
NEW HORIZONS FOR
C# DEVELOPERS

Who is Chad Green?

chadgreen@chadgreen.com

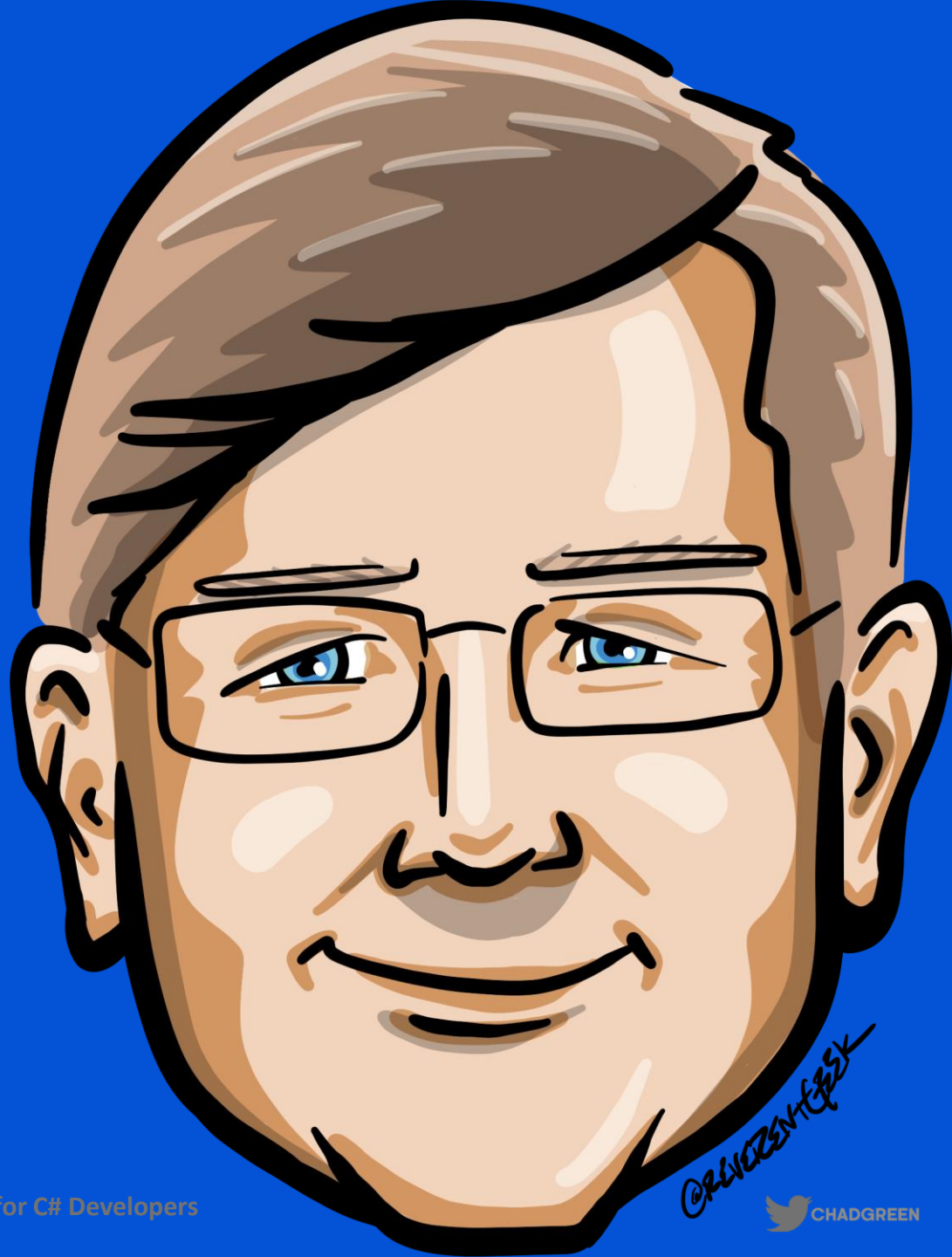
TaleLearnCode

ChadGreen.com

ChadGreen & TaleLearnCode

ChadwickEGreen

Microsoft® MVP Most Valuable Professional



Agenda

- C# 12: Pushing the Boundaries
- .NET 8: Unleashing Performance and Scalability
- ASP.NET Core 8: Crafting Modern Web Experiences
- Entity Framework Core 8: Redefining Data Access


Pushing the Boundaries
What's new in C# 12
TALELEARNCODE
Comprehensive Exploration of New Horizons for C# Developers
CHADGREEN

Pushing the Boundaries

- Primary constructors
- Collection expressions
- Inline arrays
- Optional parameters in lambda expressions
- ref readonly parameters
- Alias any type
- Experimental attribute
- Interceptors (preview)

Primary Constructors

- Add parameters to a struct or class declaration
- Scoped throughout class definition

```
public abstract class BankAccount0(string accountId, string owner)
{

    1 reference | 0 changes | 0 authors, 0 changes
    public string AccountId { get; } = accountId;
    1 reference | 0 changes | 0 authors, 0 changes
    public string Owner { get; } = owner;

    0 references | 0 changes | 0 authors, 0 changes
    public override string ToString() => $"Account Id: {AccountId}, Owner: {Owner}";

}
```

Primary Constructors

- Important to view primary constructor parameters as **parameters**
- Primary constructor rules:
  - May not be stored if they are not needed
  - Are not members of the class
  - Can be assigned to
  - Do not become properties (except in *record* types)

- Every other constructor for a class **must** call the primary constructor

Primary Constructors

- Most common uses for primary constructors:
  - As an argument to a base() constructor invocation
  - To initialize a member field or property
  - Referencing the constructor parameter in an instance member

Collection Expressions

- Use *collection expressions* to
  - Specify initial value for a collection
  - Pass as arguments to methods that take collection types
- Terse syntax that can be assigned to many different collection types

```
int[] a = [1, 2, 3, 4, 5, 6, 7, 8];
```

ref readonly Parameters

- C# added `in` parameters as a way to pass readonly references
- `in` parameters allow both variables and values
- `in` parameters can be used without any annotation on arguments

- `ref readonly` parameters enables more clarify for APIs that might be using `ref` parameters or `in` parameters

Default Lambda Parameters

- Can provide *default values* for parameters on lambda expressions
- Syntax same as for methods and local functions

Alias Any Type

- Can use the `using` alias directive to alias any type, not just named types

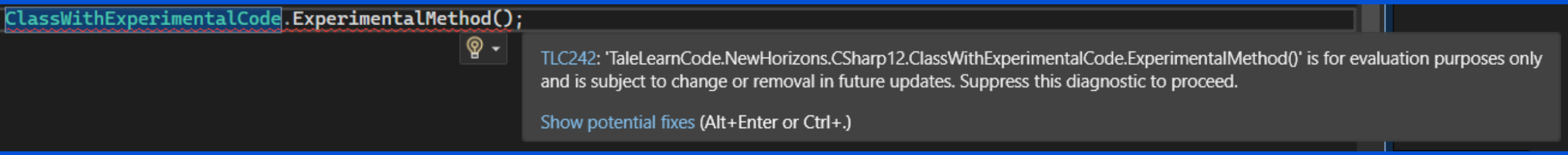
```
using Point = (int x, int y);
```

Inline Arrays

- Used by the **runtime team** to improve performance
- Enable a developer to create an array of fixed size in a `struct` type
- You likely will not declare your own inline arrays

Experimental Attribute

- Types, methods, and assemblies can be marked as Experimental
- Compiler issues warning if you access experimental logic
- You can disable these warnings to pilot experimental feature(s)



Interceptors

- Method that can declaratively substitute a call to an *interceptable* method with a call to itself at compile time


Unleashing Performance and Scalability
What's New in .NET 8
TALELEARNCODE
Comprehensive Exploration of New Horizons for C# Developers
CHADGREEN

Unleashing Performance and Scalability

- .NET Aspire
- Core .NET libraries
- Extension Libraries
- Garbage collection
- Configuration-binding source generator
- Reflection improvements
- Native AOT support
- Performance improvements
- .NET MAUI
- .NET SDK+
- Globalization
- Containers
- Source-generated COM interop
- .NET on Linux
- Cross-built Windows Apps
- AOT compilation for Android apps
- Code Analysis
- Windows Presentation Foundation
- NuGet
- Diagnotics

Unleashing Performance and Scalability

- .NET Aspire
- Core .NET libraries
- Extension Libraries
- Garbage collection
- Configuration-binding source generator
- Reflection improvements
- Native AOT support
- Performance improvements
- .NET MAUI
- .NET SDK
- Globalization
- Containers
- Source-generated COM interop
- .NET on Linux
- Cross-built Windows Apps
- AOT compilation for Android apps
- Code Analysis
- Windows Presentation Foundation
- NuGet
- Diagnotics

.NET Aspire

- Opinionated, cloud-ready stack for building observable, production-ready, distributed applications
- Helps with
  - Orchestration
  - Components
  - Tooling

Core .NET Libraries

- Serialization
- Time abstraction
- UTF8 improvements
- Methods for working with randomness
- Performance-focused types
- System.Numerics and System.Runtime.Intrinsics
- Data validation
- Metrics
- Cryptography
- Networking
- Stream-based ZipFile methods

Serialization

- Handling missing members during deserialization
- Built-in support for additional types
- Source generator
- Chain source generators
- Interface hierarchies
- Naming policies
- Read-only properties
- Disable reflection-based default
- New JsonNode API methods
- Non-public members
- Streaming deserialization APIs
- WithAddedModified Extension Method
- New JsonContent.Create overloads
- Freeze a JsonSerializerOptions instance

Serialization

- **Handling missing members during deserialization**
- **Built-in support for additional types**
- Source generator
- Chain source generators
- **Interface hierarchies**
- **Naming policies**
- **Read-only properties**
- Disable reflection-based default
- New JsonNode API methods
- **Non-public members**
- **Streaming deserialization APIs**
- WithAddedModified Extension Method
- New JsonContent.Create overloads
- Freeze a JsonSerializerOptions instance

Serialization

- Handling missing members during deserialization

Serialization

- Built-in support for additional types

Serialization

- Interface hierarchies

Serialization

- Naming policies

Serialization

- Read-only properties

Serialization

- Non-public members

Serialization

- Streaming deserialization APIs

Time Abstraction

- TimeProvider class and ITimer interface
- Essential time operations:
  - Retrieve local and UTC time
  - Obtain a timestamp for measuring performance
  - Create a timer

```
// Get system time.
DateTimeOffset utcNow = TimeProvider.System.GetUtcNow();
DateTimeOffset localNow = TimeProvider.System.GetLocalNow();

// Create a time provider that works with a
// time zone that's different than the local time zone.
private class ZonedTimeProvider : TimeProvider
{
    private TimeZoneInfo _zoneInfo;

    public ZonedTimeProvider(TimeZoneInfo zoneInfo) : base()
    {
        _zoneInfo = zoneInfo ?? TimeZoneInfo.Local;
    }

    public override TimeZoneInfo LocalTimeZone => _zoneInfo;

    public static TimeProvider FromLocalTimeZone(TimeZoneInfo zoneInfo) =>
        new ZonedTimeProvider(zoneInfo);
}

// Create a timer using a time provider.
ITimer timer = timeProvider.CreateTimer(
    callBack, state, delay, Timeout.InfiniteTimeSpan);

// Measure a period using the system time provider.
long providerTimestamp1 = TimeProvider.System.GetTimestamp();
long providerTimestamp2 = TimeProvider.System.GetTimestamp();

var period = GetElapsedTime(providerTimestamp1, providerTimestamp2);
```

Random Methods

- GetItems<T>()
- Shuffle<T>()

Performance-Focused Types

- System.Collections.Frozen

Performance-Focused Types

- System.Collections.Frozen
- System.Buffers.SearchValues<T>

Performance-Focused Types

- System.Collections.Frozen
- System.Buffers.SearchValues<T>
- System.Text.CompositeFormat

```
private static readonly CompositeFormat s_rangeMessage =
    CompositeFormat.Parse(LoadRangeMessageResource());

// ...
static string GetMessage(int min, int max) =>
    string.Format(CultureInfo.InvariantCulture, s_rangeMessage, min, max);
```

Performance-Focused Types

- System.Collections.Frozen
- System.Buffers.SearchValues<T>
- System.Text.CompositeFormat
- System.IO.Hashing.XxHash3
- System.IO.Hashing.XxHash128

Data Validation

New API	Description
MinimumIsExclusive MaximumIsExclusive	Specifies whether bounds are included in the allowable range.
Length	Specifies both lower and upper bounds for string or collections. For example `[Length(10, 20)]` requires at least 10 elements and at most 20 elements in a collection.
Base64String	Validates that a string is a valid Base64 representation.
AllowedValues DeniedValues	Specify allow lists and deny lists. For example, `[AllowedValues("apple", "banana", "mango")]`.

Support for HTTPS proxy

- To enable HTTPS proxy, set the all_proxy environment variable
- Or use the WebProxy class to control the proxy programmatically

Stream-based ZipFile methods

- CreateFromDirectory
- ExtractToDirectory

Native AOT Support

- Support for x64 and Arm64 on macOS
- Reduced Native AOT app size on Linux
- Specify optimization of size or speed

Operating System	.NET 7	.NET 8
Linux x64	3.76-Mb	1.84-Mb
Windows x64	2.85-Mb	1.77-Mb

Native AOT Support

- Support for x64 and Arm64 on macOS
- Reduced Native AOT app size on Linux
- Specify optimization of size or speed
- Console App template

```
dotnet new console --aot
```

Native AOT Support

- Support for x64 and Arm64 on macOS
- Reduced Native AOT app size on Linux
- Specify optimization of size or speed
- Console App template
- Target iOS-like platforms

Performance Improvements

- Arm64 performance improvements
- SIMD improvements
- Support for AVX-512 ISA extensions
- Cloud-native improvements
- JIT throughput improvements
- Loop and general optimizations
- Optimized access for fields marked with ThreadStaticAttribute
- Consecutive register allocation
- JIT/NativeAOT can now unroll and auto-vectorize some memory operations

.NET SDK Enhancements

- CLI-based project evaluation

.NET SDK Enhancements

- CLI-based project evaluation
- Terminal build output

.NET SDK Enhancements

- CLI-based project evaluation
- Terminal build output
- Simplified output paths

.NET SDK Enhancements

- CLI-based project evaluation
- Terminal build output
- Simplified output paths
- `dotnet workload clean` command

.NET SDK Enhancements

- CLI-based project evaluation
- Terminal build output
- Simplified output paths
- `dotnet workload clean` command
- `dotnet publish` and `dotnet pack` assets

.NET SDK Enhancements

- CLI-based project evaluation
- Terminal build output
- Simplified output paths
- `dotnet workload clean` command
- `dotnet publish` and `dotnet pack` assets
- `dotnet restore` security auditing

.NET SDK Enhancements

- CLI-based project evaluation
- Terminal build output
- Simplified output paths
- `dotnet workload clean` command
- `dotnet publish` and `dotnet pack` assets
- `dotnet restore` security auditing
- More secure template engine experience

Container Images

- Generated-image defaults
- Debian 12
- Non-root user
- Chiseled Ubuntu images
- Build multi-platform container images
- ASP.NET composite imagees

Container Publishing

- Performance and compatibility
- Authentication
- Publish to tar.gz archive

Code Analysis

Rule Id	Category	Description
CA1856	Performance	Fires when the ConstantExpected attribute is not applied correctly on a parameter.
CA1857	Performance	Fires when a parameter is annotated with ConstantExpected attribute but the provided argument is not a constant.
CA1858	Performance	Recommends upgrading the type of specific local variables, fields, properties, methods, parameters, and method return types from interface or abstract types to concrete types when possible. Using concrete types leads to higher quality generated code.
CA1860	Performance	To determine with a collection type has any elements, it is better to use Length, Count, or IsEmpty, than to call Any.
CA1861	Performance	Constant arrays passed as arguments are not reused when called repeatedly, which implies a new array is created each time. To improve performance, consider extracting the array to a static readonly field.
CA1865 -1867	Reliability	Enumerable.Cast<TResult>(IEnumerable) and Enumerable.OfType<TResult>(IEnumerable) require compatible types to function correctly. Widening and user-defined conversions aren't supported with generic types.
CA1510 -1513	Maintainability	Throw helpers are simpler and more efficient than an if block constructing a new exception instance. These were created for ArgumentNullException, ArgumentException, ArgumentOutOfRangeException, and ObjectDisposedException.

Code Analysis

Rule Id	Category	Description
CA1856	Performance	Fires when the ConstantExpected attribute is not applied correctly on a parameter.
CA1857	Performance	Fires when a parameter is annotated with ConstantExpected attribute but the provided argument is not a constant.
CA1858	Performance	Recommends upgrading the type of specific local variables, fields, properties, methods, parameters, and method return types from interface or abstract types to concrete types when possible. Using concrete types leads to higher quality generated code.
CA1860	**Performance**	**To determine with a collection type has any elements, it is better to use Length, Count, or IsEmpty, than to call Any.**
CA1861	Performance	Constant arrays passed as arguments are not reused when called repeatedly, which implies a new array is created each time. To improve performance, consider extracting the array to a static readonly field.
CA1865 -1867	Reliability	Enumerable.Cast<TResult>(IEnumerable) and Enumerable.OfType<TResult>(IEnumerable) require compatible types to function correctly. Widening and user-defined conversions aren't supported with generic types.
CA1510 -1513	Maintainability	Throw helpers are simpler and more efficient than an if block constructing a new exception instance. These were created for ArgumentNullException, ArgumentException, ArgumentOutOfRangeException, and ObjectDisposedException.

Code Analysis

Rule Id	Category	Description
CA1856	Performance	Fires when the ConstantExpected attribute is not applied correctly on a parameter.
CA1857	Performance	Fires when a parameter is annotated with ConstantExpected attribute but the provided argument is not a constant.
CA1858	Performance	Recommends upgrading the type of specific local variables, fields, properties, methods, parameters, and method return types from interface or abstract types to concrete types when possible. Using concrete types leads to higher quality generated code.
CA1860	Performance	To determine with a collection type has any elements, it is better to use Length, Count, or IsEmpty, than to call Any.
CA1861	Performance	Constant arrays passed as arguments are not reused when called repeatedly, which implies a new array is created each time. To improve performance, consider extracting the array to a static readonly field.
CA1865 -1867	Reliability	Enumerable.Cast<TResult>(IEnumerable) and Enumerable.OfType<TResult>(IEnumerable) require compatible types to function correctly. Widening and user-defined conversions aren't supported with generic types.
CA1510 -1513	**Maintainability**	**Throw helpers are simpler and more efficient than an if block constructing a new exception instance. These were created for ArgumentNullException, ArgumentException, ArgumentOutOfRangeException, and ObjectDisposedException.**


Crafting Moden Web Experiences
What's New in ASP.NET Core 8
TALELEARNCODE
Comprehensive Exploration of New Horizons for C# Developers
CHADGREEN

Blazor United

- Render content at either the component or page level with:
  - Static server rendering to generate static HTML
  - Interactive Server rendering
  - Interactive WebAssembly rendering
  - Interactive Auto rending

Blazor

- New Blazor Web App template

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps

beforeWebStart

afterWebStarted

beforeServerStart

aferServerStarted

beforeWebAssemblyStart

afterWebAssemblyStarted

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding
- Enhanced navigation and form handling

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding
- Enhanced navigation and form handling
- Streaming rendering

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding
- Enhanced navigation and form handling
- Streaming rendering
- Render Razor components outside of ASP.NET Core

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding
- Enhanced navigation and form handling
- Streaming rendering
- Render Razor components outside of ASP.NET Core
- QuickGrid

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding
- Enhanced navigation and form handling
- Streaming rendering
- Render Razor components outside of ASP.NET Core
- QuickGrid
- Blazor WebAssembly debugging improvements

Blazor

- New Blazor Web App template
- New JS initializers for Blazor Web Apps
- Form handling and model binding
- Enhanced navigation and form handling
- Streaming rendering
- Render Razor components outside of ASP.NET Core
- QuickGrid
- Blazor WebAssembly debugging improvements
- Blazor Identity UI

SignalR

- New approach to set the server timeout and Keep-Alive interval
- SignalR stateful reconnect

Minimal APIs

- User override culture

Minimal APIs

- User override culture
- Binding to forms

Minimal APIs

- User override culture
- Binding to forms
- Antiforgery

Native AOT

- New project template
- New CreateSlimBuilder

Native AOT

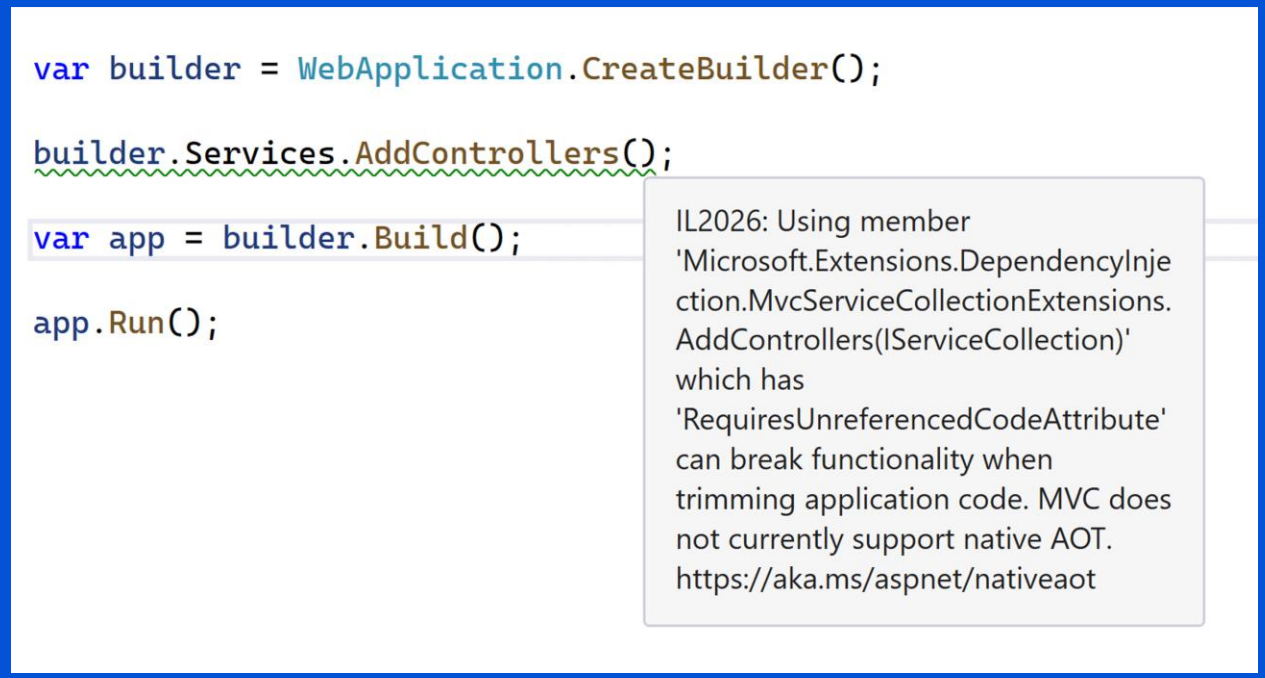
- New project template
- New CreateSlimBuilder
- New CreateEmptyBuilder

Native AOT

- New project template
- New CreateSlimBuilder
- New CreateEmptyBuilder
- Reduced app seize with configurable HTTPS support

Native AOT

- New project template
- New CreateSlimBuilder
- New CreateEmptyBuilder
- Reduced app seize with configurable HTTPS support
- Top-level APIs annotated for trim warnings



Authentication and Authorization

- Identity API endpoints
- Securing Swagger UI endpoints


Redefining
Data Access
What's New in Entity Framework Core 8
TALELEARNCODE
Comprehensive Exploration of New Horizons for C# Developers
CHADGREEN

Complex Types

- Complex types:
  - Are not identified or tracked by key value.
  - Must be defined as part of an entity type (you cannot have a DbSet)
  - Can be either .NET value types or reference types
  - Instances can be shared by multiple properties

Complex Types

- Limitations
  - Support collections of complex types
  - Allow complex type properties to be null
  - Map complex type properties to JSON columns
  - Constructor injection for complex types
  - Add seed data support for complex types
  - Map complex type properties for the Cosmos provider
  - Implement complex types for the in-memory datbase