Day 1 : Minutes of Session

Date: 18th July
Topic: Introduction to .NET & C# Programming

Session Agenda

1. **Introduction to .NET**
    - Evolution of Programming Languages(A, B, C) - C++ → C#(Anders Heilsberg)
    - History and Overview of .NET Framework(2002), .NET Core(2016), and .NET 5/6/7
    - Importance of .NET in Modern Software Development
2. **.NET Ecosystem Deep Dive**
    - **CLR (Common Language Runtime):** Manages code execution and memory
    - **CTS (Common Type System):** Ensures type compatibility across languages
    - **BCL (Base Class Library):** Standard APIs for I/O, data handling, and more
3. **.NET Architecture**
    - Architecture Diagram covering runtime, libraries, application models
    - Explanation of platform independence and cross-language capabilities
4. **Visual Studio (VS) Installation & Setup**
    - Step-by-step guide for installing Visual Studio Community Edition 2022
    - Required workloads: .NET Desktop Development
5. **Visual Studio IDE Walkthrough**
    - Overview of IDE features: Solution Explorer, Output Window, Debug Console

    - Usage of predefined templates for different application types

6. **C# Basics Lab Session**
    - Data Types: int, string, float, char, bool, DateTime, var
    - Console Input/Output: `Console.WriteLine()`, `Console.ReadLine()`
    - Basic operations and conversions
7. **Hello World Console App**
    - Writing and running a C# program from scratch
    - Explanation of:
        - `using System; // Standard Global Package`
        - `class Program // Blueprint or container`
        - `static void Main() // Static does not require class and void returns you nothing 9 Execution of program starts from Main)`

Hands-on Activities

- Installed and configured Visual Studio IDE
- Created and executed first C# Console Application
- Practiced with basic syntax, data types, and I/O operations

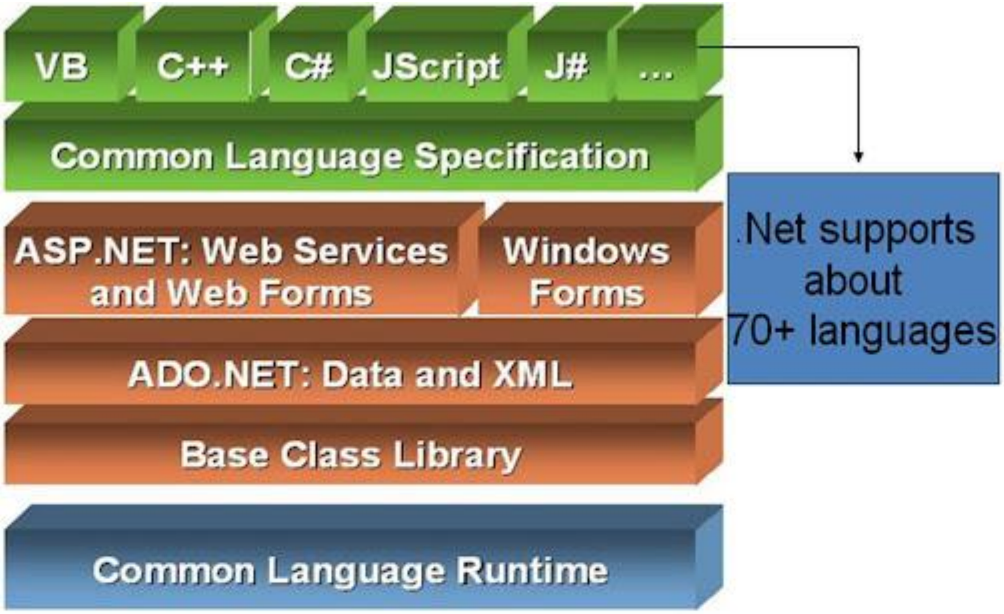
---

Key Discussion Points

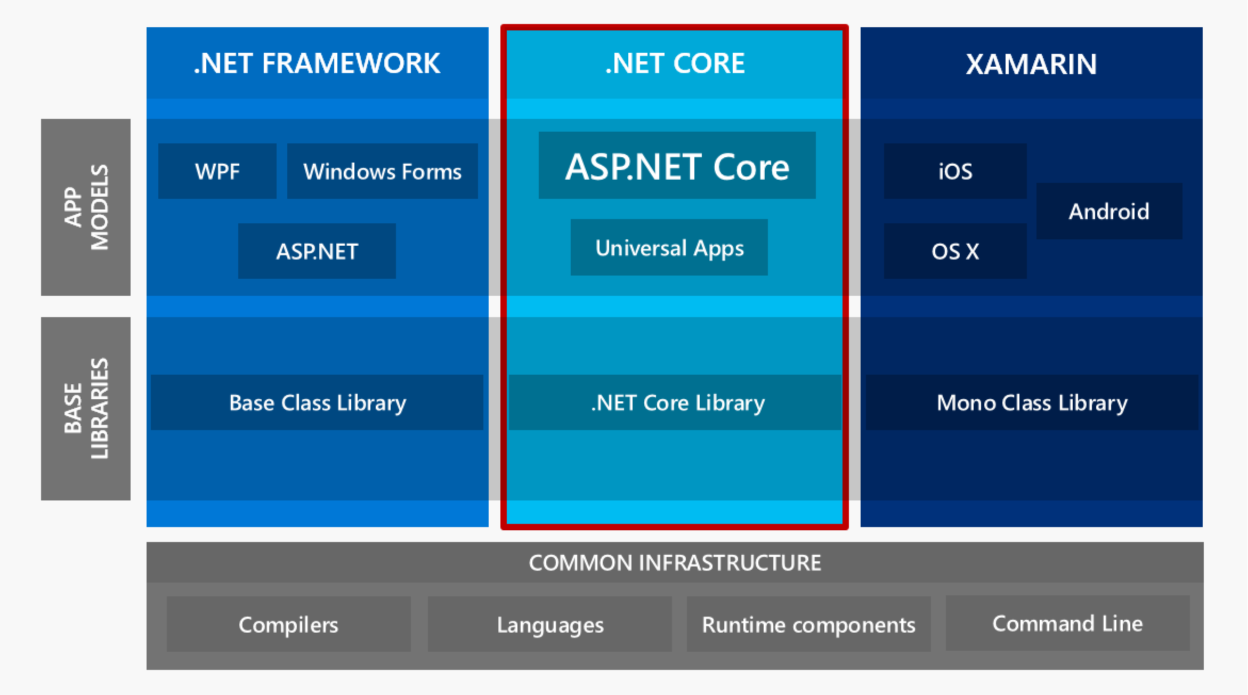
- Difference between .NET Framework and .NET Core
- Use of BCL in different types of .NET apps

- IDE selection: Visual Studio vs Visual Studio Code
- C# as a language for backend, desktop, mobile, and game development

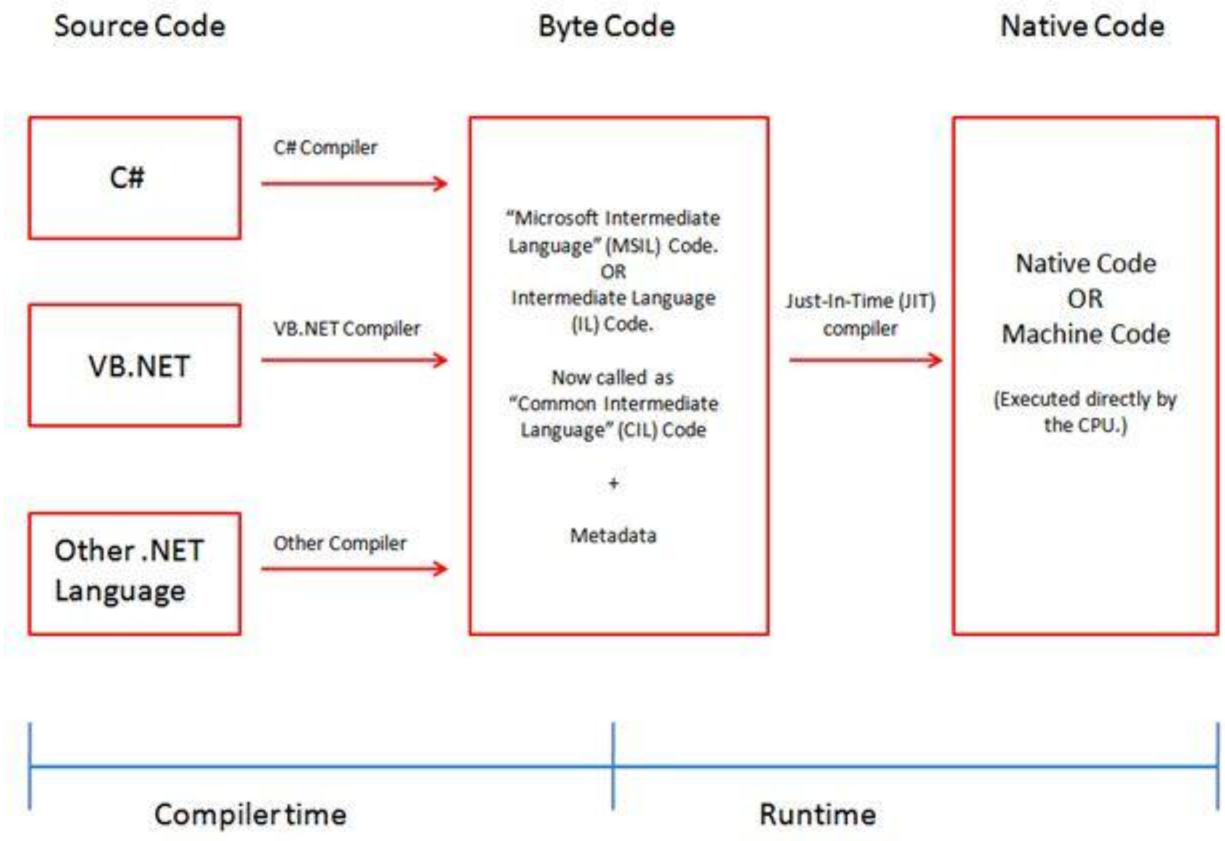
How it started in 2022 !!

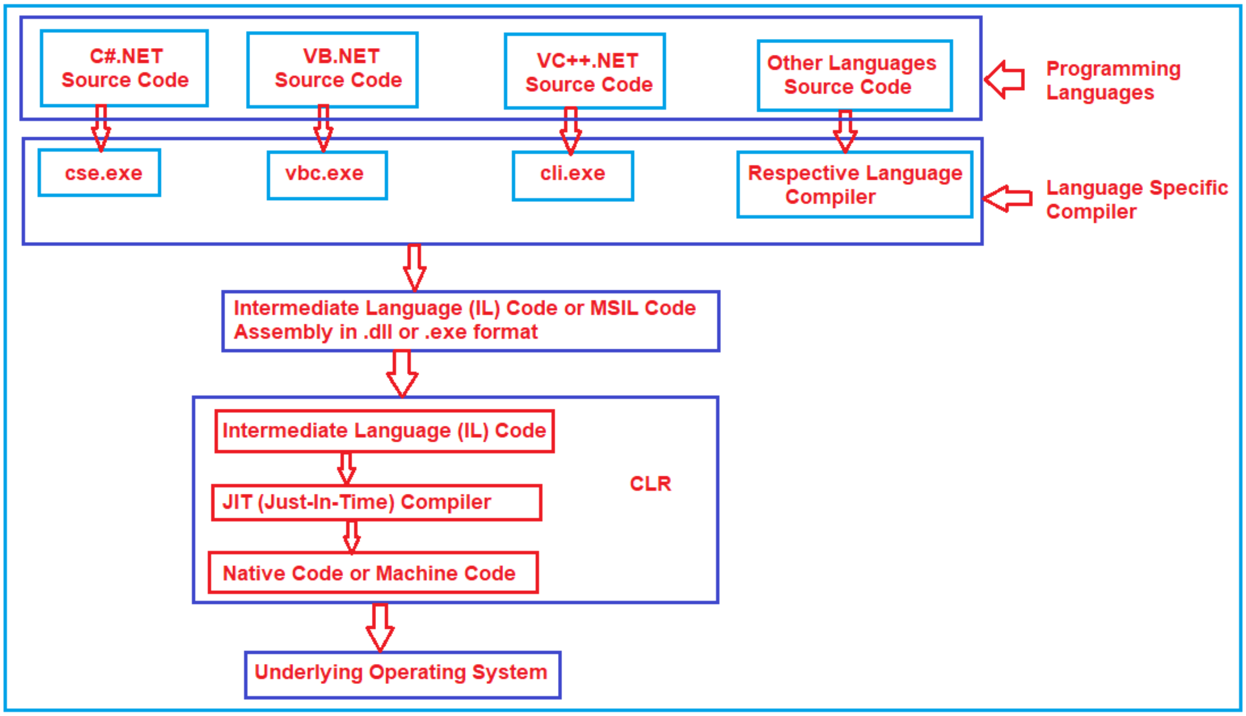

VB
C++
C#
JScript
J#
...
Common Language Specification
ASP.NET: Web Services and Web Forms
Windows Forms
ADO.NET: Data and XML
Base Class Library
Common Language Runtime
.Net supports about 70+ languages

How it is going in 2025 !!



Stages of code compilation in .NET environment:


Source Code
Byte Code
Native Code
C#
C# Compiler
VB.NET
VB.NET Compiler
Other .NET Language
Other Compiler
"Microsoft Intermediate Language" (MSIL) Code.
OR
Intermediate Language (IL) Code.
Now called as "Common Intermediate Language" (CIL) Code
+
Metadata
Just-In-Time (JIT) compiler
Native Code
OR
Machine Code
(Executed directly by the CPU.)
Compiler time
Runtime


C#.NET Source Code
VB.NET Source Code
VC++.NET Source Code
Other Languages Source Code
Programming Languages
cse.exe
vbc.exe
cli.exe
Respective Language Compiler
Language Specific Compiler
Intermediate Language (IL) Code or MSIL Code Assembly in .dll or .exe format
Intermediate Language (IL) Code
JIT (Just-In-Time) Compiler
Native Code or Machine Code
CLR
Underlying Operating System

Viva and Interview Questions

.NET Ecosystem & Architecture

1. What is CLR and what is its role in .NET?
2. How does CTS ensure interoperability in .NET?
3. What are the key components of the .NET Architecture?
4. Explain the difference between .NET Framework, .NET Core, and .NET 5/6/7.
5. What is the use of BCL in .NET?

Visual Studio and Project Setup

6. What workloads are needed to install for a .NET Desktop application?
7. How do project templates help in software development?
8. What is the role of Solution Explorer in Visual Studio?

C# Basics

9. What are value types and reference types in C#?

	Value Types	Reference Types
Ex	**Int, float, bool, struct, enum**	**Class, objects, string, array, interface, delegate**
Storage	Store on Stack	Stored Memory on Heap, reference (pointers) are stored on stack
Copy behavior	It creates a new copy of the value	It creates a reference to the same object
Memory allocation	Memory allocation is immediate	Memory allocation is done at run time (Dynamic)
Data handling	Holds actual data	It holds the address/ reference of the data.

Performance	Slightly faster for small data size	Slightly slower due to heap memory allocation and garbage collection
Null assignment	Cannot be null(unless made nullable with ?)	Can be null

10. How is `var` different from other data types in C#?

11. How does `Console.ReadLine()` work? How is it different from `Read()`?

12. What is the structure of a basic C# console program?

13. Why do we use `static void Main()` as the entry point?

Application Development

14. What happens when you run a C# program in Visual Studio?

15. What is the purpose of the `using System;` directive?