

- Online Training
- C#.NET Online Training Program
 - ASP.NET Core Training
 - Microservices Online Training using .NET Core
 - Microsoft Azure Training

- Introduction & Environment Setup
- How Computer Works
 - Introduction to Programming Languages
 - How Computer Programs Works
 - Different Types of Applications
 - Programming Methodologies
 - Algorithm, Pseudocode, Programs, and Flowcharts
 - Introduction to .NET Framework
 - .NET Framework Architecture and Components
 - Introduction to C# Programming Language
 - How to Download and Install Visual Studio on Windows
 - Creating First Console Application using Visual Studio
 - .NET Developer Roadmap for 2025
 - Coding Standard Best Practices

- C# .NET Basics
- Basic Structure of C# Program
 - Methods and Properties of Console Class in C#
 - Data Types in C#
 - Literals in C#
 - Type Casting in C#
 - Variables in C#
 - Operators in C#
 - Control Flow Statements in C#
 - If-Else Statements in C#
 - Switch Statements in C#
 - Loops in C#
 - While Loop in C#
 - Do While Loop in C#
 - For Loop in C#
 - Break Statement in C#
 - Continue Statement in C#
 - Goto Statement in C#
 - Functions in C#
 - User-Defined Functions in C#
 - Call By Value and Call By Reference in C#
 - Recursion in C#
 - User Input and Output in C#
 - Command Line Arguments in C#
 - String in C#
 - Static Keyword in C#
 - Static vs Non-Static Members in C#
 - Const and Read-Only in C#
 - Properties in C#
 - Why we Should Override ToString Method in C#
 - Override Equals Method in C#
 - Difference Between Convert.ToString and ToString Method in C#
 - Checked and Unchecked Key words in C#
 - Stack and Heap Memory in .NET
 - Boxing and Unboxing in C#

- OOPs in C#
- Object Oriented Programming (OOPs) in C#
 - Class and Objects in C#
 - Constructors in C#
 - Types of Constructors in C#
 - Why We Need Constructors in C#
 - Static vs Non-Static Constructors in C#
 - Private Constructors in C#
 - Destructors in C#
 - Garbage Collection in .NET Framework
 - Differences Between Finalize and Dispose in C#
 - Access Specifiers in C#
 - Encapsulation in C#
 - Abstraction in C#
 - Inheritance in C#
 - Types of Inheritance in C#
 - How to use Inheritance in Application Development
 - IsA and HasA Relationship in C#
 - Generalization and Specialization in C#
 - Abstract Class and Abstract Methods in C#
 - Abstract Class and Abstract Methods Interview Questions in C#
 - How to Use Abstract Classes and Methods in C#
 - Application
 - Interface in C#
 - Interface Interview Questions and Answers in C#
 - Interface Realtime Examples in C#
 - Multiple Inheritance in C#
 - Multiple Inheritance Realtime Example in C#
 - Polymorphism in C#
 - Method Overloading in C#
 - Operator Overloading in C#
 - Method Overriding in C#
 - Method Hiding in C#
 - Partial Class and Partial Methods in C#
 - Sealed Class and Sealed Methods in C#
 - Extension Methods in C#
 - Static Class in C#
 - Variable Reference and Instance of a Class in C#

- OOPs Real-Time Examples
- Real-Time Examples of Encapsulation Principle in C#
 - Real-Time Examples of Abstraction Principle in C#
 - Real-Time Examples of Inheritance Principle in C#
 - Real-Time Examples of Polymorphism Principle in C#
 - Real-Time Examples of Interface in C#
 - Real-Time Examples of Abstract Class in C#

- Exception Handling
- Exception Handling in C#
 - Multiple Catch Blocks in C#
 - Finally Block in C#
 - How to Create Custom Exceptions in C#
 - Inner Exception in C#
 - Exception Handling Abuse in C#

- Events, Delegates and Lambda Expression in C#
- Course Structure of Events, Delegates and Lambda Expression
 - Roles of Events, Delegates and Event Handler in C#
 - Delegates in C#
 - Multicast Delegates in C#
 - Delegates Real-Time Example in C#
 - Generic Delegates in C#
 - Anonymous Method in C#
 - Lambda Expressions in C#
 - Events in C# with Examples

- Multi-Threading
- Multithreading in C#
 - Thread class in C#
 - How to Pass Data to Thread Function in Type Safe Manner in C#
 - How to Retrieve Data from a Thread Function in C#
 - Join Method and Joinive Property of Thread Class in C#
 - Thread Synchronization in C#
 - Lock in C#
 - Monitor Class in C#
 - Mutex Class in C#
 - SemaphoreSlim Class in C#
 - Deadlock in C#
 - Performance Testing of a Multithreaded Application
 - Thread Pool in C#
 - Foreground and Background Threads in C#
 - AutoResetEvent and ManualResetEvent in C#
 - Thread Life Cycle in C#
 - Threads Priorities in C#
 - How to Terminate a Thread in C#
 - Inter Thread Communication in C#
 - How to Debug a Multi-threaded Application in C#

- Collections in C#
- Arrays in C#
 - 2D Arrays in C#
 - Advantages and Disadvantages of Arrays in C#
 - Collections in C#
 - ArrayList in C#
 - Hashtable in C#
 - Non-Generic Stack in C#
 - Non-Generic Queue in C#
 - Non-Generic SortedList in C#
 - Advantages and Disadvantages of Non-Generic Collection in C#
 - Generic Collections in C#
 - Generics in C#
 - Generic Constraints in C#
 - Generic List Collection in C#
 - How to Sort a List of Complex Type in C#
 - Comparison Delegate in C#
 - Dictionary Collection Class in C#
 - Conversion Between Array List and Dictionary in C#
 - List vs Dictionary in C#
 - Generic Stack Collection Class in C#
 - Generic Queue Collection Class in C#
 - Foreach Loop in C#
 - Generic HashSet Collection Class in C#
 - Generic SortedList Collection Class in C#
 - Generic SortedSet Collection Class in C#
 - Generic SortedDictionary Collection Class in C#
 - Generic LinkedList Collection Class in C#
 - Concurrent Collection in C#
 - ConcurrentDictionary Collection Class in C#
 - ConcurrentQueue Collection Class in C#
 - ConcurrentStack Collection Class in C#
 - ConcurrentBag Collection Class in C#
 - BlockingCollection in C#

- File Handling
- File Handling in C#
 - FileStream Class in C#
 - StreamReader and StreamWriter in C#
 - File Class in C#
 - TextWriter and TextReader in C#
 - BinaryWriter and BinaryReader in C#
 - StringWriter and StringReader in C#
 - FileInfo Class in C#
 - DirectoryInfo Class in C#
 - Export and Import Excel Data in C#

- Asynchronous Programming
- Introduction to Concurrency
 - Async and Await in C#
 - Task in C#
 - How to Return a Value from Task in C#
 - How to Execute Multiple Tasks in C#
 - How to Limit Number of Concurrent Tasks in C#
 - How to Cancel a Task in C# using Cancellation Token
 - How to Create Synchronous Method using Task in C#
 - Retry Pattern in C#
 - Only One Pattern in C#
 - How to Control the Result of a Task in C#
 - Task-Based Asynchronous Programming in C#
 - Chaining Tasks by Using Continuation Tasks
 - How to Attached Child Tasks to a Parent Task in C#
 - ValueTask in C#
 - How to Cancel a Non-Cancellable Task in C#
 - Asynchronous Streams in C#
 - How to Cancel Asynchronous Stream in C#

- Parallel Programming
- Task Parallel Library in C#
 - Parallel For in C#
 - Parallel Foreach Loop in C#
 - Parallel Invoke in C#
 - Maximum Degree of Parallelism in C#
 - How to Cancel Parallel Operations in C#
 - Atomic Methods Thread Safety and Race Conditions in C#
 - Interlocked vs Lock in C#
 - Parallel LINQ in C#
 - Multithreading vs Asynchronous Programming vs Parallel Programming in C#

- AutoMapper
- AutoMapper in C#
 - AutoMapper Complex Mapping in C#
 - How to Map Complex Type to Primitive Type using AutoMapper in C#
 - AutoMapper Reverse Mapping in C#
 - AutoMapper Conditional Mapping in C#
 - AutoMapper Ignore Method in C#
 - Fixed and Dynamic Values in Destination Property in AutoMapper

- Optional Parameter, Indexers and Enums
- How to make Optional Parameters in C#
 - Indexers in C#
 - Indexers Real-Time Example in C#
 - Enums in C#

- .NET Framework Architecture
- DOT NET Framework
 - Common Language Runtime in .NET Framework
 - .NET Program Execution Process
 - Intermediate Language (ILDASM & ILASM) Method in C#
 - Common Type System in .NET Framework
 - Common Language Specification in .NET Framework
 - Managed and Unmanaged Code in .NET Framework
 - Assembly DLL EXE in .NET Framework
 - App Domain in .NET Framework
 - Strong and Weak Assemblies in .NET Framework
 - How to Install an Assembly into GAC in .NET Framework
 - DLL Hell Problem and Solution in .NET Framework

- Var, Dynamic and Reflection
- Reflection in C#
 - Dynamic Type in C#
 - Var Keyword in C#
 - Var vs Dynamic in C#
 - Dynamic vs Reflection in C#
 - Volatile Keyword in C#
 - Ref vs Out in C#
 - Named Parameters in C#

- C# 7.X New Features
- C# 7 New Features
 - Enhancement in Out Variables in C# 7
 - Pattern Matching in C#
 - Digit Separators in C# 7
 - Tuples in C# 7
 - Local Functions in C# 7
 - Ref Returns and Ref Locals in C# 7
 - Generalized Async Return Types in C# 7
 - Expression Bodied Members in C#
 - Thrown Expression in C#
 - Async Main in C#

- C# 8 New Features
- C# 8 New Features
 - ReadOnly Structs in C#
 - Default Interface Methods in C#
 - Pattern Matching in C#
 - Using Declarations in C#
 - Static Local Functions in C#
 - Disposable Ref Structs in C#
 - Nullable Reference Types in C# 8
 - Asynchronous Streams in C#
 - Asynchronous Disposable in C#
 - Indices and Ranges in C#
 - Null-Coalescing Assignment Operator in C#
 - Unmanaged Constructed Types in C#
 - Stackalloc in C#

- Most Popular C# Books
- Most Recommended C# Books
 - Most Recommended Data Structure and Algorithms Books using C#

Introduction to .NET Framework

Back to: [C#.NET Tutorials For Beginners and Professionals](#)

Introduction to .NET Framework

Here, I am going to give you an overview of the DOT NET Framework. This is important for you as a dot net developer to know the history and evolution of the DOT NET Framework. Here, in this session, first, we will discuss what was there before the .NET Framework and what problems we face in that, and how we overcome all those problems in the .NET Framework. Before DOT NET Framework COM is there. So, let us first discuss what is COM and what problems we face in COM.

What is COM?

COM stands for **Component Object Model**. The COM is one of the Microsoft Frameworks. Using this Framework, we can develop **Windows Applications** (Desktop or Standalone Applications for Windows OS) as well as **Web Applications**. In earlier COM, **VB** is the programming language that is used to implement Windows applications and **ASP** is the technology used to implement web applications.

For more info: https://en.wikipedia.org/wiki/Component_Object_Model

What are the disadvantages of COM?

There are two major disadvantages of the COM Framework. They are as follows:

1. Incomplete Object-Oriented Programming means it will not support all the features of OOPs.
2. Platform Dependent means COM applications can run on only Windows OS.

To overcome the above problems, Microsoft introduces **.NET Framework**.

What does .NET Represent?

.NET stands for **Network Enabled Technology** (Internet). In .NET, dot (.) refers to **Object-Oriented**, and NET refers to the internet. So, the complete .NET means through Object-Oriented we can implement internet-based applications.

According to Microsoft, **.NET** is a **Free, Cross-Platform, Open-Source** developer platform for building many different types of applications. With .NET, we can use multiple languages (C#, VB, F#, etc.), Editors (Visual Studio, Visual Studio Code, Visual Studio for Mac, OmniSharp, JetBrains Rider, etc), and Libraries to build for Web, Mobile, Desktop, Games, IoT, and more.

Cross Platform: Whether you are working in C#, F#, or Visual Basic, your code will run on any compatible operating system. You can build many types of apps with .NET. Some are Cross-Platform, and some target a specific set of operating systems and devices.

Libraries: To extend functionality, Microsoft and others maintain a healthy .NET package ecosystem. **NuGet** is a package manager built specifically for .NET that contains over 100,000 packages.

For more info: <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet>

What is a Framework?

A framework is a software. Or you can say a framework is a collection of many small technologies integrated together to develop applications that can be executed anywhere.

What does the .NET Framework Provide?

The DOT NET Framework provides two things as follows

1. **BCL** (Base Class Libraries)
2. **CLR** (Common Language Runtime)

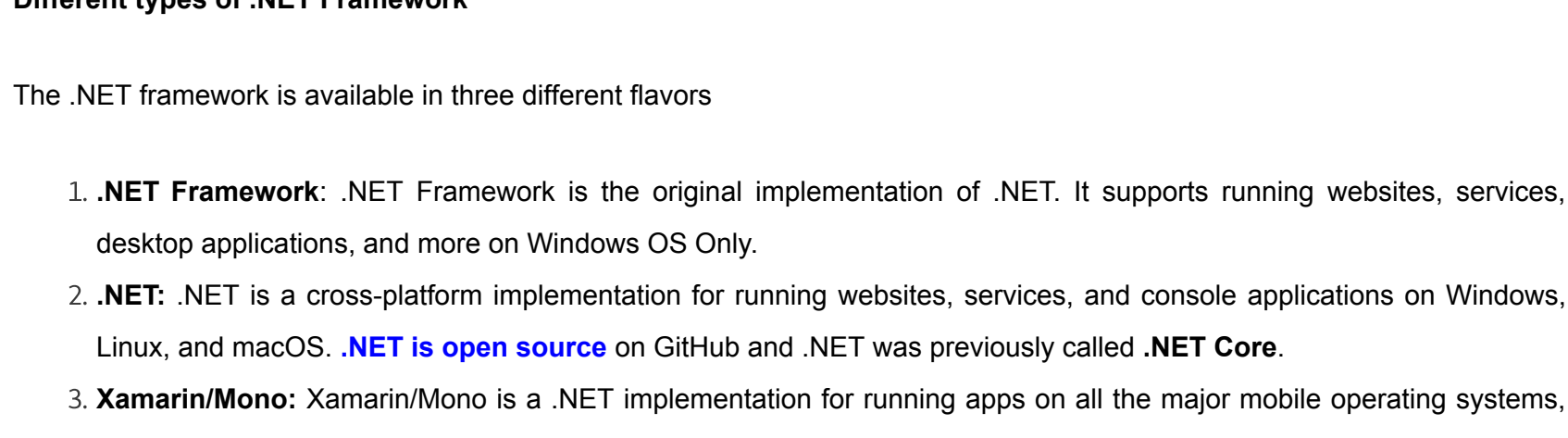
What is BCL?

Base Class Libraries (BCL) are designed by Microsoft. Without BCL we can't write any code in .NET. So, BCL is also known as the basic building block of .NET Programs. These are installed into the machine when we installed the .NET framework. BCL contains pre-defined classes and these classes are used for the purpose of application development.

The physical location of BCL is **C:\Windows\assembly**

What is CLR?

CLR stands for Common Language Runtime and it is the core component under the .NET framework which is responsible for converting the MSIL (Microsoft Intermediate Language) code into native code. In our CLR session, we will discuss **CLR** in detail.



In the **.NET** framework, the code is compiled twice.

1. In the 1st compilation, the source code is compiled by the respective language compiler and generates the intermediate code which is known as **MSIL (Microsoft Intermediate Language)** or **IL (Intermediate language code)**, or **Managed Code**.
2. In the 2nd compilation, **MSIL** is converted into **Native code** (native code means code specific to the Operating system so that the code is executed by the Operating System) and this is done by **CLR**.

Always 1st compilation is slow and 2nd compilation is fast.

What is JIT?

JIT stands for the **Just-in-Time** compiler. It is the component of **CLR** that is responsible for converting **MSIL** code into **Native Code**. Native code is code that is directly understandable by the operating system.

Different types of .NET Framework

The .NET framework is available in three different flavors

1. **.NET Framework**: .NET Framework is the original implementation of .NET. It supports running websites, services, desktop applications, and more on Windows OS Only.
2. **.NET**: .NET is a cross-platform implementation for running websites, services, and console applications on Windows, Linux, and macOS. **.NET is open source** on GitHub and .NET was previously called **.NET Core**.
3. **Xamarin/Mono**: Xamarin/Mono is a .NET implementation for running apps on all the major mobile operating systems, including iOS and Android.

Note: .NET Framework is Platform-Dependent while **.NET or .NET Core is Platform Independent**. Here, we are not talking about Web Applications. Web Applications are independent of Operating Systems.

For more info: <https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework>

What is not .NET?

1. .NET is not an Operating system.
2. .NET is not a database
3. It is not an ERP application.
4. .NET is not a Testing Tool.
5. It is not a programming language.

What is Exactly .NET?

.NET is a framework tool that supports many programming languages and many technologies. .NET support 60+ programming languages. Of 60+ programming languages, 9 are designed by Microsoft and the remaining are designed by non-Microsoft. Microsoft-designed programming languages are as follows:

1. **VB.NET**
2. **C#.NET**
3. **VC++ .NET**
4. **J#.NET**
5. **F#.NET**
6. **Jscript.NET**
7. **WindowsPowerShell**
8. **Iron python**
9. **Iron Ruby**

Technologies supported by the .NET framework are as follows

1. **ASP.NET (Active Server Pages.NET)** – MVC, Web API, Core MVC, Core Web API, Core Blazor, etc.
2. **ADO.NET** (Active Data Object.NET)
3. **VC++** (Windows Communication Foundation)
4. **WCF** (Windows Presentation Foundation)
5. **WPF** (Windows Workflow Foundation)
6. **AJAX** (Asynchronous JavaScript and XML)
7. **LINQ** (Language Integrated Query)
8. **Entity Framework**

What are a Language and its need?

1. Language acts as the mediator between the programmer and the system.
2. It offers some rules and regulations for writing the program.
3. The language also offers some libraries which are required for writing the program.

What are Technology and its Needs?

Technology is always designed for a particular purpose. For example, the development of web-related applications in .NET using the technology ASP.NET. But the technology does not offer any specific rules for writing the programs. That's why technology can't be implemented individually. VB.NET, C#.NET, and F#.NET are programming languages. Using any of these languages we can implement windows/desktop applications individually. Every language is having its own compiler

Language	File Extensions	Compiler
VB .NET	.VB	VBC(Visual Basic Compiler)
C# .NET	.CS	CSC (C Sharp Compiler)

In the next article, I am going to discuss **.NET Framework Architecture and Components**. Here, in this article, I try to explain Introduction to .NET Framework, and I hope you enjoy this Introduction to .NET Framework article.



Previous Lesson	Next Lesson
Algorithm, Pseudocode, Programs, and Flowcharts	.NET Framework Architecture and Components

1 thought on “Introduction to .NET Framework”

ALEXANDRE
JANUARY 9, 2025 AT 5:17 PM

Hello!
Thank you very much for this tutorials!
Do you have a PDF version of the C# .net lessons?

From France with Love!

Alexandre

[Reply](#)

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name*

Email*

Website

[Post Comment](#)

