

Starting with VB.NET

Language fundamentals, Windows Forms and legacy MVC awareness

FA-0698 · 2 days

Software Development

Introduction

This two-day course, also deliverable as four half-day sessions, introduces VB.NET through short coding exercises and a small Windows Forms application. It covers Visual Studio, language fundamentals, object-oriented design, exceptions, collections, LINQ and file operations.

A prepared demonstration explains classic ASP.NET MVC on .NET Framework and VB Razor views for legacy-system understanding. This is not presented as a modern ASP.NET Core VB web-development path; the practical course outcome is a bounded introductory application.

Learning outcomes

- Create, run and debug simple VB.NET projects in Visual Studio.
- Use variables, control flow, procedures, modules and basic classes.
- Apply selected inheritance, interface and encapsulation concepts.
- Use exceptions, generic collections, LINQ and text-file operations in small examples.
- Build a simple event-driven Windows Forms data-entry application.
- Explain classic ASP.NET MVC structure and VB legacy-web maintenance considerations.

Prerequisites

- Basic understanding of programming logic (variables, conditionals, loops) in any language.
- Familiarity with using a Windows PC and installing/running software.
- Visual Studio (Community Edition or higher) installed on the machine for hands-on exercises.
- (Optional but helpful) Basic familiarity with C# or another object-oriented language to ease transition.

Use appropriate Visual Studio workloads and SDKs; the optional classic MVC demonstration needs a prepared .NET Framework environment.

Training outline

1. Day 1 — Environment and syntax

- Overview of VB.NET in the modern .NET ecosystem: what's changed, what remains.
- Installing Visual Studio, selecting VB.NET project templates.
- Exploring the IDE: Solution Explorer, code editor, debug windows, toolbox for Windows Forms.
- Creating your first "Hello World" VB.NET console application: structure of a VB.NET project, Main method, basic output, build & run.

Core language constructs

- VB.NET data types: Integer, Long, Short, String, Boolean, Date, Decimal, Double and Object; Variant is legacy VB terminology, not a VB.NET type
- Variable declaration and initialization: Dim, As, initialization rules.
Control flow statements:
 - Conditional: If...Then...Else...End If
 - Select Case ... Case ... End Select
- Loops and iteration: For ... Next, For Each ... Next, While ... End While, Do...Loop.
- Methods (Sub and Function): declaration, parameters, return values, overloading.
- Modules vs Classes: how VB.NET organizes code.
- Comments, regions, and best practices for code readability.

2. Day 1 — Objects and standard libraries

- Understanding classes and objects: definition, instantiation, namespaces.
- Properties, fields, methods, constructors.
- Encapsulation: Public, Private, Protected, Friend keywords.
- Inheritance: Inherits keyword, base class, derived class, method overriding (Overrides), base invocation (MyBase).
- Polymorphism: base class references, interface implementation.
- Interfaces: Implements keyword, multiple interfaces, when to use.
- Abstract concepts (MustOverride, MustInherit) where VB.NET supports them.
- Practical coding exercise: build a small class hierarchy (e.g., Shape → Circle, Rectangle) with shared behavior and polymorphic methods.

Libraries, queries and diagnostics

- Using the .NET class libraries in VB.NET: imports statements, namespaces (System, System.Collections.Generic, System.Linq).
- Exception handling: Try...Catch...Finally, Throw, custom exceptions.
- Collections and generics: List(Of T), Dictionary(Of TKey, TValue), etc.
- LINQ in VB.NET: query syntax and method syntax, filtering, projection, ordering, grouping. VB.NET specific LINQ support (e.g., Visual Basic query keywords)
- Working with files and I/O: reading/writing text files using System.IO.
- Debugging and error handling within Visual Studio: breakpoints, watches, exception settings.

3. Day 2 – Windows Forms application

- Introduction to Windows Forms: what it is, when to use it.
- Creating a Windows Forms project in Visual Studio (VB.NET).
- Form design: toolbox controls (Button, TextBox, Label, ListBox, DataGridView), layout, properties window.
- Event-driven programming: Click, Load and TextChanged events and the Handles keyword
- Designing a simple Form application: e.g., a basic data-entry form that stores input into a List(Of T) and displays it in a DataGridView.
- Validating input, handling exceptions, showing messages (MessageBox).
- Packaging and deployment basics: building .exe, setting application icon, configurations.

4. Day 2 – Classic MVC demonstration and platform choices

Review a prepared classic ASP.NET MVC application on .NET Framework, not ASP.NET Core.

- Overview of web development in VB.NET: where web fits, contrast to desktop.
- Introduction to ASP.NET MVC architecture: Models, Views, Controllers, routing.
- Inspect a prepared classic ASP.NET MVC VB project on .NET Framework
- Folder structure: Controllers folder, Models folder, Views folder, Shared folder.
- Creating a simple model class in VB.NET (e.g., Product with properties ID, Name, Price).
- Creating a controller in VB.NET: actions (Index, Create, Edit, Delete).
- Legacy VB Razor views (.vbhtml), HTML helpers and output encoding
- Data binding, form submission, model validation (DataAnnotations).
- Basic persistence: using in-memory collection for demo (full database integration beyond course scope).
- Demo: build a simple CRUD web application for the Product model.

- Discussion: when to choose web vs. desktop, and limitations/considerations (maintenance, hosting, platform support).

Compare desktop and legacy-web maintenance needs; review the small desktop application and identify further practice.