Agenda: About ASP.Net Core

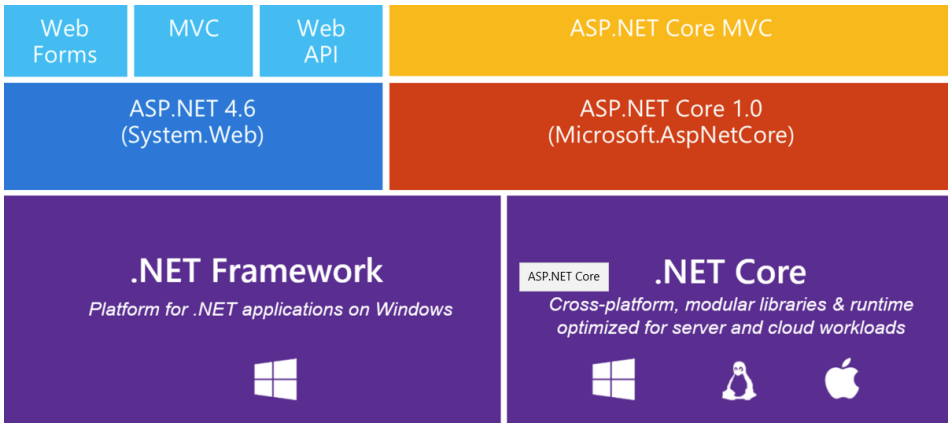
- Introduction to ASP.NET Core
- Advantages of ASP.NET Core
- ASP.NET Core vs ASP.NET
- New Features of ASP.NET Core

Introduction to ASP.NET Core

- ASP.NET Core is a **cross-platform, high-performance, open-source** framework for building modern, cloud-based, Internet-connected applications.
- Some people think that many things remain the same, but this is not entirely true. ASP.NET Core is a big fundamental change to the ASP.NET landscape.

With ASP.NET Core, you can:

- Build web apps and services, IoT apps, and mobile backends.
- Use your favorite development tools on Windows, macOS, and Linux.
- Deploy to the cloud or on-premises.
- Run on .NET Core or .NET Framework.



With ASP.NET Core, you can:

- Build web apps and services, IoT apps, and mobile backends.
- Use your favorite development tools on Windows, macOS, and Linux.
- Deploy to the cloud or on-premises.
- Run on .NET Core or .NET Framework.

ASP.NET Core ships entirely as NuGet packages. Using NuGet packages allows you to optimize your app to include only the necessary dependencies. In fact, ASP.NET Core 2.x apps targeting .NET Core only require a single NuGet package. The benefits of a smaller app surface area include tighter security, reduced servicing, and improved performance.

Advantages of ASP.NET Core

The main design principle in Asp.Net core is to improve the **development experience and to optimize performance**.

- **Unified framework** for MVC, Web API and SignalR.
- **Cross platform** support (build & run on Windows, Linux and MAC).
- Integration of modern, **client-side frameworks** and development workflows.
- Modular libraries & runtime optimized for the **server and cloud** workloads.
- Built-in **dependency injection**.
- Performance improvement and much **faster** than the old versions.
- Ability to **host** on IIS, Nginx, Apache, Docker, or self-host in your own process.
- Side-by-side app **versioning** when targeting .NET Core.
- Full support of **NuGet** Package Manager.
- Automatically **recompile** the application.
- Light weight and **fast development** cycle.

ASP.NET Core vs ASP.NET

ASP.NET Framework	ASP.NET Core
Build for Windows	Build for Windows, macOS, or Linux
Use .NET Framework runtime	Choose .NET Framework or .NET Core runtime
Web Server: Only IIS	Web Server: Kestrel, IIS, Apache, Nginx, Docker Kestrel is cross-platform server bundled into .NET
Use Web Forms, SignalR, MVC, Web API, or Web Pages	Razor Pages is the recommended approach to create a Web UI as of ASP.NET Core 2.x. Also available MVC, Web API, and SignalR.
One version per machine	Multiple versions per machine
Develop with Visual Studio using C#, VB, or F#	Develop with Visual Studio, Visual Studio for Mac, or Visual Studio Code using C# or F#
Good performance System.Web. Everything is included by default	Higher performance than ASP.NET No System.Web. Everything is Nuget packages. Excluded all by default
Web.config	.json, .ini, environment variables, etc.
ASPX, ASMX, Global.asax, AppDomain	Not Supported

New Features of ASP.NET Core

- Razor Pages
- URL Rewriting Middleware
- Response Caching Middleware
- View Components as Tag Helpers
- Middleware as MVC filters
- Cookie-based TempData provider

- Azure App Service logging provider
- Azure Key Vault configuration provider
- Azure and Redis Storage Data Protection Key Repositories
- WebListener Server for Windows
- WebSockets support