# ASP.NET Core Interview Questions & Answers

## What is the Differenece Between Asp.Net Framework and Asp.Net Core?

| **Feature** | **ASP.NET Framework** | **ASP.NET Core** |
| --- | --- | --- |
| **Platform Support** | Runs only on **Windows** | **Cross-platform** (Windows, Linux, macOS) |
| **Base Runtime** | Built on **.NET Framework** (monolithic) | Built on **.NET Core / .NET 5+** (modular) |
| **Hosting** | Works only with **IIS** | Runs on **Kestrel, IIS, Nginx, Docker, Cloud** |
| **Performance** | Slower, heavy, less optimized | High-performance, lightweight, cloud-ready |
| **Open Source** | Mostly closed-source | Fully **open-source** and community-driven |
| **Modern Features** | Limited support | Supports microservices, gRPC, minimal APIs |

## What is Tag Helper in ASP.NET Core?

A Tag Helper in ASP.NET Core allows you to use server-side code to generate and manage HTML elements in Razor views. It makes Razor pages look like normal HTML instead of mixing C# code.   
  
Example:  
Without Tag Helper: @Html.TextBoxFor(m => m.Name)  
With Tag Helper: <input asp-for="Name" />  
  
Benefits:  
- Cleaner HTML-like markup  
- IntelliSense support  
- Strongly typed binding  
- Easier to maintain

## What is TempData?

TempData is used to store data temporarily between two requests. It survives a redirect but is cleared once read. Useful for passing messages/notifications.  
  
Example:  
Controller1: TempData["Message"] = "Saved!";  
Controller2: var msg = TempData["Message"];  
  
Key points:  
- Uses session internally  
- Lives for one request  
- Good for alerts/messages

## What is ViewBag?

ViewBag is a dynamic object to pass data from controller to view during the current request only. Does not survive redirects.  
  
Example:  
Controller: ViewBag.Message = "Hello";  
View: @ViewBag.Message

## What is ViewData?

ViewData is a dictionary (key-value pairs) used to pass data from controller to view for the current request. Requires type casting.  
  
Example:  
Controller: ViewData["Message"] = "Hello";  
View: @ViewData["Message"]

## Difference between ViewData, ViewBag, and TempData

ViewData → Dictionary, needs casting, only current request.  
ViewBag → Dynamic wrapper, no casting, only current request.  
TempData → Uses session, survives one redirect, cleared after read.

## What is Partial View?

Partial View is a reusable portion of a view (UI component) that can be embedded inside other views. Used for headers, footers, forms, etc.  
  
Example:  
@Html.Partial("\_LoginPartial")

## Difference between Partial View and Layout View

Partial View → For small reusable sections (menu, form, footer).  
Layout View → Defines full page structure (like master page).

## What is Anti-Forgery Token and how to use it?

Anti-Forgery Token is used to prevent CSRF attacks. It generates a hidden token in forms and validates it on post.  
  
Example:  
View: @Html.AntiForgeryToken()  
Controller: [ValidateAntiForgeryToken]

## How to Manage Session in ASP.NET Core MVC?

Steps:  
1. Configure in Program.cs: AddSession()  
2. Enable middleware: app.UseSession()  
3. Use in controller: HttpContext.Session.SetString("User", "Krish")  
4. Retrieve: HttpContext.Session.GetString("User")  
5. Clear: HttpContext.Session.Clear()

## What is Dependency Injection (DI)?

DI is a design pattern where dependencies are provided to a class instead of creating them inside it. In ASP.NET Core, DI is built-in. Makes code loosely coupled and testable.  
  
Service lifetimes:  
- Transient: new instance every time  
- Scoped: one per request  
- Singleton: one for entire app

## What is Entity Framework Core?

EF Core is Microsoft’s ORM for .NET. It lets you work with the database using C# classes and LINQ instead of SQL queries. Supports multiple databases and migrations.  
  
Example:  
\_dbContext.Students.ToList(); // LINQ → SQL

## Difference between IEnumerable and IQueryable

IEnumerable → Works in memory, queries executed client-side, loads all data first.  
IQueryable → Works with database, queries translated into SQL, executed server-side.  
  
Use IQueryable for large datasets, IEnumerable for in-memory collections.

## What is Middleware in ASP.NET Core?

Middleware is software in the request pipeline that handles requests and responses. Each middleware can run code before/after the next middleware.  
  
Examples: app.UseRouting(), app.UseAuthentication(), app.UseStaticFiles()

## What is IActionResult and ActionResult<T>?

IActionResult → Interface representing different action results (View, Json, Redirect, etc.).  
ActionResult<T> → Generic type that allows returning a specific type (model) or standard results like NotFound(). Mostly used in APIs for strong typing.

## What is Repository Pattern?

Repository Pattern is a design pattern that separates data access logic from business logic. It acts as an abstraction layer between database and application, making code clean, testable, and maintainable.

## Difference between Synchronous and Asynchronous

Synchronous → Tasks run one after another, blocking until complete.  
Asynchronous → Tasks don’t block, thread can do other work while waiting.  
  
Synchronous is like waiting in line; Asynchronous is like taking a token and doing other work until your turn.

### What is Tag Helper in ASP.NET Core?

A Tag Helper in ASP.NET Core allows you to use server-side code to generate and manage HTML elements in Razor views. It makes Razor pages look like normal HTML instead of mixing C# code.

Example: Without Tag Helper: @Html.TextBoxFor(m => m.Name) With Tag Helper:

Benefits: - Cleaner HTML-like markup - IntelliSense support - Strongly typed binding - Easier to maintain

### What is async/await in C#?

**Answer:** async and await are keywords in C# for asynchronous programming. - async marks a method as asynchronous (usually returns Task or Task<T>).  
- await pauses method execution until the awaited task completes, without blocking the thread.

**Example:**

public async Task<string> GetDataAsync()  
{  
 HttpClient client = new HttpClient();  
 string result = await client.GetStringAsync("https://example.com");  
 return result;  
}

**Use:** Improves responsiveness (UI doesn’t freeze during long tasks).

### What are the filters in ASP.NET Core?

**Answer:**

Filters allow custom code to run before or after certain pipeline stages.

Types:

1. **Authorization Filters** → security checks (e.g., [Authorize]).

2. **Resource Filters** → caching, resource setup.

3. **Action Filters** → pre/post logic around action execution.

4. **Exception Filters** → handle unhandled errors.

5. **Result Filters** → run before/after the action result executes.

**Use:** Handle cross-cutting concerns (logging, error handling, caching, security).

### What is appsettings.json used for?

**Answer:** appsettings.json is a configuration file used to store application settings like connection strings, logging, API keys, etc.

**Example:**

{  
 "ConnectionStrings": {  
 "DefaultConnection": "Server=.;Database=MyDb;Trusted\_Connection=True;"  
 },  
 "AppSettings": {  
 "JwtSecret": "my-secret-key"  
 }  
}

Access via IConfiguration.

**Use:** Centralizes config, supports environment-specific files, allows strong typed binding.

### What is ASP.NET Core Web API?

**Answer:** ASP.NET Core Web API is a framework for building HTTP-based RESTful services. It is cross-platform, lightweight, and high-performance.

**Features:** - Supports JSON by default. - Uses HTTP verbs for CRUD (GET, POST, PUT, DELETE). - Built-in Dependency Injection. - Middleware-based pipeline. - Swagger/OpenAPI support for documentation.

**Use:** To build APIs consumed by web apps, mobile apps, microservices.

### What is Model Binding in ASP.NET Core?

**Answer:** Model Binding automatically maps data from HTTP requests (query string, route values, form, headers, body) to action method parameters or model objects.

**Example:**

public IActionResult Create(User user)  
{  
 return Ok(user);  
}

Request JSON { "Id":1, "Name":"John" } will bind to User.

**Use:** Avoids manual parsing, supports validation, reduces boilerplate.

### What is Swagger and why is it used?

**Answer:** Swagger is a tool for API documentation and testing (via OpenAPI spec). In ASP.NET Core, integrated using Swashbuckle.

**Uses:** - Provides interactive UI to explore/test endpoints. - Auto-generates documentation. - Shares clear API contract with front-end teams.

**Setup:** Add AddSwaggerGen() in Program.cs and enable via app.UseSwagger(); app.UseSwaggerUI();.

### What is JWT Authentication?

**Answer:** JWT (JSON Web Token) Authentication is a stateless authentication mechanism where server issues a signed token after login. Clients send this token in headers for subsequent requests.

**Structure:** Header + Payload (claims like userId, role, expiry) + Signature.

**Use:** Secure APIs, enable stateless auth, support mobile/web/microservices.

**Example:**

Authorization: Bearer <jwt-token>

### How do you version an ASP.NET Core Web API?

**Answer:** API versioning ensures backward compatibility while introducing new features. Done using Microsoft.AspNetCore.Mvc.Versioning.

**Strategies:** 1. URL Path: /api/v1/products 2. Query String: /api/products?api-version=1.0 3. Header: api-version: 1.0 4. Media Type: Accept: application/json; version=1.0

**Setup:**

builder.Services.AddApiVersioning(options =>  
{  
 options.DefaultApiVersion = new ApiVersion(1, 0);  
 options.AssumeDefaultVersionWhenUnspecified = true;  
 options.ReportApiVersions = true;  
});

### What is CORS, why do we use it, and how to configure it?

**Answer:** CORS (Cross-Origin Resource Sharing) allows a web app hosted on one domain to access resources from another domain. Browsers block cross-origin requests by default (Same-Origin Policy).

**Why:** Needed when frontend (e.g., Angular/React) calls APIs from a different domain/port.

**Setup in Program.cs:**

builder.Services.AddCors(options =>  
{  
 options.AddPolicy("AllowSpecificOrigin",  
 policy => policy.WithOrigins("http://localhost:4200", "https://myshop.com")  
 .AllowAnyHeader()  
 .AllowAnyMethod());  
});  
  
app.UseCors("AllowSpecificOrigin");

**Use:** Securely allow only trusted domains to access APIs.

### What is Global.asax in ASP.Net Framework?

Answer:

In **ASP.NET Framework**, Global.asax was used to handle application-level events like Application\_Start, Session\_Start, etc.

### What is Route.config in ASP.Net Framework?

Answer :

In **ASP.NET Framework (MVC)**, RouteConfig.cs was used to define URL routing rules.

### What is CLR , CLS , And CTS?

**CLR (Common Language Runtime):**  
It’s the execution engine of .NET — manages memory, executes code, handles garbage collection, exceptions, and security.

**CLS (Common Language Specification):**  
A set of rules that all .NET languages must follow to ensure cross-language interoperability.

**CTS (Common Type System):**  
Defines how data types are declared and used in .NET so that all languages share the same type system (e.g., int in C# = System.Int32 in IL).

### What is Partial Class in asp.Net?

Answer:

A partial class in .NET allows a class to be split across multiple files. At compile time, all parts are combined into a single class. It’s useful for separating auto-generated code from developer code or organizing large classes.”

### What is Sealed keyword?

Answer :

Used to prevent Inheritance for sealed class

Ex. Public sealed class User{}

### What is Readonly, Const and Static ?

Answer:

const is a compile-time constant,

readonly is a runtime constant that can be set in the constructor,

static belongs to the class, shared by all instances.

### How to Create a Object of Static Class?

Answer:

You cannot create an object of a static class; its members are accessed directly using the class name.

### Can we Create a Non-Static Methods in Static Class?

Answer :

No, a static class can only have static members; non-static methods are not allowed.

### What is the Connection-Oriented and Connection-Less Connection in Ado.Net?

**Connection-Oriented Communication :**

A type of database/network communication where a **dedicated connection** is established between the client and server before data transfer.

**Characteristics:**

1. Reliable data transfer.
2. Connection must be established before queries are executed.
3. Suitable for **transactions** where integrity is important.

Example :

SqlConnection conn = new SqlConnection(connectionString);

conn.Open();

// Execute SQL commands

conn.Close();

**Connection-Less Communication :**

A type of communication where queries are sent **without establishing a dedicated connection**.

**Characteristics:**

1. Faster because there is no connection overhead.
2. Less reliable (no session/transaction guarantees).
3. Suitable for **simple queries or stateless operations**.

**Difference :**

| **Feature** | **Connection-Oriented** | **Connection-Less** |
| --- | --- | --- |
| **Definition** | Dedicated connection established before communication | No dedicated connection; queries sent directly |
| **Reliability** | Reliable, supports transactions | Less reliable, no guarantee of delivery |
| **Overhead** | Higher (connection setup required) | Lower (no connection setup) |
| **Use Case** | Banking transactions, ACID operations | Simple queries, reporting, stateless operations |
| **Example** | JDBC, ODBC, ADO.NET connections | REST API calls executing SQL remotely |