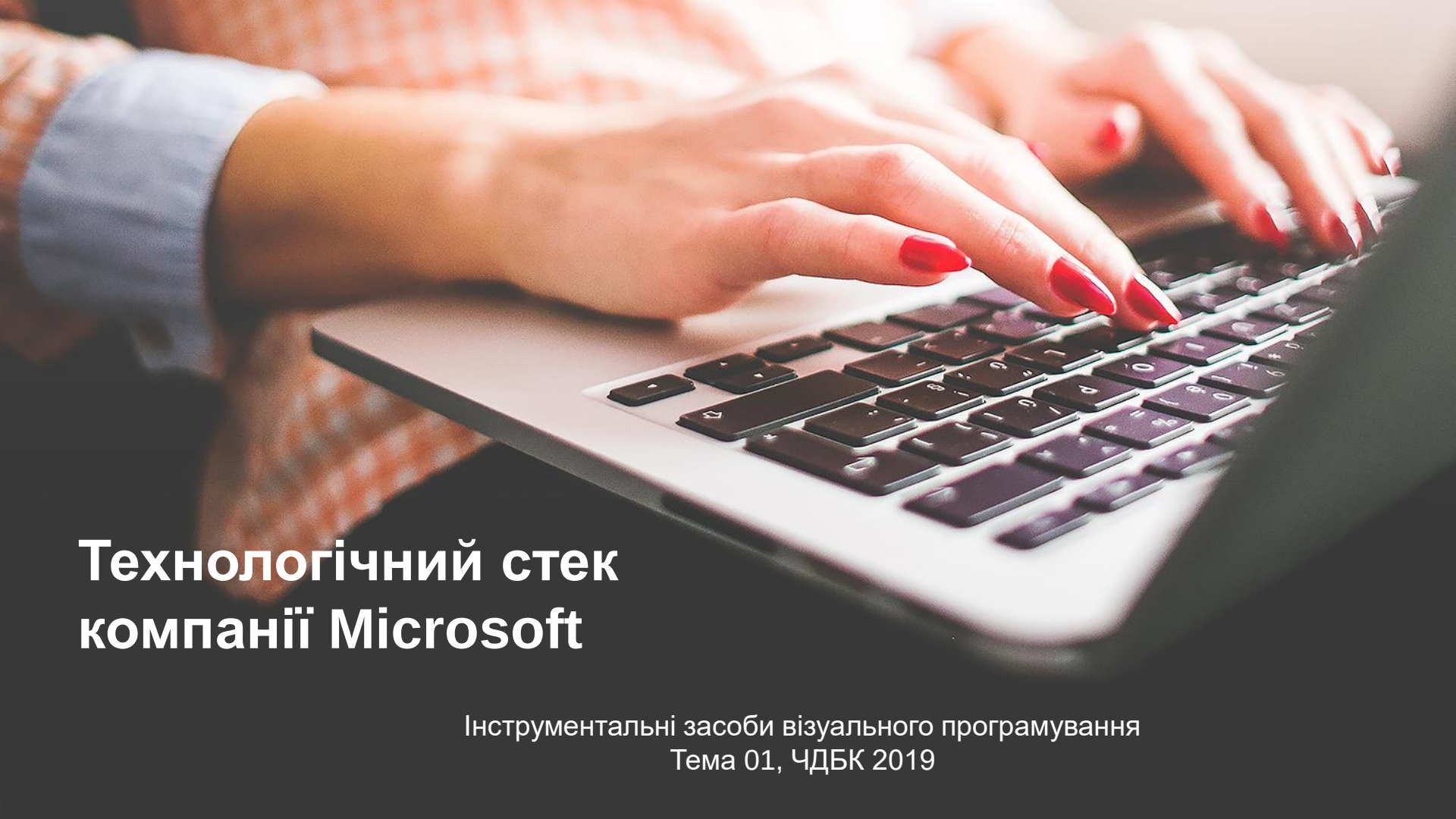
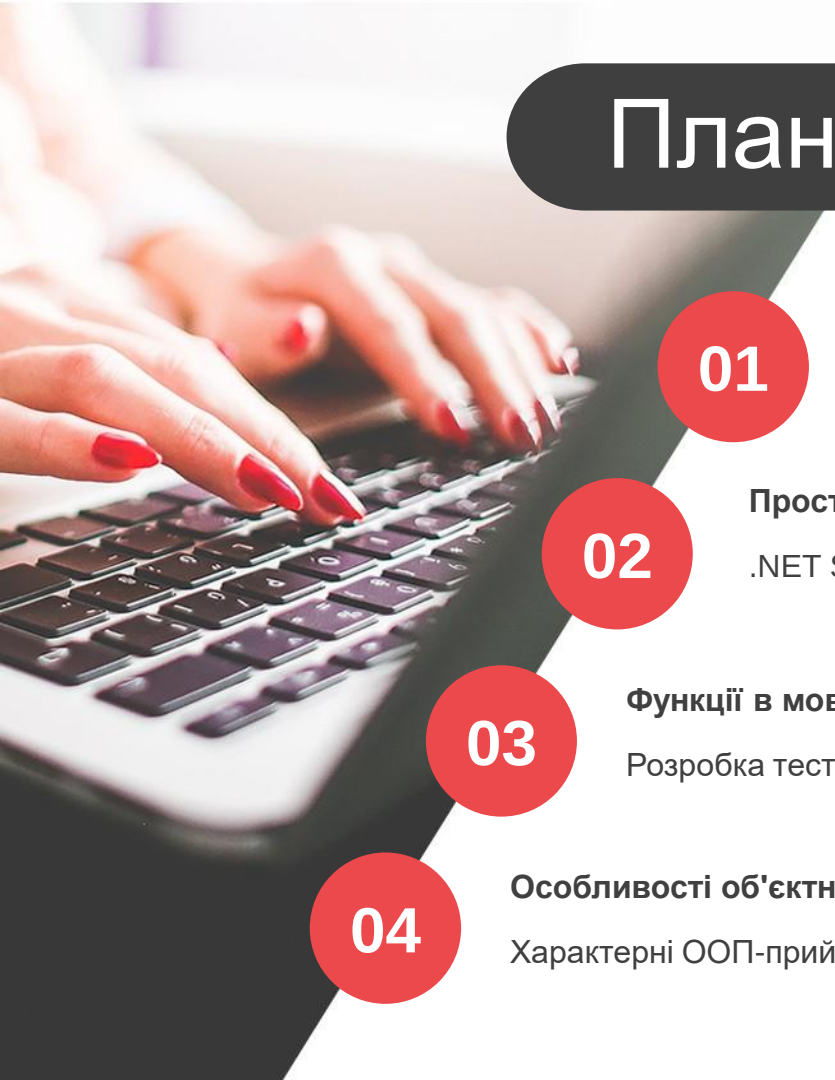


Технологічний стек компанії Microsoft

Інструментальні засоби візуального програмування
Тема 01, ЧДБК 2019



План лекції

01

Огляд платформи .NET

.NET Framework, .NET Core, .NET Standard, .NET Native

02

Простори імен та система типів .NET

.NET Standard та типи даних в мові C#

03

Функції в мові C#: створення, налагодження, тестування

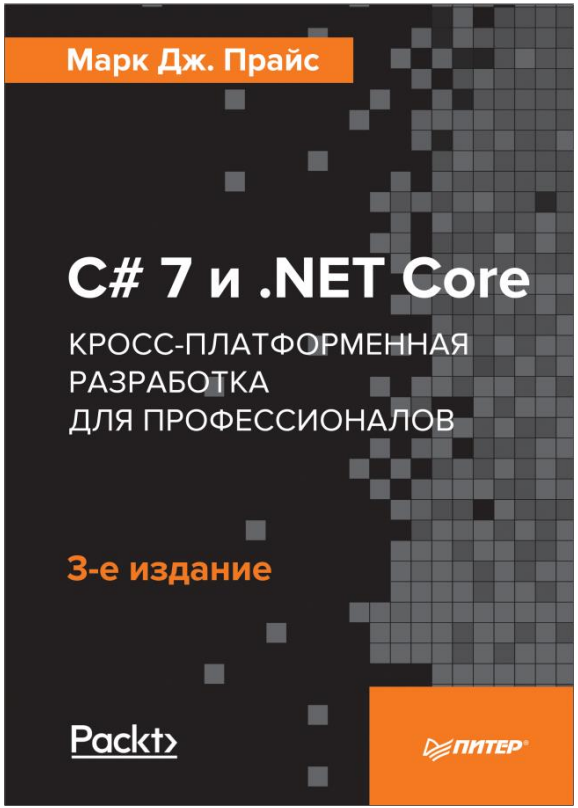
Розробка тестових та тестованих модулів коду

04

Особливості об'єктно-орієнтованого програмування мовою C#

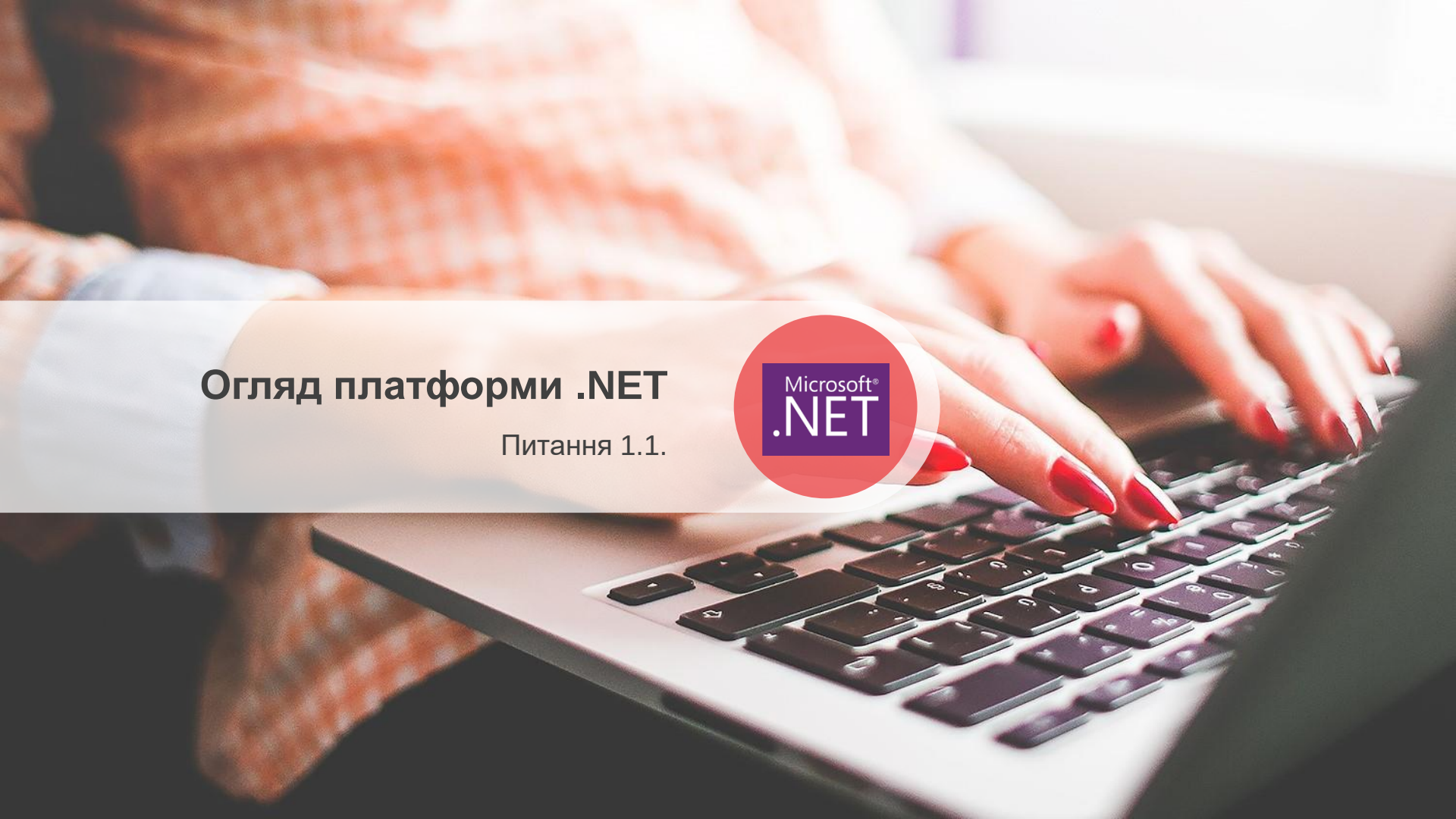
Характерні ООП-прийоми в мові C#

Література


Марк Дж. Прайс
C# 7 и .NET Core
КРОСС-ПЛАТФОРМЕННАЯ
РАЗРАБОТКА
ДЛЯ ПРОФЕССИОНАЛОВ
3-е издание
Packt
ПИТЕР

Огляд платформи .NET

Питання 1.1.



Технології платформи .NET

Пов’язані платформи для розробки додатків та сервісів, інколи перетинаються

.NET
Native

Технология	Возможности	Компилирует в	Хостовая ОС
.NET Framework	Как отработанные, так и современные	Промежуточный язык	Только Windows
Xamarin	Только мобильные	Промежуточный язык	iOS, Android, Windows Mobile
.NET Core	Только современные	Промежуточный язык	Windows, Linux, macOS
.NET Native	Только современные	Машинный код	Windows, Linux, macOS

Еволюція стеку технологій Microsoft

.NET Framework

- Common Language Runtime (CLR) – спільне середовище виконання програм
  - Автоматично знаходить, завантажує та керує об'єктами .NET замість програміста
  - Опікується деякими низькорівневими деталями: управління пам'яттю, розміщення додатку, координація потоків та виконання перевірок, пов'язаних з безпекою

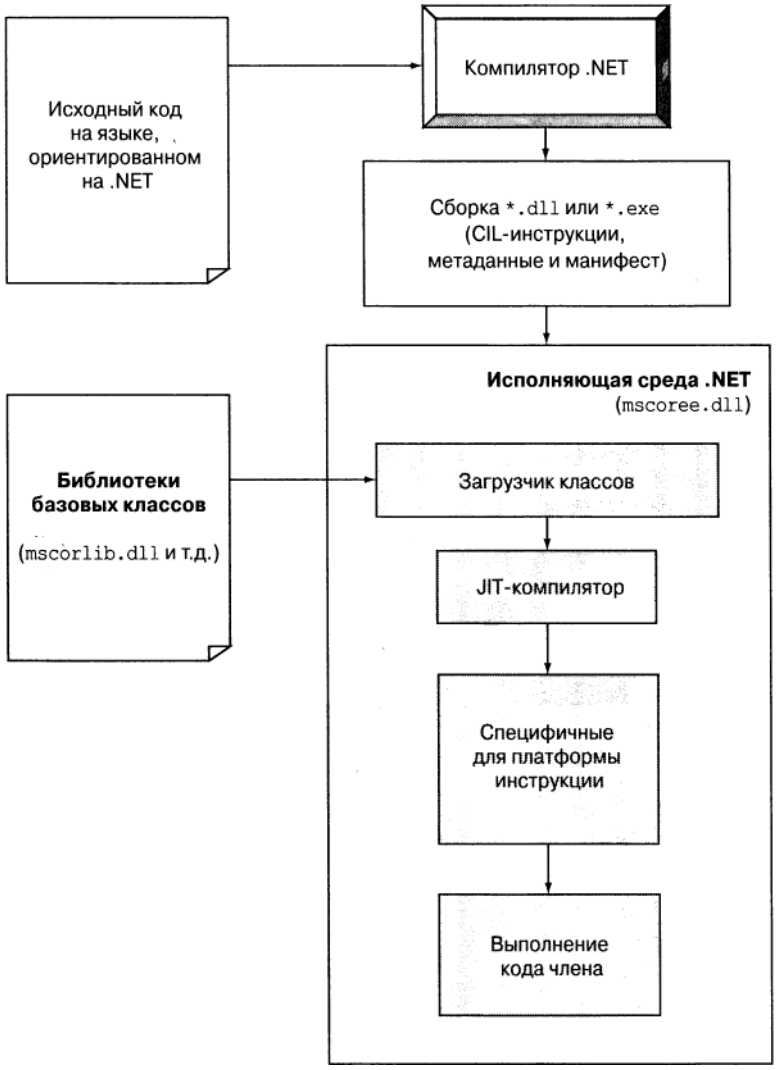
Спільне середовище виконання програм

- С точки зрения программирования под термином исполняющая среда может пониматься коллекция служб, которые требуются для выполнения скомпилированной единицы кода.
  - Основной механизм CLR физически имеет вид библиотеки по имени mscoree.dll (также известной как общий механизм выполнения исполняемого кода объектов (Common Object Runtime Execution Engine)).
  - Когда на сборку производится ссылка с целью ее использования, библиотека mscoree.dll загружается автоматически и, в свою очередь, загружает требуемую сборку в память.
  - Исполняющая среда отвечает за решение множества задач.

Спільне середовище виконання програм

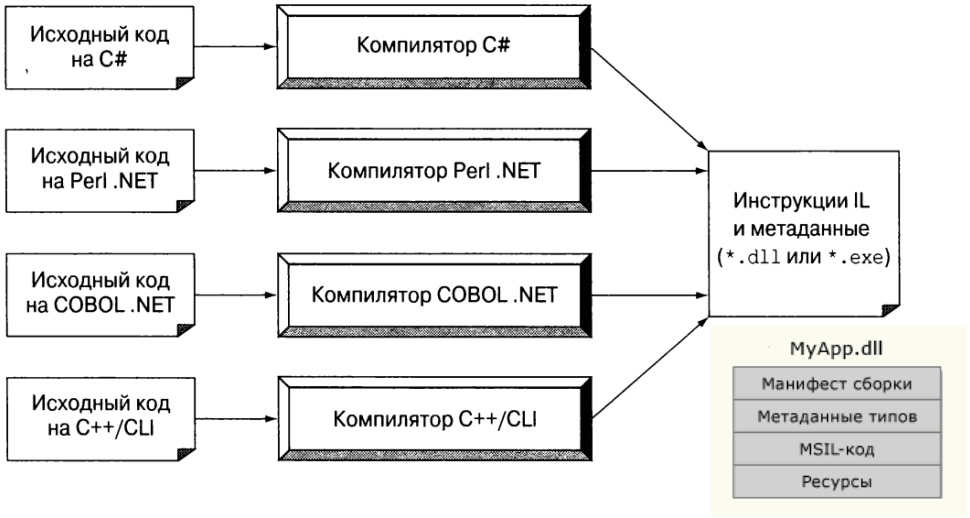
Збірка mscoree.dll в дії

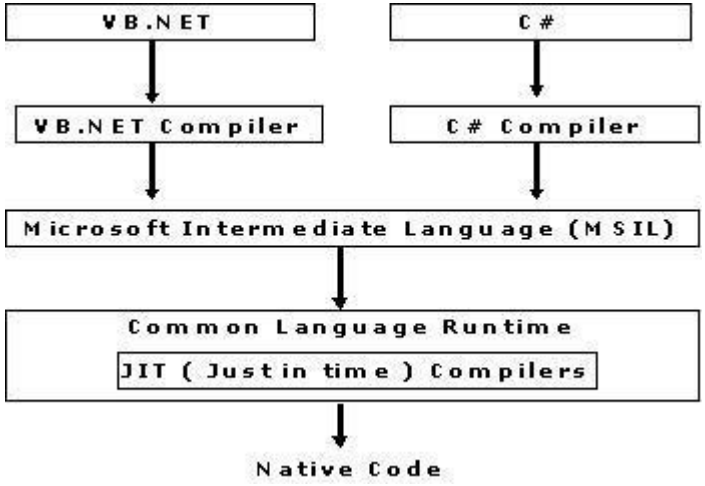
- Головна задача — определение местоположения сборки и нахождение запрошенного типа в двоичном файле за счет чтения содержащихся в нем метаданных.
  - Затем CLR размещает тип в памяти, преобразует связанный с ним CIL-код в специфичные для платформы инструкции, производит все необходимые проверки безопасности и после этого выполняет нужный код.



Крос-компіляція коду в .NET Framework

Роль спільного середовища виконання



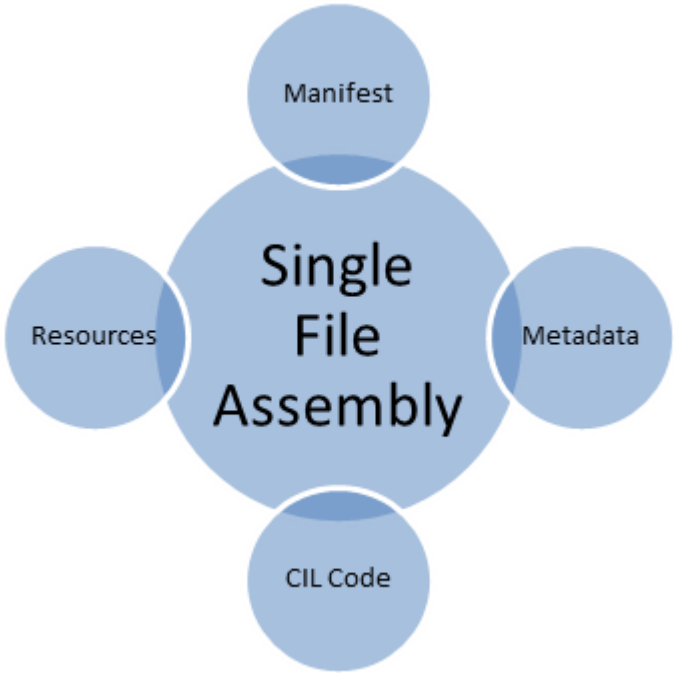


двоичные модули .NET содержат не специфические, а независимые от плат формы инструкции на промежуточном языке (Intermediate Language — IL) и метаданные типов.

Когда файл *.dll или *.exe был создан с помощью .NET-компилятора, получен ный большой двоичный объект называется сборкой (assembly).

Збірки .NET (файли *.dll/*.exe)

CIL-код

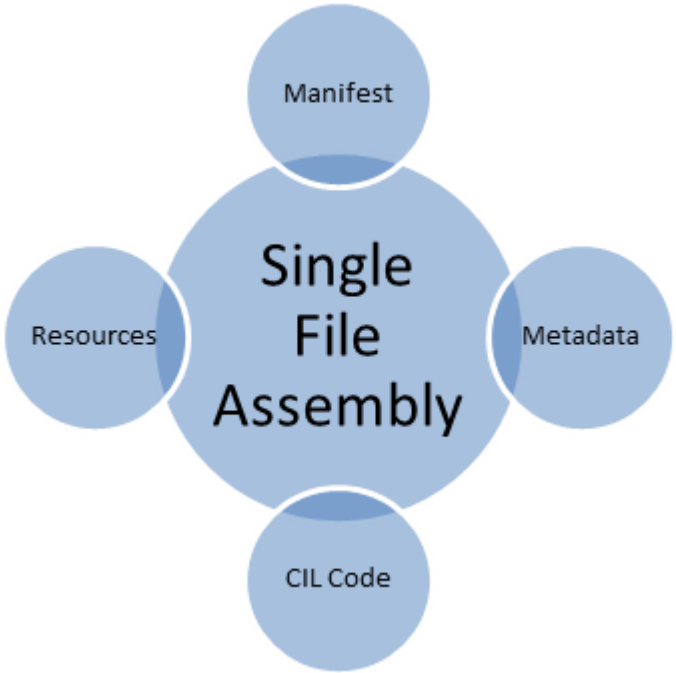



```
using System;
namespace CalculatorExample
{
  // Этот класс содержит точку входа приложения.
  class Program
  {
    static void Main()
    {
      Calc c = new Calc();
      int ans = c.Add(10, 84);
      Console.WriteLine("10 + 84 is {0}.", ans);
      // Ожидать нажатия пользователем клавиши <Enter>
      Console.ReadLine();
    }
  }
  // Калькулятор на C#.
  class Calc
  {
    public int Add(int x, int y)
    { return x + y; }
  }
}
```

```
.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
```

Збірки .NET (файли *.dll/*.exe)

Маніфест типів (метадані для методу Add())



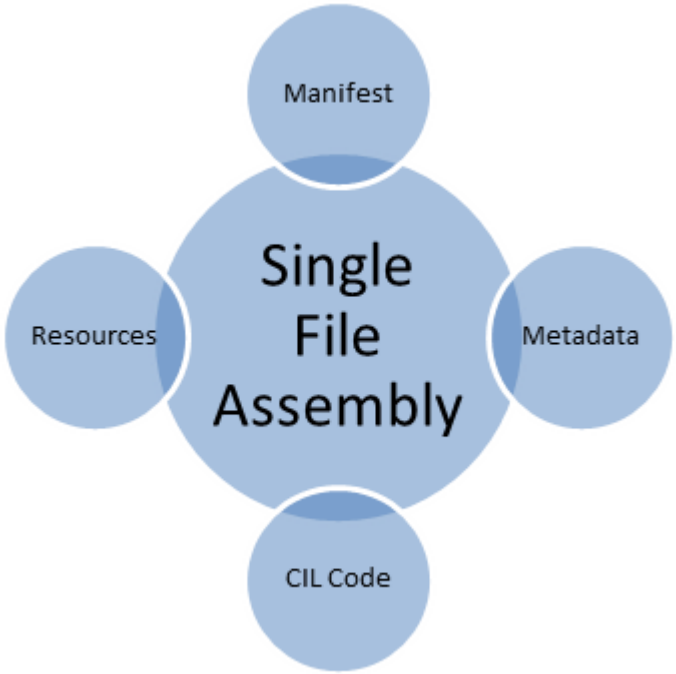
Type Descriptions

- Classes
- Base classes
- Implemented interfaces
- Attributes
- Methods

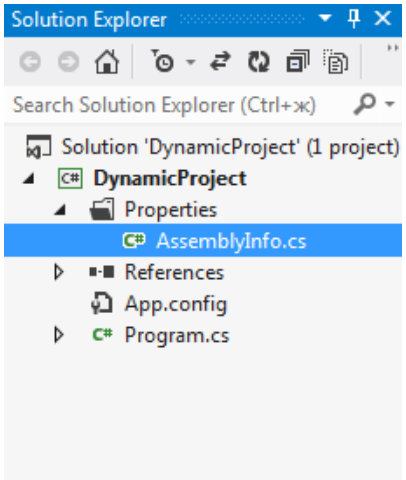
```
TypeDef #2 (02000003)
-------------------------------------------------------
  TypDefName: CalculatorExample.Calc  (02000003)
  Flags     : [NotPublic] [AutoLayout] [Class]
  [AnsiClass] [BeforeFieldInit]  (00100001)
  Extends   : 01000001 [TypeRef] System.Object
  Method #1 (06000003)
-------------------------------------------------------
  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)
```

Збірки .NET (файли *.dll/*.exe)

Маніфест збірки

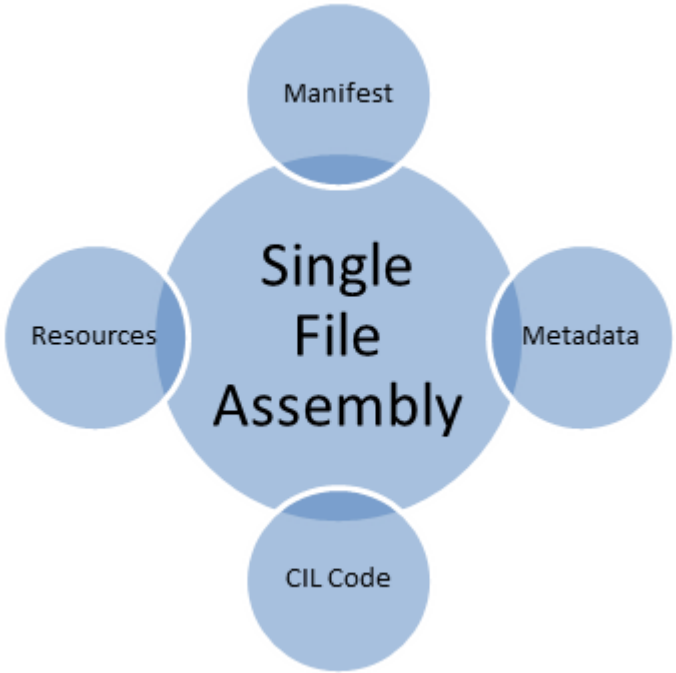


Манифест содержит информацию о текущей вер сии сборки, сведения о культуре (применяемые для локализации строковых и графических ресу рсов) и список ссылок на все внешние сборки, к оторые требуются для правильного функционир ования



Збірки .NET (файли *.dll/*.exe)

Ресурси



Сборка помимо исходного кода может содержать ресурсы: текст, изображения и XML-файлы.

Обычно в качестве ресурсов в сборку добавляются изображения или локализуемые данные.

Ресурс сборки в конечном итоге представляет собой байтовый поток с именем.

Сборка хранит их в словарях со строковыми ключами.

Головні компоненти .NET Framework

Спільна система типів та спільна специфікація мов

Common Type System (CTS) – спільна система типів

- Специфікація описує всі можливі типи даних та програмні конструкції, що підтримуються середовищем.
- Показано, як сутності можуть взаємодіяти та як вони представлені у вигляді метаданих .NET.

- Common Language Specification (CLS) – спільна специфікація мов
  - Описує підмножину спільних типів та програмних конструкцій, які повинні підтримувати всі мови програмування
  - Якщо створювані .NET-типи відповідають CLS, ними можуть користуватись усі .NET-мови
    - Інакше гарантувати можливість взаємодії з такою бібліотекою коду неможливо
    - Можна вказати компілятору, щоб перевірив код на відповідність CLS

Спільна система типів

Вбудовані типи даних CTS

Тип данных CTS	Ключевое слово VB	Ключевое слово C#	Ключевое слово C++/CLI
System.Byte	Byte	byte	unsigned char
System.SByte	SByte	sbyte	signed char
System.Int16	Short	short	short
System.Int32	Integer	int	int или long
System.Int64	Long	long	__int64
System.UInt16	UShort	ushort	unsigned short
System.UInt32	UInteger	uint	unsigned int или unsigned long
System.UInt64	ULong	ulong	unsigned __int64
System.Single	Single	float	float
System.Double	Double	double	double
System.Object	Object	object	object^
System.Char	Char	char	wchar_t
System.String	String	string	String^
System.Decimal	Decimal	decimal	Decimal
System.Boolean	Boolean	bool	bool

Спільна система типів

Класи та їх характеристики

- Класс может включать любое количество членов (таких как конструкторы, свойства, методы и события) и элементов данных (полей).

```
// Тип класса C# с одним методом.
class Calc
{
  public int Add(int x, int y)
  {
    return x + y;
  }
}
```

Характеристика классов	Описание
Является ли класс запечатанным?	Запечатанные классы не могут выступать в качестве базовых для других классов
Реализует ли класс какие-то интерфейсы?	Интерфейс — это коллекция абстрактных членов, которая предоставляет контракт между объектом и пользователем объекта. CTS позволяет классе реализовывать любое количество интерфейсов
Является класс абстрактным или конкретным?	Абстрактные классы не допускают прямого создания экземпляров и предназначены для определения общих поведений для производных типов. Экземпляры конкретных классов могут создаваться напрямую
Какова видимость класса?	Каждый класс должен конфигурироваться с ключевым словом видимости, таким как `public` или `internal`. По сути, оно управляет тем, может ли класс использоваться внешними сборками или только внутри определяющей его сборки

Спільна система типів

Інтерфейси

- Інтерфейси — це іменовані колекції визначень абстрактних членів, які можуть підтримуватись (бути реалізованими) в заданому класі або структурі.

```
// Тип интерфейса в C# обычно объявляется
// открытым, чтобы позволить типам в других
// сборках реализовать его поведение.
public interface IDraw
{
  void Draw();
}
```

Спільна система типів

Структури та перелічення

- структуру можна вважати полегшеним типом класу, який має семантику на основі значень

```
// Тип структуры в C#.
struct Point
{
  // Структуры могут содержать поля.
  public int xPos, yPos;

  // Структуры могут содержать параметризованные конструкторы.
  public Point(int x, int y)
  { xPos = x; yPos = y;}

  // Структуры могут определять методы.
  public void PrintPosition()
  {
    Console.WriteLine("({0}, {1})", xPos, yPos);
  }
}
```

- Перелічення — зручна програмна конструкція, яка дозволяє групувати пари «назва-значення».
  - По умолчанию для хранения каждого элемента выделяется блок памяти, соответствующий 32-битному целому, однако при необходимости это можно изменить.

```
enum CharacterType
{
  Wizard = 100,
  Fighter = 200,
  Thief = 300
}
```

Спільна система типів

Делегати

- Делегати є .NET-еквівалентом безпечних до типів вказівників на функції в стилі C.
  - Основна відмінність: делегат .NET – це клас, породжений від System.MulticastDelegate, а не просто вказівник на низькорівневу адресу в пам'яті.
  - У C# делегати оголошуються за допомогою ключового слова delegate.
  - Вони критично важливі, коли потрібно забезпечити об'єкт можливістю перенаправлення виклику іншому об'єкту та формують основу архітектури подій .NET.

```
// Этот тип делегата в C# может "указывать" на любой метод,
// возвращающий значение int и принимающий два значения int.
delegate int BinaryOp(int x, int y);
```

Спільна система типів

Члени типів CTS

- Формально член типу обмежений набором {конструктор, фіналізатор, статичний конструктор, вкладений тип, операція, метод, властивість, індексатор, поле, поле тільки для зчитування, константа, подія}.
- У специфікації CTS описуються різні ***характеристики*** (adornments), які можуть з ним асоціюватись:
  - Наприклад, характеристика ***доступності*** (відкритий, закритий або захищений).
  - Деякі члени можуть оголошуватись як ***абстрактні*** (для забезпечення поліморфної поведінки похідними типами) або як віртуальні (для забезпечення фіксованої реалізації, яка допускає переозначення).
  - Більшість членів можуть конфігуруватись як ***статичні*** (пов'язані з рівнем класу) або члени екземпляру (пов'язані з рівнем об'єкта).

Спільна специфікація мов (CLS)

Специфікація CLS та забезпечення сумісності

- Это набор правил, подробно описывающих минимальное и полное множество характеристик, которые должен поддерживать отдельный компилятор .NET, чтобы генерировать программный код, обслуживаемый средой CLR и в то же время доступный в унифицированной манере всем языкам, ориентированным на платформу .NET.
  - Во многих отношениях CLS можно рассматривать как подмножество полной функциональности, определяемой CTS.
- компилятор C# можно заставить выполнять проверку кода на предмет совместимости с CLS, используя единственный атрибут .NET:
  - атрибут [CLSCompliant] заставляет компилятор C# проверять каждую строку кода на соответствие правилам CLS.
  - В случае обнаружения нарушений любых правил CLS компилятор сообщит об ошибке с указанием проблемного кода.

```
// Сообщить компилятору C# о необходимости проверки на совместимость с CLS.
[assembly: CLSCompliant(true)]
```

Збірки, типи та простори імен

- C# не поставляется с какой-то специфичной для языка библиотекой кода.
  - разработчики на C# пользуются нейтральными к языкам библиотеками .NET.
- Для поддержания всех типов внутри библиотек базовых классов в хорошо организованном виде в рамках платформы .NET широко применяется концепция пространства имен.
  - Пространство имен — это группа семантически связанных типов, которые содержатся в одной или нескольких связанных друг с другом сборках.
  - System.IO содержит типы, имеющие отношение к файловому вводу-выводу, System.Data — типы для работы с базами данных и т.д.
  - в одной сборке (вроде mscorlib.dll) может содержаться любое количество пространств имен, каждое из которых может иметь любое число типов.

Простори імен та збірка mscorlib.dll

Будь-яка .NET-мова використовує однакові простори імен та типи

Object Browser | Start Page

Browse: .NET Framework 4.6.1 ...

<Search>

- Microsoft.VisualC.STLCLR
- mscorlib
  - Microsoft.Win32
  - Microsoft.Win32.SafeHandles
  - System
  - System.Collections
  - System.Collections.Concurrent
  - System.Collections.Generic
  - System.Collections.ObjectModel
  - System.Configuration.Assemblies
  - System.Deployment.Internal
  - System.Diagnostics
  - System.Diagnostics.CodeAnalysis
  - System.Diagnostics.Contracts
  - System.Diagnostics.Contracts.Internal
  - System.Diagnostics.SymbolStore
  - System.Diagnostics.Tracing
  - System.Globalization
  - System.IO
    - BinaryReader
      - Base Types
      - Derived Types
    - BinaryWriter
    - BufferedStream

- BinaryReader(System.IO.Stream)
- BinaryReader(System.IO.Stream, System.Text.Encoding)
- BinaryReader(System.IO.Stream, System.Text.Encoding, bool)
- Close()
- Dispose()
- Dispose(bool)
- FillBuffer(int)
- PeekChar()
- Read()
- Read(byte[ ], int, int)
- Read(char[ ], int, int)
- Read7BitEncodedInt()
- ReadBoolean()
- ReadByte()
- ReadBytes(int)
- ReadChar()
- ReadChars(int)
- ReadDecimal()

public class **BinaryReader**
Member of System.IO

Summary:
Reads primitive data types as binary values in a specific encoding

```
// Приложение "Hello World" на языке C#
using System;

public class MyApp
{
  static void Main()
  {
    Console.WriteLine("Hi from C#");
  }
}
```

```
// Приложение "Hello World" на языке C++/CLI
#include "stdafx.h"
using namespace System;

int main(array<System::String ^> ^args)
{
  Console::WriteLine(L"Hi from C++/CLI");
  return 0;
}
```

```
' Приложение "Hello World" на языке VB.
Imports System

Public Module MyApp
  Sub Main()
    Console.WriteLine("Hi from VB")
  End Sub
End Module
```

Програмний доступ до просторів імен

Ключове слово using

- пространство имен — это не более чем удобный способ логической организации связанных типов для упрощения работы с ними.
- В C# ключевое слово using упрощает процесс добавления ссылок на типы, определенные в заданном пространстве имен.

```
// Некоторые возможные пространства имен, используемые
// для построения WPF-приложения.
using System;                    // Общие типы из библиотек базовых классов.
using System.Windows.Shapes;     // Типы для графической визуализации.
using System.Windows.Controls;   // Типы виджетов графического интерфейса
                                 // Windows Forms.
using System.Data;               // Общие типы, связанные с данными.
using System.Data.SqlClient;     // Типы доступа к данным MS SQL Server.
```

	Версия .NET Framework															
ОС	**1.0**	**1.1**	**2.0**	**3.0**	**3.5**	**4.0**	**4.5**	**4.5.1**	**4.5.2**	**4.6**	**4.6.1**	**4.6.2**	**4.7**	**4.7.1**	**4.7.2**	**4.8**
Windows 98	+															
Windows NT	+															
Windows Me	+															
Windows 2000	+	+	+													
Windows XP	+	+	+	+	+	+										
Windows Server 2003		+	+	+	+	+										
Windows Server 2008			+	+	+	+	+	+	+	+						
Windows Vista			+	+	+	+	+	+	+	+						
Windows 7			+	+	+	+	+	+	+	+	+	+	+	+	+	+
Windows Server 2008 R2			+	+	+	+	+	+	+	+	+	+	+	+	+	+
Windows Server 2012			+	+	+	+	+	+	+	+	+	+	+	+	+	+
Windows 8			+	+	+	+	+	+	+	+	+					
Windows 8.1			+	+	+	+	+	+	+	+	+	+	+	+	+	+
Windows Server 2012 R2			+	+	+	+	+	+	+	+	+	+	+	+	+	+
Windows 10			+	+	+	+	+	+	+	+	+	+	+	+	+	+

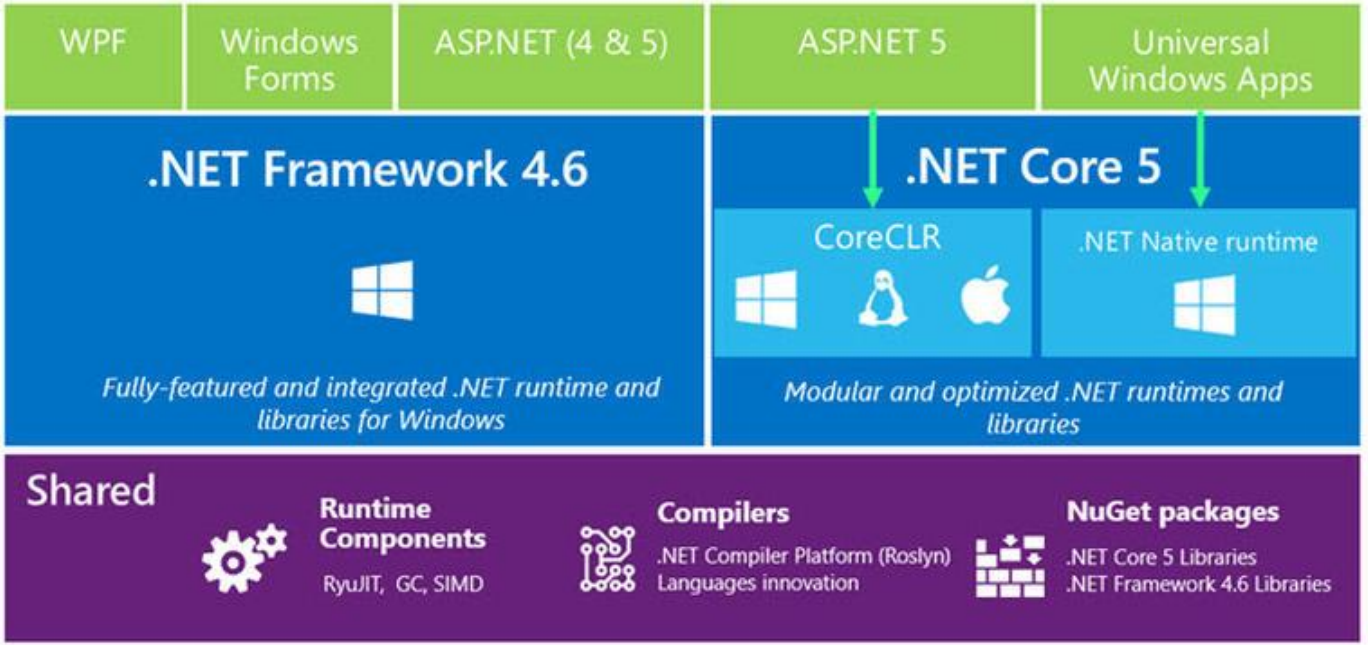
Тренд на кросплатформність

Проекти Mono та Xamarin

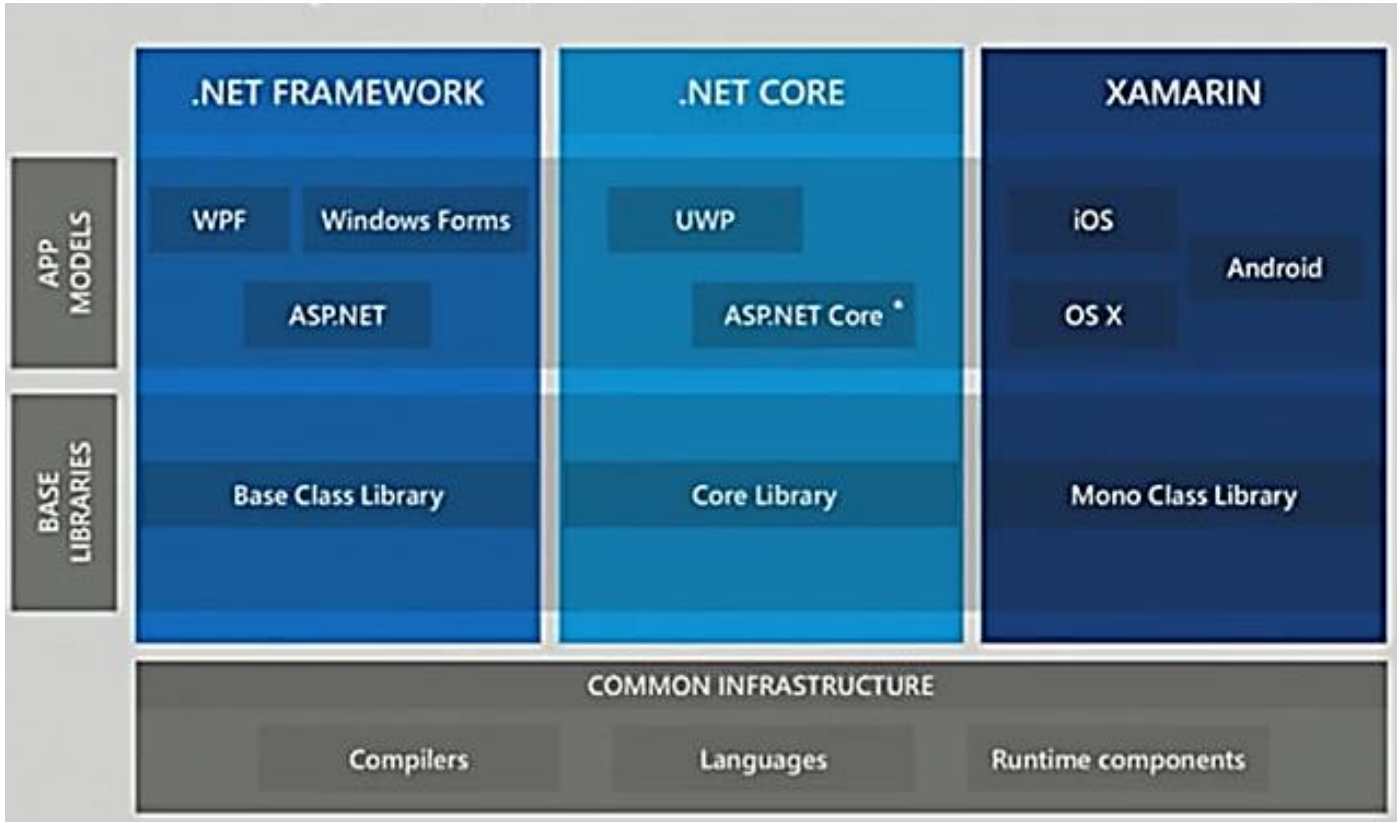
- Сторонние разработчики создали реализацию .NET под названием Mono.
  - Хоть Mono и был кросс-платформенным проектом, он значительно отставал от о фициальной реализации .NET Framework.
  - Позднее Mono занял нишу в качестве основы мобильной платформы Xamarin од ноименной компании.
- В 2016 году корпорация Microsoft приобрела компанию Xamarin и теперь бе сплатно предоставляет пользователям когда-то дорогостоящее решение Xa marin в качестве расширения для среды разработки Visual Studio 2017.
  - Кроме того, Microsoft переименовала инструменты для разработчика Xamarin St udio в Visual Studio для Mac и внедрила возможность создавать веб-API с помощ ью ASP.NET Core.
  - Проект Xamarin нацелен на создание мобильных приложений и облачных серви сов для их поддержки.
- Современные методы мобильной и облачной разработки уменьшили прежн юю значимость операционной системы Windows. Поэтому Microsoft активно

.NET у 2015 році

Поява .NET Core



.NET Framework 4.8 – остання версія .NET Framework


.NET FRAMEWORK
.NET CORE
XAMARIN
APP MODELS
BASE LIBRARIES
WPF
Windows Forms
ASP.NET
UWP
ASP.NET Core
iOS
Android
OS X
Base Class Library
Core Library
Mono Class Library
COMMON INFRASTRUCTURE
Compilers
Languages
Runtime components

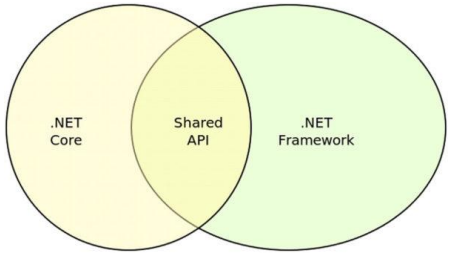
.NET Core vs .NET Framework

К примеру, компоненты Windows Forms и Windows Presentation Foundation (WPF) могут служить для создания приложений с *графическим п ользовательским интерфейсом* (graphical user interface, GUI), но тесно связаны с Windows, вследствие чего были удалены из .NET Core. В настоящее время для проектирования приложений Windows применяется технология под названием «Универсальная платформа Windo ws» (Universal Windows Platform, UWP), внедренная в .NET Core.

Компоненты ASP.NET Web Forms и Windows Communication Foundation (WCF) — устаревшие технологии для создания веб-приложений и с ервисов, используемые сегодня лишь некоторыми разработчиками, так что эти компоненты тоже были удалены из .NET Core.
Вместо них разработчики предпочитают компоненты ASP.NET MVC и ASP.NET Web API.
Две последние технологии были реорганизованы и объединены в новый продукт, ASP.NET Core, работающий на платформе .NET Core.

Entity Framework (EF) 6 — технология объектно-реляционного отображения для работы с информацией, хранящейся в реляционных базах данных, таких как Oracle и Microsoft SQL Server. За годы развития данная технология накопила приличный багаж дополнений, вследствие чего кросс-платформенная версия была оптимизирована и приобрела название Entity Framework Core.

Помимо удаления крупных компонентов из .NET Framework при подготовке платформы .NET Core, Microsoft распределила компоненты .NE T Core в виде небольших NuGet-пакетов, которые могут быть развернуты независимо друг от друга.



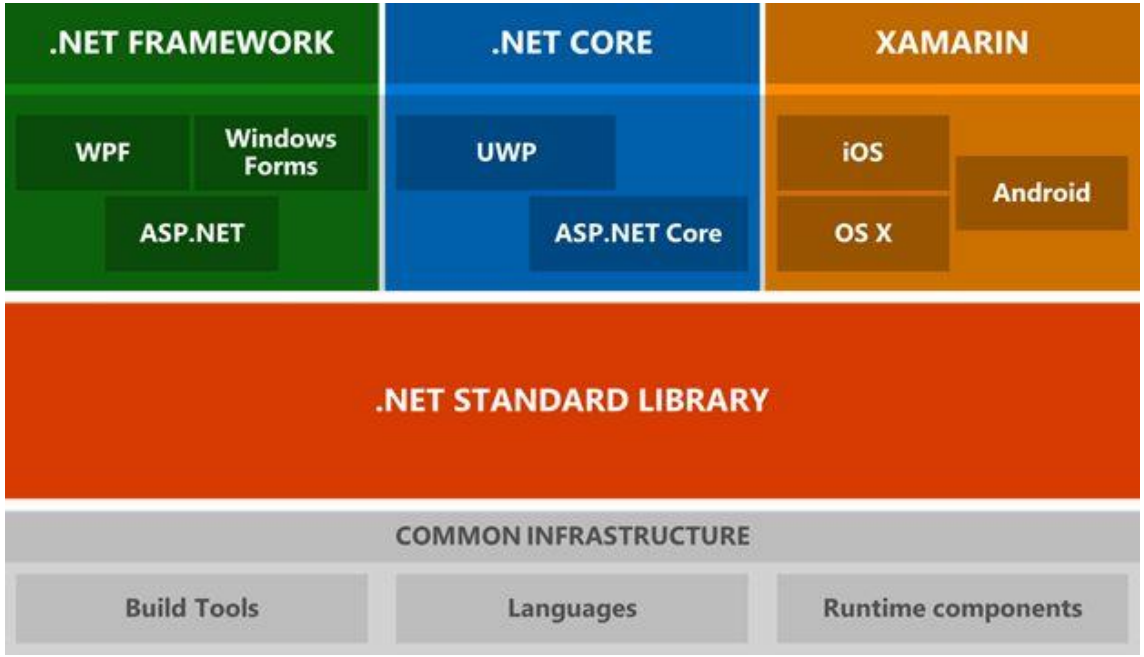
Еволюція .NET Core

Version	Original Release Date	Latest Patch Version	Patch Release Date	Support Level	End of Support
.NET Core 3.1	Scheduled for November 2019			Will be LTS when released	
.NET Core 3.0	Scheduled for September 23, 2019			Will be Current when released	
.NET Core 2.2	December 4, 2018	2.2.6	July 09, 2019	Current	December 23, 2019
.NET Core 2.1	May 30, 2018	2.1.12	July 09, 2019	LTS	At least three years from LTS declaration (August 21, 2018)
.NET Core 2.0	August 14, 2017	2.0.9	July 10, 2018	EOL	October 1, 2018
.NET Core 1.1	November 16, 2016	1.1.13	May 14, 2019	EOL	June 27 2019
.NET Core 1.0	June 27, 2016	1.0.16	May 14, 2019	EOL	June 27 2019

Платформа .NET Core стала существенно меньше, чем текущая версия .NET Framework, так как многие компоненты были удалены.
Підтримуються .NET-мови: C#, F#, VB (частково)

Технологія .NET Standard

Сучасний стан платформи .NET



Xamarin is not a debate at all. When you want t o build mobile (iOS, Android, and Windows Mobi le) apps using C#, Xamarin is your only choice.

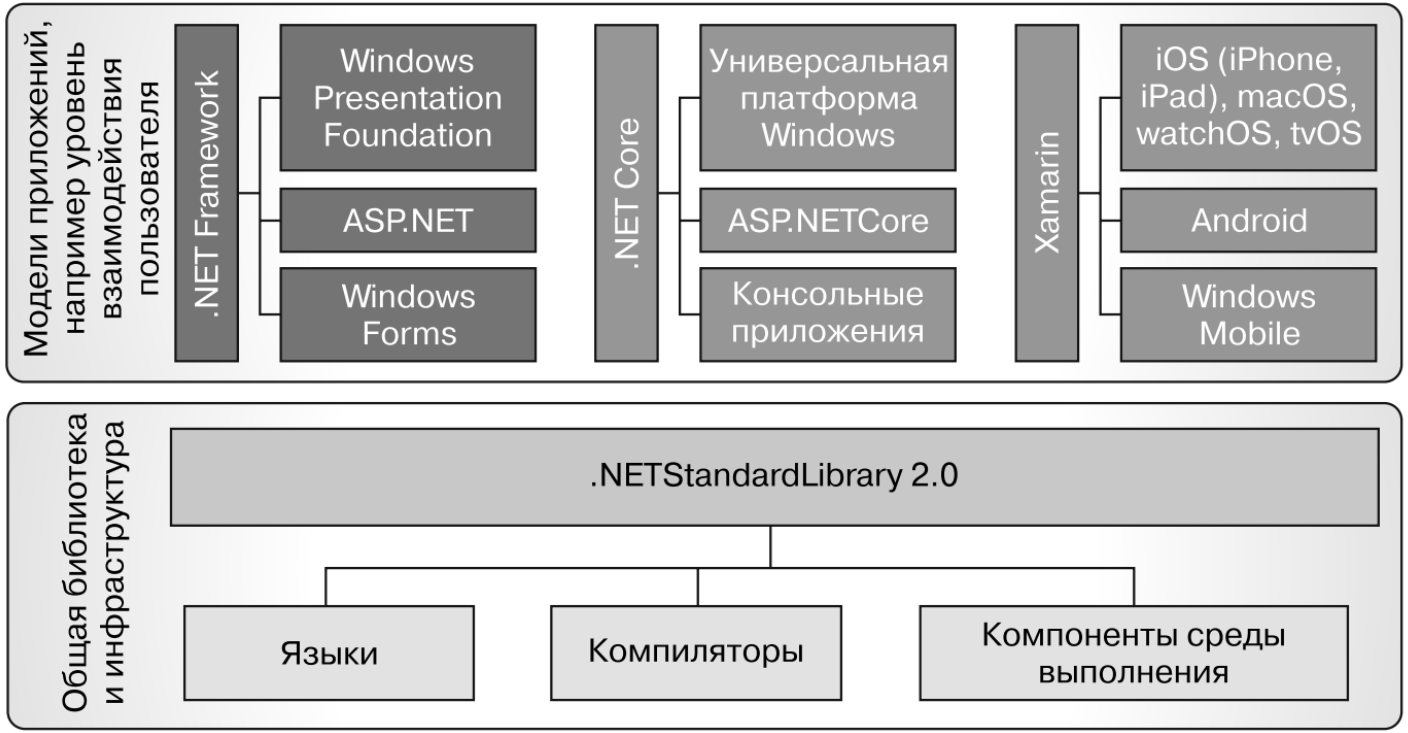
The .NET Framework supports Windows and Web applications. Today, you can use Windows Forms, WPF, and UWP to build Windows applic ations in .NET Framework. ASP.NET MVC is us ed to build Web applications in .NET Framework .

.NET Core is the new open-source and cross-pl atform framework to build applications for all ope rating systems including Windows, Mac, and Lin ux. .NET Core supports UWP and ASP.NET Cor e only. UWP is used to build Windows 10 targets Windows and mobile applications. ASP.NET Cor e is used to build browser based web application s.

Технологія .NET Standard

Специфікація спільного для всіх .NET-платформ набору API

для применения .NE
T Standard 2.0 следу
ет установить платф
орму .NET, реализую
щую данную специф
икацию.
NET Standard 2.0 реа
лизована в последни
х версиях .NET Fram
ework, .NET Core и X
amarin.
Она значительно упр
ощает процесс совм
естной работы над к
одом на разных плат
формах .NET.



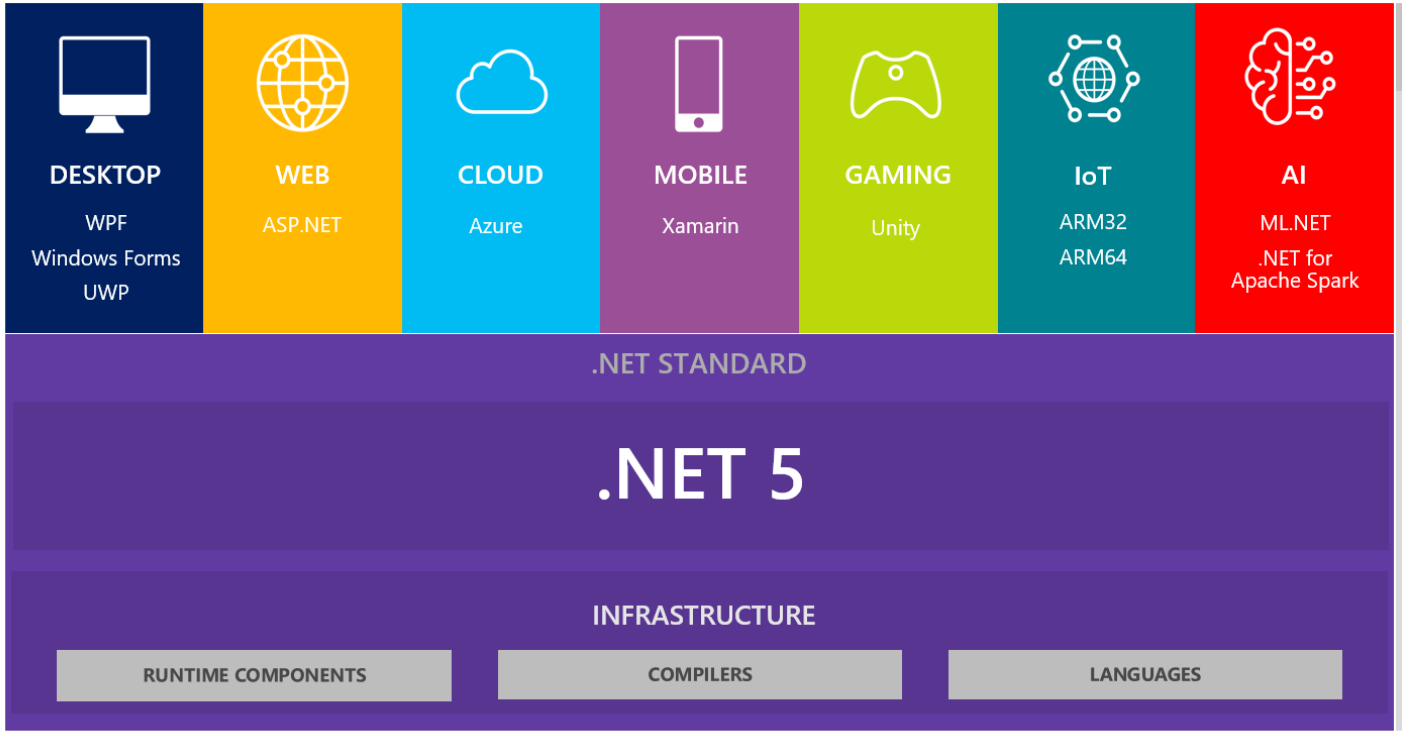
Технологія .NET Native

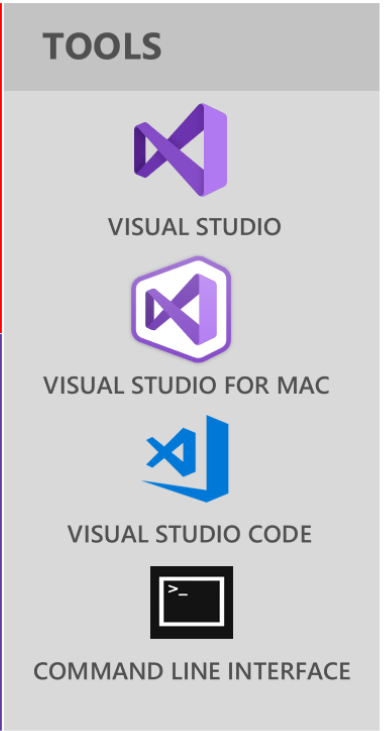
Другой инициативой в области .NET является технология .NET Native. С ее помощь
ю код на языке C# компилируется в машинный код методом компиляции перед испо
лнением (ahead-of-time, AoT) вместо общеязыковой исполняющей среды, применяе
мой для динамической компиляции (just-in-time, JIT) в промежуточный язык
(intermediate language, IL) и только затем в машинный язык.
Технология .NET Native увеличивает скорость выполнения приложений
и уменьшает выделяемые объемы памяти.
Поддерживаются следующие типы приложений:

- UWP-приложения для систем Windows 10, Windows 10 Mobile, Xbox One, HoloLe
  ns и устройств «Интернет вещей» (Internet of Things, IoT) типа микрокомпьютера
  Raspberry Pi;
- серверная веб-разработка с помощью ASP.NET Core;
- консольные приложения для использования в командной строке.

Уніфікована платформа .NET

Погляд у майбутнє





Плани щодо еволюції .NET

- .NET Core 3.0 release in September
- .NET Core 3.1 = Long Term Support (LTS)
- .NET 5.0 release in November 2020
- Major releases every year, LTS for even numbered releases
- Predictable schedule, minor releases if needed