UNIT I
An Introduction to .NET Framework

MCA Semester III

Prof. Hetal M. Patel

AGENDA

- The Evolution of Web Development
- Active Server Page (ASP).NET
- Server-Side and Client Side Programming
- The .NET Framework: CLR
- The .NET Class Library
- Features of Visual Studio .NET

Introduction

- **.NET** Framework is a software development framework for building and running applications on Windows.
- **.NET** Framework is part of the .NET platform, a collection of technologies for building apps for Linux, macOS, Windows, iOS, Android, and more.
- It supports running websites, services, desktop apps, and more on Windows.
- **.NET** is a cross-platform implementation for running websites, services, and console apps on Windows, Linux, and macOS.
- **.NET** is open source on GitHub. .NET was previously called .NET Core.
- **Xamarin/Mono** is a .NET implementation for running apps on all the major mobile operating systems, including iOS and Android.

The Evolution of Web Development

- The Internet began in the late 1960s as an experiment with a goal to create a truly resilient information network—one that could withstand the loss of several computers without preventing the others from communicating.
- Driven by potential disaster scenarios (such as nuclear attack), the U.S. Department of Defense provided the initial funding.
- The early Internet was mostly limited to educational institutions and defense contractors.
- It flourished as a tool for academic collaboration, allowing researchers across the globe to share information.
- In the early 1990s, modems were created that could work over existing phone lines, and the Internet began to open up to commercial users.
- In 1993, the first HTML browser was created, and the Internet revolution began.

The Evolution of Web Development

- **HTML and HTML Forms**
- Early websites actually cannot be called web applications but often looked more like brochures, consisting mostly of fixed HTML pages that needed to be updated manually.
- A basic HTML page is a little like a word-processing document—it contains formatted static content that can be displayed on your computer.
- An HTML document has two types of content: the text and the elements (or tags) that tell the browser how to format it.
- HTML 2.0 introduced the first seed of web programming with a technology called HTMLforms.
- HTML forms expand HTML so that it includes not only formatting tags but also tags for graphical widgets, or controls. These controls include common ingredients such as drop-down lists, text boxes, and buttons.

The Evolution of Web Development

- **HTML and HTML Forms**
- The controls used with HTML forms more than ten years ago are still the basic foundation that you'll use to build dynamic ASP.NET pages.
- The difference is the type of application that runs on the server side. Then CGI (Common Gateway Interfaces) was used to handle server side processing.
- Now it is replaced by mush more capable ASP.NET platform

The Evolution of Web Development

- **Server-Side Programming**
- With the original CGI standard the web server must launch a completely separate instance of the application for each web request.
- If the website is popular, the web server struggles under the weight of hundreds of separate copies of the application, eventually becoming a victim of its own success.
- Higher level features like to authenticate users, store personalized information, or display records you've retrieved from a database required to write a lot of code from the scratch.
- To counter these problems, Microsoft created higher-level development platforms, such as ASP and ASP.NET.
- It allows developers to program dynamic web pages without worrying about the low-level implementation details.

The Evolution of Web Development

- **Server-Side Programming**
- The original ASP platform garnered a huge audience of nearly one million developers but felled victim of its own being applied to unusual places resulting in performance, security, and configuration problems.
- That’s where ASP.NET comes into the picture. ASP.NET was developed as an industrial strength web application framework that could address the limitations of ASP.
- ASP.NET came up with better performance, better design tools and a reach set of readymade features.

The Evolution of Web Development

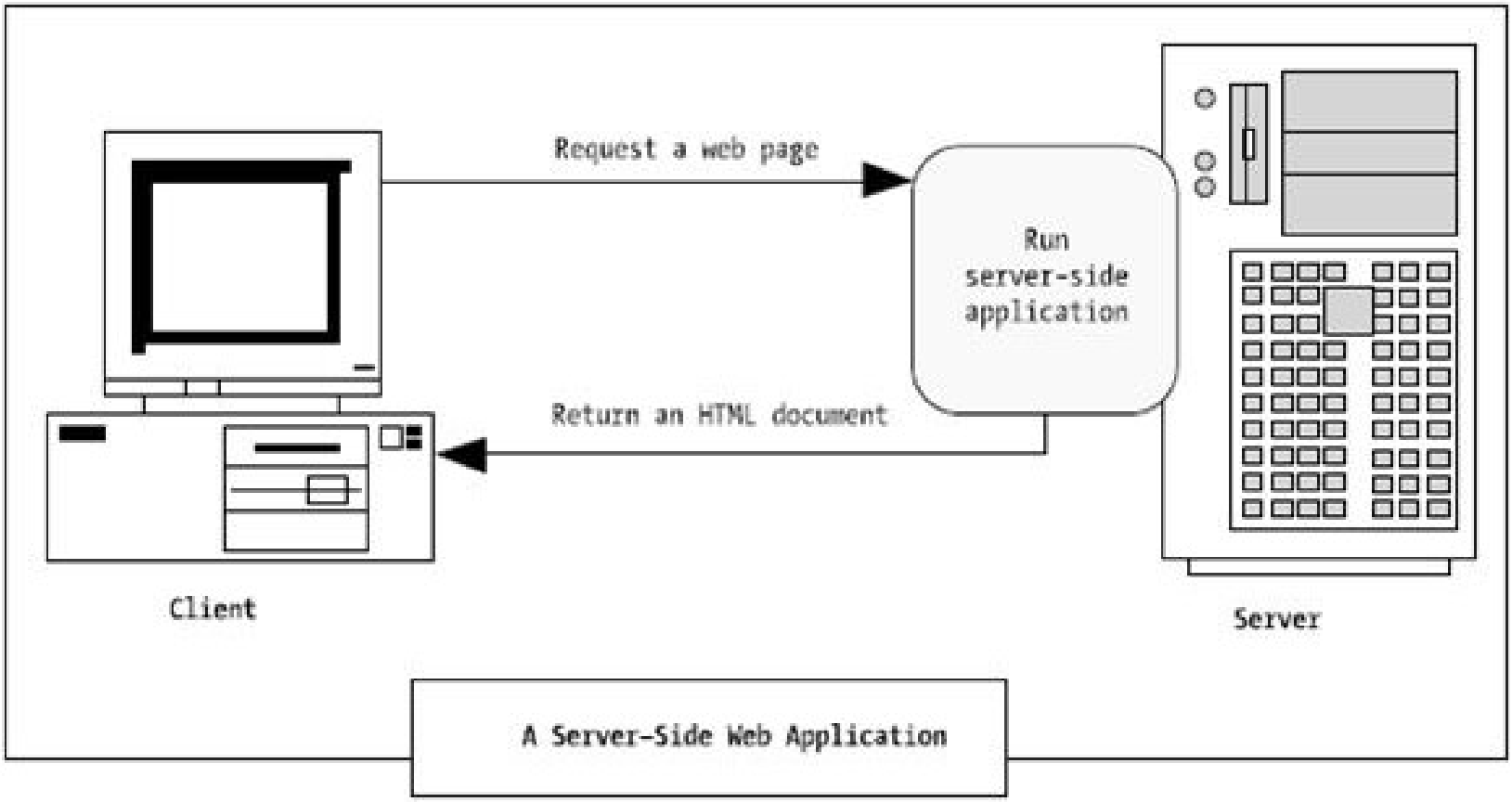
- **Client-Side Programming**
- With the client side technologies, the complete application is downloaded to the browser and runs locally but ironically all applications are not equally supported by all browsers.
- Web development is so popular because using a web application only requires an internet connection.
- But when developers use client-side technologies, they encounter a few familiar headaches. Suddenly, cross-browser compatibility becomes a problem.
- Developers are forced to test their websites with different operating systems and browsers, and they might even need to distribute browser updates to their clients.

The Evolution of Web Development

- **Client-Side Programming**
- So the client-side model sacrifices some of the most important benefits of web development.
- For that reason, ASP.NET is designed as a server-side technology. All ASP.NET code executes on the server. When the code is finished executing, the user receives an ordinary HTML page, which can be viewed in any browser.
- Some other issues with client side programming are
  - isolation (Way difficult to access server resources),
  - Security(users can view client-side code) and
  - thin clients (web enabled devices can communicate with web servers, but they don't support all the features of a traditional browser).
- However, client side programming is not completely dead

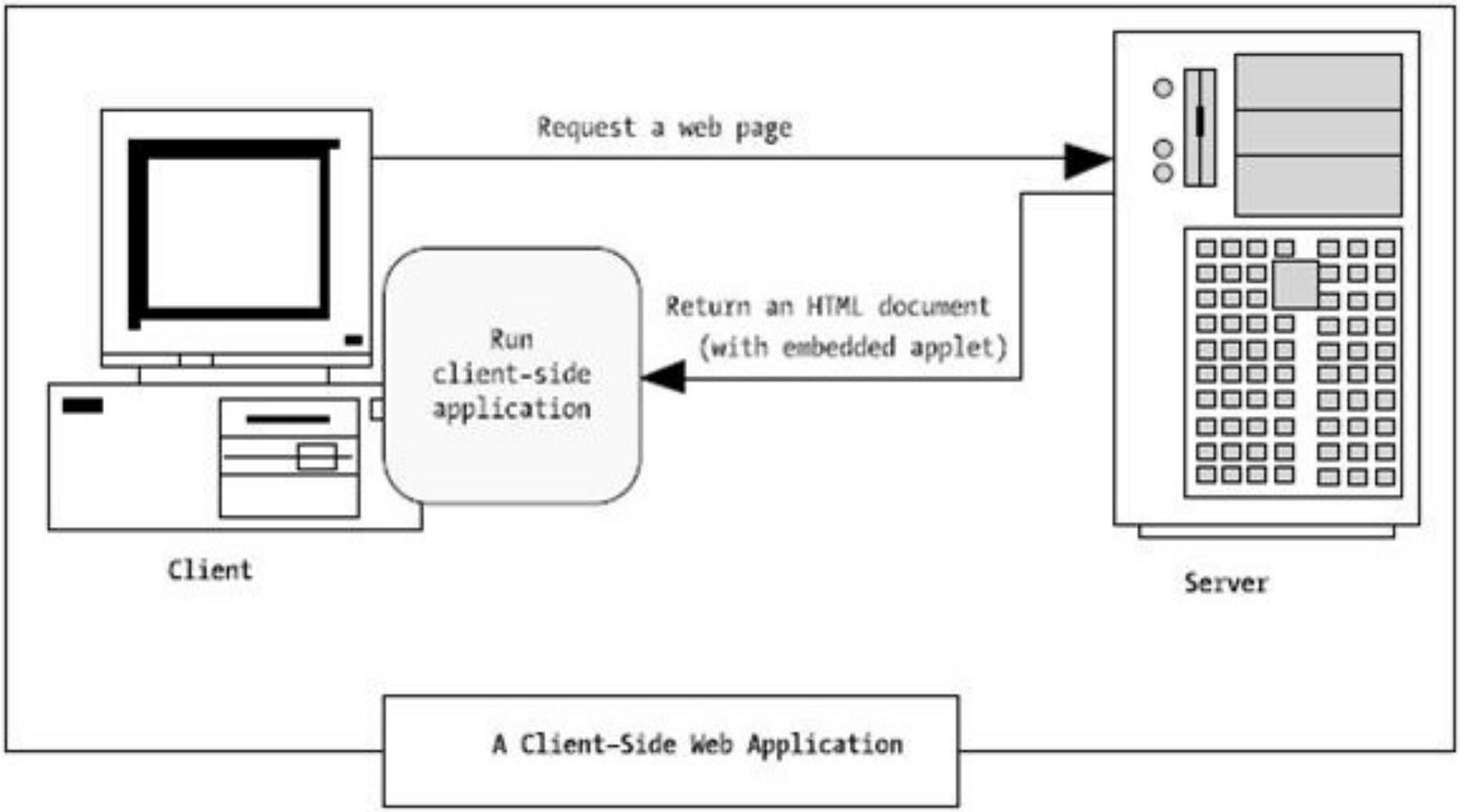
The Evolution of Web Development

- Figure shows the server-side model.



The Evolution of Web Development

- Figure the client-side model.



Architecture of .NET Framework

- The two major components of .NET Framework are the Common Language Runtime and the .NET Framework Class Library.
- The **Common Language Runtime (CLR)** is the execution engine that handles running applications. It provides services like thread management, garbage collection, type-safety, exception handling, and more.
- The **Class Library** provides a set of APIs and types for common functionality. It provides types for strings, dates, numbers, etc. The Class Library includes APIs for reading and writing files, connecting to databases, drawing, and more.

Architecture of .NET Framework

- .NET applications are written in the C#, F#, or Visual Basic programming language. Code is compiled into a language-agnostic Common Intermediate Language (CIL).
- Compiled code is stored in assemblies—files with a .dll or .exe file extension.
- When an app runs, the CLR takes the assembly and uses **a just-in-time compiler (JIT)** to turn it into machine code that can execute on the specific architecture of the computer it is running on.

Architecture of .NET Framework

- The services that .NET Framework provides to running apps include the following:
- Memory management.
- In many programming languages, programmers are responsible for allocating and releasing memory and for handling object lifetimes. In .NET Framework apps, the CLR provides these services on behalf of the app.
- A common type system
- In traditional programming languages, basic types are defined by the compiler, which complicates cross-language interoperability. In .NET Framework, basic types are defined by the .NET Framework type system and are common to all languages that target .NET Framework.
- Version compatibility
- With rare exceptions, apps that are developed by using a particular version of .NET Framework run without modification on a later version.

Architecture of .NET Framework

- An extensive class library
- Instead of having to write vast amounts of code to handle common low-level programming operations, programmers use a readily accessible library of types and their members from the .NET Framework Class Library.
- Development frameworks and technologies
- .NET Framework includes libraries for specific areas of app development, such as ASP.NET for web apps, ADO.NET for data access, Windows Communication Foundation for service-oriented apps, and Windows Presentation Foundation for Windows desktop apps.
- Language interoperability
- Language compilers that target .NET Framework emit an intermediate code named Common Intermediate Language (CIL), which, in turn, is compiled at run time by the common language runtime. With this feature, routines written in one language are accessible to other languages, and programmers focus on creating apps in their preferred languages.

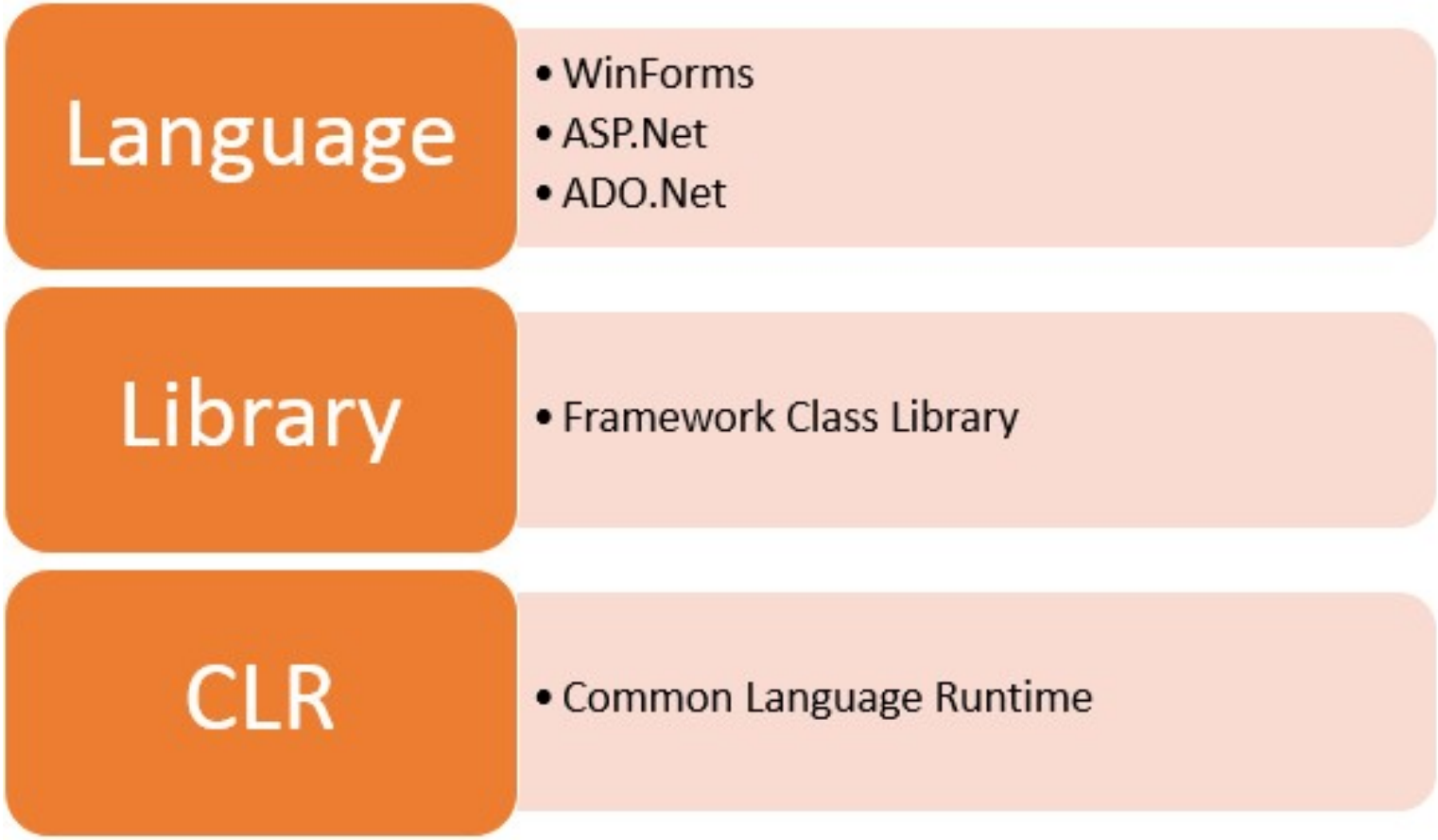
Architecture of .NET Framework

- Side-by-side execution
- .NET Framework helps resolve version conflicts by allowing multiple versions of the common language runtime to exist on the same computer. This means that multiple versions of apps can coexist and that an app can run on the version of .NET Framework with which it was built. Side-by-side execution applies to the .NET Framework version groups 1.0/1.1, 2.0/3.0/3.5, and 4/4.5.x/4.6.x/4.7.x/4.8.x.
- Multitargeting
- By targeting .NET Standard, developers create class libraries that work on multiple .NET Framework platforms supported by that version of the standard.
- For example, libraries that target .NET Standard 2.0 can be used by apps that target .NET Framework 4.6.1, .NET Core 2.0, and UWP 10.0.16299.

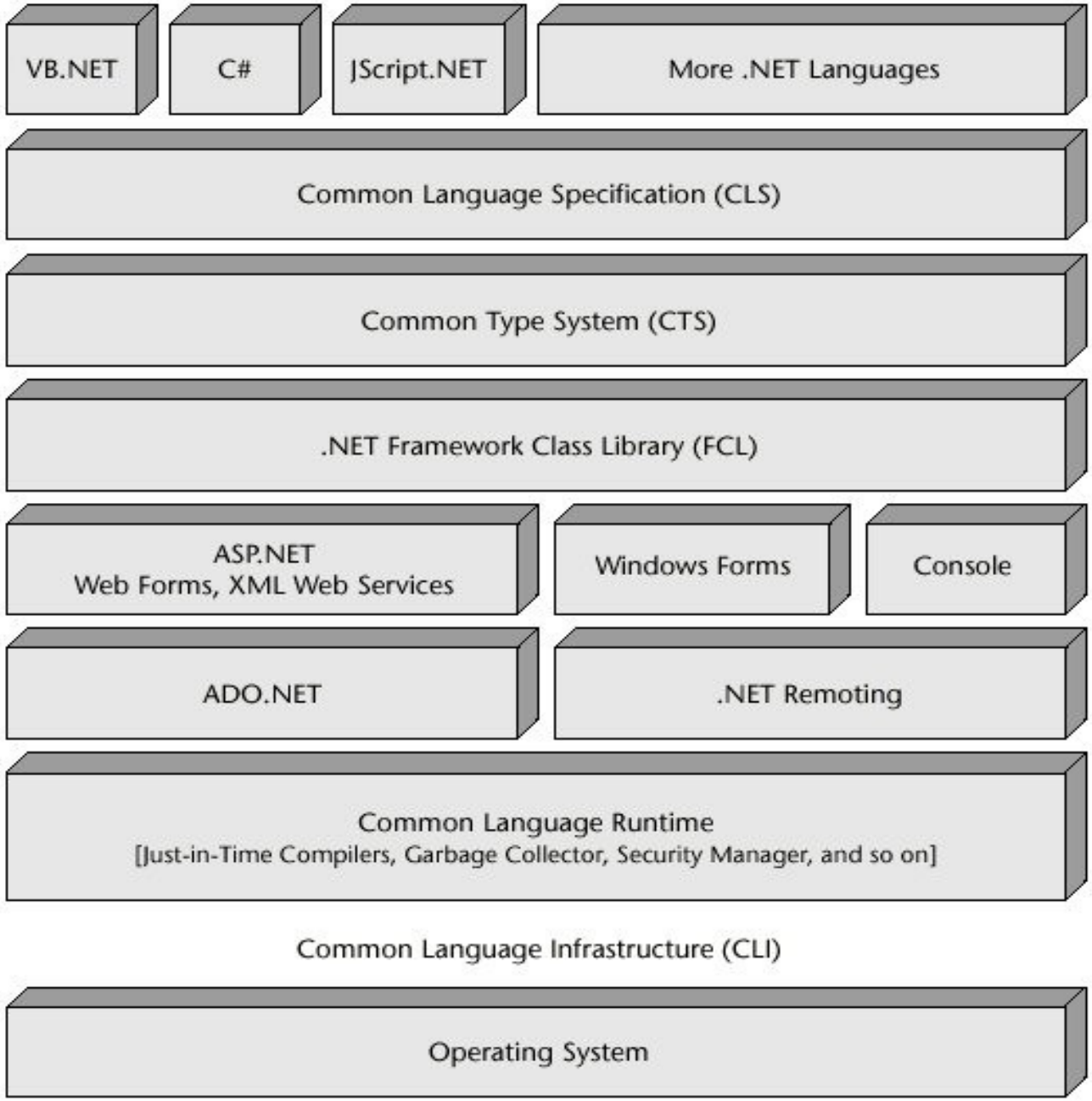
Overview of .NET Framework release history

Generation	Version number	Release date	Development tool	Distributed with
1.0	1.0	13 February 2002	Visual Studio .NET	N/A
1.1	1.1	24 April 2003	Visual Studio .NET 2003	Windows Server 2003
2.0	2.0	7 November 2005	Visual Studio 2005	Windows Server 2003 R2
3.0	3.0	6 November 2006	Expression Blend	Windows Vista, Windows Server 2008
3.5	3.5	19 November 2007	Visual Studio 2008	Windows 7, Windows Server 2008 R2
4.0		12 April 2010	Visual Studio 2010	N/A
4.5	4.5.2	15 August 2012	Visual Studio 2012	Windows 8, Windows Server 2012
4.6	4.6.2	July 20, 2015	Visual Studio 2015	Windows 10, Windows 8.1. Windows Server 2012 R2
4.7	4.7.2	April 5, 2017	Visual Studio 2017	Windows 10, Windows 8.1. Windows Server 2016
4.8	4.8.1	April 18, 2019	Visual Studio 2019/2022	Windows 11, Windows 10. Windows Server 2019/2022

Architecture of .NET Framework



.Net Framework Architecture Diagram


VB.NET
C#
JScript.NET
More .NET Languages
Common Language Specification (CLS)
Common Type System (CTS)
.NET Framework Class Library (FCL)
ASP.NET
Web Forms, XML Web Services
Windows Forms
Console
ADO.NET
.NET Remoting
Common Language Runtime
[Just-in-Time Compilers, Garbage Collector, Security Manager, and so on]
Common Language Infrastructure (CLI)
Operating System

.NET Framework

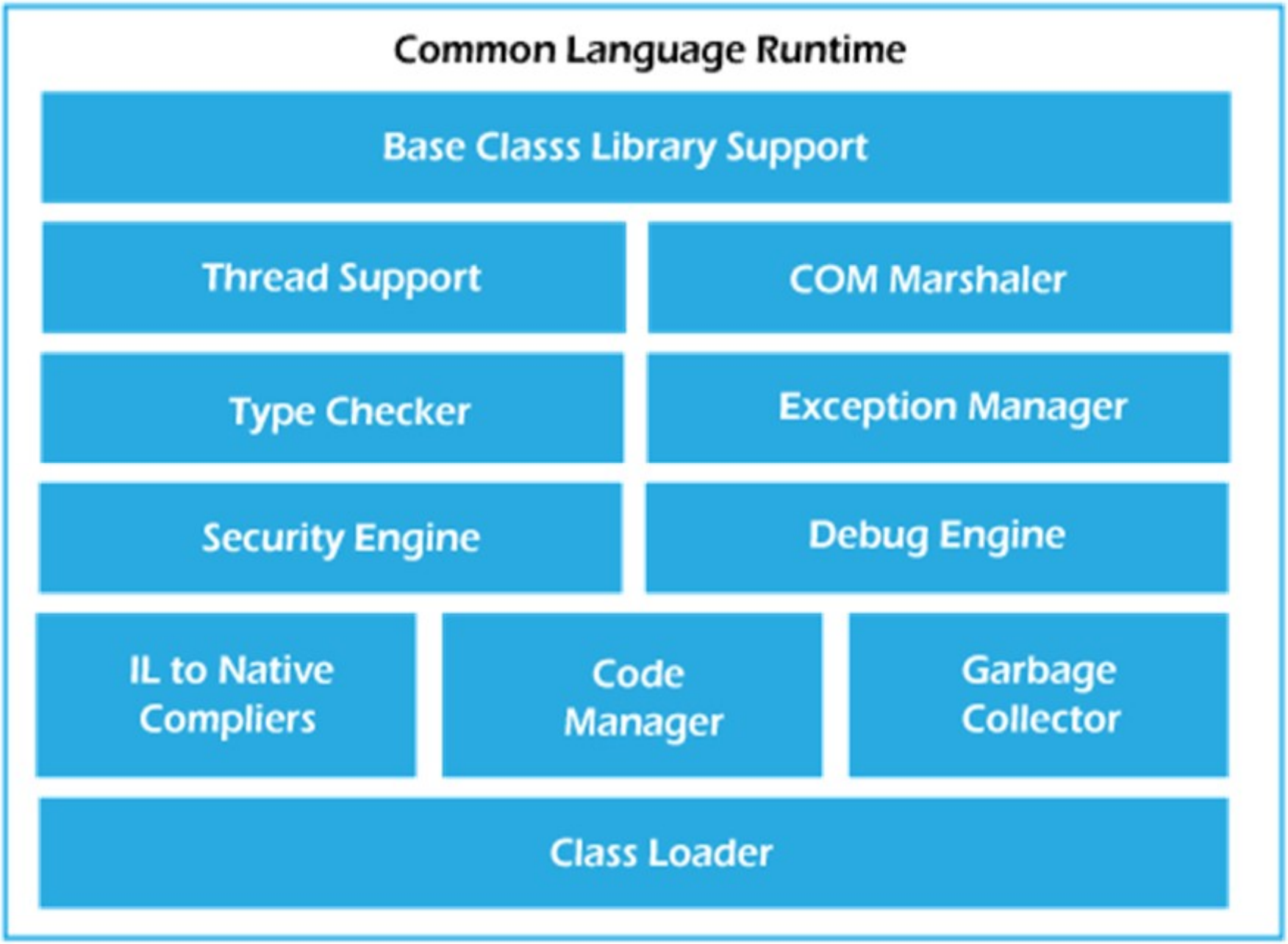
- The .NET Framework is composed of four main components:
- Common Language Runtime (CLR)
- Framework Class Library (FCL),
- Core Languages (WinForms, ASP.NET, and ADO.NET), and
- Other Modules (WCF, WPF, WF, Card Space, LINQ, Entity Framework, Parallel LINQ, Task Parallel Library, etc.)

.NET Framework

- The .NET Framework is composed of four main components:
- Common Language Runtime (CLR)
- Framework Class Library (FCL)
- Core Languages (WinForms, ASP.NET, and ADO.NET), and
- Other Modules (WCF, WPF, WF, Card Space, LINQ, Entity Framework, Parallel LINQ, Task Parallel Library, etc.)

.NET Framework: CLR

- The Common Language Runtime handles program execution for various languages such as C#, F#, Visual Basic .NET, etc.
- The managed execution environment provides various services such as memory management, security handling, exception handling, garbage collection, thread management, etc.
- The Common Language Runtime implements the VES (Virtual Execution System) which is a run time system that provides a managed code execution environment.
- The VES is defined in Microsoft's implementation of the CLI (Common Language Infrastructure).


Common Language Runtime
Base Classs Library Support
Thread Support
COM Marshaler
Type Checker
Exception Manager
Security Engine
Debug Engine
IL to Native Compliers
Code Manager
Garbage Collector
Class Loader

.NET Framework: CLR

- There are multiple components in the architecture of Common Language Runtime. Details about these are given as follows:
- **Base Class Library Support:** The Common Language Runtime provides support for the base class library. The BCL contains multiple libraries that provide various features such as *Collections*, *I/O*, *XML*, *DataType definitions*, etc. for the multiple *.NET* programming languages.
- **Thread Support:** The CLR provides thread support for managing the parallel execution of multiple threads. The *System.Threading class* is used as the base class for this.
- **COM Marshaller:** Communication with the COM (Component Object Model) component in the .NET application is provided using the COM marshaller. This provides the COM interoperability support.

.NET Framework: CLR

- **Type Checker:** Type safety is provided by the type checker by using the Common Type System (CTS) and the Common Language Specification (CLS) that are provided in the CLR to verify the types that are used in an application.
- **Exception Manager:** The exception manager in the CLR handles the exceptions regardless of the *.NET Language* that created them. F
- or a particular application, the catch block of the exceptions are executed in case they occur and if there is no catch block then the application is terminated.
- **Security Engine:** The security engine in the CLR handles the security permissions at various levels such as the code level, folder level, and machine level.
- This is done using the various tools that are provided in the *.NET* framework.

.NET Framework: CLR

- **Debug Engine:** An application can be debugged during the run-time using the debug engine. There are various ICorDebug interfaces that are used to track the managed code of the application that is being debugged.
- **JIT Compiler:** The JIT compiler in the CLR converts the Microsoft Intermediate Language (MSIL) into the machine code that is specific to the computer environment that the JIT compiler runs on. The compiled MSIL is stored so that it is available for subsequent calls if required.
- **Code Manager:** The code manager in CLR manages the code developed in the .NET framework i.e. the managed code.
- The managed code is converted to intermediate language by a language-specific compiler and then the intermediate language is converted into the machine code by the Just-In-Time (JIT) compiler.

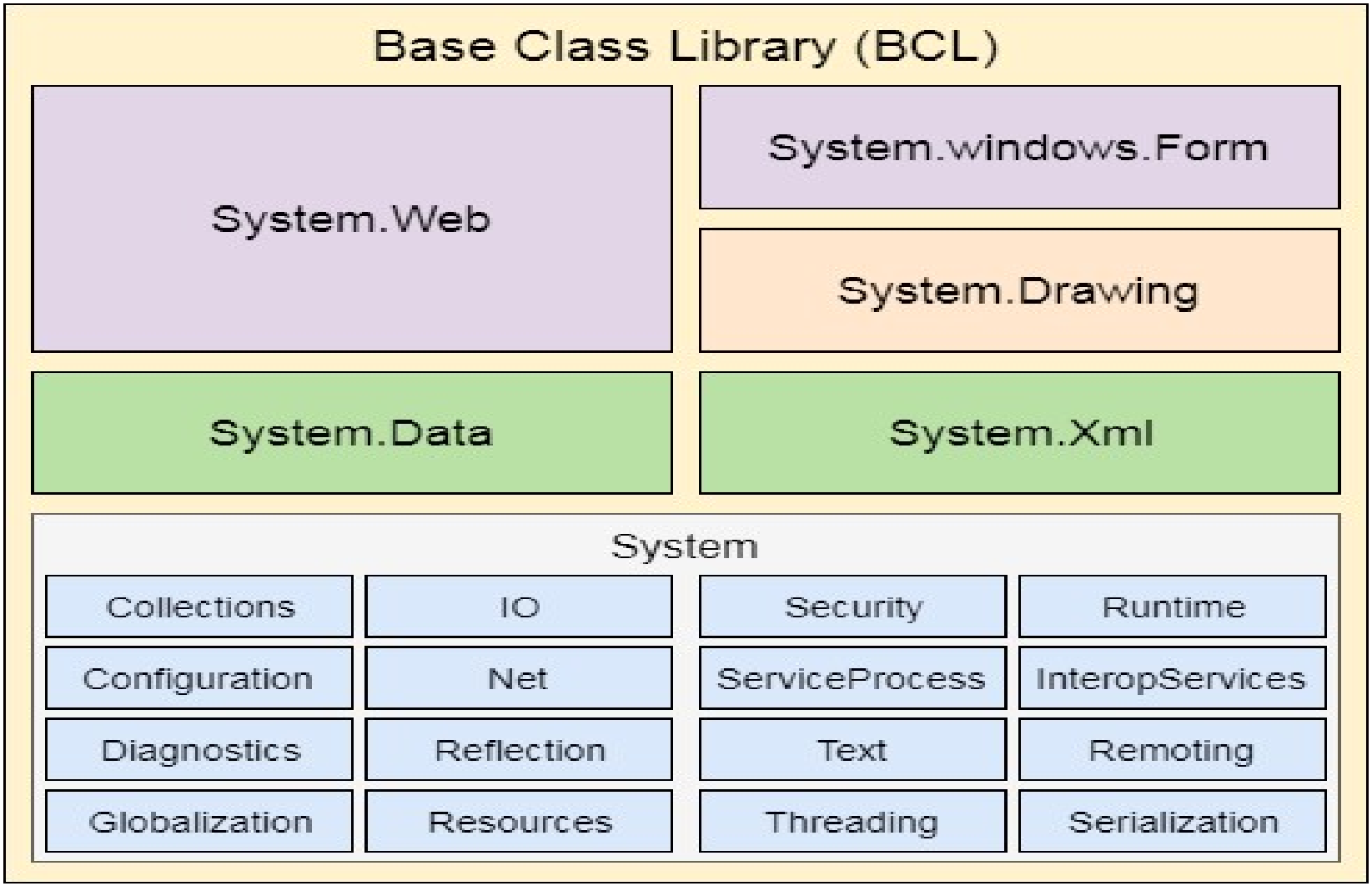
.NET Framework: CLR

- **Garbage Collector:** Automatic memory management is made possible using the garbage collector in CLR.
- The garbage collector automatically releases the memory space after it is no longer required so that it can be reallocated.
- **CLR Loader:** Various modules, resources, assemblies, etc. are loaded by the CLR loader. Also, this loader loads the modules on demand if they are actually required so that the program initialization time is faster and the resources consumed are lesser.

.NET Class Library

- NET Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.
- It contains thousands of classes that supports the following functions.
- Base and user-defined data types
- Support for exceptions handling
- input/output and stream operations
- Communications with the underlying system
- Access to data
- Ability to create Windows-based GUI applications
- Ability to create web-client and server applications
- Support for creating web services

.NET Class Library



.NET Framework Base Class Library

.NET Class Library

- **Namespaces in the Framework Class Library**
- Namespace is a group of classes, structures, interfaces, enumerations, and delegates, organized in a logical hierarchy by function, that enable you to access the core functionality you need in your applications.
- Namespaces are the way that .NET avoids name clashes between classes.
- A namespace is no more than a grouping of data types, but it has the effect that the names of all data types within a namespace automatically get prefixed with the name of the namespace. It is also possible to nest namespace within each other.
- Ex. . System. The System.Activities namespace handles the creation and working with activities in the Window Workflow Foundation using various classes.

.NET Class Library

- **Namespaces in the Framework Class Library**
- Each namespace in the FCL can contain multiple namespace, or they can contain classes that expose properties and methods that you call in your applications.
- The namespaces within the FCL are grouped by the functionality they provide, which makes it very easy to find what you're looking for.

.NET Class Library

- **The ASP.NET Framework gives you the most commonly used namespaces for free. These namespaces are**
- System
- System.Collections
- System.Collections.Specialized
- System.Configuration
- System.Text
- System.Text.RegularExpressions
- System.Web
- System.Web.Caching
- System.Web.SessionState
- System.Web.Security
- System.Web.Profile
- System.Web.UI
- System.Web.UI.WebControls
- System.Web.UI.WebControls.WebParts

Assemblies

- An assembly is the actual .dll file on your hard drive where the classes in the .NET Framework are stored.
- For example, all the classes contained in the ASP.NET Framework are located in an assembly named System.Web.dll.
- More accurately, an assembly is the primary unit of deployment, security, and version control in the .NET Framework.
- Because an assembly can span multiple files, an assembly is often referred to as a "logical" dll.

Assemblies

- There are two types of assemblies: private and shared. A private assembly can be used by only a single application.
- A shared assembly, on the other hand, can be used by all applications located on the same server.
- Shared assemblies are located in the Global Assembly Cache (GAC).
- For example, the System.Web.dll assembly and all the other assemblies included with the .NET Framework are located in the Global Assembly Cache.
- The Global Assembly Cache is located physically in your computer's Assembly folder.
- There is a separate copy of every assembly in your .NET Framework.
- The first set of assemblies is used at runtime and the second set is used at compile time.

Assemblies

- Before you can use a class contained in an assembly in your application, you must add a reference to the assembly.
- By default, an ASP.NET application references the most common assemblies contained in the Global Assembly Cache:
- mscorlib.dll
- System.dll
- System.Configuration.dll
- System.Web.dll
- System.Data.dll
- System.Web.Services.dll
- System.Xml.dll
- System.Drawing.dll
- System.EnterpriseServices.dll
- System.Web.Mobile.dll

Assemblies

- To use any particular class in the .NET Framework, you must do two things.
- First, your application must reference the assembly that contains the class. Second, your application must import the namespace associated with the class.
- In most cases, you won't worry about referencing the necessary assembly because the most common assemblies are referenced automatically.
- However, if you need to use a specialized assembly, you need to add a reference explicitly to the assembly.
- For example, if you need to interact with Active Directory by using the classes in the System.DirectoryServices namespace then you will need to add a reference to the System.DirectoryServices.dll assembly to your application.

Assemblies

- Each class entry in the .NET Framework SDK documentation lists the assembly and namespace associated with the class.
- For example, if you look up the MessageQueue class in the documentation, you'll discover that this class is located in the System.Messaging namespace located in the System.Messaging.dll assembly.
- If you are using Visual Web Developer, you can add a reference to an assembly explicitly by selecting the menu option Website, Add Reference, and selecting the name of the assembly that you need to reference.
- For example, adding a reference to the System.Messaging.dll assembly results in the web configuration file.

Assemblies

- An assembly is a grouping of files deployed as a single file. An assembly almost always consists of at least two files: the executable and the manifest.
- The manifest is a list of all the files that exist inside the assembly. The executable content inside the assembly is referred to individually as a module.
- Conceptually, modules correspond to DLLs or EXEs; each module contains metadata, in addition to the metadata of its parent assembly.
- The assembly format is an enhanced version of the current Portable Executable (PE) format (your normal Windows .EXE file format).

.NET Features

1. Cross-platform interoperability
2. Multi-language support
3. Code reuse
4. Automatic resource management
5. Debugging
6. Error handling
7. Simplified deployment
8. Elimination of DLL hell
9. Security

Cross-platform Interoperability

- ***Examples***
- A routine written in a language L1 may call another routine written in a different language L2.
- A module in L1 may declare a variable whose type is a class declared in L2, and then call the corresponding L2 routines on that variable.
- If both languages are object oriented, a class in L1 can inherit from a class in L2.
- Exceptions triggered by a routine written in L1 and not handled on the L1 side will be passed to the caller, which—if written in L2—will process it using L2's own exception-handling mechanism.

Multi Language Support

- The .NET platform supports many programming languages. A new compiler must be implemented for each language.

- Programmers do not need to be retrained in a completely new language in order to gain the benefits of .NET.

Code Reuse

- Apps do not need to be rewritten in a completely new language in order to gain the benefits of .NET.

- For example all the billions of lines of COBOL code with some porting effort, could become useable within the .NET environment.

Automatic resource management

- No need to allocate memory
- No need to deallocate memory
  - Garbage collector

The runtime environment automatically handles object layout and manages references to objects, releasing them when they are no longer being used.

This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

Debugging

During a debugging session, you may move freely and seamlessly across modules written in L1 and L2.

Error handling

- .NET provides structured exception handling, similar to that in C++ or Java, as a fundamental feature available to all languages.
- This architecture solves many of the problems that have dogged error handling in the past.

Simplified deployment

- The .NET Framework includes design features and tools which help manage the installation of computer software to ensure it does not interfere with previously installed software, and it conforms to security requirements.

DLL hell

- Maintaining a Windows PC is a chore, because applications are quite complex. They consist of many files, registry entries, shortcuts, and so on.
- Different applications can share certain DLLs, and installing a new application can overwrite a DLL an existing application depends on, possibly breaking an old application (“DLL hell”).
- Removing an application is complex and is often imperfectly done.

DLL

- Stands for “Dynamic Link Library”.
- Pieces of code that apps could take runtime.

Security

- Managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer).
- This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions.

Security

- The runtime enforces code security.
- For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network.
- The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally feature rich.

ADVANTAGES

- The .Net Framework supports more than 60 programming languages such as C#, F#, VB.NET, J#, VC++, JScript.NET, APL, COBOL, Perl, Oberon, ML, Pascal, Eiffel, Smalltalk, Python, Cobra, ADA, etc.SQL server provide complete protection of data layer.
- Support Graphical user interface to form design.
- .Net base class library provide collection of 22000 inbuilt classes.
- .Net support cross language.
- CLR , a component of .Net provide garbage collection.
- Provide exception handling.

DISADVANTAGES

- Project can run only on system that support Microsoft windows.