BASTA!

by entwickler.de

Be ready for C# 12

Christian Nagel

https://www.cninnovation.com

Christian Nagel

- Training
- Coaching
- Consulting
- Development

- Microsoft MVP
- www.cninnovation.com
- csharp.christiannagel.com
- @christiannagel

What's new with...

- Types
- Arrays and Collections
- Lambda Expressions
- Something special...

Writing code today…

Sample: minimal APIs with scaffolding…

Types and members Enhancements

Alias any type

- using alias relaxed with C# 12
- alias tuples, pointers, arrays, generic types…

Primary Constructors

- Class records
  - get & init accessors
- Struct records
  - get & set accessors
- Readonly struct records
  - get & init accessors
- Classes and structs
  - Parameters

Parameter ref readonly

- ref
  - Needs initialization before calling the method
- out
  - Initialization not required
  - Method must assign a value
- ref readonly
  - Must be initialized
  - Method cannot assign a new value
- in
  - Must be initialized
  - Method cannot assign a new value
  - Compiler can use a temporary variable within the method

Arrays and Collections

Inline Arrays

- Optimized creation for fixed sizes
- Directly assign *Span<T>*
- *InlineArray* attribute
- Performance optimization

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

Collection Expressions (Collection Literals)

- Conversion to many different collection types using square brackets *[]*

```
int[] arr = [1, 2];
List<int> list1 = [3, 4];
IEnumerable<int> list2 = [5, 6];
```

Spread Operator

- Expand elements without manual iteration
- Can be used together with the range operator

```
int[] arr = [1, 2];
List<int> list1 = [3, 4];
IEnumerable<int> list2 = [.. arr, .. list1];
```

CollectionBuilder Attribute

- Allow collection expressions with custom collection types

```
[CollectionBuilder(typeof(MyCustomCollection),
  nameof(MyCustomCollection.Create))]
internal class MyCustomCollection<T> : Collection<T>
{
}
```

```
internal static class MyCustomCollection
{
  public static MyCustomCollection<T> Create<T>(ReadOnlySpan<T> items)
  {
    MyCustomCollection<T> collection = new();
    foreach (T item in items)
    {
      collection.Add(item);
    }
    return collection;
  }
}
```

Lambda Expressions

Natural delegate type (C# 10)

- Natural type of lambda expression
- Doesn’t require to declare a delegate type (e.g. Func<>)

Default lambda parameters (C# 12)

- Default values for parameters on lambda expressions
- Convenient with minimal APIs

Something special…

Unsafe Accessor

- With reflection it is possible to access private members of a type
- *UnsafeAccessor* doesn't need reflection!
- Serialization, EF Core…

```
internal class ChangeIt
{
  [UnsafeAccessor(UnsafeAccessorKind.Field,
    Name = "_title")]
  public extern static ref string GetTitle(Book @this);
}
```

- Compiler-Feature
- Access private members

Interceptors

- Replace implementation
- Usually used by source generators
- Pre-release with .NET 8
- Used from source generators
- *InterceptsLocation* Attribute

Native AOT

- Compile .NET to native code
- Self-contained
- Quick startup, less memory usage
- Can run where JIT is not allowed
- Compilation to a single file

Native AOT Restrictions

- No dynamic loading
- No reflection emit
- No C++/CLI
- Trimming required
- Many libraries don’t support native AOT (yet)

Native AOT
For Action

- Make libraries AOT compatible
  - if possible
  - *IsAotCompatible* adds checks
- Create native AOT services
  - if useful and possible

C# v.next

- Semi-auto properties
- Params collections
- Default in deconstructions
- Roles / extensions
- Escape character
- Method group natural type improvements
- Lock statement pattern

Semi-Auto Properties

- field keyword

```
// full property
private int _x;
public int X
{
  get => _x;
  set => _x = value;
}
```

```
// auto property
public int X { get; set; }
```

```
// semi-auto property
public int X
{
  get => field;
  set => field = value;
}
```

```
public MyType X => field ??= ComputeValue();
```

Roles / extensions

- Extensions are “transparent wrappers”
- Augmented with additional members and interfaces

```
public extension E for U
{
  public static U Create() { … }
  public static U operator+(U e1, U e2) { … }
  public int M() { }
  public string this[int i] { }
}
```

Lock Object

- .NET 9 includes *System.Threading.Lock* type
- The lock keyword will be enhanced to not only support *Monitor*, but also *Lock*

Summary

- Productivity
  - Primary constructors
  - Collection expressions
- Performance
  - Inline Array
  - Native AOT
  - Source generators

Thank you for joining!

Questions?

- https://github.com/cnilearn/bastaspring2024
- https://csharp.christiannagel.com
- https://www.cninnovation.com