# 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: <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.