ASP.NET Core Interview Questions & Answers

### 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 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?

* **IActionResult** → Interface representing different action results (View, Json, Redirect, etc.).
* **ActionResult** → 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.

Example: Synchronous is like waiting in line; Asynchronous is like taking a token and doing other work until your turn.

### 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.