

คู่มือ



Coding และพัฒนาแอปพลิเคชันด้วย

C#

ฉบับสมบูรณ์



Desktop



Web



Cloud



Mobile



Games



IoT



AI

- ☒ เรียนรู้พื้นฐานและเทคนิคการพัฒนาแอปพลิเคชันด้วยภาษา C#
- ☒ อธิบายตั้งแต่พื้นฐานจนสามารถนำไปพัฒนาแอปพลิเคชันใช้งานได้จริง
- ☒ มีโค้ดตัวอย่างและคำอธิบายอย่างละเอียดอ่านเข้าใจง่าย
- ☒ อธิบายเป็นขั้นตอน เหมาะสำหรับนักเรียนนักศึกษา และผู้เริ่มต้น



ดาวน์โหลดอย่างปลอดภัย
[https://serazu.com/
9786164874848](https://serazu.com/9786164874848)

ศุภชัย สมพานิช

สารบัญ

บทที่ 1	ก้าวแรกก่อนเข้าสู่โลกของ .NET	1
	ทำความรู้จักกับแพลตฟอร์ม .NET (.NET Platform).....	1
	ขอบเขตการนำเสนอของหนังสือเล่มนี้.....	2
	ทำความรู้จักกับประเภทโปรเจกต์ที่น่าสนใจในขั้นต้น.....	2
	การดาวน์โหลดและติดตั้งโปรแกรม Visual Studio 2022.....	3
	การติดตั้ง .NET SDK เพิ่มเติม	7
	สถานะของ .NET SDK.....	8
	การติดตั้งฟอนต์ Cascadia Code สำหรับเขียนโค้ดโดยเฉพาะ.....	9
	การกำหนดให้แสดงหมายเลขบรรทัด.....	11
	การสร้างโปรเจกต์ Windows Forms App.....	12
	การออกแบบส่วนแสดงผลด้วยคอนโทรล (User Interface - UI)	15
	การปรับแต่งคอนโทรลขั้นต้นด้วยวิธีแก้ไขคุณสมบัติ (Property)	17
	การทดสอบโปรเจกต์.....	19
	การยกเลิกฟิเจอร์ Nullable	20
	วิธีการจัดเก็บโปรเจกต์ .NET	21
บทที่ 2	พื้นฐานการทำงานกับโปรเจกต์ประเภท Windows Forms App (.NET)	25
	ทำความรู้จักกับฟอร์ม (Form)	25
	คุณสมบัติของ Form ที่สำคัญ.....	26
	การเพิ่มฟอร์มใหม่เข้ามาในโปรเจกต์.....	27
	วิธีการสั่งให้เปิดฟอร์มใหม่โดยการเขียนโค้ด.....	28
	หลักการเขียนโปรแกรมแบบ Event Driven Programming.....	30
	พื้นฐานการใช้งานฟอร์มแบบ MDI และสร้างระบบเมนู (Menu)	34
	การสร้างเมนูให้กับฟอร์มหลัก	37
	การยกเลิกเหตุการณ์.....	39
บทที่ 3	พื้นฐานการเขียนโปรแกรมด้วยภาษา C#.....	41
	การประกาศตัวแปรขึ้นมาใช้เก็บข้อมูล (Variable).....	41
	การใช้งานข้อมูลชนิดตัวเลข	42
	กฎการตั้งชื่อตัวแปรในภาษา C#	43
	คำสงวน (Reserved Words) ของภาษา C#.....	44
	ขอบเขตการจัดเก็บข้อมูลแต่ละชนิดข้อมูลและขนาดหน่วยความจำที่ใช้.....	45
	การกำหนดชนิดข้อมูลโดยอัตโนมัติ (Type Inference)	49

ตัวดำเนินการด้านคณิตศาสตร์	51
ตัวดำเนินการด้านอ็อปเตดค่า	52
ตัวดำเนินการสำหรับเพิ่มหรือลดค่า	53
การเขียนหมายเหตุ (Comment) ในโค้ด	53
ขอบเขตของตัวแปร (Variable Scope)	54
การใช้งานตัวแปรระดับ Local และระดับฟอร์มในเวลาเดียวกัน	56
ตัวแปรระดับ Block	58
การสร้างตัวแปรเก็บหลายค่าในเวลาเดียวกันด้วย ValueTuple	60
ค่าคงที่ (Constant)	62

บทที่ 4	ทำงานกับข้อมูลชนิดข้อความ string และวันเวลา DateTime	63
	พื้นฐานการทำงานกับข้อมูลชนิดข้อความ string	63
	วิธีการตรวจสอบจำนวนตัวอักษรใน string	64
	การต่อข้อความเข้าด้วยกัน	65
	การแปลงตัวเลขที่อยู่ในฐานะข้อความ string ให้กลายเป็นข้อมูลชนิดตัวเลข	67
	การแทนที่ข้อความด้วยเมธอด Replace()	68
	การแบ่งข้อความด้วยฟังก์ชัน Split()	68
	การใช้งานข้อมูลชนิดตัวอักษร char	69
	การแปลงชนิดข้อความ string เป็นชนิดข้อมูลอาร์เรย์ของ char	70
	การจัดรูปแบบของตัวเลขทศนิยม	71
	การแทรกค่าของตัวแปรเข้าไปในข้อความ string ด้วยเครื่องหมาย \$	73
	การทำงานกับข้อความหลายบรรทัด (Raw string literals)	74
	การแทรกค่าของตัวแปรเข้าไปใน "" ""	75
	การทำงานกับข้อมูลชนิดวันที่และเวลา (DateTime)	76
	การสร้างข้อมูลเฉพาะวันที่ด้วยคลาส DateOnly	79
	การสร้างข้อมูลเฉพาะเวลาด้วยคลาส TimeOnly	80

บทที่ 5	การตรวจสอบและควบคุม	81
	พื้นฐานการตรวจสอบเงื่อนไขด้วยคำสั่ง if...else	81
	การใช้ตัวดำเนินการด้านเปรียบเทียบ	83
	การใช้ตัวดำเนินการ "และ" (&&) และตัวดำเนินการ "หรือ" ()	84
	การตรวจสอบเงื่อนไขร่วมกับตัวดำเนินการด้านตรรกะ	85
	การตรวจสอบเงื่อนไขด้วยตัวดำเนินการ ?	88
	การตรวจสอบเงื่อนไขด้วยคำสั่ง switch case	89
	การวนลูปด้วยคำสั่ง for	91
	การวนลูปด้วยคำสั่ง while	93
	การวนลูปแบบ do...while	94

บทที่ 6	การใช้งานคอนโทรลเบื้องต้น	97
	ทำความเข้าใจกับแถบคอนโทรล	97
	พื้นฐานการสร้างโปรแกรมรับข้อความจากผู้ใช้งานด้วยคอนโทรล TextBox.....	98
	คุณสมบัติของคอนโทรล TextBox ที่สำคัญ.....	102
	การตรวจสอบข้อมูลว่างในคอนโทรล TextBox.....	103
	การแสดงรูปภาพด้วยคอนโทรล PictureBox.....	106
	การสร้างส่วนแสดงผลแบบ list ด้วย ComboBox และ ListBox.....	108
	การเพิ่ม, ถอด และล้างรายการใน ListBox.....	113
	การสร้างส่วนแสดงผลแบบรายการด้วยคอนโทรล ListView.....	118
	การสร้างตัวเลือกแบบเลือกได้ 1 อย่างด้วยคอนโทรล RadioButton.....	123
	การสร้างตัวเลือกแบบหลายตัวเลือกด้วยคอนโทรล CheckBox.....	126
	ทำงานกับไฟล์ข้อความด้วยคอนโทรล RichTextBox, OpenFileDialog และ SaveFileDialog	128
	คุณสมบัติของคอนโทรล RichTextBox ที่สำคัญ	128
	คุณสมบัติของคอนโทรล OpenFileDialog ที่สำคัญ.....	129
	คุณสมบัติของคอนโทรล SaveFileDialog ที่สำคัญ.....	129
บทที่ 7	โครงสร้างข้อมูลแบบ Array และ Range	135
	พื้นฐานการใช้งานตัวแปรชุดแบบอาร์เรย์ (Array).....	135
	การอ่านค่าตัวแปรอาร์เรย์ด้วยรูปแบบ foreach...in.....	137
	การสร้างตัวแปรอาร์เรย์แบบ Type Inference.....	139
	การอ่านค่าจากตัวแปรอาร์เรย์จากลำดับย้อนหลัง	140
	การกลับด้านข้อมูลในอาร์เรย์	141
	การสร้างอาร์เรย์ด้วยคลาส ArrayList.....	143
	การถอดค่าซ้ำกันในอาร์เรย์	145
	การหาค่ามากที่สุดและน้อยสุดในอาร์เรย์.....	147
	การสร้างอาร์เรย์แบบหลายมิติ	149
	การเข้าถึงข้อมูลแบบระบุช่วงด้วย Range.....	150
บทที่ 8	พื้นฐานการดักจับและตรวจสอบข้อผิดพลาด (Debug & Exception)	155
	พื้นฐานการดักจับข้อผิดพลาดด้วย try...catch.....	155
	การดักจับข้อผิดพลาดกับโครงสร้างข้อมูล Array.....	157
	การดักจับข้อผิดพลาดด้วยคำสั่ง try...catch...when	158
บทที่ 9	พื้นฐานการเขียนโปรแกรมเชิงวัตถุ (OOP Programming)	163
	การเขียนโปรแกรมเชิงวัตถุ (OOP) คืออะไร	163
	พื้นฐานการสร้างคลาสขึ้นมาใช้งานเอง	164
	การสร้างคุณสมบัติด้วยวิธีเขียนโค้ดแบบย่อ (Auto-Implemented Properties)	171



การกำหนดค่าเริ่มต้นให้กับคุณสมบัติ.....	171
การสร้างคุณสมบัติที่อ่านค่าได้เพียงอย่างเดียว	172
การเขียนโค้ดลดรูปแบบ Expression Bodied ด้วยตัวดำเนินการ =>	173
การสร้างคุณสมบัติเก็บหลายค่าด้วย ValueTuple	174
การจัดกลุ่มคลาสด้วยระบบเนมสเปซ (Namespaces)	175
วิธีการเรียกใช้เนมสเปซ	176
แบบที่ 1 อาศัยคำสั่ง using	176
แบบที่ 2 อาศัยการระบุชื่อเต็ม	178
แบบที่ 3 อาศัยการตั้งชื่อเล่น Alias.....	178
การใช้งานเนมสเปซแบบ File Scoped Namespaces.....	179
การระบุเนมสเปซอัตโนมัติด้วยพีเจียร์ Implicit Usings.....	181
การอ้างอิงเนมสเปซครั้งเดียวด้วยคำสั่ง global	182
ค่า null คืออะไร	185
การตรวจสอบค่า null กับตัวแปรประเภทออบเจกต์ด้วยเครื่องหมาย ? และ ??.....	187
ทำความเข้าใจกับชนิดข้อมูลแบบ Anonymous Type	193
การแก้ไขค่าของตัวแปร Anonymous Type ด้วยคำสั่ง with.....	195
ทำความเข้าใจกับ Constructor.....	196
การบังคับให้กำหนดค่าให้กับคุณสมบัติด้วยคำสั่ง required.....	199
การใช้งาน record (Immutable Object).....	200
การแก้ไขข้อมูลใน record.....	203
พื้นฐานการใช้โครงสร้างข้อมูลแบบ struct	204
การสร้างคุณสมบัติและเมธอดใน struct	205
ข้อแตกต่างระหว่าง Struct กับ Class.....	207
การสร้างคลาสแบบ static (Static Class)	211
การสร้างออบเจกต์แบบปกติ, var และฟังก์ชัน new().....	212

บทที่ 10 Inheritance & Polymorphism 215

กฎพื้นฐาน 3 ข้อของ OOP.....	215
พื้นฐานการสืบทอดคลาส (Inheritance).....	216
การสร้างคลาสแบบ Abstract Class	219
สถานะที่หลากหลาย (Polymorphism)	223

บทที่ 11 การตรวจสอบเงื่อนไขด้วย Pattern Matching..... 227

พื้นฐานการตรวจสอบเงื่อนไขด้วย Pattern Matching.....	227
การตรวจสอบคุณสมบัติของคลาสด้วย Property Pattern.....	230
การตรวจสอบเงื่อนไขแบบช่วงด้วย Pattern Matching.....	234
การตรวจสอบเงื่อนไขแบบหลายรายการ (List Pattern)	235
การอ่านข้อมูลจากเงื่อนไข .. ของ List Pattern	239

บทที่ 12	พื้นฐานการใช้งานระบบฐานข้อมูล SQL Server 2022 Express Edition	241
	การดาวน์โหลดและติดตั้งฐานข้อมูล SQL Server 2022 Express Edition	241
	การทำงานกับระบบฐานข้อมูล SQL Server	248
	การดาวน์โหลดและติดตั้ง SQL Server Management Studio (SSMS) และ Azure Data Studio (ADS)	248
บทที่ 13	พื้นฐานการใช้งานฐานข้อมูล SQL Server 2022	251
	พื้นฐานการใช้งานฐานข้อมูล SQL Server 2022 Express Edition ด้วย SSMS.....	251
	การยกเลิกป้องกันการแก้ไขโครงสร้างตาราง	258
	การใช้ฟิลด์ Primary Key แบบฟิลด์ใส่ค่าโดยอัตโนมัติ.....	260
	การใช้ฟิลด์ Primary Key แบบมีการเพิ่มค่า.....	260
	การใช้ฟิลด์ Primary Key แบบใช้ค่า GUID	261
	การสร้างความสัมพันธ์ระหว่างตาราง.....	263
	การ Backup และ Restore ฐานข้อมูล	266
	การจัดการฐานข้อมูล SQL Server ด้วย Azure Data Studio (ADS).....	270
	การทำงานกับฐานข้อมูล SQL Server ในขั้นต้นด้วย ADS.....	272
บทที่ 14	การทำงานกับระบบฐานข้อมูล SQL Server ด้วย Entity Framework Core (EF Core)	275
	การอ้างอิง Library ด้วยระบบ Dependencies.....	275
	การติดตั้ง EF Core ให้กับโปรเจกต์ Windows Forms App.....	276
	ทำความเข้าใจกับฐานข้อมูล thaivBizShop.....	279
	การแปลงตารางในฐานข้อมูล SQL Server เป็นคลาสของภาษา C#	281
	พื้นฐานการเรียกดูข้อมูลจากฐานข้อมูล SQL Server แบบไม่มีเงื่อนไขด้วย EF Core	284
	การเลือกบางฟิลด์ที่ต้องการจากตาราง	286
	Anonymous Type และการใช้งาน ViewModel.....	288
	การใช้งาน ViewModel แบบ record สำหรับงานอ่านข้อมูล.....	292
	การทำงานกับข้อมูลที่มีความสัมพันธ์กันด้วย EF Core.....	292
	การแสดงผลข้อมูลที่มีความสัมพันธ์กันด้วยคอนโทรล ComboBox	296
	การอัปเดต DbContext ของ EF Core.....	302
บทที่ 15	การจัดการข้อมูล (CRUD) ในฐานข้อมูล SQL Server ด้วย EF Core.....	305
	การจัดการข้อมูลในฐานข้อมูล SQL Server ด้วย EF Core.....	305
บทที่ 16	ADO.NET ในยุค .NET	319
	ADO.NET ในยุค .NET (เนมสเปซ Microsoft.Data).....	319

บทที่ 17	พื้นฐานการพัฒนา Web Application ด้วย ASP.NET Core MVC	337
	โครงสร้างพื้นฐานของหน้าเว็บเพจด้วยภาษา HTML5	338
	วิธีการเข้าถึงหน้าเว็บเพจให้สามารถแสดงผลภาษาไทย	339
	พื้นฐานการสร้างโปรเจกต์แบบ ASP.NET Core MVC	339
	การทดสอบรันโปรเจกต์ ASP.NET Core MVC.....	342
	ทำความเข้าใจกับโครงสร้างโปรเจกต์ของ ASP.NET Core MVC	345
	รู้จักกับส่วนแสดงผลของโปรเจกต์ ASP.NET Core MVC	346
	การใช้งานไฟล์ HTML (Static Files) ในโฟลเดอร์ wwwroot	350
บทที่ 18	พื้นฐานการสร้าง Web API ด้วย ASP.NET Core Web API	355
	ASP.NET Core Web API คืออะไร	355
	การสร้างโปรเจกต์ ASP.NET Core Web API.....	356
	การทดสอบและหยุดรันโปรเจกต์ ASP.NET Core Web API.....	359
	ทำความเข้าใจกับโครงสร้างโปรเจกต์ ASP.NET Core Web API	362
	การสร้าง Web API แรกด้วย ASP.NET Core Web API.....	366
บทที่ 19	การสร้างการทำงาน CRUD ให้กับ Web API แบบเบื้องต้น (API Controller)	377
	การสร้าง CRUD แบบ API Controller	377
	การจำลองฐานข้อมูลในหน่วยความจำ (RAM).....	385
	การทดสอบการทำงานของ Web API ด้วย Swagger.....	389
บทที่ 20	การสร้าง Web API แบบ Minimal APIs.....	405
	Minimal APIs คืออะไร?	405
	การสร้างโปรเจกต์ ASP.NET Core Web API แบบ Minimal APIs	406
	การสร้าง CRUD แบบ Minimal APIs เบื้องต้น.....	412
บทที่ 21	พื้นฐานการพัฒนา Web Apps แบบคอนเทนเนอร์ด้วย Docker.....	423
	การดาวน์โหลดและติดตั้ง Windows Subsystem for Linux (WSL).....	424
	การดาวน์โหลดและติดตั้ง Docker	426
	การทำงานกับ Docker Image/Container ในเบื้องต้น	429
	การลบอิมเมจ	436
บทที่ 22	การสร้าง ASP.NET Core Web API ด้วย Docker	439
	การสร้างโปรเจกต์ ASP.NET Core Web API ที่มีการใช้งาน Docker	439
บทที่ 23	การเผยแพร่อิมเมจแบบสาธารณะกับ Docker Registry	447
	การ push อิมเมจสู่ Docker Registry.....	447

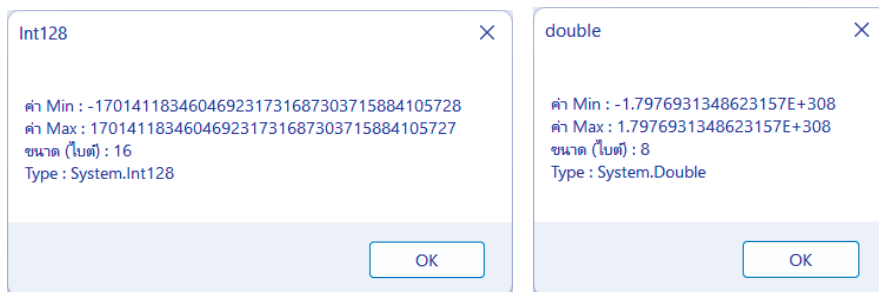


จากรูปที่ 2-1 เมื่อผู้อ่านโฟกัสที่ Form1 หน้าต่างคุณสมบัติก็จะแสดงรายการคุณสมบัติของ Form1 เช่นกัน

คุณสมบัติของ Form ที่สำคัญ

คุณสมบัติของ Form ที่สำคัญ แสดงดังตารางต่อไปนี้

คุณสมบัติ	หน้าที่
Name	ทำหน้าที่ตั้งชื่อให้กับฟอร์ม ใช้สำหรับอ้างอิงตอนเขียนโค้ด
Text	กำหนดข้อความที่จะแสดงในแถบ Title Bar
BackColor	กำหนดสีพื้นหลังให้กับฟอร์ม
ForeColor	กำหนดสีตัวอักษรที่อยู่ในฟอร์ม
FormBorderStyle	กำหนดรูปแบบเส้นขอบของฟอร์ม โดยที่ <ul style="list-style-type: none">• None หมายถึง ไม่มีแถบใดเดิลบาร์• FixedSingle หมายถึง ปรับขนาดฟอร์มไม่ได้ แต่มีปุ่ม Minimize, Maximize และปุ่ม Close• Fixed3D หมายถึง กำหนดให้มีเส้นขอบแบบ FixedSingle แต่มีแนวลิคด้วย• 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 <ul style="list-style-type: none">• True หมายถึง ให้มีปุ่ม Maximize มุมขวา-บน• False หมายถึง ซ่อนปุ่ม Maximize
MinimizeBox	กำหนดให้แบบฟอร์มแสดงปุ่มย่อขนาดฟอร์ม โดยที่ <ul style="list-style-type: none">• True หมายถึง ให้มีปุ่ม Minimize มุมขวา-บน• False หมายถึง ซ่อนปุ่ม Minimize
IsMdiContainer	กำหนดให้ฟอร์มทำหน้าที่บรรจุฟอร์มอื่นๆ แบบหลายหน้าต่างได้หรือไม่ โดยที่ <ul style="list-style-type: none">• True หมายถึง กำหนดให้ฟอร์มเป็นตัวบรรจุแบบ MDI• False หมายถึง กำหนดให้เป็นแบบฟอร์มปกติ



รูปที่ 3-1 ชนิดข้อมูลพื้นฐานบางส่วน เช่น short, int, Int128 และ double

จากรูปที่ 3-1 เป็นผลการตรวจสอบชนิดข้อมูลบางส่วน ได้แก่ short, int, Int128 และ double ผู้เขียนสรุปขอบเขตและขนาดหน่วยความจำที่ใช้ ดังตารางต่อไปนี้

VB Type	C# Type	.NET Type (CLR Type)	ขนาดบิต	ขอบเขตของข้อมูล
SByte	sbyte	System.Sbyte	1	-128 ถึง 127
Short	short	System.Int16	2	-32768 ถึง 32767
Integer	int	System.Int32	4	-2147483648 ถึง 2147483647
Long	long	System.Int64	8	-9223372036854775808 ถึง 9223372036854775807
Int128	Int128	System.Int128	16	-170141183460469231731687303715884105728 ถึง 170141183460469231731687303715884105727
Byte	byte	System.Byte	1	0 ถึง 255
UShort	ushort	System.UInt16	2	0 ถึง 65535
UInteger	uint	System.UInt32	4	0 ถึง 4294967295
ULong	ulong	System.UInt64	8	0 ถึง 18446744073709551615
Single	float	System.Single	4	-3.402823E+38 ถึง 3.402823E+38
Double	double	System.Double	8	-1.79769313486232E+308 ถึง 1.79769313486232E+308
Decimal	decimal	System.Decimal	16	-79228162514264337593543950335 ถึง 79228162514264337593543950335

การตรวจสอบเงื่อนไขด้วยคำสั่ง switch case

ในกรณีเงื่อนไขที่กำลังตรวจสอบมีจำนวนมาก หลายกรณี ผู้อ่านไม่ควรตรวจสอบด้วยคำสั่ง if มีข้อเสนอแนะว่า ขอให้ตรวจสอบด้วยคำสั่ง **switch case** แทน มีรูปแบบไวยากรณ์ดังนี้

C# 11

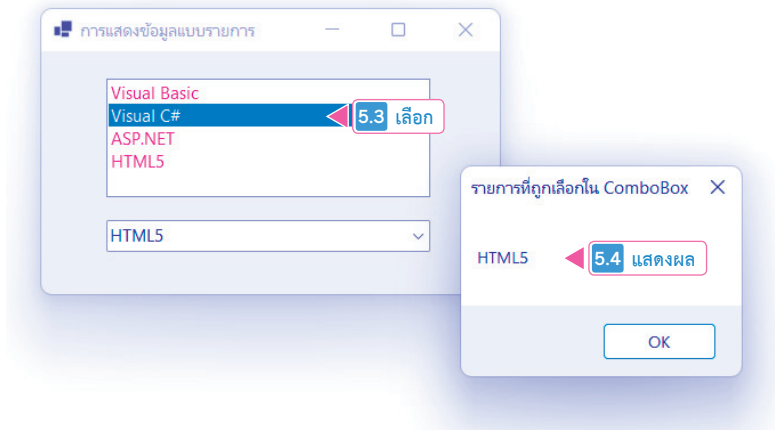
```
switch (ตัวแปรที่ตรวจสอบเงื่อนไข)
{
    case เงื่อนไขที่ 1:
        ทำงาน กรณีตรงกับเงื่อนไขที่ 1
        break;
    case เงื่อนไขที่ 2:
        ทำงาน กรณีตรงกับเงื่อนไขที่ 2
        break;
    case เงื่อนไขที่ 3:
        ทำงาน กรณีตรงกับเงื่อนไขที่ 3
        break;
    default:
        ทำงาน กรณีไม่ตรงกับเงื่อนไขข้างต้น
        break;
}
```

หลักการตรวจสอบเงื่อนไขของคำสั่ง switch case ก็คือ ถ้าเงื่อนไขตรงกับกรณีใด ก็จะเข้าไปทำงานในกรณีนั้นๆ ที่น่าสนใจก็คือ ถ้าเงื่อนไขไม่ตรงกับกรณีใดเลย ก็จะเข้าสู่กรณี **default** ผู้เขียนจะลองสร้างโปรแกรมสำหรับตรวจสอบว่า ผู้ใช้งานป้อนตัวอักษรใด ดังโค้ดต่อไปนี้

Form1.cs

```
string userInput = "";
string str = "";

switch (userInput)
{
    case "A":
        str = "A";
        break;
    case "B":
        str = "B";
```



รูปที่ 6-18 แสดงการตรวจสอบในคอนโทรล ComboBox

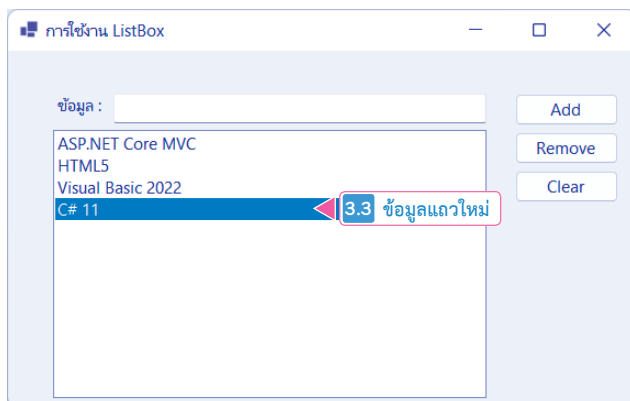
อธิบายการทำงานของโค้ด

1. ในเหตุการณ์ `Form1_Load()` ที่คอนโทรลทั้ง 2 ตัว ผู้เขียนเพิ่มรายการเข้าไป 4 รายการ และกำหนดให้รายการแรก อยู่ในสถานะถูกเลือกโดยอัตโนมัติด้วย (`.SelectedIndex = 0`)

Form1.cs

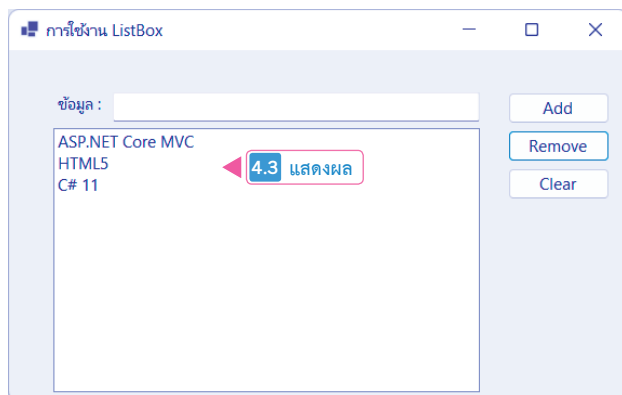
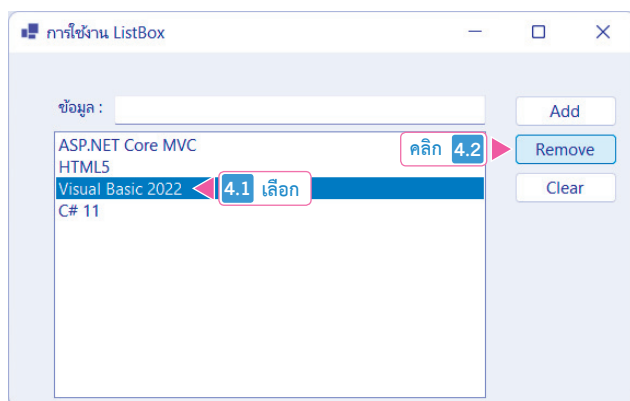
```
private void Form1_Load(object sender, EventArgs e)
{
    lsbLanguage.Items.Add("Visual Basic");
    lsbLanguage.Items.Add("Visual C#");
    lsbLanguage.Items.Add("ASP.NET");
    lsbLanguage.Items.Add("HTML5");
    lsbLanguage.SelectedIndex = 0;

    cboLanguage.Items.Add("Visual Basic");
    cboLanguage.Items.Add("Visual C#");
    cboLanguage.Items.Add("ASP.NET");
    cboLanguage.Items.Add("HTML5");
    cboLanguage.SelectedIndex = 0;
}
```



รูปที่ 6-20 กรณีเพิ่มรายการ

4. การถอดรายการให้ผู้อ่านเลือกรายการที่ต้องการถอดก่อน จากนั้นคลิกปุ่ม Remove เพื่อถอดรายการที่ถูกเลือก ดังรูปที่ 6-21



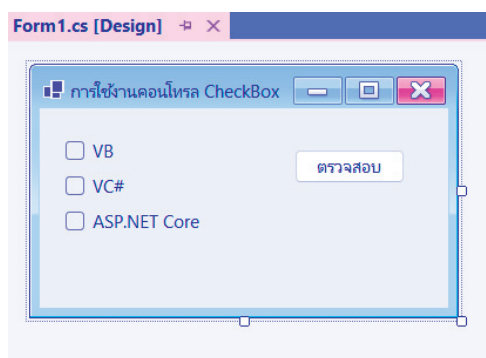
รูปที่ 6-21 กรณีถอดรายการที่ถูกคลิกเลือกออก

การสร้างตัวเลือกแบบหลายตัวเลือกด้วยคอนโทรล CheckBox

ในกรณีที่ผู้อ่านต้องการสร้างตัวเลือกแบบยินยอมให้ผู้ใช้งาน สามารถเลือกได้หลายรายการในเวลาเดียวกัน เป็นหน้าที่ของคอนโทรล CheckBox

ตัวอย่างที่ 6-8 การสร้างตัวเลือกแบบหลายตัวเลือกด้วยคอนโทรล CheckBox มีขั้นตอนดังนี้

1. ให้ผู้อ่านออกแบบฟอร์มดังรูปที่ 6-27

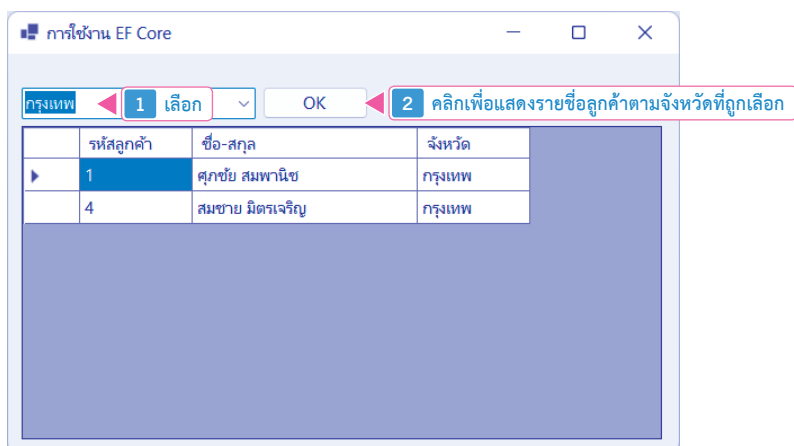


รูปที่ 6-27 แสดงฟอร์มในขณะออกแบบ

ให้ผู้อ่านกำหนดคุณสมบัติต่างๆ ดังต่อไปนี้

คอนโทรล	คุณสมบัติ	ค่าที่กำหนด
CheckBox1	Name	chkVB
	Text	VB
CheckBox2	Name	chkCS
	Text	VC#
CheckBox3	Name	chkASP
	Text	ASP.NET Core
Button1	Name	cmdCheck
	Text	ตรวจสอบ

2. จากนั้นให้ผู้อ่านเขียนโค้ดดังต่อไปนี้



รูปที่ 14-21 ผลการรันตัวอย่างที่ 14-6

จากรูปที่ 14-21 รายชื่อลูกค้าที่ปรากฏขึ้นมาจะต้องอยู่ในจังหวัดที่ถูกเลือกเท่านั้น

อธิบายการทำงานของโค้ด

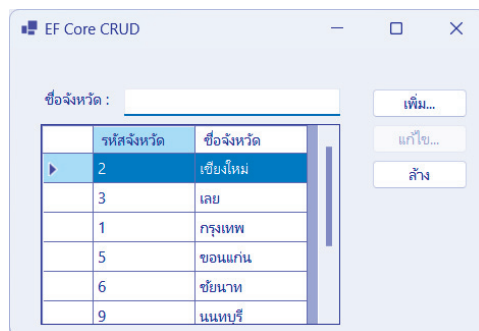
1. ในเหตุการณ์ `Form1_Load()` ให้คิวรีรายชื่อจังหวัดออกมาเก็บไว้ที่ตัวแปร `ps` ก่อน จากนั้นกำหนดให้แสดงรายชื่อจังหวัดใน `ComboBox` ที่ชื่อว่า `cboProvince` กล่าวคือ
 - **คุณสมบัติ `DisplayMember`** หมายถึง รายการที่ปรากฏขึ้นมาให้ผู้ใช้งานเลือก ในกรณีนี้คือ รายชื่อจังหวัด (ฟิลด์ `ProvinceName`)
 - **คุณสมบัติ `ValueMember`** หมายถึง รหัสจังหวัดที่ผูกกับตาราง `Customer` นั่นคือ ฟิลด์ `ProvinceId`
 - **คุณสมบัติ `DataSource`** หมายถึง ข้อมูลรายชื่อจังหวัดให้อ่านข้อมูลได้จากตัวแปร `ps`

Form1.cs

```
private void Form1_Load(object sender, EventArgs e)
{
    using var db = new ThaivbBizShopContext();

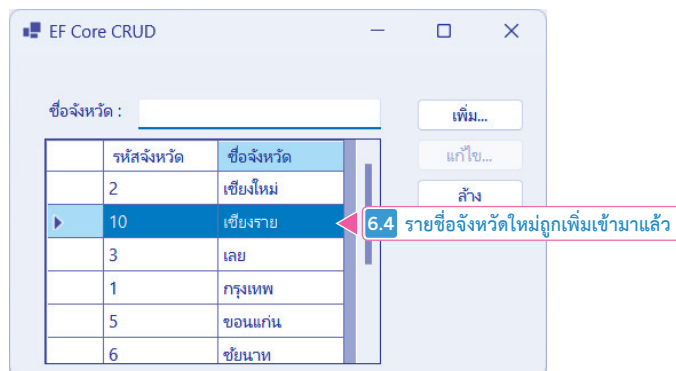
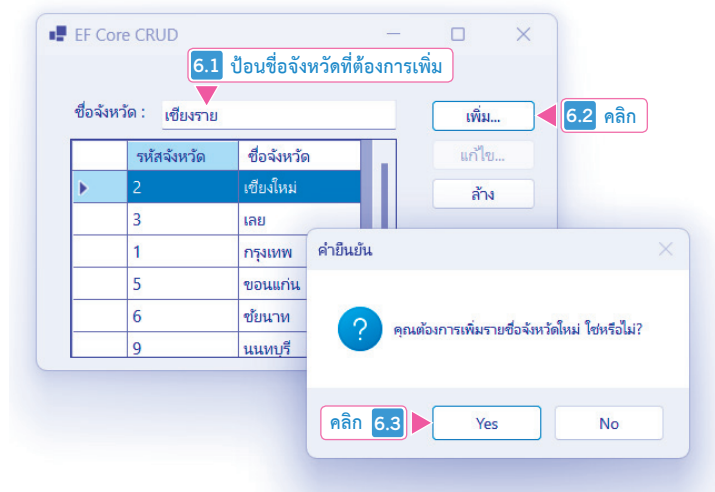
    var ps = from p in db.Provinces select p;
    cboProvince.BeginUpdate();
    cboProvince.DisplayMember = "ProvinceName";
    cboProvince.ValueMember = "ProvinceId";
    cboProvince.DataSource = ps.ToList();
    cboProvince.EndUpdate();
}
```


5. ให้ทดสอบรันโปรแกรม แสดงดังรูปที่ 15-2



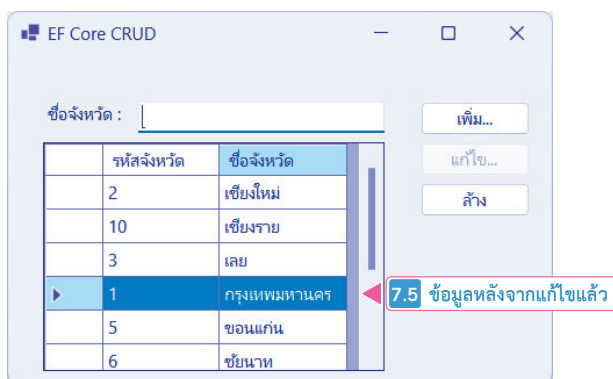
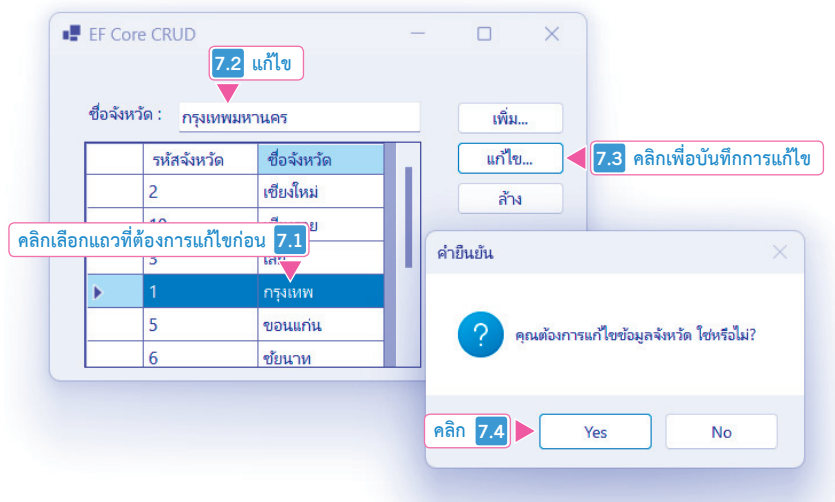
รูปที่ 15-2 ผลการรันตัวอย่างที่ 15-1

6. การเพิ่มรายชื่อจังหวัดใหม่ ทำได้โดยการคลิกปุ่ม **เพิ่ม...** ดังรูปที่ 15-3



รูปที่ 15-3 กรณีเพิ่มข้อมูลใหม่

7. ในกรณีที่ต้องการแก้ไขข้อมูล ให้คลิกเลือกก่อนแล้วค่อยแก้ไขข้อมูล ดังรูปที่ 15-4



รูปที่ 15-4 กรณีต้องการแก้ไขข้อมูล

อธิบายการทำงานของโค้ด

1. เริ่มต้นสร้างตัวแปรที่ชื่อว่า CurrentPId กำหนดค่าเริ่มต้นเป็นตัวเลข 0 ทำหน้าที่เก็บรหัสจังหวัดของแถวที่ถูกคลิกเลือก

Form1.cs

```
int CurrentPId = 0;
```

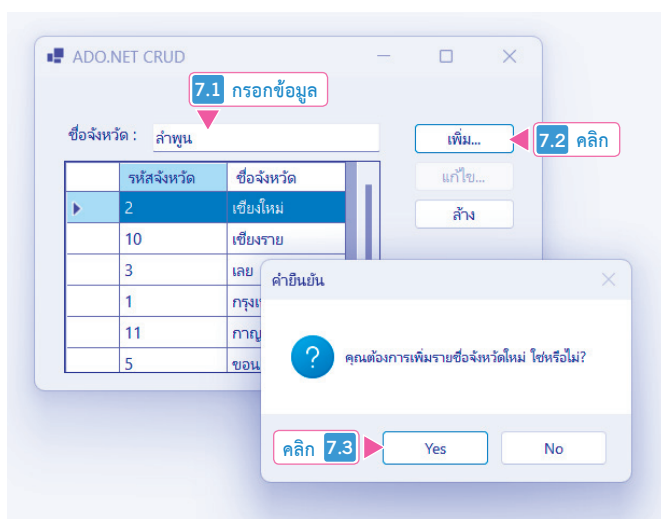
2. ในเหตุการณ์ Form1_Load() สั่งให้ฟังก์ชัน LoadData() และฟังก์ชัน ClearData() ทำงาน

6. ให้ทดสอบรันโปรแกรม แสดงดังรูปที่ 16-4



รูปที่ 16-4 ผลการรันตัวอย่างที่ 16-1

7. การเพิ่มข้อมูล ให้ป้อนข้อมูลจังหวัดใหม่ก่อน แล้วคลิกปุ่ม **เพิ่ม...** เพื่อเพิ่มข้อมูลลงฐานข้อมูล ดังรูปที่ 16-5



อธิบายการทำงานของโค้ด

1. เริ่มต้นสร้างตัวแปรขึ้นมา 3 ตัว ประกอบด้วย

- ตัวแปรออบเจกต์ **SqlConnection** ที่ชื่อว่า **Conn** ทำหน้าที่เชื่อมต่อกับฐานข้อมูล SQL Server
- ตัวแปรออบเจกต์ **SqlDataAdapter** ที่ชื่อว่า **da** ทำหน้าที่คิวรีข้อมูลประเภทอ่านข้อมูล
- ตัวแปรออบเจกต์ **DataSet** ที่ชื่อว่า **ds** ทำหน้าที่เก็บข้อมูลที่คิวรีออกมาจากฐานข้อมูล

Form1.cs

```
SqlConnection Conn = new SqlConnection();
SqlDataAdapter da;
DataSet ds = new DataSet();
```

2. สร้างตัวแปรที่ชื่อว่า CurrentPid ทำหน้าที่เก็บรหัสจังหวัดที่ถูกผู้ใช้งานคลิกเลือก และ ตัวแปร IsFind ทำหน้าที่บอกสถานะข้อมูลจังหวัดว่ามีหรือไม่ กำหนดค่าเริ่มต้น false หมายถึง ไม่มีข้อมูลจังหวัด

Form1.cs

```
int CurrentPid = 0;
bool IsFind = false;
```

3. ในเหตุการณ์ Form1_Load() สร้างตัวแปรที่ชื่อว่า strConn ทำหน้าที่เก็บข้อความเชื่อมต่อกับฐานข้อมูล thaivbBizShop

Form1.cs

```
private void Form1_Load(object sender, EventArgs e)
{
    string strConn = "Server=.\SQLExpress;Database=thaivbBizShop;Trusted_Connection=True;
    TrustServerCertificate=True;";
```

4. ตรวจสอบว่าตัวแปรออบเจกต์ Conn มีการเชื่อมต่อเดิมค้างอยู่หรือไม่ (Conn.State == ConnectionState.Open) แยกออกเป็น 2 กรณีคือ

- **กรณีมีการเชื่อมต่อเปิดค้างอยู่** ให้ปิดก่อน (Conn.Close())
- **กรณีไม่มีการเชื่อมต่อเดิม** กำหนดให้เชื่อมต่อ (Conn.Open()) โดยอาศัยข้อความเชื่อมต่อที่เก็บอยู่ในตัวแปร strConn

เมื่อได้มีการเชื่อมต่อกับฐานข้อมูลมาแล้ว ก็จะสั่งให้ฟังก์ชัน LoadData() ทำงาน



Unit 17

พื้นฐานการพัฒนา Web Application ด้วย ASP.NET Core MVC

การสร้างเว็บไซต์ หรือ Web Apps ขึ้นมาในโลกของ .NET Core ของภาษา C# เป็นหน้าที่ของโปรเจกต์ประเภท ASP.NET Core โดยที่ผู้เขียนเลือกนำเสนอโปรเจกต์ 2 ประเภทคือ

1. **โปรเจกต์ ASP.NET Core MVC** เป็นการพัฒนา Web Apps โดยการแบ่งงานออกเป็น 3 ส่วน ได้แก่
 - **M ย่อมาจากคำว่า Model** หมายถึง ส่วนของระบบจัดเก็บข้อมูล เช่น ระบบฐานข้อมูล, คลาสต่างๆ เป็นต้น
 - **V ย่อมาจากคำว่า View** หมายถึง ส่วนแสดงผล เกิดจากภาษาสคริปต์ Razor มีนามสกุล *.cshtml
 - **C ย่อมาจากคำว่า Controller** หมายถึง “คนกลาง” ทำหน้าที่ติดต่อกับส่วน Model เพื่อสั่งให้สร้างส่วนแสดงผลร่วมกับส่วนของ View ส่งผลให้ผู้ใช้งานเห็นส่วนแสดงผลต่างๆ ในบราวเซอร์
2. **โปรเจกต์ ASP.NET Core Web API** การแบ่งแยกพัฒนาโปรเจกต์แบบ Front End กับ Back End ถือเป็นหัวใจหลักของการพัฒนาในยุคปัจจุบัน โดยที่หนังสือเล่มนี้แนะนำการสร้างโปรเจกต์ประเภท ASP.NET Core Web API

จากรูปที่ 19-12 ผลการเชื่อมต่อที่พาร /api/Customers สมบูรณ์แบบ (รหัส 200 Success) แต่ไม่มีข้อมูลลูกค้า (Response body เป็นค่าว่าง) เพราะว่าเป็นฐานข้อมูลจำลองที่เราเพิ่งสร้างขึ้นมาในหน่วยความจำนั่นเอง

การทดสอบการทำงานของ Web API ด้วย Swagger

Web API คือ การทำงานฝั่งหลังบ้าน Back End มีแต่ข้อมูลเพียงอย่างเดียวไม่มีส่วนแสดงผล โปรเจกต์ ASP.NET Core Web API กำหนดให้ใช้เครื่องมือที่เรียกว่า Swagger ทำหน้าที่เป็นรายงานประจำโปรเจกต์ของเรา ผู้อ่านสามารถใช้ Swagger ทดสอบการทำงาน CRUD ได้อีกด้วย

การทำงานของ Web API ที่เราได้มาจากเครื่องมือของโปรแกรม Visual Studio ประกอบด้วย

1. **การทำงานแบบ GET** ของพาร /api/Customers หมายถึง การแสดงข้อมูลลูกค้าทั้งหมดแบบไม่มีเงื่อนไข
2. **การทำงานแบบ POST** ของพาร /api/Customers หมายถึง การเพิ่มข้อมูลลูกค้า 1 รายการ
3. **การทำงานแบบ GET** ของพาร /api/Customers/{id} หมายถึง การร้องขอข้อมูลลูกค้า 1 รายการตามรหัส id ที่ถูกส่งเข้ามา
4. **การทำงานแบบ PUT** ของพาร /api/Customers/{id} หมายถึง การอัปเดตหรือแก้ไขข้อมูลลูกค้า 1 รายการตามรหัส id ที่ถูกส่งเข้ามา
5. **การทำงานแบบ DELETE** ของพาร /api/Customers/{id} หมายถึง การลบข้อมูลลูกค้า 1 รายการตามรหัส id ที่ถูกส่งเข้ามา

5. จากนั้นให้ผู้อ่านเขียนโค้ดต่อไปนี้

โค้ด C# 11 ที่ 20-1 การสร้าง CRUD แบบ Minimal APIs (Program.cs)

```
using Microsoft.EntityFrameworkCore;
using UsingMinimalAPIs.Models;

var builder = WebApplication.CreateBuilder(args);
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

builder.Services.AddDbContext<CustomerDbContext>(options =>
    options.UseInMemoryDatabase("CustomerDb")
);

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}
app.UseHttpsRedirection();

app.MapPost("/customer", async (CustomerDbContext _db, Customer c) =>
{
    _db.Customer.Add(c);
    await _db.SaveChangesAsync();

}).WithName("CreateCustomer")
.Produces<Customer>(StatusCodes.Status201Created);

app.MapGet("/customer", async (CustomerDbContext _db) =>
{
    return await _db.Customer.ToListAsync();
}).WithName("GetCustomers");

app.MapGet("/customer/{id}", async (CustomerDbContext _db, int id) =>
{
    var cs = await _db.Customer.FindAsync(id);
    if (cs is null)
    {
        return Results.NotFound(StatusCodes.Status404NotFound);
    }

    return Results.Ok(cs);
}
```



บทที่
21

พื้นฐานการพัฒนา Web Apps แบบคอนเทนเนอร์ด้วย Docker

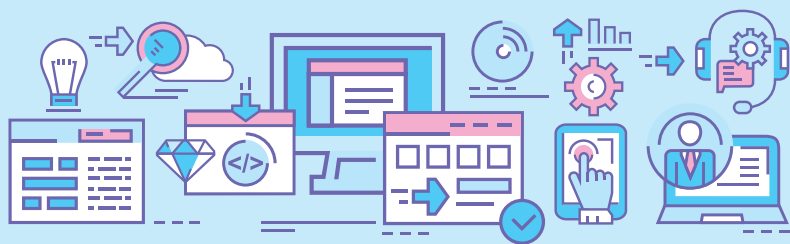
การพัฒนาแอปพลิเคชันในยุคเดิม เป็นรูปแบบที่เราทำกันมาอย่างยาวนาน กล่าวคือ เมื่อเราต้องการพัฒนา Desktop Apps, Web Apps หรือ Mobile Apps บนแพลตฟอร์มใดก็ตาม ก็ต้องมีการดาวน์โหลดและติดตั้ง SDK, Library หรือ Framework ต่างๆ เพื่อให้เครื่องของเราพร้อมใช้งานเครื่องมือต่างๆ ตามข้อกำหนดของแต่ละแพลตฟอร์ม ถือเป็นรูปแบบที่เราคุ้นเคยกันเป็นอย่างดี

แต่ในยุคปัจจุบันมีรูปแบบการพัฒนาแอปอีกลักษณะหนึ่งที่เรียกว่า **"คอนเทนเนอร์" (Container)** เป็นวิธีการที่ผู้อ่านสามารถสร้างอิมเมจ (Image) ขึ้นมา และกำหนดให้อิมเมจนั้นๆ ประกอบไปด้วย SDK, Library หรือ Framework ต่างๆ ส่งผลให้อิมเมจดังกล่าว มีสภาพแวดล้อมที่สามารถพัฒนาแอปตามแพลตฟอร์มได้ตามที่ผู้อ่านต้องการ

ผู้อ่านสามารถสร้างอิมเมจได้หลายชุดหลายรูปแบบ มีอิสระต่อกัน โดยที่แต่ละอิมเมจไม่มีความเกี่ยวข้องระหว่างกันแต่อย่างใด และยังรวมไปถึงการไม่เกี่ยวข้องกับระบบปฏิบัติการที่ผู้อ่านใช้งานอีกด้วย

เมื่อได้อิมเมจมาแล้ว ผู้อ่านสามารถนำอิมเมจนั้นๆ ไปสร้างคอนเทนเนอร์ (Container) เพื่อใช้งาน SDK/Library/Framework เหล่านั้นได้เหมือนกับที่เราติดตั้งใน Windows ของเรา

โปรแกรมที่ช่วยให้ผู้อ่านสามารถพัฒนาแอปในรูปแบบคอนเทนเนอร์มีหลายตัว เนื้อหาในหนังสือเล่มนี้เป็นการใช้งานโปรแกรมที่ชื่อว่า Docker บนระบบปฏิบัติการ Windows 10/11 ผู้อ่านท่านใดที่มีความรู้คำสั่งพื้นฐานของ Docker Image และ Container มาแล้ว สามารถข้ามเนื้อหาบทนี้ไปได้เลย



บทที่
23

การเผยแพร่อิมเมจแบบสาธารณะกับ Docker Registry

จากจุดเริ่มต้นของการใช้งาน Docker โดยการดาวน์โหลด หรือ pull อิมเมจ (pull image) ของคนอื่น ๆ มาใช้งานในเครื่องของเรา มาสู่การทำงานตรงกันข้ามของบทนี้ นั่นคือ เราจะอัปโหลด หรือ push อิมเมจ (push image) ของเราให้คนอื่น ๆ ใช้งานบ้าง

พื้นที่จัดเก็บอิมเมจสาธารณะ เรียกว่า Docker Registry เปรียบเสมือนกับเป็นคลังเก็บอิมเมจต่างๆ อยู่ที่ hub.docker.com ผู้อ่านต้องสมัครสมาชิกด้วย บัญชีของผู้เขียนชื่อว่า `thaivb` เราต้องใช้ชื่อบัญชีในเนื้อหาของบทนี้

Note

ในขณะที่หนังสือเล่มนี้จัดทำนาย ผู้เขียนลบอิมเมจออกจากบัญชีทั้งหมดแล้ว

การ push อิมเมจสู่ Docker Registry

การ push อิมเมจสู่ Docker Registry มีขั้นตอนดังนี้

1. ให้ผู้อ่านพิมพ์คำสั่ง **docker images** เพื่อตรวจสอบก่อนว่า ในเครื่องของเรามีอิมเมจอะไรบ้าง ในทางปฏิบัติแล้ว เมื่อผู้อ่านพัฒนาแอปแบบคอนเทนเนอร์ด้วย Docker จะมีรายการอิมเมจต่างๆ มากมายติดตั้งอยู่ในเครื่องของเรา

Command Prompt

```
docker images
```