Contents:

VB.NET Programming: Beginner

(Integrative-Generative AI Edition)

Student Price Book Center

คำนำ

การเขียนโปรแกรมเป็นทักษะพื้นฐานสำคัญในโลกปัจจุบัน ที่ซึ่งเทคโนโลยีและซอฟต์แวร์มีบทบาทสำคัญในทุกอุตสาหกรรม VB.NET เป็นหนึ่งในภาษาโปรแกรมที่ได้รับความนิยมจากนักพัฒนาโปรแกรมทั้งมือใหม่และมืออาชีพ ด้วยความสามารถในการพัฒนาแอปพลิเคชันหลากหลายรูปแบบ ทั้ง Desktop, Web และ Mobile โดยอาศัย .NET Framework, .NET Core และ .NET 5+ ที่มีประสิทธิภาพและความยืดหยุ่นสูง

หนังสือเล่มนี้ถูกออกแบบขึ้นเพื่อช่วยให้ผู้เริ่มต้นเรียนรู้ VB.NET อย่างเป็นระบบ เริ่มตั้งแต่ความเข้าใจพื้นฐานเกี่ยวกับภาษา การติดตั้งเครื่องมืออย่าง Visual Studio หรือ Visual Studio Code ไปจนถึงการสร้างโปรเจ็กต์แรกแบบ Console Application บทแรก **Introduction to VB.NET** จะพาผู้อ่านทำความเข้าใจกับประวัติและวิวัฒนาการของ VB.NET ความแตกต่างจาก VB6 รวมถึงแนวคิดเกี่ยวกับ .NET Framework, .NET Core และ .NET 5+ ซึ่งเป็นรากฐานสำคัญก่อนที่จะเริ่มเขียนโค้ด ต่อจากนั้นหนังสือจะพาผู้อ่านเข้าสู่ **Basic Syntax** ซึ่งเป็นหัวใจของการเขียนโปรแกรมใน VB.NET โดยอธิบายโครงสร้างโปรแกรม เช่น Module และ Sub Main การเขียนคำสั่งแสดงผลด้วย Console.WriteLine การรับค่าจากผู้ใช้ด้วย Console.ReadLine และการเขียนคอมเมนต์อย่างถูกต้องเพื่อให้โค้ดอ่านง่ายและสามารถบำรุงรักษาได้สะดวก

เมื่อเข้าใจโครงสร้างพื้นฐานแล้ว ผู้อ่านจะได้เรียนรู้เรื่อง **Variables & Data Types** ซึ่งเป็นส่วนสำคัญของการเก็บและจัดการข้อมูลในโปรแกรม ครอบคลุมชนิดข้อมูลพื้นฐาน เช่น Integer, Double, Boolean, String, Char และ DateTime พร้อมทั้งการประกาศตัวแปรด้วย Dim, Const และ Static และการแปลงชนิดข้อมูลด้วยฟังก์ชัน CInt, CDbl, CStr เพื่อให้สามารถประมวลผลข้อมูลได้อย่างถูกต้องและปลอดภัย

หลังจากนั้น หนังสือจะนำเข้าสู่หัวข้อ **Operators** และ **Control Flow** ที่ช่วยให้โปรแกรมสามารถคำนวณและตัดสินใจตามเงื่อนไขต่าง ๆ ได้อย่างมีระบบ ครอบคลุม Arithmetic, Comparison, Logical และ Assignment Operators รวมถึงโครงสร้างการควบคุม เช่น If / ElseIf / Else, Select Case, For Loop, For Each Loop, While, Do While, Do Until และการใช้ Exit และ Continue เพื่อจัดการลูปและเงื่อนไขอย่างเหมาะสม

บทต่อไปจะกล่าวถึง **Functions & Procedures** ซึ่งช่วยให้นักพัฒนาสามารถจัดระเบียบโค้ด แยกงานเป็นส่วนย่อย และใช้งานซ้ำได้อย่างมีประสิทธิภาพ โดยอธิบายความแตกต่างระหว่าง Sub และ Function การส่งค่าแบบ ByVal และ ByRef การใช้ Optional Parameters และ Default Values รวมถึงการส่งค่ากลับด้วย Return Values ทั้งหมดนี้จะช่วยให้ผู้อ่านสามารถสร้างโปรแกรมที่มีโครงสร้างชัดเจน อ่านง่าย และบำรุงรักษาได้ง่าย

หนังสือเล่มนี้ยังมีตัวอย่างบูรณาการในแต่ละบท เพื่อให้ผู้อ่านได้ฝึกเขียนโปรแกรมจริงและเข้าใจหลักการทำงานของโค้ดในบริบทต่าง ๆ การฝึกปฏิบัติจะช่วยสร้างความมั่นใจและความเชี่ยวชาญในการเขียนโปรแกรม VB.NET ตั้งแต่ระดับเริ่มต้นจนสามารถต่อยอดไปสู่การพัฒนาแอปพลิเคชันขั้นสูง

ท้ายที่สุด ผู้อ่านที่ศึกษาเนื้อหาในหนังสือเล่มนี้จะได้รากฐานที่มั่นคงในภาษา VB.NET เข้าใจหลักการเขียนโปรแกรมอย่างเป็นระบบ และสามารถสร้างโปรแกรมที่มีตรรกะชัดเจน มีประสิทธิภาพ และพร้อมต่อการพัฒนาต่อในโลกของเทคโนโลยีสมัยใหม่ หนังสือเล่มนี้จึงเหมาะสำหรับผู้เริ่มต้นทุกคนที่ต้องการเข้าใจ VB.NET อย่างลึกซึ้งและพร้อมนำไปใช้งานจริง

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 1

Introduction to VB.NET

(Introduction to VB.NET)

เนื้อหา

- Introduction to VB.NET
- Introduction to VB.NET (ฉบับเชิงลึก)
- ประวัติและวิวัฒนาการของ VB.NET
- ความแตกต่างระหว่าง VB6 และ VB.NET
- NET Framework, .NET Core และ .NET 5+
- การติดตั้ง Visual Studio / Visual Studio Code สำหรับ VB.NET
- การสร้างโปรเจ็กต์แรก (Console Application) ด้วย VB.NET

บทนำ: Introduction to VB.NET

Visual Basic .NET (VB.NET) เป็นหนึ่งในภาษาการเขียนโปรแกรมเชิงวัตถุที่พัฒนาขึ้นโดย Microsoft เพื่อเป็นภาษารุ่นใหม่ทดแทน Visual Basic 6 (VB6) ซึ่งได้รับความนิยมอย่างแพร่หลายในช่วงปลายทศวรรษ 1990 และต้นปี 2000 VB.NET ถูกออกแบบให้สามารถทำงานร่วมกับ .NET Framework ได้อย่างเต็มประสิทธิภาพ พร้อมรองรับแนวคิดการพัฒนาโปรแกรมสมัยใหม่ เช่น การเขียนโปรแกรมเชิงวัตถุ (OOP) และการจัดการข้อผิดพลาดเชิงโครงสร้าง

ความแตกต่างสำคัญระหว่าง VB6 และ VB.NET อยู่ที่โครงสร้างและความสามารถของภาษา VB.NET ได้รวมแนวคิดของ .NET Framework ทำให้สามารถใช้คุณสมบัติขั้นสูง เช่น การจัดการหน่วยความจำอัตโนมัติ (Garbage Collection), การสนับสนุนชนิดข้อมูลที่ปลอดภัย (Strongly Typed Data) และการเข้าถึงคลาสและไลบรารีมาตรฐานของ .NET ได้โดยตรง

นอกจาก .NET Framework เดิมแล้ว Microsoft ยังได้พัฒนา .NET Core และล่าสุดคือ .NET 5+ ซึ่งเป็นแพลตฟอร์มแบบข้ามระบบปฏิบัติการ (Cross-platform) ที่รองรับการสร้างแอปพลิเคชันทั้งบน Windows, macOS และ Linux VB.NET จึงสามารถนำไปใช้สร้างโปรแกรมได้หลากหลายประเภท ตั้งแต่แอปพลิเคชันเดสก์ท็อป ไปจนถึงเว็บและบริการบนคลาวด์

การเริ่มต้นใช้งาน VB.NET จำเป็นต้องติดตั้งเครื่องมือพัฒนา (IDE) เช่น Visual Studio หรือ Visual Studio Code ซึ่งทั้งสองรองรับการสร้างโปรเจ็กต์ VB.NET ได้สะดวก Visual Studio ให้ฟีเจอร์ครบครันสำหรับนักพัฒนาที่ต้องการเครื่องมือ GUI แบบเต็มรูปแบบ ขณะที่ Visual Studio Code เหมาะสำหรับผู้ที่ต้องการ IDE ขนาดเบาและสามารถปรับแต่งได้

บทแรกนี้จะพาผู้อ่านเรียนรู้ตั้งแต่การติดตั้งและตั้งค่า Visual Studio จนถึงการสร้างโปรเจ็กต์แรกของ VB.NET ซึ่งเป็น Console Application การเรียนรู้การสร้างโปรแกรมคอนโซลจะช่วยให้เข้าใจพื้นฐานของโครงสร้างโปรแกรม การเขียนคำสั่ง และการรันโปรแกรมบน IDE ได้อย่างมั่นใจ

นอกจากนี้ยังมีการอธิบายพื้นฐานของ .NET Framework, การเรียกใช้คลาสมาตรฐาน และการทำงานของคอนโซลอย่างละเอียด เพื่อให้ผู้อ่านสามารถพัฒนาความเข้าใจอย่างเป็นระบบ และสามารถต่อยอดไปสู่การพัฒนาแอปพลิเคชันแบบกราฟิก (GUI) และแอปพลิเคชันเว็บในบทต่อไปได้

ด้วยความสามารถและความยืดหยุ่นของ VB.NET บทนี้จึงเป็นจุดเริ่มต้นที่สำคัญสำหรับนักพัฒนาทุกระดับ ไม่ว่าจะเป็นผู้เริ่มต้นหรือผู้ที่มีพื้นฐาน VB6 อยู่แล้ว การเข้าใจประวัติ ความแตกต่าง และการใช้งานเบื้องต้นของ VB.NET จะช่วยสร้างรากฐานที่แข็งแรงสำหรับการพัฒนาโปรแกรมขั้นสูงและการใช้เทคโนโลยี .NET รุ่นใหม่ในอนาคต

Introduction to VB.NET

- ประวัติและวิวัฒนาการของ VB.NET
- ความแตกต่างจาก VB6
- .NET Framework, .NET Core และ .NET 5+
- การติดตั้ง **Visual Studio / Visual Studio Code**
- การสร้างโปรเจ็กต์แรก (Console Application)

บทที่ 1: Introduction to VB.NET

1) ประวัติและวิวัฒนาการของ VB.NET

- **จุดกำเนิด**: VB.NET เกิดในยุค .NET 1.0 (ประมาณปี 2002) เพื่อสืบทอดจาก Visual Basic 6 (VB6) แต่ยกระดับให้เป็นภาษา **Object-Oriented** เต็มรูปแบบ ทำงานบน **CLR (Common Language Runtime)** และเข้าถึง **Base Class Library (BCL)** ของ .NET ได้ทั้งหมด
- **ยุค .NET Framework**: ช่วง .NET 2.0–4.x เพิ่มฟีเจอร์สำคัญ เช่น **Generics**, **LINQ**, **Async/Await**, รวมถึงไลบรารีจำนวนมาก (WinForms/WPF/ASP.NET)
- **ยุค .NET Core และ .NET 5+**: Microsoft รวมแบรนด์เป็น “**.NET**” (เริ่มจาก .NET 5) ปัจจุบันคือสาย **cross-platform** (Windows, macOS, Linux) เน้นประสิทธิภาพสูง โครงสร้างโปรเจ็กต์แบบ **SDK-style** และเครื่องมือสมัยใหม่
- **สถานะปัจจุบัน**: VB.NET ยังรองรับบน .NET สำหรับงาน **Console, Class Library, WinForms, WPF (บน Windows)** และงาน Enterprise/Legacy ที่ต้องการความเสถียร แม้จังหวะเพิ่มฟีเจอร์ภาษาใหม่จะช้ากว่า C# แต่ยังคง **Stable & Supported** สำหรับงานเดสก์ท็อปและระบบองค์กร

---

2) ความแตกต่างจาก VB6 (สิ่งที่ควรรู้เวลาเปลี่ยนมา VB.NET)

ประเด็น	VB6	VB.NET
รันไทม์	COM/ActiveX	**CLR** (.NET Runtime) + BCL
Paradigm	Event-driven, OOP ไม่เต็มรูปแบบ	**OOP เต็มรูปแบบ** (Class, Inheritance, Interfaces, Generics)
ชนิดข้อมูล	Variant ใช้กันแพร่หลาย, Integer = 16 บิต	**Strongly-typed**, Integer = **32 บิต**, แทน Variant ด้วย Object/ชนิดที่ชัดเจน
ข้อผิดพลาด	On Error GoTo ...	**Try … Catch … Finally**, Throw Exceptions
UI Tech	VB6 Forms	**WinForms / WPF**
Deploy	ลงทะเบียน COM, DLL Hell	**XCOPY/ClickOnce/MSIX**, จัดการแพ็กเกจผ่าน **NuGet**
อาเรย์	อาจเริ่มที่ index 1	ปกติ **เริ่มที่ 0**
ดีฟอลต์พร็อพเพอร์ตี	มี default properties มาก	ตัด default properties ออก ⟶ โค้ด **ชัดเจนกว่า**
มัลติเธรด	จำกัด	**Tasks/Async-Await**, TPL
กับดักยอดฮิตตอนย้ายมา .NET		

- Integer ของ VB.NET คือ 32 บิต (เท่า Long ของ VB6) ⟶ ระวังการแปลงชนิด
- หลีกเลี่ยงการพึ่งพา default properties/late binding; เปิด Option Strict On เสมอเพื่อความปลอดภัยและประสิทธิภาพ
- โครง UI และอีเวนต์ใน WinForms/WPF มีรูปแบบ **event/delegate** ต่างจาก VB6

---

3) .NET Framework, .NET Core และ .NET 5+ (ปัจจุบันเรียกรวมว่า “.NET”)

- **.NET Framework (4.x)**: รันบน **Windows เท่านั้น** เหมาะกับระบบองค์กร/เดสก์ท็อปที่ยังผูกกับ Windows API และเทคโนโลยีเดิม
- **.NET Core ⟶ .NET 5+ (เรียกรวม “.NET”)**:
  - สาย **Cross-platform**, เร็ว ประหยัดหน่วยความจำ, รองรับ **self-contained** publish
  - โครงสร้างโปรเจ็กต์แบบ **SDK-style** ง่ายต่อการจัดการแพ็กเกจด้วย **NuGet**
  - **VB.NET บน .NET ปัจจุบัน**: รองรับ **Console, Class Library, WinForms, WPF (Windows)**
  - **ASP.NET Core** อย่างเป็นทางการ **ไม่มีเทมเพลต VB** (นิยมใช้ C#) แต่ VB library นำไปอ้างอิงในโซลูชัน C# ได้

- **เลือกใช้เมื่อไรดี**
  - ถ้าสร้างระบบใหม่/ต้องการย้ายสู่อนาคต ⟶ เลือก **.NET (5+)**
  - ถ้าระบบเดิมพึ่งพา .NET Framework/COM/Windows-only หนักมาก ⟶ อาจอยู่บน **.NET Framework** ต่อจนกว่าจะวางแผนย้าย

---

4) การติดตั้งเครื่องมือ: Visual Studio / Visual Studio Code

ทางเลือก A: Visual Studio (แนะนำสำหรับ VB.NET)

เหมาะสุดสำหรับ WinForms/WPF/Designer และ Debugger ที่สมบูรณ์

1. ดาวน์โหลด **Visual Studio Community** (ฟรีสำหรับบุคคล/ทีมเล็ก)
2. ตอนเลือก **Workloads** ให้ติ๊ก:
   - **.NET desktop development** (จำเป็นสำหรับ WinForms/WPF/Console)
   - (ทางเลือก) **Data storage and processing** ถ้าจะเชื่อมฐานข้อมูล/SSDT
3. หลังติดตั้ง คุณจะได้ **Designer**, **IntelliSense**, **Windows Form Designer**, **WPF Designer**, Debugger ครบถ้วน

ทางเลือก B: Visual Studio Code

- เหมาะกับ **Console/Class Library** และงานแก้ไขโค้ดเบื้องต้น
- ติดตั้ง **.NET SDK** และส่วนขยายที่เกี่ยวข้อง (เช่น C# Dev Kit ใช้ช่วยเรื่องดีบัก/เทสต์ระดับหนึ่ง)
- หมายเหตุ: **ภาษา VB ใน VS Code ยังไม่มีภาษาเสริมที่สมบูรณ์แบบเท่า C#** (ไม่มี Form Designer) เหมาะกับคนที่คุ้นกับ CLI

สิ่งที่ต้องมีร่วมกัน

- **.NET SDK** เวอร์ชันปัจจุบัน (เพียงติดตั้ง SDK ก็ได้ dotnet CLI มาให้)
- โฟกัสงาน Windows Desktop ⟶ ใช้ **Visual Studio** จะสะดวกกว่าอย่างชัดเจน

---

5) การสร้างโปรเจ็กต์แรก (Console Application)

วิธีที่ 1: สร้างด้วย Visual Studio

1. เปิด Visual Studio ⟶ **Create a new project**
2. ค้นหา/เลือก **Console App** และเลือกภาษา **Visual Basic**
3. ตั้งชื่อโปรเจ็กต์ (เช่น HelloVbNet) และโฟลเดอร์ปลายทาง ⟶ Create
4. ใส่โค้ดต่อไปนี้ใน Module1.vb (หรือไฟล์หลักที่ VS สร้างให้)

```
Imports System


Module Program
```

```
    Sub Main(args As String())
        Console.WriteLine("Hello, VB.NET!")
        Console.Write("พิมพ์ชื่อของคุณ: ")
        Dim name As String = Console.ReadLine()
        Console.WriteLine($"สวัสดี {name}, ยินดีต้อนรับสู่ VB.NET")
        Console.WriteLine("กด Enter เพื่อออก...")
        Console.ReadLine()
    End Sub
End Module
```

5. กด **F5** เพื่อ Run (Debug) หรือ **Ctrl+F5** เพื่อ Run โดยไม่ Debug

โครงสร้างโปรเจ็กต์ (โดยย่อ)

```
HelloVbNet/
  ├── HelloVbNet.vbproj   'ไฟล์โปรเจ็กต์ (SDK-style)
  ├── Module1.vb          'จุดเริ่มโปรแกรม (Sub Main)
  └── My Project/         'ไฟล์ตั้งค่า (บางเวอร์ชัน)
```

ตัวอย่างไฟล์โปรเจ็กต์แบบ **SDK-style** (อัตโนมัติจาก VS/CLI):

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
    <RootNamespace>HelloVbNet</RootNamespace>
    <OptionStrict>On</OptionStrict>
    <OptionInfer>On</OptionInfer>
    <OptionExplicit>On</OptionExplicit>
  </PropertyGroup>
</Project>
```

เคล็ดลับ: เปิด Option Strict/Infer/Explicit เป็น **On** เพื่อโค้ดที่ชัดเจน ปลอดภัย และจับบั๊กได้ตั้งแต่ตอนคอมไพล์

วิธีที่ 2: สร้างด้วย dotnet CLI (ใช้ได้ทั้งบน Windows/macOS/Linux)

1. เปิด Terminal/Command Prompt แล้วรัน:

```
dotnet new console -lang VB -n HelloVbNet
cd HelloVbNet
dotnet run
```

2. แก้ไขไฟล์ Program.vb (หรือ Module1.vb แล้วแต่เทมเพลต) เป็นโค้ดตัวอย่าง “Hello, VB.NET!” ด้านบน จากนั้น

dotnet run

เช็กลิสต์แนวปฏิบัติที่ดีตั้งแต่วันแรก

- เปิด Option Strict On, Option Explicit On, Option Infer On
- ตั้งชื่อ **Namespace/Class/Method/Property** ให้สื่อความหมาย
- แยก **Layer** ตั้งแต่เล็ก ๆ (เช่น ชั้นสำหรับ Console/UI, ชั้นสำหรับ Logic)
- ใช้ **NuGet** จัดการไลบรารี (เช่น System.Text.Json, Dapper, Serilog)
- เริ่มฝึก **Unit Test** ตั้งแต่โปรเจ็กต์เล็ก (MSTest/xUnit + โครงสร้างที่ทดสอบได้)

Introduction to VB.NET (ฉบับเชิงลึก)

1) ประวัติและวิวัฒนาการของ VB.NET (และสิ่งที่เกิด “ข้างใต้ฝากระโปรง”)

- **จาก VB6 ⟶ VB.NET (2002)**
  VB.NET ถูกออกแบบใหม่ให้เข้ากับ **CLR (Common Language Runtime)** ของ .NET: โค้ดจะคอมไพล์เป็น **IL (Intermediate Language)** เก็บใน **Assembly** (.exe/.dll) แล้วถูก JIT (Just-In-Time) แปลงเป็นโค้ดเครื่องขณะรันจริง ⟶ ได้ **GC (Garbage Collector)**, **Type Safety**, **Security** และเข้าถึง **Base Class Library (BCL)** เดียวกับ C#
- **ช่วง .NET 2.0–4.x**
  เพิ่ม **Generics**, **Nullable**, **LINQ**, **Lambda**, **Async/Await**, WinForms/WPF/ASP.NET ครบ—VB.NET ได้ฟีเจอร์ภาษาเรื่อย ๆ (เช่น String Interpolation, Implicit Line Continuation)
- **ยุค .NET Core ⟶ .NET 5+ (เรียกรวม “.NET”)**
  รวมแบรนด์เป็น “.NET” เดียว เน้น cross-platform, ประสิทธิภาพสูง, โปรเจกต์แบบ **SDK-style** (ไฟล์ .vbproj เรียบง่าย), เครื่องมือ CLI (dotnet)
- **สถานะปัจจุบันของ VB.NET**
  - เหมาะกับ **Windows Desktop (WinForms/WPF)**, Console, Library, งาน Enterprise/Legacy
  - เทมเพลต **ASP.NET Core ไม่มีสำหรับ VB** โดยปริยาย (แต่ Library VB อ้างอิงจากโครงการ C# ได้)
  - จังหวะเพิ่มฟีเจอร์ภาษา “ช้ากว่า C#” แต่เสถียร ใช้งานระยะยาวได้ดี

คีย์เวิร์ดเบื้องลึก: **CLR / IL / JIT / GC / Assembly / BCL / SDK-style Project**
เข้าใจสิ่งเหล่านี้ = เข้าใจ “เหตุผล” ของพฤติกรรมภาษา, ประสิทธิภาพ, และการดีบัก

---

2) ความแตกต่างจาก VB6 (ย้ายบ้านอย่างปลอดภัย)

ตารางสรุปประเด็นที่ส่งผลต่อโค้ด/สถาปัตยกรรมมากที่สุด

หัวข้อ	VB6	VB.NET (แนะนำแนวทาง)
รันไทม์	COM/ActiveX	CLR + BCL (ปลดล็อก GC, Type Safety, Multithreading)
Paradigm	Event-driven, OOP ยังไม่เต็ม	**OOP เต็มรูปแบบ** (Class, Inheritance, Interface, Generics)
ชนิดข้อมูล	Variant ใช้บ่อย, Integer 16 บิต	**Strongly Typed**, Integer = **32 บิต**, ใช้ Object/Generic เมื่อจำเป็น
ข้อผิดพลาด	On Error GoTo	**Try … Catch … Finally**, Throw Exception
อาเรย์/คอลเลกชัน	มักเริ่ม index 1	เริ่ม **0-based**; ใช้ **List(Of T)**, Dictionary(Of TKey,TValue)
UI	VB6 Forms	**WinForms / WPF** (Event/Delegate Model ต่างไป)
ดีฟอลต์พร็อพเพอร์ตี/late binding	พบได้บ่อย	**ปิด** ด้วย Option Strict On ⟶ โค้ดชัด/เร็ว/ปลอดภัยกว่า
ดีพลอย	DLL Hell, regsvr32	NuGet, ClickOnce/MSIX, Self-contained publish

กับดักยอดฮิตเมื่อย้าย:

1. Integer ของ VB.NET เป็น **32 บิต** (เท่า Long ของ VB6) ⟶ ตรวจการแปลงชนิด
2. เลิก **Default Property** (เขียนให้ชัดเจน)
3. ปิด **Late Binding** ⟶ เปิด Option Strict On เสมอ
4. อีเวนต์/ดีไซน์เนอร์ของ **WinForms/WPF** ต่างจาก VB6—เรียนรู้แนวคิด **Event + Delegate**
5. **COM Interop** ยังทำได้ แต่ควรจำกัด/ห่อด้วยชั้น adapter เพื่อความปลอดภัยในระยะยาว

---

3) .NET Framework, .NET Core, .NET 5+ (เลือกเส้นทางให้ถูกตั้งแต่วันแรก)

- **.NET Framework 4.x (Windows-only)**: เหมาะกับระบบเดิมที่พึ่งพา Windows/COM หนัก ๆ
- **.NET (5/6/7/8/9…)**: สายหลักปัจจุบัน (ขอเรียกสั้น ๆ ว่า ".NET")
    - **Cross-platform** (แต่ WinForms/WPF ยัง Windows-only)
    - โปรเจกต์แบบ **SDK-style**, ใช้ CLI dotnet ได้

- **TFM (Target Framework Moniker)** เช่น net8.0, net8.0-windows (สำหรับ WinForms/WPF)
- **VB.NET บน .NET**: รองรับ **Console, Class Library, WinForms/WPF** อย่างเป็นทางการ
- **ASP.NET Core**: ไม่มีเทมเพลต VB (ปฏิบัติจริงมักใช้ C# เป็น Host + อ้างอิง Library VB ได้)

คำแนะนำการเลือก:

- สร้างของใหม่ ⟶ เลือก **.NET เวอร์ชันล่าสุดที่เป็น LTS** (เช่น .NET 8 LTS)
- ต้อง UI เดสก์ท็อป ⟶ **net8.0-windows** (หรือเวอร์ชัน LTS ปัจจุบัน)
- ระบบเดิมบน .NET Framework ⟶ วางแผนย้ายแบบเป็นเฟส: แยก **Business Logic** ⟶ **Class Library (netstandard/netX)** เพื่อรีไซเคิลในอนาคต

---

4) เครื่องมือพัฒนา: Visual Studio vs Visual Studio Code

A) Visual Studio (แนะนำสุดสำหรับ VB.NET)

- ครบทั้ง **Form Designer**, WPF Designer, Debugger, Profiler, Test Explorer
- ขั้นตอนติดตั้ง (Community Edition ก็พอสำหรับเริ่มต้น):
  1. ดาวน์โหลด Visual Studio ⟶ ติดตั้ง **Workload: .NET desktop development**
  2. ถ้าจะทำฐานข้อมูล/SSDT ให้เพิ่ม **Data storage and processing**
  3. เปิด **Tools** ⟶ **Options** ⟶ **VB Defaults** แล้วตั้งค่า:
     - Option Strict = **On**
     - Option Explicit = **On**
     - Option Infer = **On**
- ส่วนเสริมที่เป็นประโยชน์: Productivity Power Tools, .NET Analyzers

B) Visual Studio Code

- เบา คล่องตัว แต่ **ไม่มี WinForms/WPF Designer** สำหรับ VB
- ใช้ดีเมื่อทำ **Console/Library** + เน้น CLI
- ต้องติดตั้ง **.NET SDK** และส่วนขยายพื้นฐาน (แม้ชื่อจะเน้น C# แต่ช่วยเรื่อง Debug/Test/OmniSharp)

ร่วมกันต้องมี:

- **.NET SDK** เวอร์ชันล่าสุด (จะได้ dotnet CLI มาด้วย)

---

5) สร้างโปรเจกต์แรก (Console) + เข้าใจโครงสร้างโปรเจกต์เชิงลึก

วิธีที่ 1: Visual Studio

1. **Create a new project** ⟶ เลือก **Console App (Visual Basic)**
2. ตั้งชื่อเช่น HelloVbNet ⟶ Create
3. เปิดไฟล์ Module1.vb (หรือ Program.vb) แล้วใส่โค้ด:

```
Imports System
Imports System.Text


Module Program
    Sub Main(args As String())
        ' ให้คอนโซลรองรับ UTF-8 (เหมาะกับข้อความภาษาไทย)
        Console.OutputEncoding = Encoding.UTF8
        Console.InputEncoding = Encoding.UTF8


        Console.WriteLine("Hello, VB.NET on .NET!")
        Console.Write("พิมพ์ชื่อของคุณ: ")
        Dim name As String = Console.ReadLine()


        ' String interpolation (VB 14+): ใช้ $"..."
        Console.WriteLine($"สวัสดี {name}, ยินดีต้อนรับสู่ VB.NET")
        Console.WriteLine($"เวลาปัจจุบัน: {DateTime.Now}")
        Console.WriteLine("กด Enter เพื่อออก...")
        Console.ReadLine()
    End Sub
End Module
```

4. กด **F5** (Debug) หรือ **Ctrl+F5** (Run)

โครงสร้างโปรเจกต์ (สำคัญต่อการดูแลระยะยาว)

```
HelloVbNet/
  ├── HelloVbNet.vbproj      ' SDK-style project
  ├── Program.vb (หรือ Module1.vb)
  └── My Project/            ' การตั้งค่าบางอย่าง (บางเทมเพลต)
```

ตัวอย่าง .vbproj ที่ “ดีตั้งแต่ต้น” (ปรับได้เอง):

```
<Project Sdk="Microsoft.NET.Sdk">
  <PropertyGroup>
    <OutputType>Exe</OutputType>
    <TargetFramework>net8.0</TargetFramework>
```

```
    <RootNamespace>HelloVbNet</RootNamespace>

    <!-- ช่วยให้โค้ดปลอดภัย/ชัดเจน -->
    <OptionStrict>On</OptionStrict>
    <OptionExplicit>On</OptionExplicit>
    <OptionInfer>On</OptionInfer>

    <!-- เปิดคำเตือนให้เข้มงวดขึ้น ถ้าต้องการคุณภาพโค้ด -->
    <WarningsAsErrors>NU1605;BC42024</WarningsAsErrors>
    <TreatWarningsAsErrors>false</TreatWarningsAsErrors>
  </PropertyGroup>
</Project>
```

หมายเหตุ: VB.NET **ไม่มี "top-level statements"** แบบ C#—จุดเริ่มต้นยังคงเป็น Sub Main ใน Module หรือใช้ Application Framework ของ WinForms

วิธีที่ 2: dotnet CLI (Windows/macOS/Linux)

```
dotnet new console -lang VB -n HelloVbNet
cd HelloVbNet
dotnet run
```

- แก้ Program.vb ด้วยโค้ดด้านบน แล้ว dotnet run อีกครั้ง
- ใช้ dotnet build เพื่อคอมไพล์, dotnet publish เพื่อแพ็กไปปล่อยใช้งาน
    - ตัวอย่าง publish แบบ **self-contained** (ไม่ต้องติดตั้ง .NET Runtime ที่เครื่องปลายทาง):
    - dotnet publish -c Release -r win-x64 --self-contained true /p:PublishSingleFile=true

---

เคล็ดลับคุณภาพ/ประสิทธิภาพตั้งแต่วันแรก

- **เปิด Option Strict/Explicit/Infer = On** ⟶ ลด late binding, เพิ่มความเร็ว & ความชัดเจน
- ตั้ง **Console.Input/OutputEncoding = UTF8** เมื่อมีภาษาไทย
- แยก **ชั้น UI/Logic** แม้เป็นโปรเจกต์เล็ก ๆ (เช่น Program.vb เรียกฟังก์ชันจาก Services/Greeter.vb) เพื่อฝึกโครงสร้างที่ทดสอบง่าย
- ใช้ **NuGet** ตั้งแต่ต้น (เช่น Serilog สำหรับล็อก, System.Text.Json สำหรับ JSON, Dapper สำหรับ data access เบื้องต้น)
- ใช้ **Analyzers** และ "Warnings as Errors" เฉพาะบางรหัสที่อยากเข้มงวด เพื่อกันหนี้เทคนิค

สรุปใจความสำคัญ

- VB.NET รันบน **CLR**: ได้ GC, IL, JIT, Type Safety, ไลบรารีร่วมกับ C#
- ต่างจาก VB6 อย่างมีนัยสำคัญ: **Strong typing, OOP เต็มรูปแบบ, Exception-based error handling, 0-based array**
- ยุคปัจจุบันให้เลือก **.NET (LTS)**, โดย VB.NET เหมาะกับ **Console/Library/WinForms/WPF (Windows)**
- เครื่องมือแนะนำ: **Visual Studio + .NET SDK**, เปิด Option Strict/Explicit/Infer
- เริ่มด้วย Console App ⟶ เข้าใจโครงสร้างโปรเจกต์/TFM/CLI ⟶ ปูพื้นสู่บทต่อไป

ประวัติและวิวัฒนาการของ VB.NET

ประวัติและวิวัฒนาการของ VB.NET ผมจะอธิบายเชิงลึก ครอบคลุมตั้งแต่ต้นกำเนิด Visual Basic จนถึงยุค .NET 5+

☐ **ประวัติและวิวัฒนาการของ VB.NET**

1. จุดเริ่มต้นของ Visual Basic (ก่อน VB.NET)

- **Visual Basic (VB)** กำเนิดครั้งแรกในปี **1991** โดย Microsoft
- ออกแบบมาเพื่อให้ง่ายต่อการพัฒนาแอปพลิเคชัน **Windows Desktop**
- VB ใช้แนวคิด **RAD (Rapid Application Development)** ⟶ พัฒนาโปรแกรมได้เร็วมาก เพราะมี **GUI Designer** (ลากปุ่ม, วาง Label, สร้าง Form)
- **จุดแข็งของ VB ดั้งเดิม (VB1 – VB6)**
  - เข้าใจง่าย แม้ผู้เริ่มต้นเขียนโค้ดครั้งแรกก็เรียนรู้ได้เร็ว
  - ทำงานร่วมกับ Windows API และ COM ได้ง่าย
  - ได้รับความนิยมสูงสุดในยุค 1990s – early 2000s

☐ **VB6 (1998)** คือเวอร์ชันสุดท้ายของ Visual Basic แบบคลาสสิก ก่อนเข้าสู่ยุค .NET

2. การเกิดของ VB.NET (2002)

- Microsoft เปิดตัว **.NET Framework 1.0 (2002)**
- VB ได้ถูกออกแบบใหม่ทั้งหมด กลายเป็น **VB.NET**
- เปลี่ยนแปลงสำคัญ:
  - ไม่รองรับ **Backward Compatibility** กับ VB6 โดยตรง
  - กลายเป็น **ภาษาเชิงวัตถุเต็มรูปแบบ (Fully Object-Oriented Language)**

- ใช้ **CLR (Common Language Runtime)** ⟶ ทำให้ทำงานร่วมกับภาษาอื่นใน .NET (C#, F#, C++/CLI) ได้
- สนับสนุนคุณสมบัติใหม่ เช่น:
  - Inheritance (การสืบทอดคลาส)
  - Exception Handling (Try…Catch)
  - Garbage Collection (จัดการหน่วยความจำอัตโนมัติ)
  - Asynchronous Programming (ภายหลังเพิ่มใน VB.NET 2012+)

---

3. วิวัฒนาการของ VB.NET แต่ละเวอร์ชันหลัก

☐ ***VB.NET 2002 – 2003***

- เปิดตัวพร้อม **.NET Framework 1.0 – 1.1**
- สนับสนุน OOP อย่างเต็มรูปแบบ
- ยังขาดคุณสมบัติหลายอย่าง เช่น Generics

☐ ***VB.NET 2005***

- มาพร้อม **.NET Framework 2.0**
- เพิ่ม **Generics**, Partial Classes, Nullable Types
- รองรับ Windows Forms และ Web Forms

☐ ***VB.NET 2008***

- มาพร้อม **.NET Framework 3.5**
- รองรับ **LINQ (Language Integrated Query)** และ **Lambda Expressions**

☐ ***VB.NET 2010***

- มาพร้อม **.NET Framework 4.0**
- เพิ่ม **Auto-Implemented Properties, Collection Initializers**
- สนับสนุน **Parallel Programming (Task Parallel Library)**

☐ ***VB.NET 2012 – 2015***

- รองรับ **Async / Await** ⟶ เขียนโปรแกรมแบบ Asynchronous ง่ายขึ้น
- รองรับ **WinRT API** สำหรับ Windows 8 / UWP

☐ ***VB.NET 2017 – 2019***

- ปรับให้ทำงานร่วมกับ **.NET Core** ได้
- Microsoft ลดการลงทุนในการพัฒนา VB.NET ต่อ แต่ยังคงรองรับ

☐ ***ปัจจุบัน (VB.NET บน .NET 5/6/7/8)***

- สามารถทำงานบน **.NET 5+ (Unified Platform)**

- VB.NET ยังอยู่ แต่ Microsoft **ประกาศว่าเน้น Maintenance Mode** ⟶ ไม่เพิ่มฟีเจอร์ใหม่แล้ว
- ใช้ได้กับ **Windows Desktop Apps (WinForms, WPF)** เป็นหลัก
- ในงาน **Cross-platform (Linux, macOS, Web, Mobile)** ⟶ Microsoft แนะนำให้ใช้ **C#**

---

4. เหตุผลที่ VB.NET สำคัญในวงการ

- เป็นภาษาที่ช่วยให้ผู้เริ่มต้นเขียนโค้ดง่าย แต่ก็มีพลังมากพอสำหรับงานจริง
- ถูกใช้กว้างขวางใน **Enterprise Applications** (โดยเฉพาะในองค์กรที่ใช้ Windows Server)
- ยังมี Legacy Code จำนวนมากที่เขียนด้วย VB.NET ⟶ ต้องการ Developer ที่เข้าใจ

สรุป

- **VB (Visual Basic)** ⟶ เริ่มต้นปี 1991, เน้น RAD
- **VB6 (1998)** ⟶ จุดสูงสุดของ VB แบบดั้งเดิม
- **VB.NET (2002)** ⟶ เปลี่ยนแปลงครั้งใหญ่, รองรับ OOP เต็มรูปแบบ, ทำงานบน .NET CLR
- **VB.NET + .NET 5+ (ปัจจุบัน)** ⟶ ยังใช้ได้ โดยเฉพาะ Windows Desktop Apps, แต่ Microsoft เน้นให้ใช้ C# เป็นหลักสำหรับงานใหม่

ความแตกต่างระหว่าง VB6 และ VB.NET

---

1. สถาปัตยกรรม (Architecture)

- **VB6**
  - เป็นภาษา **COM-based (Component Object Model)**
  - ทำงานบน Windows เท่านั้น
  - โปรแกรมที่เขียน VB6 จะต้องอาศัย **VB Runtime** (เช่น MSVBVM60.DLL)
- **VB.NET**
  - เป็นภาษา **Managed Code** ทำงานบน **.NET CLR (Common Language Runtime)**
  - มี **Cross-platform** ได้ (ถ้าใช้ .NET Core หรือ .NET 5+)
  - ไม่ขึ้นอยู่กับ COM โดยตรง แต่สามารถทำงานร่วมกับ COM ได้

---

2. Paradigm การเขียนโปรแกรม

- **VB6**
  - เป็น **Event-driven programming** สำหรับ Windows Form

- รองรับ OOP แบบบางส่วน (Encapsulation, Inheritance ผ่าน Interface เท่านั้น, ไม่มี Polymorphism ที่แท้จริง)

- **VB.NET**
  - เป็น **Object-Oriented Programming (OOP) เต็มรูปแบบ**
  - รองรับ Class, Inheritance, Polymorphism, Interfaces ได้ครบถ้วน
  - รองรับ Generics, LINQ, และ Asynchronous programming

---

3. Compilation และ Execution

- **VB6**
  - คอมไพล์เป็น **Native Code** หรือ **P-code (Pseudo Code)**
  - ทำงานโดยตรงบน Windows OS
- **VB.NET**
  - คอมไพล์เป็น **Intermediate Language (IL)**
  - CLR แปลง IL เป็น Native Code ตอน Runtime (JIT – Just-In-Time compilation)
  - ทำให้ Cross-language Integration เป็นไปได้ (เช่น เขียน VB.NET ร่วมกับ C#, F# ได้)

---

4. GUI และ Forms

- **VB6**
  - ใช้ **Windows Forms แบบดั้งเดิม** ที่ผูกติดกับ Win32 API
  - ออกแบบ GUI ได้ง่าย แต่จำกัดเฉพาะ Windows
- **VB.NET**
  - รองรับ **Windows Forms, WPF (Windows Presentation Foundation), UWP (Universal Windows Platform)**
  - GUI มีความยืดหยุ่นมากกว่า และรองรับ XAML

---

5. Type System และการจัดการ Memory

- **VB6**
  - **Weakly typed**: สามารถใช้ Variant type ที่ไม่แน่นอนได้
  - ไม่มี **Garbage Collection** (นักพัฒนาต้องจัดการ memory และ object references เอง)
- **VB.NET**
  - **Strongly typed**: มี type-checking ที่ชัดเจน

- CLR จัดการ **Garbage Collection (GC)** ให้อัตโนมัติ
- ปลอดภัยกว่าในแง่ของ memory leaks และ dangling pointers

---

6. การเชื่อมต่อฐานข้อมูล

- **VB6**
  - ใช้ **ADO (ActiveX Data Objects)**
  - มีข้อจำกัดเรื่องความยืดหยุ่นและ performance
- **VB.NET**
  - ใช้ **ADO.NET**, Entity Framework, LINQ to SQL
  - จัดการการเชื่อมต่อข้อมูลได้ทันสมัยและรองรับหลายรูปแบบ (SQL Server, MySQL, Oracle, etc.)

---

7. ความสามารถใหม่ใน VB.NET ที่ไม่มีใน VB6

- Exception Handling ด้วย Try...Catch...Finally
- Inheritance (class-based)
- Generics
- Attributes & Reflection
- Asynchronous Programming (Async / Await)
- LINQ (Language Integrated Query)
- Multi-threading ที่ง่ายขึ้น

---

8. การ Support

- **VB6**: Microsoft หยุดสนับสนุนแล้ว (ตั้งแต่ปี 2008) แต่ยังมี Runtime รองรับบน Windows
- **VB.NET**: ยังมีการสนับสนุนใน **.NET Framework และ .NET 5+**, แต่ Microsoft ประกาศว่า VB.NET จะไม่พัฒนา feature ใหม่มากนัก (focus อยู่ที่ C#)

☐ **สรุปสั้น ๆ**

- **VB6** = ง่าย, เร็วสำหรับ Windows Form เล็กๆ, แต่จำกัดมาก
- **VB.NET** = เป็นภาษา OOP เต็มรูปแบบ, ทำงานบน .NET CLR, ทันสมัยและยืดหยุ่นกว่า

NET Framework, .NET Core และ .NET 5+

NET Framework, .NET Core และ .NET 5+ เป็นหัวใจสำคัญในการเข้าใจ **VB.NET** เพราะภาษา VB.NET ทำงานอยู่บน **.NET Platform** ซึ่งมีวิวัฒนาการอย่างต่อเนื่องตั้งแต่ยุคแรกจนถึงปัจจุบัน

☐ **.NET Framework**

- **เปิดตัวปี 2002** พร้อมกับ VB.NET และ C#
- เป็น **Windows-only platform**
- ใช้ **CLR (Common Language Runtime)** สำหรับจัดการ memory, garbage collection, JIT compilation
- มี **Base Class Library (BCL)** และ **Framework Class Library (FCL)** ที่รวมคลาสและ API สำหรับงานต่างๆ เช่น IO, Collections, XML, ADO.NET
- เหมาะกับการพัฒนา **Desktop Application (Windows Forms, WPF)** และ **ASP.NET Web Forms/MVC**
- **ข้อจำกัด**:
    - รันได้เฉพาะ Windows
    - การอัปเดตขึ้นกับ Windows Update
    - ไม่ยืดหยุ่นสำหรับ cross-platform

☐ **.NET Core**

- เปิดตัวปี 2016 (Microsoft Build)
- เป็น **cross-platform** ⟶ ทำงานได้ทั้ง **Windows, Linux, macOS**
- ออกแบบให้เป็น **Open Source** บน GitHub ⟶ community มีส่วนร่วม
- ใช้ **CoreCLR** ซึ่งเบากว่า CLR ของ .NET Framework
- มี **CoreFX** library ที่แยกส่วนเล็ก กระทัดรัด และโหลดเฉพาะที่จำเป็น
- รองรับ **modern development** เช่น microservices, Docker, Kubernetes
- รองรับ **VB.NET, C#, F#** (แต่ VB.NET ถูกลดการสนับสนุนในบางฟีเจอร์ใหม่)
- **ข้อดี**:
    - เร็วกว่า .NET Framework
    - Deploy แบบ self-contained ได้ (ไม่ต้องติดตั้ง runtime แยก)
    - อัปเดต release cycle เร็วกว่า

☐ **.NET 5+ (Unified Platform)**

- เปิดตัว **พฤศจิกายน 2020**
- Microsoft รวม **.NET Framework + .NET Core + Xamarin + Mono** ⟶ เป็น **.NET Unified Platform**
- ตั้งแต่ .NET 5 เป็นต้นไปจะใช้ชื่อเพียง **“.NET”** ไม่ใช้ “Core” อีก

- รองรับทุก platform: **Web, Desktop, Mobile, Cloud, IoT, Gaming**
- ใช้ **Single BCL (Base Class Library)** ที่統一กันหมด
- ใช้ **RyuJIT compiler** ที่ปรับปรุง performance อย่างมาก
- **Roadmap**:
  - .NET 5 ⟶ รุ่น transition
  - .NET 6 (LTS) ⟶ ปลายปี 2021
  - .NET 7, 8, 9 … ⟶ ออกทุกปี (รุ่นคู่เป็น **LTS**)
- **VB.NET** บน .NET 5+:
  - Microsoft ประกาศว่า **VB.NET จะยังคงอยู่ แต่ไม่พัฒนา feature ใหม่**
  - เหมาะสำหรับการ **maintenance legacy project**
  - ฟีเจอร์ใหม่ส่วนใหญ่จะไปที่ **C#**

☐ **ตารางสรุปเปรียบเทียบ**

คุณสมบัติ	.NET Framework	.NET Core	.NET 5+
ปีเปิดตัว	2002	2016	2020
ระบบปฏิบัติการรองรับ	Windows เท่านั้น	Windows, Linux, macOS	Cross-platform (Unified)
Open Source	☐	☐	☐
Performance	ปานกลาง	สูง	สูงมาก
Deployment	Global install	Self-contained / Portable	Self-contained / Portable
สนับสนุน VB.NET	☐ เต็มรูปแบบ	☐ แต่จำกัด	☐ (เฉพาะ legacy)
สถานะ	Legacy	Active แต่ถูกรวม	Unified main platform

☐ **บทสรุป**

- ถ้าเป็น **โปรเจ็กต์เก่า (Legacy Windows App)** ⟶ ใช้ **.NET Framework + VB.NET**
- ถ้าเป็น **โปรเจ็กต์ใหม่ Cross-Platform** ⟶ ควรใช้ **.NET 6/7/8** (VB.NET ยังใช้ได้ แต่ไม่แนะนำสำหรับงานใหม่ ⟶ ควรใช้ C#)
- อนาคตของ **VB.NET**:
  - Microsoft จะยัง **ซัพพอร์ต** แต่เน้น **stability** ไม่เน้น **feature ใหม่**
  - เหมาะกับองค์กรที่มีโค้ดเดิม (legacy system)
  - นักพัฒนาที่เริ่มต้นใหม่ควรเรียน **C#** เป็นหลัก

การติดตั้ง Visual Studio / Visual Studio Code สำหรับ VB.NET

การติดตั้ง Visual Studio / Visual Studio Code สำหรับ VB.NET เป็นสิ่งสำคัญมาก เพราะเครื่องมือพัฒนา (IDE) จะกำหนด workflow และ productivity ตั้งแต่แรก

☐ **1. Visual Studio (IDE แนะนำสำหรับ VB.NET)**

Visual Studio เป็น IDE ที่ครบเครื่องที่สุดสำหรับ **VB.NET** ทั้ง Console, Windows Forms, WPF และ Class Library

---

ขั้นตอนติดตั้ง Visual Studio

A) ดาวน์โหลด

1. เข้าเว็บไซต์: Visual Studio Download
2. เลือก **Community Edition** (ฟรีสำหรับนักเรียน/นักพัฒนาอิสระ) หรือ Professional/Enterprise

---

B) ติดตั้ง Workload

หลังจากเปิดตัวติดตั้ง:

1. เลือก **Workloads** ⟶ สำหรับ VB.NET เลือก
   - **.NET desktop development** ⟶ สำหรับ Windows Forms, WPF, Console, Class Library
2. ตัวเลือกเสริม (Optional)
   - **Universal Windows Platform development** ⟶ ถ้าต้องการ UWP
   - **ASP.NET and web development** ⟶ ถ้าต้องการเว็บ (แต่ VB.NET ไม่ค่อยใช้)
   - **Data storage and processing** ⟶ สำหรับฐานข้อมูล / SQL Server

---

C) ตัวเลือกสำคัญหลังติดตั้ง

- **Tools ⟶ Options ⟶ VB Defaults**
  - Option Strict = **On** ⟶ ป้องกัน late binding
  - Option Explicit = **On** ⟶ บังคับประกาศตัวแปร
  - Option Infer = **On** ⟶ ช่วยให้ type inference ชัดเจน
- **Extensions**
  - Productivity Power Tools, Code analyzers, NuGet package manager

---

D) สร้างโปรเจ็กต์ VB.NET

1. File ⟶ New ⟶ Project
2. เลือก **Console App (.NET)** หรือ **Windows Forms App (.NET)**
3. ตั้งชื่อโปรเจ็กต์ ⟶ สร้างไฟล์ Program.vb หรือ Form1.vb

☐ **2. Visual Studio Code (เบา, CLI-based)**

VS Code เป็น **editor เบา ๆ** ที่รองรับหลายภาษาและ platform cross-platform

- **เหมาะสำหรับ:**
  - Console Application
  - Class Library
  - การใช้ **dotnet CLI**

ขั้นตอนติดตั้ง VS Code + VB.NET

A) ดาวน์โหลด

- VS Code Download ⟶ ติดตั้งตาม OS

B) ติดตั้ง .NET SDK

- ดาวน์โหลด .NET SDK ล่าสุด: dotnet.microsoft.com/download
- ตรวจสอบเวอร์ชัน:
- dotnet --version

C) Extension สำคัญ

- **C# extension (OmniSharp)** ⟶ สนับสนุน IntelliSense, Debug
- **VB.NET** ⟶ ใช้ได้กับ C# extension เพราะใช้ **.NET SDK** ร่วมกัน

D) การสร้างโปรเจ็กต์

สร้างโปรเจ็กต์ Console VB.NET

dotnet new console -lang VB -n HelloVbNet

cd HelloVbNet

dotnet run

- แก้ไข Program.vb ผ่าน VS Code
- ใช้ dotnet build เพื่อคอมไพล์
- ใช้ dotnet publish เพื่อเผยแพร่ (publish)

☐ **เปรียบเทียบ Visual Studio กับ VS Code**

ฟีเจอร์	Visual Studio	Visual Studio Code
Debugger	ครบเครื่อง	ต้องติดตั้ง extension

ฟีเจอร์	Visual Studio	Visual Studio Code
Designer (WinForms/WPF)	☐ GUI Designer	☐ ไม่มี GUI Designer
IntelliSense	☐ สมบูรณ์	☐ ต้องผ่าน extension
Project Templates	☐ ครบทุกชนิด	☐ แต่จำกัด
Cross-platform	☐ Windows เท่านั้น	☐ Windows/Linux/macOS
CLI	ใช้ได้	☐ ใช้ dotnet CLI เต็มรูปแบบ

☐ **สรุปแนวทางการติดตั้ง**

- **แนะนำเริ่มต้น**: Visual Studio Community Edition
  - เหมาะสำหรับเรียนรู้และพัฒนา VB.NET Desktop / Console แบบเต็มรูปแบบ
- **ถ้าเน้นเบา, Cross-platform, CLI**: VS Code + .NET SDK
- **เปิด Option Strict/Explicit/Infer** ตั้งแต่เริ่ม ⟶ ลด bug, เพิ่มประสิทธิภาพ

การสร้างโปรเจ็กต์แรก (Console Application) ด้วย VB.NET

☐ **1. สร้างโปรเจ็กต์ Console Application บน Visual Studio**

A) ขั้นตอน

1. เปิด **Visual Studio** ⟶ File ⟶ New ⟶ Project
2. เลือก **Console App (.NET)** ⟶ ภาษา **Visual Basic**
3. ตั้งชื่อโปรเจ็กต์ เช่น HelloVBNet
4. เลือกโฟลเดอร์เก็บโปรเจ็กต์ ⟶ กด **Create**

B) โครงสร้างโปรเจ็กต์ (พื้นฐาน)

เมื่อสร้างเสร็จ จะได้ไฟล์และโฟลเดอร์ประมาณนี้:

```
HelloVBNet/
├── HelloVBNet.vbproj      # ไฟล์ project ของ VB.NET
├── Program.vb             # ไฟล์หลักของโปรแกรม
└── bin/obj                # Output / build files
```

C) ตัวอย่างโค้ด Program.vb แบบพื้นฐาน

```
Module Program

    Sub Main()
```