# ASP.NET

**ASP.NET** je open-source webový framework vyvinutý společností Microsoft pro tvorbu moderních webových aplikací a služeb. Je postaven na .NET Frameworku a umožňuje vývojářům vytvářet dynamické webové stránky, webové aplikace a RESTful API.

## Historie a vývoj

ASP.NET je nástupcem klasické technologie **ASP (Active Server Pages)**:

- **ASP.NET 1.0** (2002) – první verze jako součást .NET Frameworku 1.0
- **ASP.NET 2.0** (2005) – master pages, membership systém
- **ASP.NET 3.5** (2007) – zavedení ASP.NET MVC jako alternativa k Web Forms
- **ASP.NET 4.x** (2010-2015) – vylepšení výkonu, async/await
- **ASP.NET Core** (2016+) – kompletně přepsaný, multiplatformní framework

## Hlavní technologie ASP.NET

ASP.NET nabízí několik různých přístupů k vývoji webových aplikací:

### ASP.NET Web Forms

Tradiční model vývoje podobný desktop aplikacím:

- **Event-driven** programovací model
- **Drag-and-drop** ovládací prvky ve Visual Studiu
- **ViewState** – automatická správa stavu
- **Server controls** – tlačítka, textboxy atd. s událostmi
- Vhodné pro rychlý vývoj aplikací s formuláři

```
<%@ Page Language="C#" %>
<!DOCTYPE html>
<html>
<head>
    <title>Příklad Web Forms</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Label ID="lblMessage" runat="server" Text="Vítejte!" />
        <asp:Button ID="btnClick" runat="server" Text="Klikni"
OnClick="btnClick_Click" />
    </form>
</body>
</html>
```

## ASP.NET MVC

Model-View-Controller pattern pro lepší separaci logiky:

- **Model** – datové objekty a business logika
- **View** – prezentační vrstva (Razor syntax)
- **Controller** – zpracování požadavků a koordinace
- Plná kontrola nad HTML výstupem
- Lepší testovatelnost kódu
- RESTful routing

```
// Controller
public class HomeController : Controller
{
    public IActionResult Index()
    {
        var model = new HomeViewModel
        {
            Message = "Vítejte v ASP.NET MVC!"
        };
        return View(model);
    }
}
```

```
@* View (Razor) *@
@model HomeViewModel

<h1>@Model.Message</h1>
<p>Toto je stránka vytvořená v ASP.NET MVC.</p>
```

## ASP.NET Web API

Framework pro tvorbu RESTful webových služeb:

- Vytváření HTTP služeb dostupných pro různé klienty
- Automatická serializace do JSON/XML
- Podpora pro content negotiation
- RESTful routing

```
[ApiController]
[Route("api/[controller]")]
public class ProductsController : ControllerBase
{
    [HttpGet]
    public IEnumerable<Product> GetAll()
    {
        return _repository.GetAllProducts();
    }
```

```
    [HttpGet("{id}")]
    public ActionResult<Product> GetById(int id)
    {
        var product = _repository.GetProduct(id);
        if (product == null)
            return NotFound();
        return product;
    }

    [HttpPost]
    public ActionResult<Product> Create(Product product)
    {
        _repository.AddProduct(product);
        return CreatedAtAction(nameof(GetById),
            new { id = product.Id }, product);
    }
}
```

### ASP.NET Web Pages

Zjednodušený model pro menší webové projekty:

- Kombinace HTML a kódu v jednom souboru
- Razor syntax pro vkládání C# kódu
- Jednodušší než Web Forms nebo MVC
- Vhodné pro jednoduché weby a prototypy

## Razor syntaxe

**Razor** je šablonovací engine pro ASP.NET:

```
@{
    var jmeno = "Jan";
    var cas = DateTime.Now;
}

<h1>Ahoj, @jmeno!</h1>
<p>Aktuální čas: @cas.ToString("HH:mm:ss")</p>

@if (cas.Hour < 12)
{
    <p>Dobré ráno!</p>
}
else
{
    <p>Dobrý den!</p>
}
```

```
<ul>
@foreach (var item in Model.Items)
{
    <li>@item.Name - @item.Price Kč</li>
}
</ul>
```

Klíčové vlastnosti Razor:

- Symbol `@` pro přechod mezi HTML a C# kódem
- IntelliSense podpora ve Visual Studiu
- Automatické HTML kódování (ochrana proti XSS)
- Čistá a čitelná syntaxe

# Klíčové funkce ASP.NET

## Správa stavu

ASP.NET poskytuje několik mechanismů pro ukládání stavu:

- **ViewState** – stav mezi postbacky (Web Forms)
- **Session** – data specifická pro uživatele
- **Application** – data sdílená mezi všemi uživateli
- **Cache** – dočasné uložení dat pro výkon
- **Cookies** – data na straně klienta
- **TempData** – dočasná data mezi požadavky (MVC)

## Autentizace a autorizace

Integrované bezpečnostní funkce:

- **ASP.NET Identity** – správa uživatelů a rolí
- **Windows Authentication** – integrace s Active Directory
- **Forms Authentication** – vlastní přihlašovací formuláře
- **OAuth/OpenID Connect** – integrace s externími poskytovateli
- **JWT tokens** – pro API autentizaci

```
[Authorize(Roles = "Admin")]
public class AdminController : Controller
{
    public IActionResult Dashboard()
    {
        return View();
    }
}
```

### Validace dat

Podpora pro validaci vstupních dat:

```
public class RegisterViewModel
{
    [Required(ErrorMessage = "Email je povinný")]
    [EmailAddress(ErrorMessage = "Neplatný email")]
    public string Email { get; set; }

    [Required]
    [StringLength(100, MinimumLength = 6)]
    [DataType(DataType.Password)]
    public string Password { get; set; }

    [Compare("Password", ErrorMessage = "Hesla se neshodují")]
    public string ConfirmPassword { get; set; }
}
```

## ASP.NET Core

Moderní, multiplatformní verze ASP.NET:

- **Cross-platform** – běží na Windows, Linux, macOS
- **Open-source** – vyvíjeno veřejně na GitHubu
- **Vysoký výkon** – jeden z nejrychlejších webových frameworků
- **Modulární** – používáte jen co potřebujete
- **Cloud-ready** – optimalizováno pro kontejnery a cloud
- **Unified** – sloučení MVC a Web API do jednoho frameworku

Hlavní rozdíly oproti klasickému ASP.NET:

| Klasický ASP.NET | ASP.NET Core |
|---|---|
| Pouze Windows | Windows, Linux, macOS |
| Závislost na IIS | Samostatný webový server (Kestrel) |
| System.Web.dll | Modulární balíčky |
| Web.config | appsettings.json + kód |
| Těžký framework | Lehký a rychlý |

```
// Program.cs v ASP.NET Core
var builder = WebApplication.CreateBuilder(args);

// Přidání služeb
builder.Services.AddControllersWithViews();
builder.Services.AddDbContext<ApplicationDbContext>();

var app = builder.Build();
```

```
// Konfigurace HTTP pipeline
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

## Deployment a hosting

ASP.NET aplikace lze nasadit různými způsoby:

- **IIS (Internet Information Services)** – tradiční hosting na Windows Server
- **Azure App Service** – cloud hosting od Microsoftu
- **Docker kontejnery** – pro ASP.NET Core
- **Linux servery** – s Nginx nebo Apache (ASP.NET Core)
- **Self-hosted** – vlastní konzolová aplikace s webovým serverem

## Výhody ASP.NET

- **Integrace s .NET** – přístup ke všem knihovnám .NET ekosystému
- **Výkonnost** – kompilovaný kód, caching, async operace
- **Bezpečnost** – integrované ochrany proti CSRF, XSS, SQL injection
- **Škálovatelnost** – podpora pro load balancing a distribuované aplikace
- **Vývojářské nástroje** – vynikající podpora ve Visual Studio
- **Komunita** – rozsáhlá dokumentace a aktivní komunita
- **IntelliSense** – automatické dokončování kódu
- **NuGet** – tisíce dostupných balíčků

## Databázový přístup

ASP.NET podporuje různé způsoby práce s databázemi:

- **ADO.NET** – nízkoúrovňový přístup k databázím
- **Entity Framework** – ORM (Object-Relational Mapping)
- **Dapper** – lehký micro-ORM
- **LINQ to SQL** – dotazy pomocí LINQ syntaxe

```
// Entity Framework příklad
public class ApplicationDbContext : DbContext
{
    public DbSet<Product> Products { get; set; }
```

```
    public DbSet<Order> Orders { get; set; }
}

// Použití v controlleru
public class ProductsController : Controller
{
    private readonly ApplicationDbContext _context;

    public ProductsController(ApplicationDbContext context)
    {
        _context = context;
    }

    public async Task<IActionResult> Index()
    {
        var products = await _context.Products
            .Where(p => p.IsActive)
            .OrderBy(p => p.Name)
            .ToListAsync();
        return View(products);
    }
}
```

## SignalR

**ASP.NET SignalR** – knihovna pro real-time webové aplikace:

- **WebSockets** podpora
- Automatický fallback na jiné technologie (long polling, server-sent events)
- Obousměrná komunikace mezi serverem a klientem
- Vhodné pro chaty, notifikace, dashboardy

```
public class ChatHub : Hub
{
    public async Task SendMessage(string user, string message)
    {
        await Clients.All.SendAsync("ReceiveMessage", user, message);
    }
}
```

## Blazor

**Blazor** – framework pro tvorbu interaktivních webových UI s C#:

- **Blazor Server** – UI logika běží na serveru
- **Blazor WebAssembly** – C# kód běží přímo v prohlížeči
- Sdílení kódu mezi klientem a serverem
- Komponenty napsané v C# místo JavaScriptu

```
@page "/counter"

<h1>Counter</h1>

<p>Počet: @currentCount</p>

<button @onclick="IncrementCount">Klikni</button>

@code {
    private int currentCount = 0;

    private void IncrementCount()
    {
        currentCount++;
    }
}
```

## Middleware Pipeline

ASP.NET Core používá middleware pipeline pro zpracování požadavků:

```
app.Use(async (context, next) =>
{
    // Kód před dalším middleware
    await next.Invoke();
    // Kód po dalším middleware
});

app.UseAuthentication();
app.UseAuthorization();
app.UseStaticFiles();
app.UseRouting();
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();
});
```

## Výkon a optimalizace

Techniky pro optimalizaci ASP.NET aplikací:

- **Output caching** – cache celých stránek nebo částí
- **Response compression** – komprese HTTP odpovědí
- **Bundling a minification** – spojení a minimalizace CSS/JS
- **CDN** – distribuce statických souborů
- **Async/await** – asynchronní operace pro lepší škálovatelnost
- **Connection pooling** – opakované použití databázových připojení

## Související pojmy

- .NET Framework – základní platforma
- .NET Core – multiplatformní verze
- CLR – běhové prostředí
- CSharp – primární programovací jazyk
- IIS – Internet Information Services
- Visual Studio – vývojové prostředí
- Entity Framework – ORM framework
- NuGet – správce balíčků
- Razor – view engine
- MVC – architektonický vzor

## Externí odkazy

- Oficiální stránky ASP.NET
- ASP.NET Core dokumentace
- ASP.NET Core na GitHubu
- ASP.NET komunita