NOVOSTI V C# 14

Damir Arh, Razum d.o.o.

Microsoft MVP

O MENI

- Razum d.o.o.
- Microsoft MVP
- https://damirscorner.com
- @DamirArh@mas.to
- @damirscorner.com
- @DamirArh

PREGLED NOVOSTI

- Razširitveni člani
- Preobloženi sestavljeni operatorji prirejanja
- Prirejanje s preverjanjem vrednosti `null`
- Ključna beseda `field` v lastnostih
- Delni dogodki in konstruktorji
- Imena nevezanih generičnih tipov
- Implicitne pretvorbe v `Span<T>`
- Izboljšave lambda izrazov
- Programi v eni datoteki

TRENUTNA RAZPOLOŽLJIVOST

- Popolna podpora:
  - Visual Studio 2026 Insiders
  - .NET 10 Release Candidate 1
- Delna podpora:
  - Visual Studio 2022
  - .NET 9
  - Vklop predogledne različice C# v projektu

```
<PropertyGroup>
   <LangVersion>preview</LangVersion>
</PropertyGroup>
```

RAZŠIRITVENE METODE PRED C# 14

- Navidezno dodajanje instančnih metod obstoječim tipom

```
public static class StringExtensions
{
  public static string? FirstCharToUpper(this string? input)
  {
    if (string.IsNullOrEmpty(input))
      return input;
    else
      return string.Concat(input[..1].ToUpper(), input[1..]);
  }
}

var foo = StringExtensions.FirstCharToUpper("foo");
var bar = "bar".FirstCharToUpper();
```

RAZŠIRITVENI ČLANI V C# 14

- Navidezno dodajanje instančnih in statičnih članov obstoječim tipom

```
public static class StringExtensions
{
  extension(string? receiver)
  {
    public string? FirstCharToUpper() { /*...*/ }
    public bool IsEmptyField { get { /*...*/ } }
    public static string? Create(string? pattern, int count)
    { /*...*/ }
    public static string? Null { get { /*...*/ } }
    public static string? operator *(string? pattern, int coun
    { /*...*/ }
  }
}
```

RAZŠIRITVENI ČLANI KOT STATIČNE METODE

- Klicanje razširitvenih članov prek pripadajočih statičnih metod

```
var result1 = StringExtensions.FirstCharToUpper(input);
var result2 = StringExtensions.Create(pattern, count);
var result3 = StringExtensions.get_IsEmptyField(input);
var result4 = StringExtensions.get_Null();
var result5 = StringExtensions.op_Multiply(pattern, count);
```

GENERIČNI RAZŠIRITVENI ČLANI

- Tipi razširitvenih članov so lahko generični

```
public static class EnumerableExtensions
{
  extension<TItem>(IEnumerable<TItem> receiver)
  {
    public IEnumerable<TItem> WhereWithinRangeBy<TValue>(
      Func<TItem, TValue> projection, TValue min, TValue max
    ) where TValue : INumber<TValue>
    {
      return receiver.Where(item =>
        projection(item) >= min && projection(item) <= max
      );
    }
  }
}
```

RAZŠIRITVENI ČLANI

Demo

SESTAVLJENI OPERATORJI PRIREJANJA

- Kombinacija operatorja prirejanja z drugim operatorjem

```
backpack += item;
```

- Le krajši zapis za prirejanje rezultata levemu operandu

```
backpack = backpack + item;
```

PREOBLAGANJE BINARNEGA OPERATORJA

- Na voljo že pred C# 14

```
public static Backpack operator +(Backpack oldBackpack, Item i
{
  var newBackpack = oldBackpack.Clone();
  newBackpack.Add(item);
  return newBackpack;
}
```

- Uporabi se tudi za sestavljeni operator

PREOBLAGANJE SESTAVLJENEGA OPERATORJA

- Na voljo v C# 14

```
public void operator +=(Item item)
{
  Add(item);
}
```

- Tudi kot razširitveni operator

```
public static class BackpackExtensions
{
  extension (Backpack receiver)
  {
    public void operator +=(Item item) => receiver.Add(item);
  }
}
```

PREOBLOŽENI SESTAVLJENI OPERATORJI PRIREJANJA

Demo

POGOJNI OPERATORJI PRED C# 14

- Preprečijo izjemo pri dostopanju do članov vrednosti `null`

```
Report? report = null;
var items = report?.ItemsProcessed;

List<int>? list = null;
var item = list?[0];
```

POGOJNO PRIREJANJE V C# 14

- Prepreči izjemo pri prirejanju članom vrednosti `null`

```
Report? report = null;
report?.ItemsProcessed = processor.Process();

List<int>? list = null;
list?[0] = processor.Process();
```

PRIREJANJE S PREVERJANJEM VREDNOSTI `null`

Demo

SAMODEJNE LASTNOSTI PRED C# 14

- Ni potrebno definirati polja za hrambo podatka

```
public int AutoImplementedProperty { get; set; }
```

- Ni mogoče dodati lastne kode za dostopni metodi

KLJUČNA BESEDA `field` V C# 14

- omogoča dostop do samodejno kreiranega polja za hrambo

```
public int ValidatedProperty
{
  get;
  set
  {
    field =
      value >= 0
        ? value
        : throw new ArgumentOutOfRangeException(
          nameof(value),
          "The value must be non-negative."
        );
  }
}
```

KLJUČNA BESEDA `field` V LASTNOSTIH

Demo

PODPORA DELNIM ČLANOM

```
public partial class Partial
{
  // C# 1
  partial void PrivateVoidMethod();
  // C# 9
  protected partial bool AnyMethod(double input, out int outpu
  // C# 13
  private static partial string AnyProperty { get; private set
  private partial string this[int i] { get; }
  // C# 14
  private partial event EventHandler<EventArgs> AnyEvent;
  private partial Partial(bool flag);
}
```

DELNI DOGODKI IN KONSTRUKTORJI

Demo

IMENA GENERIČNIH TIPOV

- Pred C# 14 je bilo nujno podati generične parametre

```
var name = nameof(List<int>);
```

  - S C# 14 generični parametri niso več potrebni

```
var name = nameof(List<>);
```

IMENA NEVEZANIH GENERIČNIH TIPOV

Demo

IZBOLJŠANO PREPOZNAVANJE PRETVORB V Span<T>

- Implicitno pretvarjanje polj in nizov v dodatnih scenarijih

```
// C# 13
var intLength = ((ReadOnlySpan<int>)array).DetermineLength();
var charLength = ((ReadOnlySpan<char>)str).DetermineLength();
// C# 14
var intLength = array.DetermineLength();
var charLength = str.DetermineLength<char>();
```

IMPLICITNE PRETVORBE V Span<T>

Demo

UPORABA MODIFIKATORJEV V LAMBDA IZRAZIH

- Pred C# 14 je bila dovoljena le ob navedbi tipov parametrov

```
MyDelegate handler = (int input, out int output) => output = i
```

  - V C# 14 je dovoljena tudi brez tipov parametrov

```
MyDelegate handler = (input, out output) => output = input;
```

NEOBVEZNI IN POIMENOVANI PARAMETRI V DREVESIH IZRAZOV

```
private int Add(int x, int y = 0) => x + y;
Expression<Func<int, int, int>> expression;
// pred C# 14
expression = (x, y) => Add(x, y);
// v C# 14
expression = (x, y) => Add(x) + y;
expression = (x, y) => Add(x, y: y);
// niti v C# 14
expression = (x, y) => Add(y: y, x: x);
```

IZBOLJŠAVE LAMBDA IZRAZOV

Demo

KODA NA GORNJEM NIVOJU

- Vstopna datoteka brez razreda in metode `Main`

```
using Humanizer;

var dotNet10ReleaseDate = DateTimeOffset.Parse("2025-11-11");
var left = DateTimeOffset.Now - dotNet10ReleaseDate;

Console.WriteLine($"{left.Humanize()} until .NET 10 release.")
```

PROGRAM BREZ PROJEKTNE DATOTEKE

- Zgolj datoteka `.cs`

```
#!/usr/bin/dotnet run
#:package Humanizer@2.14.1

using Humanizer;

var dotNet10ReleaseDate = DateTimeOffset.Parse("2025-11-11");
var left = DateTimeOffset.Now - dotNet10ReleaseDate;

Console.WriteLine($"{left.Humanize()} until .NET 10 release.")
```

- Vsebina projektne datoteke v direktivah

PROGRAMI V ENI DATOTEKI

Demo

VIRI

- damirscorner.com/link/
  - Cs14GitHub
  - Cs14Blogposts
  - Cs14Docs
  - Cs14Status