.NET технологии. Архитектура и особености на Microsoft .NET Framework/. Net Core

Съдържание на лекцията

- Развитието на софтуерните технологии и Microsoft.NET
- Архитектура и особености на .NET Framework
  - Framework Class Library
  - Common Language Runtime (CLR)
- Езикът C#
- Заключение

Развитието на софтуерните технологии и Microsoft.NET

- Спокойно можем да твърдим, че .NET технологиите (най-мощните продукти на Microsoft, съпровождани с окончание .Net) са изпълнение на най-модерните идеи в развитието на софтуерните технологии:
  - Разработка на компонентни приложения, допускащи пре-използване;
  - Осъществяване на отворени стандарти за развитие;
  - Повишаване ролята на езика XML и използването му за различни цели
  - и др.

- Това са технологии, създадени с идеята да облекчат в най-висока степен **създаването на архитектурата** и **кодирането на програмните приложения** като осигуряват абстрахиране, позволяващо на разработчика съсредоточаване върху задачата, а не върху особеностите на операционното и апаратно осигуряване.

.NET Framework/ .NET Core

- .NET Framework/ **.NET Core** е създадена от Microsoft платформа, предоставяща:
  - програмен модел,
  - библиотека от класове (FCL,Framework Class Library/ **CoreFX**) и
  - среда за изпълнение на написан специално за нея програмен код (CLR, Common Language Runtime/**.NET Core Runtime**).
- Microsoft .NET Framework 1.0 е първата официално пусната версия на .NET Framework. Това се случи в началото на 2002.
- От средата на август 2012 са налице .NET Framework 4.5 и Visual Studio 2012.
- От юли 2015 - .NET Framework 4.6 и Visual Studio 2015 и **C#6.0,**
- **2016 –** NET.Core 1.0
- 2017 - NET.Core 2.0 със C#7.0,
- септември **2019: .Net Core 3.0, Visual Studio 2019, C#8.0**

Версия	CLR	Версия - номер	Година	Visual Studio	в Windows	Заменя
1.0	1.0	1.0.3705.0	2002	Visual Studio .NET C# 1.0	-	-
1.1	1.1	1.1.4322.573	2003	Visual Studio .NET 2003	Windows Server 2003	1.0
2.0	2.0	2.0.50727.42	2005	Visual Studio 2005 C# 2.0	Windows Vista, Windows 7, Windows Server 2008 R2	-
3.0	2.0	3.0.4506.30	2006	Visual Studio 2005 + расширения	Windows Vista, Windows Server 2008, Windows 7, Windows Server 2008 R2	2.0
3.5	2.0	3.5.21022.8	2007	Visual Studio 2008 C# 3.0	Windows 7, Windows Server 2008 R2	2.0; 3.0
4.0	4	4.0.30319.1	2010	Visual Studio 2010 C# 4.0	Windows 8, Windows Server 2012	-
4.5	4	4.5.50709.17929	2012	Visual Studio 2012 C# 5.0	Windows 8, Windows Server 2012	4.0
4.5.1	4	4.5.50938.18408	2013	Visual Studio 2013	Windows 8.1, Windows Server 2012 R2	4.5
4.5.2	4	4.5.51209.34209	2014			4.0; 4.5; 4.5.1
4.6	4		2015 20.юли	Visual Studio 2015 C# 6.0	Windows 10, Windows Server 2016	4.0, 4.5, 4.5.1, 4.5.2

.Net Core 1.0 (2016), Visual Studio 2017 (16 November 2016)
.Net Core 2.0 (2017), Visual Studio 2017, C# 7.0
.Net Core 3.0 (2019); Visual Studio 2019; C# 8.0

.NET Framework/ поддържа езиците

- –Visual C++.Net,
- –Visual C#.Net,
- – J#.Net (във версия 2.0),
- –Visual Basic.Net и
- –F# (във версии 4.0,4.5,4.6)

.NET Framework се разпространява със съответната версия на Windows Server за съответния Windows, като може да бъде инсталиран и на по-стари версии на Windows.

Visual Studio.Net е:

- Един, намиращ се в непрекъснато развитие продукт на Microsoft (както се вижда от показаната таблица), предоставящ интегрирана среда за разработка (IDE, Integrated Development Environment) на софтуер за .NET Framework.
- Позволява както разработване на конзолни приложения, така и на приложения с графичен потребителски интерфейс, в това число desktop приложения, web приложения и услуги, приложения за мобилни телефони и др.

Visual Studio.Net е:

- **Отворена езикова среда** за разработка на софтуер. Отворена среда не означава пълна свобода!
- Всички разработчици на езикови компилатори, се задължават, при включване на нов език, да спазват определени ограничения. Главното ограничение, което може да се счита и за главно достойнство на стратегията за развитие на Microsoft, се състои в това, че всички езици, включени в средата за разработка Visual Studio .Net, трябва да използват единната платформа .NET Framework. Благодарение на това се постигат много желаните свойства:
  - възможност за интегриране в едно приложение на компоненти, написани на различни езици;
  - възможност един клас, написан на един език, да бъде наследен и разширен от клас, написан на друг език.

.NET Framework се счита за:

“революция в разработката на софтуер”!

- Още при **.NET Framework 1.0** се обединиха няколко важни възможности, които до онзи момент съществуваха само при Java (Java 2 Enterprise Edition), а именно:
  - Принципно нов подход (в сравнение с Visual Studio 6.0) при построяване, базиран на **виртуална машина**
  - Работа със **силно типизирани данни**
  - Улеснено управление на паметта, чрез „събирач на боклук“ (**Garbage Collector**)
  - Управление на грешки, базирано на **обработване на изключения**
  - **Обширни библиотеки**, които покриват голям диапазон от функционалност.

.NET Framework 2.0

- Поддръжка на частични класове (**partial classes**) – позволяват **разделянето на един клас в няколко файла**, например клас Cylinder (описващ геометричната фигура „цилиндър”) е дефиниран в 2 файла: Cyl1.cs и Cyl2.cs по следния начин:

```
partial class Cylinder
{
private double rad;
private double hgt;
  …
}
```

Cyl1.cs

```
partial class Cylinder
{ …
public double Volume()
{ … }
}
```

Cyl2.cs

.NET Framework 2.0 предлага още:

- **Генерични типове (generics),** най - ефективни при работа с колекции
- **Поддръжка на итератори**: улеснена реализацията на метод GetEnumerator() при контейнерните типове
- Ключова дума **yield**, която създава итератори
- **Анонимни методи** (методи, без изрично зададен идентификатор), изключително полезни за работа с делегати и събития и др.

.NET Framework (3.0, 3.5 и 4.0) и Visual Studio.Net :

предлагат множество нови средства, позволяващи повишаване удобството на работа и увеличаване на производителността на софтуерните инженери:

- **Поддръжка на LINQ** – Language InNtegrated Query (Заявка, Вградена в езика), тоест поддръжка на синтаксис за управление на заявки
- **Подобрена инициализация** на обекти и на колекции
- **Подразбиране на типа на променливите**: въвеждат се с ключова дума var и типът им е според присвоената стойност: var i = 10;
  // implicitly typed

Още:

- Поддръжка на ламбда изрази: компилаторът превежда ламбда изразите към силно типизирани делегати или към дървета от изрази (expression trees)
- Автоматично реализиране на свойства
- Частични методи: тук тези методи са добавени към частичните класове на .NET Framework 2.0
- Разширяеми методи и др.

.NET Framework 4.0 предлага:

- нов тип - тип dynamic, имащ за идея улесняване на взаимодействието с COM обекти и типове, създадени от динамични .Net езици.
- нов език е официално включен във Visual Studio.Net 2010 – езикът F#.

.NET Framework 4.5:

- акцентира върху Metro стила за изграждането на приложения за Windows 8.
- Поддържа новите HTML5 типове, Unicode (UTF-16) кодиране за конзола, подобрение на ZIP архивиране и др.

.NET Framework 4.6

- **Поддържа нов JIT-компилатор** за 64-разрядни системи (RyuJIT);
- **WPF и WinForms са обновени за поддръжка на екрани с висока разделителна способност** (висок DPI);
- в WCF (Windows Communication Foundation - WCF) е добавена поддръжка на TLS 1.1 и TLS 1.2. (TLS - Transport Layer Security)
- Криптографският API (Application Programming Interface) в .NET Framework 4.6 използва последната API от Microsoft CryptoAPI, благодарение на което стана достъпен набор от алгоритми за шифриране «Suite B» — AES, SHA-2, Elliptic curve Diffie-Hellman, ECDSA[16].

.NET Core

- Чрез open-source проекта "**.NET Core**" на фондацията (dotnetfoundation.org) през 2014 г. . езикът **C#** и **платформата .NET** (т.е **.NET Core**) постепенно се отварят и за външния свят:
  - C# става многоплатформен **език с отворен код**.
  - .NET имплементация **.NET Core** е вече **крос-платформена** и **свободна за изтегляне** (може да се изтегля от сайта на Microsoft: https://microsoft.com/net.) и е също **с отворен код**.
- Дотогава, езикът C# и .NET платформата (.Net Framework) не са достатъчно разпространени, тъй като такива софтуерни гиганти като Google, Apple, IBM, Oracle, Facebook и SAP предпочитат да базират своите решения на отворени езици и платформи като Java, JavaScript, Go, Python и PHP. С .NET Core ситуацията вече се променя и **C# и .NET стават водещи в софтуерната разработка** глобално.

- Забележка: **Софтуер с отворен код** (*open-source software*) е софтуер, за който притежателят на авторските права на изходния код предоставя правата за обучение, промяна и разпространение на софтуера на всекиго и за всякакви цели.

.NET Core

- **.NET Core** — това е универсална, модулна платформа за разработка на софтуер, с отворен изходен код. Създадена е и се поддържа от компанията Microsoft и съобществото .NET на сайте GitHub. Явява се кросплатформена, вече съвместима както с **Windows** такаи с **macOS и Linux** и може да се използва за създаване на приложения за различни устройства, за облака и за Интернет на вещите.
- Поддържа следните езици: C#, Visual Basic .NET (частично) и F#.
- .NET Core 3.0 (със C# 8.0) е последната версия на .NET Core, пусната 2019 (.NET Core 1.0 е пусната 2016г., .NET Core 2.0 е пусната 2017г. със C# 7.0).

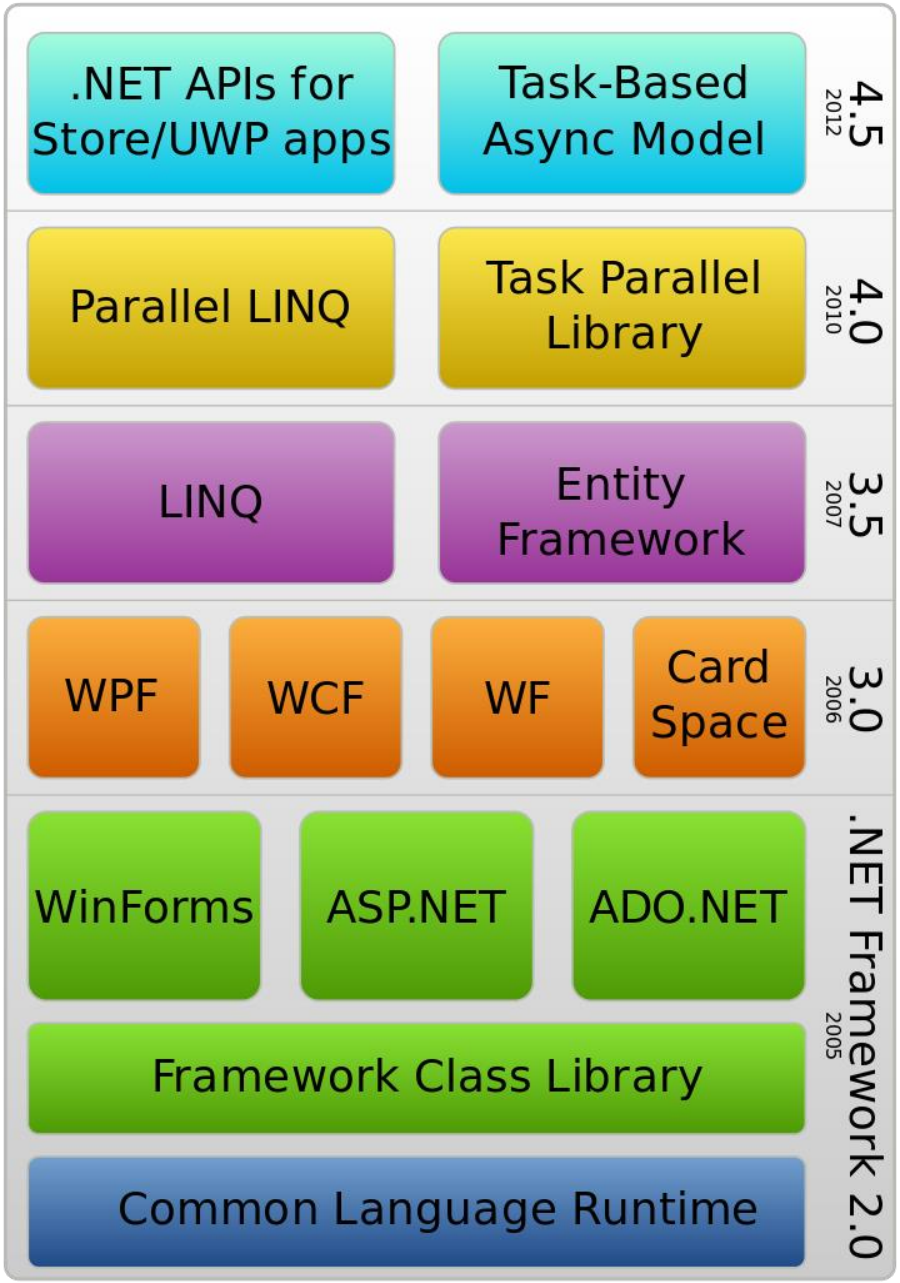
.NET Core

Version	Release Date	Released with
.NET Core 1.0[17]	2016-06-27	Visual Studio 2015 Update 3
.NET Core 1.1[18]	2016-11-16	Visual Studio 2017 Version 15.0
.NET Core 2.0[19]	2017-08-14	Visual Studio 2017 Version 15.3
.NET Core 2.1[20]	2018-05-30	Visual Studio 2017 Version 15.7
.NET Core 2.2[21]	2018-12-04	Visual Studio 2017 Version 15.9
.NET Core 3.0[22]	2019-09-23[13]	Visual Studio 2019 Version 16.3
.NET 5[23]	2020-11	

Архитектура и особености на .NET Framework/ .NET Core

- .NET Framework/**.Net Core** се състои от 2 основни компонента:
  - статична компонента - **FCL** (Framework Class Library/ **CoreFX**): библиотека от класове на .Net Framework/ .Net Core;
  - динамична компонента - **CLR** (Common Language Runtime/ **CoreCLR**): обща среда за изпълнение на .Net Framework/ .Net Core. Благодарение на нея **CLR програмите са преносими** и след като веднъж бъдат написани, могат да работят почти без промени върху различни хардуерни платформи и операционни системи.

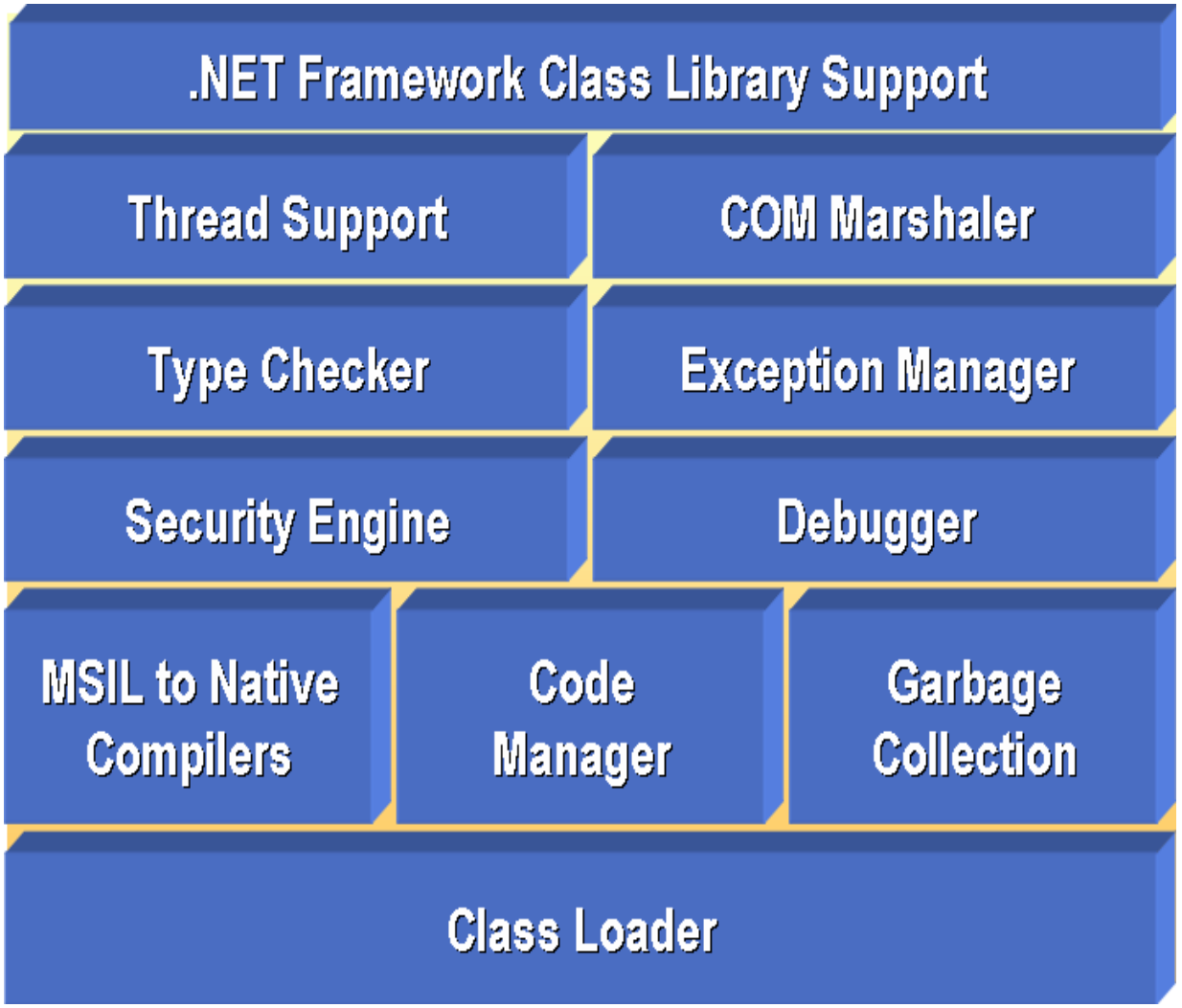
Архитектура на .NET Framework



Common Language Runtime (CLR)

- Отделните слоеве на .NET Framework лежат върху операционната система, която изпълнява .NET приложенията - най-често Microsoft Windows, но .NET Framework има имплементации и за други операционни системи
- Следва Common Language Runtime (CLR) – това е програмната среда, в която се изпълнява кодът на .NET приложенията. Представлява виртуална машина, която:
  - зарежда (чрез модул „Class Loader”) и компилира (модул „JIT Compilers”) междинния IL (Intermediate Language) код до машинен код (JIT компилация), при това съобразено с конкретната хардуерна платформа и операционна система, с която работи потребителят;
  - управлява (модул „Code Manager”) изпълнението на кода.
- CLR не е интерпретатор. Веднъж компилиран, CLR съхранява компилирания код и го използва наготово, когато е извикан, без повторно компилиране.
- В допълнение CLR осигурява редица услуги, като:
  - управление на паметта, включително автоматично "събиране на боклуци" (garbage collection),
  - управление на прекъсванията, многонишковост (multithreading) и др.

Архитектурата на CLR



Модули на CLR:

- MSIL to Native Compilers („JIT Compilers”) – това е един от най-важните модули на CLR. По време на изпълнение на програмата се извършва компилиране на IL кода в специфичен за процесора код („native” код).
- Модул „Class Loader” служи за зареждане на класове и типове. Използва се при началното изпълнение на приложението, както и при динамично зареждане на код по време на изпълнение.

Модули на CLR:

- Модул „Code Manager” управлява изпълнението на кода.
- Модул „Net Framework Class Library Support” предоставя системни услуги, необходими за работата на Base Class Library (BCL).
- Модул „Thread Support” предоставя услуги за работа с нишки в .NET приложенията: създаване на нишка, управление на състоянието на нишка, синхронизация и др.

Модули на CLR:

- Модул „COM Marshaler” се грижи се за комуникацията с COM обекти. Осигурява извикването на COM сървъри от .NET код и извикването на .NET код от COM. Негова грижа са прехвърлянето на заявки, преобразуването на данни, управлението на жизнения цикъл на COM обектите и др.

Модули на CLR:

- Модул „Type Checker” се грижи за безопасността на типовете: .NET Framework е среда за контролирано изпълнение на програмен код, тя не позволява директен достъп до паметта, не позволява преобразуване от един тип към друг, който не е съвместим с него, не позволява излизане от границите на масив, както и други, считани за „опасни” операции. По тази причина .NET Framework се нарича „управлявана среда за изпълнение” (managed execution environment) на кода, за разлика от традиционните „неуправлявани среди”.

Модули на CLR:

- Модул „Exception Manager” се грижи се за управление на изключенията: предизвикване на изключение, прихващане, обработване и др.
- Модул „Security Engine” отговаря за проверките на сигурността при изпълнение на кода.
- Модул „Debug Engine” осигурява функционалност, свързана с дебъгването и оптимизирането на управляван код.
- Модул „Garbage Collector” управлява паметта и ресурсите и се грижи за автоматичното им почистване. Контролира живота на обектите.

FCL (Framework Class Library)/**CoreFX**

- FCL (Framework Class Library)/**CoreFX** е обща за всички .Net езици библиотека от класове, предоставяща богата функционалност.
- Тези класове (повече от 20000) са организирани логически в йерархия от именовани пространства.
- Едно именовано пространство съдържа както класове, така и вложени именовани пространства.

FCL (Framework Class Library)/ CoreFX

Ето някои основни именовани пространства от FCL и тяхното предназначение:

- **System – съдържа основни типове, използвани от всяко .NET приложение. В пространството System се намира, например, базовият за всички типове в .NET Framework клас System.Object, както и класът System.Console, който позволява вход и изход от конзолата.**
- **System.Collections, System.IO и System.Security, са именовани пространства, вложени в System.**
- **Типовете в езиците за програмиране се проектират върху съответстващите типове от FCL. Така например, типът Integer на Visual Basic и съответно тип int на C# се проектират в един и същи тип на FCL, а именно клас System.Int32.**
- **В System.Collections се разполагат често използвани типове за управление на колекции от обекти.**
- **System.IO съдържа типовете, които осигуряват входно-изходните операции с потоци в .NET Framework.**

FCL (Framework Class Library)

- System.Xml – съдържа типове за работа с XML и технологиите, свързани с него.
- System.Web съдържа именовани пространства System.Web.UI и System.Web.Services, поддържащи съответно разработката на уеб приложения и уеб услуги.
- System.Windows.Forms съдържа типове, използвани при създаването на Windows приложения с графичен потребителски интерфейс.

FCL (Framework Class Library)

- System.Drawing осигурява основните типове за разработка на графика.
- System.Data е именовано пространство, съдържащо цялата функционалност даваща възможност за работа с бази от данни. Това пространство по нататък се дели на System.Data.SqlClient, пространство което предлага функционалност специфична за SQL Server, и System.Data.OleDb, което предлага функционалност за достъп до OLEDB data sources.

FCL - Основни библиотеки

- Base Class Library – стандартна библиотека от класове, съдържаща основните класове използвани при разработка на приложения: за организация на вход и изход, многозадачност, колекции, символни низове, достъп до мрежови ресурси, сигурност, отдалечено извикване и други.
- ADO.NET и XML – библиотека на .Net за работа с бази от данни и средства за обработка на XML.

FCL - Основни библиотеки

- ASP.NET – предоставя класове за разработка на web приложения с богата функционалност, както и средства за създаване и консумиране на web услуги.
- Windows Forms – библиотека на .Net за работа с прозоречно-базиран графичен потребителски интерфейс.
- LINQ (Language Integrated Query) — библиотека на .Net, която добавя в .Net езиците за програмиране синтаксис на език за заявки, наподобяващ SQL.

FCL - Основни библиотеки

- Windows Presentation Foundation (WPF) — графична (презентационна) подсистема в състава .NET Framework (от версия 3.0 нагоре). Позволява изграждане на Windows-базирани Desktop приложения с привлекателни интерфейси ([2]). Всяко WPF приложение има поне 2 слоя: презентационен (интерфейса на приложението) и code-behind (слой, съдържащ самата програмна логика).
- Windows CardSpace — .NET Framework компонента, която предоставя технология за унифицирана идентификация на потребителите: при разместване на Интернет ресурсите, не се изисква повторно въвеждане на име и парола (от версия 3.0 нагоре).

FCL - Основни библиотеки

- Windows Communication Foundation (WCF) е компонента в .NET Framework, задаваща рамката за разработка на свързани, ориентирани към услуги приложения (от версия 3.0 нагоре).
- ADO.NET Entity Framework (EF) — е обектно-ориентирана технология за достъп до данните, едно object-relational mapping (ORM) решение. Предоставя возможност за взаимодействие с объектите както посредством LINQ във вид LINQ to Entities, так и с използване на Entity SQL (от версия 3.5 нагоре).

FCL - Основни библиотеки

- Windows Workflow Foundation (WF) е технология на Microsoft за определяне, изпълнение и управление на работните процеси (от версия 3.0 нагоре).
- Включената в NET Framework 4.0 компонента Parallel LINQ (PLINQ) е част от разширенията, свързани с реализацията на паралелизъм (Parallel Extensions) на .NET и се явява паралелна имплементация на LINQ to Objects и LINQ to XML. Съответно Task Parallel Library (TPL) е паралелна компонента, касаеща паралелна имплементация на задачите.

Езикът C#

- C# е **нов**, **обектно-ориентиран**, **силно типизиран език** от **високо ниво** с **общо предназначение**, специално проектиран за .NET Framework (официално пуснат през 2002г.), така че да използва по най-добрия начин нейните възможности.
- Проектиран е от Андерс Хейлсберг (Anders Hejlsberg), водещ проектант на Turbo Pascal и Delphi.

https://en.wikipedia.org/wiki/C_Sharp_(programming_language)

Езикът C#

- Езикът беше анонсиран от създателя си (1999÷2002), като мощно средство за разработване на високопроизводителни **конзолни**, **Desktop** и **Web приложения**, компоненти и услуги.
- Той произлиза от C++, но:
  - съчетава мощта и ефективността на C++ с най-доброто от езиковите спецификации на Visual Basic и Object Pascal и от програмната среда Delphi.
  - C# заимства много и от Java, но базиран на богатия натрупан опит с други езици и среди е с много по-големи възможности от Java особено при разработване на визуални приложения.

Езикът C# към C++

- Тъй като по идея C# е проектиран да бъде лесен за използване, той наследява от C++ само част от синтаксиса и някои негови най-силни страни, като например предефинирането на оператори.
- В C# са премахнати указателите, хедърните файлове, някои считани за неефективни и доста объркващи оператори от C++ (като “->”, с който се обозначава принадлежност към структури).
- Вместо това - в C# се използва стандартният начин за обозначаване на вложени имена, както е например в Pascal, с dot оператора (“.”).

Езикът C# към C++

- За подобряване на сигурността, в езика C# целочислените и булевите типове данни са напълно различни типове. Това означава,че грешно присвоени данни в един оператор, веднага ще бъде санкционирано от компилатора.
- Тъй като указателите не са част от наличните инструменти на езика, управлението на паметта не е задължение на програмиста – то е поверено на така наречения „garbage collector”, който е отговорен за управлението на паметта.
- Основавайки се на обектно ориентираната концепция, в C# няма глобални функции, променливи и константи. Всичко трябва да се капсулира в класа. С цел типова безопасност, C# не поддържа класове с множествено наследяване, не е невъзможно да се използват неинициализирани променливи, да се надхвърлят границите на масивите и други.

C#

- **C# 1.0** released with .NET 1.0 and VS2002 (January 2002)
- **C# 1.2** (bizarrely enough); released with .NET 1.1 and VS2003 (April 2003). First version to call Dispose on IEnumerators which implemented IDisposable. A few other small features.
- **C# 2.0** released with .NET 2.0 and VS2005 (November 2005). Major new features: generics, anonymous methods, nullable types, iterator blocks
- **C# 3.0** released with .NET 3.5 and VS2008 (November 2007). Major new features: lambda expressions, extension methods, expression trees, anonymous types, implicit typing (var), query expressions
- **C# 4.0** released with .NET 4 and VS2010 (April 2010). Major new features: late binding (dynamic), delegate and interface generic variance, more COM support, named arguments and optional parameters
- **C# 5.0** released with .NET 4.5 and VS2012 (August 2012). Major features: async programming, caller info attributes. Breaking change: loop variable closure.
- **C# 6.0** released with .NET 4.6 and VS2015 (July 2015). Implemented by Roslyn. Features: initializers for automatically implemented properties, using directives to import static members, exception filters, binary literals and digit separators, indexed members and element initializers, await in catch and finally, extension Add methods in collection initializers.
- **C# 7.0** излиза с .NET Core 2.0 - 2017г. (.NET Core 1.0 е пусната 2016г.)
- **C# 8.0** излиза с .NET Core 3.0 - последна версия на .NET Core, пусната 2019.

- През 2014 г. се появява **платформата с отворен код .NET Core**, която пренася официално C# разработката върху **Linux** и **macOS**.
- Оттогава C# и .NET платформата се развиват в духа на **отворения код** и се радват на растяща общност от **open-source** разработчици и съмишленици, които допринасят към проекта в GitHub: https://github.com/dotnet/core.
- Поради отварянето си към много широка общност разработчици и технологични доставчици днес C# вече не е “технологията на Microsoft”, а се развива независимо чрез .NET фондацията (dotnetfoundation).

Заключение:

.NET Framework е ключовият елемент от стратегията за развитие на най-новите софтуерни технологии на Microsoft, имащи окончание .Net. От появата си през 2000 година досега.NET Framework се стреми към осигуряване на:

- **Платформена и хардуерна независимост** - създаване на приложения, които могат да работят на различни операционни системи и хардуерни конфигурации
  - Microsoft има имплементации на .NET Framework за x86 и x64 персонални компютри, за Windows CE базирани устройства - PDA (Personal Digital Assistant - Персонален асистент, тоест малък преносим компютър без собствена клавиатура- с писалка), SmartPhone, Tablet PC, както и за игровата конзола XBox 360. Спецификациите на CLI (Common Language Infrastructure - общоезиковата инфраструктура) касаещи BCL (Base C), CTS (Common Type System) и CIL (Common Intermediate Language), както и езиците C# и C++/CLI са предадени на организациите за стандартизация ECMA и ISO и са приети за отворени стандарти. По този начин независими разработчици могат да поставят .NET Framework и прилежащите му езици на други софтуерни и хардуерни платформи. .NET Framework се превърна в свободно распространяема виртуална машина. Това съществено разширява сферата на неговото използване.

- **Съвместимост с вече съществуващите технологии**: . Net е мост между традиционния неуправляван и управляван („managed”) стил на програмиране: програмистът има възможност за достъп до функционалност, реализирана в приложения извън .NET средата, тоест до „native” код.
- **Езикова независимост**: Приложения за .NET Framework могат да бъдат създавани на няколко съвместими програмни езици, сред които C#, VB.NET и C++/CLI.
  - Това е възможно благодарение на съвместимостта на типовете данни, които отделните езици поддържат. CTS (Common Type System) дефинира всички базови типове данни, както и начинът, по който те могат да бъдат конвертирани един в друг. Тези типове са споделени между всички .NET езици и са стандартизирани в CLI.
- **Сигурност**: Програмният код, написан на .NET, се нарича **управляван код**, а също и **защитен код**, тъй като е изолиран от хардуерната среда, в която оперира, и това го предпазва от някои видове програмни грешки, които правят кода уязвим за атаки, например препълване на буфера (buffer overflow).

- **Унифицирано, ефективно и безопасно изпълнение на кода**, чрез CLR (обща изпълнителна среда):
  - **Програмните езици**, които се поддържат от .NET Framework **биват компилирани в** междинен код известен като **Intermediate Language код** (CIL -Common Intermediate Language, а по-рано MSIL - Microsoft Intermediate Language).
  - **При изпълнението** си този междинен код **не се интерпретира** както това се случва при други виртуални машини, например тази на Java, а вместо това се **компилира от CLR по начин**, известен като JIT (Just In Time) компилация **в платформено-зависим машинен код** (native code).
  - Може твърдо да се каже, че **изпълнителната среда на Net (CLR) е по-добра от виртуалната машина на Java**. Microsoft използва получилия широко признание опит с виртуалната машина на Java, но подобри процеса като за разлика от Java, междинният код не се интерпретира от изпълнителната среда, а се компилира при първото изпълнение (JIT компилация) с отчет на всички особености на текущата платформа.
  - Така до голяма степен се решава проблема с производителността. Трябва да се отбележи също, че работейки с IL-код, **CLR осигурява достатъчно ефективно не само оптимизация** (отчитат се особеностите на процесора) **на изпълнението**, но така **също и безопасност при това изпълнение**.

- **Разнообразие от приложения и непрекъснато развитие**: .NET Framework позволява разработка на **Windows приложения**, **web приложения** и **услуги**, **приложения за мобилни телефони**, **бази от данни** и **вградени (embedded) приложения** (чрез .Net compact).

Има данни, че Microsoft заделят около **80%** от своите финансови инвестиции и работни ресурси **за развитието на .Net** Framework, което е една гаранция за непрекъснатото развитие на тези технологии.