

Изменения в С# 12

Хто я?

Федотов Евгений

Младший бэкенд разработчик в компании
“Монополия”



Содержание выступления

- Чуть чуть про primary конструкторы
- Упрощение определения типов
- Collection expression и spread operator
- ref readonly параметры
- Intercept Location
- Дефолтные значения для лямбда-выражений
- Псевдонимы для любых типов
- Доработка nameof
- Атрибут Experimental
- Inline массивы

Primary конструкторы

Как было

```
public class Human
{
    public Human(string name, string surname, bool isDead)
    {
        Name = name;
        Surname = surname;
        IsDead = isDead;
    }
    public string Name { get; }

    public string Surname { get; }

    public bool IsDead { get; }
}
```

Как стало

```
var human = new Human("Ryan", "Gosling", false);
Console.WriteLine($"Имя: {human.Name} " +
                  $"Фамилия: {human.Surname} " +
                   $"Умер в конце драйва: {human.IsDead}";);

public class Human(string name, string surname, bool isDead)
{
    public string Name { get; } = name;

    public string Surname { get; } = surname;

    public bool IsDead { get; } = isDead;
}
```

**Результат
выполнения:**

Имя: Ryan Фамилия: Gosling Умер в конце драйва: False

Упрощение определения типов

Как было

```
public interface IEmptyInterface {}
```

```
public class EmptyClass {}
```

```
public struct EmptyStruct {}
```


Как стало

```
public interface IEmptyInterface;  
public class EmptyClass;  
public struct EmptyStruct;
```

Collection expression и spread operator

Обычные и двумерные массивы/списки

```
int[] row0 = [1, 2, 3];  
int[] row1 = [4, 5, 6];  
int[] row2 = [7, 8, 9];  
int[][] twoDArrayFromVariables = [row0, row1, row2];  
  
int[][] twoDArray = [[1,2,3], [1,4,6], [3,6,3]];  
  
List<string> truth = ["Да", "не", "умер", "он", "в", "конце", "Драйва"];
```

Spread оператор

```
List<int> numbers = [1, 2, 3, 4, 5];
```

```
int[] numbers2 = [6, 7, 8, 9, 10];
```

```
List<int> numbersMegazord = [.. numbers, .. numbers2];
```

```
foreach (var numberRanger in numbersMegazord)
```

```
{  
    //ВЫВОД: 1 2 3 4 5 6 7 8 9 10  
    Console.Write($"{numberRanger} ");  
}
```

А что там с самописными коллекциями?

Collection expression в своей коллекции

```
ExampleList example = ["Pineapple", "Watermelon", "Egg"];  
[CollectionBuilder(typeof(ExampleList), nameof(ExampleList.Create))]  
public class ExampleList : IEnumerable<string>  
{  
    private string[] items = new string[50];  
  
    public IEnumerator<string> GetEnumerator()  
        => items.AsEnumerable().GetEnumerator();  
  
    IEnumerator IEnumerable.GetEnumerator()  
        => items.GetEnumerator();  
  
    public static ExampleList Create(ReadOnlySpan<string> source)  
        => new ExampleList(source);  
  
    private ExampleList(ReadOnlySpan<string> source) => source.CopyTo(items);  
}
```

Атрибут Experimental



Как было?

```
var veryDangerous = new VeryDangerous ();  
//CS0618: Method 'VeryDangerous.DoSomeScaryStuff()'   
//is obsolete: Не лезь, оно тебя сожрёт  
veryDangerous.DoSomeScaryStuff ();
```

```
class VeryDangerous  
{  
    [Obsolete("Не лезь, оно тебя сожрёт")]  
    public void DoSomeScaryStuff ()  
    {  
        Console.WriteLine("Very scary");  
    }  
}
```


Как стало

```
using System.Diagnostics.CodeAnalysis;

var veryDangerous = new VeryDangerous();
veryDangerous.DoSomeScaryStuff();

class VeryDangerous
{
    [Experimental("Amogus")]
    public void DoSomeScaryStuff()
    {
        Console.WriteLine("Very scary");
    }
}
```

Что происходит при запуске

0>[Program.cs\(4,1\)](#): Error Amogus : "VeryDangerous.DoSomeScaryStuff()" предназначен только для оценки и может быть изменен или удален в будущих обновлениях. Чтобы продолжить, скройте эту диагностику.

0>----- Finished building project: ExperimentalAttribute. Succeeded: False. Errors: 1. Warnings: 0

Ref readonly



Что такое lvalue и rvalue

`int e = 5;`

The diagram illustrates the concepts of lvalue and rvalue using the code snippet `int e = 5;`. A blue arrow points from the label `lvalue` to the variable `e`. Another blue arrow points from the label `rvalue` to the constant `5`.

```
Test.InVoid(5);  
//Warning: Argument 1 should be a variable  
//because it is passed to a 'ref readonly' parameter  
Test.ReadOnlyRefVoid(5);  
//Error: 'ref' argument must be an assignable  
//variable, field, or an array element  
Test.ReadOnlyRefVoid(ref 5);  
  
int e = 5;  
Test.ReadOnlyRefVoid(ref e);  
Test.InVoid(e);  
//Warning: Argument 1 should be passed with 'ref' or 'in' keyword  
Test.ReadOnlyRefVoid(e);  
  
public static class Test  
{  
    public static void ReadOnlyRefVoid(ref readonly int test) { }  
  
    public static void InVoid(in int test) { }  
}
```

Перехватчики кода

Пример

```
var talkingEntity = new TalkingEntity();  
  
talkingEntity.Talk("бее");  
  
talkingEntity.Talk("гав гав");  
  
talkingEntity.Talk("во славу императора");  
  
class TalkingEntity  
{  
    public void Talk(string phrase)  
        => Console.WriteLine($"Барашек говорит: {phrase}");  
}
```

Результат выполнения

```
тестскрипты\bin\debug\netso.0\inetsores\тестскрипты.exe
```

```
Барашек говорит: бее
```

```
Собачка говорит: гав гав
```

```
Боевой перехватчик космодесанта говорит: во славу императора
```



Сгенерированный код

```
namespace InterceptorTestApp.Generated;

static class Generated
{
    [System.Runtime.CompilerServices.InterceptsLocation("Program.cs", line: 3, character: 15)]
    public static void Intercept1(this TalkingEntity talkingEntity, string phrase)
        => Console.WriteLine($"Собачка говорит: {phrase}");

    [System.Runtime.CompilerServices.InterceptsLocation("Program.cs", line: 4, character: 15)]
    public static void Intercept2(this TalkingEntity talkingEntity, string phrase)
        => Console.WriteLine($"Боевой перехватчик космодесанта говорит: {phrase}");
}
```

Пример Minimal API

```
todosApi.MapGet(pattern: "/", handler: () => sampleTodos);  
todosApi.MapGet(pattern:("/{id}", handler: (int id) =>  
    sampleTodos.FirstOrDefault(a: Todo => a.Id == id) is { } todo  
        ? Results.Ok(todo)  
        : Results.NotFound());
```

Перехватчик

```
[InterceptsLocation("NativeAotTest\Program.cs", 22, 10)]  
  
internal static RouteHandlerBuilder MapGet0(  
    this IEndpointRouteBuilder endpoints,  
    [StringSyntax("Route")] string pattern,  
    Delegate handler)  
{  
    MetadataPopulator populateMetadata = (methodInfo, options) =>  
    {  
        Debug.Assert(options != null, "RequestDelegateFactoryOptions not found.");  
    }  
}
```

Как включить их в своём проекте?

```
<PropertyGroup>  
  
<InterceptorsPreviewNamespaces> $(InterceptorsPreviewNamespaces);$(RootNamespace).Generated </InterceptorsPreviewNamespaces>  
</PropertyGroup>
```

```
namespace System.Runtime.CompilerServices;
```

```
[AttributeUsage(AttributeTargets.Method, AllowMultiple = true, Inherited = false)]  
public sealed class InterceptsLocationAttribute : Attribute  
{  
    public InterceptsLocationAttribute(string filePath, int line, int character)  
    {  
    }  
}
```

Дефолтные значения для лямбда выражений

```
var IsDividableBy = (int source, int divider = 2) =>  
    source % divider == 0;
```

```
Console.WriteLine(IsDividableBy(4));  
Console.WriteLine(IsDividableBy(4, 3));
```

Результат
выполнения:

```
True  
False  
  
Process finished with exit code 0.
```

Псевдонимы для любых типов

```
using Vector3 = (int X, int Y, int Z);
```

```
var vector = new Vector3();
```

```
vector.X = 30;
```

```
vector.Y = 50;
```

```
vector.Z = 20;
```

```
Console.WriteLine($"{vector.X} {vector.Y} {vector.Z}");
```

Доработка nameof

```
() using System.ComponentModel;

[1 usage] [1 exposing API]
public class ClassA
{
    [1 usage]
    public int Hehe { get; set; }
}

public class ClassB
{
    [Description($"ClassA part is {nameof(ClassA.Hehe)}")]
    [1 usage]
    public ClassA ClassA { get; set; }
}
```

Cannot access non-static property 'ClassA' in static context

[Description("ClassA part is Hehe")]
public ClassA ClassA { get; set; }
in class ClassB

<description>ClassA part is Hehe</description>

C# 11



```
using System.ComponentModel;

[1 usage] [1 exposing API]
public class ClassA
{
    [1 usage]
    public int Hehe { get; set; }
}

public class ClassB
{
    [Description($"ClassA part is {nameof(ClassA.Hehe)}")]
    [1 usage]
    public ClassA ClassA { get; set; }
}
```

C# 12

Inline массивы

```
using System.Runtime.CompilerServices;
```

```
var array = new InlineArray();  
for (int i = 0; i < 10; i++)  
{  
    array[i] = i;  
}
```

```
foreach (var item in array)  
{  
    Console.Write($"{item},");  
}
```

```
[InlineArray(10)]  
public struct InlineArray  
{  
    private int arrayValueType;  
}
```

Результат выполнения

```
bin/Debug/net8.0/InlineArrays.exe  
0,1,2,3,4,5,6,7,8,9,  
Process finished with exit code 0.
```


Спасибо за внимание

Соцсети:

VK: <https://vk.com/sp1ceforce>

LinkedIn: <https://www.linkedin.com/in/sp1ceforce/>

Telegram: @sp1ceforce

GitHub: <https://github.com/Sp1ceForce>

Код и ссылки на полезные статьи:

<https://github.com/Sp1ceForce/CSharp12FeaturesProject>

