.NET Fundamentals

NYU

School of Continuing & Professional Studies
Division of Programs in Business

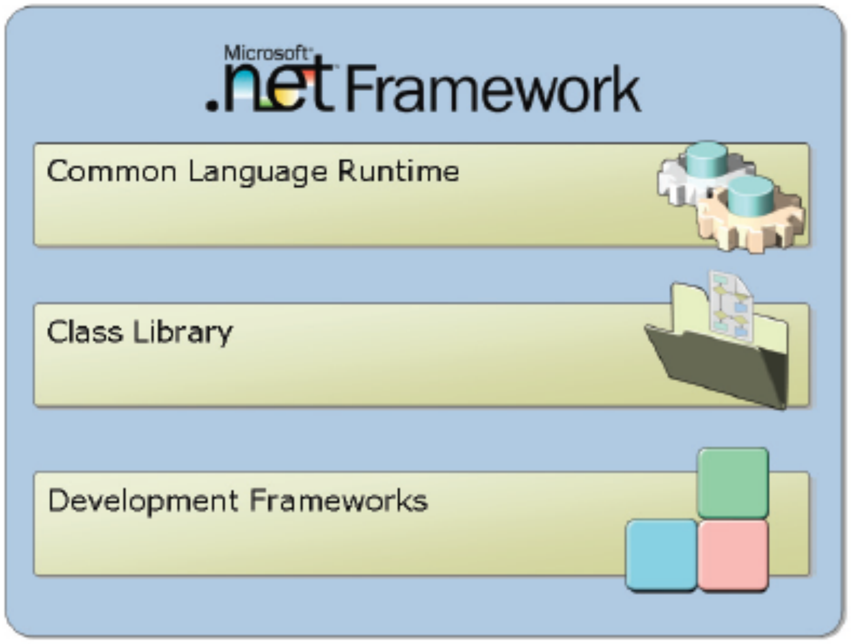
Keith R. Harris (kh83@nyu.edu)

Agenda

- **Introduction**
  - About me
  - About you
- **Review Syllabus**
- **Session 1**
- **Lab 1**

What is the .NET Framework?

- **Comprehensive development platform**
- **Comprises:**

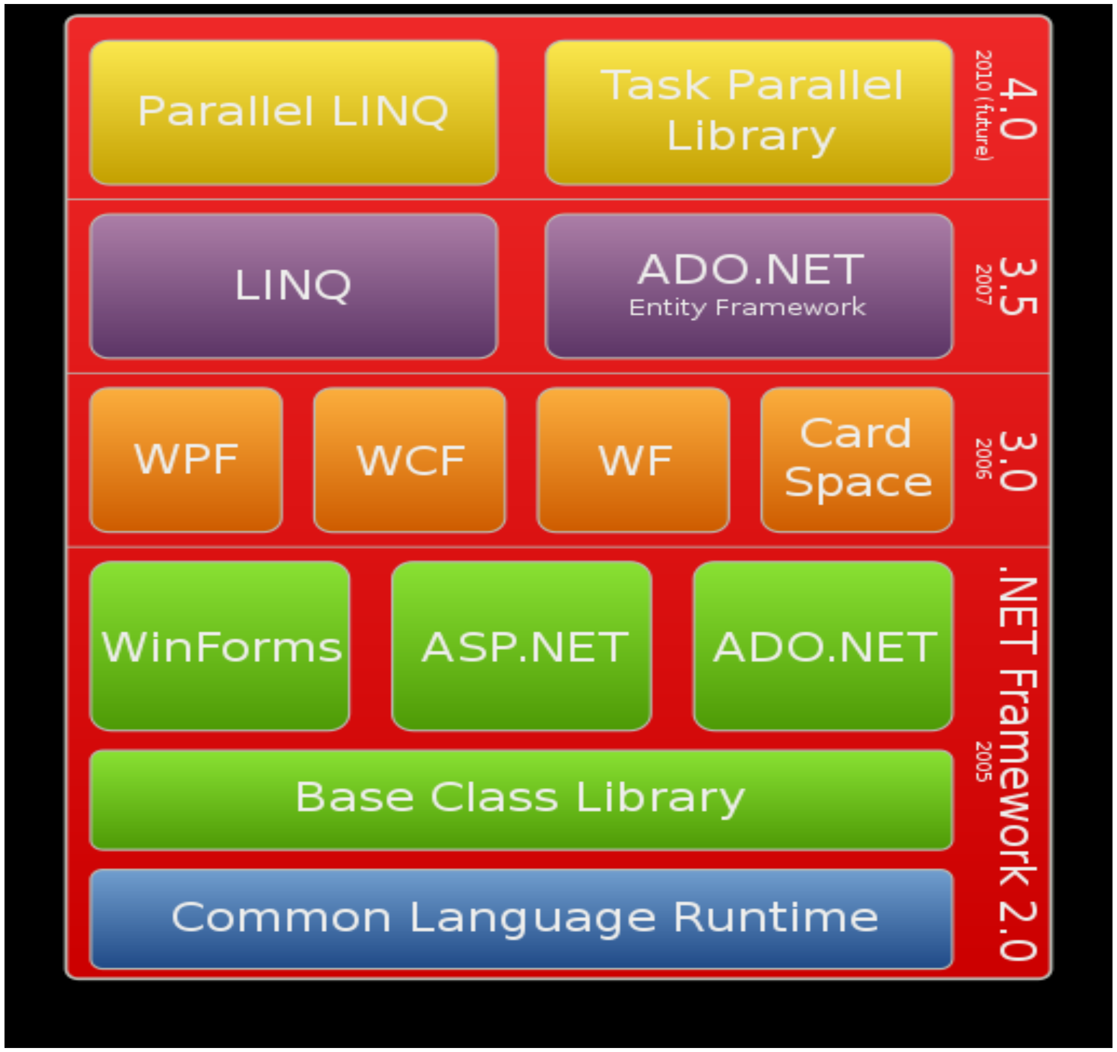


- Manages execution & provides common services such as memory management, transactions, inter-process communications, exception handling, multi-threading, etc…
- Library of reusable classes used to build applications
- Provide the necessary components and infrastructure used to build different types of applications

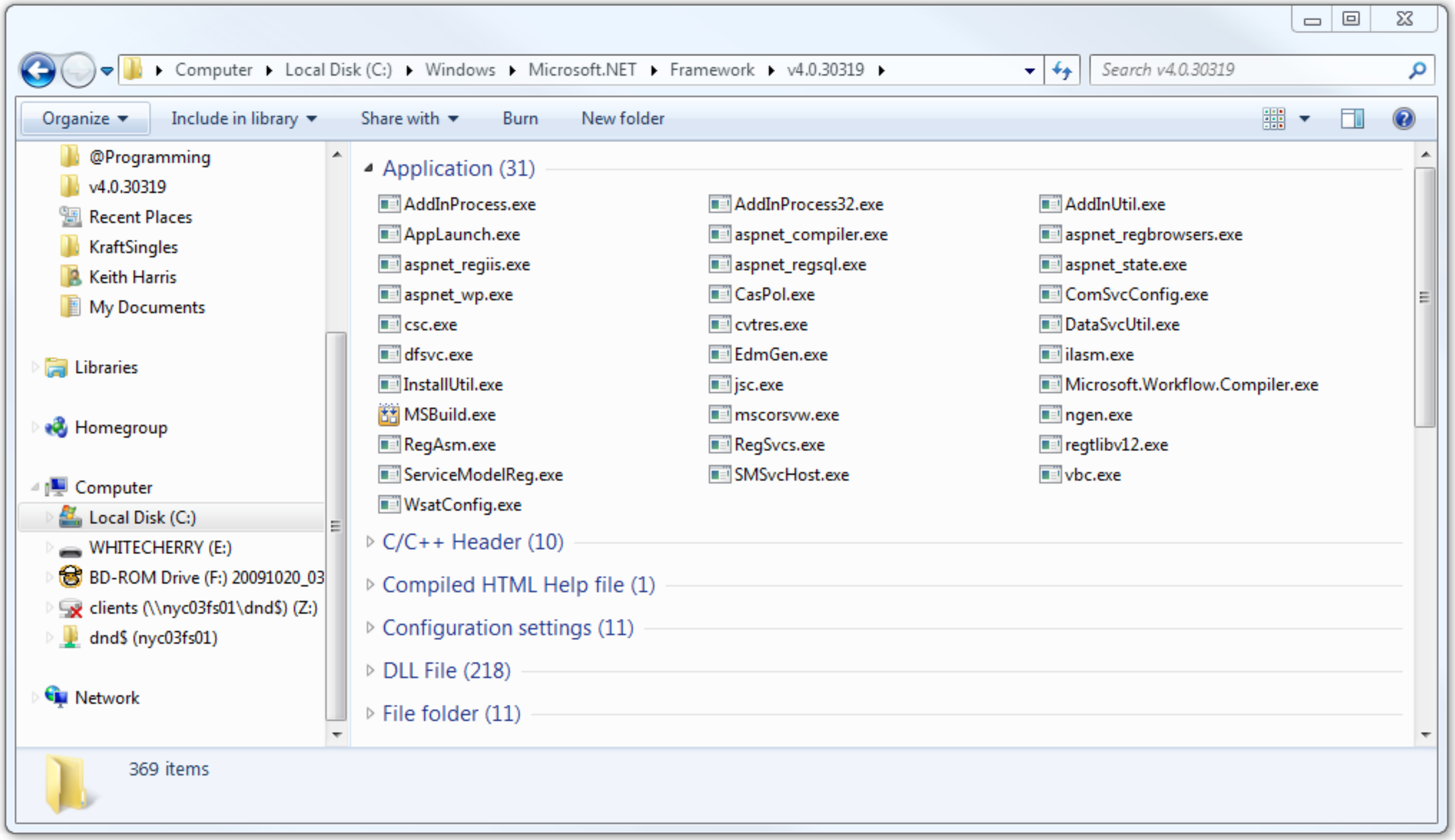
.NET Runtime Versions

Version Name	Release Date
1.0 RTM	01/05/2002
1.0 SP1	03/19/2002
1.0 SP2	07/08/2002
1.0 SP3	08/31/2004
1.1 RTM	04/01/2003
1.1 SP1	08/30/2004
1.1 SP1 (Windows Server 2003 Version)	03/30/2005
2.0 RTM	11/07/2005
3.0 RTM	11/06/2006
3.0 RTM (Vista)	01/30/2007
3.0 SP1	11/19/2007
3.5 RTM	11/19/2007
4.0 RTM	04/12/2010

.NET Framework Stack



.NET Framework



Application (31)

AddInProcess.exe	AddInProcess32.exe	AddInUtil.exe
AppLaunch.exe	aspnet_compiler.exe	aspnet_regbrowsers.exe
aspnet_regiis.exe	aspnet_regsql.exe	aspnet_state.exe
aspnet_wp.exe	CasPol.exe	ComSvcConfig.exe
csc.exe	cvtres.exe	DataSvcUtil.exe
dfsvc.exe	EdmGen.exe	ilasm.exe
InstallUtil.exe	jsc.exe	Microsoft.Workflow.Compiler.exe
MSBuild.exe	mscorsvw.exe	ngen.exe
RegAsm.exe	RegSvcs.exe	regtlibv12.exe
ServiceModelReg.exe	SMSvcHost.exe	vbc.exe
WsatConfig.exe		

C/C++ Header (10)

Compiled HTML Help file (1)

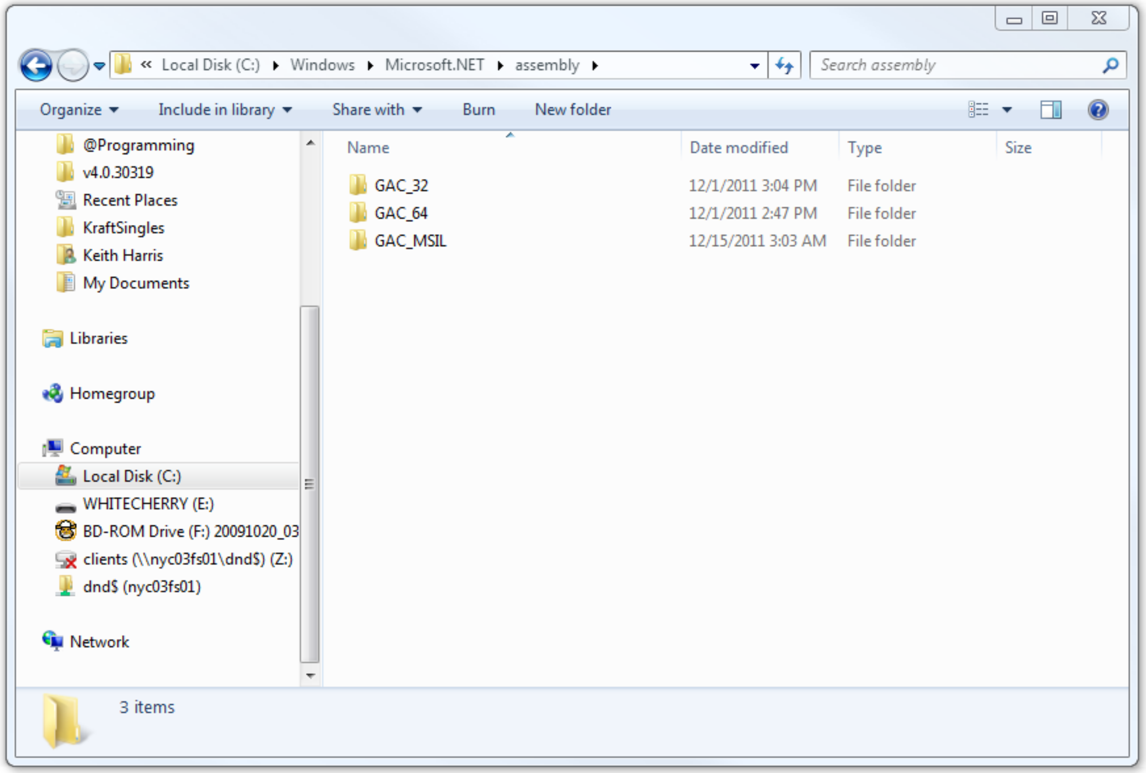
Configuration settings (11)

DLL File (218)

File folder (11)

Global Access Cache (GAC)

- **Machine-wide, shared library**
- **Use `gacutil.exe` to install/remove**
- **Different versions of same assembly can live side-by-side**

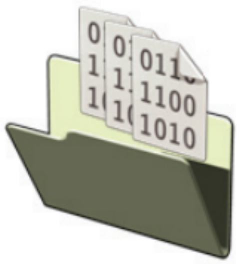


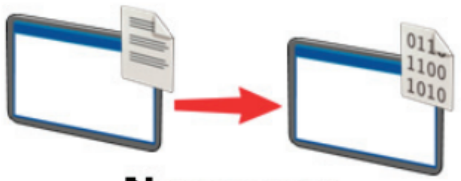
Name	Date modified	Type	Size
GAC_32	12/1/2011 3:04 PM	File folder	
GAC_64	12/1/2011 2:47 PM	File folder	
GAC_MSIL	12/15/2011 3:03 AM	File folder	

.NET Framework Tools

Caspol.exe

Makecert.exe

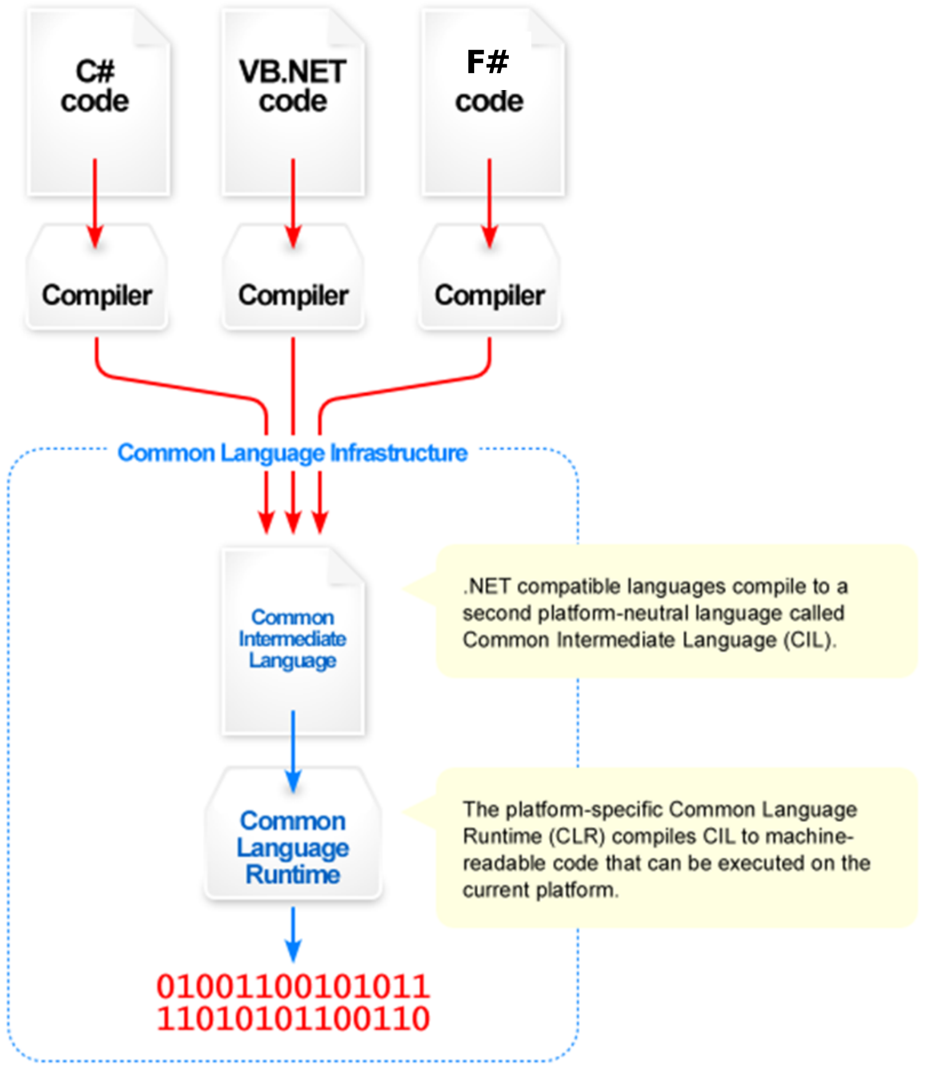

Gacutil.exe


Ngen.exe

Ildasm.exe

Sn.exe

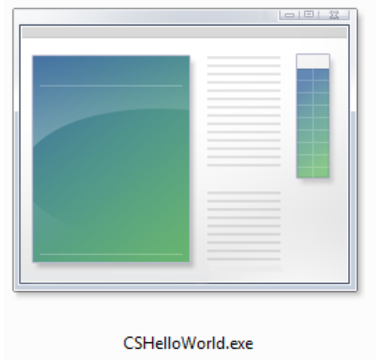
.NET Code Lifecycle



.NET compatible languages compile to a second platform-neutral language called Common Intermediate Language (CIL).

The platform-specific Common Language Runtime (CLR) compiles CIL to machine-readable code that can be executed on the current platform.

Compile



```
class Program
{
    static void Main(string[] args)
    {
        System.Console.WriteLine("CS says 'Hello World!'");
    }
}
```

Source Code

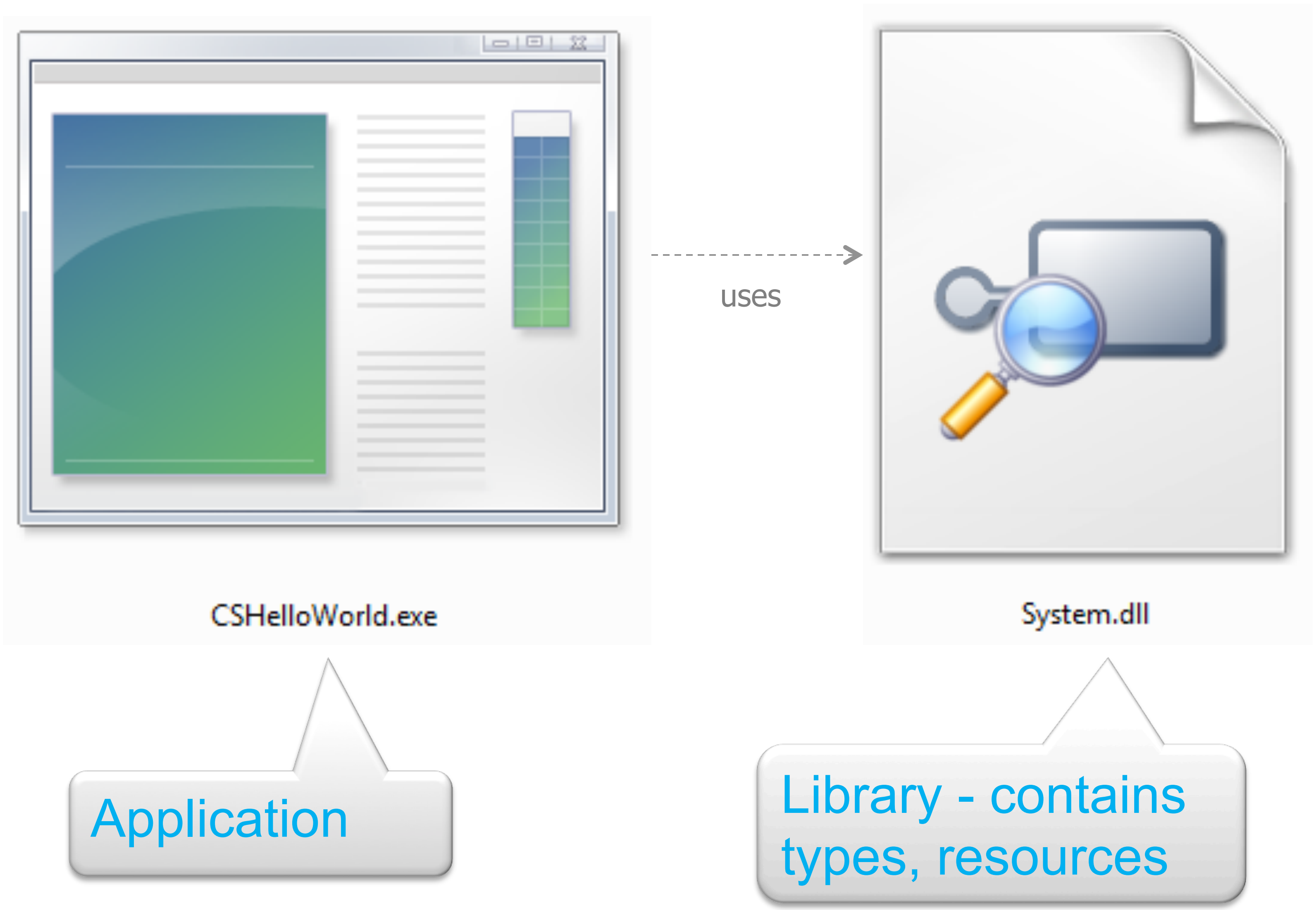
```
.method private hidebysig static
    void Main (
        string[] args
    ) cil managed
{
    // Method begins at RVA 0x2050
    // Code size 13 (0xd)
    .maxstack 8
    .entrypoint

    IL_0000: nop
    IL_0001: ldstr "CS says 'Hello World!'"
    IL_0006: call void [mscorlib]System.Console::WriteLine(string)
    IL_000b: nop
    IL_000c: ret
}
```

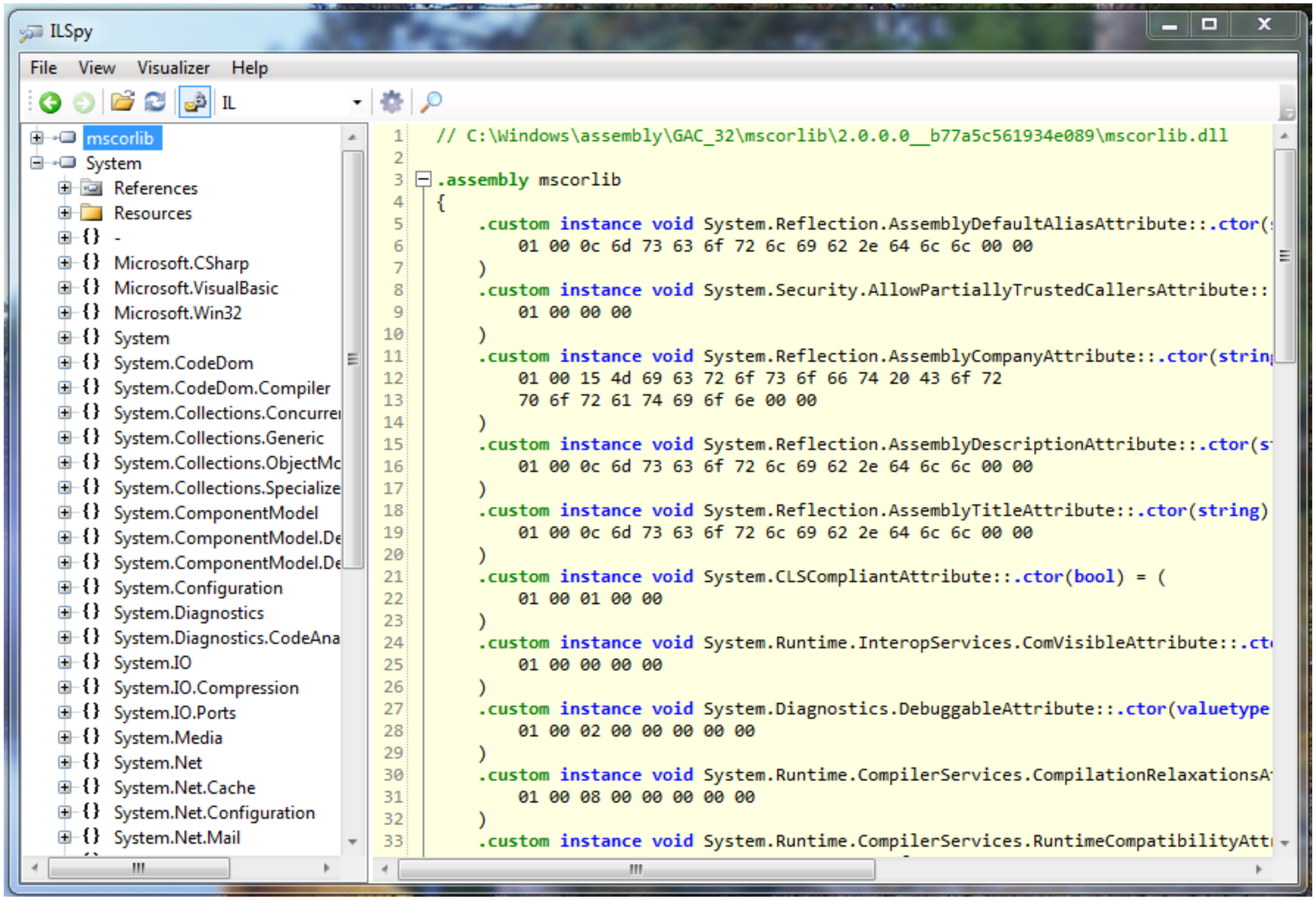
.NET Assembly

***IL code** to be run by the CLR*

.NET Assemblies



ILSpy - .NET Assembly Browser

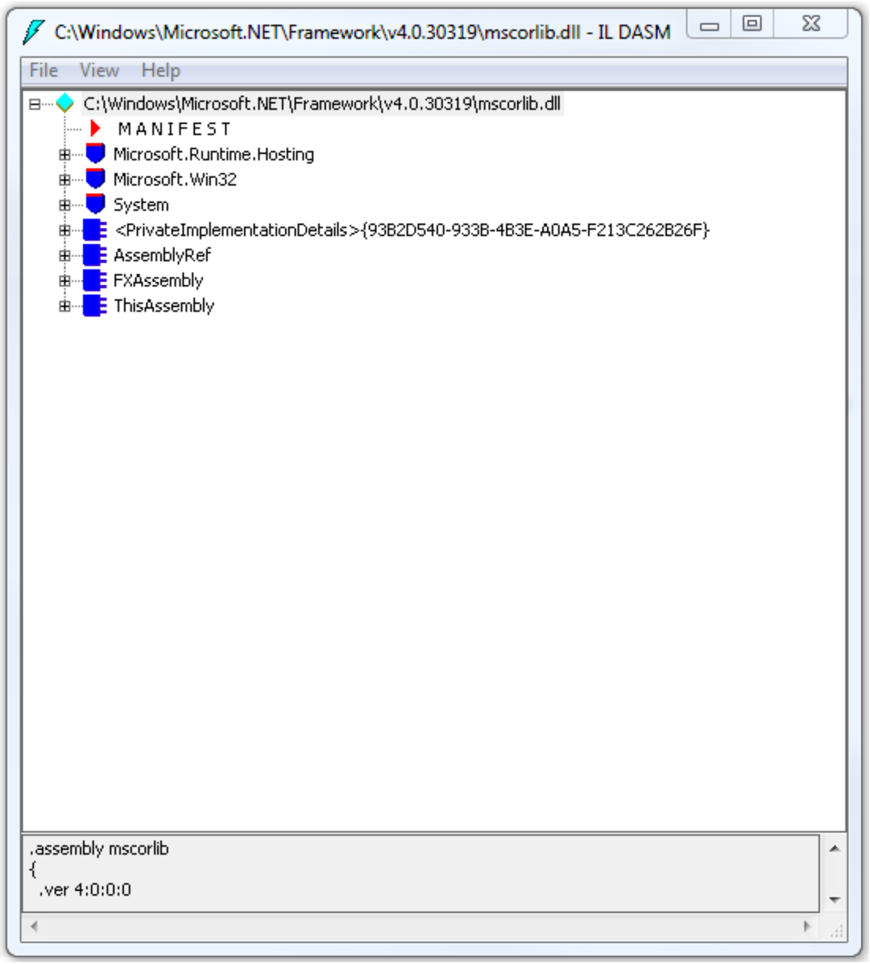


```
// C:\Windows\assembly\GAC_32\mscorlib\2.0.0.0__b77a5c561934e089\mscorlib.dll

.assembly mscorlib
{
    .custom instance void System.Reflection.AssemblyDefaultAliasAttribute::.ctor(
        01 00 0c 6d 73 63 6f 72 6c 69 62 2e 64 6c 6c 00 00
    )
    .custom instance void System.Security.AllowPartiallyTrustedCallersAttribute::
        01 00 00 00
    )
    .custom instance void System.Reflection.AssemblyCompanyAttribute::.ctor(strin
        01 00 15 4d 69 63 72 6f 73 6f 66 74 20 43 6f 72
        70 6f 72 61 74 69 6f 6e 00 00
    )
    .custom instance void System.Reflection.AssemblyDescriptionAttribute::.ctor(s
        01 00 0c 6d 73 63 6f 72 6c 69 62 2e 64 6c 6c 00 00
    )
    .custom instance void System.Reflection.AssemblyTitleAttribute::.ctor(string)
        01 00 0c 6d 73 63 6f 72 6c 69 62 2e 64 6c 6c 00 00
    )
    .custom instance void System.CLSCompliantAttribute::.ctor(bool) = (
        01 00 01 00 00
    )
    .custom instance void System.Runtime.InteropServices.ComVisibleAttribute::.ct
        01 00 00 00 00
    )
    .custom instance void System.Diagnostics.DebuggableAttribute::.ctor(valuetype
        01 00 02 00 00 00 00 00
    )
    .custom instance void System.Runtime.CompilerServices.CompilationRelaxationsA
        01 00 08 00 00 00 00 00
    )
    .custom instance void System.Runtime.CompilerServices.RuntimeCompatibilityAtt
```

ILDASM - .NET Framework Assembly Browser

- **C:\Program Files (x86)\Microsoft SDKs\Windows\v8.0A\bin\NETFX 4.0 Tools**


C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorlib.dll - IL DASM
File View Help

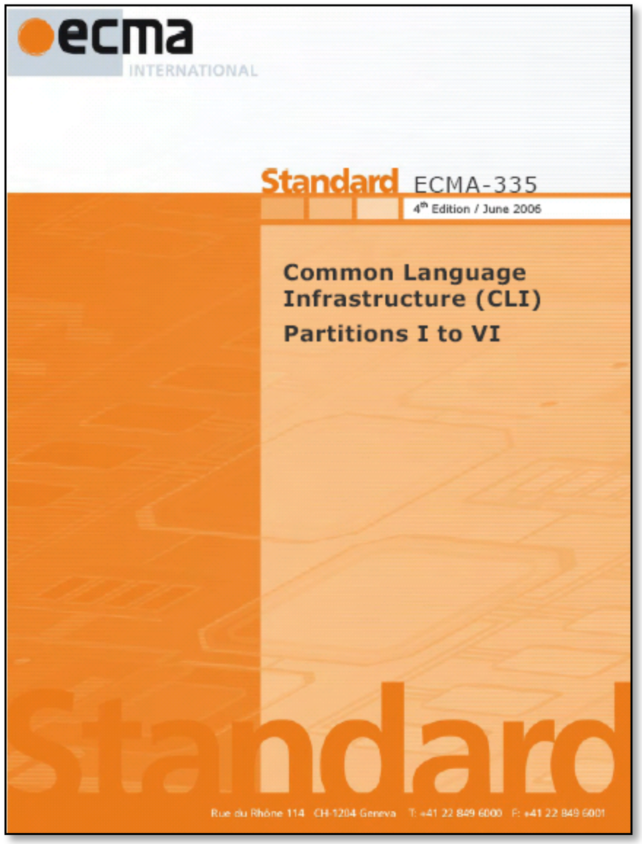
```
C:\Windows\Microsoft.NET\Framework\v4.0.30319\mscorlib.dll
  MANIFEST
  Microsoft.Runtime.Hosting
  Microsoft.Win32
  System
  <PrivateImplementationDetails>{93B2D540-933B-4B3E-A0A5-F213C262B26F}
  AssemblyRef
  FXAssembly
  ThisAssembly
```

```
.assembly mscorlib
{
  .ver 4:0:0:0
```

What is the CLR?

- **Common Language Runtime**
- **Microsoft's implementation of the CLI standard**
- **Generally refers to the runtime engine**
- **Compiles IL (bytecodes) to machine instructions**
- **Provides run-time services such as:**
  - Memory management
  - Thread management
  - Exception handling
  - Garbage collection
  - Security

The CLI Standard


ecma INTERNATIONAL

Standard ECMA-335

4th Edition / June 2006

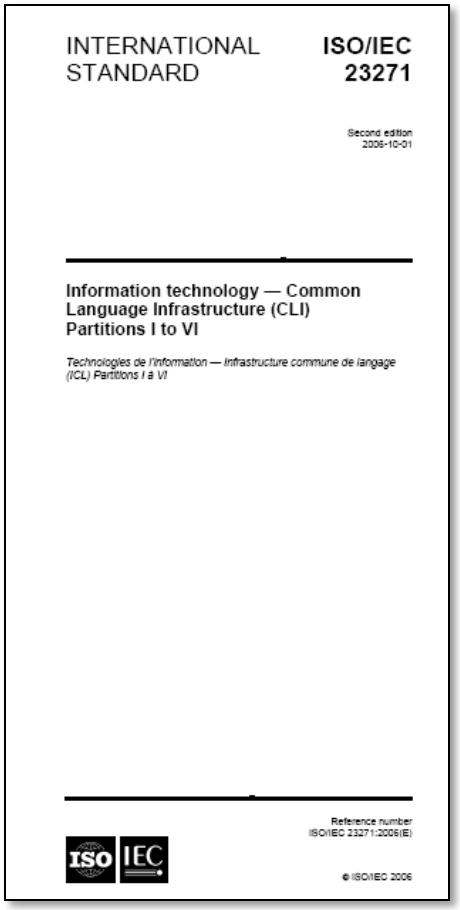
Common Language Infrastructure (CLI)

Partitions I to VI

Standard

Rue du Rhône 114 CH-1204 Geneva T: +41 22 849 6000 F: +41 22 849 6001

12/2001


INTERNATIONAL STANDARD

ISO/IEC 23271

Second edition 2006-10-01

Information technology — Common Language Infrastructure (CLI) Partitions I to VI

Technologies de l'information — Infrastructure commune de langage (ICL) Partitions I à VI

Reference number ISO/IEC 23271:2006(E)

ISO IEC

© ISO/IEC 2006

04/2003

CLI
Common Language Infrastructure

- **Open specification developed by MS**
- **Describes the executable code and runtime environment that form the core and the .NET framework**
- **Also describes:**
  - Common Type System (CTS)
  - Metadata
  - Common Language Specification (CLS)
  - Virtual Execution System (VES)

CLI Implementaions

- **Microsoft:**
  - Shared Source CLI (formerly Rotor)
  - .NET Framework
  - .NET Compact Framework
- **Others:**
  - Mono development platform (open source project)
  - Portable .NET (dotGNU project)

Metadata

- Information about compiled types
- Similar to COM type library
- Enables applications to discover interfaces, classes, types, methods and fields in assembly
- Read using reflection

Common Intermediate Language

- .NET instruction set
- Known as IL (formerly MSIL)
- “Bytecodes”
- Machine independent
- Executed by a Virtual Execution System (part of CLI)

Common Type System (CTS)

- **.NET languages must abide by this**
- **Allows interoperability among languages**
- **Concerns types:**
  - Naming rules
  - Visibility
  - Arrays
  - Casting
  - Value types
  - Object types
  - Hiding, overriding, and layout
  - Method definitions
  - Field definitions

Common Language Specification (CLS)

- **Rules to promote language interoperability**
- **Concerns:**
  - Type names
  - Inheritance
  - Polymorphism
  - Exceptions
  - Operators
  - Operator overloading
  - Nested types
  - Custom attributes
  - Abstract and virtual methods

Base Class Library (BCL)

- Standard library available to all languages of the framework
- Provides fundamental functionality
- Updated with each new release of .NET
- Contains CLI standard namespaces
- Contains non standard namespaces

Framework Class Libraries (FCL)

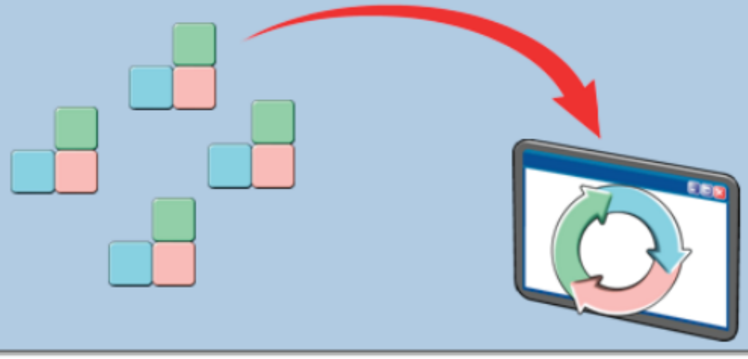
- **Includes BCL**
- **Superset of BCL**
- **Includes Microsoft.* namespace**
- **Contains many common functions such as:**
  - Database interaction
  - XML manipulation
  - Graphics rendering
  - Windows applications
  - Web pages

How the CLR Executes Code

Assemblies contain ~~MSIL~~ code, which is not actually executable

The CLR loads the ~~MSIL~~ code from an assembly and converts it into the machine code that the computer requires

1. Loads assemblies that the application references
2. Verifies and compiles assemblies into machine code
3. Runs the executable assembly



Why C#?

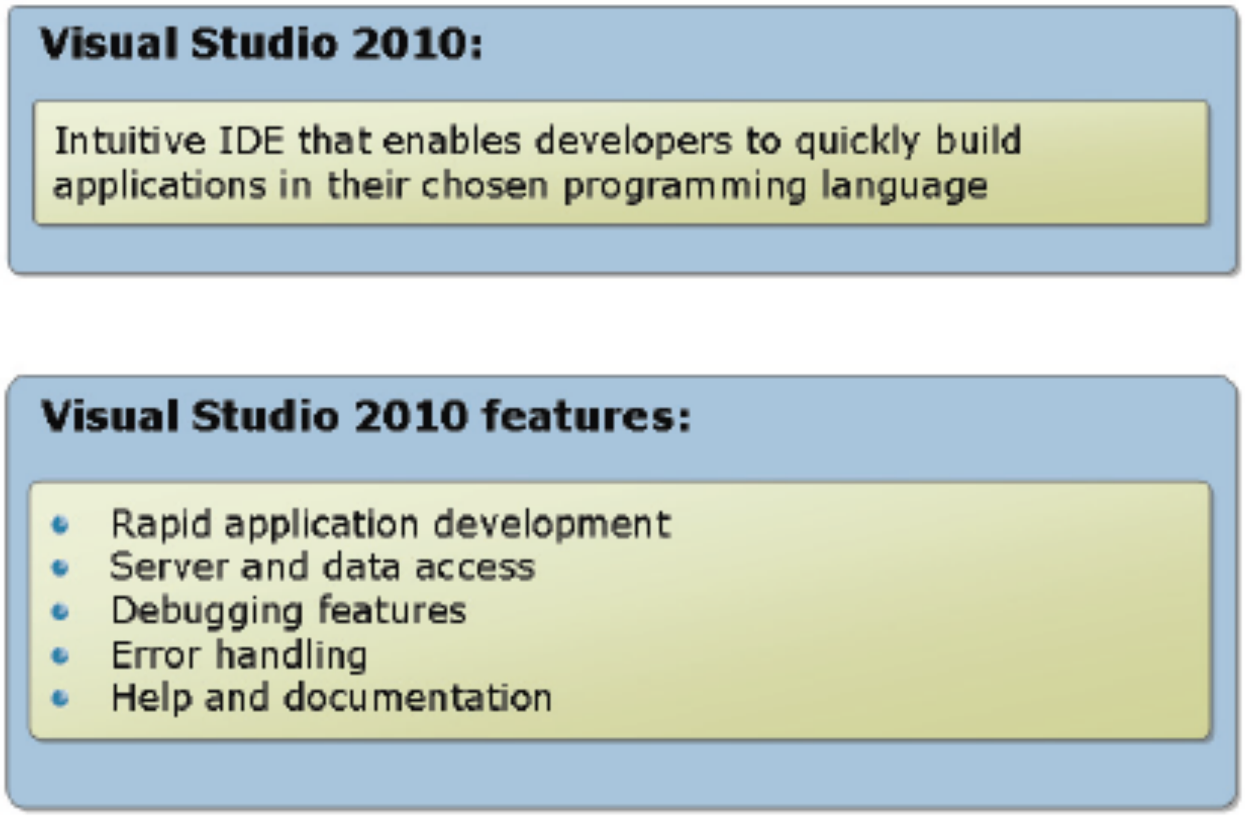
C#

C# is the language of choice for many developers who build .NET Framework applications

C# uses a very similar syntax to C, C++, and Java

C# has been standardized and is described by the ECMA-334 C# Language Specification

Key Features of Visual Studio



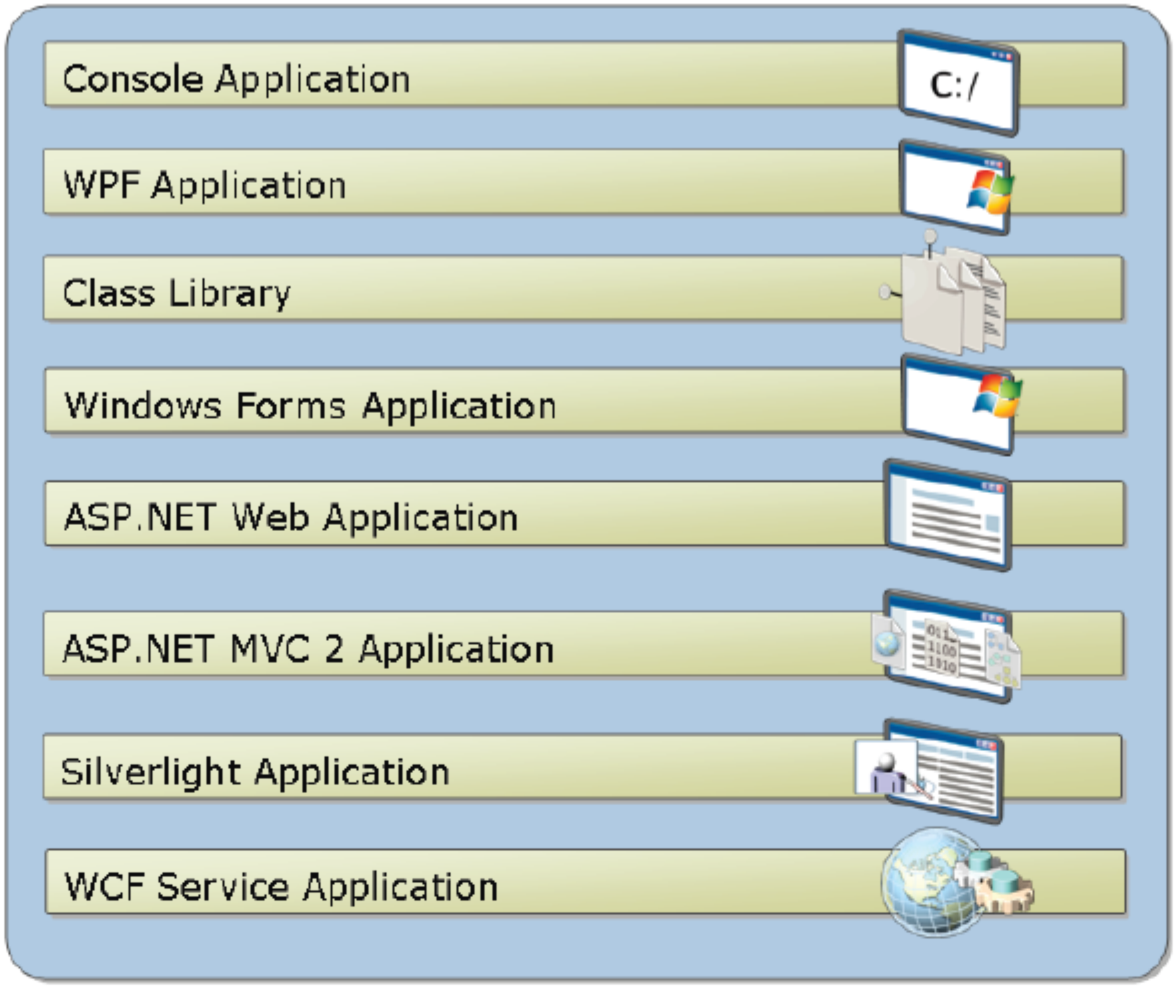
Visual Studio 2010:

Intuitive IDE that enables developers to quickly build applications in their chosen programming language

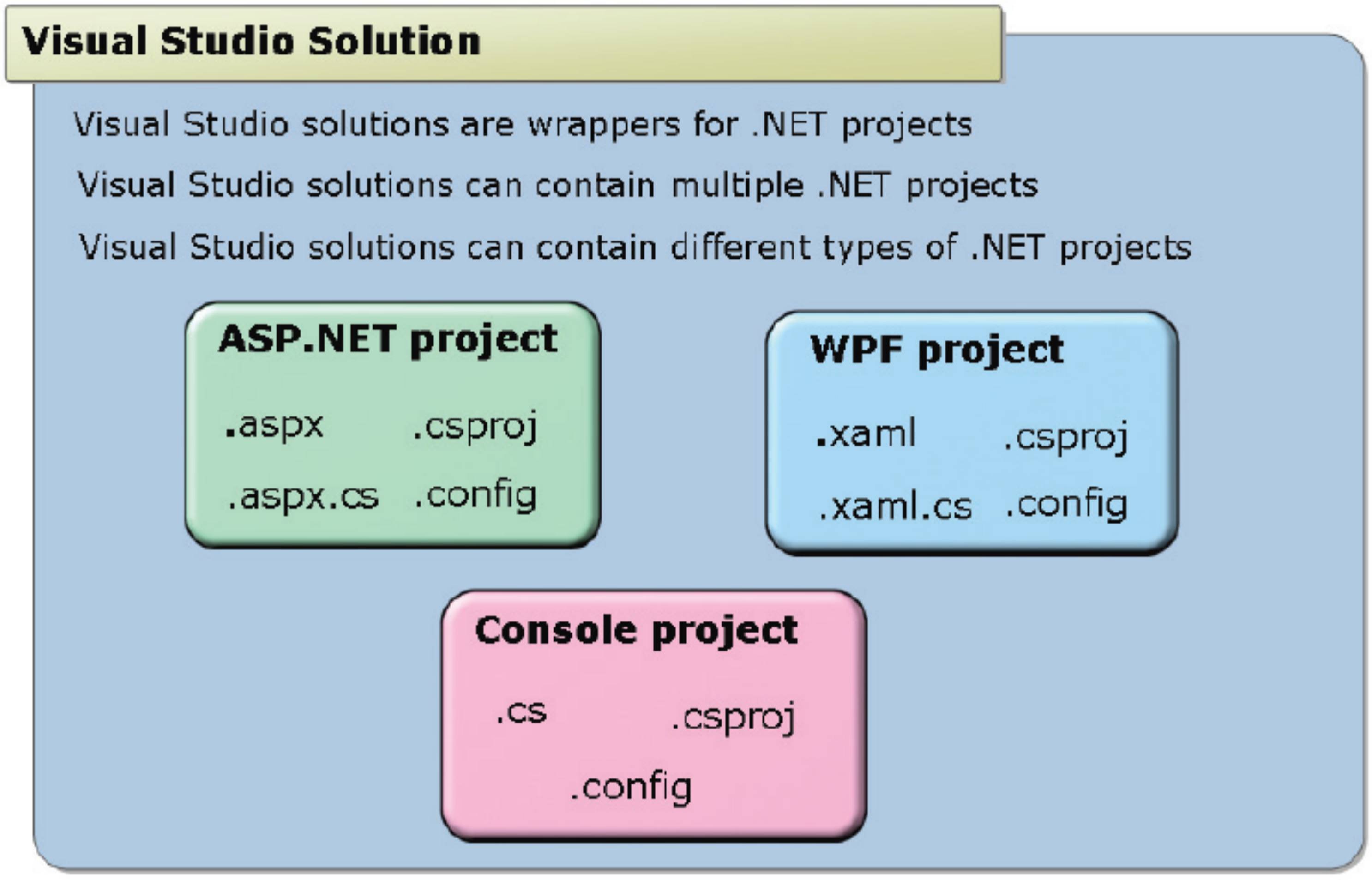
Visual Studio 2010 features:

- Rapid application development
- Server and data access
- Debugging features
- Error handling
- Help and documentation

Project Templates in Visual Studio



Structure of VS Projects and Solutions

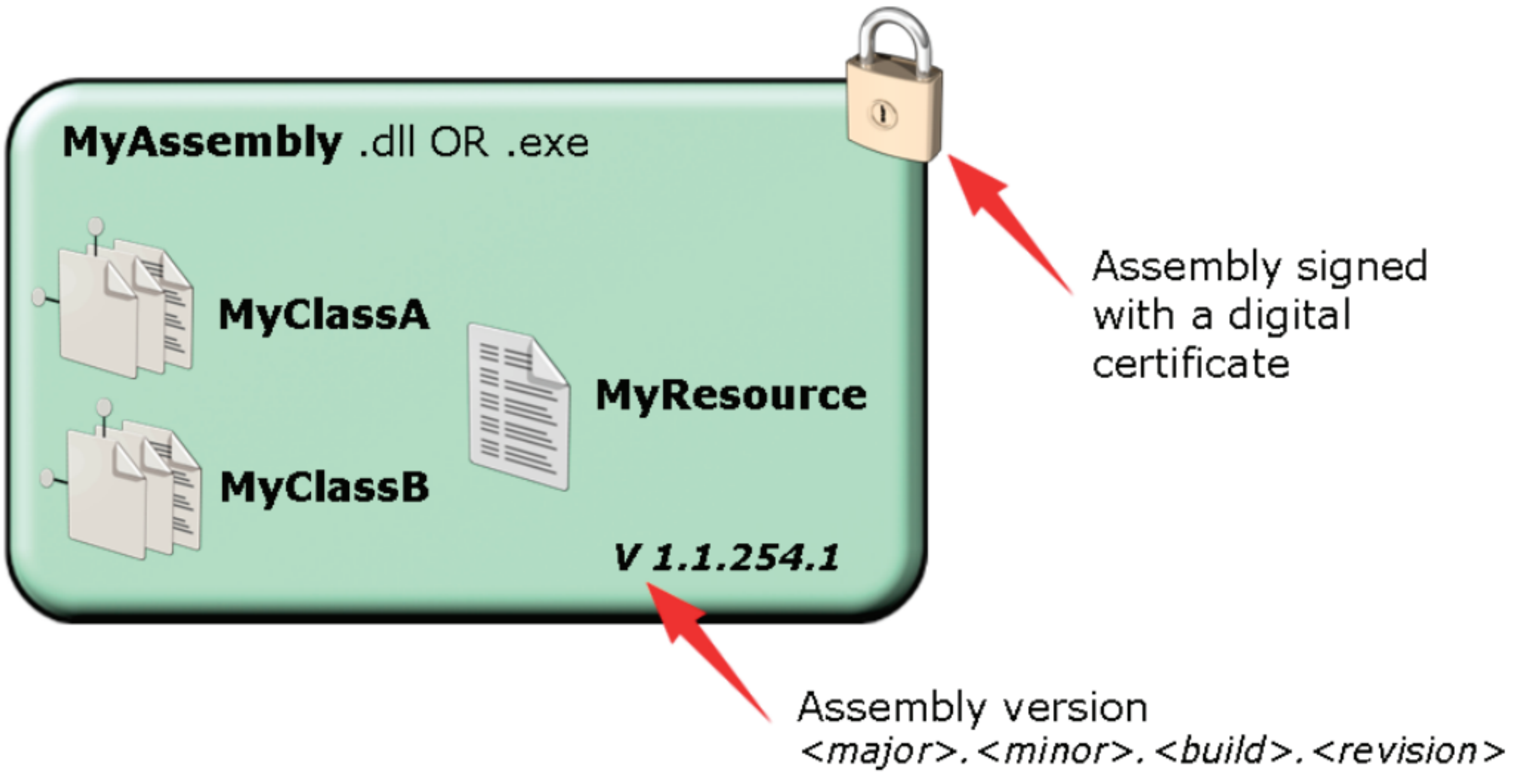


Visual Studio Solution
Visual Studio solutions are wrappers for .NET projects
Visual Studio solutions can contain multiple .NET projects
Visual Studio solutions can contain different types of .NET projects
ASP.NET project
.aspx
.csproj
.aspx.cs
.config
WPF project
.xaml
.csproj
.xaml.cs
.config
Console project
.cs
.csproj
.config

Assembly

Building blocks of .NET Framework applications

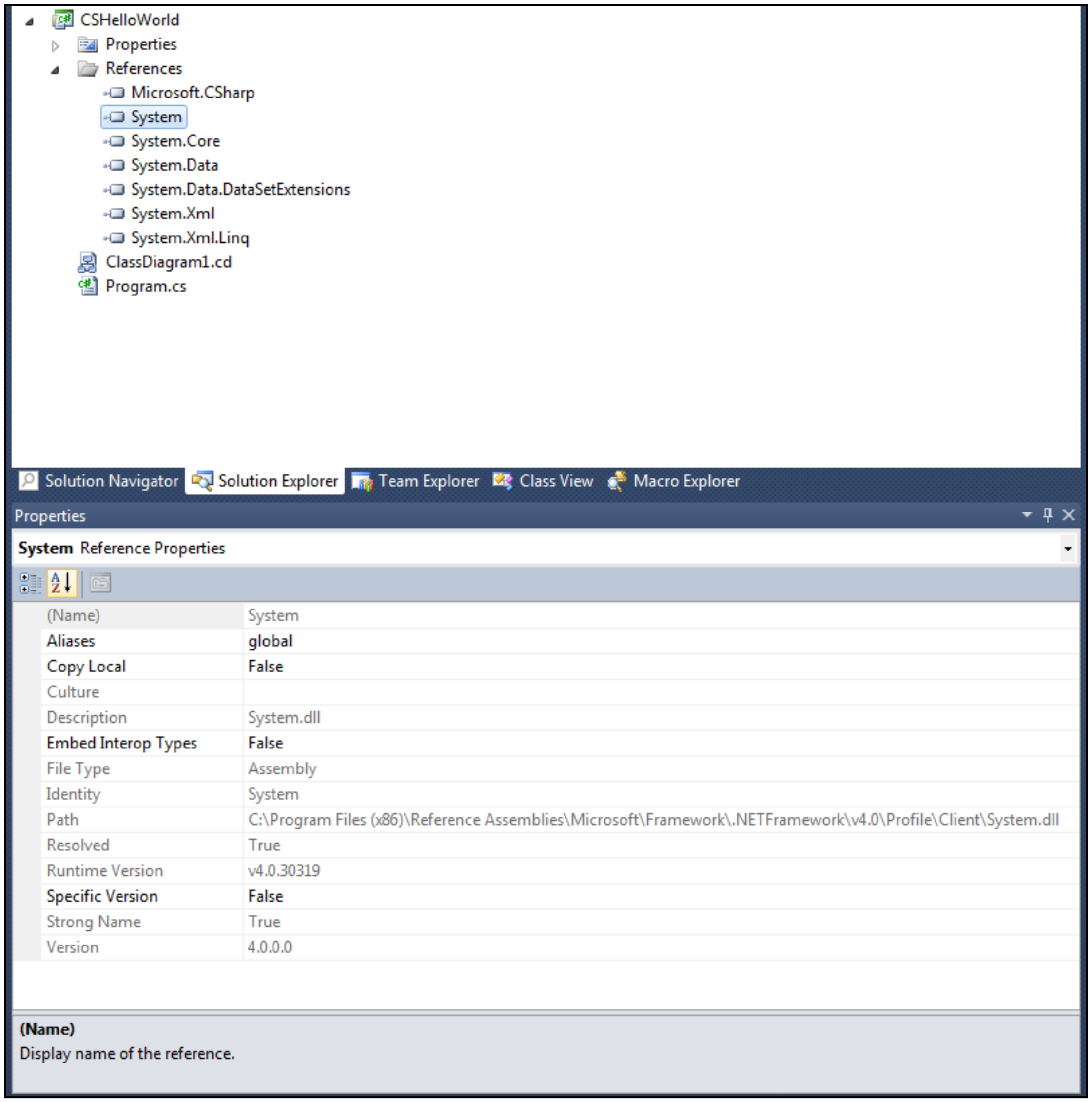
Collection of types and resources that form a logical unit of functionality


MyAssembly .dll OR .exe
MyClassA
MyClassB
MyResource
V 1.1.254.1
Assembly signed with a digital certificate
Assembly version
<major>.<minor>.<build>.<revision>

Library Assembly

- .dll
- Contains types for an application to use
- Must be referenced

Referenced Assembly



System Reference Properties

(Name)	System
Aliases	global
Copy Local	False
Culture	
Description	System.dll
Embed Interop Types	False
File Type	Assembly
Identity	System
Path	C:\Program Files (x86)\Reference Assemblies\Microsoft\Framework\.NETFramework\v4.0\Profile\Client\System.dll
Resolved	True
Runtime Version	v4.0.30319
Specific Version	False
Strong Name	True
Version	4.0.0.0

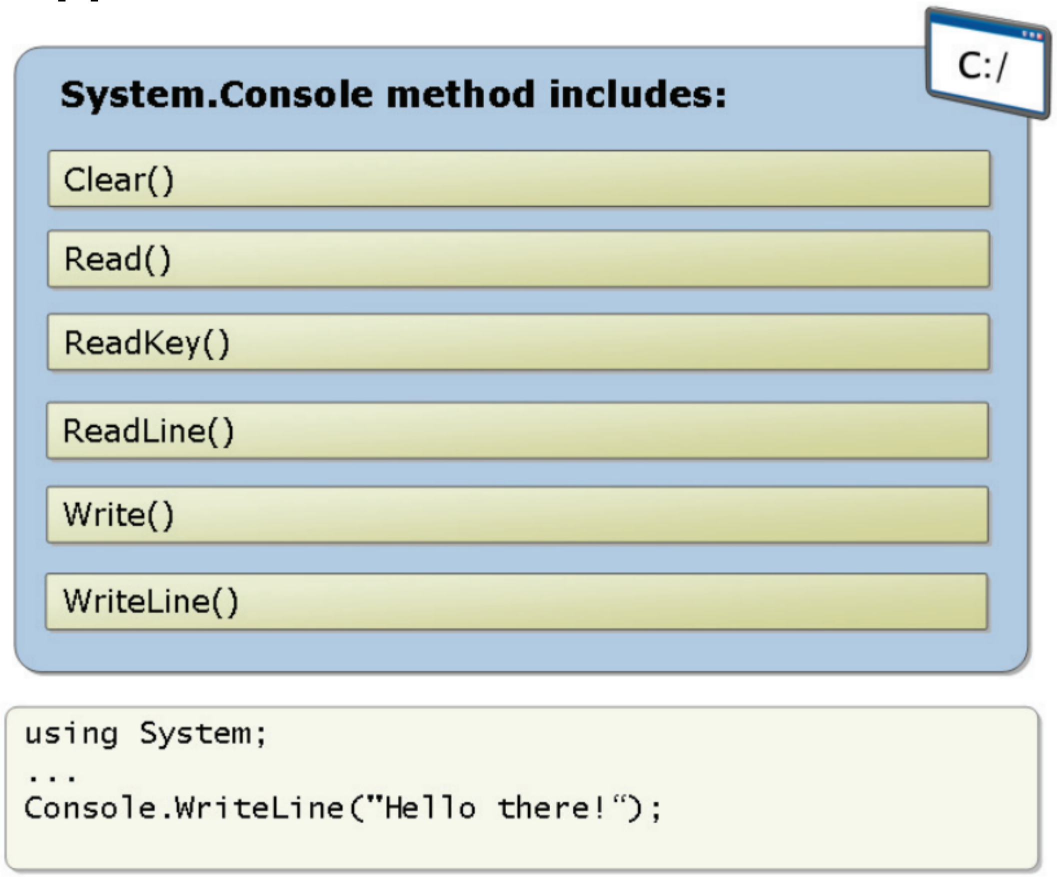
(Name)
Display name of the reference.

Console Projects

- “DOS” applications
- No GUI
- Have .exe file extension
- Run from the command line or called from batch files
- Accept input from standard input (keyboard, no mouse)
- Usually write to standard output (screen)

Console Class

- **Represents the standard input and output for console applications**



System.Console method includes:

- Clear()
- Read()
- ReadKey()
- ReadLine()
- Write()
- WriteLine()

```
using System;
...
Console.WriteLine("Hello there!“);
```

Compiling Code

Visual Studio

1 In Visual Studio 2010, on the **Build** menu, click **Build Solution**

2 On the **Debug** menu, click **Start Debugging**

Command line

C:/

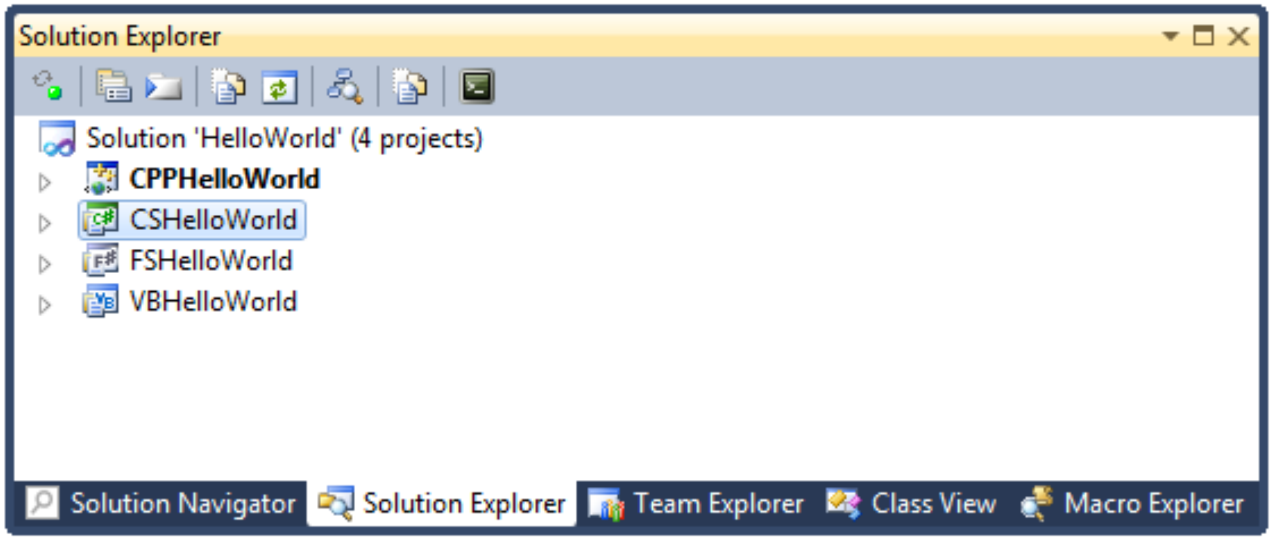
```
csc.exe /t:exe /out:" C:\Users\Student\Documents\Visual Studio
2010\MyProject\myApplication.exe"
"C:\Users\Student\Documents\Visual Studio 2010\MyProject\*.cs"
```

Hello World (CIL)

```
.method public static void Main() cil managed
{
   .entrypoint
   .maxstack 1
   ldstr "Hello, world!"
   call void
   [mscorlib]System.Console::WriteLine(string) ret
}
```

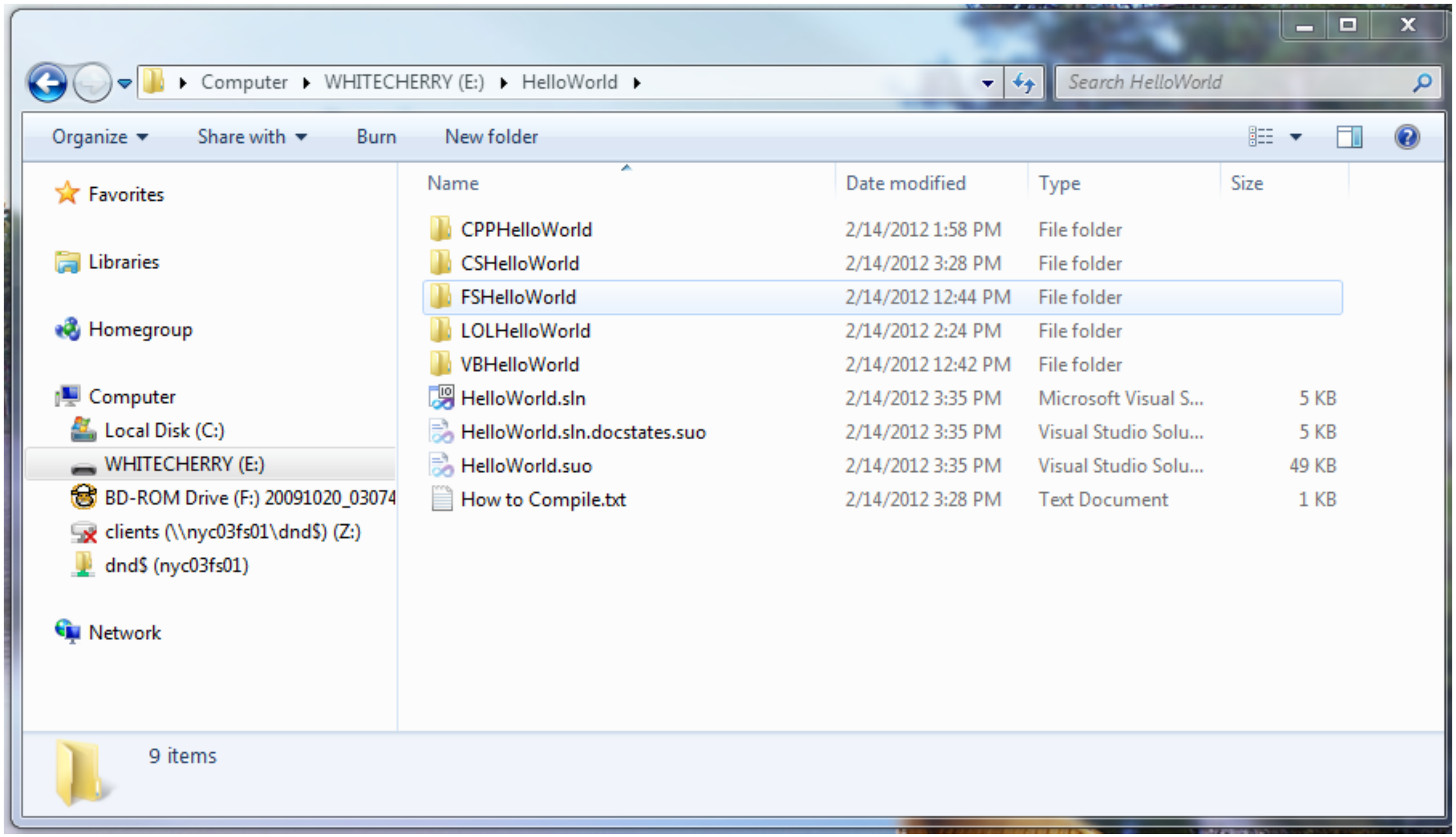
File Organization in Visual Studio

- **Solution – Top level container:**
  - contains projects and solution items
  - On disk, it's a file *.sln
  - .sln file should be at a directory above project folders
- **Project – contains source code files**
- **Each project has its own folder**



Solution 'HelloWorld' (4 projects)
- CPPHelloWorld
- CSHelloWorld
- FSHelloWorld
- VBHelloWorld

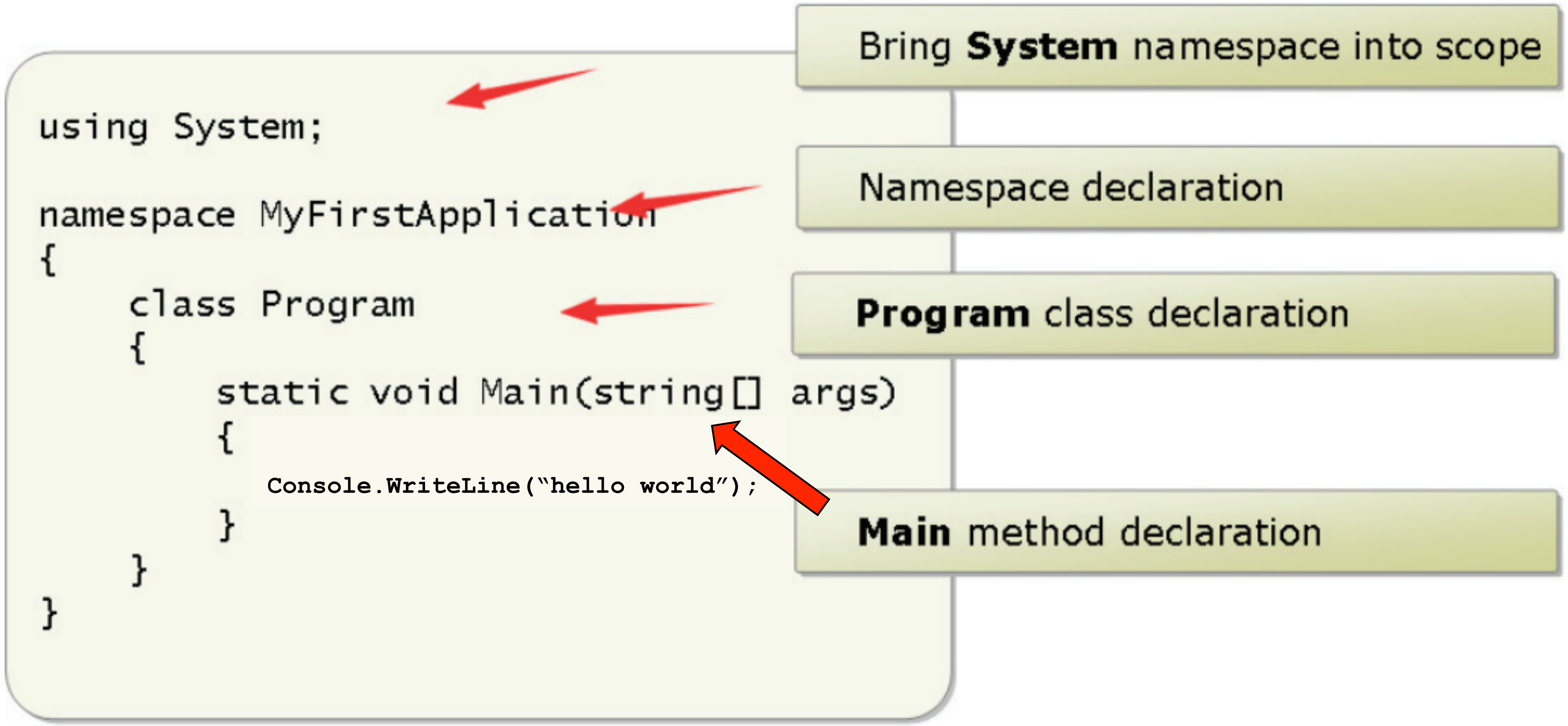
File Organization on Disk



Name	Date modified	Type	Size
CPPHelloWorld	2/14/2012 1:58 PM	File folder	
CSHelloWorld	2/14/2012 3:28 PM	File folder	
FSHelloWorld	2/14/2012 12:44 PM	File folder	
LOLHelloWorld	2/14/2012 2:24 PM	File folder	
VBHelloWorld	2/14/2012 12:42 PM	File folder	
HelloWorld.sln	2/14/2012 3:35 PM	Microsoft Visual S...	5 KB
HelloWorld.sln.docstates.suo	2/14/2012 3:35 PM	Visual Studio Solu...	5 KB
HelloWorld.suo	2/14/2012 3:35 PM	Visual Studio Solu...	49 KB
How to Compile.txt	2/14/2012 3:28 PM	Text Document	1 KB

9 items

Structure of a C# Program



```
using System;

namespace MyFirstApplication
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("hello world");
        }
    }
}
```

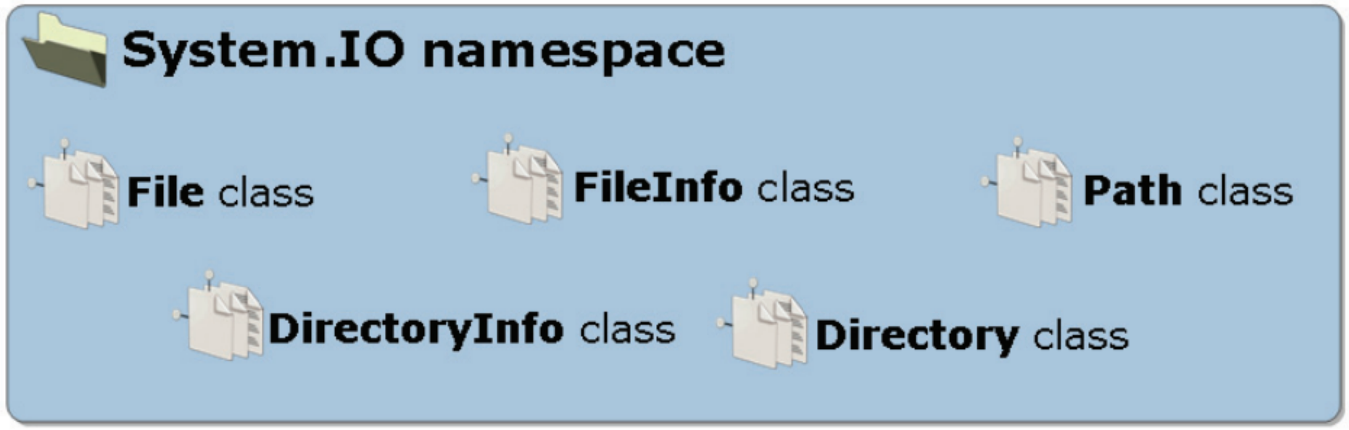
Bring **System** namespace into scope
Namespace declaration
Program class declaration
Main method declaration

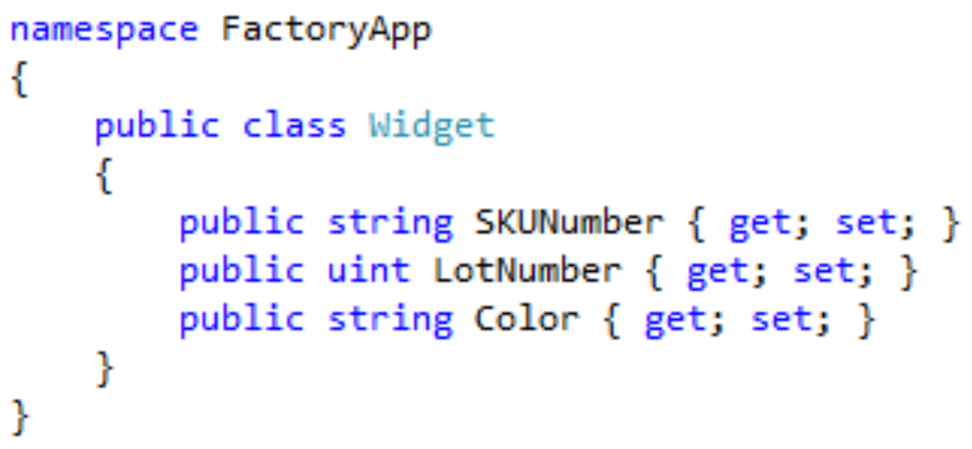
Namespace

- **Organizes code**
- **Group type names, reducing chance of collision**

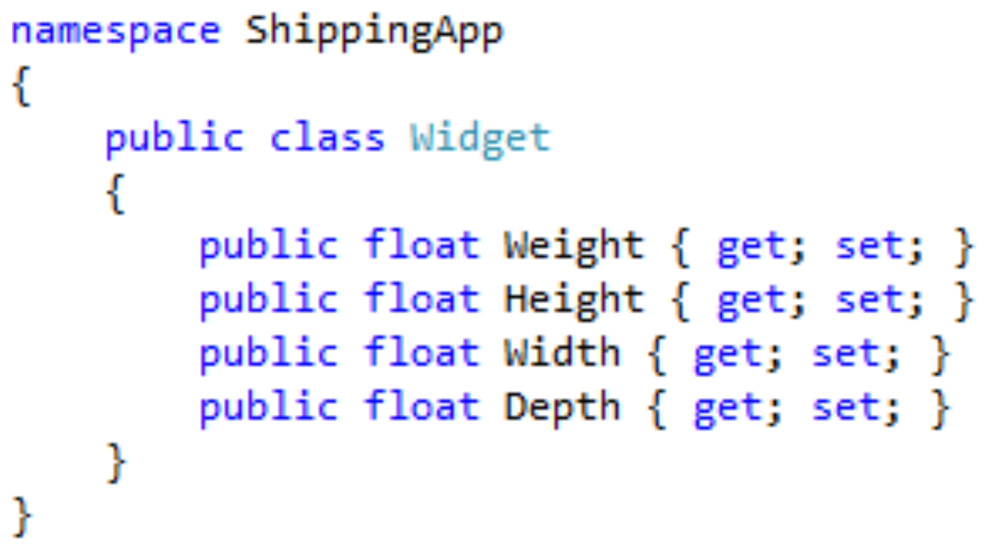
A class is essentially a blueprint that defines the characteristics of an entity

A namespace represents a logical collection of classes





```
namespace FactoryApp
{
    public class Widget
    {
        public string SKUNumber { get; set; }
        public uint LotNumber { get; set; }
        public string Color { get; set; }
    }
}
```



```
namespace ShippingApp
{
    public class Widget
    {
        public float Weight { get; set; }
        public float Height { get; set; }
        public float Width { get; set; }
        public float Depth { get; set; }
    }
}
```