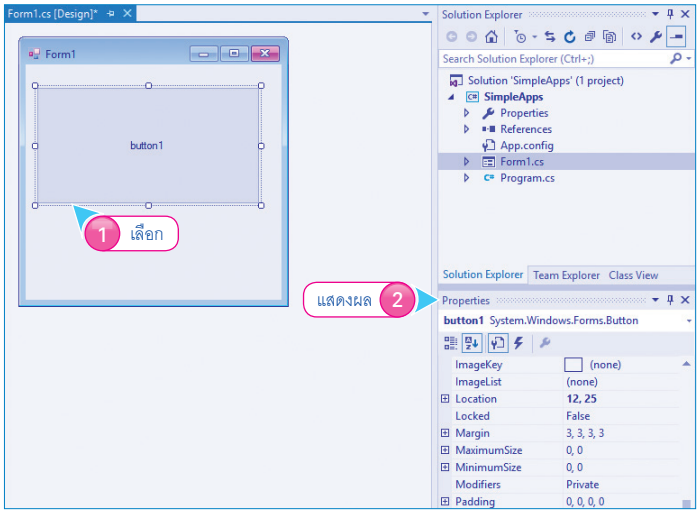


จากรูปที่ 1-16 เราได้ส่วนแสดงผลง่ายๆ มาแล้ว 1 หน้าจอ มีปุ่มกด Button เพียง 1 ปุ่ม

การปรับแต่งคอนโทรลขั้นต้นด้วยวิธีแก้ไขคุณสมบัติ (Property)

คอนโทรลต่างๆ ที่นำมาใช้งานสามารถปรับแต่งหรือแก้ไขได้ในหน้าต่างคุณสมบัติ (Properties) กล่าวคือ เมื่อโฟกัสที่คอนโทรลตัวใดก็ตามหน้าต่างคุณสมบัติจะเปลี่ยนไปโดยอัตโนมัติ

รูปที่ 1-17 แสดงรายการคุณสมบัติของปุ่มกด Button



จากรูปที่ 1-17 ส่วนแสดงผลปัจจุบันของผู้เขียนมีเพียง 1 ปุ่ม เมื่อถูกโฟกัสหน้าต่างคุณสมบัติก็จะแสดงรายการคุณสมบัติต่างๆ ของคอนโทรล Button ให้เราโดยอัตโนมัติ

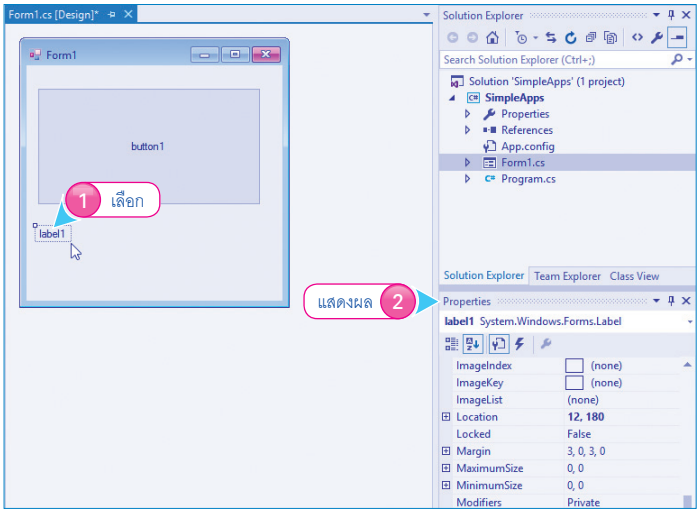
รายการคุณสมบัติของคอนโทรล Button ที่น่าสนใจ แสดงดังตารางต่อไปนี้

คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับตัวคอนโทรล ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงในปุ่มกด
BackColor	กำหนดสีพื้นหลังให้กับปุ่มกด
ForeColor	กำหนดสีตัวอักษรที่อยู่ในปุ่มกด

คุณสมบัติใดก็ตามที่ถูกแก้ไขจะแสดงด้วยตัวอักษรตัวหนา ส่วนตัวอักษรปกติ หมายถึง เป็นค่าเริ่มต้นที่โปรแกรม Visual Studio ตั้งไว้ โดยที่ไม่จำเป็นต้องไปแก้ไขให้ครบทุกคุณสมบัติ แก้ไขเฉพาะเท่าที่จำเป็น

ต่อมาผู้เขียนลองใช้คอนโทรล Label ซึ่งทำหน้าที่แสดงข้อความ เมื่อคอนโทรล Label ได้รับโฟกัสพบว่าที่หน้าต่างคุณสมบัติจะแสดงรายการคุณสมบัติของคอนโทรล Label ให้เราโดยอัตโนมัติ

รูปที่ 1-18 แสดงรายการคุณสมบัติของคอนโทรล Label



รายการคุณสมบัติที่น่าสนใจแสดงดังตารางต่อไปนี้

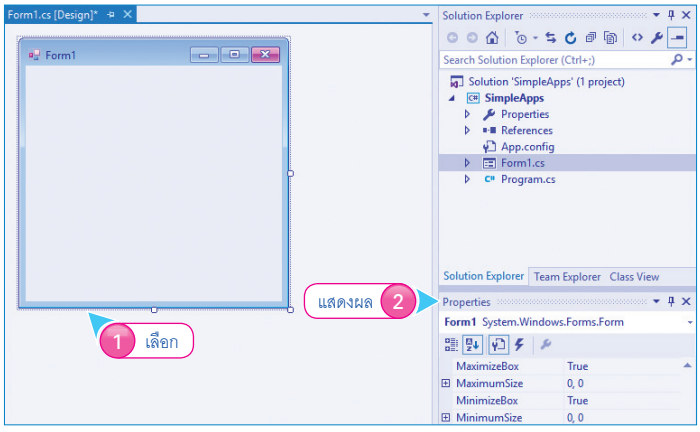
คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับตัวคอนโทรล ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงใน Label
BackColor	กำหนดสีพื้นหลังให้กับคอนโทรล Label
ForeColor	กำหนดสีตัวอักษรที่อยู่ในคอนโทรล Label
AutoSize	กำหนดขนาดของคอนโทรล Label มี 2 แบบ คือ • True หมายถึง กำหนดให้ Label มีขนาดเท่าที่ข้อความแสดงอยู่ • False หมายถึง คุณสามารถกำหนดขนาดของ Label ได้อย่างอิสระ

คอนโทรลแต่ละตัวก็จะมีรายการคุณสมบัติทั้งที่เหมือนกันและต่างกันไปตามหน้าที่ของตัวมันเอง

ทำความรู้จักกับฟอร์ม (Form)

ไม่ว่าจะสร้างโปรเจ็กต์ประเภทใดก็ตาม โดยส่วนใหญ่จะมีส่วนแสดงผลเริ่มต้นมาให้ผู้อ่านใช้ออกแบบหน้าจอแรก (โปรเจ็กต์บางประเภทไม่ต้องมีส่วนแสดงผลเริ่มต้น) เช่น ในกรณีนี้เราสร้างโปรเจ็กต์แบบ Windows Forms Application ก็จะได้ฟอร์ม (Form) ที่มีชื่อว่า Form1 เป็นหน้าจอเริ่มต้น ดังรูปที่ 1-19

รูปที่ 1-19 แสดงรายการคุณสมบัติของฟอร์ม

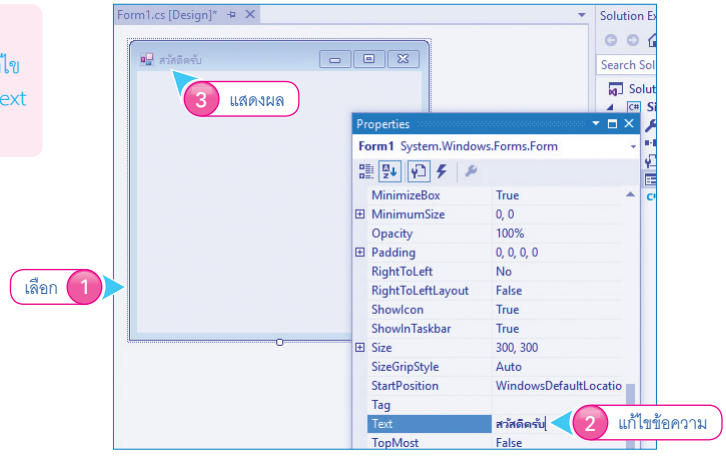


จากรูปที่ 1-19 เมื่อโฟกัสที่ Form1 หน้าต่างคุณสมบัติจะแสดงรายการคุณสมบัติของ Form1 เช่นกัน ที่น่าสนใจก็คือ

คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับฟอร์ม ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงในแถบ Title Bar
BackColor	กำหนดสีพื้นหลังให้กับฟอร์ม
ForeColor	กำหนดสีตัวอักษรที่อยู่ในฟอร์ม

ในที่นี้ทดลองแก้ไขข้อความที่แสดงในแถบ Title Bar สามารถทำได้ดังนี้

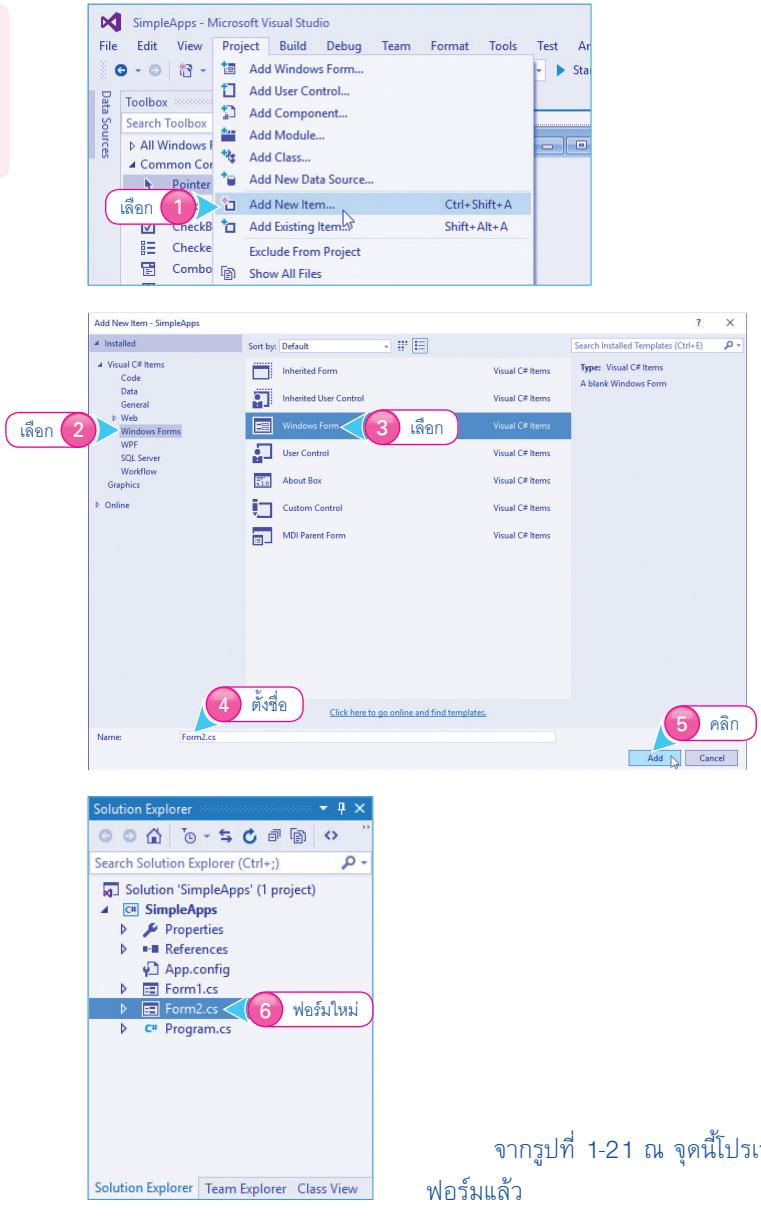
รูปที่ 1-20 แสดงการแก้ไขคุณสมบัติ Text ของ Form1



การเพิ่มฟอร์มใหม่เข้ามาในโปรเจ็กต์

ในการพัฒนาแอพฯ ขึ้นมาใช้งาน โดยส่วนใหญ่จะต้องประกอบไปด้วยส่วนแสดงผลตั้งแต่ 2 หน้าจอขึ้นไป วิธีการเพิ่มหน้าจอใหม่เข้ามา ให้คลิกเมนู Project > Add New Item... ในกรณีนี้เราต้องการเพิ่มหน้าจอใหม่ประเภท Windows Forms Application โดยตั้งชื่อว่า Form2 ดังรูปที่ 1-21

รูปที่ 1-21 แสดงการเพิ่มฟอร์มที่ชื่อว่า Form2 เข้ามาในโปรเจ็กต์



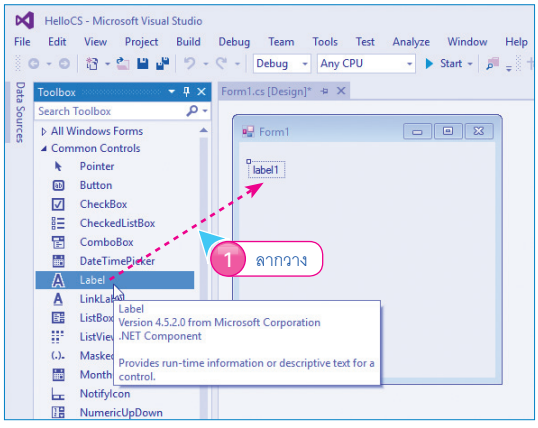
จากรูปที่ 1-21 ณ จุดนี้โปรเจ็กต์ของเรามี 2 ฟอร์มแล้ว

เริ่มต้นสร้างแอพพลิเคชันแรก

ในหัวข้อนี้ผู้อ่านจะได้ใช้งาน Visual Studio 2015 เขียนโปรแกรมด้วยภาษา Visual C# 2015 เป็นครั้งแรก โดยเป้าหมายของแอพพลิเคชันนี้ก็คือแสดงข้อความ “Hello C#” โดยมีขั้นตอนดังนี้

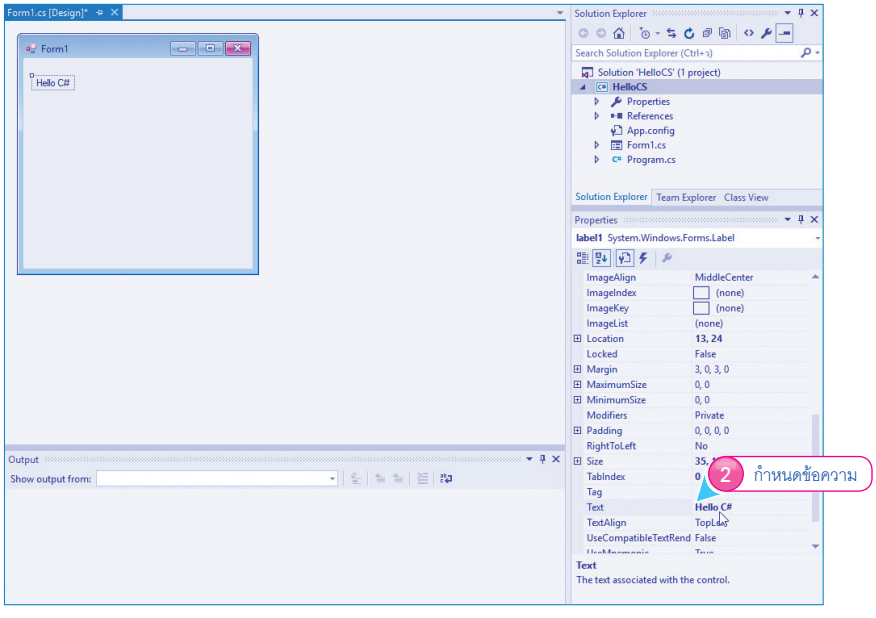
1. ให้ผู้อ่านเลือกใช้คอนโทรล Label เข้ามาทำหน้าที่แสดงข้อความ “Hello C#” โดยการลาก & วางใน Form1 ซึ่งทำหน้าที่เป็นส่วนแสดงผล ดังรูปที่ 1-22

รูปที่ 1-22 แสดงการใช้งานคอนโทรล Label ใน Form1



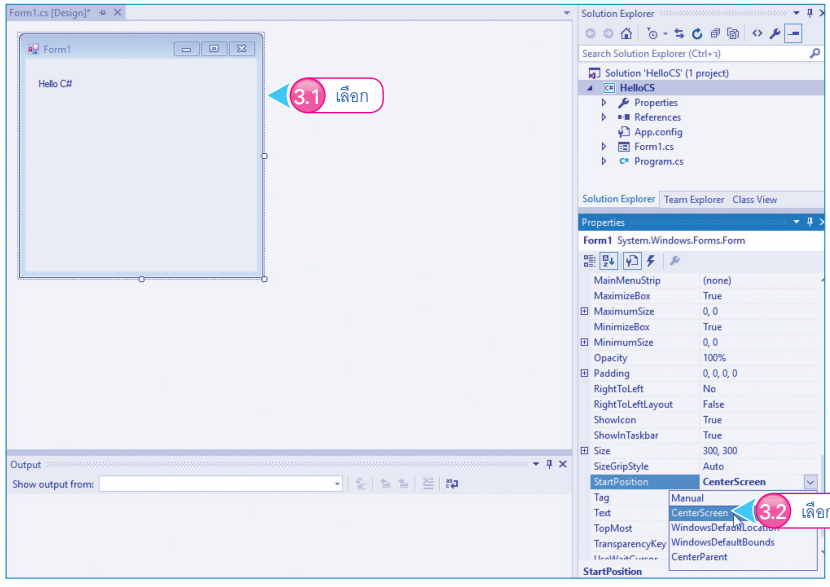
2. การกำหนดข้อความใน Label ตัวนี้ ให้ผู้อ่านแก้ไขคุณสมบัติ Text ได้ที่หน้าต่าง Properties ให้เป็นข้อความตามที่ต้องการ

รูปที่ 1-23 แสดงการแก้ไขคุณสมบัติ Text ของคอนโทรล Label ตัวนี้



3. การกำหนดให้ Form1 ปรากฏอยู่กึ่งกลางหน้าจอที่คุณสมบัติ StartPosition เพราะเป็นคุณสมบัติที่ทำหน้าที่กำหนดตำแหน่งของ Form1 โดยผู้เขียนเลือกแบบ CenterScreen หมายถึง ให้ปรากฏขึ้นมากึ่งกลางหน้าจอ ดังรูปที่ 1-24

รูปที่ 1-24 แสดงการแก้ไขตำแหน่งของฟอร์มให้อยู่กึ่งกลางหน้าจอ

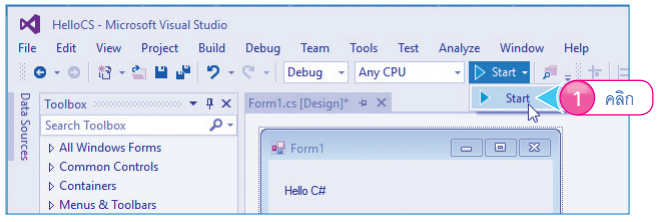


จากรูปที่ 1-24 ณ จุดนี้ เราจะได้โปรเจ็กต์ตามที่ต้องการแล้ว

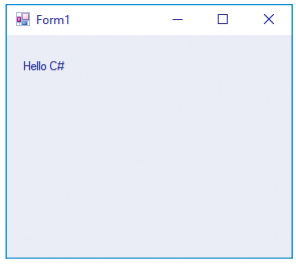
การทดสอบโปรเจ็กต์

หลังจากที่ผู้อ่านออกแบบและเขียนโค้ดแล้วก็จะเข้าสู่ขั้นตอนการทดสอบโปรเจ็กต์ เพื่อดูว่าแอพพลิเคชันที่สร้างขึ้นทำงานตามที่ต้องการหรือไม่ โดยการคลิกปุ่ม ▶ Start เพื่อเริ่มต้นทดสอบรันโปรเจ็กต์

รูปที่ 1-25 แสดงการเริ่มทดสอบโปรเจ็กต์



รูปที่ 1-26 ผลการทำงานของโปรเจ็กต์ปัจจุบัน



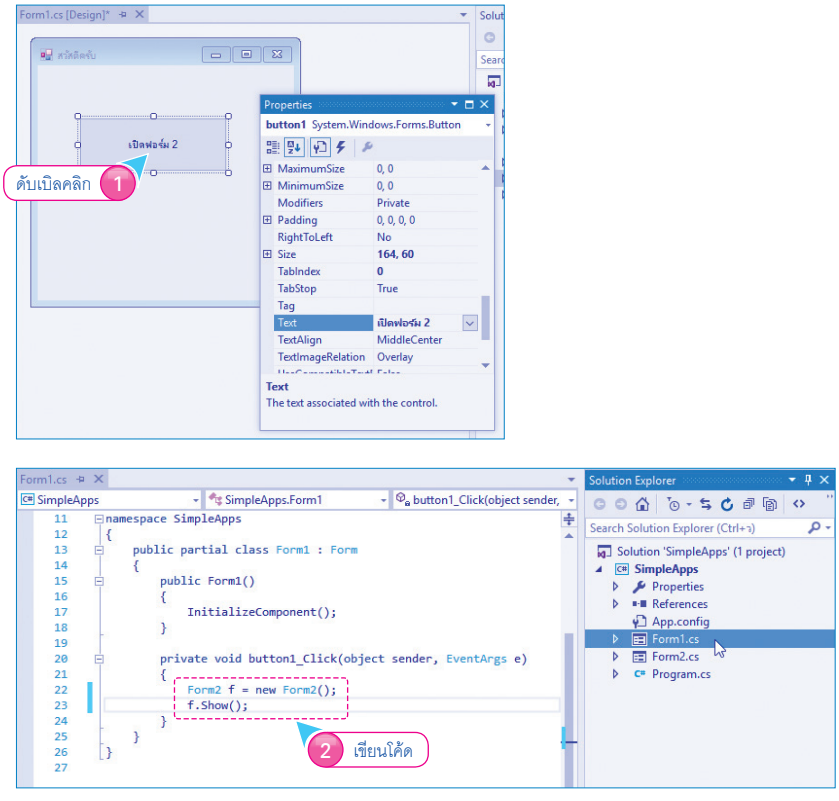
จากรูปที่ 1-26 เห็นได้ว่าเราได้ข้อความ “Hello C#” ปรากฏขึ้นมาใน Form1 ตามที่ต้องการแล้ว

วิธีการสั่งให้เปิดฟอร์มใหม่โดยการเขียนโค้ด

สำหรับวิธีการสั่งให้เปิดฟอร์มใหม่ที่เราเพิ่มเข้ามามีขั้นตอนดังนี้

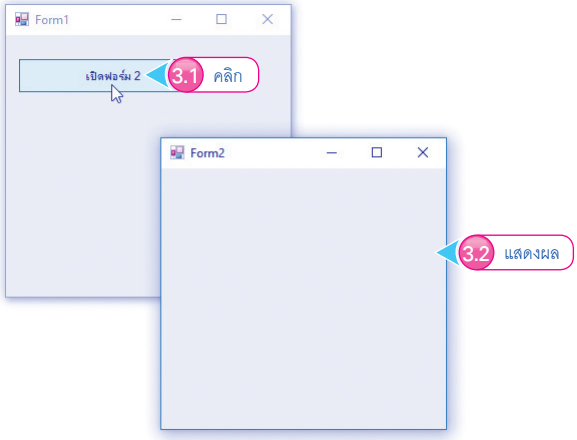
1. ให้ดับเบิลคลิกปุ่ม Button1 เพื่อสร้างเหตุการณ์ Click()
2. เขียนโค้ดดังรูป

รูปที่ 1-27 แสดงโค้ดที่ใช้สำหรับเปิด Form2



3. ให้ทดสอบรันโปรเจ็กต์ พบว่าเราสามารถเปิด Form2 ได้แล้ว ดังรูปที่ 1-28

รูปที่ 1-28 Form2 ที่ถูกเปิดจากปุ่มกด Button1 ใน Form1



หลักการทำงานของภาษา C#

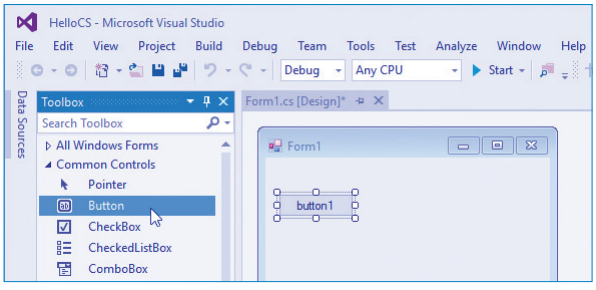
หลักการทำงานของภาษา C# เรียกว่า **การเขียนโปรแกรมตอบสนองเหตุการณ์ที่เกิดขึ้น (Event Driven Programming)** กล่าวคือ ผู้อ่านเป็นผู้กำหนดเองว่าเมื่อเกิดเหตุการณ์ Click() ให้ทำอะไร, เมื่อเกิดเหตุการณ์ Load() ให้ทำอะไร เป็นต้น

หมายความว่า แท้ที่จริงแล้วการกระทำต่างๆ ที่ผู้ใช้งานกระทำต่อโปรแกรมของผู้อ่านเป็นผู้กำหนดขึ้นมาเองทั้งสิ้นว่า “ให้ทำอะไร” ประเด็นที่ตามมาก็คือ วิธีการสร้างเหตุการณ์ให้กับคอนโทรลแต่ละตัวทำอย่างไร

สมมติว่าผู้เขียนต้องการสร้างเหตุการณ์ Click() ให้กับปุ่มกด Button จะมีวิธีการดังนี้

1. ให้ลากปุ่มกด Button มาวางใน Form1 ดังรูปที่ 1-29

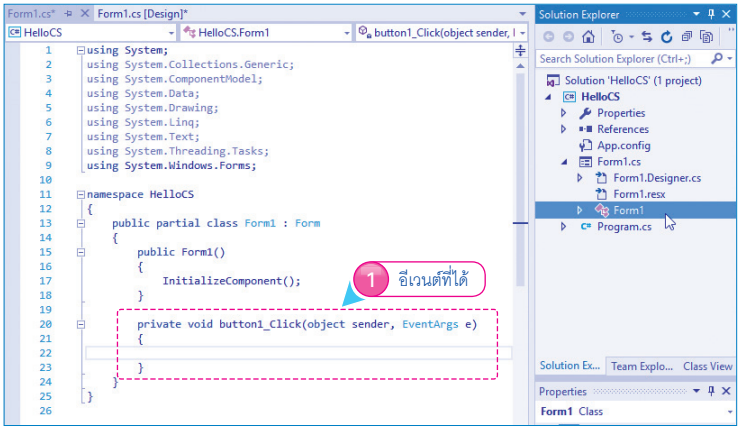
รูปที่ 1-29 แสดงการใช้งานคอนโทรล Button ใน Form1



โดยการสร้างเหตุการณ์ให้กับปุ่มกด Button1 นี้ มี 2 วิธี

1.1 ดับเบิลคลิกคอนโทรล Button1 เพื่อสั่งให้ VS สร้างเหตุการณ์ประจำตัวของคอนโทรล Button ขึ้นมา คอนโทรล Button ถูกสร้างขึ้นมาเพื่อทำหน้าที่รองรับการคลิกจากผู้ใช้งาน ดังนั้น เหตุการณ์ประจำตัวของคอนโทรล Button ก็คือ เหตุการณ์ Click()

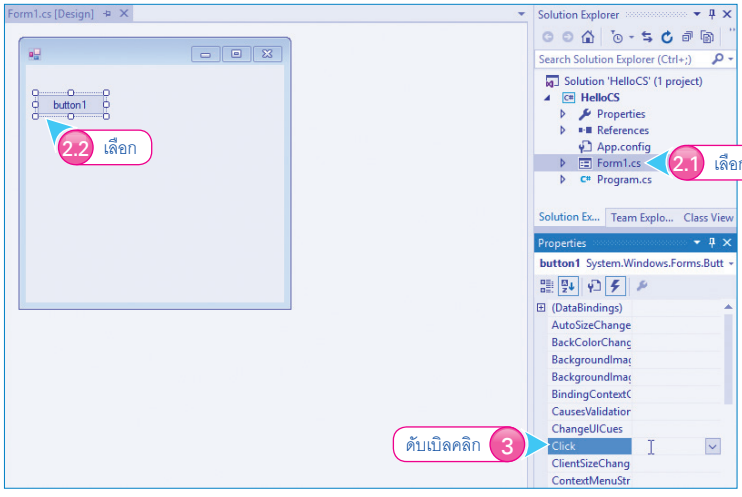
รูปที่ 1-30 แสดงการสร้างเหตุการณ์ Click() ด้วยวิธีดับเบิลคลิก



แต่ละคอนโทรลมีเหตุการณ์ประจำตัวแตกต่างกัน ขึ้นอยู่กับว่าคอนโทรลตัวนั้นๆ ถูกสร้างขึ้นมาเพื่อ “ทำหน้าที่อะไร”

1.2 อีกวิธีหนึ่งก็คือ ให้โฟกัสที่คอนโทรล Button1 ก่อน จากนั้นคลิกปุ่มในหน้าต่าง Properties เพื่อเลือกสร้างเหตุการณ์ที่ต้องการ จะเห็นได้ว่าปุ่มกด Button รองรับหลายเหตุการณ์ ให้ดับเบิลคลิกที่ Click เพื่อสร้างเหตุการณ์ Click() ขึ้นมาสำหรับปุ่มกด Button1 ปุ่มนี้

รูปที่ 1-31 แสดงเหตุการณ์ Click() ที่ได้จากวิธีเลือกสร้างเหตุการณ์



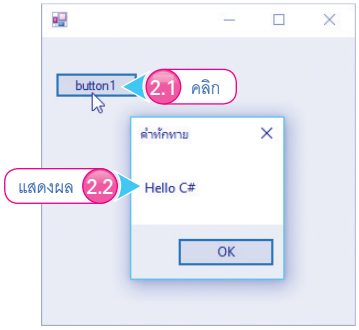
2. ผู้เขียนลองตั้งโจทย์ง่าย ๆ ว่า ต้องการแสดงข้อความ “Hello C#” เมื่อผู้ใช้งานคลิกปุ่ม Button1 สามารถเขียนโค้ดกำหนดการทำงานได้ดังนี้

Form1.cs

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

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

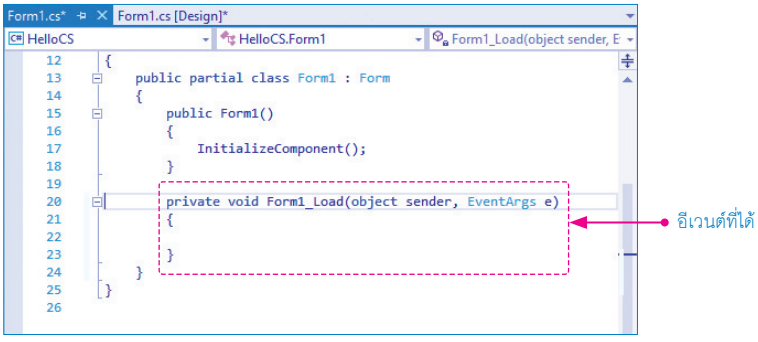
จากโค้ดข้างต้นเป็นการสั่งให้แสดงข้อความ “Hello C#” โดยอาศัยคลาส MessageBox เข้ามาทำหน้าที่สร้างไดอะล็อกแสดงข้อความ ดังรูปที่ 1-32

รูปที่ 1-32
แสดงข้อความที่ได้



3. ในกรณีที่ดับเบิลคลิกบริเวณพื้นที่ว่างๆ ของ Form1 ส่งผลให้ C# จะสร้างเหตุการณ์ Load() ขึ้นมา เหตุการณ์ประจำตัวของ Form ก็คือเหตุการณ์ Load() เพื่อให้ผู้อ่านกำหนดว่า เมื่อ Form1 ถูกโหลดขึ้นมาแล้วให้ทำอะไร

รูปที่ 1-33
แสดงที่จัดเก็บโค้ดภาษา C#



จากรูปที่ 1-33 ที่หน้าต่าง Solution Explorer ทำหน้าที่แสดงโครงสร้างของโปรเจ็กต์ปัจจุบัน ให้ผู้อ่านคลิกที่ Form1 (.cs) เพื่อแสดงโค้ดทั้งหมดที่อยู่ใน Form1

สรุปท้ายบท

เนื้อหาในบทนี้เป็นเพียงขั้นตอนการเตรียมสภาพแวดล้อมให้เครื่องของผู้อ่านพร้อมใช้งานเท่านั้น เนื้อหาในบทต่อไปจะเข้าสู่โลกของการเขียนโปรแกรมอย่างแท้จริง

Professional Visual C# 2015

CHAPTER
02

พื้นฐานการเขียนโปรแกรมด้วยภาษา Visual C#

เนื้อหาในบทนี้จะเข้าสู่โลกของการเขียนโปรแกรมด้วยภาษา VC# 2015 ด้วย Visual Studio 2015 ผู้อ่านจะได้ศึกษาพื้นฐานและไวยากรณ์ทางภาษาที่ต้องทราบเป็นลำดับแรกก่อนที่จะเข้าสู่หัวข้อต่อ ๆ ไป

การประกาศตัวแปรขึ้นมาใช้เก็บข้อมูล

ไม่ว่าผู้อ่านจะเริ่มต้นศึกษาภาษาใดก็ตาม ไวยากรณ์ที่ต้องทำความเข้าใจเป็นลำดับแรก นั่นคือ การสร้างตัวแปร (Variable) ขึ้นมา เพื่อทำหน้าที่เก็บข้อมูลโดยการประกาศตัวแปรในภาษา VC# 2015 มีไวยากรณ์ดังต่อไปนี้

VC# 2015

```
ชนิดข้อมูล ชื่อตัวแปร;
```

กฎการตั้งชื่อตัวแปรในภาษา C#

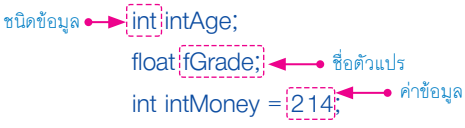
1. ตัวอักษรแรกของตัวแปรต้องขึ้นต้นด้วยตัวอักษรภาษาอังกฤษหรือเครื่องหมาย _ เท่านั้น ห้ามเป็นตัวเลขเด็ดขาด
2. ตัวอักษรตั้งแต่ตัวที่ 2 เป็นต้นไปสามารถเป็นตัวเลขได้
3. ชื่อตัวแปรห้ามมีช่องว่าง นิยมใช้เครื่องหมาย _ แทนช่องว่าง
4. ตัวอักษรภาษาอังกฤษพิมพ์ใหญ่และพิมพ์เล็กถือว่าเป็นคนละตัวกัน
5. ชื่อตัวแปรที่ตั้งขึ้นมาห้ามซ้ำกับคำสงวนของภาษา C#

คำสงวน (Reserved Words) ของภาษา C#

คำสงวน (Reserved Words) เป็นคำที่ภาษา C# กำหนดไว้เพื่อเป็นคำสั่งหรือไวยากรณ์ของภาษา C# สำหรับการเขียนโปรแกรม ส่งผลให้ไม่สามารถใช้คำเหล่านี้ตั้งชื่อตัวแปรได้ เพราะว่าจะซ้ำกับคำสั่งของภาษา C# นั่นเอง คำสงวนที่น่าสนใจมีดังนี้

abstract	double	new	static
as	else	null	string
base	enum	object	struct
bool	false	operator	switch
break	finally	out	this
byte	float	override	throw
case	for	params	true
catch	foreach	private	try
char	goto	protected	typeof
checked	if	public	uint
class	in	readonly	ulong
const	int	ref	ushort
continue	interface	return	using
decimal	internal	sbyte	virtual
default	is	sealed	void
delegate	long	short	while
do	namespace	sizeof	

ตัวอย่างเช่น



ข้อควรจำ

ในการเขียนโปรแกรมภาษา C# คำสั่งแต่ละคำสั่งต้องปิดด้วยเครื่องหมาย ; เสมอ

ขอบเขตของตัวแปร (Variable Scope)

หลังจากที่ผู้อ่านประกาศตัวแปรที่ต้องการใช้งานแล้ว เนื้อหาลำดับถัดมาที่ควรรับทราบก็คือ ตัวแปรดังกล่าวมีขอบเขตที่สามารถเรียกใช้งานได้หรือไม่ ซึ่งมีขอบเขตอยู่ 3 ระดับคือ

1. **ตัวแปรระดับฟอร์ม** ถ้าผู้อ่านประกาศตัวแปรชนิดนี้ไว้ในฟอร์มจะมีขอบเขตกว้างมากที่สุด กล่าวคือ ทุกเหตุการณ์ที่อยู่ในฟอร์มนั้น ๆ สามารถเรียกใช้งานได้

2. **ตัวแปรระดับ Local** หรือเรียกอีกอย่างว่า **ตัวแปรระดับ Procedure** ก็ได้ มีขอบเขตขนาดกลาง เป็นระดับที่เหมาะสมกับการใช้งานมากที่สุด มีขอบเขตอยู่ในแต่ละเหตุการณ์
3. **ตัวแปรระดับ Block** มีขอบเขตขนาดเล็กสุด มักใช้เก็บค่าชั่วคราวหรือตัวแปรใช้วนลูปเสียเป็นส่วนใหญ่ เช่น ตัวแปรที่อยู่ในบล็อกของคำสั่งต่าง ๆ

NOTE

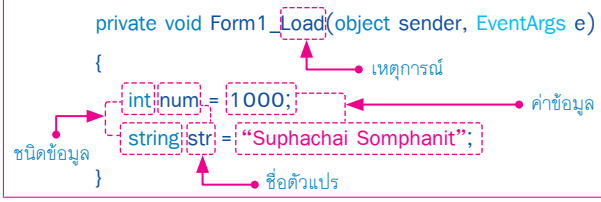
คำว่า เหมาะสม หมายถึง ในการเรียกใช้งานตัวแปรทั้งหมดที่อยู่ในโปรเจ็กต์ของผู้อ่าน สิ่งหนึ่งที่ต้องคำนึงถึงก็คือ หน่วยความจำ (RAM) การเขียนโปรแกรมที่ดีจะต้องมีการใช้งานทรัพยากรระบบให้เหมาะสมกับความสามารถของตัวมันเอง

สำหรับการเลือกใช้ตัวแปรระดับใดก็ตาม ไม่มีข้อกำหนดตายตัว หรือข้อบังคับใด ๆ ทั้งสิ้น ขึ้นอยู่กับลักษณะของโปรเจ็กต์ของผู้อ่านเสียมากกว่า ขอให้เลือกใช้ตามความเหมาะสม

ขอบเขตของตัวแปรจะเป็นตัวบ่งบอกว่า ตัวแปรนั้น ๆ จะมีอายุในการทำงานนานเท่าไร เพราะเมื่อสิ้นสุดการทำงานของตัวแปรนั้น ๆ แอพพลิเคชันจะคืนทรัพยากรที่จองมาใช้งานกลับสู่ระบบ ถ้าผู้อ่านเลือกใช้ขอบเขตและประเภทของตัวแปรได้อย่างเหมาะสมแล้ว โปรเจ็กต์ที่ได้ก็จะมีทั้งประสิทธิภาพในด้านการทำงานและการใช้ทรัพยากรระบบ ตัวอย่างเช่น

ผู้เขียนประกาศตัวแปรขึ้นมา 2 ตัว คือ ตัวแปร num กำหนดให้มีชนิดข้อมูลเป็นตัวเลขจำนวนเต็ม int และกำหนดค่าเริ่มต้นให้เก็บค่า 1000 ส่วนตัวแปร str กำหนดให้มีชนิดข้อมูลเป็นข้อความ string และกำหนดค่าเริ่มต้นเป็นชื่อ-สกุลผู้เขียน

VC# 2015



ตัวแปรทั้ง 2 ตัว ถูกประกาศให้อยู่ภายในเหตุการณ์ Form1_Load() เราเรียกตัวแปรประเภทนี้ว่า **ตัวแปรที่มีขอบเขตแบบ Local** หมายถึง สามารถใช้งานได้เฉพาะ “ภายในเหตุการณ์ Form1_Load()” เท่านั้น

ต่อมาผู้เขียนลองประกาศตัวแปรอีกลักษณะหนึ่ง ตำแหน่งที่ประกาศตัวแปรอยู่ภายนอกเหตุการณ์ต่าง ๆ กล่าวคือ สร้างตัวแปรที่ชื่อว่า str ขึ้นมา กำหนดให้มีชนิดข้อมูลเป็นข้อความ string

VC# 2015

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    string str = "";
    private void Form1_Load(object sender, EventArgs e)
    {
        str = "Visual C#";
```

```
    }                                   เหตุการณ์
    private void button1_Click(object sender, EventArgs e)
    {
        str = "Visual C#"
    }
}
```

ในกรณีนี้ตัวแปร str ถูกประกาศอยู่ภายนอกเหตุการณ์ต่าง ๆ เรียกว่า **ตัวแปรที่มีขอบเขตระดับฟอร์ม** ผู้อ่านสามารถเรียกใช้ตัวแปร str นี้ในเหตุการณ์ต่าง ๆ ได้ ในกรณีนี้คือเหตุการณ์ Form1_Load() กับเหตุการณ์ button1_Click()

ตัวแปรระดับ Local และระดับฟอร์ม

เพื่อเพิ่มความเข้าใจในการใช้งานตัวแปรระดับ Local และระดับฟอร์ม ให้ผู้อ่านศึกษาการใช้งานตัวแปรจากตัวอย่างต่อไปนี้

1. ออกแบบฟอร์มดังรูป 2-1
2. เขียนโค้ดในเหตุการณ์ Form1_Load() ดังนี้

ตัวอย่างที่ 2-1 ตัวแปรระดับ Local และระดับฟอร์ม (Form1.cs)

```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    string str = “ตัวแปรระดับฟอร์ม”;

    private void Form1_Load(object sender, EventArgs e)
    {
        //string str = "ตัวแปรระดับ Local";
        label1.Text = str;
    }
}
```

3. เขียนโค้ดในเหตุการณ์ button1_Click() ดังนี้

ตัวอย่างที่ 2-1 ตัวแปรระดับ Local และระดับฟอร์ม (Form1.cs)

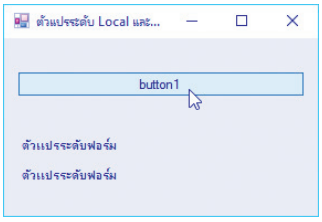
```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }
    string str = “ตัวแปรระดับฟอร์ม”;

    private void Form1_Load(object sender, EventArgs e)
    {
        //string str = "ตัวแปรระดับ Local";
        label1.Text = str;
    }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            label2.Text = str;
        }
    }
```

เหตุการณ์ (Click)

4. เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังนี้

รูปที่ 2-1 ผลการรันตัวอย่างที่ 2-1



จากรูปที่ 2-1 ผู้เขียนประกาศตัวแปรในระดับฟอร์มที่ชื่อว่า str เห็นได้ว่าคอนโทรล label1 ถูกกำหนดค่าในเหตุการณ์ Form1_Load() ส่วนคอนโทรล label2 ถูกกำหนดค่าในเหตุการณ์ button1_Click()

เห็นได้ว่าการประกาศตัวแปรในลักษณะนี้ ผู้อ่านสามารถใช้งานตัวแปร str นี้ได้หลายเหตุการณ์ คือ เหตุการณ์ Form1_Load() และเหตุการณ์ button1_Click()

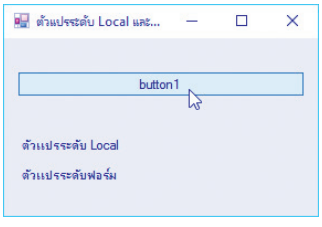
5. ให้ผู้อ่านนำหมายเหตุโค้ดบรรทัดต่อไปนี้ออก แล้วลองรันตัวอย่างอีกครั้ง ดังรูปที่ 2-2

VC# 2015

```
        private void Form1_Load(object sender, EventArgs e)
        {
            string str = "ตัวแปรระดับ Local";
            label1.Text = str;
        }
```

6. เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังนี้

รูปที่ 2-2 หลังจากเอาหมายเหตุออก



จากรูปที่ 2-2 ผู้เขียนสร้างตัวแปรอีก 1 ตัว มีขอบเขตระดับ Local ถูกประกาศภายในเหตุการณ์ Form1_Load() มีขอบเขตเรียกใช้งานเฉพาะเหตุการณ์ Form1_Load() เท่านั้น ผู้เขียนจงใจให้มีชื่อเดียวกับตัวแปรระดับฟอร์มข้างต้น พบว่าคอนโทรล label1 แสดงข้อความของตัวแปรในระดับ Local ก่อน

สรุปได้ว่าถ้ามีตัวแปรชื่อเดียวกันทั้งในระดับฟอร์มกับระดับ Local ตัวแปรที่อยู่ในระดับ Local มีความสำคัญมากกว่าตัวแปรที่อยู่ในระดับฟอร์ม

ตัวแปรระดับ Block

เพื่อเพิ่มความเข้าใจในการใช้งานตัวแปรระดับ Block ซึ่งเป็นตัวแปรที่มีขอบเขตแคบที่สุด ให้ผู้อ่านศึกษาการใช้งานตัวแปรจากตัวอย่างต่อไปนี้

1. ออกแบบฟอร์มดังรูป 2-3
2. เขียนโค้ดในเหตุการณ์ Form1_Load() ดังนี้

ตัวอย่างที่ 2-2 ตัวแปรระดับ Block (Form1.cs)

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

    private void Form1_Load(object sender, EventArgs e)
    {
        int i = 0;
        for (i = 0; i <= 2; i++)
        {
            string strVal = "ตัวแปรระดับ Block";
            label1.Text = strVal;
        }
        //label1.Text = strVal;
    }
}
```

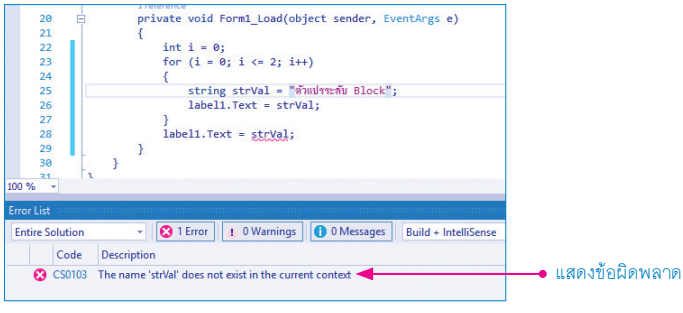
3. เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังนี้

รูปที่ 2-3 ผลการรันตัวอย่างที่ 2-2

จากรูปที่ 2-3 ผู้เขียนประกาศตัวแปรที่ชื่อว่า strVal ในบล็อกคำสั่งของลูป for ส่งผลให้ตัวแปรตัวนี้ มีขอบเขตการใช้งานในบล็อกของลูป for นี้เท่านั้น

4. ให้เอาหมายเหตุออก พบว่าไม่สามารถอ่านค่าของตัวแปร strVal ตัวนี้นอกลูป for ได้เลย ดังรูปที่ 2-4

รูปที่ 2-4 แสดงการใช้งานตัวแปร strVal นอกบล็อกคำสั่ง for



จากรูปที่ 2-4 ข้อผิดพลาดที่ปรากฏขึ้นมา เกิดจากตัวแปร strVal ยังไม่มีการประกาศใช้งาน เพราะว่าเราเรียกใช้งานตัวแปรตัวนี้เกินขอบเขตของมันนั่นเอง

ชนิดข้อมูล (Type)

ในภาษา VC# 2015 มีการแบ่งข้อมูลออกเป็น 2 กลุ่มใหญ่ คือ

1. **Value types** ประกอบด้วยข้อมูลชนิดตัวเลขจำนวนเต็มกับตัวเลขทศนิยม (Numeric), ค่าทางตรรกะ (Boolean), ตัวเลขจำนวนจริง หรือตัวเลขทศนิยม (Real Number) และตัวอักษร (Character) อาจจะเรียกข้อมูลกลุ่มนี้ว่า **ข้อมูลพื้นฐาน (Primitive Data Types)** ก็ได้
2. **Reference types** ประกอบด้วยข้อมูล 2 ประเภท คือ ข้อความ (String) และออบเจ็กต์ (Object) เนื้อหาของหัวข้อนี้จะกล่าวถึงเฉพาะชนิดข้อมูลในกลุ่ม Value types เท่านั้น ซึ่งถือเป็นชนิดข้อมูลพื้นฐานที่ต้องทำความเข้าใจเป็นลำดับแรก

ขอบเขตการจัดเก็บข้อมูลแต่ละชนิดข้อมูล และขนาดหน่วยความจำที่ใช้

ชนิดข้อมูล (Type) แต่ละประเภทที่ผู้อ่านกำหนดให้ตัวแปรแต่ละตัว มีความสามารถในการจัดเก็บตัวเลขไม่เท่ากัน มากน้อยไปตามแต่ละชนิด เกิดมาจากการใช้พื้นที่ในหน่วยความจำ (RAM) ไม่เท่ากัน

ในหัวข้อนี้ผู้เขียนต้องการตรวจสอบว่า ชนิดข้อมูลแต่ละประเภทมีขอบเขตและใช้ขนาดหน่วยความจำเท่าใด โดยตรวจสอบด้วยโค้ดโปรแกรมในตัวอย่างที่ 2-3

ตัวอย่างที่ 2-3 ขอบเขตการจัดเก็บข้อมูลแต่ละชนิดข้อมูลและขนาดหน่วยความจำที่ใช้ มีดังโค้ดต่อไปนี้

ตัวอย่างที่ 2-3 ขอบเขตการจัดเก็บข้อมูลแต่ละชนิดข้อมูล และขนาดหน่วยความจำที่ใช้ (Form.cs)

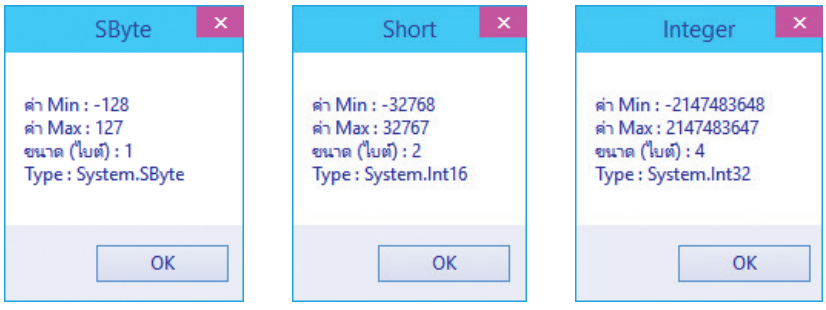
```
public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    private void Form1_Load(object sender, EventArgs e)
    {
        string result = "";

        sbyte sb = 0;
        result = "ค่า Min : " + sbyte.MinValue.ToString() + Environment.NewLine;
        result += "ค่า Max : " + sbyte.MaxValue.ToString() + Environment.NewLine;
        result += "ขนาด (ไบต์) : " + Marshal.SizeOf(sbyte.MaxValue.GetType()) + Environment.
NewLine;
        result += "Type : " + sb.GetType().ToString();
        MessageBox.Show(result.ToString(), "SByte");

        short sh = 0;
        result = "ค่า Min : " + short.MinValue.ToString() + Environment.NewLine;
        result += "ค่า Max : " + short.MaxValue.ToString() + Environment.NewLine;
        result += "ขนาด (ไบต์) : " + Marshal.SizeOf(short.MaxValue.GetType()) + Environment.
NewLine;
```

```
            result += "Type : " + sh.GetType().ToString();
            MessageBox.Show(result.ToString(), "Short");

            int i = 0;
            result = "ค่า Min : " + int.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + int.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(int.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + i.GetType().ToString();
            MessageBox.Show(result.ToString(), "Integer");

            long l = 0;
            result = "ค่า Min : " + long.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + long.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(long.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + l.GetType().ToString();
            MessageBox.Show(result.ToString(), "Long");

            byte b = 0;
            result = "ค่า Min : " + byte.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + byte.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(byte.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + b.GetType().ToString();
            MessageBox.Show(result.ToString(), "Byte");
            ushort us = 0;
            result = "ค่า Min : " + ushort.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + ushort.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(ushort.MaxValue.GetType()) + Environment.
NewLine;            result += "Type : " + us.GetType().ToString();
            MessageBox.Show(result.ToString(), "UShort");

            uint ui = 0;
            result = "ค่า Min : " + uint.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + uint.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(uint.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + ui.GetType().ToString();
            MessageBox.Show(result.ToString(), "UInteger");

            ulong ul = 0;
            result = "ค่า Min : " + ulong.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + ulong.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(ulong.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + ul.GetType().ToString();
            MessageBox.Show(result.ToString(), "ULong");

            float sng = 0;
            result = "ค่า Min : " + float.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + float.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(float.MaxValue.GetType()) + Environment.
NewLine;
```

```
            result += "Type : " + sng.GetType().ToString();
            MessageBox.Show(result.ToString(), "Single");

            double dbl = 0;
            result = "ค่า Min : " + double.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + double.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(double.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + dbl.GetType().ToString();
            MessageBox.Show(result.ToString(), "Double");

            decimal dc = default(decimal);
            result = "ค่า Min : " + decimal.MinValue.ToString() + Environment.NewLine;
            result += "ค่า Max : " + decimal.MaxValue.ToString() + Environment.NewLine;
            result += "ขนาด (ไบต์) : " + Marshal.SizeOf(decimal.MaxValue.GetType()) + Environment.
NewLine;
            result += "Type : " + dc.GetType().ToString();
            MessageBox.Show(result.ToString(), "Decimal");

            char ch = '\0';
            result = "Type : " + ch.GetType().ToString();
            MessageBox.Show(result.ToString(), "Char");

            bool bl = false;
            result = "Type : " + bl.GetType().ToString();
            MessageBox.Show(result.ToString(), "Boolean");
        }
    }
```

เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังนี้

รูปที่ 2-5 ผลการรันตัวอย่างที่ 2-3



จากรูปที่ 2-5 เป็นผลการตรวจสอบชนิดข้อมูลบางส่วน ได้แก่ sbyte, short และ int ผู้เขียนสรุปขอบเขตและขนาดหน่วยความจำที่ใช้ ดังตารางต่อไปนี้

ชนิดข้อมูล	C# Type	.NET Framework Type (CLR Type)	ขนาดไบต์	ขอบเขตของข้อมูล
จำนวนเต็ม	sbyte	System.Sbyte	1	-128 ถึง 127
	short	System.Int16	2	-32768 ถึง 32767
	int	System.Int32	4	-2147483648 ถึง 2147483647