.NET Technology

Subjects:

- NET Solution
-Building Blocks of the .NET Platform
-Common Language Runtime
-.NET Assemblies
-The Role of .NET Type Metadata
-Just-in-time compiling
-Platform independence

Introduction

What's a program?

What is a program? In a practical sense, a Windows OS program is an executable file that you can run by double-clicking its icon. For example, the version of Microsoft Word is a program. You call that an *executable program,* or *executable* for short. The names of executable program files generally end with the extension **.exe**. Word, for example, is **Winword.exe**.

But a program is something else, as well. **An executable program consists of one or more *source files*. A C# *source file,* for instance, is a text file that contains a sequence of C# commands, which fit together according to the laws of C# grammar. This file is known as a *source file.***

What's C#?

The C# programming language is one of those intermediate languages that programmers use to create executable programs. C# combines the range of the powerful but complicated C++ (pronounced "see plus plus") with the ease of use of the friendly but more verbose Visual Basic. (Visual Basic's newer .NET incarnation is almost on par with C# in most respects. As the flagship language of .NET, C# tends to introduce most new features first.) A C# program file carries the extension **.cs**.

Introduction

C# is

***Flexible*:** C# programs can execute on the current machine, or they can be transmitted over the Web and executed on some distant computer.

***Powerful*:** C# has essentially the same command set as C++ but with the rough edges filed smooth.

***Easier to use*:** C# error-proofs the commands responsible for most C++ errors, so you spend far less time chasing down those errors.

***Visually oriented*:** The .NET code library that C# uses for many of its capabilities provides the help needed to readily create complicated display frames with drop-down lists, tabbed windows, grouped buttons, scroll bars, and background images, to name just a few.

***Internet-friendly*:** C# plays a pivotal role in the .NET Framework, Microsoft's current approach to programming for Windows, the Internet, and beyond. .NET is pronounced *dot net*.

***Secure*:** Any language intended for use on the Internet must include serious security to protect against malevolent hackers.

Finally, C# is an integral part of .NET.

.NET Solution

What's .NET?

.NET began several years ago as Microsoft's strategy to open up the Web to mere mortals like you and me. Today, it's bigger than that, encompassing everything Microsoft does. In particular, it's the new way to program for Windows. It also gives a C-based language, C#, the simple, visual tools that made Visual Basic so popular.

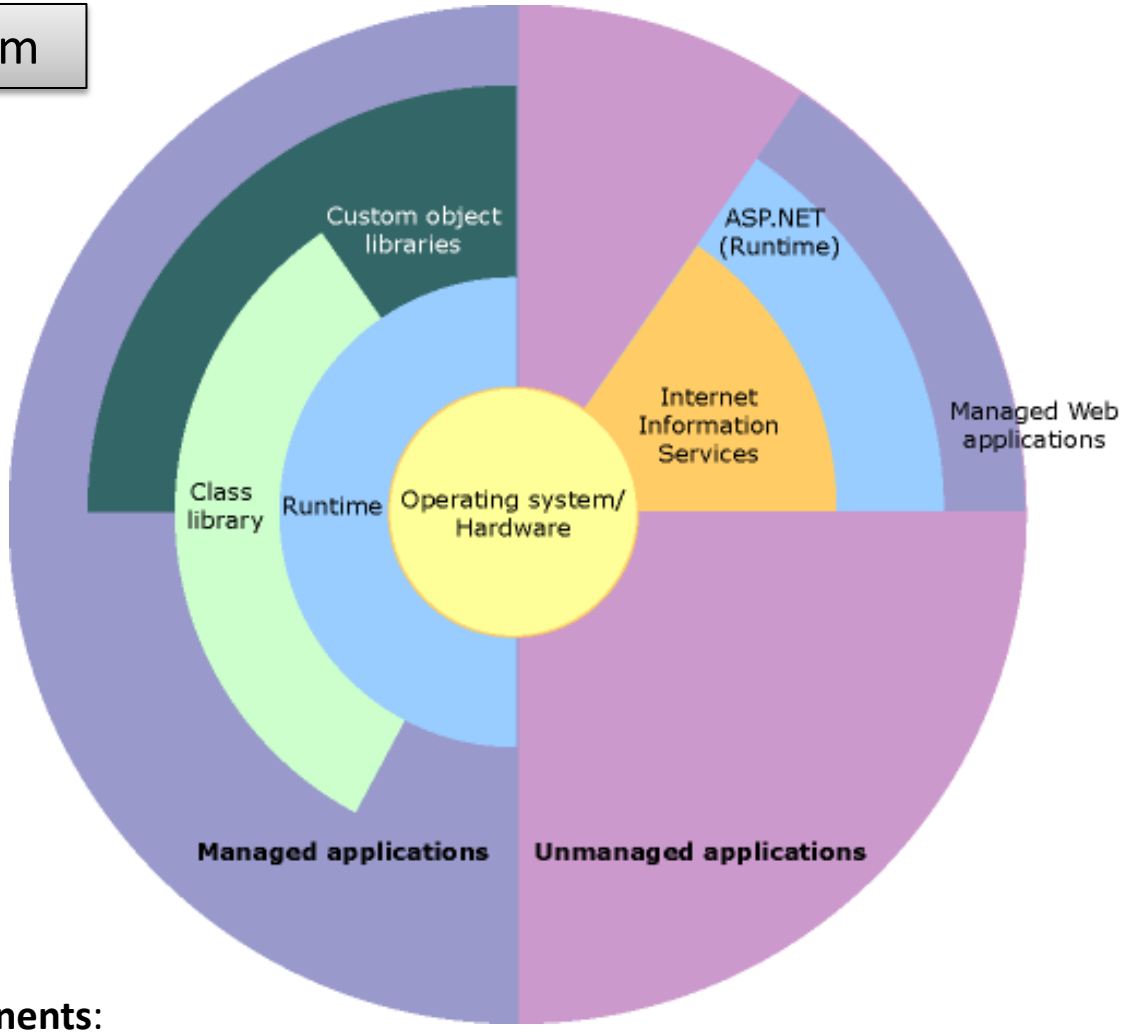
.NET Solution

The .NET Framework is an integral Windows component that supports building and running the next generation of applications and XML Web services.

The .NET Framework is designed to fulfill the following objectives:

- to **provide a consistent object-oriented programming environment** whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely;
- to **provide a code-execution environment** that **minimizes software deployment and versioning conflicts**;
- to **provide a code-execution environment** that promotes **safe execution of code**, including code created by an unknown or semi-trusted third party;
- to **provide a code-execution environment** that **eliminates the performance problems** of scripted or interpreted environments;
- to **make the developer experience consistent across widely varying types of applications**, such as Windows-based applications and Web-based applications;
- to **build all communication on industry standards** to ensure that code based on the .NET Framework can integrate with any other code.

Building Blocks of the .NET Platform



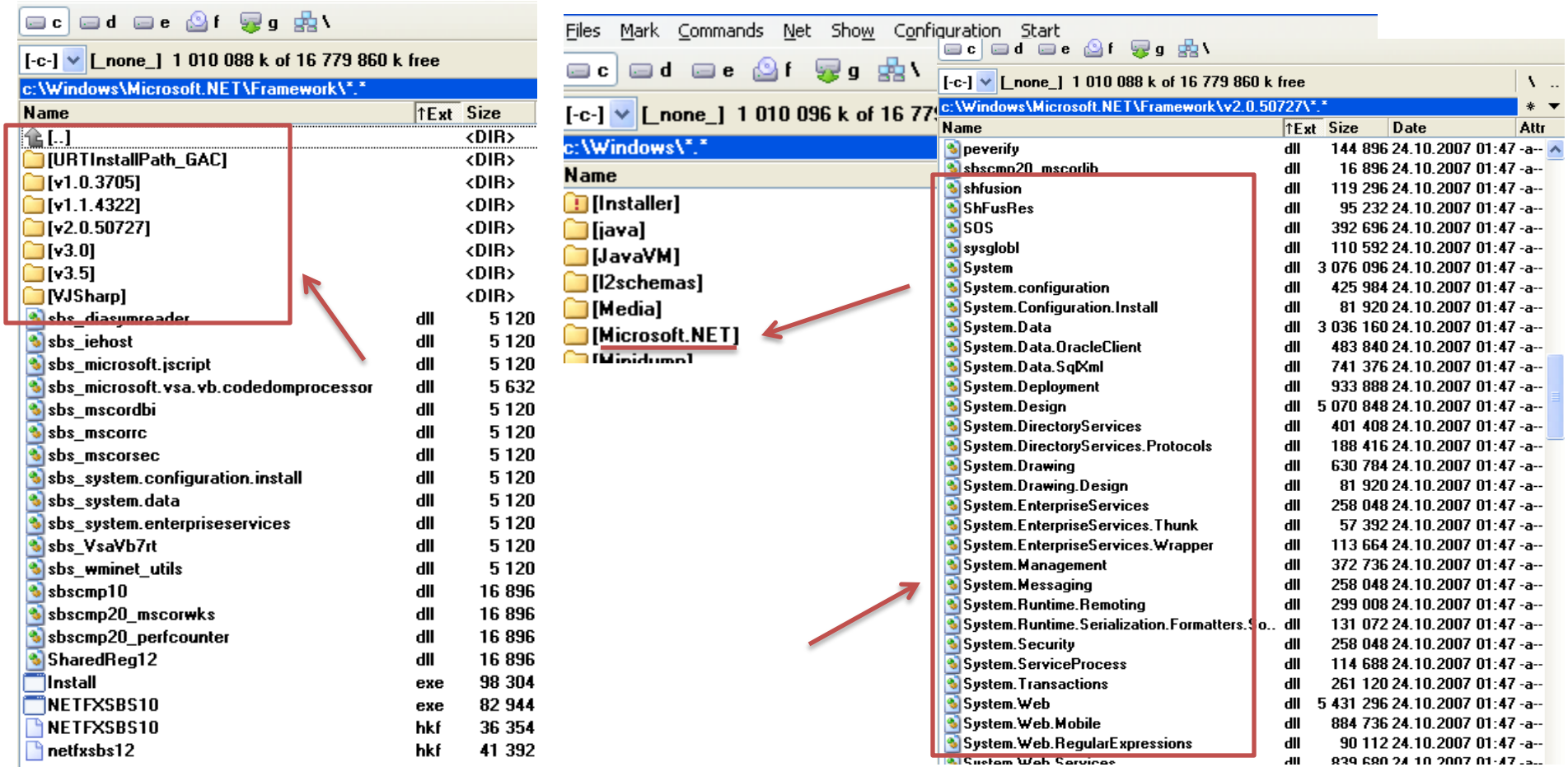
The .NET Framework has two main components:

- **the common language runtime** (**an agent that manages code at execution time**, providing core services such as **memory management, thread management, and remoting**, while also enforcing strict type safety and other forms of code accuracy that promote security and robustness)
- **the .NET Framework class library** (a comprehensive, **object-oriented collection of reusable types that you can use to develop applications** ranging from traditional command-line or graphical user interface applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services).

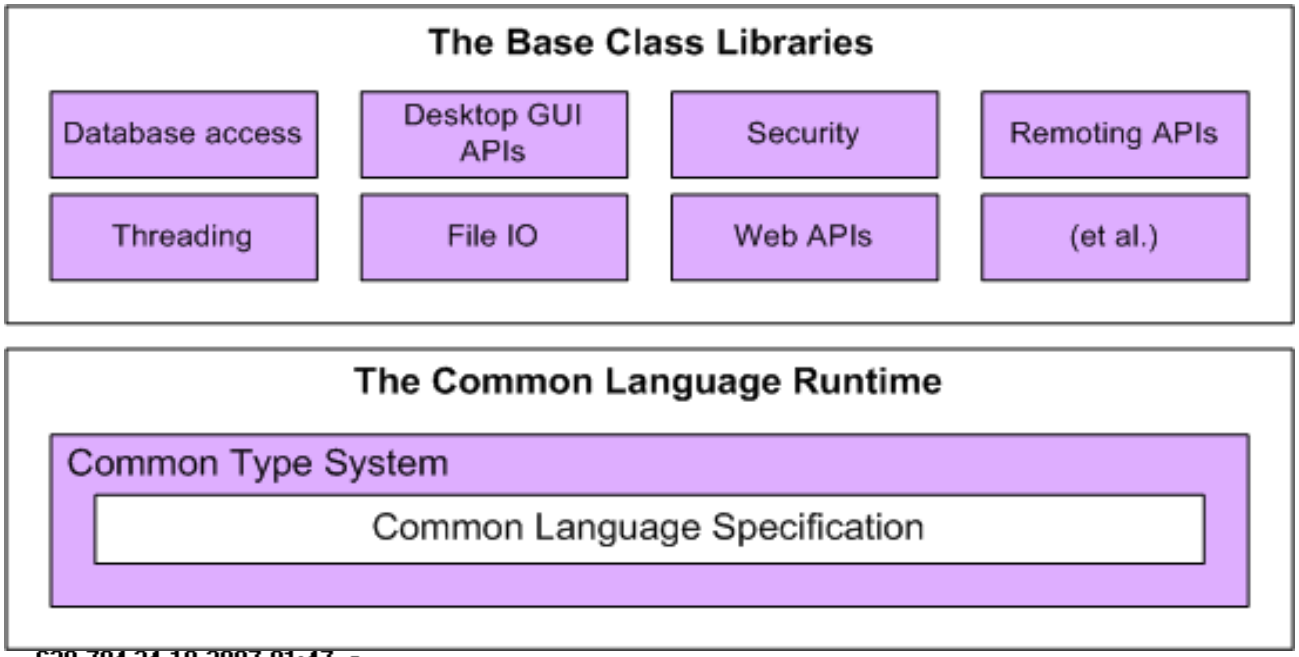
Common Language Runtime

Manages memory, thread execution, code execution, code safety verification, compilation, and other system services.

It enforces code robustness by implementing a **strict type-and-code-verification infrastructure** called the **common type system** (CTS).



Role of the Base Class Libraries

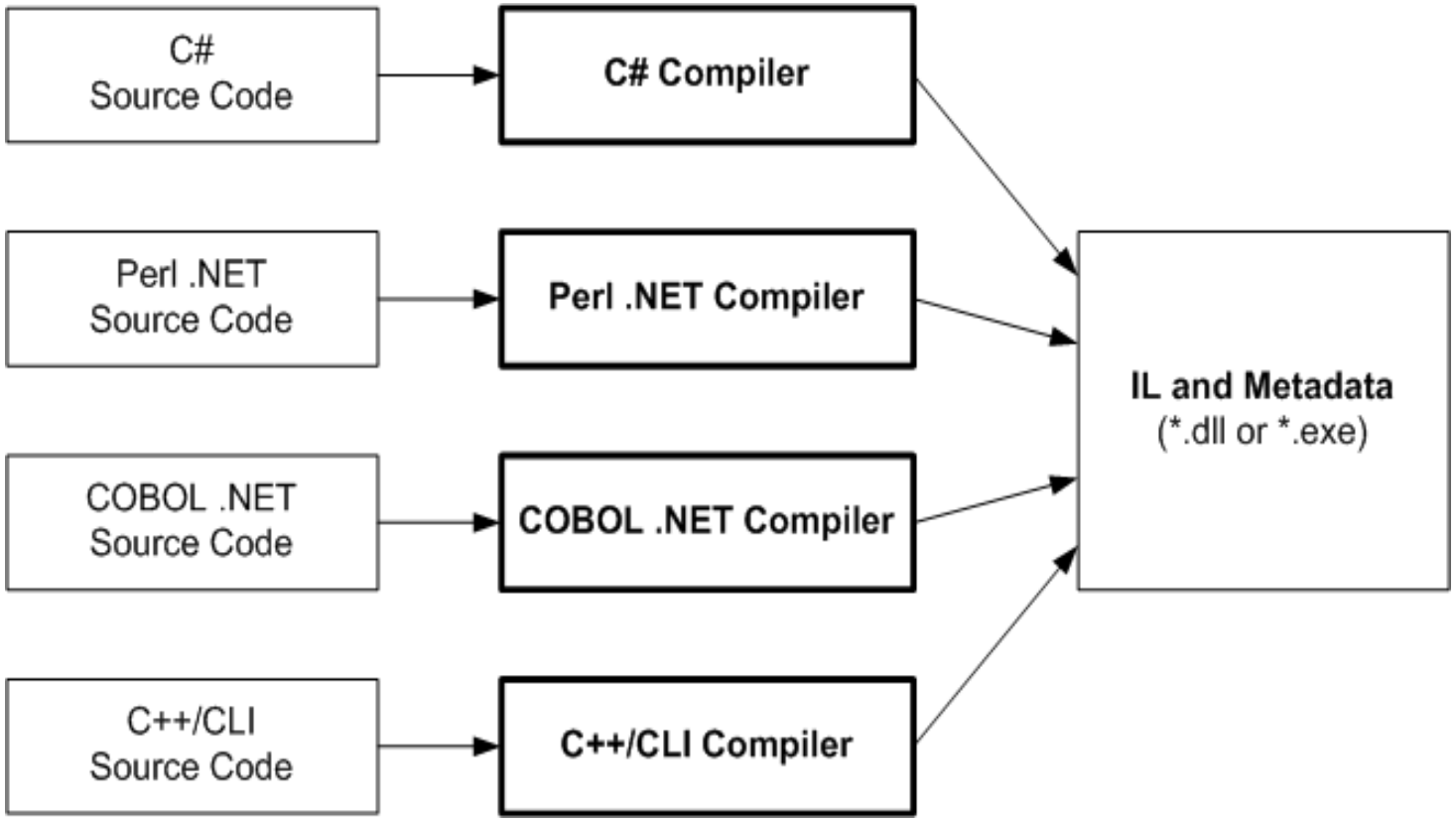


System	dll				
System.configuration	dll				
System.Configuration.Install	dll				
System.Data	dll				
System.Data.OracleClient	dll				
System.Data.SqlXml	dll				
System.Deployment	dll				
System.Design	dll				
System.DirectoryServices	dll				
System.DirectoryServices.Protocols	dll				
System.Drawing	dll	630	784	24.10.2007 01:47	-a-
System.Drawing.Design	dll	81	920	24.10.2007 01:47	-a-
System.EnterpriseServices	dll	258	048	24.10.2007 01:47	-a-
System.EnterpriseServices.Thunk	dll	57	392	24.10.2007 01:47	-a-
System.EnterpriseServices.Wrapper	dll	113	664	24.10.2007 01:47	-a-
System.Management	dll	372	736	24.10.2007 01:47	-a-
System.Messaging	dll	258	048	24.10.2007 01:47	-a-
System.Runtime.Remoting	dll	299	008	24.10.2007 01:47	-a-
System.Runtime.Serialization.Formatters.So..	dll	131	072	24.10.2007 01:47	-a-
System.Security	dll	258	048	24.10.2007 01:47	-a-
System.ServiceProcess	dll	114	688	24.10.2007 01:47	-a-
System.Transactions	dll	261	120	24.10.2007 01:47	-a-
System.Web	dll	5 431	296	24.10.2007 01:47	-a-
System.Web.Mobile	dll	884	736	24.10.2007 01:47	-a-
System.Web.RegularExpressions	dll	90	112	24.10.2007 01:47	-a-
System.Web.Services	dll	839	680	24.10.2007 01:47	-a-
System.Windows.Forms	dll	5 013	504	24.10.2007 01:47	-a-
System.XML	dll	2 068	480	24.10.2007 01:47	-a-

NET platform provides a base class library that is available to **all .NET programming languages.**

.NET Assemblies

Regardless of which .NET language you choose to program with, understand that despite the fact that **.NET binaries take the same file extension** as COM servers and unmanaged Win32 binaries (*.dll or *.exe), they have absolutely **no internal similarities**.



NET binaries do not contain platform-specific instructions, but rather platform-agnostic **intermediate language** (IL) and **type metadata**.

When a *.dll or an *.exe has been created using a .NET-aware compiler, the resulting module is bundled into an **assembly**.

Simple program (C# and Visual Basic .NET)

```
using System;
namespace CalculatorExample
{
 class Program // app's entry point.
 {
  static void Main()
  {
   Calc c = new Calc();
   int ans = c.Add(10, 84);
   Console.WriteLine("10 + 84 is {0}.", ans);
   Console.ReadLine();
  }
 }

 class Calc   // The C# calculator.
 {
  public int Add(int x, int y)
  { return x + y; }
 }
}
```

```
Imports System

Namespace CalculatorExample
  Module Program
    Sub Main()
      Dim c As New Calc
      Dim ans As Integer = c.Add(10, 84)
      Console.WriteLine("10 + 84 is {0}.", ans)
      Console.ReadLine()
    End Sub
  End Module

  Class Calc
    Public Function Add(ByVal x As Integer,
                        ByVal y As Integer) As Integer
      Return x + y
    End Function
  End Class
End Namespace
```

```
.method public hidebysig instance
int32 Add(int32 x, int32 y) cil managed
{
// Code size 9 (0x9)
.maxstack 2
.locals init (int32 V_0)
IL_0000: nop
IL_0001: ldarg.1
IL_0002: ldarg.2
IL_0003: add
IL_0004: stloc.0
IL_0005: br.s IL_0007
IL_0007: ldloc.0
IL_0008: ret
} // end of method Calc::Add
```

```
.method public instance
int32 Add(int32 x, int32 y) cil managed
{
// Code size 8 (0x8)
.maxstack 2
.locals init (int32 V_0)
IL_0000: ldarg.1
IL_0001: ldarg.2
IL_0002: add.ovf
IL_0003: stloc.0
IL_0004: br.s IL_0006
IL_0006: ldloc.0
IL_0007: ret
} // end of method Calc::Add
```

The Role of .NET Type Metadata

In addition to CIL instructions, a .NET assembly contains **full**, **complete**, and **accurate** metadata, which describes each and every type (class, structure, enumeration, and so forth) defined in the binary, as well as the members of each type (properties, methods, events, and so on).

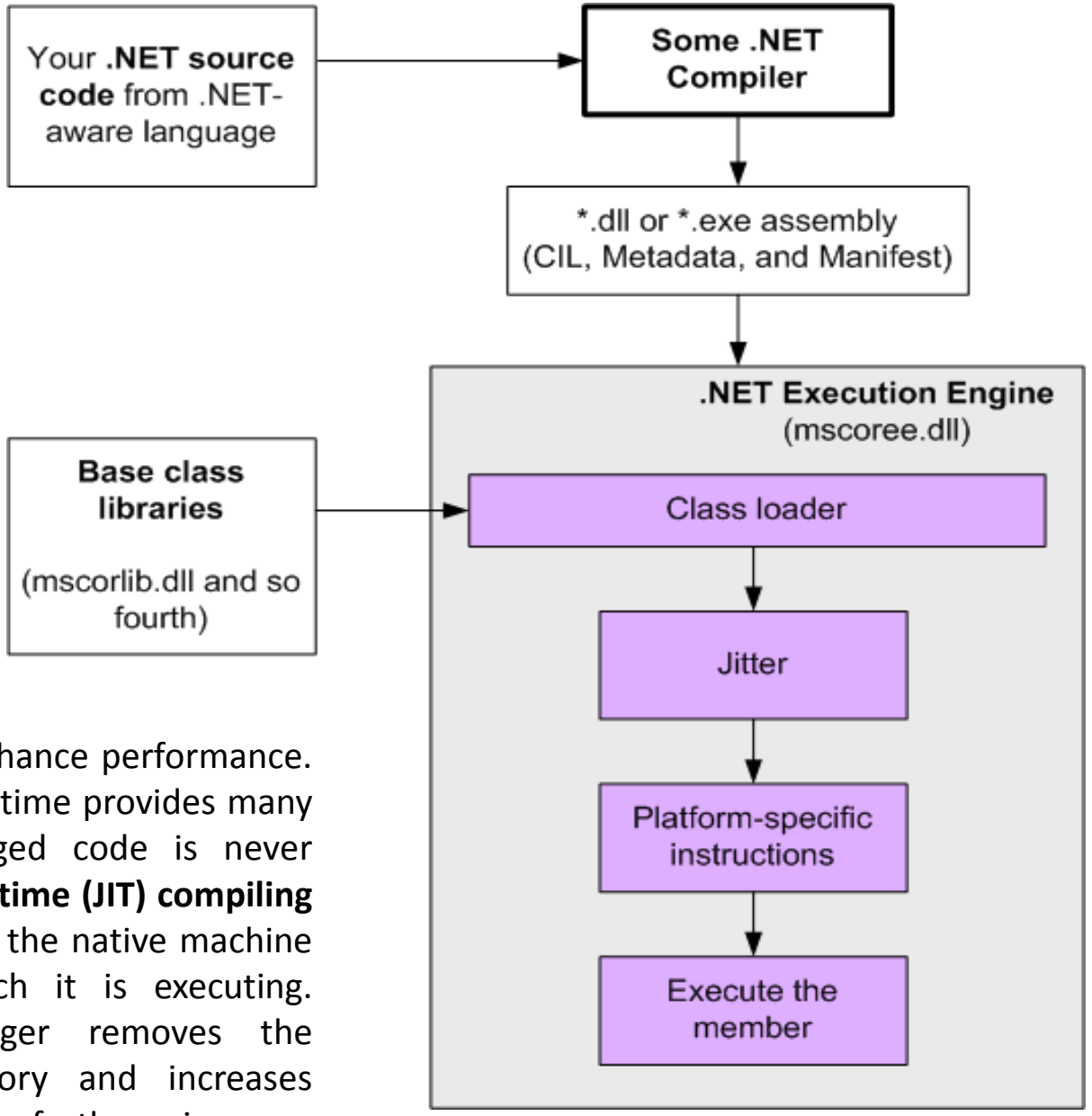
CREATING OF METADATA IS COMPILER JOB !!!

Because .NET metadata is so wickedly meticulous, assemblies are **completely self-describing entities**.

```
TypeDef #2 (02000003)
-------------------------------------------------------
TypDefName: CalculatorExample.Calc (02000003)
Flags : [NotPublic] [AutoLayout] [Class]
[AnsiClass] [BeforeFieldInit] (00100001)
Extends : 01000001 [TypeRef] System.Object
Method #1 (06000003)
-------------------------------------------------------
MethodName: Add (06000003)
Flags : [Public] [HideBySig] [ReuseSlot] (00000086)
RVA : 0x00002090
ImplFlags : [IL] [Managed] (00000000)
CallCnvntn: [DEFAULT]
hasThis
ReturnType: I4
2 Arguments
Argument #1: I4
Argument #2: I4
2 Parameters
(1) ParamToken : (08000001) Name : x flags: [none] (00000000)
(2) ParamToken : (08000002) Name : y flags: [none] (00000000)
```

```
class Calc   // The C# calculator.
{
 public int Add(int x, int y)
 { return x + y; }
}
```

Just-in-time compiling



The **runtime** is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called **just-in-time (JIT) compiling** enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Platform independence

When Microsoft released the C# programming language and the .NET platform, they also crafted a set of formal documents that described the syntax and semantics of the C# and CIL languages, the .NET assembly format, core .NET namespaces, and the mechanics of a hypothetical .NET runtime engine (known as the **Virtual Execution System**, or VES).

Better yet, these documents have been submitted to (and ratified by) ECMA International as official international standards. The specifications of interest are (**http://www.ecma-international.org**):

- ECMA-334: The C# Language Specification
- ECMA-335: The Common Language Infrastructure (CLI)

Partitions of ECMA-335	Meaning in Life
Partition I: Architecture	Describes the overall architecture of the CLI, including the rules of the CTS and CLS, and the mechanics of the .NET runtime engine
Partition II: Metadata	Describes the details of .NET metadata
Partition III: CIL	Describes the syntax and semantics of CIL code
Partition IV: Libraries	Gives a high-level overview of the minimal and complete class libraries that must be supported by a .NET distribution.
Partition V: Annexes	Provides a collection of “odds and ends” details such as class library design guidelines and the implementation details of a CIL compiler

Open source .NET distributions

Partition IV (Libraries) **defines** a minimal set of namespaces that represent the core services expected by a CLI distribution (collections, console I/O, file I/O, threading, reflection, network access, core security needs, XML manipulation, and so forth).

The CLI does **not define** namespaces that facilitate web development (ASP.NET), database access (ADO.NET), or desktop graphical user interface (GUI) application development (Windows Forms/Windows Presentation Foundation).

Distribution	Meaning in Life
http://www.mono-project.com	The Mono project is an open source distribution of the CLI that targets various Linux distributions (e.g., SuSE, Fedora, and so on) as well as Win32 and Mac OS X.
http://www.dotgnu.org	Portable.NET is another open source distribution of the CLI that runs on numerous operating systems. Portable.NET aims to target as many operating systems as possible (Win32, AIX, BeOS, Mac OS X, Solaris, all major Linux distributions, and so on).

Thanks for attention