ASP.NET Core Web API

Construindo APIs Modernas e Performáticas

Guia Introdutório ao Desenvolvimento de Web APIs com ASP.NET Core

O que é ASP.NET Core Web API?

- **Framework para criação de APIs REST** e HTTP services
- **Parte do ecossistema ASP.NET Core**
- **Cross-platform**: Windows, Linux, macOS
- **High-performance** e escalável
- **Integração nativa** com dependency injection, logging, configuration

Características Principais

✅ Lightweight e modular
✅ Built-in JSON serialization
✅ Automatic model binding
✅ Comprehensive routing system

Por que ASP.NET Core Web API?

🚀 Performance

- Um dos frameworks mais rápidos
- Minimal overhead
- Async/await nativo

🔧 Produtividade

- Scaffolding automático
- Hot reload
- Rich tooling ecosystem

🛠️ Ferramentas

- Visual Studio
- Visual Studio Code
- JetBrains Rider

Evolução das Web APIs

```
public class ValuesController : ApiController
{
    public IEnumerable<string> Get()
    {
        return new string[] { "value1", "value2" };
    }
}
```

```
[ApiController]
[Route("api/[controller]")]
public class ValuesController : ControllerBase
{
    [HttpGet]
    public ActionResult<IEnumerable<string>> Get()
    {
        return Ok(new[] { "value1", "value2" });
    }
}
```

Minimal APIs vs Controller-based APIs

Minimal APIs (.NET 6+)

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

app.MapGet("/products",
    () => Products.GetAll());

app.MapPost("/products",
    (Product product) =>
        Products.Create(product));
app.Run();
```

Ideal para:

- APIs simples, microservices e Prototipagem rápida

Controller-based APIs

```
[ApiController]
[Route("api/[controller]")]
public class ProductsController
{
    [HttpGet]
    public IActionResult GetAll()
        => Ok(Products.GetAll());
    [HttpPost]
    public IActionResult Create(Product p)
        => CreatedAtAction(nameof(Get),
            new { id = p.Id }, p);
}
```

Ideal para: APIs complexas, Múltiplas actions e melhor organização

Estrutura de um Projeto Web API

```
MyWebAPI/
├── Controllers/           # API Controllers
├── Models/               # Data models/DTOs
├── Services/             # Business logic
├── Data/                 # Data access layer
├── Middleware/           # Custom middleware
├── Extensions/           # Extension methods
├── appsettings.json      # Configuration
└── Program.cs            # Application entry point
```

Principais Arquivos

- **Program.cs**: Configuração e startup
- **appsettings.json**: Configurações da aplicação
- **Controllers**: Endpoints da API

Exemplo Completo - Controller

```
[ApiController]
[Route("api/[controller]")]
[Produces("application/json")]
public class ProductsController : ControllerBase
{
    private readonly IProductService _productService;
    private readonly ILogger<ProductsController> _logger;

    public ProductsController(IProductService productService,
                              ILogger<ProductsController> logger)
    {
        _productService = productService;
        _logger = logger;
    }

    /// <summary>
    /// Retorna todos os produtos ativos
    /// </summary>
    [HttpGet]
    [ProducesResponseType(typeof(IEnumerable<ProductDto>), 200)]
    public async Task<ActionResult<IEnumerable<ProductDto>>> GetProducts(
        [FromQuery] int page = 1,
        [FromQuery] int pageSize = 10,
        [FromQuery] string search = "")
    {
        var products = await _productService.GetProductsAsync(page, pageSize, search);
        return Ok(products);
    }

    /// <summary>
    /// Retorna um produto específico
    /// </summary>
    [HttpGet("{id}")]
    [ProducesResponseType(typeof(ProductDto), 200)]
    [ProducesResponseType(404)]
    public async Task<ActionResult<ProductDto>> GetProduct(int id)
    {
        var product = await _productService.GetProductByIdAsync(id);
        return product == null ? NotFound() : Ok(product);
    }
}
```

Roadmap e Futuro

Novas Features (.NET 9+)

- **Native AOT support** melhorado
- **Minimal APIs** com mais recursos
- **gRPC improvements**
- **Cloud-native optimizations**

Tendências

- **GraphQL integration**
- **Real-time APIs** com SignalR
- **AI/ML integration**
- **Serverless optimizations**

Recursos para Aprender Mais

📚 Documentação Oficial

- ASP.NET Core Documentation
- Web API Tutorial
- EF Core Guide

🎥 Vídeos e Cursos

- Microsoft Learn
- Pluralsight ASP.NET Core Path
- YouTube: .NET Channel

🛠️ Ferramentas Úteis

- Postman/Insomnia (API testing)
- Swagger/OpenAPI (Documentation)
- EF Core Power Tools
- dotnet CLI templates

👥 Comunidade

- Stack Overflow
- Reddit: r/dotnet
- Discord: .NET Community
- GitHub Discussions

Conclusão

ASP.NET Core Web API é a escolha ideal para:

✨ APIs Modernas e Performáticas

- Framework maduro e bem documentado
- Performance excepcional
- Ecossistema rico e ativo
- Suporte empresarial robusto

🚀 Desenvolvimento Produtivo

- Tooling de primeira classe
- Integração nativa com Azure
- Padrões de mercado estabelecidos

Obrigado!

Perguntas?

Próximos passos:

- Crie sua primeira Web API
- Explore os templates do .NET CLI
- Pratique com projetos reais

Recursos úteis:

- dotnet new webapi
- ASP.NET Core Samples

Vamos construir APIs incríveis com ASP.NET Core! 🚀