নিশ্চিতই! নিচে ASP.NET Core এর কিছু সাধারণ ইন্টারভিউ প্রশ্ন এবং উত্তর বাংলায় দেওয়া হলো যা তোমার প্রস্তুতিতে সাহায্য করবে:

**১. ASP.NET Core কি?**

**উত্তর:**  
ASP.NET Core হলো একটি ওপেন সোর্স, ক্রস-প্ল্যাটফর্ম ফ্রেমওয়ার্ক যা দিয়ে ওয়েব অ্যাপ্লিকেশন, API এবং মাইক্রোসার্ভিস তৈরি করা যায়। এটি Microsoft দ্বারা তৈরি এবং ASP.NET এর পরবর্তী সংস্করণ। এটি Windows, Linux, এবং macOS এ চলতে পারে।

**২. ASP.NET Core এবং ASP.NET MVC এর মধ্যে পার্থক্য কি?**

**উত্তর:**

* ASP.NET Core হলো নতুন এবং ক্রস-প্ল্যাটফর্ম।
* ASP.NET MVC শুধুমাত্র Windows-এ চলে।
* ASP.NET Core অনেক হালকা ও দ্রুত।
* ASP.NET Core এ ইনবিল্ট ডিপেন্ডেন্সি ইনজেকশন রয়েছে।
* ASP.NET Core এ Configuration, Logging, Middleware ইত্যাদি অনেক উন্নত হয়েছে।

**৩. Middleware কি এবং ASP.NET Core এ এর ব্যবহার কী?**

**উত্তর:**  
Middleware হলো সফটওয়্যারের একটি অংশ যা HTTP রিকোয়েস্ট এবং রেসপন্স এর মধ্যবর্তী ধাপে কাজ করে। ASP.NET Core এ Middleware ব্যবহার করে রিকোয়েস্ট প্রসেসিং পাইপলাইন তৈরি করা হয়, যেমন Authentication, Logging, Error Handling ইত্যাদি।

**৪. Dependency Injection কি? ASP.NET Core এ কিভাবে কাজ করে?**

**উত্তর:**  
Dependency Injection (DI) হলো একটি ডিজাইন প্যাটার্ন যা ক্লাসের ডিপেন্ডেন্সি বাইরে থেকে সরবরাহ করে। ASP.NET Core এ DI বিল্ট-ইন আছে, যেখানে সার্ভিসগুলো ConfigureServices() মেথডে রেজিস্টার করে Constructor Injection ব্যবহার করে ক্লাসে ইনজেক্ট করা হয়।

**৫. Configuration কি? ASP.NET Core এ কিভাবে করা হয়?**

**উত্তর:**  
Configuration হলো অ্যাপ্লিকেশনের সেটিংস সংরক্ষণের প্রক্রিয়া। ASP.NET Core এ Configuration অনেক সোর্স থেকে নেয়া যায় যেমন appsettings.json, environment variables, command-line arguments ইত্যাদি। Configuration API ব্যবহার করে সহজেই অ্যাপ্লিকেশন কনফিগার করা যায়।

**৬. Razor Page কি?**

**উত্তর:**  
Razor Page হলো ASP.NET Core এর একটি ফিচার যা পেজ ভিত্তিক ওয়েব অ্যাপ্লিকেশন তৈরি করতে ব্যবহার হয়। এটি MVC এর থেকে সহজ এবং কম কোডে UI তৈরির সুবিধা দেয়। প্রতিটি Razor Page এ UI এবং কোড একসাথে থাকে।

**৭. ASP.NET Core এ Logging কিভাবে করা হয়?**

**উত্তর:**  
ASP.NET Core এ বিল্ট-ইন Logging API আছে। এটি Console, Debug, EventSource, এবং অন্যান্য Third-party লোগিং ফ্রেমওয়ার্কের সাথে কাজ করে। ILogger ইন্টারফেস ব্যবহার করে কোডে লগ মেসেজ লেখা হয়।

**৮. ASP.NET Core এ Session এবং Cookie এর মধ্যে পার্থক্য কী?**

**উত্তর:**

* Cookie ক্লায়েন্ট সাইডে ছোট ডাটা সংরক্ষণ করে, যা ব্রাউজারে থাকে।
* Session সার্ভার সাইডে ডাটা রাখে এবং ক্লায়েন্টকে একটি Session ID কুকি দেয়।
* Cookie সহজে এক্সেসযোগ্য, কিন্তু নিরাপত্তায় Session বেশি ভালো।

**৯. ASP.NET Core এ Authorization এবং Authentication এর মধ্যে পার্থক্য?**

**উত্তর:**

* Authentication হলো ইউজার কে নিশ্চিত করা, অর্থাৎ ইউজার কে সে বলছে।
* Authorization হলো ইউজার কোন রিসোর্স বা ফাংশনে অ্যাক্সেস পাবে কিনা তা নির্ধারণ করা।

**১০. ASP.NET Core এ Middleware কিভাবে যোগ করবেন?**

**উত্তর:**  
Startup.cs এর Configure() মেথডে app.UseMiddleware<YourMiddleware>(); লিখে Middleware যোগ করা হয়। এছাড়া app.UseAuthentication(); বা app.UseRouting(); এর মত বিল্ট-ইন Middleware গুলো ব্যবহার করা হয়।

নিশ্চিতভাবে! নিচে আরও বিস্তারিতভাবে ASP.NET Core ইন্টারভিউ প্রশ্ন এবং উত্তর বাংলায় ব্যাখ্যা করা হলো:

**🔹 ১. ASP.NET Core কী এবং কেন ব্যবহার করা হয়?**

**উত্তর:**  
ASP.NET Core একটি **উন্নত এবং আধুনিক ওয়েব ফ্রেমওয়ার্ক**, যা Microsoft দ্বারা তৈরি। এটি হলো ASP.NET Framework-এর পুনর্গঠিত সংস্করণ। এতে রয়েছে:

* **Cross-platform support** (Windows, Linux, macOS এ চলে)
* **High performance**
* **Modular architecture**
* **Cloud-ready configuration**
* **Built-in Dependency Injection**

👉 উদাহরণস্বরূপ, আপনি একটি REST API তৈরি করতে পারেন যা Windows server ছাড়াও Linux-based Docker container এ চালাতে পারবেন।

**🔹 ২. ASP.NET Core ও ASP.NET MVC-এর মধ্যে মূল পার্থক্য কী?**

| **ASP.NET MVC** | **ASP.NET Core** |
| --- | --- |
| শুধুমাত্র Windows এ চলে | Cross-platform (Windows, Linux) |
| System.Web নির্ভর | Modular ও Lightweight |
| Global.asax, web.config আছে | Middleware-based pipeline |
| DI নেই, করতে হয় external tool দিয়ে | Built-in Dependency Injection |

👉 সহজভাবে বললে, ASP.NET Core পুরনো ASP.NET-এর সীমাবদ্ধতাগুলো দূর করেছে এবং অনেক আধুনিক ফিচার এনেছে।

**🔹 ৩. Middleware কী এবং এটি কীভাবে কাজ করে?**

**উত্তর:**  
Middleware হলো একেকটি **কম্পোনেন্ট যা HTTP রিকোয়েস্ট প্রসেস করে**। প্রতিটি Middleware:

* রিকোয়েস্ট আসার পর কিছু কাজ করে (যেমন: লগিং, অথেন্টিকেশন)
* তারপর পরবর্তী Middleware-এ পাঠায়
* বা এখানেই রেসপন্স দিয়ে থামিয়ে দেয়

**Configure() method এর ভিতরে** middleware গুলো app.Use... বা app.Run... দিয়ে লেখা হয়।

public void Configure(IApplicationBuilder app)

{

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

**🔹 ৪. Dependency Injection (DI) কী এবং ASP.NET Core-এ কীভাবে কাজ করে?**

**উত্তর:**  
DI হলো এমন একটি প্রক্রিয়া যেখানে একটি ক্লাসের নির্ভরতা (অন্য ক্লাস বা সার্ভিস) বাইরে থেকে সরবরাহ করা হয়।

ASP.NET Core এ তিনটি লাইফটাইম আছে:

* **Transient** – প্রতি রিকোয়েস্টে নতুন ইনস্ট্যান্স
* **Scoped** – প্রতি HTTP রিকোয়েস্টে একটি ইনস্ট্যান্স
* **Singleton** – পুরো অ্যাপজুড়ে একটি ইনস্ট্যান্স

**Register করা হয়:**

services.AddScoped<IProductService, ProductService>();

**ব্যবহার করা হয়:**

public class HomeController : Controller

{

private readonly IProductService \_productService;

public HomeController(IProductService productService)

{

\_productService = productService;

}

}

**🔹 ৫. ASP.NET Core-এ Configuration কীভাবে কাজ করে?**

**উত্তর:**  
Configuration মানে অ্যাপ্লিকেশনের সেটিংস সংরক্ষণ ও ব্যবহার। যেমন: ডাটাবেজ কানেকশন স্ট্রিং, API Key ইত্যাদি।

ASP.NET Core এ Configuration সাধারণত নিচের ফাইলগুলো থেকে পড়ে:

* appsettings.json
* appsettings.Development.json
* Environment Variables
* Command Line

**উদাহরণ:**

// appsettings.json

{

"ConnectionStrings": {

"DefaultConnection": "Server=.;Database=AppDb;Trusted\_Connection=True;"

}

}

**ব্যবহার:**

var conn = Configuration.GetConnectionString("DefaultConnection");

**🔹 ৬. Razor Page কী এবং এটি MVC থেকে কীভাবে আলাদা?**

**উত্তর:**  
Razor Page হলো একটি **Page-based Programming Model**, যেখানে .cshtml ফাইলের সাথেই একটি কোডবিহাইন্ড ক্লাস (.cshtml.cs) থাকে।

**MVC:** Controller + View + Model  
**Razor Page:** Page (.cshtml) + PageModel (.cshtml.cs)

👉 Razor Page সহজ এবং ছোট প্রজেক্টের জন্য উপযুক্ত।

**🔹 ৭. Logging কীভাবে ASP.NET Core-এ কাজ করে?**

**উত্তর:**  
Logging মানে অ্যাপ্লিকেশনের চলার সময় বিভিন্ন ইনফরমেশন লগ করে রাখা। যেমন: কোন রিকোয়েস্ট এসেছে, কোন exception হয়েছে, ইত্যাদি।

ASP.NET Core-এ ILogger<T> ইন্টারফেস ব্যবহার করে লগ করা হয়:

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

\_logger.LogInformation("Index page requested.");

return View();

}

}

**🔹 ৮. Session এবং Cookie-এর মধ্যে পার্থক্য কী?**

| **Session** | **Cookie** |
| --- | --- |
| সার্ভার-সাইড | ক্লায়েন্ট-সাইড (ব্রাউজারে থাকে) |
| বেশি নিরাপদ | কম নিরাপদ |
| বড় ডেটা সংরক্ষণ করা যায় | সীমিত ডেটা (৪ কেবি) রাখা যায় |
| Server RAM এ জায়গা লাগে | ক্লায়েন্টে জায়গা লাগে |

**🔹 ৯. Authentication এবং Authorization এর মধ্যে পার্থক্য কী?**

| **Authentication** | **Authorization** |
| --- | --- |
| ইউজার কে সে, তা যাচাই করা | ইউজার কী করতে পারবে তা নির্ধারণ করা |
| উদাহরণ: Login | উদাহরণ: Admin-only পেজে অ্যাক্সেস অনুমোদন |
| আগে সম্পন্ন হয় | Authentication এর পরে সম্পন্ন হয় |

ASP.NET Core-এ JWT, Cookies, এবং External Provider (Google, Facebook) দিয়ে Authentication করা যায়।

**🔹 ১০. ASP.NET Core-এ Routing কীভাবে কাজ করে?**

**উত্তর:**  
Routing এর মাধ্যমে নির্ধারণ করা হয়, কোন URL এর জন্য কোন Controller/Action চলবে।

**Attribute Routing:**

[Route("api/products")]

public class ProductsController : Controller

{

[HttpGet("{id}")]

public IActionResult GetProduct(int id)

{

// Return product

}

}

**Conventional Routing:**

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

অসাধারণ! এখন আমরা **Advanced Level ASP.NET Core Interview Questions and Answers** বাংলায় বিস্তারিত ব্যাখ্যা করব, যা একজন **Lead Software Engineer** বা **Senior Developer** ইন্টারভিউতে জিজ্ঞেস করতে পারে। প্রতিটি প্রশ্নের সাথে প্রকৃত বাস্তব প্রজেক্ট ভিত্তিক ব্যাখ্যা দেওয়া হয়েছে।

**🔹 ১. Custom Middleware কীভাবে তৈরি ও ব্যবহার করবেন?**

**উত্তর:**  
Custom Middleware তৈরি করলে আপনি নিজের মতো করে HTTP pipeline-এ কাজ করতে পারেন। এটি রিকোয়েস্ট/রেসপন্স এর মাঝে এক্সট্রা প্রসেসিং বা লজিক বসানোর সুযোগ দেয়।

**➤ Custom Middleware ক্লাস:**

public class CustomLoggingMiddleware

{

private readonly RequestDelegate \_next;

public CustomLoggingMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task Invoke(HttpContext context)

{

Console.WriteLine("👉 Request Path: " + context.Request.Path);

await \_next(context);

Console.WriteLine("✅ Response Status: " + context.Response.StatusCode);

}

}

**➤ Middleware রেজিস্টার করা:**

public void Configure(IApplicationBuilder app)

{

app.UseMiddleware<CustomLoggingMiddleware>();

}

📌 **Interview Tip:** আপনি বলতে পারেন, “আমরা Custom Middleware ব্যবহার করি Audit Trail বা Request Timing হিসেব করতে।”

**🔹 ২. Filters গুলো কি? কত ধরনের ASP.NET Core Filter আছে?**

**উত্তর:**  
Filters হলো এমন কম্পোনেন্ট যা ASP.NET Core pipeline-এ **কোনো অ্যাকশন এক্সিকিউট হওয়ার আগে বা পরে কাজ করে।**

**➤ Filters এর ধরন:**

1. **Authorization Filter** – ইউজার অথরাইজ কিনা চেক করে
2. **Resource Filter** – এক্সিকিউশন এর আগে/পর Resource হ্যান্ডেল করে
3. **Action Filter** – Action Method এর আগে/পরে কাজ করে
4. **Exception Filter** – Exception ধরতে ব্যবহার হয়
5. **Result Filter** – Result তৈরি হওয়ার আগে/পরে প্রসেস করে

**➤ উদাহরণ:**

public class MyActionFilter : IActionFilter

{

public void OnActionExecuting(ActionExecutingContext context)

{

// অ্যাকশন শুরু হওয়ার আগে

}

public void OnActionExecuted(ActionExecutedContext context)

{

// অ্যাকশন শেষ হওয়ার পরে

}

}

**Register:**

services.AddControllers(options => {

options.Filters.Add<MyActionFilter>();

});

📌 **Interview Tip:** "Exception Filter ব্যবহার করে আমরা Global Error Handling করি এবং JSON format এ error response পাঠাই।"

**🔹 ৩. Hosting Model কী? In-process vs Out-of-process hosting পার্থক্য**

**উত্তর:**

**➤ In-process Hosting:**

* IIS একই প্রোসেসে (w3wp.exe) ASP.NET Core অ্যাপ চালায়
* পারফরম্যান্স বেশি ভালো

**➤ Out-of-process Hosting:**

* অ্যাপটি আলাদা Kestrel server এ চলে এবং IIS শুধু রিকোয়েস্ট ফরওয়ার্ড করে
* বেশি কনফিগারেশন দরকার

<!-- web.config -->

<aspNetCore processPath="dotnet" arguments="MyApp.dll" hostingModel="InProcess" />

📌 **Interview Tip:** "In-process hosting আমরা production এ রাখি কারণ এতে latency কম ও throughput বেশি হয়।"

**🔹 ৪. Dependency Injection-এ Scoped, Transient, Singleton লাইফটাইমের বাস্তব ব্যবহার কী?**

| **Lifetime** | **বর্ণনা** | **কখন ব্যবহার করবেন** |
| --- | --- | --- |
| Transient | প্রতি রিকোয়েস্টে নতুন | Light service যেমন validation |
| Scoped | প্রতি HTTP রিকোয়েস্টে একটি | DB Context, Repository |
| Singleton | অ্যাপজুড়ে একটি | Config loader, Cache Manager |

📌 **Interview Tip:** “DBContext আমরা Scoped রাখি কারণ একাধিক DbContext একসাথে ট্রানজ্যাকশনে সমস্যা করতে পারে।”

**🔹 ৫. ASP.NET Core-এ JWT Authentication কীভাবে কাজ করে?**

**উত্তর:**  
JWT (JSON Web Token) হলো টোকেন ভিত্তিক অথেন্টিকেশন, যেখানে ইউজার লগইন করলে সার্ভার একটি স্বাক্ষরযুক্ত টোকেন দেয় যা পরে প্রতিটি রিকোয়েস্টে পাঠানো হয়।

**➤ Token Structure:**

Header.Payload.Signature

**➤ Configure JWT:**

services.AddAuthentication(JwtBearerDefaults.AuthenticationScheme)

.AddJwtBearer(options =>

{

options.TokenValidationParameters = new TokenValidationParameters

{

ValidateIssuer = true,

ValidateAudience = true,

ValidIssuer = "myapp.com",

ValidAudience = "myapp.com",

IssuerSigningKey = new SymmetricSecurityKey(

Encoding.UTF8.GetBytes("superSecretKey"))

};

});

📌 **Interview Tip:** “JWT আমরা Mobile API, SPA (Angular, React) এর জন্য ব্যবহার করি কারণ এটি stateless।”

**🔹 ৬. Minimal APIs কী এবং কখন ব্যবহার করবেন?**

**উত্তর:**  
ASP.NET Core 6/7 এ Minimal APIs এসেছে, যেখানে Controller না লিখেও API তৈরি করা যায়।

**➤ উদাহরণ:**

var builder = WebApplication.CreateBuilder(args);

var app = builder.Build();

app.MapGet("/hello", () => "Hello World");

app.Run();

📌 **Interview Tip:** “আমরা Microservices অথবা ছোট utility API-এর জন্য Minimal API ব্যবহার করি। এতে latency কম থাকে।”

**🔹 ৭. SignalR কী এবং এর ব্যবহার কোথায়?**

**উত্তর:**  
SignalR হলো ASP.NET Core এর একটি লাইব্রেরি যা দিয়ে **real-time bi-directional communication** করা যায়।

**➤ ব্যবহার:**

* Live chat
* Real-time notifications
* Live dashboards

📌 **Interview Tip:** “আমরা SignalR দিয়ে ওয়েব সিস্টেমে user approval status, online presence এবং live logs শো করি।”

**🔹 ৮. Caching Techniques গুলো কী কী ASP.NET Core এ?**

**➤ ধরনসমূহ:**

* **In-memory caching**: Runtime এ RAM এ ডাটা রাখে
* **Distributed caching**: Redis বা SQL Server দিয়ে shared cache
* **Response caching**: Controller response cache করে

**➤ উদাহরণ:**

services.AddMemoryCache();

public class HomeController : Controller

{

private readonly IMemoryCache \_cache;

public HomeController(IMemoryCache cache)

{

\_cache = cache;

}

public IActionResult Index()

{

if (!\_cache.TryGetValue("key", out string value))

{

value = "cached value";

\_cache.Set("key", value, TimeSpan.FromMinutes(5));

}

return View("Index", value);

}

}

**🔹 ৯. Model Binding কী এবং কিভাবে customize করা যায়?**

**উত্তর:**  
Model Binding স্বয়ংক্রিয়ভাবে HTTP Request থেকে ডাটা নিয়ে Action Method এর প্যারামিটারে বসিয়ে দেয়।

**Custom Model Binder তৈরি করা যায়** জটিল টাইপের জন্য যেমন JSON, File, বা Composite Object।

📌 **Interview Tip:** “আমরা Custom Model Binder ব্যবহার করি Excel Upload এর সময় যেখানে ফাইল থেকে ডেটা read করে object-এ রূপান্তর করি।”

**🔹 ১০. Production-ready ASP.NET Core অ্যাপ তৈরির জন্য কী কী Best Practices অনুসরণ করবেন?**

✅ Environment-based configuration  
✅ Proper exception handling (UseExceptionHandler)  
✅ Input validation ও anti-XSS  
✅ Logging (Serilog, Seq)  
✅ Health Check API  
✅ CORS configuration  
✅ Secure Headers (HSTS, CSP)  
✅ HTTPS enforcement  
✅ API Versioning  
✅ Unit Test / Integration Test