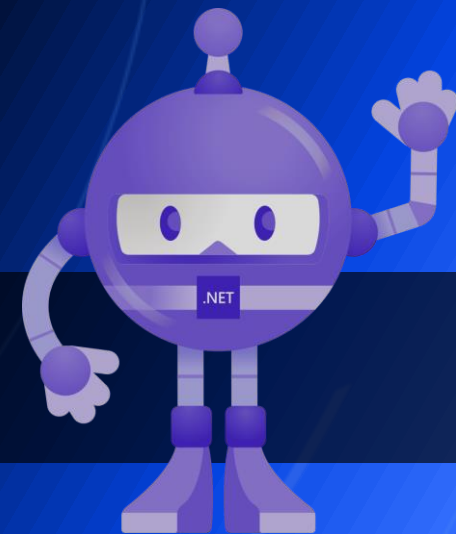


ЧТО НОВОГО

В .NET9 И C#13

📍 @SBenzenko

Сергей Бензенко





```
home.asp
1 <!--#INCLUDE FILE=include/functions.asp-->
2
3
4 <%
5     Dim iTranType : iTranType = CInt(RQ("tp"))
6
7     if ( RQ("di") = "" and GetSes("sup_id") = "" and RQ("trans") = "" ) then _
8         AbortToPage iSecure, GLM("SES_EXP"), "history.go(-2);", SIDE_NOTIFY
9
10    Dim bCheckSend : bCheckSend = True
11    Dim sNtfAction : sNtfAction = "CLOSE"
12    Dim iMarkAsRead : iMarkAsRead = 0
13    Dim sTransID : sTransID = RQ("trans")
14    Dim sPriceFld : sPriceFld = GLM("PRICE")
15    Dim sCurrMsg : sCurrMsg = GLM("SPECIFY_CUR")
16 >%
17
18 <BODY TOPMARGIN=5 LEFTMARGIN=5 RIGHTMARGIN=5 BOTTOMMARGIN=5 LINK=<%=sLink%> ALINK=<%=sAlink%> VLINK=<%=sVLink%>
19 <form NAME="f" METHOD="POST" ACTION="home.htm?tp=<%=iTranType%>#form">
20 <input TYPE="HIDDEN" NAME="act" VALUE="">
21 <input TYPE="HIDDEN" NAME="sbt" VALUE="">
22
23 <table WIDTH="100%" CELLPADDING="0" CELLSPACING="0" BORDER="0">
24     <tr>
25         <td>
26             <SPAN CLASS=title><%=sTitle%></span><br>
27             <img SRC="..img/dtc.gif" WIDTH=4><br>
28
29         <td ALIGN=RIGHT WIDTH=25% CLASS=small NOWRAP>
30
31             <A onmouseover="st(<%=GLM("SUP_INQ") %>);return true" onmouseout="st(ds);return true" HREF="Javascript:history.go(-1);"><%=GLM("SUP_INQ") %></a><img SRC="..img/dtc.gif" WIDTH=8><br>
32             <A onmouseover="st(<%=GLM("CLOSE") %>);return true" onmouseout="st(ds);return true" HREF="Javascript>window.close();"><%=GLM("CLOSE") %></a><img SRC="..img/dtc.gif" WIDTH=8></td>
33
34         <img SRC="..img/dtc.gif" WIDTH=8><br>
35
36     <% if ( iStyle <= 1 ) then %>
37
38         <SPAN>
39
40         <center>
41
42             <a onmouseover="return true;" onmouseout="st(ds);return true;" HREF="Javascript:act(document.f.txtAdd.value)"><FONT COLOR=BLACK><b><%=Replace(GLM("ADD_LINES"),[NUM],"/")></b><input NAME=txtAdd SIZE=2 MAXLENGTH=1 VALUE=1><a onmouseover=""return
43             <%=GLM("ADD_LINES") %></a></font></a><br>
44             <%=GLM("ADD_LINES") %><br>
45             <a onmouseover="return true;" onmouseout="st(ds);return true;" HREF="Javascript:act(0)"><FONT COLOR=BLACK><b><%=GLM("DEL_CHECKED") %></b></font></a><br>
46             <%=GLM("DEL_CHECKED") %><br>
47
48         <img SRC="..img/dtc.gif" HEIGHT="4"><br>
49     <% end if %>
50
51     <table WIDTH="100%" CELLPADDING="1" CELLSPACING="0" BORDER="0" BGCOLOR=#CC0033>
52         <tr CLASS=headbar>
53             <td ALIGN="MIDDLE" NOWRAP>
54                 <input TYPE="CHECKBOX" NAME="parts" VALUE="0" onclick="bchk(this.checked)" CHECKED><%=GLM("PARTS") %></td>
55             <td NOWRAP VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("EN") %></b></font></td>
56             <td NOWRAP VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("MFR") %></b></font></td>
57             <td NOWRAP VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("DC") %></b></font></td>
58             <td NOWRAP WIDTH="100%" VALIGN="BOTTOM" CLASS="headbar"><font color=BLACK><b><%=GLM("DESC") %></b></font></td>
59         </tr>
60     </table>
61 </BODY>
```

1

2

3

{.Net}



➤ .NET Разработчик
[@NetDeveloperDiary](#)

.NET 6

Nov 2021

.NET 7

Nov 2022

.NET 8

Nov 2023

May 2024

.NET 9

Nov 2024

.NET 10

Nov 2025



LONG TERM SUPPORT
Patches for 36 months

STANDARD TERM SUPPORT
Patches for 18 months



.NET 8.0 upgrade report

Generation time 06/18/2024 3:56:02 PM (duration 11 seconds)

[Dashboard](#)[Projects](#)[Aggregate issues](#)

ISSUE	DESCRIPTION	STATE	SEVERITY	INCIDENTS	STORY POINTS
Api.0001	API does not exist	Active	Mandatory	225	225
<p>LOCATION</p> <p>eShopPorted\Controllers\PicController.cs (T:System.Web.Mvc.Controller) (9,34)</p> <p>eShopPorted\Controllers\PicController.cs (M:System.Web.Mvc.Controller.#ctor) (17,9)</p> <p>eShopPorted\Controllers\PicController.cs (T:System.Web.Mvc.ActionResult) (23,9)</p> <p>eShopPorted\Controllers\PicController.cs (T:System.Web.Mvc.HttpGetAttribute) (23,10)</p> <p>eShopPorted\Controllers\PicController.cs (M:System.Web.Mvc.HttpGetAttribute.#ctor) (23,10)</p>		<p>STATE</p> <p>Active</p>	<p>Mandatory</p> <p>T:System.Web.Mvc.Controller</p> <p>API does not exist in the selected version of .NET. See breaking changes documentation for alternative or replacement functionality.</p> <p><i>Source</i></p> <p>Controller</p> <p>Breaking changes in .NET API documentation</p>	<p>Ask Copilot</p>	
Api.0002	API available in NuGet package	Active	Potential	274	274
Api.0003	API is obsolete	Active	Optional	3	3
NuGet.0001	NuGet package is incompatible	Active	Mandatory	10	10
NuGet.0002	NuGet package upgrade is recommended	Active	Potential	14	14

СТРАТЕГИЯ

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire

The image compares two development environments. On the left, 'MY WEB APP' is a simple form with fields for 'Full Name', 'Username', 'Password', 'Re Password', 'Address', 'Age', and 'Gender'. On the right, 'MY ASPIRE DASHBOARD' is a complex monitoring interface titled 'net mono' showing a table of 'Resources' with columns for Type, Name, State, Start time, Source, Endpoints, Logs, and Details. Below the table are several charts and summary cards for metrics like Total Requests (39339), Total Unhandled Exceptions (117), and Requests by Protocol (http: 39339, https: 7361, https: 31978).

Type	Name	State	Start time	Source	Endpoints	Logs	Details
Container	postgres	Running	9:59:13 PM	postgres:16.2	localhost:56212	View	View
Container	postgres	Running	9:59:13 PM	postgres:admin@16.2	http://localhost:56213	View	View
Container	storage	Running	9:59:13 PM	ms-microsoft.com/azur...	localhost:10000 localhost:56214 localhost:56215	View	View
Project	backend	Running	9:59:13 PM	GameStore.Api.csproj	http://localhost:1115	View	View
Project	frontend	Running	9:59:13 PM	GameStore.Frontend.cs...	http://localhost:5250	View	View

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire
- Native AOT и DATAS

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire
- Native AOT и DATAS
- Visual Studio и VS Code

СТРАТЕГИЯ

1. Развитие .NET для облачной разработки

- .NET Aspire
- Native AOT и DATAS
- Visual Studio и VS Code

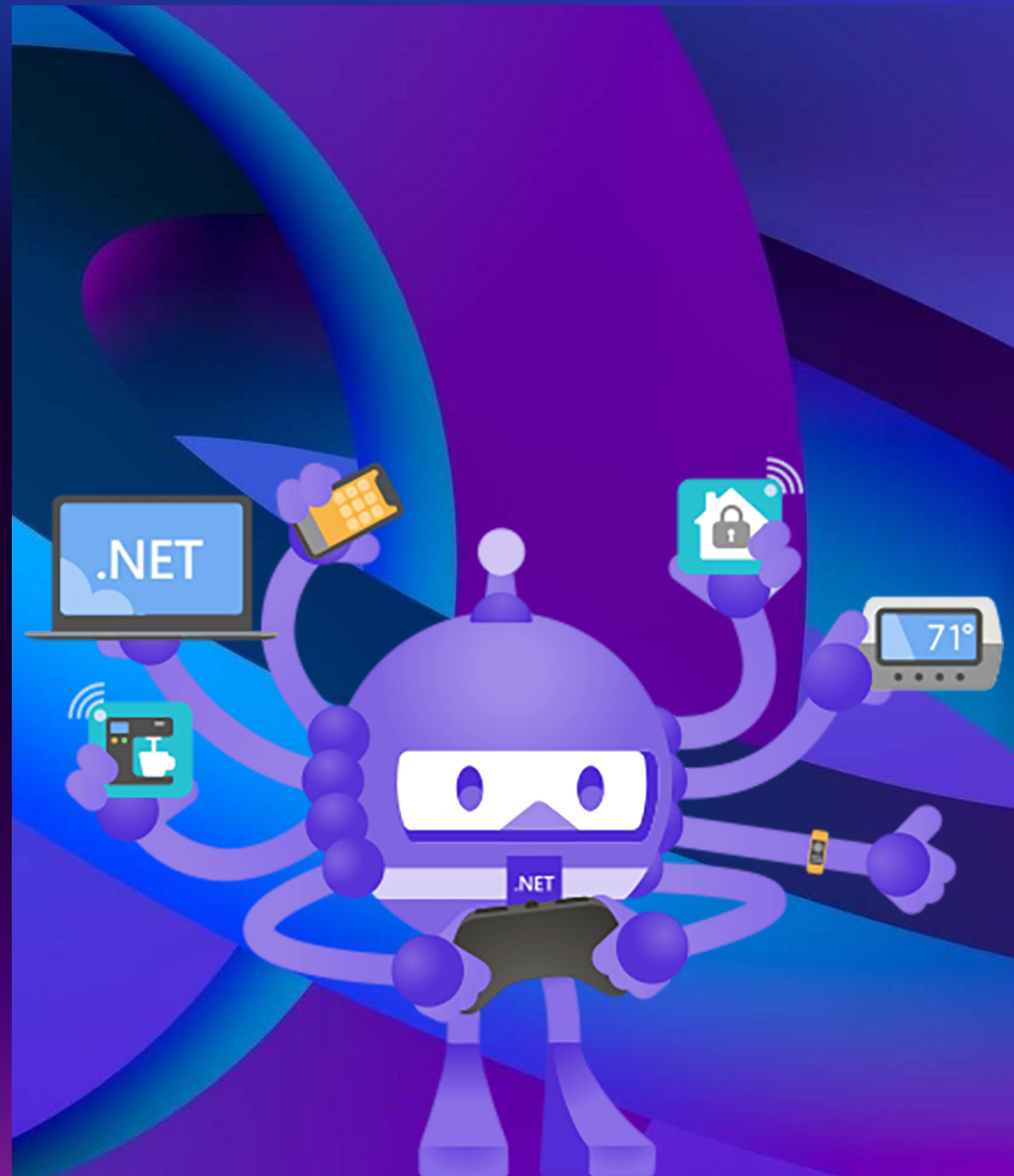
2. AI

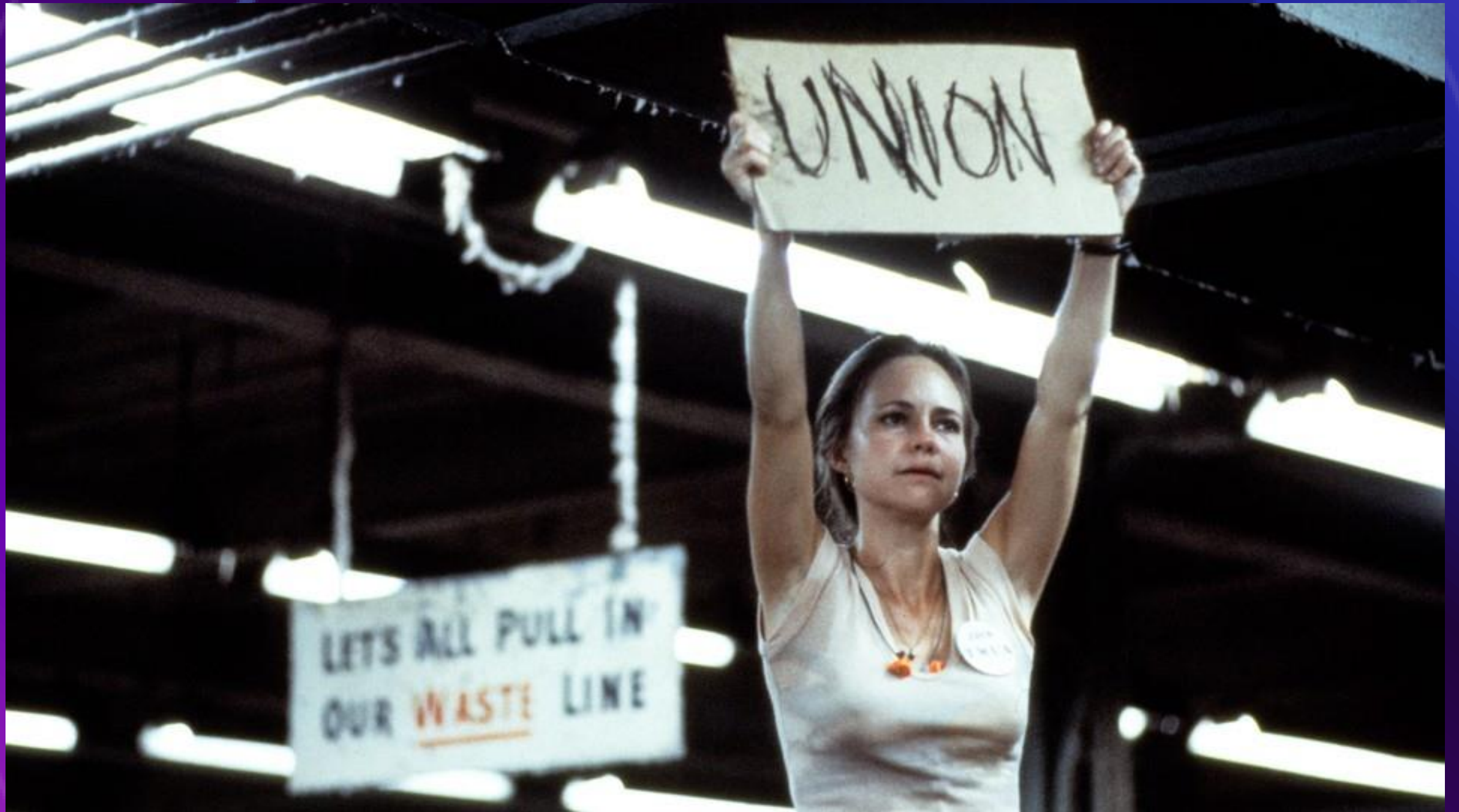
1. C# и .NET

2. ASP.NET Core и Blazor

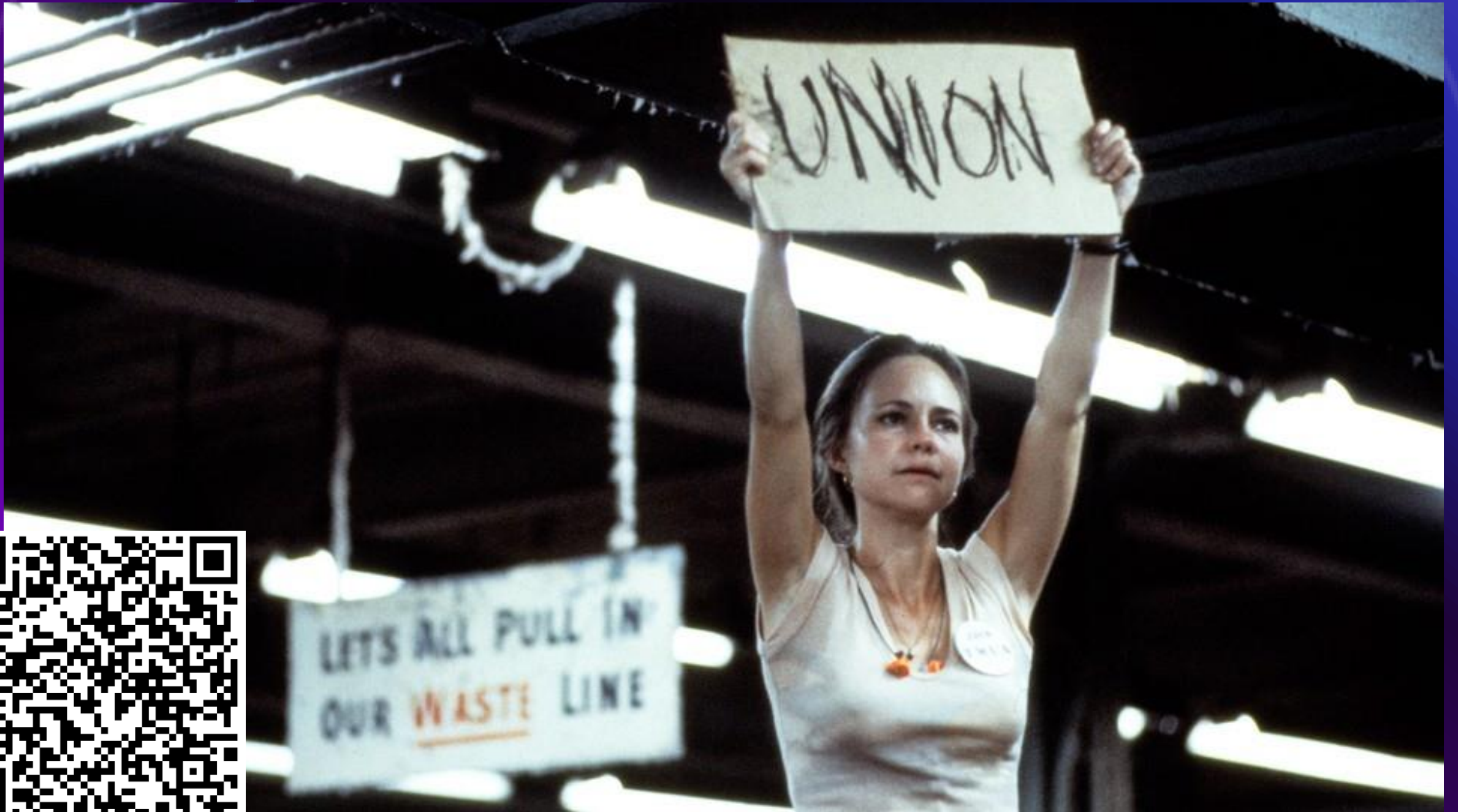
3. EF Core

4. Desktop





Картинка: <https://www.theatlantic.com/entertainment/archive/2019/03/norma-rae-40th-anniversary-racial-solidarity-unions-labor-movement/583924/>



Картинка: <https://www.theatlantic.com/entertainment/archive/2019/03/norma-rae-40th-anniversary-racial-solidarity-unions-labor-movement/583924/>
Installing .NET 9 (14/160) ...

C# И .NET

1. Полуавтоматические свойства
2. Улучшения SearchValues
3. Коллекции в ranges
4. Обработка асинхронных задач
5. Изменения ref-переменных
6. Regex.EnumerateSplits
7. Доступ к коллекциям с помощью Span'ов
8. Флаги функций
9. Новые методы LINQ
10. Новый тип блокировки
11. UUID версии 7
12. Парсинг имени типа

Типы-расширения



Типы-расширения

```
1 | JsonData
25 | var data = JsonData.ParseAsJson();
26 | var json = string.CreateIndented(data);
27 | WriteLine(json);
28 |
29 |
30 | implicit extension JsonString for string
31 | {
32 |     private static readonly JsonSerializerOptions s_indentedOptions = new() { WriteIndented = true };
33 |
34 |     public JsonElement ParseAsJson()
35 |     => JsonDocument.Parse(this.Trim()).RootElement;
36 |
37 |     public static string CreateInde
38 |     => element.ValueKind != Jso
39 |     ? JsonSerializer.Serial
40 |     : string.Empty;
```

IntelliCode suggestion based on this context:

- ★ Replace: string string.Replace(string oldValue, string? newValue) (+ 3 overloads) Returns a new string in which all occurrences of a specified string in the current instance are replaced with another specified string.
- ★ ParseAsJson
- ★ Equals
- ★ Contains
- ★ CompareTo



Типы-расширения



Полуавтоматические свойства

Полуавтоматические свойства

```
public string Name { get; set; }
```

Полуавтоматические свойства

```
public string Name { get; set; }
```

```
private string name;
```

```
public string Name  
{  
    get => name;  
    set => name = value;  
}
```

C#12

Полуавтоматические свойства

```
public string Name { get; set; }
```

```
private string name;
```

```
public string Name  
{  
    get => name;  
    set => name = value.Trim();  
}
```

C#12

Полуавтоматические свойства

```
public string Name { get; set; }
```

```
private string name;  
  
public string Name  
{  
    get => name;  
    set => name = value.Trim();  
}
```

C#12

```
public string Name  
{  
    get;  
    set => field = value.Trim();  
}
```

C#13+

Полуавтоматические свойства

```
private string field;  
  
public string Field  
{  
    get => field;  
    set => field = value.Trim();  
}
```


Полуавтоматические свойства

```
private string field;  
  
public string Field  
{  
    get => field;  
    set => field = value.Trim();  
}
```

```
private string field;  
  
public string Field  
{  
    get => @field;  
    set => @field = value.Trim();  
}
```

Улучшения SearchValues

Улучшения SearchValues

```
var vowels = SearchValues.Create(new[] { 'a', 'e', 'i', 'o', 'u' });
```

Улучшения SearchValues

```
var vowels = SearchValues.Create(new[] { 'a', 'e', 'i', 'o', 'u' });  
ReadOnlySpan<char> text = "Hello, world!";
```

Улучшения SearchValues

```
var vowels = SearchValues.Create(new[] { 'a', 'e', 'i', 'o', 'u' });  
ReadOnlySpan<char> text = "Hello, world!";  
Console.WriteLine(text.ContainsAny(vowels));
```

True

Улучшения SearchValues

```
var sorts = SearchValues.Create(  
    ["Bubble", "Gnome", "Bogosort"],  
    StringComparison.OrdinalIgnoreCase);
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(  
    ["Bubble", "Gnome", "Bogosort"],  
    StringComparison.OrdinalIgnoreCase);
```

```
var text = "I think the bubble sort would be the wrong way to go.";
```

Улучшения SearchValues

```
var sorts = SearchValues.Create(  
    ["Bubble", "Gnome", "Bogosort"],  
    StringComparison.OrdinalIgnoreCase);
```

```
var text = "Искусство программирования";
```



Улучшения SearchValues

```
var sorts = SearchValues.Create(  
    ["Bubble", "Gnome", "Bogosort"],  
    StringComparison.OrdinalIgnoreCase);
```

```
var text = " ;
```



```
console.WriteLine(text.AsSpan().IndexOfAny(sorts));
```

3141592

Улучшения SearchValues

```
var sorts = SearchValues.Create(  
    ["Bubble", "Gnome", "Bogosort"],  
    StringComparison.OrdinalIgnoreCase);
```

```
var text = "Искусство программирования";
```



```
console.WriteLine(text.AsSpan().IndexOfAny(sorts));
```

3141592

Коллекции в рамках

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}

PrintNums(1, 2, 3);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}

PrintNums(1, 2, 3);
PrintNums([1, 2, 3]);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}

PrintNums(1, 2, 3);
PrintNums([1, 2, 3]);
```


Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}

PrintNums(1, 2, 3);
PrintNums([1, 2, 3]);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}

List<int> list = [1, 2, 3];
PrintNums(list);
```

Коллекции в params

```
void PrintNums(params int[] nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params IEnumerable<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}
void PrintNums(params ReadOnlySpan<int> nums)
{
    foreach (var n in nums)
        Console.WriteLine(n);
}

List<int> list = [1, 2, 3];
PrintNums(list);
```

Обработка задач по мере их завершения

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

var results = await Task.WhenAll(tasks);
foreach (var r in results)
    Console.WriteLine($"{r} ");
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

var results = await Task.WhenAll(tasks);
foreach (var r in results)
    Console.WriteLine($"{r} ");
```



Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

var results = await Task.WhenAll(tasks);
foreach (var r in results)
    Console.WriteLine($"{r} ");
```

1 2 3 4 5 6 7 8 9 10

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

await foreach (var finished in Task.WhenEach(tasks))
{
    Console.WriteLine(await finished);
}
```

Обработка задач по мере их завершения

```
async Task<int> Calculate(int order)
{
    var wait = Random.Shared.Next(500, 5000);
    await Task.Delay(wait);
    return order;
}

var tasks = Enumerable.Range(1, 10).Select(Calculate);

await foreach (var finished in Task.WhenEach(tasks))
{
    Console.WriteLine(await finished);
}
```

4 7 1 5 2 10 6 3 8 9

Изменения gef-переменных

Изменения ref-переменных



1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()  
{  
    var result = await AsyncMethod1();  
    var span = result.AsSpan();  
    DoSomething(span);  
    await AsyncMethod2();  
}
```

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()  
{  
    var result = await AsyncMethod1();  
    var span = result.AsSpan();  
    DoSomething(span);  
    await AsyncMethod2();  
}
```

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()  
{  
    var result = await AsyncMethod1();  
    var span = result.AsSpan();  
    DoSomething(span);  
    await AsyncMethod2();  
}
```

2. Интерфейсы для ref-структур

Изменения ref-переменных

1. ref и unsafe в async методах и итераторах

```
async Task MyMethodAsync()  
{  
    var result = await AsyncMethod1();  
    var span = result.AsSpan();  
    DoSomething(span);  
    await AsyncMethod2();  
}
```

2. Интерфейсы для ref-структур

3. Ограничение allows ref struct

```
public void DoSomething<T>() where T : allows ref struct  
{  
    //...  
}
```


Regex.EnumerateSplits

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";  
  
foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))  
{  
    Console.WriteLine(s);  
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine($"{input[r]}");
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine($"{input[r]}");
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine($"{input[r]}");
}
```

Regex.EnumerateSplits

Regex.Split

```
string input = "Привет, DotNext! Как дела?";

foreach (var s in Regex.Split(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine(s);
}
```

Regex.EnumerateSplits

```
ReadOnlySpan<char> input = "Привет, DotNext! Как дела?";

foreach (Range r in Regex.EnumerateSplits(input, "[аоуиэыеёюя]"))
{
    Console.WriteLine($"{input[r]}");
}
```

Доступ к коллекциям с помощью Span'ов

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```


Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplits(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

Доступ к коллекциям с помощью Span'ов

```
static Dictionary<string, int> CountWords(ReadOnlySpan<char> input)
{
    Dictionary<string, int> counts = new(StringComparer.OrdinalIgnoreCase);

    Dictionary<string, int>.AlternateLookup<ReadOnlySpan<char>> lookup =
        counts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();

    foreach (Range r in Regex.EnumerateSplices(input, @"\b\w+\b"))
    {
        ReadOnlySpan<char> word = input[r];
        lookup[word] = lookup.TryGetValue(word, out int count) ? count + 1 : 1;
    }

    return counts;
}
```

```
var text = "Искусство программирования Искусство программирования Искусство программирования Искусство программирования Искусство программирования Искусство программирования Искусство программирования";
foreach (var word in CountWords(text))
    Console.WriteLine(word);
```

Флаги функций с поддержкой тримминга

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
    [FeatureSwitchDefinition("Feature.IsSupported")]
    internal static bool IsSupported =>
        AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
            ? isEnabled
            : true;

    internal static void Implementation() => ...;
}
```

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
    [FeatureSwitchDefinition("Feature.IsSupported")]
    internal static bool IsSupported =>
        AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
            ? isEnabled
            : true;

    internal static void Implementation() => ...;
}
```

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
    [FeatureSwitchDefinition("Feature.IsSupported")]
    internal static bool IsSupported =>
        AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
            ? isEnabled
            : true;

    internal static void Implementation() => ...;
}
```

```
<ItemGroup>
  <RuntimeHostConfigurationOption Include="Feature.IsSupported"
    value="false" Trim="true" />
</ItemGroup>
```

MyApp.csproj

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
    [FeatureSwitchDefinition("Feature.IsSupported")]
    internal static bool IsSupported =>
        AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
            ? isEnabled
            : true;

    internal static void Implementation() => ...;
}
```

```
<ItemGroup>
  <RuntimeHostConfigurationOption Include="Feature.IsSupported"
    value="false" Trim="true" />
</ItemGroup>
```

MyApp.csproj

Флаги функций с поддержкой тримминга

FeatureSwitchDefinitionAttribute

```
public class Feature
{
    [FeatureSwitchDefinition("Feature.IsSupported")]
    internal static bool IsSupported =>
        AppContext.TryGetSwitch("Feature.IsSupported", out bool isEnabled)
            ? isEnabled
            : true;

    internal static void Implementation() => ...;
}
```

```
<ItemGroup>
  <RuntimeHostConfigurationOption Include="Feature.IsSupported"
    value="false" Trim="true" />
</ItemGroup>
```

MyApp.csproj

```
if (Feature.IsSupported)
    Feature.Implementation();
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
    [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
    internal static bool IsDynamicSupported =>
        RuntimeFeature.IsDynamicCodeSupported;

    [RequiresDynamicCode("Feature requires dynamic code support.")]
    internal static void DynamicImplementation() => ...; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
    [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
    internal static bool IsDynamicSupported =>
        RuntimeFeature.IsDynamicCodeSupported;

    [RequiresDynamicCode("Feature requires dynamic code support.")]
    internal static void DynamicImplementation() => ...; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
    [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
    internal static bool IsDynamicSupported =>
        RuntimeFeature.IsDynamicCodeSupported;

    [RequiresDynamicCode("Feature requires dynamic code support.")]
    internal static void DynamicImplementation() => ...; // Использует dynamic
}
```


Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
    [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
    internal static bool IsDynamicSupported =>
        RuntimeFeature.IsDynamicCodeSupported;

    [RequiresDynamicCode("Feature requires dynamic code support.")]
    internal static void DynamicImplementation() => ...; // Использует dynamic
}
```

Флаги функций с поддержкой тримминга

FeatureGuardAttribute

```
public class Feature
{
    [FeatureGuard(typeof(RequiresDynamicCodeAttribute))]
    internal static bool IsDynamicSupported =>
        RuntimeFeature.IsDynamicCodeSupported;

    [RequiresDynamicCode("Feature requires dynamic code support.")]
    internal static void DynamicImplementation() => ...; // Использует dynamic
}
```

```
<PublishAot>true</PublishAot>
```

```
MyApp.csproj
```

```
if (Feature.IsDynamicSupported)
    Feature.DynamicImplementation();
```

Новые методы LINQ

Новые методы LINQ

1. CountBy

```
public record Person(string FirstName, string LastName);

List<Person> people = [
    new("Steve", "Jobs"),
    new("Elon", "Musk"),
    new("Steve", "Smith")
];

foreach (var p in people.CountBy(p => p.FirstName))
    Console.WriteLine($"{p.Value} людей с именем {p.Key}");
```

```
2 людей с именем Steve
1 людей с именем Elon
```

НОВЫЕ МЕТОДЫ LINQ

2. AggregateBy

```
public record Person(string Name, string Department, int Salary);
```

```
List<Person> people = [  
    new("Jobs", "Sales", 180),  
    new("Musk", "IT", 290),  
    new("Smith", "IT", 120)  
];
```

```
var aggregateBy = people.AggregateBy(  
    p => p.Department, x => 0, (x, y) => x + y.Salary);
```

```
foreach (var agg in aggregateBy)  
    Console.WriteLine($"ФОРТ отдела {agg.Key}: {agg.Value}");
```

```
ФОРТ отдела Sales: 180  
ФОРТ отдела IT: 410
```

Новые методы LINQ

3. Index^o

```
public record Person(string FirstName, string LastName);
```

```
List<Person> people = [  
    new("Steve", "Jobs"),  
    new("Elon", "Musk"),  
    new("Steve", "Smith")  
];
```

```
foreach (var (index, item) in people.Index())  
    Console.WriteLine($"№{index}: {item}");
```

```
№1: Steve Jobs  
№2: Elon Musk  
№3: Steve Smith
```

Новый тип блокировки

Новый тип блокировки

```
object _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 12

Новый тип блокировки

```
object _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 12

```
Lock _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 13+

Новый тип блокировки

```
object _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 12

```
Lock _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 13+

Новый тип блокировки

```
object _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 12

```
Lock _lock = new();  
void DoSomething()  
{  
    lock (_lock)  
    {  
        // что-то делаем  
    }  
}
```

C# 13+

UUID версии 7

UUID версии 7

```
for (int i = 0; i < 5; i++)  
    Console.WriteLine(Guid.NewGuid());
```

```
f529a20a-6d23-4568-9d3e-01b18f1ff40e  
b2c9f7ee-fcf4-4dbf-8194-9f0354a05d79  
528419ee-ab84-451d-b568-4bd26fca27be  
720c11de-e9fa-4959-bb1f-6bf1c6d3bea6  
d581b627-eb46-41d7-876d-58c0d9af78d7
```

UUID версии 7

```
for (int i = 0; i < 5; i++)  
    Console.WriteLine(Guid.NewGuid());
```

```
f529a20a-6d23-4568-9d3e-01b18f1ff40e  
b2c9f7ee-fcf4-4dbf-8194-9f0354a05d79  
528419ee-ab84-451d-b568-4bd26fca27be  
720c11de-e9fa-4959-bb1f-6bf1c6d3bea6  
d581b627-eb46-41d7-876d-58c0d9af78d7
```

```
for (int i = 0; i < 5; i++)  
    Console.WriteLine(Guid.CreateVersion7());
```

```
01918fe2-56f1-7131-8e68-f1d37b8dc2ce  
01918fe2-56f1-7e85-825c-217376c27fa8  
01918fe2-56f1-78ea-a008-606df446098e  
01918fe2-56f1-7023-bdfe-9fe6a8f49b9c  
01918fe2-56f1-71fc-80fb-791913750271
```

UUID версии 7

```
for (int i = 0; i < 5; i++)  
    Console.WriteLine(Guid.NewGuid());
```

```
f529a20a-6d23-4568-9d3e-01b18f1ff40e  
b2c9f7ee-fcf4-4dbf-8194-9f0354a05d79  
528419ee-ab84-451d-b568-4bd26fca27be  
720c11de-e9fa-4959-bb1f-6bf1c6d3bea6  
d581b627-eb46-41d7-876d-58c0d9af78d7
```

```
for (int i = 0; i < 5; i++)  
    Console.WriteLine(Guid.CreateVersion7());
```

```
01918fe2-56f1-7131-8e68-f1d37b8dc2ce  
01918fe2-56f1-7e85-825c-217376c27fa8  
01918fe2-56f1-78ea-a008-606df446098e  
01918fe2-56f1-7023-bdfe-9fe6a8f49b9c  
01918fe2-56f1-71fc-80fb-791913750271
```

```
var uuidTest = Guid.CreateVersion7(timeProvider.GetUtcNow());
```

Парсинг имени типа

```
using System.Reflection.Metadata;
```

```
class SerializationBinder
```

```
{
```

```
    public Type GetType(ReadOnlySpan<char> input)
```

```
    {
```

```
        if (!TypeName.TryParse(input, out TypeName parsed))
```

```
        {
```

```
            ...
```

```
        }
```

```
        if (parsed.IsSimple // не дженерик, не указатель, не массив  
            && parsed.AssemblyName.Name == "TrustedAssembly")
```

```
        {
```

```
            return Type.GetType(parsed.AssemblyQualifiedName, throwOnError: true);
```

```
        }
```

```
        ..
```

```
    }
```

```
}
```


Парсинг имени типа

```
using System.Reflection.Metadata;

class SerializationBinder
{
    public Type GetType(ReadOnlySpan<char> input)
    {
        if (!TypeName.TryParse(input, out TypeName parsed))
        {
            ...
        }
        if (parsed.IsSimple // не дженерик, не указатель, не массив
            && parsed.AssemblyName.Name == "TrustedAssembly")
        {
            return Type.GetType(parsed.AssemblyQualifiedName, throwOnError: true);
        }
        ...
    }
}
```

НОВЫЕ ТИПЫ И API

Base64Url

```
ReadOnlySpan<byte> bytes = ...;  
string encoded = Base64Url.EncodeToString(bytes);
```

НОВЫЕ ТИПЫ И API

Base64Url

```
ReadOnlySpan<byte> bytes = ...;  
string encoded = Base64Url.EncodeToString(bytes);
```

OrderedDictionary<TKey, TValue>

```
OrderedDictionary<string, int> d = new() { { "a", 1 }, { "b", 2 } };  
d.RemoveAt(1);  
d.Insert(0, "d", 4);
```

```
[d, 4] [a, 1]
```

НОВЫЕ ТИПЫ И API

Base64Url

```
ReadOnlySpan<byte> bytes = ...;  
string encoded = Base64Url.EncodeToString(bytes);
```

OrderedDictionary<TKey, TValue>

```
OrderedDictionary<string, int> d = new() { { "a", 1 }, { "b", 2 } };  
d.RemoveAt(1);  
d.Insert(0, "d", 4);
```

```
[d, 4] [a, 1]
```

ReadOnlySet<T>

```
var set = new HashSet<int> { 1, 2, 3 };  
var readOnlySet = new ReadOnlySet<int>(set);
```

НОВЫЕ ТИПЫ И API

Base64Url

```
ReadOnlySpan<byte> bytes = ...;  
string encoded = Base64Url.EncodeToString(bytes);
```

OrderedDictionary<TKey, TValue>

```
OrderedDictionary<string, int> d = new() { { "a", 1 }, { "b", 2 } };  
d.RemoveAt(1);  
d.Insert(0, "d", 4);
```

```
[d, 4] [a, 1]
```

ReadOnlySet<T>

```
var set = new HashSet<int> { 1, 2, 3 };  
var readOnlySet = new ReadOnlySet<int>(set);
```

TensorPrimitives и Tensor<T>

```
using System.Numerics.Tensors;  
ReadOnlySpan<double> vector1 = [1, 2, 3];  
ReadOnlySpan<double> vector2 = [4, 5, 6];  
Console.WriteLine(TensorPrimitives.CosineSimilarity(vector1, vector2));
```

```
0.9746318461970762
```

BinaryFormatter

```
System.Runtime.Serialization.Formatters (8.1.0.0) System.Runtime.Serialization.Formatters.Binary.BinaryFormatter Deserialize(Stream serializationSt

1 // Licensed to the .NET Foundation under one or more agreements.
2 // The .NET Foundation licenses this file to you under the MIT license.
3
4 > using ...
5
6
7 namespace System.Runtime.Serialization.Formatters.Binary
8 {
9     public sealed partial class BinaryFormatter : IFormatter
10    {
11        [RequiresDynamicCode(message: IFormatter.RequiresDynamicCodeMessage)]
12        [RequiresUnreferencedCode(message: IFormatter.RequiresUnreferencedCodeMessage)]
13        public object Deserialize(Stream serializationStream)
14            => throw new PlatformNotSupportedException(SR.BinaryFormatter_Removed);
15
16        [RequiresUnreferencedCode(message: IFormatter.RequiresUnreferencedCodeMessage)]
17        public void Serialize(Stream serializationStream, object graph)
18            => throw new PlatformNotSupportedException(SR.BinaryFormatter_Removed);
19    }
20 }
21
```



ASP.NET CORE И BLAZOR

1. Гибридный кэш
2. Умные компоненты
3. Новый вид сжатия статических ресурсов
4. OpenAPI
5. Blazor

Гибридный кэш

Гибридный кэш

Кэш в памяти

```
builder.Services.AddMemoryCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        async entry => await _repo.GetDataAsync(name, id, ct);  
    );  
}
```

Гибридный кэш

Кэш в памяти

```
builder.Services.AddMemoryCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        async entry => await _repo.GetDataAsync(name, id, ct);  
    );  
}
```

Распределённый кэш

```
builder.Services.AddStackExchangeRedisCache();  
...
```

Гибридный кэш

Кэш в памяти

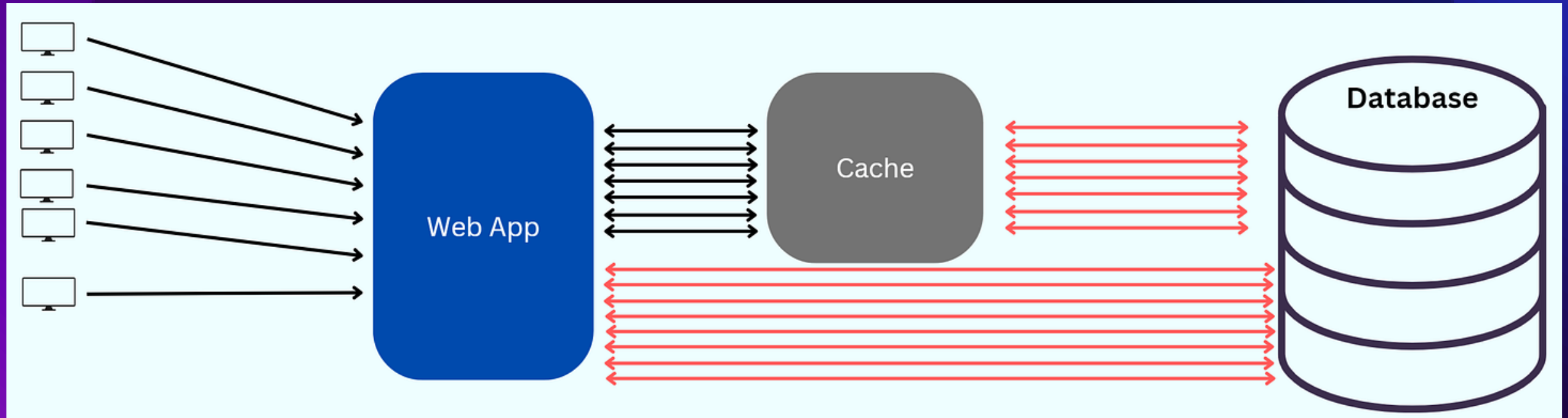
```
builder.Services.AddMemoryCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        async entry => await _repo.GetDataAsync(name, id, ct);  
    );  
}
```

Распределённый кэш

```
builder.Services.AddStackExchangeRedisCache();  
...
```

Гибридный кэш

«Паника в кэше»



Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        async cancel => await _repo.GetPostAsync(name, id, cancel),  
        token: ct  
    );  
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        async cancel => await _repo.GetPostAsync(name, id, cancel),  
        token: ct  
    );  
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        async cancel => await _repo.GetPostAsync(name, id, cancel),  
        token: ct  
    );  
}
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        (name, id), // вся информация для получения данных  
        static async (state, cancel) =>  
            await _repo.GetPostAsync(state.name, state.id, cancel),  
        token: ct  
    );  
}
```


Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        (name, id), // вся информация для получения данных  
        static async (state, cancel) =>  
            await _repo.GetPostAsync(state.name, state.id, cancel),  
        token: ct  
    );  
}
```

```
[ImmutableObject(true)]  
public sealed class SomeData { ... }
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        (name, id), // вся информация для получения данных  
        static async (state, cancel) =>  
            await _repo.GetPostAsync(state.name, state.id, cancel),  
        token: ct  
    );  
}
```

```
[ImmutableObject(true)]  
public sealed class SomeData { ... }
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache();  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        (name, id), // вся информация для получения данных  
        static async (state, cancel) =>  
            await _repo.GetPostAsync(state.name, state.id, cancel),  
        token: ct  
    );  
}
```

```
[ImmutableObject(true)]
```

```
public sealed class SomeData { ... }
```

Гибридный кэш

HybridCache

```
builder.Services.AddHybridCache().WithSerializer(...);  
...  
public async Task<SomeData> GetDataAsync(  
    string name, int id, CancellationToken ct = default)  
{  
    return await _cache.GetOrCreateAsync(  
        $"data:{name}:{id}", // уникальный ключ  
        (name, id), // вся информация для получения данных  
        static async (state, cancel) =>  
            await _repo.GetPostAsync(state.name, state.id, cancel),  
        token: ct  
    );  
}
```

```
[ImmutableObject(true)]  
public sealed class SomeData { ... }
```

.NET Smart Components



.NET Smart Components

Умное поле для ввода

Response

Respond to employee enquiry here...

.NET Smart Components

Умный выпадающий список

Expense category

.NET Smart Components

Умная вставка

Recipient

First name

Last name

Phone number

Address

Line 1

Line 2

City

State

Zip

Country

Submit



Smart Paste

MapStaticAssets

```
WebApplication1
1  var builder = WebApplication.CreateBuilder(args);
2
3  // Add services to the container.
4  builder.Services.AddRazorPages();
5
6  var app = builder.Build();
7
8  // Configure the HTTP request pipeline.
9  if (!app.Environment.IsDevelopment())
10 {
11     app.UseExceptionHandler("/Error");
12     // The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.
13     app.UseHsts();
14 }
15
16 app.UseHttpsRedirection();
17
18 app.UseRouting();
19
20 app.UseAuthorization();
21
22 app.MapStaticAssets();
23 app.MapRazorPages();
24
25 app.Run();
26
```

Solution Explorer

Search Solution Explorer (Ctrl+):

Solution 'WebApplication1' (1 of 1 project)

- WebApplication1
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - css
 - site.css
 - js
 - site.js
 - lib
 - bootstrap
 - dist
 - css
 - bootstrap.css
 - bootstrap-grid.css
 - bootstrap-reboot.css
 - bootstrap-utilities.css
 - js
 - bootstrap.js
 - LICENSE
 - jquery
 - dist
 - jquery.js
 - LICENSE.txt
 - jquery-validation
 - dist
 - additional-methods.js
 - jquery.validate.js
 - LICENSE.md
 - jquery-validation-unobtrusive
 - jquery.validate.unobtrusive.js
 - LICENSE.txt
 - favicon.ico
 - Pages

MapStaticAssets

Шаблон Razor Pages

File	Original	Compressed	% Reduction
bootstrap.min.css	163	17.5	89.26%
jquery.js	89.6	28	68.75%
bootstrap.min.js	78.5	20	74.52%
Total	331.1	65.5	80.20%

Fluent UI Blazor

File	Original	Compressed	% Reduction
fluent.js	384	73	80.99%
fluent.css	94	11	88.30%
Total	478	84	82.43%

MudBlazor

File	Original	Compressed	% Reduction
MudBlazor.min.css	541	37.5	93.07%
MudBlazor.min.js	47.4	9.2	80.59%
Total	588.4	46.7	92.07%

OpenAPI

```
var builder = webApplication.CreateBuilder();
```

```
builder.Services.AddOpenApi();
```

```
var app = builder.Build();
```

```
app.MapOpenApi();
```

```
app.MapGet("/hello/{name}", (string name) => $"Hello {name}!");  
app.Run();
```

OpenAPI

```
localhost:7167/openapi/v1.json
Pretty-print
{
  "openapi": "3.0.1",
  "info": {
    "title": "Net9MinimalAPI | v1",
    "version": "1.0.0"
  },
  "servers": [
    {
      "url": "https://localhost:7167"
    },
    {
      "url": "http://localhost:5006"
    }
  ],
  "paths": {
    "/hello/{name}": {
      "get": {
        "tags": [
          "Net9MinimalAPI"
        ],
        "parameters": [
          {
            "name": "name",
            "in": "path",
            "required": true,
            "schema": {
              "type": "string"
            }
          }
        ],
        "responses": {
          "200": {
            "description": "OK",
            "content": {
              "text/plain": {
                "schema": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  },
  "components": { },
  "tags": [
    {

```

Blazor

-
- Загрузки страниц и времени старта
- Пользовательский опыт
- Интерактивный рендеринг
- Логика восстановления соединения в Blazor Server
- Сжатие сообщений WebSocket
- Скаффолдинг моделей и Identity в Visual Studio
- Производительность и функциональность Hot Reload
- Отладка Blazor WebAssembly



EF CORE

EF CORE

1. AOT

EF CORE

1. AOT

```
> dotnet ef dbcontext optimize
```


EF CORE

1. AOT

```
> dotnet ef dbcontext optimize
```

```
protected override void  
OnConfiguring(DbContextOptionsBuilder options)  
=> options  
    .UseModel(CompiledModels.MyContextModel.Instance);
```

EF CORE

1. AOT

с помощью `Microsoft.EntityFrameworkCore.Tasks`

```
<PropertyGroup>  
  <EFOptimizeContext>true</EFOptimizeContext>  
</PropertyGroup>
```

MyApp.csproj

EF CORE

1. AOT

с помощью `Microsoft.EntityFrameworkCore.Tasks`

```
<PropertyGroup>  
  <EFOptimizeContext>true</EFOptimizeContext>  
  <EFStartupProject>..\App\App.csproj</EFStartupProject>  
</PropertyGroup>
```

MyApp.csproj

EF CORE

1. AOT
2. Перевод LINQ в SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
 - GroupBy по сложным типам

```
var groupByAddress = await context.Stores
    .GroupBy(b => b.Address)
    .Select(g => new { g.Key, Count = g.Count() })
    .ToListAsync();
```

LINQ

```
SELECT Address_City, Address_Line1, Address_Zip, COUNT(*) AS Count
FROM Stores
GROUP BY Address_City, Address_Line1, Address_Zip;
```

SQL

EF CORE

1. AOT

2. Перевод LINQ в SQL

- GroupBy по сложным типам
- Обработка констант

```
var conf = await context.Conferences
    .where(e => e.Title == "DotNext" && e.Year == year)
    .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = @__year_0
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
 - GroupBy по сложным типам
 - Обработка констант

```
var conf = await context.Conferences
    .where(e => e.Title == "DotNext" && e.Year == year)
    .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = @__year_0
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
 - GroupBy по сложным типам
 - Обработка констант

```
var conf = await context.Conferences
    .where(e => e.Title == "DotNext" && e.Year == year)
    .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = @__year_0
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
 - GroupBy по сложным типам
 - Обработка констант

```
var conf = await context.Conferences
    .Where(e => e.Title=="DotNext" && e.Year == EF.Constant(year))
    .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE Title = N'DotNext' AND Year = 2024
```

SQL

EF CORE

1. AOT
2. Перевод LINQ в SQL
 - GroupBy по сложным типам
 - Обработка констант

```
var conf = await context.Conferences
    .Where(e => e.Title==EF.Parameter("DotNext") && e.Year==year)
    .ToListAsync();
```

LINQ

```
SELECT * FROM Conferences
WHERE title = @__p_0 AND Year = @__year_1
```

SQL

EF CORE

1. AOT

2. Перевод LINQ в SQL

- GroupBy по сложным типам
- Обработка констант
- Встраивание некоррелирующих подзапросов

```
var lectures = await context.Lectures
    .where(l => l.Title.Contains(".NET"));
```

```
var results = lectures
    .where(l => l.Year > 2020)
    .select(p => new { Lecture = l, Total = lectures.Count() })
    .ToArray();
```

EF CORE

1. AOT

2. Перевод LINQ в SQL

- GroupBy по сложным типам
- Обработка констант
- Встраивание некоррелирующих подзапросов
- Count > 0 -> EXISTS

```
var blogswithPost = await context.Blogs
    .where(b => b.Posts.Count > 0)
    .ToListAsync();
```

LINQ

```
SELECT * FROM Blogs b
WHERE EXISTS (
    SELECT 1 FROM Posts p WHERE b.Id = p.BlogId);
```

SQL

EF CORE

1. AOT

2. Перевод LINQ в SQL

- GroupBy по сложным типам
- Обработка констант
- Встраивание некоррелирующих подзапросов
- `Count > 0 -> EXISTS`
- Оптимизация `OPENJSON WITH`

EF CORE

1. AOT

2. Перевод LINQ в SQL

- GroupBy по сложным типам
- Обработка констант
- Встраивание некоррелирующих подзапросов
- `Count > 0` -> `EXISTS`
- Оптимизация `OPENJSON WITH`
- `Math.Max / Min` -> `GREATEST / LEAST`

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях

```
await context.Stores
    .ExecuteUpdateAsync(
        s => s.SetProperty(b => b.Address.Line1, newAddr.Line1)
            .SetProperty(b => b.Address.City, newAddr.City)
            .SetProperty(b => b.Address.Zip, newAddr.Zip)); EF8
```

```
await context.Stores
    .ExecuteUpdateAsync(
        s => s.SetProperty(b => b.Address, newAddr)); EF9+
```

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения
5. Кэш последовательностей

EF CORE

1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Прimitives коллекции только для чтения
5. Кэш последовательностей
6. Fill-factor для ключей и индексов

EF CORE

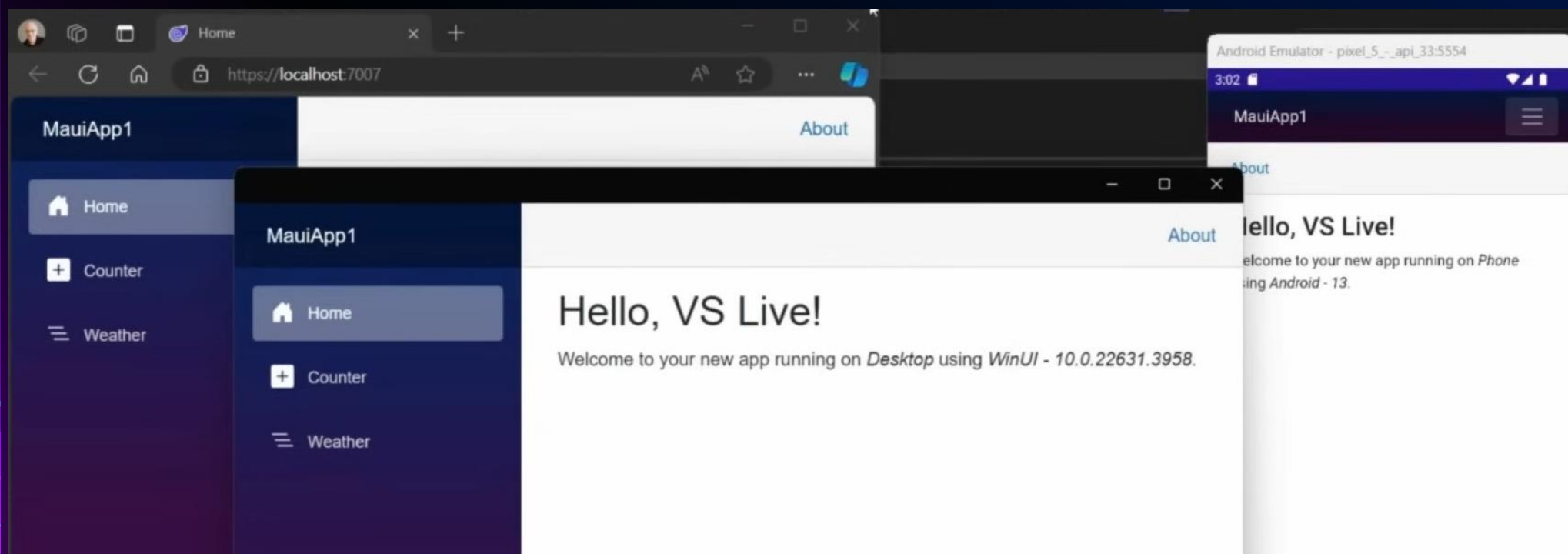
1. AOT
2. Перевод LINQ в SQL
3. Сложные типы в пакетных обновлениях
4. Примитивные коллекции только для чтения
5. Кэш последовательностей
6. Fill-factor для ключей и индексов
7. Azure Cosmos DB для NoSQL

MAUI



MAUI

1. Blazor Hybrid



MAUI

1. Blazor Hybrid
2. Новые компоненты
 - HybridWebView



Картинка: <https://devclass.com/wp-content/uploads/2024/08/hybridwebview-1024x763.jpg>

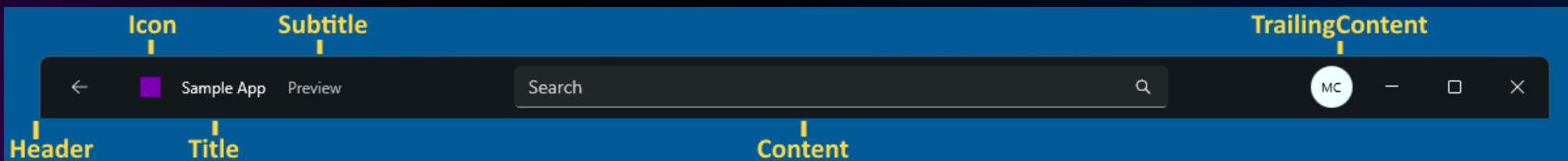


MAUI

1. Blazor Hybrid

2. Новые компоненты

- HybridWebView
- TitleBar



MAUI

1. Blazor Hybrid
2. Новые компоненты
3. Android 15, iOS 17.2 и macOS 14.2



MAUI

1. Blazor Hybrid
2. Новые компоненты
3. Android 15, iOS 17.2 и macOS 14.2
4. Asset packs



```
<ItemGroup>  
  <AndroidAsset Update="Assets/MyLargeAsset.mp4"  
    AssetPack="myassets" />  
</ItemGroup>
```

MyApp.csproj

WINFORMS



WINFORMS

1. Поддержка тёмной темы



WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов



WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов
3. Поддержка асинхронности в UI



WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов
3. Поддержка асинхронности в UI
4. Поддержка AI компонентов



WINFORMS

1. Поддержка тёмной темы
2. Отрисовка компонентов и шрифтов
3. Поддержка асинхронности в UI
4. Поддержка AI компонентов
5. MVVM для байндинга данных с помощью `CommunityToolkit.Mvvm`



СПАСИБО

Сергей Бензенко

📩 @SBenzenko

📩 Телеграм канал «.NET Разработчик»

[@NetDeveloperDiary](https://t.me/NetDeveloperDiary)

