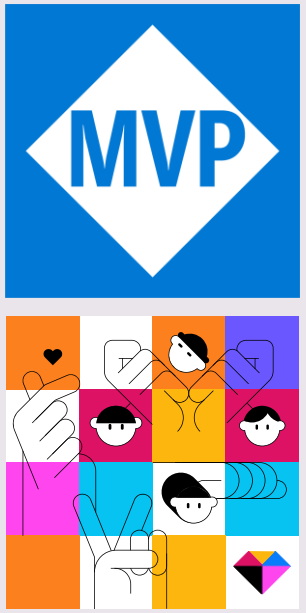
.NET Conf San Antonio

Presenters

LEAN TECHNIQUES



Daniel Ward

daninacan.com

daniel-ward-dev

danielwarddev.bsky.social

@danielwarddev

Patrick Robinson

patrickarobinson.com

patrick-robinson-dev

Sponsors!

- DevSA
https://devsanantonio.com/
- Geekdom
https://geekdom.com/
- Lean TECHniques
https://leantechniques.com/

.NET Versions

01

One major version per year

02

Even numbers are LTS

03

9 is STS, will be supported for 18 months

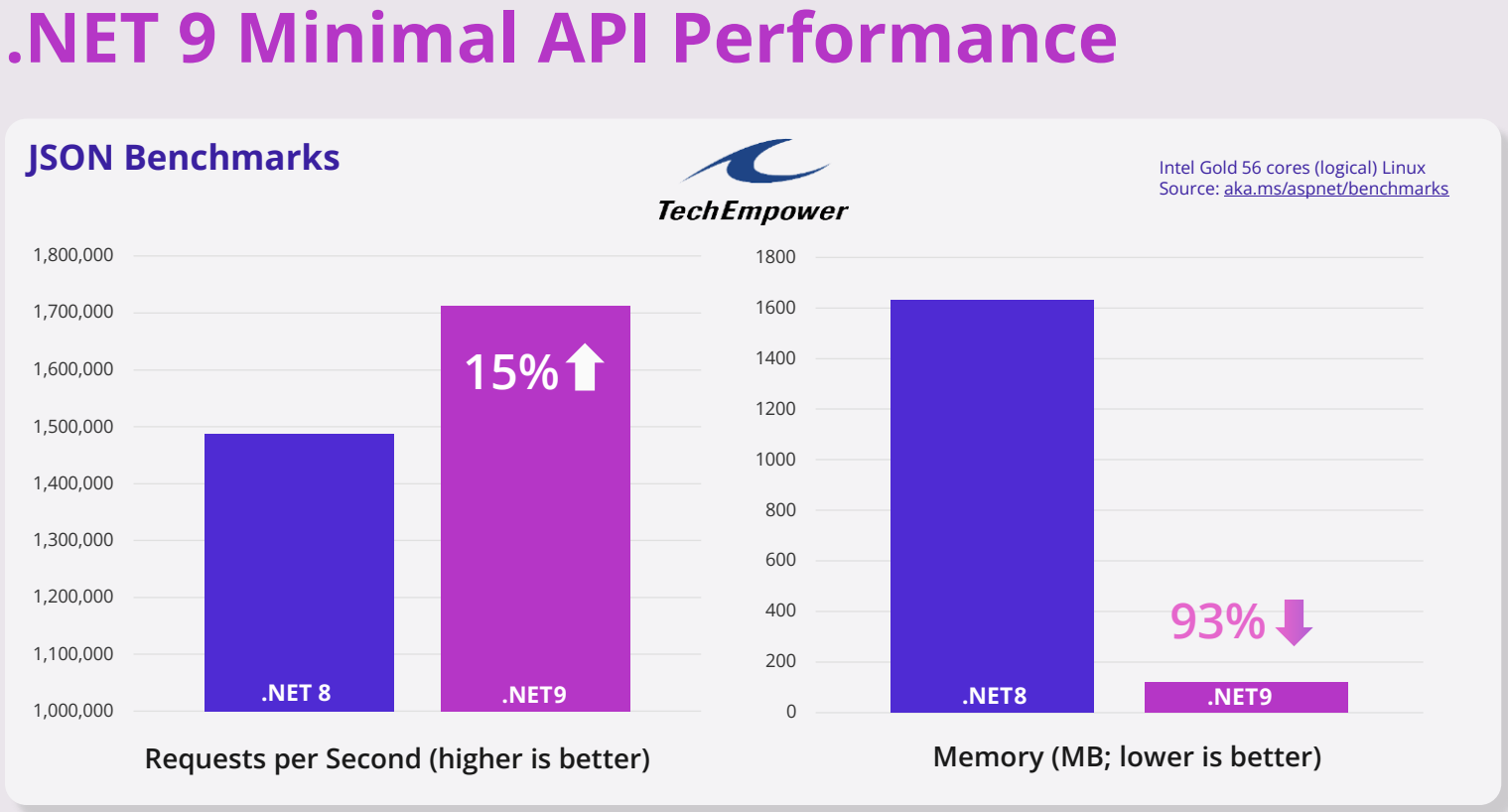
Outline

1. .NET
2. ASP.NET Core
3. C#
4. Entity Framework

.NET

Performance improvements

- Performance Improvements in .NET 9 - Stephen Toub



Feature toggles

- If a feature isn't enabled, the code itself is removed when trimming or compiling with Native AOT

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

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

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

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

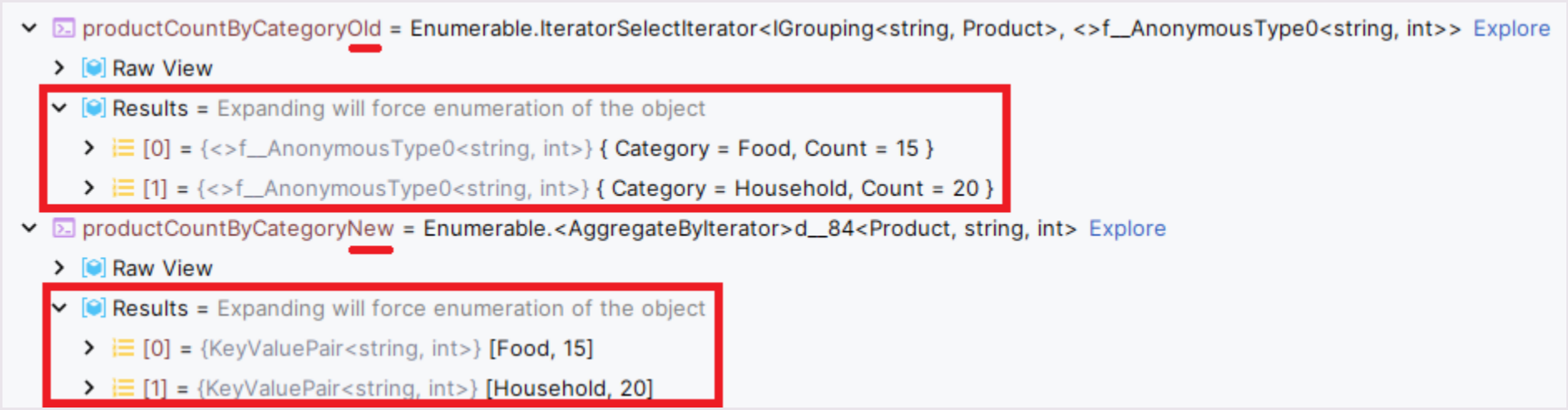
New LINQ methods

- CountBy, AggregateBy

```
1 {}  List<Product> products = [
2         new(1, "Bananas", "Food", 10),
3         new(2, "Apples", "Food", 5),
4         new(3, "Paper Towels", "Household", 20)
5     ];
6
7     var productCountByCategoryOld = products
8         .GroupBy(x => x.Category)
9         .Select(x => new { Category = x.Key, Count = x.Count() });
10
11    var productCountByCategoryNew = products
12        .CountBy(x => x.Category);
13
14    public record Product(int Id, string Name, string Category, int Price);
```

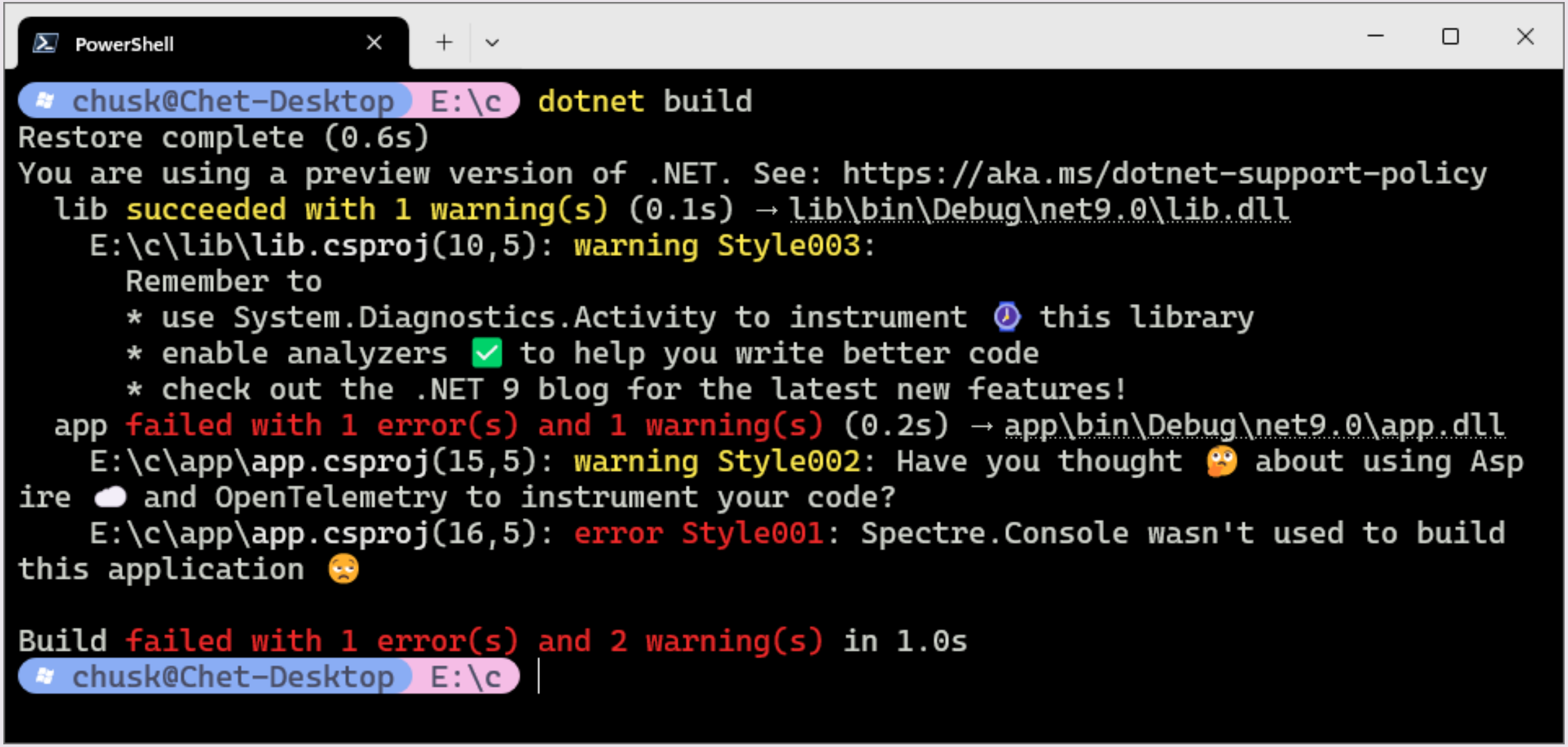
```
productCountByCategoryOld = Enumerable.IteratorSelectIterator<IGrouping<string, Product>, <>f__AnonymousType0<string, int>> Explore
  Raw View
  Results = Expanding will force enumeration of the object
    [0] = {<>f__AnonymousType0<string, int>} { Category = Food, Count = 2 }
    [1] = {<>f__AnonymousType0<string, int>} { Category = Household, Count = 1 }
productCountByCategoryNew = Enumerable.<CountByIterator>d__87<Product, string> Explore
  Raw View
  Results = Expanding will force enumeration of the object
    [0] = {KeyValuePair<string, int>} [Food, 2]
    [1] = {KeyValuePair<string, int>} [Household, 1]
```

```
{}
List<Product> products = [
    new(1, "Bananas", "Food", 10),
    new(2, "Apples", "Food", 5),
    new(3, "Paper Towels", "Household", 20)
];

var productCountByCategoryOld = products
    .GroupBy(x => x.Category)
    .Select(x => new { Category = x.Key, Count = x.Sum(y => y.Price) });

var productCountByCategoryNew = products
    .AggregateBy(x => x.Category, seed: 0, (totalPrice, current) => totalPrice + current.Price);

public record Product(int Id, string Name, string Category, int Price);
```



Terminal logger

- Released in .NET 8 and is now enabled by default
- More readable output
- Uses modern terminal capabilities such as:
  - Clickable links
  - Duration timers for MSBuild tasks
  - Color coding for warning/error messages


PowerShell
chusk@Chet-Desktop E:\c dotnet build
Restore complete (0.6s)
You are using a preview version of .NET. See: https://aka.ms/dotnet-support-policy
lib succeeded with 1 warning(s) (0.1s) → lib\bin\Debug\net9.0\lib.dll
E:\c\lib\lib.csproj(10,5): warning Style003:
Remember to
* use System.Diagnostics.Activity to instrument this library
* enable analyzers to help you write better code
* check out the .NET 9 blog for the latest new features!
app failed with 1 error(s) and 1 warning(s) (0.2s) → app\bin\Debug\net9.0\app.dll
E:\c\app\app.csproj(15,5): warning Style002: Have you thought about using Aspire and OpenTelemetry to instrument your code?
E:\c\app\app.csproj(16,5): error Style001: Spectre.Console wasn't used to build this application
Build failed with 1 error(s) and 2 warning(s) in 1.0s
chusk@Chet-Desktop E:\c

NuGet

- As of .NET 8, Nuget audits direct package references for known vulnerabilities
- In .NET 9, by default it now audits both direct AND transitive package references
- Faster dependency resolution for large repos
  - On a large internal Microsoft repo, restore went from 16 mins to 2 mins

GUIDs

- Previously, Guid.NewGuid()
- Now has Guid.CreateVersion7() and Guid.CreateVersion7(DateTimeOffset)
- Still cryptographically secure random data, but now also partially based on timestamp
- Guids can now be sorted

ASP.NET Core

OpenAPI

- Default template no longer uses Swashbuckle.AspNetCore. Included package is now Microsoft.AspNetCore.OpenApi
- Swagger still supported
- https://github.com/dotnet/aspnetcore/issues/54599

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

builder.Services.AddOpenApi();

var app = builder.Build();

app.MapOpenApi();

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

app.Run();
```

OpenAPI

- Microsoft.AspNetCore.OpenApi supports trimming and Native AOT
- OpenAPI documents can now be generated at build-time with Microsoft.Extensions.ApiDescription.Server
- OpenIdConnectHandler supports Pushed Authorization Requests (PAR) - moves authorization parameters from the front channel (redirect URLs) to the back channel (direct backend HTTP calls)
- PAR enabled by default if the identity provider supports it
- Can optionally disable it or require it

OpenAPI

- Simpler way to customize authorization message parameters

```
builder.Services.AddAuthentication().AddOpenIdConnect(options =>
{
    options.Events.OnRedirectToIdentityProvider = context =>
    {
        context.ProtocolMessage.SetParameter("prompt", "login");
        context.ProtocolMessage.SetParameter("audience", "https://api.example.com");
        return Task.CompletedTask;
    };
});
```

```
builder.Services.AddAuthentication().AddOpenIdConnect(options =>
{
    options.AdditionalAuthorizationParameters.Add("prompt", "login");
    options.AdditionalAuthorizationParameters.Add("audience", "https://api.example.com");
});
```

Static Assets

```
app.UseRouting();

app.UseAuthorization();

+app.MapStaticAssets();
-app.UseStaticFiles();
app.MapRazorPages();

app.Run();
```

- MapStaticAssets() instead of UseStaticFiles()
- Usually a drop-in replacement for UseStaticFiles()
- Works best with assets the app has knowledge of at build + publish time. If it serves assets outside of the app (disk, embedded resources, elsewhere), use UseStaticFiles()
- Compression at build time for all assets in the app
- gzip during development, gzip + brotli during publish
- ~70%-90% size reduction for CSS and JS files
- Uses content-based ETags. ETag for each resources is an encoded hash of the content. Ensures a browser only redownloads a file if its contents have changed
- You CAN compress on the server, but more complicated, less performant, less reduction

File	Original (KB)	Compressed (KB)	% Reduction
Default Razor Pages template			
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 component library			
fluent.js	384	73	80.99%
fluent.css	94	11	88.30
Total	**478**	**84**	**82.43**
MudBlazor Blazor component library			
MudBlazor.min.css	541	37.5	93.07%
MudBlazor.min.js	47.4	9.2	80.59%
Total	**588.4**	**46.7**	**92.07%**

Blazor

- New template maui-blazor-web. Easier to create .NET MAUI native + Blazor web client apps that share same UI
- New simplified API for querying component states at runtime
- RendererInfo.Name
  - Where component is executing (Static, Server, WebAssembly, WebView)
  - Useful for debugging and optimizing component performance
- RendererInfo.IsInteractive
  - True when rendering interactively, false when prerendering or static SSR
  - Useful if component has different behaviors based on this
- ComponentBase.AssignedRenderMode
  - InteractiveServer, InteractiveAuto, InteractiveWebAssembly
  - Useful for optimizing a component’s performance

Blazor

- InputNumber now takes type=”range” to make a slider/dial rather than a text box
- Razor components now support constructor injection

```
public partial class ConstructorInjection(NavigationManager navigation)
{
    private void HandleClick()
    {
        navigation.NavigateTo("/counter");
    }
}
```

SignalR

- SignalR supports trimming and Native AOT, both server and client
- Hub methods can now accept a base class instead of the derived class, allowing for polymorphism
- Must annotate the base class

```
public class MyHub : Hub
{
    public void Method(JsonPerson person)
    {
        if (person is JsonPersonExtended)
        {
        }
        else if (person is JsonPersonExtended2)
        {
        }
        else
        {
        }
    }
}

[JsonPolymorphic]
[JsonDerivedType(typeof(JsonPersonExtended), nameof(JsonPersonExtended))]
[JsonDerivedType(typeof(JsonPersonExtended2), nameof(JsonPersonExtended2))]
private class JsonPerson
{
    public string Name { get; set; }
    public Person Child { get; set; }
    public Person Parent { get; set; }
}

private class JsonPersonExtended : JsonPerson
{
    public int Age { get; set; }
}

private class JsonPersonExtended2 : JsonPerson
{
    public string Location { get; set; }
}
```

Minimal APIs

- InternalServerError and InternalServerError<T> added to TypedResults

```
var app = WebApplication.Create();

app.MapGet("/", () => TypedResults.InternalServerError("Something went wrong!"));

app.Run();
```

HybridCache

- Drop-in replacement for both IDistributedCache and IMemoryCache
- Simple, unified API for both in-process and out-of-process caching

IDistributedCache

```
public class SomeService(IDistributedCache cache)
{
    public async Task<SomeInformation> GetSomeInformationAsync
        (string name, int id, CancellationToken token = default)
    {
        var key = $"someinfo:{name}:{id}"; // Unique key for this combination.
        var bytes = await cache.GetAsync(key, token); // Try to get from cache.
        SomeInformation info;
        if (bytes is null)
        {
            // Cache miss; get the data from the real source.
            info = await SomeExpensiveOperationAsync(name, id, token);

            // Serialize and cache it.
            bytes = SomeSerializer.Serialize(info);
            await cache.SetAsync(key, bytes, token);
        }
        else
        {
            // Cache hit; deserialize it.
            info = SomeSerializer.Deserialize<SomeInformation>(bytes);
        }
        return info;
    }

    // This is the work we're trying to cache.
    private async Task<SomeInformation> SomeExpensiveOperationAsync(string name, int id,
        CancellationToken token = default)
    { /* ... */ }
}
```

HybridCache

```
public class SomeService(HybridCache cache)
{
    public async Task<SomeInformation> GetSomeInformationAsync
        (string name, int id, CancellationToken token = default)
    {
        return await cache.GetOrCreateAsync(
            $"someinfo:{name}:{id}", // Unique key for this combination.
            async cancel => await SomeExpensiveOperationAsync(name, id, cancel),
            token: token
        );
    }
}
```

Developer Exception Page

- Now includes endpoint metadata
- Some small quality of life improvements - better text wrapping, bigger text, more consistent table sizes

An unhandled exception occurred while processing the request.

InvalidOperationException: A test error

Program+<>c__DisplayClass0_0.<<Main>$>b__0() in **Program.cs**, line 37

Stack Query Cookies Headers Routing

Endpoint

Name	Value
Display Name	HTTP: GET /weatherforecast
Route Pattern	/weatherforecast
Route Order	0
Route HTTP Method	GET

Endpoint Metadata

Type	Detail
RuntimeMethodInfo	WeatherForecast[] <<Main>$>b__0()
HttpMethodMetadata	HttpMethods: GET, Cors: False
ProducesResponseTypeMetadata	Produces StatusCode: 200, ContentTypes: application/json, Type: WeatherForecast[]
EndpointNameMetadata	EndpointName: GetWeatherForecast
RouteNameMetadata	RouteName: GetWeatherForecast
OpenApiOperation	Microsoft.OpenApi.Models.OpenApiOperation
RouteDiagnosticsMetadata	Route: /weatherforecast

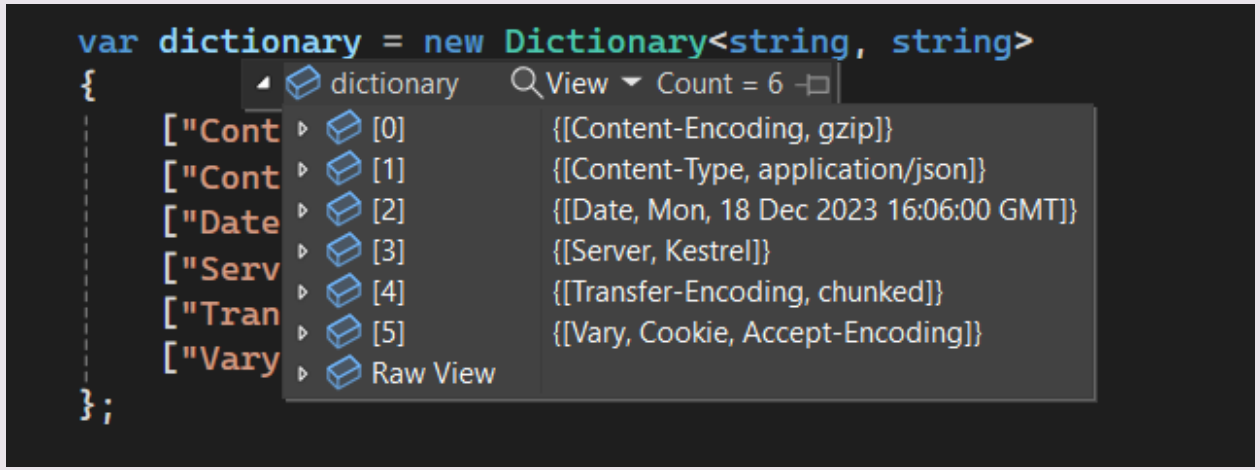
Route Values

No route values.

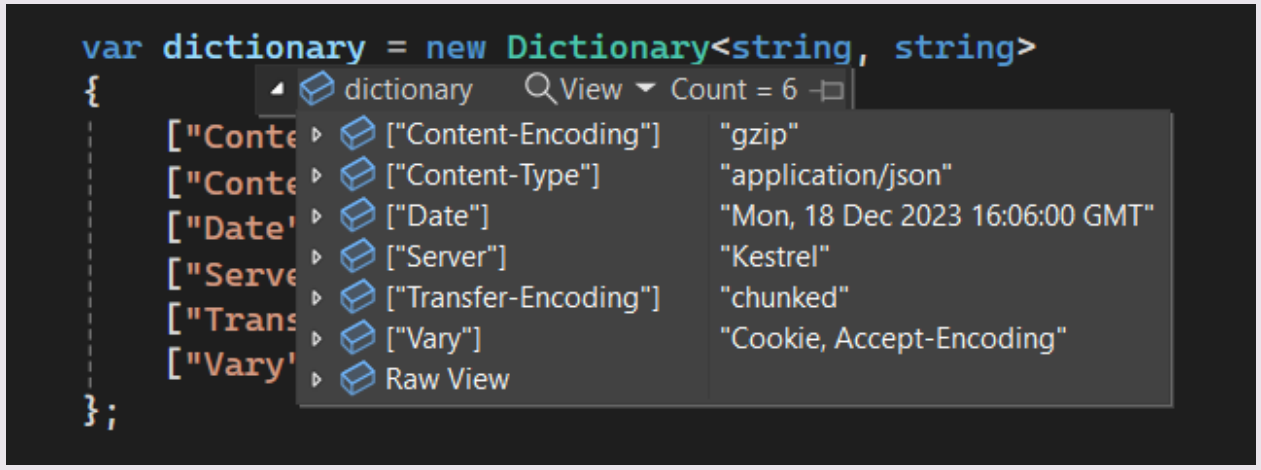
Debugging

- Improved debugging display of some key-value collection types

Old



New



- HTTP headers, query strings, forms, cookies, view data, route data, features

Improved Kestrel connection metrics

- Metadata about why a connection failed is now included
- The kestrel.connection.duration metric now includes the connection close reason in the error.type attribute
- Examples: tls_handshake_failed, connection_reset, request_headers_timeout, max_request_body_size_exceeded
- Previously, low-level logging was required, which can be expensive to generate and store, and difficult to parse to find info
- Useful for debugging production connection issues
- Can be used for dashboards and alerts

C#

Params Collections

- You can now use any recognized collection type for params, not just an array

```
public static void Concat<T>(params ReadOnlySpan<T> items)
{
    for (var i = 0; i < items.Length; i++)
    {
        Console.Write(items[i]);
        Console.Write(" ");
    }
    Console.WriteLine();
}
```

New Escape Sequence

- You can now represent the ESCAPE character as “\e”

```
public class EscapeCharacter
{
    public string NewEscapeCharacter = "\e";
    public string[] OldEscapeCharacters = ["\u001b", "\x1b"];
}
```

Implicit Index Access

- The “from the end” index operator, “^”, can now be used in object initializer expressions
- The example below sets the array to [9,8,7,6,5,4,3,2,1,0]

```
1 usage
public class TimerRemaining
{
    1 usage
    public int[] buffer { get; set; } = new int[10];
}

public class ImplicitIndexAccess
{
    public ImplicitIndexAccess()
    {
        var countdown = new TimerRemaining()
        {
            buffer =
            {
                [^1] = 0,[^2] = 1,[^3] = 2,[^4] = 3,[^5] = 4,[^6] = 5,[^7] = 6,[^8] = 7,[^9] = 8,[^10] = 9
            }
        };
    }
}
```

Partial Properties

- You can now declare partial properties and partial indexers

```
public partial class C
{
    // declaration
    public partial string Name { get; set; }
}

public partial class C
{
    // implementation
    private string _name;
    public partial string Name
    {
        get ⇒ _name;
        set ⇒ _name = value;
    }
}
```

Ref Updates

- Ref and unsafe in iterators and async methods
- Allows ref struct as a new anti-constraint
- Ref structs can now implement interfaces
  - But cannot convert to interface types

```
public class C<T> where T : allows ref struct
{
    // use T as a ref struct
    public void Method(scoped T p)
    {
        // the parameter p must follow ref safety rules
    }

}
```

Other Updates

- New lock type and semantics
- Method group natural type improvements
- Overload resolution priority

Entity Framework

Azure Cosmos DB for NoSql

- Some high-impact breaking changes
- Improvements querying with partition keys and document IDs
- Hierarchical partition keys - now supports subpartitioning up to three levels
- Improved LINQ querying capabilities
- Role-based access
- Synchronous I/O is now blocked by default

Azure Cosmos DB for NoSql

- Simplified ID properties without discriminators
- Discriminator property renamed to $type

```
{
    "id": "Blog|1099",
    ...
}
```

```
{
    "id": 1099,
    "$type": "Blog",
    ...
}
```

Azure Cosmos DB for NoSql

- (Preview) Vector similarity search
- Pagination support
- FromSql for safer SQL querying

```
var firstPage = await context.Posts
    .OrderBy(p => p.Id)
    .ToPageAsync(pageSize: 10, continuationToken: null);

var continuationToken = firstPage.ContinuationToken;
foreach (var post in page.Values)
{
    // Display/send the posts to the user
}
```

```
var nextPage = await context.Sessions.OrderBy(s => s.Id).ToPageAsync(10, continuationToken);
```

```
var maxAngle = 8;
_ = await context.Blogs
    .FromSql($"SELECT VALUE c FROM root c WHERE c.Angle1 <= {maxAngle}")
    .ToListAsync();
```

AOT and Pre-Compiled Queries

- Highly experimental feature – not yet suited for production use
- Generates C# interceptors that execute each specific query
- Each interceptor contains the finalized SQL for the query

Linq and Sql Translation

- Support for GroupBy with a complex type
- ExecuteUpdate accepts complex type properties

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

```
var newAddress = new Address("Gressenhall Farm Shop", null, "Beetley", "Norfolk", "NR20 4DR");

await context.Stores
    .Where(e => e.Region == "Germany")
    .ExecuteUpdateAsync(s => s.SetProperty(b => b.StoreAddress, newAddress));
```

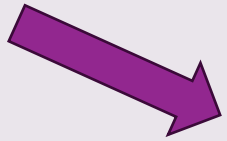
Linq and Sql Translation

- Prune unneeded elements from SQL

```
var customers = await context.Customers.Where(o => o.Orders.Any()).ToListAsync();
```

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE EXISTS (
    SELECT 1
    FROM [Orders] AS [o]
    LEFT JOIN [DiscountedOrders] AS [d] ON [o].[Id] = [d].[Id]
    WHERE [c].[Id] = [o].[CustomerId])
```

EF8



EF9

```
SELECT [c].[Id], [c].[Name]
FROM [Customers] AS [c]
WHERE EXISTS (
    SELECT 1
    FROM [Orders] AS [o]
    WHERE [c].[Id] = [o].[CustomerId])
```

Linq and Sql Translation

- Translations involving GREATEST/LEAST
- Force or prevent query parameterization

```
async Task<List<Post>> GetPosts(int id)
    => await context.Posts
        .Where(e => e.Title == ".NET Blog" && e.Id == id)
        .ToListAsync();
```

```
async Task<List<Post>> GetPostsForceConstant(int id)
    => await context.Posts
        .Where(e => e.Title == ".NET Blog" && e.Id == EF.Constant(id))
        .ToListAsync();
```

```
async Task<List<Post>> GetPostsForceParameter(int id)
    => await context.Posts
        .Where(e => e.Title == EF.Parameter(".NET Blog") && e.Id == id)
        .ToListAsync();
```

Linq and Sql Translations

- Parameterized primitive collections

```
async Task<List<Post>> GetPostsPrimitiveCollection(int[] ids)
    => await context.Posts
        .Where(e => e.Title == ".NET Blog" && ids.Contains(e.Id))
        .ToListAsync();
```

```
async Task<List<Post>> GetPostsForceConstantCollection(int[] ids)
    => await context.Posts
        .Where(
            e => e.Title == ".NET Blog" && EF.Constant(ids).Contains(e.Id))
        .ToListAsync();
```

Linq and Sql Translation

- Inlined uncorrelated subqueries – reduced round trips!

```
var dotnetPosts = context
    .Posts
    .Where(p => p.Title.Contains(".NET"));

var results = dotnetPosts
    .Where(p => p.Id > 2)
    .Select(p => new { Post = p, TotalCount = dotnetPosts.Count() })
    .Skip(2).Take(10)
    .ToArray();
```

Linq and Sql Translation

- Aggregate functions over subqueries and aggregates on SQL Server

```
var latestPostsAverageRatingByLanguage = await context.Blogs
    .Select(x => new
    {
        x.Language,
        LatestPostRating = x.Posts.OrderByDescending(xx => xx.PublishedOn).FirstOrDefault()!.Rating
    })
    .GroupBy(x => x.Language)
    .Select(x => x.Average(xx => xx.LatestPostRating))
    .ToListAsync();
```

Linq and Sql Translation

- Queries using Count != 0 optimized
- C# semantics for comparison operations on nullable values

```
var negatedNullableComparisonFilter = await context.Entities
    .Where(x => !(x.NullableIntOne > x.NullableIntTwo))
    .Select(x => new { x.NullableIntOne, x.NullableIntTwo }).ToListAsync();
```

EF8

```
SELECT [e].[NullableIntOne], [e].[NullableIntTwo]
FROM [Entities] AS [e]
WHERE NOT ([e].[NullableIntOne] > [e].[NullableIntTwo])
```

EF9

```
SELECT [e].[NullableIntOne], [e].[NullableIntTwo]
FROM [Entities] AS [e]
WHERE CASE
    WHEN [e].[NullableIntOne] > [e].[NullableIntTwo] THEN CAST(0 AS bit)
    ELSE CAST(1 AS bit)
END = CAST(1 AS bit)
```

Linq and Sql Translation

- Order and OrderDescending Linq operators
- Improved translation of logical negation operator (!)

```
var negatedContainsSimplification = await context.Posts
    .Where(p => !p.Content.Contains("Announcing"))
    .Select(p => new { p.Content }).ToListAsync();
```

EF8

```
SELECT "p"."Content"
FROM "Posts" AS "p"
WHERE NOT (instr("p"."Content", 'Announcing') > 0)
```

EF9

```
SELECT "p"."Content"
FROM "Posts" AS "p"
WHERE instr("p"."Content", 'Announcing') <= 0
```

Migrations & Tooling

- Protection against concurrent migrations
- Warn when multiple migration operations can’t be run inside a transaction
- Improved data seeding
- Reduced code size for migration of existing table to Sql Server temporal table
- Tooling updates for fewer rebuilds

Model Building

- Auto-compiled models
- MSBuild Integration (Microsoft.EntityFrameworkCore.Tasks)
- Read-only primitive collections
- Specify fill-factor for keys and indexes
- Make existing model building conventions more extensible
- Update ApplyConfigurationsFromAssembly to call non-public constructors

Sql Server HierarchyId

- Syntactic sugar for HierarchyId path generation

```
var daisy = await context.Halflings.SingleAsync(e => e.Name == "Daisy");
```

```
var child1 = new Halfling(HierarchyId.Parse(daisy.PathFromPatriarch, 1), "Toast");
var child2 = new Halfling(HierarchyId.Parse(daisy.PathFromPatriarch, 2), "Wills");
```

```
var child1b = new Halfling(HierarchyId.Parse(daisy.PathFromPatriarch, 1, 5), "Toast");
```

Recap

- .NET
- ASP.NET Core
- C#
- Entity Framework

Thank you!