ASP.NET Core Tutorial

Desenvolvimento Web Moderno

1. Introducao ao ASP.NET Core

ASP.NET Core e um framework web multiplataforma, open-source e de alto desempenho para criar aplicacoes web modernas.

Principais Recursos:

â¢ MVC (Model-View-Controller)
â¢ Web API para servicos REST
â¢ Razor Pages para aplicacoes web
â¢ SignalR para comunicacao em tempo real
â¢ Blazor para SPAs com C#

Arquitetura:

â¢ Host Web integrado (Kestrel)
â¢ Pipeline de middleware personalizavel
â¢ Injecao de dependencia nativa
â¢ Sistema de configuracao flexivel
â¢ Suporte a containerizacao

2. Criando seu Primeiro Projeto

Passo a Passo:

1. Abra o terminal/prompt de comando
2. Execute: dotnet new webapi -n MinhaPrimeiraAPI
3. Navegue para a pasta: cd MinhaPrimeiraAPI
4. Execute: dotnet run
5. Acesse: https://localhost:5001/swagger

Estrutura do Projeto:

â¢ Program.cs - Configuracao da aplicacao
â¢ Controllers/ - Controladores da API
â¢ Models/ - Modelos de dados
â¢ appsettings.json - Configuracoes
â¢ Properties/launchSettings.json - Configuracoes de execucao

Program.cs Explicado:

```
var builder = WebApplication.CreateBuilder(args);
builder.Services.AddControllers();
var app = builder.Build();
app.MapControllers();
app.Run();
```

3. Criando Controllers e Actions

Exemplo de Controller:

```
[ApiController]
[Route("api/[controller]")]
public class ProdutosController : ControllerBase
{
    private static List<Produto> produtos = new();

    [HttpGet]
    public ActionResult<IEnumerable<Produto>> Get()
    {
        return Ok(produtos);
    }

    [HttpPost]
    public ActionResult<Produto> Post(Produto produto)
    {
        produtos.Add(produto);
        return CreatedAtAction(nameof(Get), produto);
    }
}
```

HTTP Verbs Suportados:

â¢ GET - Recuperar dados

â¢ POST - Criar recursos

â¢ PUT - Atualizar recursos

â¢ DELETE - Remover recursos

â¢ PATCH - Atualizacao parcial

4. Middleware e Pipeline

O que e Middleware?

Middleware sao componentes que formam o pipeline de processamento de requisicoes HTTP.

Configurando Middleware:

```
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.UseCors();
app.MapControllers();
```

Middleware Personalizado:

```
app.Use(async (context, next) =>
{
    // Logica antes da proxima middleware
    await next(context);
    // Logica apos a proxima middleware
});
```

Ordem dos Middleware:

1. Exception Handling
2. HSTS
3. HTTPS Redirection
4. Static Files
5. Routing
6. Authentication
7. Authorization

5. Entity Framework Core

Configurando o DbContext:

```
public class AppDbContext : DbContext
{
    public DbSet<Produto> Produtos { get; set; }
    protected override void OnConfiguring()
        DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(connectionString);
    }
}
```

Registrando no Container DI:

```
builder.Services.AddDbContext<AppDbContext>(options =>
    options.UseSqlServer(connectionString));
```

Migrations:

â¢ dotnet ef migrations add InitialCreate

â¢ dotnet ef database update

CRUD Operations:

â¢ context.Produtos.Add(produto)

â¢ context.Produtos.Find(id)

â¢ context.Produtos.Update(produto)

â¢ context.Produtos.Remove(produto)

â¢ context.SaveChanges()