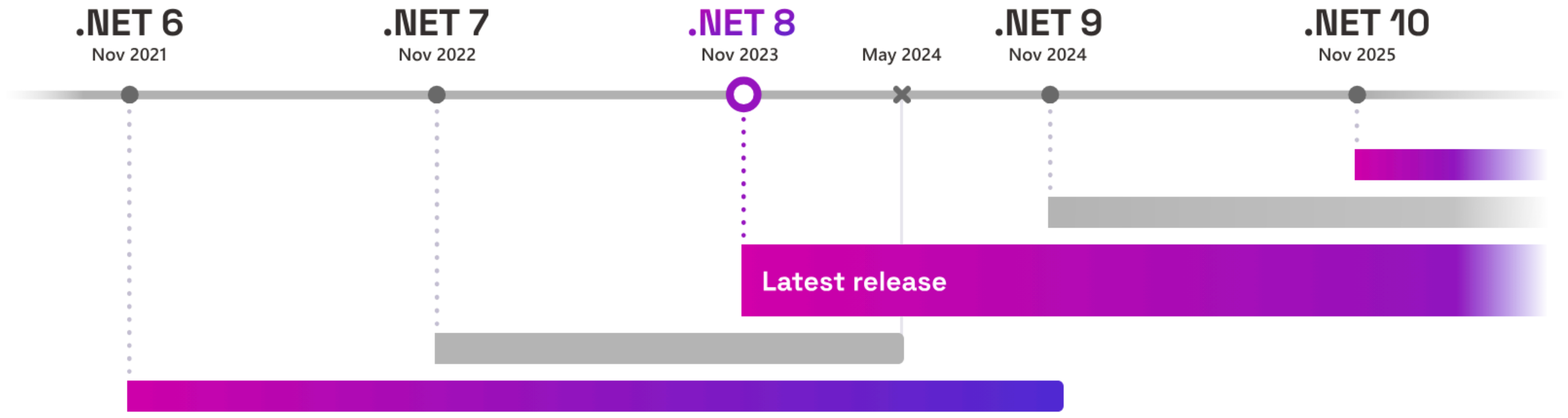
Robert Haken

Novinky .NET 9

.NET Timeline



🟪	**LONG TERM SUPPORT** Patches for 36 months
⬜	**STANDARD TERM SUPPORT** Patches for 18 months

Co se chystá do .NET 9

- Vize: Cloud native & Intelligent app development
    - .NET Apire, NativeAOT, DATAS GC
    - AI, ML.NET
    - performance

.NET Runtime - Feature Switches

- feature switches (.NET 5+)

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

- new attribute model for feature switches

```
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() => ...;
}
```

  - treated as constant when trimming

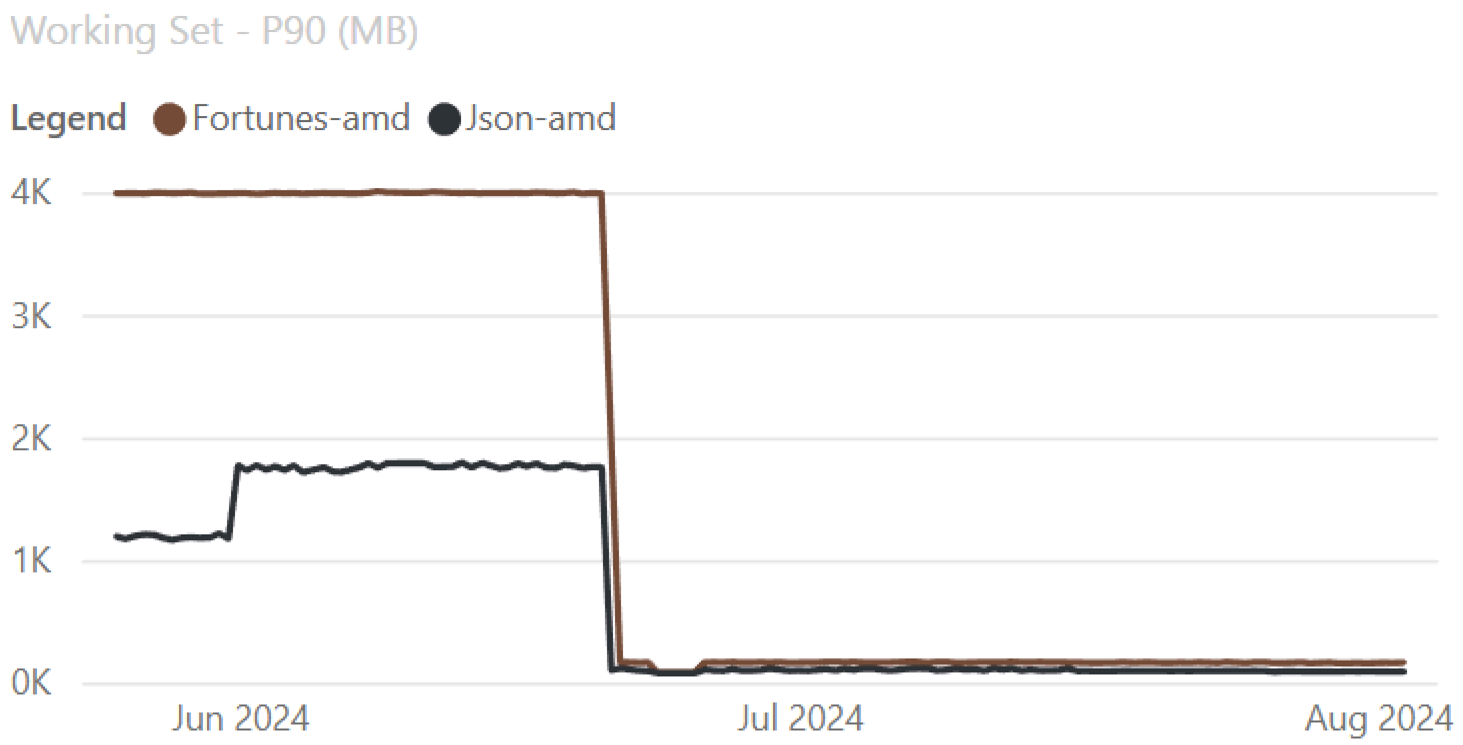
- `[FeatureGuard(typeof(RequiresDynamicCodeAttribute))]`

.NET Runtime - GC DATAS

- Garbage Collector: Dynamic Adaptation to Application Size
- .NET 9: updated, improved + enabled by default (opt-in in .NET8)
- DATAS: application heap size should be roughly proportional to the long-lived data size (LDS)
  - vs. ServerGC: treats the process as the dominant one on the machine
- DATAS
  - adjusts the number of heaps when needed (WS GC: 1, ServerGC: cores)
  - sets the allocation budget based on the long-lived data size
  - sets the actual amount of allocations allowed based on throughput
  - does full-compacting GCs when needed

.NET Runtime - GC DATAS

- 80% working set reduction vs. 2-3% throughput reduction



.NET Runtime - Performance improvements

- loop optimizations
- inlining improvements
- PGO improvements: type checks and casts (fast paths)
- Arm64 vectorization and code generation
- faster exceptions (2-4x faster)
- code layout
- reduced address exposure
- AVX10v1 support (Intel SIMD)
- hardware intrinsic code generation
- constant folding for floating point and SIMD operations
- Arm64 SVE support (SIMD)
- Object stack allocation for (unescaped) boxes

Core .NET Libraries

- `Base64Url.EncodeToString(bytes)` - používá `-` místo `+` a `_` místo `/`, nemá `=`
- `BinaryFormatter` removed (API throws exception)
- `TimeSpan.From*(Int64)` alternatives to `From*(double)`
  - eg. `TimeSpan.FromSeconds(seconds: 10, miliseconds: 5, microseconds: 3)`
- Dependency injection - `[ActivatorUtilitiesConstructor]` always wins
- `Debug.Assert(bool condition)` reports condition - `[CallerArgumentExpression("condition")]`
- `SearchValues` support for searching substrings withing a larger string `charsSpan.IndexOfAny(searchValues)`
- `Activity.AddLink()`, `Metrics.Gauge<T>` instrument (OpenTelemetry)
- `Tensor<T>` for AI (`System.Numerics.Tensors` NuGet package) - experimental

Core .NET Libraries - Collections

- Collection lookups with spans

```
var wordCounts = new Dictionary<string, int>();
var spanLookup = wordCounts.GetAlternateLookup<string, int, ReadOnlySpan<char>>();
```

- `OrderedDictionary<TKey, TValue>` - `Add`, `RemoveAt`, `Insert`, `foreach` drží pořadí
- `PriorityQueue.Remove()` pro podporu priority updates (kombinací `Remove` + `Enqueue`)
- `ReadOnlySet<T>` pro `ISet<T>` - doplnění existující dvojice `ReadOnlyCollection<T>` (`IList<T>`) a `ReadOnlyDictionary<T>` (`IDictionary<>`)

Core .NET Libraries - LINQ

- new `CountBy()` and `AggregateBy()` methods with built-in `GroupBy()`
  - no need to allocate intermediate groupings

```
KeyValuePair<string, int> mostFrequentWord = sourceText
    .Split(' ')
    .Select(word => word.ToLowerInvariant())
    .CountBy(word => word)
    .MaxBy(pair => pair.Value);
```

- `Index()` method to get implicit item index - returns `(index, item)` tuples

```
IEnumerable<string> lines2 = File.ReadAllLines("output.txt");
foreach ((int index, string line) in lines2.Index())
{
    Console.WriteLine($"Line number: {index + 1}, Line: {line}");
}
```

Core .NET Libraries - Networking

- Server-sent events (SSE) library `System.Net.ServerSentEvents` (NuGet package)

```
using var responseStream = await httpClient.GetStreamAsync(...);

await foreach (SseItem<string> e in SseParser.Create(responseStream).EnumerateAsync())
{
    Console.WriteLine(e.Data);
}

// popř.
await foreach (SseItem<T> item in SseParser.Create(responseStream,
                                (_, bytes) => JsonSerializer.Deserialize<T>(bytes))
                        .EnumerateAsync())
{
    ProcessItem(item.Data);
}
```

- SocketsHttpHandler is default in HttpClientFactory
- *TLS resume* with client certificates on Linux

Core .NET Libraries - JSON Serialization

- Indentation in `JsonSerializerOptions` - `WriteIndented`, `IndentCharacter`, `IndentSize`
- `JsonSerializerOptions.Web` singleton with ASP.NET Core defaults (read-only)
- `JsonSchemaExporter` to generate JSON schema from type

```
JsonSchemaExporter.GetJsonSchemaAsNode(JsonSerializerOptions.Default, typeof(Book))
```

- Respects nullable annotations (incl. `options.RespectNullableAnnotations` flag)
- `options.RespectRequiredConstructorParameters` (default historicky `false`)

Core .NET Libraries - Spans

- new file helpers to write span/memory to files - `File.WriteAllText(path, textSpan)`
- `span.StartsWith<T>(...)` and `span.EndsWith<T>(...)`
- `params ReadOnlySpan<T>` overloads (C# 13), over 60 methods, eg. `String.Join(...)`
- enumerable `span.Split()` overloads (enumeration over `Range`)

```
foreach (Range segment in span.Split(','))
{
        Console.WriteLine(span[segment]);
}
```

(naming inconsistence with `Regex.EnumerateSplits()`, see below)

Core .NET Libraries - Threading

- `Task.WhenEach()` iterate through tasks as they complete

```
Task<string> dotnet = httpClient.GetStringAsync("http://dot.net");
Task<string> bing = httpClient.GetStringAsync("http://www.bing.com");
Task<string> ms = httpClient.GetStringAsync("http://microsoft.com");

await foreach (Task<string> t in Task.WhenEach(bing, dotnet, ms))
{
    Console.WriteLine(t.Result);
}
```

- prioritized unbounded channels - `Channel.CreateUnboundedPrioritized<T>()` - write any / read in order
- `Interlocked.Exchange<T>()` and `CompareExchange<T>()` for more types (generic constraints removed, any type works)

Core .NET Libraries - Cryptography

- `byte[] hash = CryptographicOperations.HashData(hashAlgorithmName, data);`
- KMAC hashing algorithm
- `X509CertificateLoader` class instead of new X509Certificate2(something)
- OpenSSL providers support
- support for Windows CNG virtualization-based security (VBS)

Core .NET Libraries

- reflection - persisted assemblies - `PersistedAssemblyBuilder` (save)
- reflection - `TypeName.Parse(name)` separate parser for decoupled type-name parsing
- `Regex.EnumerateSplits()` - non-allocating splitting over regex separator

```
ReadOnlySpan<char> input = "Hello, world! How are you?";
foreach (Range r in Regex.EnumerateSplits(input, "[aeiou]"))
{
    Console.WriteLine($"Split: \"{input[r]}\"");
}
```

- `[GeneratedRegex]` on properties (see C# 13 `partial` properties)
- `Guid.CreateVersion7()` - timestamp based natural sort order (eg. for DB clustered indexes)
- `SomeInt.BigMul(num1, num2)` - returns next larger integer type `int * int = Int64`

ASP.NET Core

Blazor - Render modes

- Render mode info for `ComponentBase`
    - `RendererInfo`
        - `RendererInfo.Name = Static | Server | WebAssembly | WebView`
        - `RendererInfo.IsInteractive = true | false`
    - `AssignedRenderMode =`
        - `null` - static SSR
        - `InteractiveServer | InteractiveWebAssembly | InteractiveAuto`

Blazor - Static SSR

- Static SSR within globally-interactive BWA
    - `@attribute [ExcludeFromInteractiveRouting]`
    - `bool HttpContext.AcceptsInteractiveRouting()` extension method
    - `HttpContext.GetEndpoint()?.Metadata`

```
// App.razor
<Routes @rendermode="@PageRenderMode" />

@code {
        [CascadingParameter] private HttpContext HttpContext { get; set; }

        private IComponentRenderMode PageRenderMode
                => HttpContext.AcceptsInteractiveRouting() ? InteractiveAuto : null;
}
```

Blazor

- Constructor-injection

```
public partial class Counter : ComponentBase
{
        private readonly NavigationManager _navigationManager;

        public Counter(NavigationManager navigationManager)
        {
                _navigationManager = navigationManager;
        }
}
```

- Server: Improved reconnection experience (UI, intervals), WebSocket compression
- Simplified authentication state serialization
  - `.AddAuthenticationStateSerialization()` for server
  - `.AddAuthenticationStateDeserialization()` for browser
- *.NET MAUI Blazor Hybrid and Web App* solution template

ASP.NET Core - Static Assets Delivery Optimization

- `MapStaticAssets()` instead of `UseStaticFiles()` when files known at build + publish time
- build-time compression (gzip in development, Brotli when published)
- fingerprinting: `Cache-Control: immutable`, `ETag` for non-fingerprinted files
- minification not included - other (build-)tools involved
- Blazor, Razor Pages and MVC support
- for Blazor:
  - new `ComponentBase.Assets` property
  - `<link rel="stylesheet" href="@Assets["bootstrap/bootstrap.min.css"]" />`
  - `<StaticWebAssetProjectMode>Default</..>` in `.Client.csproj`

ASP.NET Core - `HybridCache`

- `Microsoft.Extensions.Caching.Hybrid` NuGet package + `builder.Services.AddHybridCache()`
- usage

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

- `IDistributedCache` for second-level, "stampede" protection, tags
- configurable serialization, object-reusability

ASP.NET Core

- SignalR: Polymorphic type support (hub methods can now accept a base class)
- SignalR: Improved OpenTelemetry Activities
- SignalR: Trimming and NativeAOT support (client i server)
- MinimalAPI: `TypedResults.InternalServerError(message)` (HTTP 500)
- OpenAPI
  - `builder.Services.AddOpenApi()` + `app.MapOpenApi()` generates `/openapi/v1.json` from controllers and MinimalAPIs
  - `[Required]` + `[DefaultValue]` support
  - schema transformers
- Developer exception page - added *Routing / Endpoint Metadata* section

ASP.NET Core

- Authentication and authorization
    - support for OAuth/OIDC Pushed Authorization Requests (PAR)
    - OAuth/OIDC Parameter Customization

      `options.AdditionalAuthorizationParameters.Add(name, value)`
- `ExceptionHandlerMiddleware` - exception-based status codes

```
app.UseExceptionHandler(new ExceptionHandlerOptions
{
    StatusCodeSelector = ex => ex is TimeoutException
        ? StatusCodes.Status503ServiceUnavailable
        : StatusCodes.Status500InternalServerError,
});
```

- `app.MapHealthChecks("/healthz").DisableHttpMetrics();` + `[DisableHttpMetrics]`

C# 13

C# 13

- `params` collections

```
public int Sum(params ReadOnlySpan<int> nums)
{
}
```

- new `Lock` object

```
private readonly Lock _lock = new Lock();

lock (_lock)
{
}
```

- `partial` properties and indexers

```
[GeneratedRegex(".*")]
public partial Regex AnyString { get; set; }
```

Entity Framework Core 9

- Azure Cosmos DB for NoSQL enhancements
- `GroupBy()` complex types
- `Math.Min()` and `Max()` to T-SQL `GREATEST()` and `LEAST()`
- new `.ToHashSetAsync()` methods
- Queries using `Count != 0` are optimized (`EXISTS`)
- `TimeOnly.FromDateTime()` and `FromTimeSpan()` to SQL translation
- `ExecuteUpdate()` complex types support
- auto-compiled models (NuGet, MSBuild task, auto-discovery, ...)
- read-only primitive collections
- caching for sequences, eg. `HasSequence<int>("name").UseCache(3)`
- fill-factor for keys and indexes `HasIndex(..).HasFillFactor(80)`

Reference

- What's new in .NET 9 | Microsoft Learn
    - What's new in .NET 9 runtime | Microsoft Learn
    - What's new in .NET libraries for .NET 9 | Microsoft Learn
    - What's new in the SDK for .NET 9 | Microsoft Learn
- What's new in ASP.NET Core 9.0 | Microsoft Learn
- What's new in C# 13 | Microsoft Learn
- What's new with identity in .NET 8 - .NET Blog
- What's New in EF Core 9 | Microsoft Learn
- core/release-notes/9.0/README.md at main · dotnet/core
- .NET 9 Release Index · dotnet/core · Discussion #9234
- Dynamically Adapting To Application Sizes | by Maoni0 | Medium