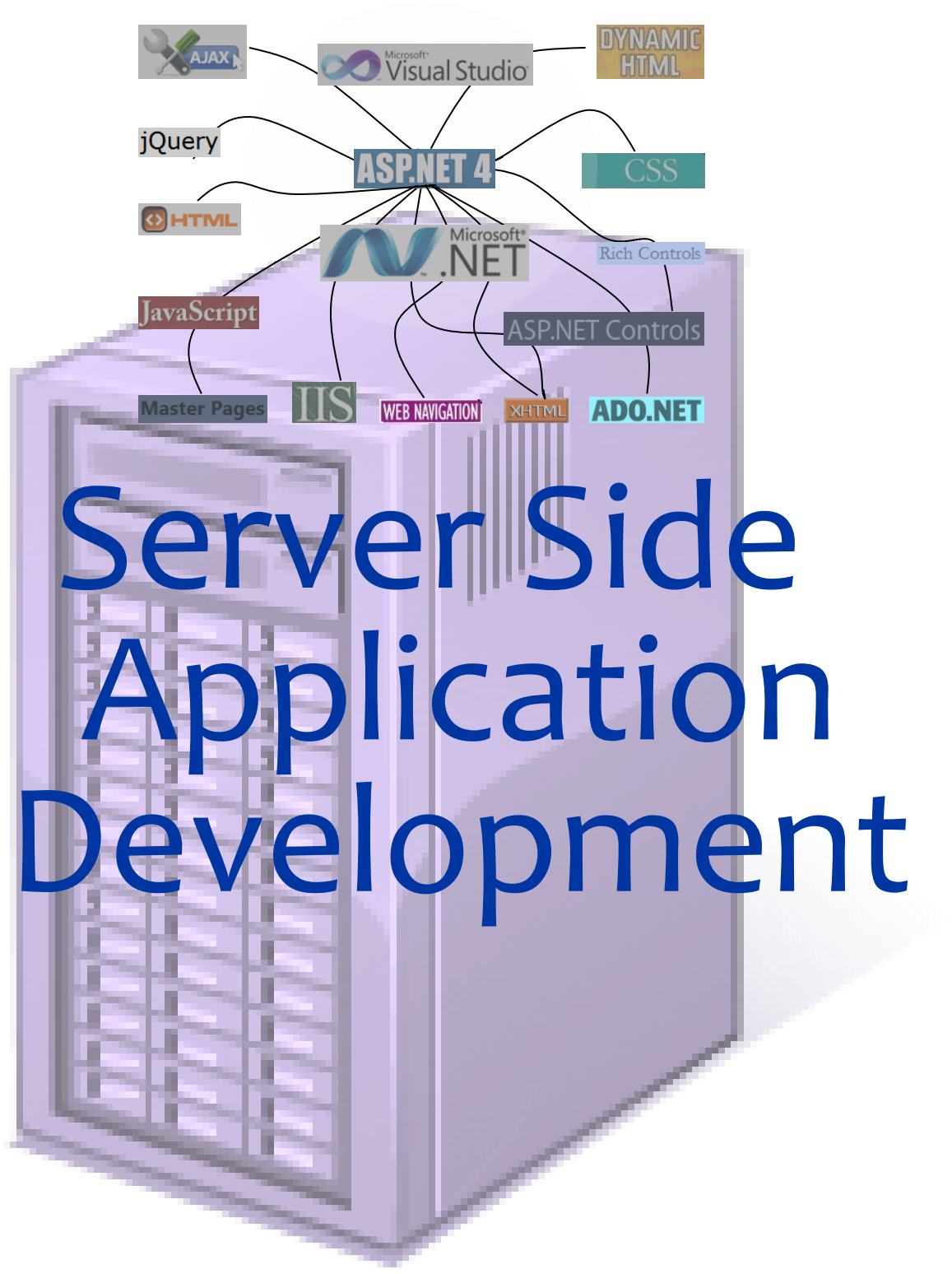
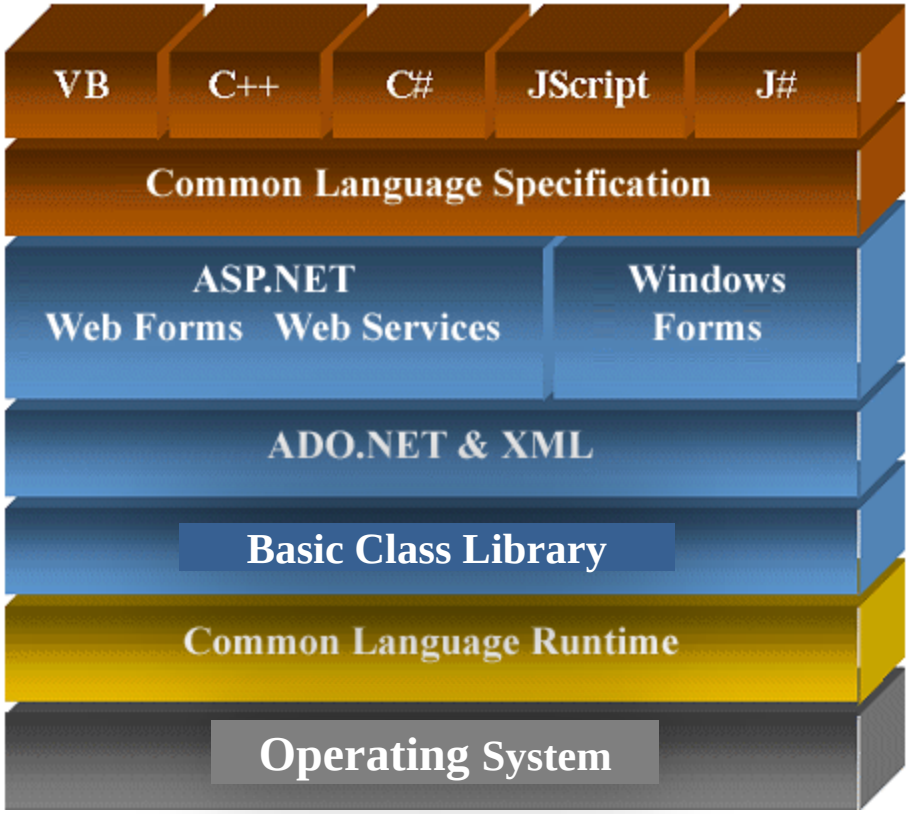


Server Side Application Development

2.0 The .NET 4.5.1 Framework (Text Book Chapter 01)



➢2.0 The .NET 4.5.1 Framework

2.0 Introduction to ASP.NET 4.5.1

The **Microsoft .NET Framework** is a software framework for Microsoft Windows operating systems. It includes a large library, and it supports several programming languages which allows language interoperability (each language can utilize code written in other languages.) The .NET library is available to all the programming languages that .NET supports. The framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. The class library is used by programmers, who combine it with their own code to produce applications.

Programs written for the .NET Framework execute in a software (as contrasted to hardware) environment, known as the Common Language Runtime (CLR). The CLR is an application virtual machine so that programmers need not consider the capabilities of the specific CPU that will execute the program. The CLR also provides other important services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

The .NET Framework is intended to be used by most new applications created for the Windows platform. In order to be able to develop and run applications, it is required to have Microsoft's SDK for Windows 7 or .NET Framework 4.5 or Visual Studio 2013 installed on your computer.

➢2.0 The .NET 4.5.1 Framework

2.1 Design Features

2.1.1 Interoperability

Because computer systems commonly require interaction between new and older applications, the .NET Framework provides means to access functionality that is implemented in programs that execute outside the .NET environment. Access to COM components is provided in the System.Runtime.InteropServices and System.EnterpriseServices namespaces of the framework; access to other functionality is provided using the P/Invoke feature.

2.1.2 Common Runtime Engine

The Common Language Runtime (CLR) is the execution engine of the .NET Framework. All .NET programs execute under the supervision of the CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

2.1.3 Language Independence

The .NET Framework introduces a Common Type System, or CTS. The CTS specification defines all possible datatypes and programming constructs supported by the CLR and how they may or may not interact with each other conforming to the Common Language Infrastructure (CLI) specification. Because of this feature, the .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming .NET language.

2.1.4 Base Class Library

The Base Class Library (BCL), part of the Framework Class Library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes which encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation and so on.
of the framework and its languages on other platforms.

2.1.5 Simplified Deployment

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

2.1.6 Security

The design is meant to address some of the vulnerabilities, such as buffer overflows, that have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

2.1.7 Portability

The design of the .NET Framework allows it to theoretically be platform agnostic, and thus cross-platform compatible. That is, a program written to use the framework should run without change on any type of system for which the framework is implemented. While Microsoft has never implemented the full framework on any system except Microsoft Windows, the framework is engineered to be platform agnostic, and cross-platform implementations are available for other operating systems (see Silverlight and the Alternative implementations section below). Microsoft submitted the specifications for the Common Language Infrastructure (which includes the core class libraries, Common Type System, and the Common Intermediate Language), the C# language, and the C++/CLI language to both ECMA and the ISO, making them available as open standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

2.2 Basic Components of .NET Framework

The .NET Framework is made up of three components, as shown in the Figure below.

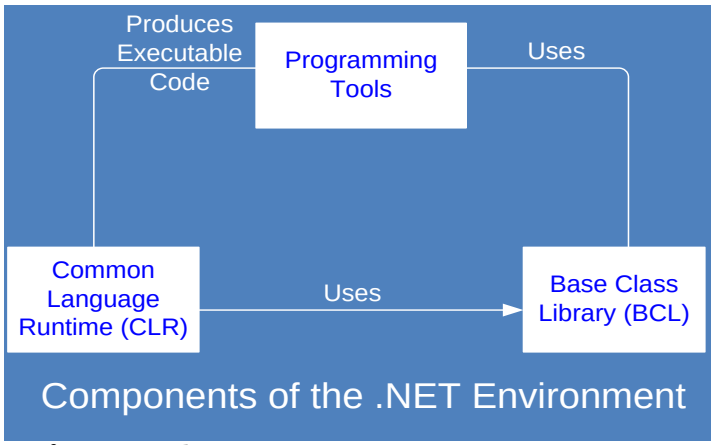


Figure 2.1

2.2.1 programming tools

Progrmming tools include everything you need for coding and debugging, including the following

- The Visual Studio Integrated Development Environment
- .Net compliant compilers (eg. C#, VB, Jscript, and Managed C++)
- Debuggers
- Server-Side improvements as ASP.NET

2.2.3 The Execution Environment [Common Language Runtime(CLR)]

The CLR manages program execution at run time, including the following.

- Memory management
- Code Safety Verification
- Code Execution
- Garbage Collection

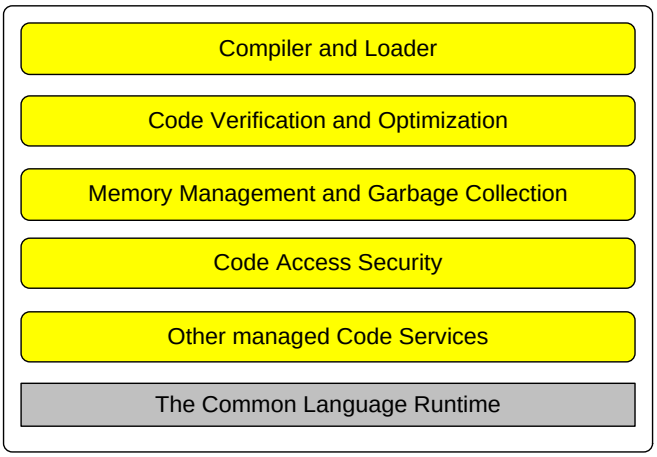


Figure 2.2 - The CLR Components

➢2.0 The .NET 4.5.1 Framework

The Common Language Runtime is the engine that compiles the source code in to an Intermediate Language (IL) (see section 2.3) and runs the code. This intermediate language is called the Microsoft Intermediate Language. It further provides services that Make the development process easier. These things happen "Inside .NET". Investigating what happens inside .NET is out of the scope of this class. However some important things you should remember with CLR are as follows.

During the execution of the program this IL is converted to the native code or the machine code. This conversion is possible through the Just-In-Time compiler. During compilation the end result is a Portable Executable (PE)file.

This portable executable file contains the IL and additional information called the metadata. This metadata describes the assembly that is created. Class names, methods, signature and other dependency information are available in the metadata. Since the CLR compiles the source code to an intermediate language, it is possible to write the code in any language of your choice. This is a major advantage of using the .Net framework.

The other advantage is that the programmers need not worry about managing the memory themselves in the code. Instead the CLR will take care of that through a process called Garbage collection. This frees the programmer to concentrate on the logic of the application instead of worrying about memory handling.

2.2.4 Base Class Library (BCL)

The Base Class Library is a large class library used by the .NET Framework and available for you to use In your programs as well (See the next page for an overview of components).

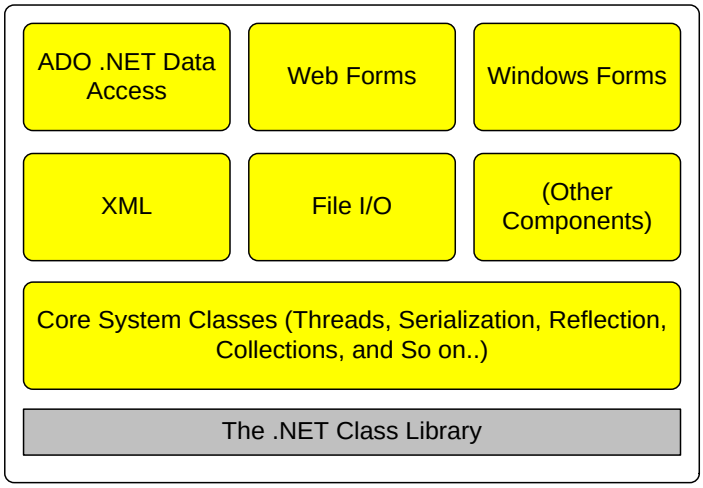


Figure 2.3 The BCL Components

2.3 Common Intermediate Language

Any code you develop using a .NET compiler such as C# or VB, it compiles into a common assembly called the intermediate language code. If you are familiar with C# Byte Code, .NET intermediate language is similar to C# byte code. No matter what language you use they all compile into one common language on the .NET environment. Basically every .exe or .DLL file you generate on .NET contains the IL code. These files can then be deployed into other computers.

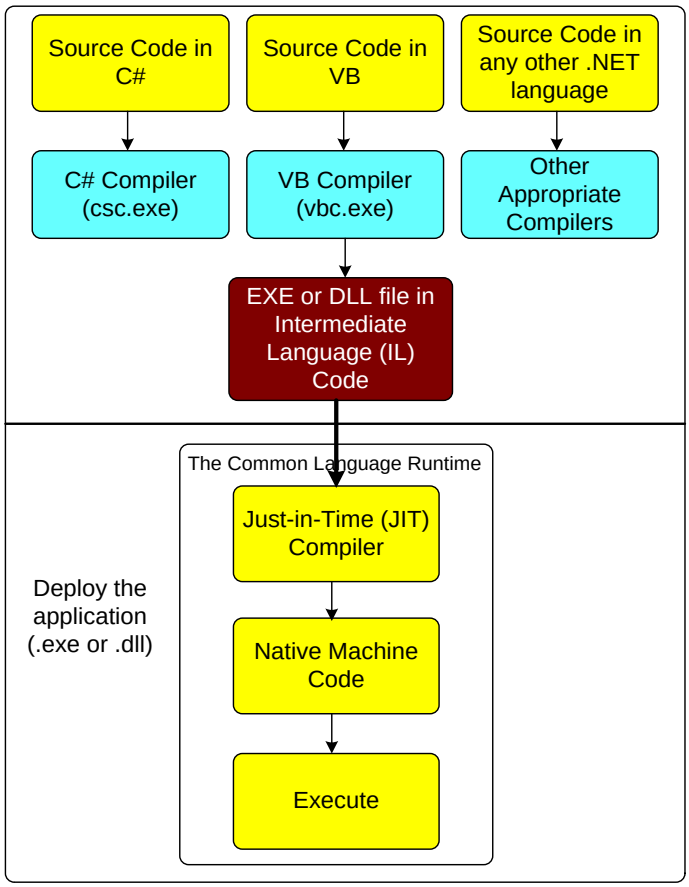


Figure 2.4 Flow of information from build to deployment using .NET