# Introduction to ASP.NET Core

By

Narasimha Rao T

***Microsoft.Net FSD Trainer***

Professional Development Trainer

tnrao.trainer@gmail.com

# 1. Introduction to ASP.NET Core

ASP.NET Core is a **cross-platform, open-source framework** developed by Microsoft for building **modern, cloud-ready, internet-connected applications**.

It's essentially the successor to **ASP.NET Framework**, designed to be **faster, modular, and cross-platform** (runs on Windows, Linux, and macOS).

## 2. What is ASP.NET Core?

- A web development framework used for building:
  - **Web APIs**
  - **Web applications** (MVC, Razor Pages, Blazor)
  - **Real-time apps** (SignalR)
  - **Microservices**
- Built on **.NET Core Runtime**.
- Supports modern features like **Dependency Injection**, **Configuration**, and **Middleware Pipeline**.

# 3. Why do we use ASP.NET Core?

| Reason | Benefit |
|---|---|
| **Cross-platform** | Works on Windows, macOS, Linux |
| **High performance** | Optimized for speed; one of the fastest web frameworks |
| **Unified model** | Single framework for Web APIs, MVC, and gRPC |
| **Built-in DI** | No need for extra libraries for dependency injection |
| **Cloud-ready** | Easily deployed to Azure, AWS, Docker |
| **Open-source** | Large community and free to use |

# 4. Different Types of Applications using ASP.NET Core

1. **Web Applications**
   - MVC (Model-View-Controller)
   - Razor Pages
2. **Web APIs**
   - REST APIs, gRPC services
3. **Real-time Applications**
   - Chat apps, live dashboards (SignalR)
4. **Microservices**
   - Distributed services communicating via HTTP

# 5. ASP.NET Core Project Structure

A typical ASP.NET Core project includes:

```
MyApp/
│
├── Controllers/       → API or MVC controllers
├── Models/            → Data models
├── Views/             → UI templates (MVC/Razor)
├── wwwroot/           → Static files (CSS, JS, images)
├── Program.cs         → Application entry point
├── appsettings.json   → Configuration settings
├── Properties/        → Launch settings
└── MyApp.csproj       → Project file
```

## 6. Program.cs File

Here’s a minimal **ASP.NET Core Web API**:

```
var builder = WebApplication.CreateBuilder(args);

// Add services
builder.Services.AddControllers();

var app = builder.Build();

// Configure middleware
app.UseHttpsRedirection();
app.UseAuthorization();

app.MapControllers();

app.Run();
```

# 7. Program.cs File Significance

- **Entry point** of the ASP.NET Core app.
- Creates **WebApplicationBuilder** → configures services & app.
- Builds the **WebApplication**.
- Configures the **HTTP request pipeline**.
- Starts the application.

Example minimal `Program.cs` :

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

## 8. Middleware Pipeline

**Middleware** are components that handle HTTP requests/responses in a pipeline. Executed in **the order they are added**.

Example:

```
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();
app.MapControllers();
```

Flow:

1. Request enters first middleware.
2. Passes to the next middleware if not handled.
3. Response flows back in reverse order.

# 9. Services in Program.cs

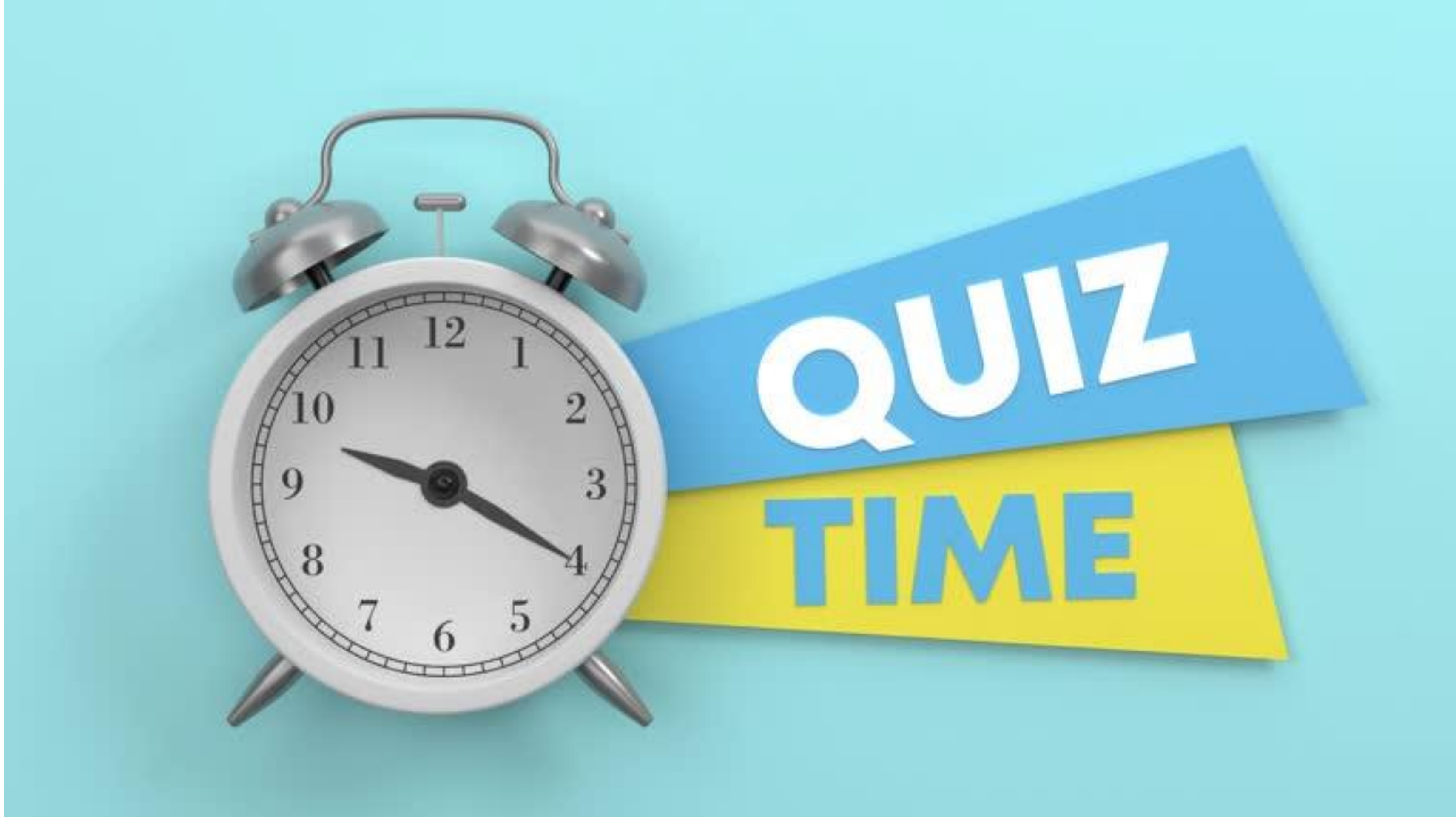
ASP.NET Core uses **Dependency Injection** to provide services.

Example:

```
builder.Services.AddControllers();
builder.Services.AddDbContext<MyDbContext>();
builder.Services.AddSingleton<IMyService, MyService>();
```

Service lifetimes:

- **Singleton** → One instance for the app's lifetime.
- **Scoped** → One instance per HTTP request.
- **Transient** → New instance every time it's requested.


QUIZ
TIME

1. **What is ASP.NET Core, and how is it different from ASP.NET Framework?**
2. **What are the key features of ASP.NET Core?**
3. **Explain the concept of cross-platform support in ASP.NET Core.**
4. **What is the difference between Kestrel and IIS in ASP.NET Core?**
5. **What is the purpose of the Program.cs file in an ASP.NET Core project?**
6. **What is the MVC pattern, and why is it used?**
7. **Explain the role of Model, View, and Controller in ASP.NET Core MVC.**
8. **What is the Razor View Engine, and how does it work?**
9. **What is an Action Method?**
10. **What are the different types of Action Results in ASP.NET Core MVC?**

# Q & A

Narasimha Rao T

tnrao.trainer@gmail.com