The .NET Framework

Today You will learn

- The Evolution of Web Development
- The .NET Framework

The evolution of Web development

HTML and HTML Forms:

- The first generation of websites consisting mostly of <u>fixed HTML pages</u> that needed to be updated by hand.

- A basic HTML page is a little like a word-processing document—it contains formatted content that can be displayed on your computer.

The evolution of Web development

HTML and HTML Forms:

- The following example shows HTML at its simplest, with a document that contains a heading and a single line of text:

```
<html>
  <head>
    <title>Sample Web Page</title>
  </head>
  <body>
    <h1>Sample Web Page Heading</h1>
    <p>This is a sample web page.</p>
  </body>
</html>
```

The evolution of Web development

HTML and HTML Forms:

- **An HTML document has two types of content:**
  - The **text** and the **elements** (or tags) that tell the browser how to format it.
  - The **elements** are easily recognizable, because they are designated with angled brackets (< >).

- HTML defines elements for different levels of headings, paragraphs, hyperlinks, italic and bold formatting, horizontal lines, and so on.

- The **<head>** element groups the header information together and includes the **<title>** element with the text that appears in the browser window.

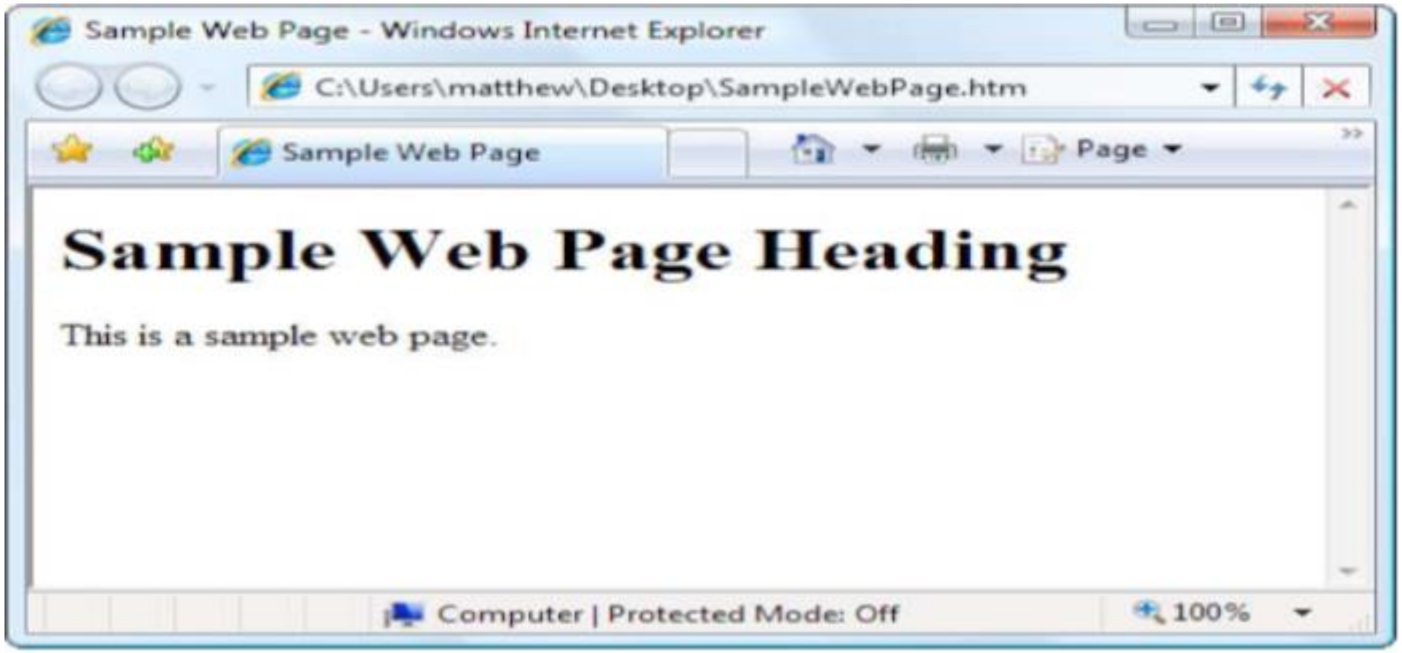
- The **<body>** element groups together the actual document content that's displayed in the browser window.

The evolution of Web development

HTML and HTML Forms:

Figure 1-1 shows this simple HTML page in a browser.

It has **no interactivity**, **doesn't require a web server**, and certainly can't be considered a web application.



Sample Web Page - Windows Internet Explorer
C:\Users\matthew\Desktop\SampleWebPage.htm
Sample Web Page
Page
Computer | Protected Mode: Off
100%

Sample Web Page Heading

This is a sample web page.

Figure 1-1 Ordinary HTML Page

The evolution of Web development

HTML and HTML Forms:

- **HTML 2.0** introduced the first seed of web programming with a technology called ***HTML forms***.
- 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:

- Here is a sample web page created with HTML form controls:

```
<html>
  <head>
    <title>Sample Web Page</title>
  </head>
  <body>
    <form>
      <input type="checkbox" />
      This is choice #1<br />
      <input type="checkbox" />
      This is choice #2<br /><br />
      <input type="submit" value="Submit" />
    </form>
  </body>
</html>
```

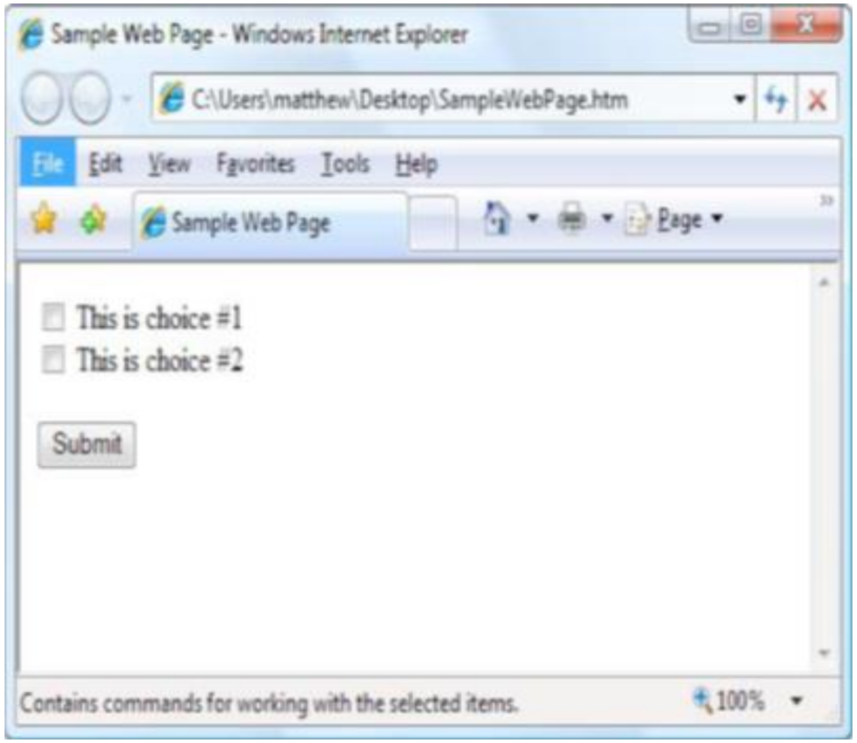

Sample Web Page - Windows Internet Explorer
C:\Users\matthew\Desktop\SampleWebPage.htm
File Edit View Favorites Tools Help
Sample Web Page
Page
This is choice #1
This is choice #2
Submit
Contains commands for working with the selected items.
100%

Figure 1-2. *An HTML form*

The evolution of Web development

HTML and HTML Forms:

- HTML forms allow web developers to design **standard input pages**.

- When the user clicks the **Submit button** on the page shown in Figure 1-2, all the data in the input controls (in this case, the two check boxes) is patched together into one long string of text and sent to the web server.

- On the **server side**, a custom application receives and processes the data.

The evolution of Web development

Server-Side Programming

- Early web development platforms had **two key problems:**
  - First, they did not always scale well.
  - Second, they provided little more than a bare-bones programming environment.

  If you wanted higher-level features, such as the ability to authenticate users, store personalized information, or display records you've retrieved from a database, you needed to write pages of code from scratch.

The evolution of Web development

Server-Side Programming

- To counter these problems, Microsoft created higher-level development platforms—first ASP and then ASP.NET.
- Allow developers to program dynamic web pages without worrying about the low-level implementation details.
- **ASP** is a <u>script-based programming language</u> that requires a thorough understanding of HTML and a good deal of painful coding.
- **ASP.NET** <u>is an object-oriented programming</u> model that lets you put together a web page as easily as you would build a Windows application.
- Compared to classic ASP, **ASP.NET offers better performance, better design tools, and a rich set of ready-made features**.

The evolution of Web development

Server-Side Programming

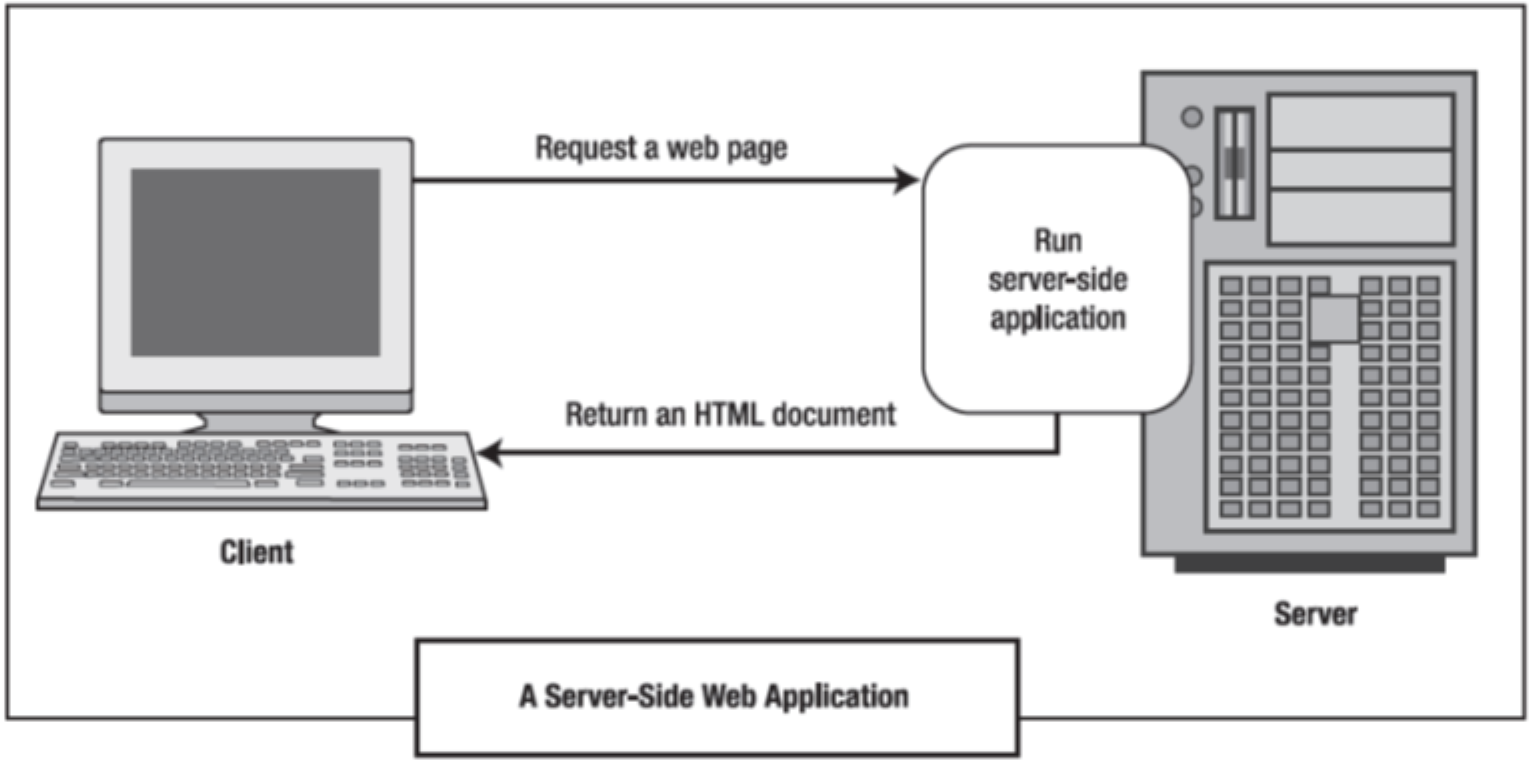


Figure 1.3 the server-side model.

The evolution of Web development

Client-Side Programming

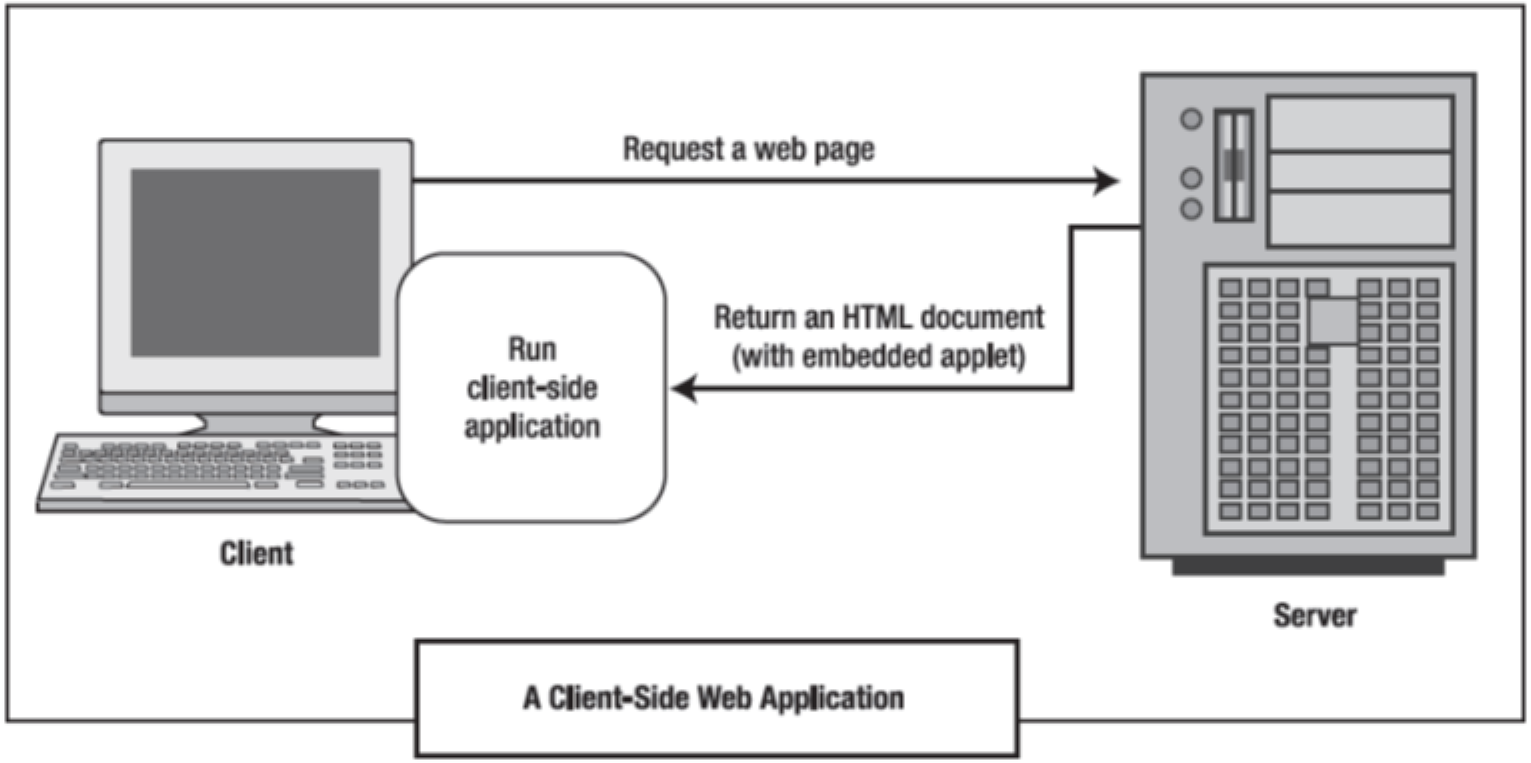

Request a web page
Run client-side application
Return an HTML document (with embedded applet)
Client
Server
A Client-Side Web Application

Figure 1.4 the server-side model.

The evolution of Web development

Client-Side Programming

- The **Client-side technologies** <u>don't involve any server processing</u>.
- Instead, the **complete application is downloaded to the client browser**, which executes it locally.

<u>Advantage:</u>

- **Web applications don't require setup CDs, downloads, client-side configuration, and other tedious (and error-prone) deployment steps**.
- Instead, a web application can be used on any computer that has Internet access.

<u>Drawback:</u>

- Client-side technologies aren't supported equally by all browsers and operating systems. Cross-browser compatibility becomes a problem.

The evolution of Web development

Client-Side Programming

These are some other reasons for avoiding client-side programming:

- ***Isolation***: Client-side code can't access server-side resources. For example, a client-side application has no easy way to read a file or interact with a database on the server.
- ***Security***: End users can view client-side code. And once malicious users understand how an application works, they can often tamper with it.
- ***Thin clients***: In today's world, web-enabled devices such as mobile phones, handheld computers, and personal digital assistants (PDAs) usually have some sort of <u>built-in web browsing ability</u>, but <u>they don't support all the features</u> of traditional desktop-based browsers.

○ For example, thin clients might not support client-side features such as JavaScript and Flash.

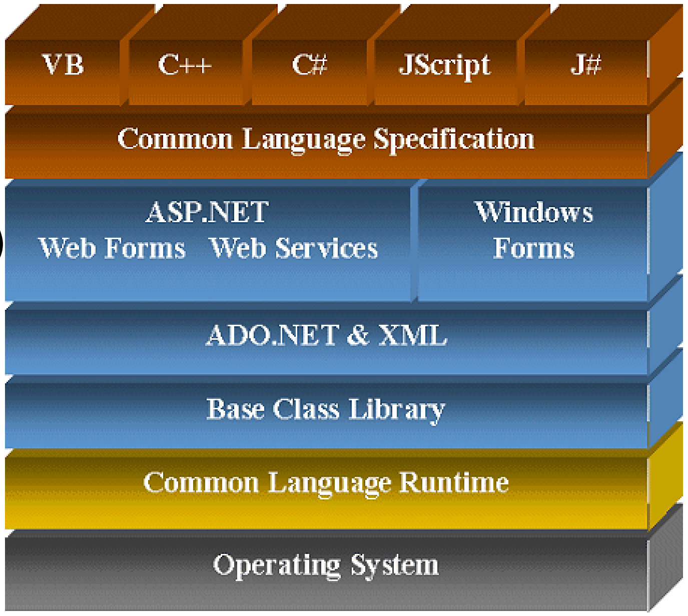
The evolution of Web development

Client-Side Programming

- In many cases, ASP.NET allows you to **combine the best of client-side programming with server-side programming.**
- For example, the best ASP.NET controls can intelligently detect the features of the client browser.
- If the browser supports JavaScript, these controls will return a web page that incorporates JavaScript for a richer, more responsive user interface.

The .NET Framework

- .NET Framework is really a cluster of several technologies:
  - *The .NET languages*
  - *The Common Language Runtime (CLR)*
  - *The .NET Framework class library*
  - *ASP.NET*
  - *Visual Studio*



The .NET Framework

VB, C#, and the .NET Languages

- VB and C# are actually quite similar.
- Though the syntax is different, both VB and C# use the .NET class library and are supported by the CLR.
- In fact, almost any block of C# code can be translated, line by line, into an equivalent block of VB code (and vice versa).

The .NET Framework

Intermediate Language

- All the .NET languages are compiled into another lower-level language before the code is executed. This lower-level language is the Common Intermediate Language (CIL, or just IL).
- The CLR, the engine of .NET, uses only IL code.
- Because all .NET languages are designed based on IL, they all have profound similarities. In fact, the languages are so compatible that a web page written with C# can use a VB component in the same way it uses a C# component, and vice versa.
- The .NET Framework formalizes this compatibility with something called the **Common Language Specification (CLS).**
- Essentially, the CLS is a contract that, if respected, guarantees that a component written in one .NET language can be used in all the others.

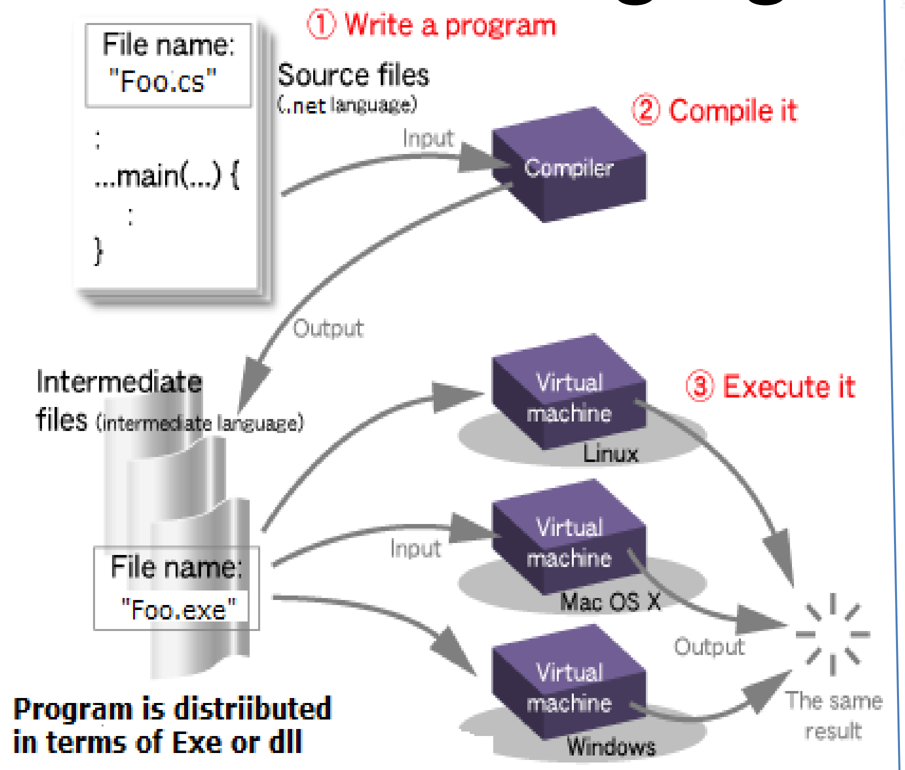
The .NET Framework

Intermediate Language

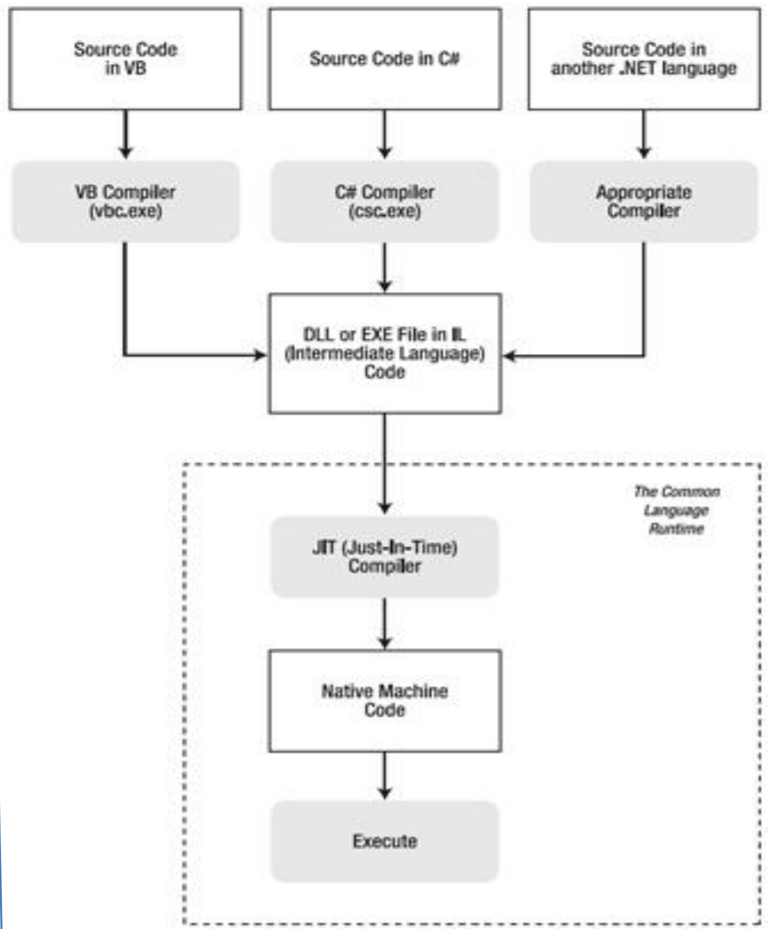
- One part of the CLS is the **common type system (CTS)**.
- Defines the rules for data types such as strings, numbers, and arrays that are shared in all .NET languages.
- The CLS also defines object-oriented ingredients such as classes, methods, events, and quite a bit more.
- Figure 1-5 shows how the .NET languages are compiled to IL. Every EXE or DLL file that you build with a .NET language contains IL code.
- This is the file you deploy to other computers. In the case of a web application, you deploy your compiled code to a live web server.

The .NET Framework

Intermediate Language



Program is distriibuted in terms of Exe or dll



Source Code in VB
Source Code in C#
Source Code in another .NET language
VB Compiler (vbc.exe)
C# Compiler (csc.exe)
Appropriate Compiler
DLL or EXE File in IL (Intermediate Language) Code
The Common Language Runtime
JIT (Just-In-Time) Compiler
Native Machine Code
Execute

Figure 1-5. *Language compilation in .NET*

The .NET Framework

The Common Language Runtime(CLR)

- The <u>CLR runs only IL code</u>, which means it has no idea which .NET language you originally used.

- Notice, however, that the CLR actually performs another compilation <u>step—it takes the IL code and transforms it to native machine language</u> <u>code</u> that's appropriate for the current platform.

- This step occurs when the application is launched, just before the code is actually executed.

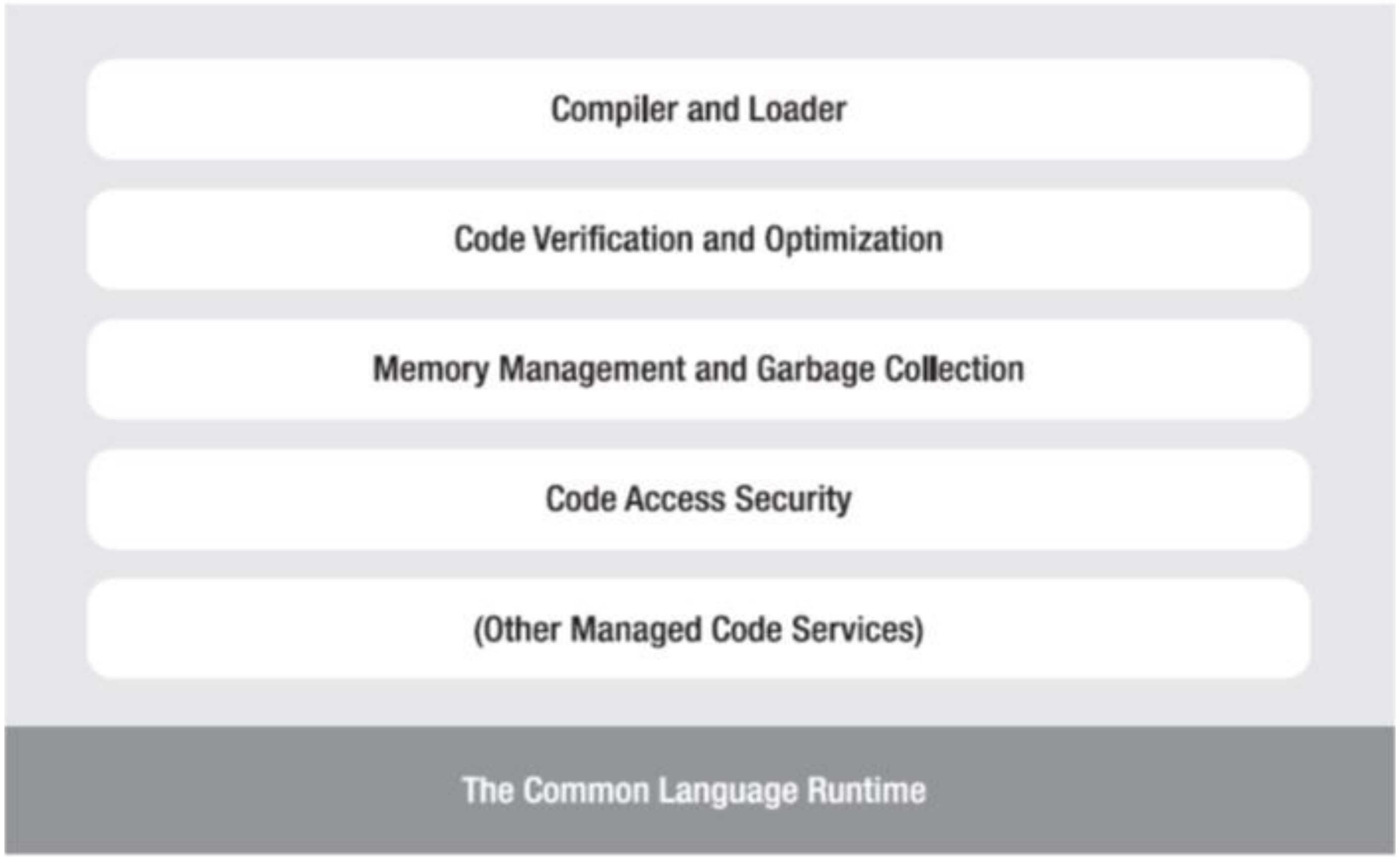
The .NET Framework

The Common Language Runtime(CLR)

- The CLR is the engine that supports all the .NET languages.
- All <u>.NET code runs inside the CLR</u>. For example, when a client requests an ASP.NET web page, the ASP.NET service runs inside the CLR environment, executes your code, and creates a final HTML page to send to the client.
- Not only does the CLR execute code, but it also provides a whole set of related services such as <u>code verification, optimization, and object management</u>.

The .NET Framework

The Common Language Runtime(CLR)


Compiler and Loader
Code Verification and Optimization
Memory Management and Garbage Collection
Code Access Security
(Other Managed Code Services)
The Common Language Runtime

The .NET Framework

The Common Language Runtime(CLR)

- **The implications of the CLR are wide-ranging:**
  - ***Deep language integration*:** VB and C#, like all .NET languages, compile to IL. In other words, the CLR makes no distinction between different languages—in fact, it has no way of knowing what language was used to create an executable.
  - ***Side-by-side execution*:** The CLR also has the ability to load more than one version of a component at a time. In other words, you can update a component many times, and the correct version will be loaded and used for each application.
  - ***Fewer errors*:** Whole categories of errors are impossible with the CLR. For example, the CLR prevents many memory mistakes that are possible with lower-level languages such as C++.

The .NET Framework

The Common Language Runtime(CLR)

<u>Drawbacks of CLR</u>

- ***<u>Performance</u>*:** A typical ASP.NET application is much faster than a comparable ASP application, because ASP.NET code is compiled to machine code before it's executed. However, processor crunching algorithms still can't match the blinding speed of well-written C++ code, because the CLR imposes some additional overhead. With ASP.NET caching and some well written database code, you can ensure excellent performance for any web application.

- ***<u>Code transparency</u>*:** IL is much easier to disassemble, meaning that if you distribute a compiled application or component, other programmers may have an easier time determining how your code works. This isn't much of an issue for ASP.NET applications, which aren't distributed but are hosted on a secure web server.

- ***<u>Questionable cross-platform support</u>*:** Thanks to the Mono project (see ttp://www.monoproject. com), developers can use a free implementation of .NET that runs on Linux, Unix, Mac OS, and Windows. However, Mono is supported by the open source community, not Microsoft, and it doesn't provide all of .NET's features. For this reason, few businesses use Mono instead of .NET.

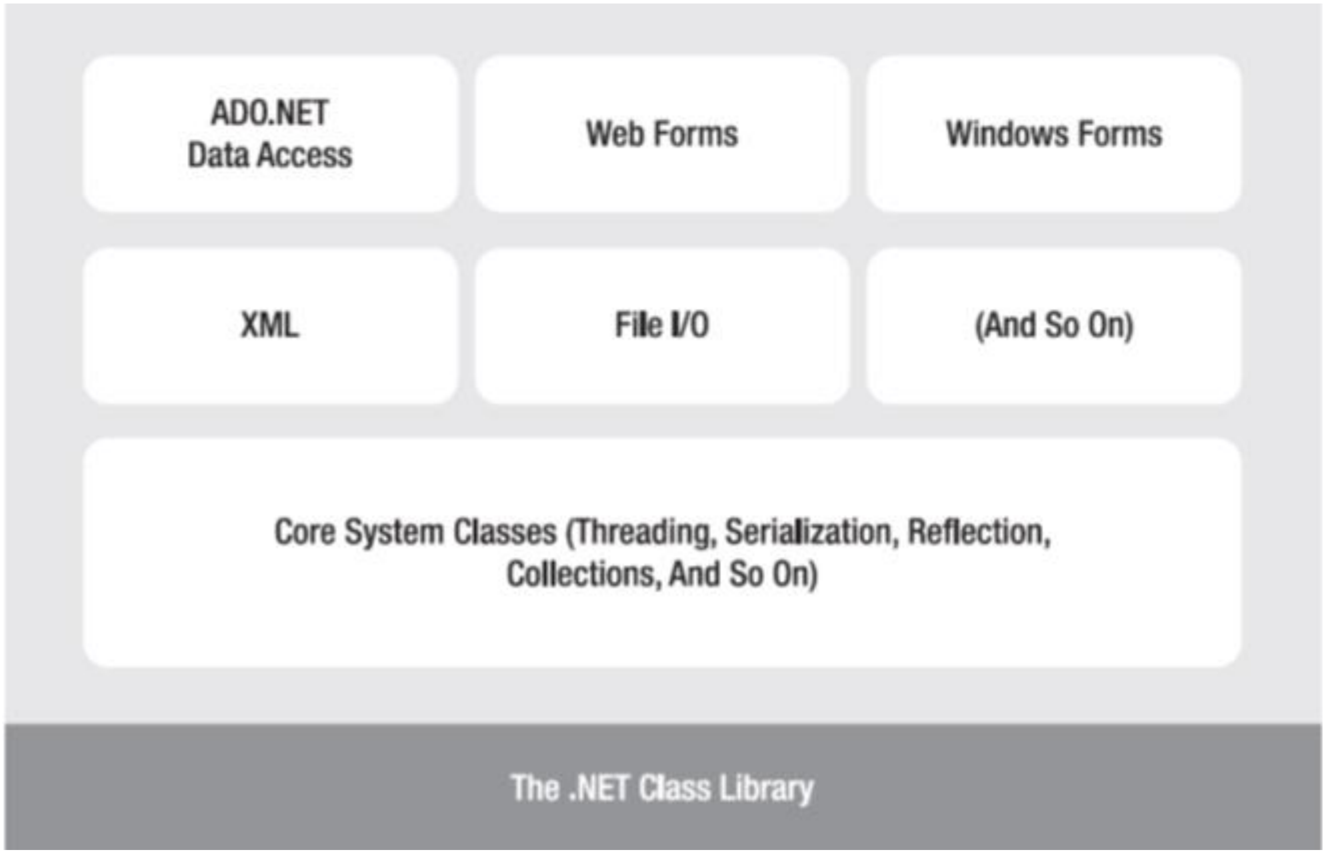
The .NET Framework

The .NET Class Library

- **The .NET class library** is a giant repository of classes that provide prefabricated functionality for everything from reading an XML file to sending an e-mail message.
- Any .NET language can use the .NET class library's features by interacting with the right objects. This helps encourage consistency among different .NET languages and removes the need to install numerous components on your computer or web server.
  - Some parts of the class library include features you'll never need to use in web applications.
  - Other parts of the class library are targeted directly at web development.
  - .NET Class Library also include the base set of classes that define common variable types and the classes for data access, to name just a few.

The .NET Framework

The .NET Class Library



The .NET Framework

ASP.NET

- This is the engine that hosts the web applications you create with .NET.
- Supports almost any feature from the .NET Framework class library.
- ASP.NET also includes a set of web specific services, such as secure authentication and data storage.

The .NET Framework

Visual Studio

- The last part of .NET is the **Visual Studio** development tool, which provides a rich environment where you can rapidly create advanced applications.
- Visual Studio includes the complete .NET Framework, so you won't need to download it separately.
- Although in theory you could create an ASP.NET application without Visual Studio (for example, by writing all the source code in a text editor and compiling it with .NET's command-line compilers), this task would be tedious, painful, and prone to error.

The .NET Framework

Visual Studio

Some of the features of Visual Studio include the following:

- ***Page design*:** You can create an attractive page with drag-and-drop ease using Visual Studio's integrated web form designer. You don't need to understand HTML.
- ***Automatic error detection*:** Potential problems are underlined, just like the "spell-as-you-go" feature found in many word processors.
- ***Debugging tools*:** Visual Studio retains its legendary debugging tools, which allow you to watch your code in action and track the contents of variables. Visual Studio has a built-in web server that works just for debugging.
- ***IntelliSense*:** Visual Studio provides statement completion for recognized objects and automatically lists information such as function parameters in helpful tooltips.

END OF LECTURE