What's new in .NET 7

Article • 03/09/2023

.NET 7 is the successor to .NET 6 and focuses on being unified, modern, simple, and *fast*. .NET 7 will be supported for 18 months as a standard-term support (STS) release (previously known as a *current* release).

This article lists the new features of .NET 7 and provides links to more detailed information on each.

To find all the .NET articles that have been updated for .NET 7, see .NET docs: What's new for the .NET 7 release.

Performance

Performance is a key focus of .NET 7, and all of its features are designed with performance in mind. In addition, .NET 7 includes the following enhancements aimed purely at performance:

- On-stack replacement (OSR) is a complement to tiered compilation. It allows the runtime to change the code executed by a currently running method in the middle of its execution (that is, while it's "on stack"). Long-running methods can switch to more optimized versions mid-execution.
- Profile-guided optimization (PGO) now works with OSR and is easier to enable (by adding `<TieredPGO>true</TieredPGO>` to your project file). PGO can also instrument and optimize additional things, such as delegates.
- Improved code generation for Arm64.
- Native AOT produces a standalone executable in the target platform's file format with no external dependencies. It's entirely native, with no IL or JIT, and provides fast startup time and a small, self-contained deployment. In .NET 7, Native AOT focuses on console apps and requires apps to be trimmed.
- Performance improvements to the Mono runtime, which powers Blazor WebAssembly, Android, and iOS apps.

For a detailed look at many of the performance-focused features that make .NET 7 so fast, see the Performance improvements in .NET 7 blog post.

System.Text.Json serialization

.NET 7 includes improvements to System.Text.Json serialization in the following areas:

- **Contract customization** gives you more control over how types are serialized and deserialized. For more information, see Customize a JSON contract.
- **Polymorphic serialization** for user-defined type hierarchies. For more information, see Serialize properties of derived classes.

- Support for **required members**, which are properties that must be present in the JSON payload for deserialization to succeed. For more information, see Required properties.

For information about these and other updates, see the What's new in System.Text.Json in .NET 7 blog post.

Generic math

.NET 7 and C# 11 include innovations that allow you to perform mathematical operations generically—that is, without having to know the exact type you're working with. For example, if you wanted to write a method that adds two numbers, previously you had to add an overload of the method for each type. Now you can write a single, generic method, where the type parameter is constrained to be a number-like type. For more information, see the Generic math article and the Generic math blog post.

Regular expressions

.NET's regular expression library has seen significant functional and performance improvements in .NET 7:

- The new option RegexOptions.NonBacktracking enables matching using an approach that avoids backtracking and guarantees linear-time processing in the length of the input. The nonbacktracking engine can't be used in a right-to-left search and has a few other restrictions, but is fast for all regular expressions and inputs. For more information, see Nonbacktracking mode.

- Regular expression source generators are new. Source generators build an engine that's optimized for *your* pattern at compile time, providing throughput performance benefits. The source that's emitted is part of your project, so you can view and debug it. In addition, a new source-generator diagnostic `SYSLIB1045` alerts you to places you use Regex that could be converted to the source generator. For more information, see .NET regular expression source generators.

- For case-insensitive searches, .NET 7 includes large performance gains. The gains come because specifying RegexOptions.IgnoreCase no longer calls ToLower on each character in the pattern and on each character in the input. Instead, all casing-related work is done when the Regex is constructed.

- Regex now supports spans for some APIs. The following new methods have been added as part of this support:
  - Regex.EnumerateMatches
  - Regex.Count
  - Regex.IsMatch(ReadOnlySpan<Char>) (and a few other overloads)

For more information about these and other improvements, see the Regular expression improvements in .NET 7 blog post.

.NET libraries

Many improvements have been made to .NET library APIs. Some are mentioned in other, dedicated sections of this article. Some others are summarized in the following table.

Description	APIs	Further information
Support for microseconds and nanoseconds in TimeSpan, TimeOnly, DateTime, and DateTimeOffset types	- DateTime.Microsecond - DateTime.Nanosecond - DateTime.AddMicroseconds(Double) - New DateTime constructor overloads - DateTimeOffset.Microsecond - DateTimeOffset.Nanosecond - DateTimeOffset.AddMicroseconds(Double) - New DateTimeOffset constructor overloads - TimeOnly.Microsecond - TimeOnly.Nanosecond - TimeSpan.Microseconds - TimeSpan.Nanoseconds - TimeSpan.FromMicroseconds(Double) - And others...	These APIs mean you no longer have to perform computations on the "tick" value to determine microsecond and nanosecond values. For more information, see the .NET 7 Preview 4 blog post.
APIs for reading, writing, archiving, and extracting Tar archives	System.Formats.Tar	For more information, see the .NET 7 Preview 4 and .NET 7 Preview 6 blog posts.
Rate limiting APIs to protect a resource by keeping traffic at a safe level	RateLimiter and others in the System.Threading.RateLimiting NuGet package	For more information, see Rate limit an HTTP handler in .NET and Announcing rate limiting for .NET.
APIs to read *all* the data from a Stream	- Stream.ReadExactly - Stream.ReadAtLeast	Stream.Read may return less data than what's available in the stream. The new `ReadExactly` methods read *exactly* the number of bytes requested, and the new `ReadAtLeast` methods read *at least* the number of bytes requested. For more information, see the .NET 7 Preview 5 blog post.

Description	APIs	Further information
New type converters for `DateOnly`, `TimeOnly`, `Int128`, `UInt128`, and `Half`	In the System.ComponentModel namespace: - DateOnlyConverter - TimeOnlyConverter - Int128Converter - UInt128Converter - HalfConverter	Type converters are often used to convert value types to and from a string. These new APIs add type converters for types that were added more recently.
Metrics support for IMemoryCache	- MemoryCacheStatistics - MemoryCache.GetCurrentStatistics()	GetCurrentStatistics() lets you use event counters or metrics APIs to track statistics for one or more memory caches. For more information, see the .NET 7 Preview 4 blog post.
APIs to get and set Unix file permissions	- System.IO.UnixFileMode enum - File.GetUnixFileMode - File.SetUnixFileMode - FileSystemInfo.UnixFileMode - Directory.CreateDirectory(String, UnixFileMode) - FileStreamOptions.UnixCreateMode	For more information, see the .NET 7 Preview 7 blog post.
Attribute to indicate what kind of syntax is expected in a string	StringSyntaxAttribute	For example, you can specify that a `string` parameter expects a regular expression by attributing the parameter with `[StringSyntax(StringSyntaxAttribute.Regex)]`.
APIs to interop with JavaScript when running in the browser or other WebAssembly architectures	System.Runtime.InteropServices.JavaScript	JavaScript apps can use the expanded WebAssembly support in .NET 7 to reuse .NET libraries from JavaScript. For more information, see Use .NET from any JavaScript app in .NET 7.

Observability

.NET 7 makes improvements to *observability*. Observability helps you understand the state of your app as it scales and as the technical complexity increases. .NET's observability implementation is primarily built around OpenTelemetry. Improvements include:

- The new Activity.CurrentChanged event, which you can use to detect when the span context of a managed thread changes.
- New, performant enumerator methods for Activity properties: EnumerateTagObjects(), EnumerateLinks(), and EnumerateEvents().

For more information, see the .NET 7 Preview 4 blog post.

.NET SDK

The .NET 7 SDK improves the CLI template experience. It also enables publishing to containers, and central package management with NuGet.

Templates

Some welcome improvements have been made to the `dotnet new` command and to template authoring.

dotnet new

The dotnet new CLI command, which creates a new project, configuration file, or solution based on a template, now supports tab completion for exploring:

- Available template names
- Template options
- Allowable option values

In addition, for better conformity, the `install`, `uninstall`, `search`, `list`, and `update` subcommands no longer have the `--` prefix.

Authoring

Template *constraints*, a new concept for .NET 7, let you define the context in which your templates are allowed. Constraints help the template engine determine which templates it should show in commands like `dotnet new list`. You can constrain your template to an operating system, a template engine host (for example, the .NET CLI or New Project dialog in Visual Studio), and an installed workload. You define constraints in your template's configuration file.

Also in the template configuration file, you can now annotate a template parameter as allowing multiple values. For example, the web template allows multiple forms of authentication.

For more information, see the .NET 7 Preview 6 blog post.

Publish to a container

Containers are one of the easiest ways to distribute and run a wide variety of applications and services in the cloud. Container images are now a supported output type of the .NET SDK, and you can create containerized versions of your applications using dotnet publish. For more information about the feature, see Announcing built-in container support for the .NET SDK. For a tutorial, see Containerize a .NET app with dotnet publish.

Central package management

You can now manage common dependencies in your projects from one location using NuGet's central package management (CPM) feature. To enable it, you add a *Directory.Packages.props* file to the root of your repository. In this file, set the MSBuild property `ManagePackageVersionsCentrally` to `true` and add versions for common package dependency using `PackageVersion` items. Then, in the individual project files, you can omit `Version` attributes from any PackageReference items that refer to centrally managed packages.

For more information, see Central package management.

P/Invoke source generation

.NET 7 introduces a source generator for platform invokes (P/Invokes) in C#. The source generator looks for LibraryImportAttribute on `static`, `partial` methods to trigger compile-time source generation of marshalling code. By generating the marshalling code at compile time, no IL stub needs to be generated at run time, as it does when using DllImportAttribute. The source generator improves application performance and also allows the app to be ahead-of-time (AOT) compiled. For more information, see Source generation for platform invokes and Use custom marshallers in source-generated P/Invokes.

Related releases

This section contains information about related products that have releases that coincide with the .NET 7 release.

Visual Studio 2022 version 17.4

For more information, see What's new in Visual Studio 2022.

C# 11

C# 11 includes support for generic math, raw string literals, file-scoped types, and other new features. For more information, see What's new in C# 11.

F# 7

F# 7 continues the journey to make the language simpler and improve performance and interop with new C# features. For more information, see Announcing F# 7.

.NET MAUI

.NET Multi-platform App UI (.NET MAUI) is a cross-platform framework for creating native mobile and desktop apps with C# and XAML. It unifies Android, iOS, macOS, and Windows APIs into a single API. For information about the latest updates, see What's new in .NET MAUI for .NET 7.

ASP.NET Core

ASP.NET Core 7.0 includes rate-limiting middleware, improvements to minimal APIs, and gRPC JSON transcoding. For information about all the updates, see What's new in ASP.NET Core 7.

EF Core

Entity Framework Core 7.0 includes provider-agnostic support for JSON columns, improved performance for saving changes, and custom reverse engineering templates. For information about all the updates, see What's new in EF Core 7.0.

Windows Forms

Much work has gone into Windows Forms for .NET 7. Improvements have been made in the following areas:

- Accessibility
- High DPI and scaling
- Databinding

For more information, see What's new in Windows Forms in .NET 7.

WPF

WPF in .NET 7 includes numerous bug fixes as well as performance and accessibility improvements. For more information, see the What's new for WPF in .NET 7 blog post.

Orleans

Orleans is a cross-platform framework for building robust, scalable distributed applications. For information about the latest updates for Orleans, see Migrate from Orleans 3.x to 7.0.

.NET Upgrade Assistant and CoreWCF

The .NET Upgrade Assistant now supports upgrading server-side WCF apps to CoreWCF, which is a community-created port of WCF to .NET (Core). For more information, see Upgrade a WCF server-side project to use CoreWCF.

ML.NET

ML.NET now includes a text classification API that makes it easy to train custom text classification models using the latest state-of-the-art deep learning techniques. For more information, see the What's new with AutoML and tooling and Introducing the ML.NET Text Classification API blog posts.

See also

- Release notes for .NET 7