

Простори імен, збірки та система типів .NET

Питання 1.2.



Платформа .NET Core

Складові частини



- Компиляторы языка превращают ваш исходный код (написанный на языке C#, F#, Visual Basic и др.) в код промежуточного языка (IL), сохраняющийся в сборках Common Language Runtime (CLR) – спільне середовище виконання програм (приложениях и библиотеках классов). В C# 6 был добавлен полностью переписанный компилятор, известный под названием Roslyn.
- Общеязыковая исполняющая среда (CoreCLR) загружает сборки, компилирует IL-код, хранящийся в них, в инструкции машинного кода для процессора вашего компьютера и выполняет код в среде с управлением такими ресурсами, как потоки и память.
- Базовые библиотеки классов и NuGet-пакеты (CoreFX) — это готовые сборки типов для выполнения универсальных задач при разработке приложений.
 - Вы можете использовать их для быстрого создания всех атрибутов приложений, как если бы конструировали из деталей Lego.
 - Платформа .NET Core 2.0 основана на версии .NET Standard 2.0, включающей функции всех предыдущих версий .NET Standard и поднимающей .NET Core до уровня современных версий .NET Framework и Xamarin.

Стандартные библиотеки классов и CoreFX

Збірки, NuGet-пакети та простори імен



- Библиотеки предварительно собранного кода (BCL, CoreFX), состоят из сборок и пространств имен, упрощающих управление десятками тысяч доступных типов.
- **Сборки** используются для хранения типов в файловой системе.
 - По сути, это механизм для развертывания кода. Например, сборка `System.Data.dll` содержит типы для управления данными. Чтобы использовать типы в других сборках, на них нужно сослаться.
 - Сборки часто распространяются в виде NuGet-пакетов, которые могут содержать несколько сборок и других ресурсов.
- **Пространство имен** — это адрес типа.
 - Пространство имен — это механизм уникальной идентификации типа через его полный адрес, а не просто короткое имя.
 - В реальном мире Серега из дома номер 34 по улице Абрикосовой отличается от Сереги из дома номер 12 по улице Виноградной.
 - Говоря о .NET Core, интерфейс `IActionFilter` пространства имен `System.Web.Mvc` отличается от интерфейса `IActionFilter` пространства имен `System.Web.Http.Filters`.

Додавання посилань на залежні збірки

Любое приложение, созданное средствами .NET Core, зависит от платформы приложений Microsoft .NET Core.



- Если сборка компилируется в виде библиотеки классов (предоставляет типы другим сборкам), то получает расширение .dll и не может выполняться автономно, а только через команду dotnet run.
- Если сборка компилируется как приложение, то получает расширение .exe (executable — исполняемый файл) и может выполняться автономно.
- Любые сборки могут ссылаться на одну или несколько сборок, содержащих библиотеку классов, определяя эти связи как зависимости, но вы не можете использовать циклические ссылки.
- Среда разработки Visual Studio предупредит вас о том, что вы пытаетесь добавить ссылку зависимости, создавая при этом циклическую ссылку.

NAMESPACE

Logical division of classes
in .NET framework

Provides a fundamental unit
of logical code grouping

ASSEMBLY

Fundamental unit of
deployment, version
control, reuse, activation
scoping and security
permissions for a .NET
based application

Provides a fundamental
unit of physical code
grouping

Зв'язані збірки та простори імен

Вид → Обозреватель объектов



Обозреватель объектов Program.cs

Обзор: Мое решение

<Поиск>

- ConsoleApp
 - Microsoft.CSharp
 - Microsoft.VisualBasic
 - Microsoft.Win32.Primitives
 - mscorlib
 - netstandard
 - System
 - System.AppContext
 - System.Buffers
 - System.Collections
 - System.Collections.Concurrent
 - System.Collections.Immutable
 - System.Collections.NonGeneric
 - System.Collections.Specialized
 - System.ComponentModel
 - System.ComponentModel.Annotations
 - System.ComponentModel.DataAnnotations
 - System.ComponentModel.EventBasedAsync
 - System.ComponentModel.Primitives
 - System.ComponentModel.TypeConverter
 - System.Configuration
 - System.Console
 - System
 - Console
 - ConsoleCancelEventArgs
 - ConsoleCancelEventHandler
 - ConsoleColor
 - ConsoleKey
 - ConsoleKeyInfo
 - ConsoleModifiers
 - ConsoleSpecialKey
 - System.Core
 - System.Data
 - System.Data.Common
 - System.Diagnostics.Contracts
 - System.Diagnostics.Debug
 - System.Diagnostics.DiagnosticSource

- Beep()
- Beep(int, int)
- Clear()
- MoveBufferArea(int, int, int, int, int, int)
- MoveBufferArea(int, int, int, int, int, int, char, System.ConsoleColor, System.ConsoleColor)
- OpenStandardError()
- OpenStandardError(int)
- OpenStandardInput()
- OpenStandardInput(int)
- OpenStandardOutput()
- OpenStandardOutput(int)
- Read()
- ReadKey()
- ReadKey(bool)
- ReadLine()
- ResetColor()
- SetBufferSize(int, int)
- SetCursorPosition(int, int)
- SetError(System.IO.TextWriter)
- SetIn(System.IO.TextReader)
- SetOut(System.IO.TextWriter)
- SetWindowPosition(int, int)
- SetWindowSize(int, int)
- Write(bool)
- Write(char)
- Write(char[])
- Write(char[], int, int)
- Write(decimal)

```
public static class Console
Элемент объекта System
```

Сводка:
Represents the standard input, output, and error streams for console applications. This class cannot be inherited.

Обозреватель решений

Обозреватель решений — поиск (Ctrl+ж)

Решение "ConsoleApp" (проекты: 1 из 1)

ConsoleApp

Зависимости

ПАКЕТ SDK

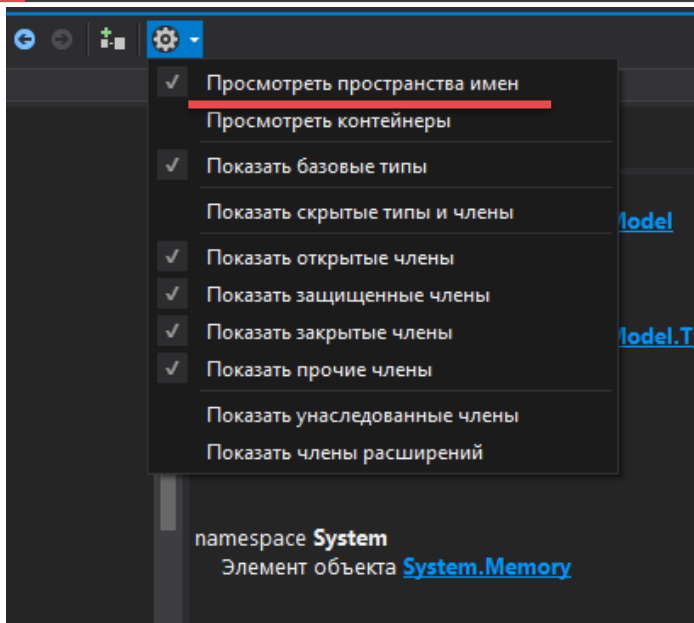
Microsoft.NETCore.App (2.2.0)

- Microsoft.NETCore.DotNetHostPolicy (2.2.0)
- Microsoft.NETCore.Platforms (2.2.0)
- Microsoft.NETCore.Targets (2.0.0)
- Microsoft.CSharp.dll
- Microsoft.VisualBasic.dll
- Microsoft.Win32.Primitives.dll
- mscorlib.dll
- netstandard.dll
- System.AppContext.dll
- System.Buffers.dll
- System.Collections.Concurrent.dll
- System.Collections.dll
- System.Collections.Immutable.dll
- System.Collections.NonGeneric.dll
- System.Collections.Specialized.dll
- System.ComponentModel.Annotations.dll
- System.ComponentModel.DataAnnotations.dll
- System.ComponentModel.dll
- System.ComponentModel.EventBasedAsync.dll
- System.ComponentModel.Primitives.dll
- System.ComponentModel.TypeConverter.dll
- System.Configuration.dll
- System.Console.dll
- System.Core.dll
- System.Data.Common.dll
- System.Data.dll
- System.Diagnostics.Contracts.dll
- System.Diagnostics.Debug.dll
- System.Diagnostics.DiagnosticSource.dll
- System.Diagnostics.FileVersionInfo.dll
- System.Diagnostics.Process.dll

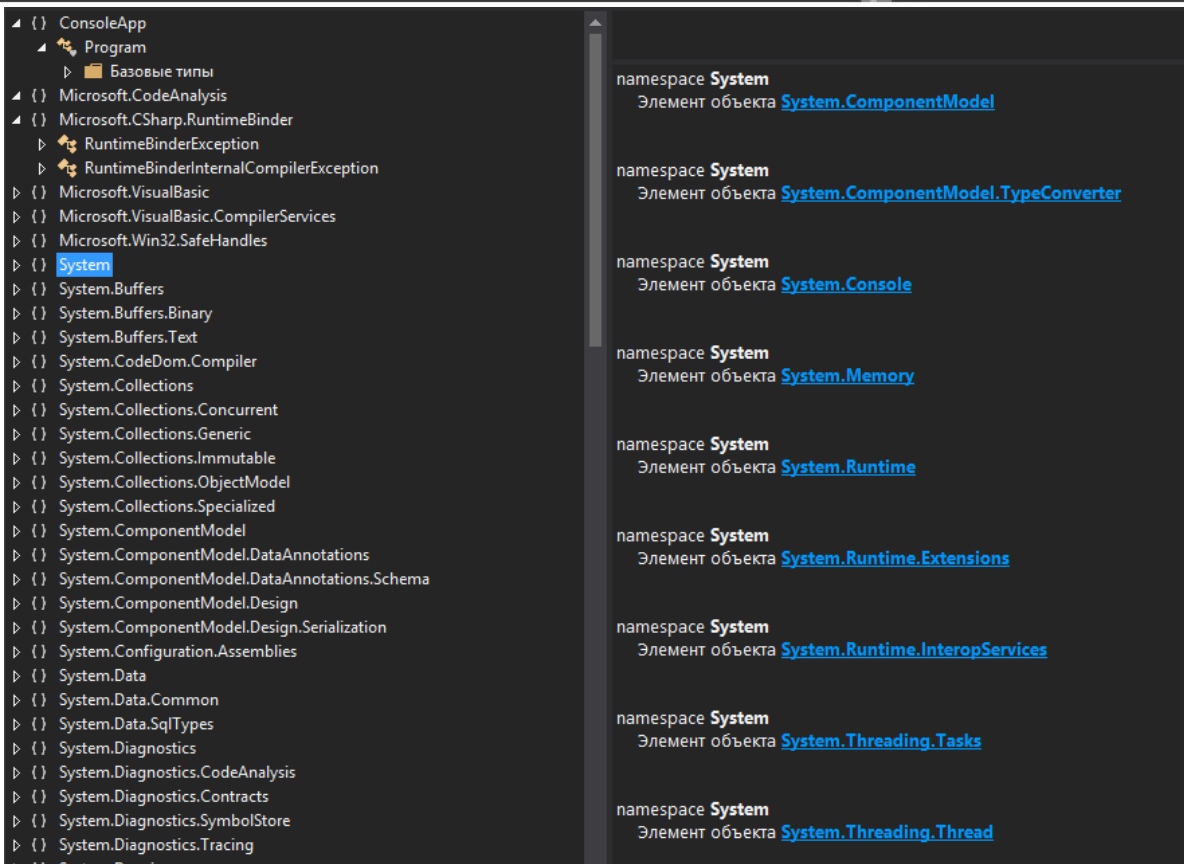
Обозреватель решений Team Explorer — Главная

Зв'язані збірки та простори імен

Перегляд просторів імен

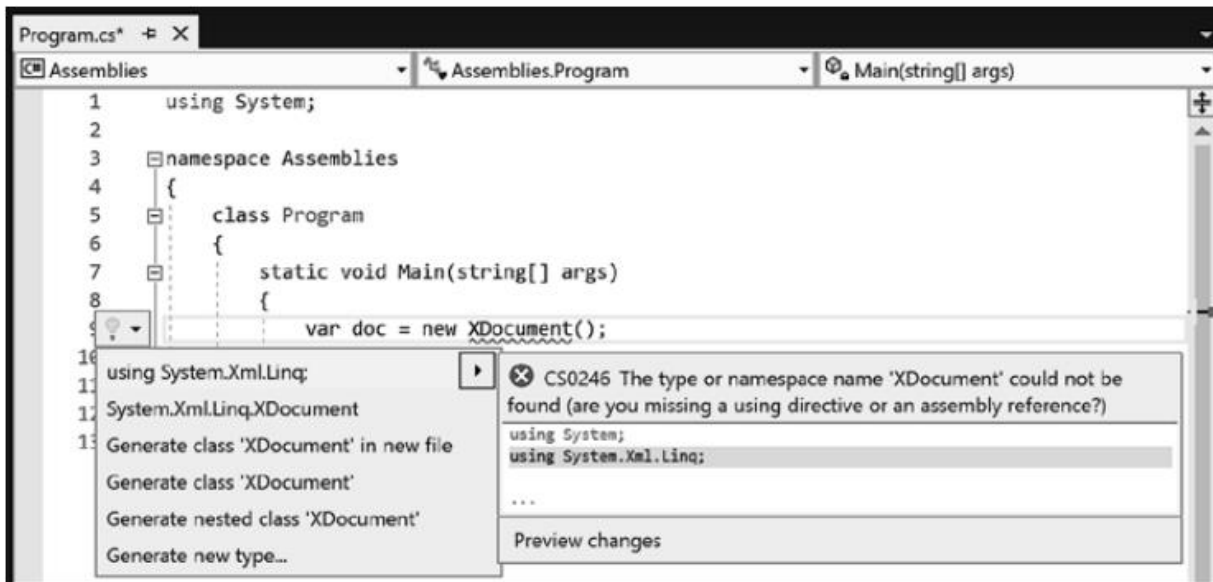


Тепер панель Object Browser (Обозреватель объектов) отображает типы, сгруппированные по сборкам.



Імпорт простору імен

Встановіть курсор на назву потрібного класу



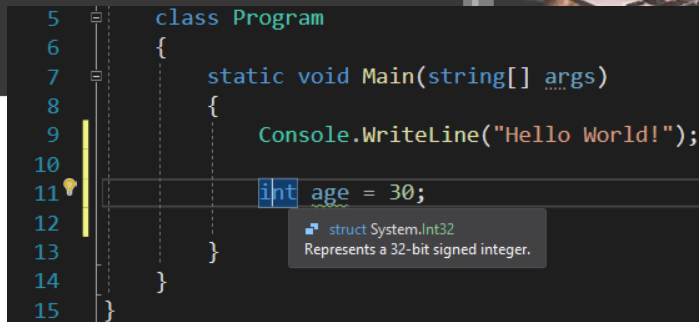
Выберите в контекстном меню пункт using System.Xml.Linq;

Можна використати команду Ctrl + .

Зв'язування ключових слів C# з типами .NET

Чи є різниця між string та String?

- все ключевые слова типов в C# являются псевдонимами для типов .NET в сборке библиотеки классов.



```
5 class Program
6 {
7     static void Main(string[] args)
8     {
9         Console.WriteLine("Hello World!");
10
11         int age = 30;
12     }
13 }
14
15
```

int age = 30;

struct System.Int32
Represents a 32-bit signed integer.

Ключевое слово	Тип .NET	Ключевое слово	Тип .NET
string	System.String	char	System.Char
sbyte	System.SByte	byte	System.Byte
short	System.Int16	ushort	System.UInt16
int	System.Int32	uint	System.UInt32
long	System.Int64	ulong	System.UInt64
float	System.Single	double	System.Double
decimal	System.Decimal	bool	System.Boolean
object	System.Object	dynamic	System.Dynamic.DynamicObject

Кроссплатформне використання коду

за допомогою бібліотек класів .NET Standard 2.0



Прежде чем появился .NET Standard 2.0, в ходу были портативные библиотеки классов (Portable Class Libraries, PCL).

- С их помощью можно было создать библиотеку кода и явно указать, на каких платформах ей требуется поддержка, например Xamarin, Silverlight, Windows 8 и т. д.
- Затем ваша библиотека могла использовать пересечение возможностей API, поддерживаемых указанными платформами.

.NET Standard 2.0 — единым API, который будет поддерживаться всеми будущими платформами .NET.

- Создать библиотеку типов, которые будут поддерживаться и .NET Framework (Windows), и .NET Core (Windows, macOS и Linux), и Xamarin (iOS, Android и Windows Mobile), проще всего с помощью .NET Standard 2.0.

Створення бібліотеки класів .NET Standard

Кросплатформна бібліотека



Бібліотека класів (.NET Standard)

Проект для створення бібліотеки класів, призначеної для .NET Standard.

C#

Android

iOS

Linux

macOS

Windows

Бібліотека

```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <TargetFramework>netstandard2.0</TargetFramework>
5   </PropertyGroup>
6
7 </Project>
8
```



Обозреватель решений — поиск (Ctrl+ж)

+ Решение "SharedLibrary" (проекты: 1 из 1)

+ C# SharedLibrary

Зависимости

ПАКЕТ SDK

NETStandard.Library (2.0.3)

Microsoft.NETCore.Platforms (1.1.0)

netstandard.dll

+ C# Class1.cs

Class1

Пакети NuGet

Платформа .NET Core разделена на несколько пакетов



- Каждый из этих пакетов представляет собой отдельную сборку с тем же именем, что и у пакета.
 - Например, пакет `System.Collections` содержит сборку `System.Collections.dll`.

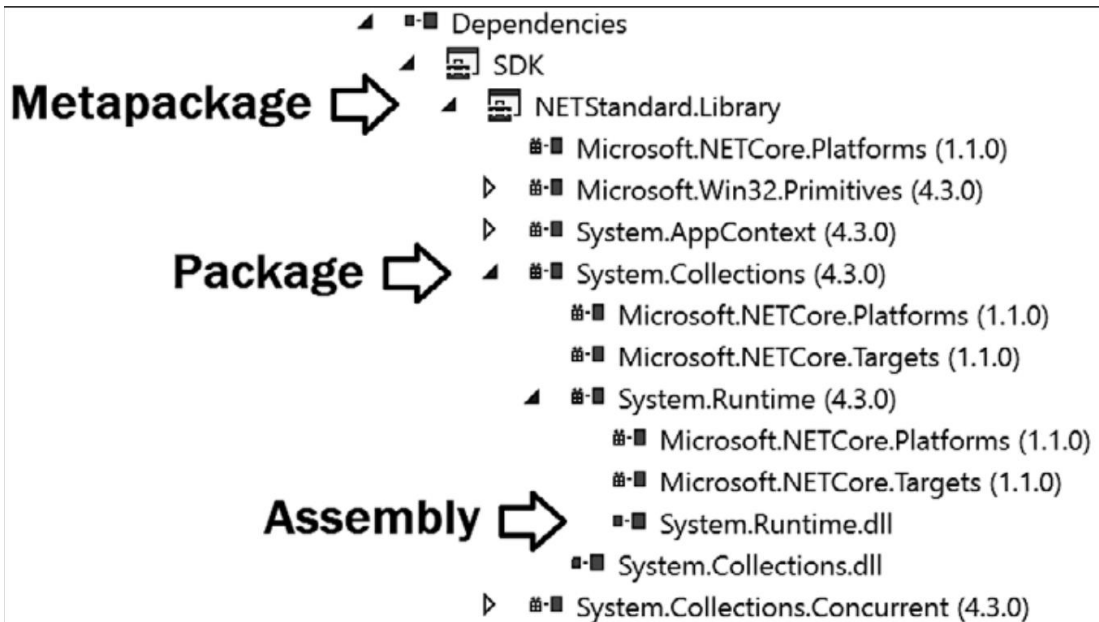
Пакет	Важные типы
<code>System.Runtime</code>	<code>Object</code> , <code>String</code> , <code>Array</code>
<code>System.Collections</code>	<code>List<T></code> , <code>Dictionary<TKey, TValue></code>
<code>System.Net.Http</code>	<code>HttpClient</code> , <code>HttpResponseMessage</code>
<code>System.IO.FileSystem</code>	<code>File</code> , <code>Directory</code>
<code>System.Reflection</code>	<code>Assembly</code> , <code>TypeInfo</code> , <code>MethodInfo</code>

Метапакети

В документації Microsoft часто називаються просто пакетами



- Метапакети представляють собою набори спільно використовуваних пакетів.
- На метапакети ссылаются так же, как и на любой другой NuGet-пакет.
- Ссылаясь на метапакет, вы фактически добавляете ссылки на каждый из вложенных пакетов.



Обратите внимание: у пакета Microsoft.NETCore.App версии 2.0.0 есть зависимость от пакета NETStandard.Library версии 2.0.0.

Платформы

Между платформами и пакетами существует двусторонняя связь



- Пакеты определяют API, а платформы группируют пакеты.
 - Платформа без пакетов не сможет определить никакой API.
- Каждый из пакетов .NET Core поддерживает определенный набор платформ.
 - Например, пакет `System.IO.FileSystem` підтримує плафторми NET Standard версії 1,3; .NET Framework версії 4,6; шесть платформ Xamarin (к примеру, Xamarin.iOS 10)

Публикация приложений

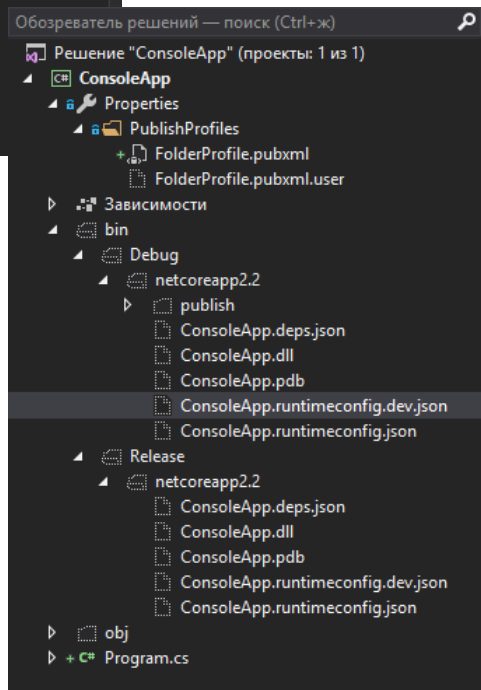
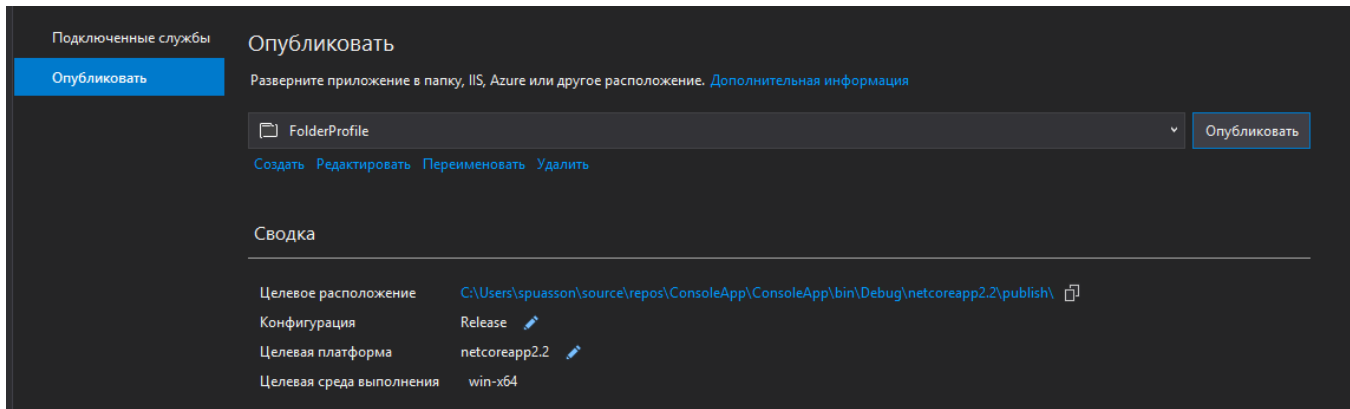
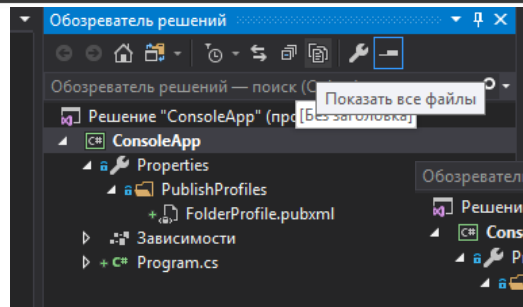
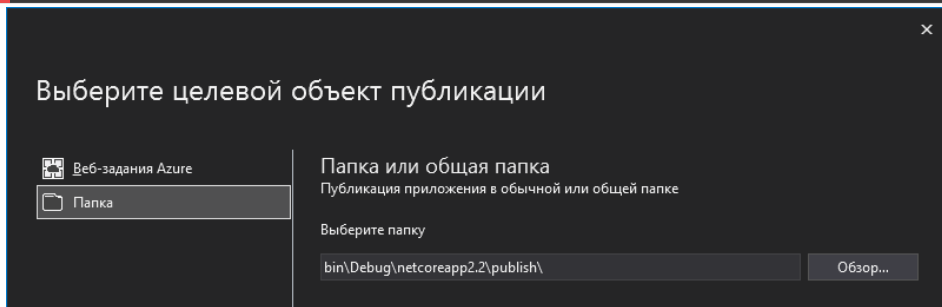
Платформозалежный та автономный способы



- При развертывании приложения с соответствующими зависимостями, но не самой платформы .NET Core вам придется полагаться на то, что она уже развернута на целевом компьютере.
 - Способ подойдет для веб-приложений, разворачиваемых на сервере, поскольку эта платформа и множество других веб-приложений, вероятно, уже установлены на сервере.
- Иногда может понадобиться передать клиенту USB-диск с приложением и быть уверенными, что он сумеет запустить его на своем компьютере.
 - Для этого желательно выполнить автономное развертывание.
 - Дистрибутив будет больше, но вы сможете быть уверены в успешном запуске.

Публікація додатків

Visual Studio (На проекті ПКМ -> Опублікувати)



Додавання посилання на пакет

Приклад з пакетом Newtonsoft.Json (ПКМ на Зависимости)



Обзор Установлено Обновления

Поиск (Ctrl+L) ☐ Включить предварительные версии

Источник пакета: nuget.org

	Newtonsoft.Json автор: James Newton-King, Скачиваний: 276M Json.NET is a popular high-performance JSON framework for .NET	v12.0.2
	Microsoft.Extensions.Logging автор: Microsoft, Скачиваний: 101M Logging infrastructure default implementation for Microsoft.Extensions.Logging.	v2.2.0
	Microsoft.Extensions.DependencyInjection автор: Microsoft, Скачиваний: 89,4M Default implementation of dependency injection for Microsoft.Extensions.DependencyInjection.	v2.2.0
	Castle.Core автор: Castle Project Contributors, Скачиваний: 71,2M Castle Core, including DynamicProxy, Logging Abstractions and DictionaryAdapter	v4.4.0
	EntityFramework автор: Microsoft, Скачиваний: 70,6M Entity Framework is Microsoft's recommended data access technology for new applications.	v6.2.0
	jQuery автор: jQuery Foundation, Inc., Скачиваний: 69,5M Несовместимо: вместо этого используйте Bower . jQuery is a new kind of JavaScript Library.	v3.4.1
	Moq автор: Daniel Cazzulino, kzu, Скачиваний: 69M Moq is the most popular and friendly mocking framework for .NET.	v4.13.0
	Microsoft.AspNet.Mvc автор: Microsoft, Скачиваний: 65,3M	v5.2.7

Newtonsoft.Json

Версия: Последняя стабильная 12.0.2

Описание
Json.NET is a popular high-performance JSON framework for .NET

Версия: 12.0.2
Авторы: James Newton-King
Лицензия: [MIT](#)
Дата публикации: 22 апреля 2019 г. (22.04.2019)
URL-адрес проекта: <https://www.newtonsoft.com/json>
Сообщить о нарушении: <https://www.nuget.org/packages/Newtonsoft.Json/12.0.2/ReportAbuse>

Теги: json

Зависимости
.NETFramework,Version=v2.0
Зависимости отсутствуют
.NETPortable,Version=v0.0,Profile=Profile259
Зависимости отсутствуют
.NETFramework,Version=v4.5
Зависимости отсутствуют

Упаковка бібліотеки для розповсюдження

За допомогою NuGet

The screenshot shows the Visual Studio IDE with a C# file named StringExtensions.cs open. The code defines a static class StringExtensions within the SharedLibrary namespace, containing three methods: IsValidXmlTag, IsValidPassword, and IsValidHex. The Solution Explorer on the right shows the project structure, including the SharedLibrary project and its dependencies (PAKET SDK). The Solution Explorer also lists the methods defined in StringExtensions.cs.

```
StringExtensions.cs
C# SharedLibrary.StringExtensions
IsValidXmlTag(string input)

1 using System;
2 using System.Text.RegularExpressions;
3
4 namespace SharedLibrary
5 {
6     public static class StringExtensions
7     {
8         public static bool IsValidXmlTag(this string input)
9         {
10             return Regex.IsMatch(input, @"^<([a-z]+)([^\<]+)*(?:>(.*)<\/\1>|\/s+\/>)$");
11         }
12         public static bool IsValidPassword(this string input)
13         {
14             // Не менше 8 допустимих символів
15             return Regex.IsMatch(input, "^[a-zA-Z0-9_]{8,}$");
16         }
17         public static bool IsValidHex(this string input)
18         {
19             // 3 або 6 допустимих символів шістнадцяткового числа
20             return Regex.IsMatch(input, "^#?([a-fA-F0-9]{3}|[a-fA-F0-9]{6})$");
21         }
22     }
23 }
```

Обозреватель решений

Обозреватель решений — поиск (Ctrl+ж)

- Решение "SharedLibrary" (проекты: 1 из 1)
 - SharedLibrary
 - Зависимости
 - PAKET SDK
 - StringExtensions.cs
 - StringExtensions
 - IsValidXmlTag(string) : bool
 - IsValidPassword(string) : bool
 - IsValidHex(string) : bool

Упаковка бібліотеки для розповсюдження

Завантаження бібліотеки на сайт NuGet



<https://www.nuget.org/packages/manage/upload>

A screenshot of the NuGet website's 'Upload' page. The header is dark blue with the 'nuget' logo and navigation links: Packages, Upload (active), Statistics, Documentation, Downloads, and Blog. A user profile 'spuasson' is in the top right. Below the header is a search bar with the placeholder text 'Search for packages...'. A message states 'Your account is now registered!'. The breadcrumb trail is 'Packages > Upload'. A paragraph explains that the package file will be uploaded to the NuGet Gallery server (https://www.nuget.org). A section titled 'Upload' contains a text box with the instruction 'Browse or Drop files to select a package (.nupkg) or symbols package (.snupkg)...' and a 'Browse...' button.

nuget Packages Upload Statistics Documentation Downloads Blog spuasson

Search for packages...

ⓘ Your account is now registered!

🔍 > Packages > Upload

Your package file will be uploaded and hosted on the NuGet Gallery server (<https://www.nuget.org>).

▽ Upload

Browse or Drop files to select a package (.nupkg) or symbols package (.snupkg)... Browse...

A screenshot of a Windows File Explorer window. The address bar shows the path: <source> > repos > SharedLibrary > SharedLibrary > bin > Debug >. The search bar contains 'Debug'. The left sidebar shows a tree view with 'OneDrive' expanded, containing 'PycharmProjects', 'source', 'repos', 'ConsoleApp', 'ConsoleApp1', and 'ConsoleApp2'. The main pane shows a table of files and folders.

Имя	Дата изменения	Тип	Размер
netstandard2.0	10.09.2019 16:33	Папка с файлами	
Packt.CS7.SharedLibrary.1.0.0.nupkg	10.09.2019 16:35	Файл "NUPKG"	4 КБ
SharedLibrary.1.0.0.nupkg	10.09.2019 16:33	Файл "NUPKG"	4 КБ

Упаковка бібліотеки для розповсюдження

Підключення готової бібліотеки











NuGet: SharedLibrary SharedLibrary.csproj StringExtensions.cs

Обзор Установлено Обновления

packt. ☐ Включить предварительные версии

Источник пакета: nuget.org

Диспетчер пакетов NuGet: SharedLibrary

 Packt.CS7.SLib автор: Mark J Price, Скачиваний: 279 v1.0.0 Three extension methods to validate a string value.	 Packt.CS7.SLib  Версия: Последняя стабильная 1.0.0 <input type="button" value="Установить"/> <input checked="" type="radio"/> Параметры Описание Three extension methods to validate a string value. Версия: 1.0.0 Авторы: Mark J Price Лицензия: Просмотреть лицензию Дата публикации: 31 января 2018 г. (31.01.2018) URL-адрес проекта: http://github.com/markjprice/cs7dotnetcore2 Сообщить о нарушении: https://www.nuget.org/packages/Packt.CS7.SLib/1.0.0/ReportAbuse Теги: string, packt, extension, cs7 Зависимости <i>Зависимости отсутствуют</i>
 Packt.Extensions.SharedLibrary автор: Manuel Bautista, Скачиваний: 136 v1.0.0 Three extension methods to validate a string value.	
 Packt.Shoujie.Package автор: Alan Jack, Скачиваний: 61 v1.0.0 Three extension methods to validate a string value.	
 Packt.CS7.SheredLibrary автор: Prolovov Stepan, Скачиваний: 226 v1.0.0 Three extension methods to validate a string value.	
 Packt.CS7.SharedLibraryjk автор: Mark J Price, Скачиваний: 88 v1.0.0 Three extension methods to validate a string value.	
 PasquiPackt.CS7.SharedLibrary автор: Mark J Price, Скачиваний: 262 v1.0.0 Three extension methods to validate a string value.	

Все пакеты лицензируются их владельцами. NuGet не несет ответственности за пакеты сторонних производителей и не предоставляет лицензии на такие пакеты.

☐ Больше не показывать

Обозреватель решений

Обозреватель решений — поиск (Ctrl+ж)

Решение "SharedLibrary" (проекты: 1 из 1)

- SharedLibrary
 - Зависимости
 - ПАКЕТ SDK
 - StringExtensions.cs
 - StringExtensions
 - IsValidXmlTag(string) : bool
 - IsValidPassword(string) : bool
 - IsValidHex(string) : bool



```
using static System.Console;
using Packt.CS7;

namespace Assemblies
{
    class Program
    {
        static void Main(string[] args)
        {
            Write("Enter a valid color value in hex: ");
            string hex = ReadLine();
            WriteLine($"Is {hex} a valid color value: {hex.IsValidHex()}");

            Write("Enter a valid XML tag: ");
            string xmlTag = ReadLine();
            WriteLine($"Is {xmlTag} a valid XML tag: {xmlTag.IsValidXmlTag()}");

            Write("Enter a valid password: ");
            string password = ReadLine();
            WriteLine($"Is {password} a valid password: {password.IsValidPassword()}");
        }
    }
}
```

Enter a valid color value in hex: 00ffc8
Is 00ffc8 a valid color value: True
Enter a valid XML tag: <h1 class="" />
Is <h1 class="" /> a valid XML tag: False
Enter a valid password: secretsauce
Is secretsauce a valid password: True

Використання розповсюджених типів

.NET Standard



Пространство имен	Примеры типов	Описание
System	SByte, Int16, Int32, Int64	Положительные и отрицательные целые числа
System	Byte, UInt16, UInt32, UInt64	Натуральные числа, то есть положительные целые числа
System	Single, Double	Вещественные числа, то есть числа с плавающей точкой
System	Decimal	Точные вещественные числа, используются для научных, инженерных или финансовых нужд
System.Numerics	BigInteger, Complex, Quaternion	Условно большие целые числа, комплексные и гиперкомплексные числа

Крупні цілі числа

Наибольшее целое число, которое может быть сохранено в типах .NET Standard, имеющих псевдонимы в языке C#, равняется примерно 18,5 квинтиллиона.



```
1 using System.Numerics;
2 using static System.Console;
3
4 namespace ConsoleApp
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             var largestLong = ulong.MaxValue;
11             WriteLine($"{largestLong,40:N0}");
12             var atomsInTheUniverse = BigInteger.Parse("123456789012345678901234567890");
13             WriteLine($"{atomsInTheUniverse,40:N0}");
14         }
15     }
16 }
17
```



biginteger.cs

Консоль отладки Microsoft Visual Studio

```
18 446 744 073 709 551 615
123 456 789 012 345 678 901 234 567 890
```

<https://docs.microsoft.com/en-us/dotnet/standard/numerics>

Робота з комплексними числами



```
1 using System.Numerics;
2 using static System.Console;
3
4 namespace ConsoleApp
5 {
6     class ComplexNumbers
7     {
8         static void Main(string[] args)
9         {
10             var c1 = new Complex(4, 2);
11             var c2 = new Complex(3, 7);
12             var c3 = c1 + c2;
13             WriteLine($"{c1} added to {c2} is {c3}");
14         }
15     }
16 }
17 }
```

Консоль отладки Microsoft Visual Studio

```
(4, 2) added to (3, 7) is (7, 9)
```

Работа з текстом



Пространство имен	Тип	Описание
System	Char	Хранение одного текстового символа
System	String	Хранение нескольких текстовых символов
System.Text	StringBuilder	Эффективное управление строками
System.Text.RegularExpressions	Regex	Эффективное управление строками, соответствующими шаблонам



```
class Program
{
    static void Main(string[] args)
    {
        string city = "London";
        WriteLine($"{city} is {city.Length} characters long.");
        WriteLine($"First char is {city[0]} and third is {city[2]}.");

        string cities = "Paris,Berlin,Madrid,New York";
        string[] citiesArray = cities.Split(',');
        foreach (string item in citiesArray)
        {
            WriteLine(item);
        }

        string fullname = "Alan Jones";
        int indexOfTheSpace = fullname.IndexOf(' ');
        string firstname = fullname.Substring(0, indexOfTheSpace);
        string lastname = fullname.Substring(indexOfTheSpace + 1);
        WriteLine($"{lastname}, {firstname}");

        string company = "Microsoft";
        bool startsWithM = company.StartsWith("M");
        bool containsN = company.Contains("N");
        WriteLine($"Starts with M: {startsWithM}, contains an N:{containsN}");
    }
}
```

Консоль отладки Microsoft Visual Studio

```
London is 6 characters long.
First char is L and third is n.
Paris
Berlin
Madrid
New York
Jones, Alan
Starts with M: True, contains an N:False
```



stringUsage.cs

Извлечение длины строки
Извлечение символов строки
Разделение строк
Извлечение фрагмента строки
Проверка содержимого строк

Інші члени класу string



Член	Описание
Trim, TrimStart и TrimEnd	Удаляют пробельные символы в начале и/или конце строки
ToUpper и ToLower	Преобразуют символы строки в прописные или строчные
Insert и Remove	Добавляют или удаляют указанный текст в переменной типа string
Replace	Замещает указанный текст
String.Concat	Конкатенирует две переменные типа string. Оператор + вызывает этот метод, если используется между переменными типа string
String.Join	Конкатенирует одну или несколько переменных типа string с указанным символом между ними
string.IsNullOrEmpty	Проверяет, хранит ли переменная типа string значение null или она пустая ("")
string.IsNullOrWhiteSpace	Проверяет, является ли переменная типа string значением null, пустой строкой или строкой, состоящей только из пробельных символов (например, табуляции, пробела, возврата каретки, перевода строки и т. д.)
String.Empty	Можно задействовать вместо выделения памяти каждый раз, когда вы применяете литеральное значение string, используя пару двойных кавычек без содержимого ("")
string.Format	Устаревший альтернативный метод вывода форматированных строк, применяющий позиционированные параметры вместо именованных

Наслідки незмінюваності класу



- Клас `string` может стать неэффективным и приводить к “разбуханию” кода при неправильном использовании, особенно при выполнении конкатенации строк.
- Но когда необходимо представлять базовые символьные данные, такие как номер к арточки социального страхования, имя и фамилия или простые фрагменты текста, используемые внутри приложения, он является идеальным вариантом.
- Тем не менее, если строится приложение, в котором будут часто изменяться текстовые данные (например, текстовый процессор), то представление обрабатываемых текстовых данных с применением объектов `string` будет очень неудачным решением, поскольку это практически наверняка (и часто косвенно) приведет к созданию ненужных копий данных `string`.

Тип `System.Text.StringBuilder`



- Как и `System.String`, класс `StringBuilder` определяет методы, которые позволяют, к примеру, заменять или форматировать сегменты.
- Чтобы использовать этот класс в файлах кода C#, первым делом понадобится импортировать следующее пространство имен в файл кода `using System.Text;`
- Уникальным в `StringBuilder` является то, что при вызове его членов производится непосредственное изменение внутренних символьных данных объекта (делая его более эффективным), без получения копии данных в модифицированном формате.
 - При создании экземпляра `StringBuilder` начальные значения объекта могут быть заданы через один из множества конструкторов.

Тип System.Text.StringBuilder



```
static void FunWithStringBuilder()
{
    Console.WriteLine("=> Using the StringBuilder:");
    StringBuilder sb = new StringBuilder("**** Fantastic Games ****");
    sb.Append("\n");
    sb.AppendLine("Half Life");
    sb.AppendLine("Morrowind");
    sb.AppendLine("Deus Ex" + "2");
    sb.AppendLine("System Shock");
    Console.WriteLine(sb.ToString());
    sb.Replace("2", " Invisible War");
    Console.WriteLine(sb.ToString());
    Console.WriteLine("sb has {0} chars.", sb.Length);
    Console.WriteLine();
}
```

- По умолчанию StringBuilder способен хранить строку длиной не более 16 символов (но при необходимости будет автоматически расширяться), однако значение начальной длины можно изменить через дополнительный аргумент конструктора:

```
// Создать StringBuilder с исходным размером в 256 символов.
```

```
StringBuilder sb = new StringBuilder("**** Fantastic Games ****", 256);
```

Зіставлення шаблонів з регулярними виразами



```
1 using static System.Console;
2 using System.Text.RegularExpressions;
3
4 namespace ConsoleApp
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             Write("Enter your age: ");
11             string input = ReadLine();
12             var ageChecker = new Regex(@"^\d+$");
13             if (ageChecker.IsMatch(input))
14             {
15                 WriteLine("Thank you!");
16             }
17             else
18             {
19                 WriteLine($"This is not a valid age: {input}");
20             }
21         }
22     }
23 }
```

Консоль отладки Microsoft Visual Studio

```
Enter your age: 70
Thank you!
```

Консоль отладки Microsoft Visual Studio

```
Enter your age: B52
This is not a valid age: B52
```

Синтаксис регулярных выражений



Символ	Значение	Символ	Значение
^	Начало ввода	\$	Конец ввода
\d	Одна цифра	\D	Любой нецифровой символ
\w	Пробельный символ	\W	Любой символ, кроме пробельного
[A-Za-z0-9]	Диапазон символов	\^	Символ ^ (каре́тки)
[aeiou]	Набор символов	[^aeiou]	Любой символ, кроме входящего в набор
.	Один символ	\.	Символ . (точка)

- Символы, які впливають на попередній символ у регулярному виразі

Символ	Значение	Символ	Значение
+	Один или больше	?	Один или ни одного
{3}	Точно три	{3,5}	От трех до пяти
{3,}	Три или больше	{,3}	До трех

Приклади регулярних виразів



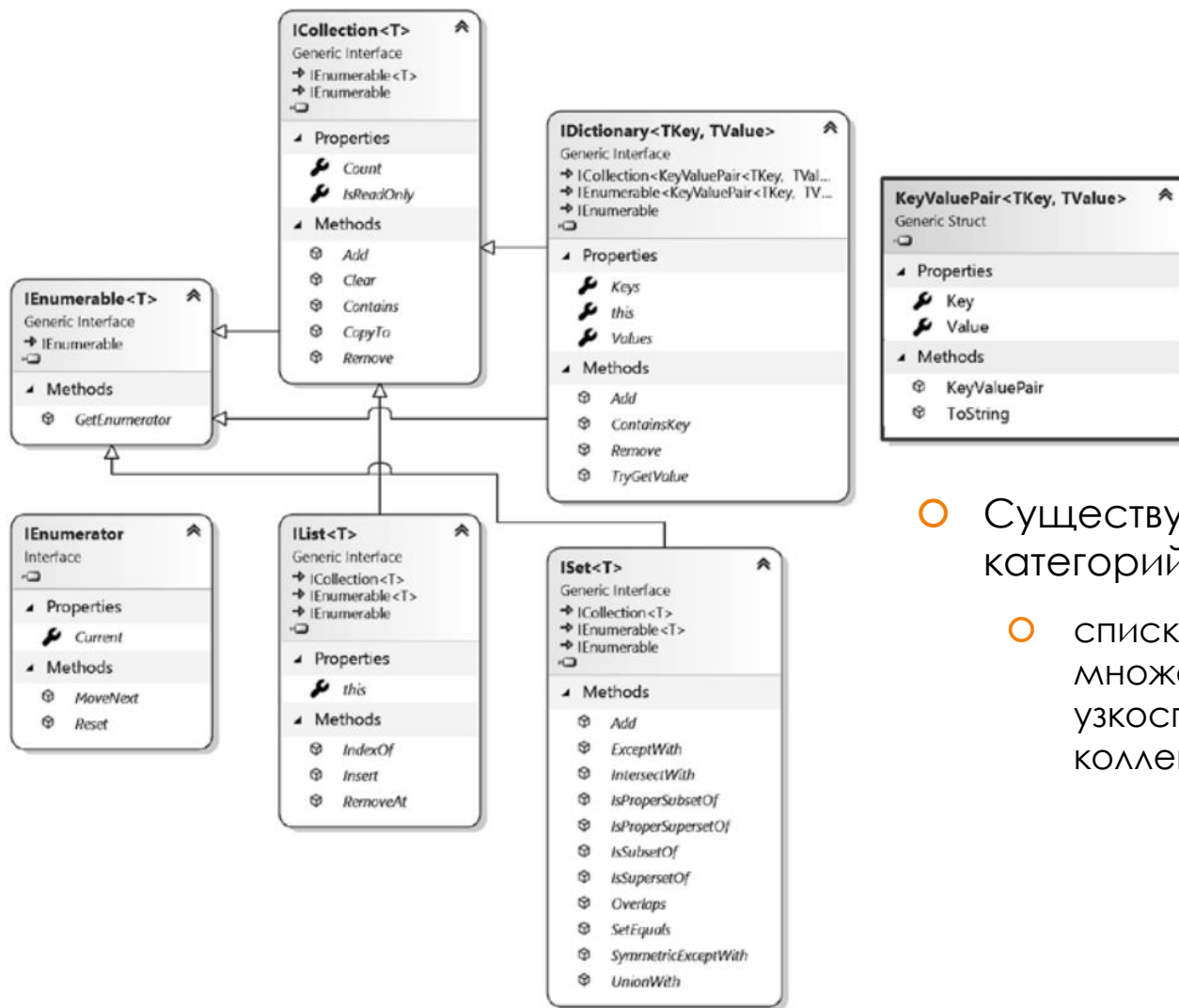
Выражение	Значение
<code>\d</code>	Одна цифра где-либо в вводе
<code>a</code>	Символ где-либо в вводе
<code>Bob</code>	Слово Bob где-либо в вводе
<code>^Bob</code>	Слово Bob в начале ввода
<code>Bob\$</code>	Слово Bob в конце ввода
<code>^\d{2}\$</code>	Точно две цифры
<code>^[0-9]{2}\$</code>	Точно две цифры
<code>^[A-Z]{4,}\$</code>	Не менее четырех прописных букв
<code>^[A-Za-z]{4,}\$</code>	Не менее четырех прописных или строчных букв
<code>^[A-Z]{2}\d{3}\$</code>	Точно две прописные буквы и три цифры
<code>^d.g\$</code>	Буква d, далее любой символ, а затем буква g, так что это выражение совпадет со словами типа dig, dog и др. с любым символом между буквами d и g
<code>^d\.g\$</code>	Буква d, далее точка (.), а затем буква g, поэтому данное выражение совпадает только с последовательностью d.g

Робота з колекціями

<https://docs.microsoft.com/en-us/dotnet/standard/collections/>



Пространство имен	Примеры типов	Описание
System.Collections	IEnumerable, IEnumerable<T>	Интерфейсы и базовые классы, используемые коллекциями
System.Collections.Generic	List<T>, Dictionary<T>, Queue<T>, Stack<T>	Стали применяться в версии C# 2 с .NET 2.0 и являются более предпочтительными, так как позволяют указать тип, который будет использован при сохранении (а это безопаснее, быстрее и эффективнее)
System.Collections.Concurrent	BlockingCollection, ConcurrentDictionary, ConcurrentQueue	Эти коллекции безопасны для применения в многопоточных приложениях
System.Collections.Immutable	ImmutableArray, ImmutableDictionary, ImmutableList, ImmutableQueue	Предназначены для сценариев, в которых содержимое коллекции никогда не должно изменяться



- Существует несколько различных категорий коллекции:
- списки, словари, стеки, очереди, множества и другие узкоспециализированные коллекции

Колекції

Робота зі списками



```
static void Main(string[] args)
{
    var cities = new List<string>();
    cities.Add("London");
    cities.Add("Paris");
    cities.Add("Milan");
    WriteLine("Initial list");
    foreach (string city in cities)
    {
        WriteLine($" {city}");
    }
    WriteLine($"The first city is {cities[0]}.");
    WriteLine($"The last city is {cities[cities.Count - 1]}.");
    cities.Insert(0, "Sydney");
    WriteLine("After inserting Sydney at index 0");
    foreach (string city in cities)
    {
        WriteLine($" {city}");
    }
    cities.RemoveAt(1);
    cities.Remove("Milan");
    WriteLine("After removing two cities");
    foreach (string city in cities)
    {
        WriteLine($" {city}");
    }
}
```

Консоль отладки Microsoft Visual Studio

```
Initial list
London
Paris
Milan
The first city is London.
The last city is Milan.
After inserting Sydney at index 0
Sydney
London
Paris
Milan
After removing two cities
Sydney
Paris
```

Колекції

Робота зі словниками



```
1 using static System.Console;
2 using System.Collections.Generic;
3
4 namespace ConsoleApp
5 {
6     class Program
7     {
8         static void Main(string[] args)
9         {
10             var keywords = new Dictionary<string, string>();
11             keywords.Add("int", "32-bit integer data type");
12             keywords.Add("long", "64-bit integer data type");
13             keywords.Add("float", "Single precision floating point number");
14             WriteLine("Keywords and their definitions");
15             foreach (KeyValuePair<string, string> item in keywords)
16             {
17                 WriteLine($" {item.Key}: {item.Value}");
18             }
19             WriteLine($"The definition of long is {keywords["long"]}");
20         }
21     }
22 }
```

Консоль отладки Microsoft Visual Studio

Keywords and their definitions
int: 32-bit integer data type
long: 64-bit integer data type
float: Single precision floating point number
The definition of long is 64-bit integer data type

Колекції

Сортування колекцій: Несколько распространенных отсортированных коллекций



- Класс `List<T>` можно отсортировать, вызвав его метод `Sort`
- Классы `Dictionary<T>`, `Stack<T>` и `Queue<T>` не могут быть отсортированы, поскольку обычно это не требуется.

Колекция	Описание
<code>SortedDictionary<TKey, TValue></code>	Представляет собой коллекцию пар «ключ — значение», которые сортируются по ключу
<code>SortedList<TKey, TValue></code>	Представляет собой коллекцию пар «ключ — значение», сортируемых по ключу, на основе связанной реализации <code>IComparer<T></code>
<code>SortedSet<T></code>	Представляет собой коллекцию объектов, хранящихся в отсортированном порядке

Колекция	Описание
<code>System.Collections.BitArray</code>	Управляет компактным массивом двоичных значений, представленных логическими значениями, где <code>true</code> соответствует включенному биту (1), а <code>false</code> — отключенному (0)
<code>System.Collections.Generic.LinkedList<T></code>	Представляет собой двусвязный список, в котором каждый элемент имеет ссылку на свой предыдущий и следующий элементы

Колекції

Використання незмінюваних колекцій

```
1 using static System.Console;  
2 using System.Collections.Generic;  
3 using System.Collections.Immutable;
```



```
static void Main(string[] args)  
{  
    var cities = new List<string>();  
    cities.Add("London");  
    cities.Add("Paris");  
    cities.Add("Milan");  
    var immutableCities = cities.ToImmutableList();  
    var newList = immutableCities.Add("Rio");  
    Write("Immutable cities:");  
    foreach (string city in immutableCities)  
    {  
        Write($" {city}");  
    }  
    WriteLine();  
    Write("New cities:");  
    foreach (string city in newList)  
    {  
        Write($" {city}");  
    }  
    WriteLine();  
}
```

Иногда необходимо сделать коллекцию неизменяемой, то есть ни один из ее членов не может быть изменен, как, впрочем, удален или добавлен.

- Если вы импортируете пространство имен `System.Collections.Immutable`, то любая коллекция, реализующая интерфейс `IEnumerable<T>`, получит шесть методов расширения для преобразования этой коллекции в неизменяемый список, словарь, набор хеш-функций и т. д.

```
Консоль отладки Microsoft Visual Studio  
Immutable cities: London Paris Milan  
New cities: London Paris Milan Rio
```

Работа с сетевыми ресурсами



Пространство имен	Тип (пример)	Описание
System.Net	Dns, Uri, Cookie, WebClient, IPAddress	Для работы с DNS-серверами, идентификаторами URI, IP-адресами и т. д.
System.Net	FtpStatusCode, FtpWebRequest, FtpWebResponse	Для работы с FTP-серверами
System.Net	HttpStatusCode, HttpWebRequest, HttpWebResponse	Для работы с HTTP-серверами, то есть с сайтами
System.Net .Mail	Attachment, MailAddress, MailMessage, SmtpClient	Для работы с SMTP-серверами, то есть для отправки сообщений электронной почты
System.Net .NetworkInformation	IPStatus, NetworkChange, Ping, TcpStatistics	Для работы с низкоуровневыми сетевыми протоколами

Работа с сетевыми ресурсами

Работа с идентификаторами URI, DNS и IP-адресами



```
1 using System;
2 using System.Net;
3 using static System.Console;
4
5 namespace ConsoleApp
6 {
7     class Program
8     {
9         static void Main(string[] args)
10        {
11            Write("Enter a valid web address: ");
12            string url = ReadLine();
13            if (string.IsNullOrEmpty(url))
14            {
15                url = "http://world.episerver.com/cms/?q=pagetype";
16            }
17            var uri = new Uri(url);
18            WriteLine($"Scheme: {uri.Scheme}");
19            WriteLine($"Port: {uri.Port}");
20            WriteLine($"Host: {uri.Host}");
21            WriteLine($"Path: {uri.AbsolutePath}");
22            WriteLine($"Query: {uri.Query}");
23        }
24    }
25 }
```

Консоль отладки Microsoft Visual Studio

```
Enter a valid web address:
Scheme: http
Port: 80
Host: world.episerver.com
Path: /cms/
Query: ?q=pagetype
```


Робота з мережевими ресурсами

Отримання IP-адреси та опитування сервера



```
IPHostEntry entry = Dns.GetHostEntry(uri.Host);
WriteLine($"{entry.HostName} has the following IP addresses:");
foreach (IPAddress address in entry.AddressList)
{
    WriteLine($" {address}");
}

var ping = new Ping();
PingReply reply = ping.Send(uri.Host);
WriteLine($"{uri.Host} was pinged, and replied: {reply.Status}.");
if (reply.Status == IPStatus.Success)
{
    WriteLine($"Reply from {reply.Address} took { reply.RoundtripTime:N0}ms");
}
```



NetworkUsage.cs

```
world.episerver.com has the following IP addresses:
 217.114.90.249
world.episerver.com was pinged, and replied: TimedOut.
```

Робота з типами та атрибутами

Рефлексія



- технологія програмування, позволяющая коду понимать самого себя и управлять им.
 - Код описується метаданими за допомогою атрибутів.
 - Атрибути можуть використовуватись на кількох рівнях: до збірок, до типів та їх членів.

```
// атрибут уровня сборки
[assembly: AssemblyTitle("Working with Reflection")]

[Serializable] // атрибут уровня типа

public class Person
// атрибут уровня члена
[Obsolete("Deprecated: use Run instead.")]
public void Walk()
{
    // ...
}
```

Робота з типами та атрибутами

Вказування версій збірок



- В .NET [номера версий](#) — это комбинация из трех чисел с двумя необязательными добавочными номерами.
- По правилам семантического указания номеров версий:
 - основной — фундаментальные изменения;
 - дополнительный — незначительные изменения, в том числе новые функции и исправление ошибок;
 - патч — незначительные исправления ошибок.
- Как вариант, номер версии может также включать:
 - предрелиз — неподдерживаемые релизы для предварительного ознакомления;
 - номер сборки — ночные сборки.

Робота з типами та атрибутами

Зчитування метаданих про збірку

```
1 using static System.Console;
2 using System;
3 using System.Reflection;
4
5 namespace ConsoleApp
6 {
7     class Program
8     {
9         static void Main(string[] args)
10         {
11             WriteLine("Assembly metadata:");
12             Assembly assembly = Assembly.GetEntryAssembly();
13             WriteLine($" Full name: {assembly.FullName}");
14             WriteLine($" Location: {assembly.Location}");
15             var attributes = assembly.GetCustomAttributes();
16             WriteLine($" Attributes:");
17             foreach (Attribute a in attributes)
18             {
19                 WriteLine($"{a.GetType()}");
20             }
21
22             var version = assembly.GetCustomAttribute<AssemblyInformationalVersionAttribute>();
23             WriteLine($" Version: {version.InformationalVersion}");
24             var company = assembly.GetCustomAttribute<AssemblyCompanyAttribute>();
25             WriteLine($" Company: {company.Company}");
26         }
27     }
28 }
```

Консоль отладки Microsoft Visual Studio

```
Assembly metadata:
Full name: ConsoleApp, Version=1.0.0.0, Culture=neutral, PublicKeyToken=null
Location: C:\Users\squasson\source\repos\ConsoleApp\ConsoleApp\bin\Release\netcoreapp2.2\ConsoleApp.dll
Attributes:
System.Runtime.CompilerServices.CompilationRelaxationsAttribute
System.Runtime.CompilerServices.RuntimeCompatibilityAttribute
System.Diagnostics.DebuggableAttribute
System.Runtime.Versioning.TargetFrameworkAttribute
System.Reflection.AssemblyCompanyAttribute
System.Reflection.AssemblyConfigurationAttribute
System.Reflection.AssemblyFileVersionAttribute
System.Reflection.AssemblyInformationalVersionAttribute
System.Reflection.AssemblyProductAttribute
System.Reflection.AssemblyTitleAttribute
Version: 1.0.0
Company: ConsoleApp
```



Робота з типами та атрибутами

Явне встановлення метаданих



- В .NET Framework эти значения устанавливаются путем добавления атрибутов в файл исходного кода на языке C#.
 - [assembly: AssemblyCompany("Cherkasy State Business College")]
 - [assembly: AssemblyVersion("1.0.0")]
- У .NET Core змінюємо відповідний csproj-файл проекту:

```
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>netcoreapp2.2</TargetFramework>
6     <Version>1.1.0</Version>
7     <Company>Cherkasy State Business College</Company>
8   </PropertyGroup>
9
10 </Project>
```

Version: 1.1.0

Company: Cherkasy State Business College

Робота з типами та атрибутами

Створення власних атрибутів



- Визначити власні атрибути можна в породженому класі від класу `Attribute`.

```
1  using System;
2
3  namespace ConsoleApp
4  {
5      [AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, AllowMultiple = true)]
6      public class CoderAttribute : Attribute
7      {
8          public string Coder { get; set; }
9          public DateTime LastModified { get; set; }
10         public CoderAttribute(string coder, string lastModified)
11         {
12             Coder = coder;
13             LastModified = DateTime.Parse(lastModified);
14         }
15     }
16 }
```

- Додамо в клас `Program` метод з атрибутами `Coder`:

```
[Coder("Stanislav Marchenko", "11 September 2019")]
[Coder("4th Course", "12 September 2019")]
public static void DoStuff()
{ }
```

Робота з типами та атрибутами

Інший код у класі Program



```
static void Main(string[] args)
{
    Assembly assembly = Assembly.GetEntryAssembly();

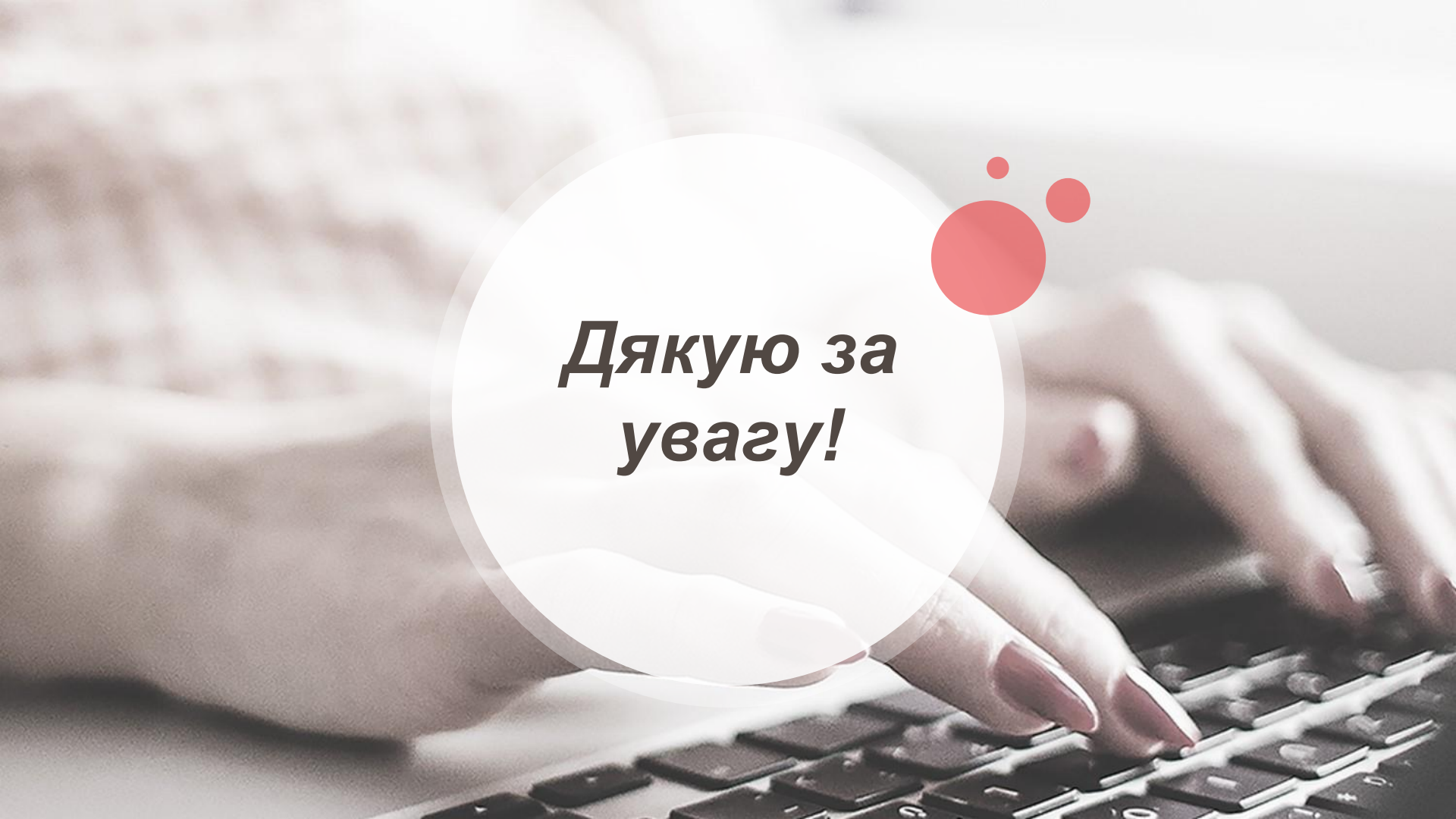
    WriteLine($"Types:");
    Type[] types = assembly.GetTypes();
    foreach (Type type in types)
    {
        WriteLine($" Name: {type.FullName}");
        MemberInfo[] members = type.GetMembers();
        foreach (MemberInfo member in members)
        {
            WriteLine($" {member.MemberType}: {member.Name} ({ member.DeclaringType.Name})");
            var coders = member.GetCustomAttributes<CoderAttribute>().OrderByDescending(c => c.LastModified);
            foreach (CoderAttribute coder in coders)
            {
                WriteLine($" Modified by {coder.Coder} on { coder.LastModified.ToShortDateString()}");
            }
        }
    }
}
```

Консоль отладки Microsoft Visual Studio

```
Types:
Name: ConsoleApp.CoderAttribute
Method: get_Coder (CoderAttribute)
Method: set_Coder (CoderAttribute)
Method: get_LastModified (CoderAttribute)
Method: set_LastModified (CoderAttribute)
Method: Equals (Attribute)
Method: GetHashCode (Attribute)
Method: get_TypeId (Attribute)
Method: Match (Attribute)
Method: IsDefaultAttribute (Attribute)
Method: ToString (Object)
Method: GetType (Object)
Constructor: .ctor (CoderAttribute)
Property: Coder (CoderAttribute)
Property: LastModified (CoderAttribute)
Property: TypeId (Attribute)
Name: ConsoleApp.Program
Method: DoStuff (Program)
Modified by 4th Course on 12.09.2019
Modified by Stanislav Marchenko on 11.09.2019
Method: ToString (Object)
Method: Equals (Object)
Method: GetHashCode (Object)
Method: GetType (Object)
Constructor: .ctor (Program)
Name: ConsoleApp.Program+<c
Method: ToString (Object)
Method: Equals (Object)
Method: GetHashCode (Object)
Method: GetType (Object)
Constructor: .ctor (<c)
Field: <9 (<c)
Field: <9_1_0 (<c)
```



Технологія Reflection та її можливості в .NET

A close-up, shallow depth-of-field photograph of a person's hands typing on a laptop keyboard. The hands have light-colored nail polish. A semi-transparent white circle is centered over the keyboard, containing the text 'Дякую за увагу!'. To the right of the circle, there are three red circles of varying sizes, resembling a bubble or a decorative element. The background is blurred, showing more of the laptop and the person's arms.

***Дякую за
увагу!***