ASP.NET Core Interview Questions

Beginner's Guide (Hinglish)

MVC | Web API | Entity Framework | SQL Server

Aapka pehla step **.NET development** ki duniya mein!

Hum dekhenge sabse **important concepts** jo interviews mein puche jaate hain.

Focus: **Simple explanation** aur **practical examples**.

🚀 Chalo Shuru Karte Hain! 🚀

Dependency Injection (DI): Code ko Clean aur Flexible Banata Hai

Jab ek class doosri class par depend karti hai, toh DI use hota hai.

DI Kya Hai?

> DI ek design pattern hai jisme ek class apni **dependencies** (jin cheezon ki use zaroorat hai) khud nahi banati.

Kaun Banata Hai?

> Ek special **Container** (jise IoC Container kehte hain) un dependencies ko bana kar class ko **inject** karta hai.

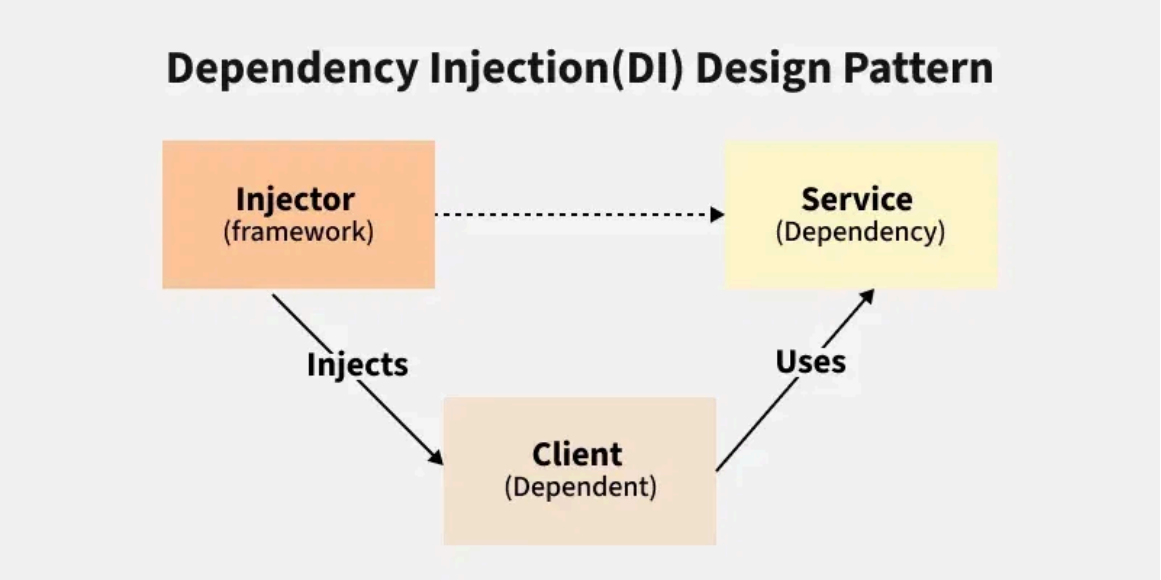
Fayda (Benefit):

> Code **loosely coupled** ho jaata hai. Agar dependency change karni ho, toh sirf ek jagah change karna padta hai.

Example:

```
builder.Services.AddScoped<IEmployeeService, EmployeeService>();

Meaning: Jab bhi koi IEmployeeService maangega, toh EmployeeService ka
object milega.
```

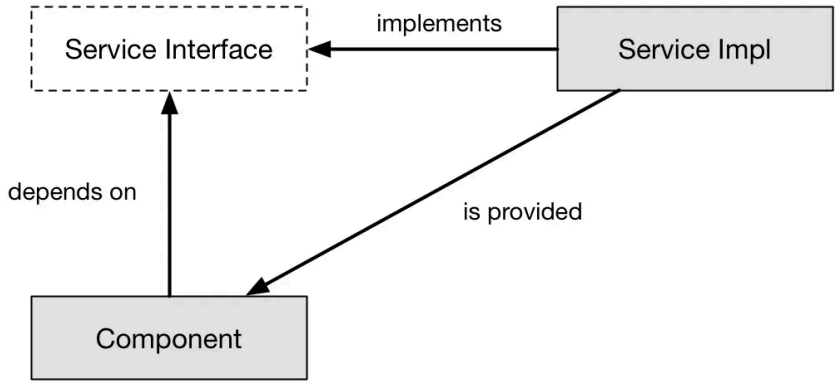


DI Lifetimes: Object ki Zindagi (Life) Kitni Hoti Hai?

Teen main tarah ki lifetimes hoti hain: Transient, Scoped, aur Singleton.

Lifetime	Meaning (Hinglish)	Example (Analogy)
Transient	Har baar naya object milega.	*Har request ke liye naya Chai ka Cup.*
Scoped	Ek HTTP Request ke liye same object.	*Ek Lunch ki Thali jo poore lunch tak chalegi.*
Singleton	Application start se end tak same object.	*Ghar ka Main Gate jo hamesha wahi rehta hai.*

💡 **Use Case:** Database connection (Scoped) ya utility service (Singleton) ke liye.

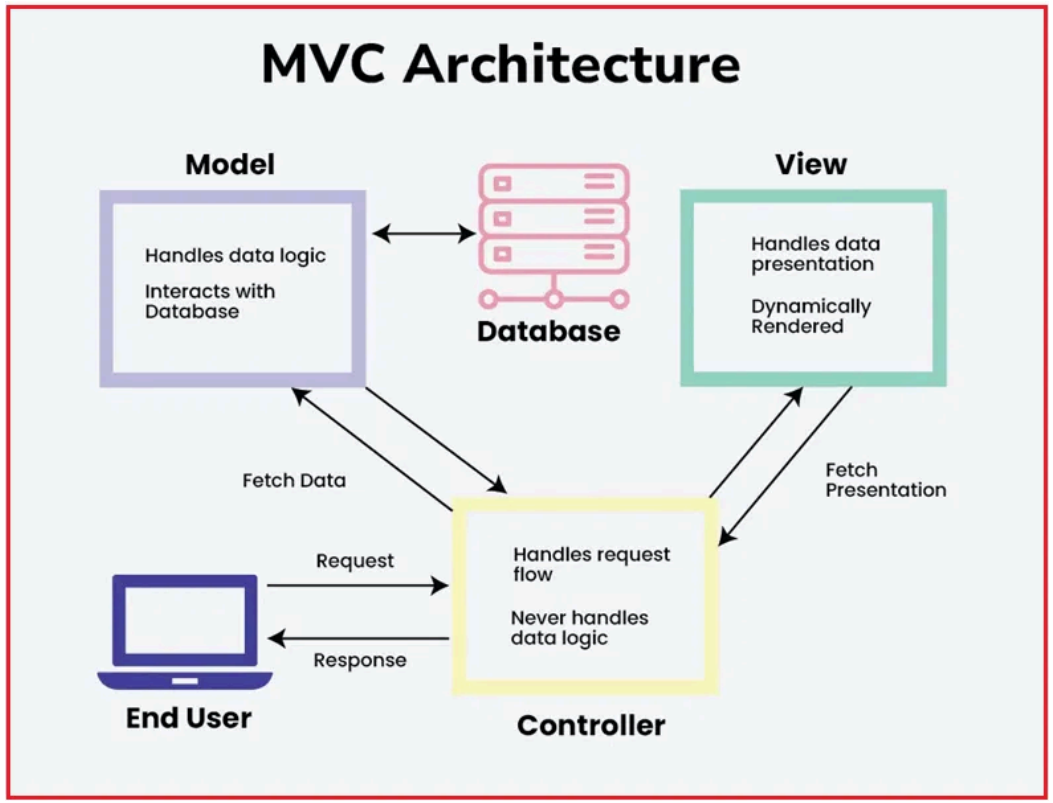


MVC aur Web API: Dono Ka Kaam Alag Hai

Dono hi ASP.NET Core mein bante hain, par unka output different hota hai.

Feature	ASP.NET Core MVC	ASP.NET Core Web API
Output	**Views** (HTML, CSS, JavaScript)	**Data** (JSON, XML)
Kaam	User Interface (UI) banana.	Mobile Apps ya Single Page Apps (SPA) ko data dena.
Example	`return View(model);`	`return Ok(data);`

🔗 **Routing:** Web API mein `[ApiController]` aur `[Route]` attributes use hote hain.



Controller se View Tak Data Kaise Bhejte Hain?

Teen main tareeke hain data ko Controller se View tak le jaane ke.

Method	Kaam (Hinglish)	Life Cycle
ViewBag Dynamic object	Simple data pass karna. Ek request ke liye.	**Sirf ek request ke liye.** Jab response bhej diya, toh ViewBag khatam ho gaya.
TempData Dictionary-based	Data ko redirect ke baad bhi use karna. One-time read.	**Next request tak survive karta hai.** Ek baar read ho gaya, toh delete ho jaata hai.
Partial View Reusable UI	Reusable UI ka tukda. Header, Footer, Navigation bar.	**Main View ke andar load hota hai.** Same request mein execute hota hai.

💡 **Best Practice Tip:**

ViewBag aur TempData se zyada better **ViewModel** use karna hota hai. ViewModel ek strongly-typed class hoti hai jo data ko safely pass karta hai aur code ko maintainable banata hai.

Entity Framework (EF): Code se Database

EF humare C# classes ko database tables mein convert karta hai.

CODE FIRST

> Sabse popular approach. Hum C# mein classes (Models) likhte hain, aur EF unse database bana deta hai.

DBCONTEXT

> Yeh class database ka **gateway** hai. Saare operations iske through hote hain.

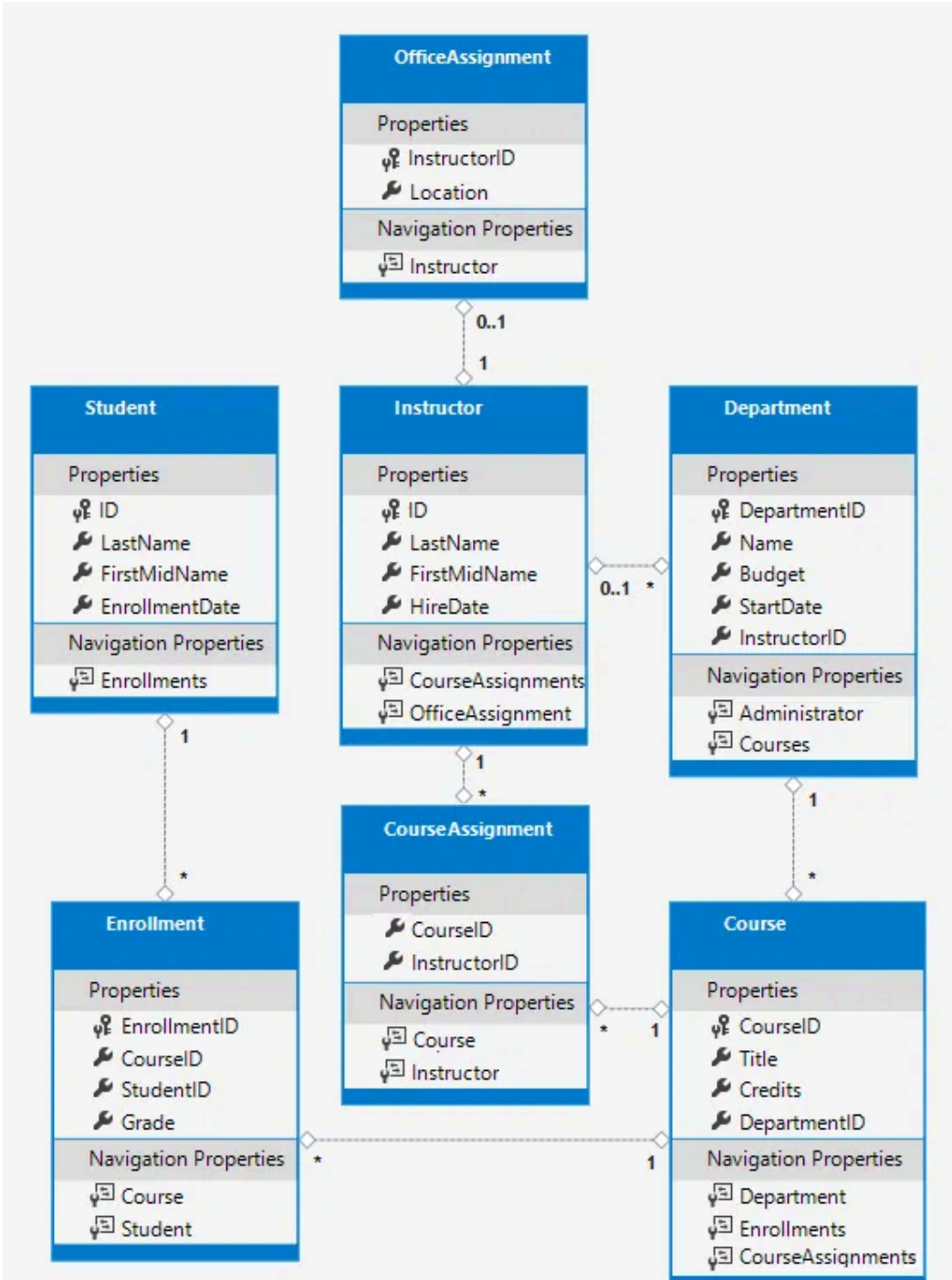
DBSET

> Yeh C# class database ke **table** ko represent karti hai.

```
public DbSet<Employee> Employees { get; set; }
```

> *Meaning: Employee class ke liye database mein ek Employees table banega.*

🔄 **Flow:** C# Models → DbContext → EF → Database Tables



Tracking vs No-Tracking: Performance aur Updates

EF ko batana ki object ko track karna hai ya nahi.

✓ Tracking

Kya Hota Hai?

EF object ko **monitor** karta hai.

Update/Delete Ke Liye:

Agar object mein change hua, toh `SaveChanges()` par DB mein update ho jayega.

✓ **Fayda:** Data ko update aur delete karna aasan ho jaata hai.

⚡ No-Tracking

Kya Hota Hai?

EF object ko **monitor nahi** karta.

Speed:

Bahut Fast hota hai, kyunki monitoring ka overhead nahi hota.

✓ **Fayda:** Sirf data ko **read** karna ho (Read-only operations).

📌 Code Example

```
// Tracking (Default)
var employees = context.Employees
    .ToList();
```

Meaning: Sab employees ko track karega. Agar kisi ka salary change hua, toh SaveChanges() par update ho jayega.

```
// No-Tracking (Fast)
var employees = context.Employees
    .AsNoTracking()
    .ToList();
```

Meaning: Sirf data read karega, kisi ko track nahi karega. Bahut fast hai!

💡 **Tip:** Jab sirf data display karna ho (List, Report), toh `AsNoTracking()` use karo.

LINQ: C# mein Data se Baat Karne ka Tareeka

Language Integrated Query (LINQ) se hum C# code mein hi data filter kar sakte hain.

▶ Where

Data ko filter karna (condition ke basis par)

Example:

```
employees.Where(e => e.Age > 18)
```

✓ 18 se upar age wale employees

▶ Select

Sirf zaroori columns nikalna

Example:

```
employees.Select(e => e.Name)
```

✓ Sirf employees ke naam

▶ Any

Check karna ki condition match ho rahi hai

Example:

```
employees.Any(e => e.Salary > 100000)
```

✓ Check if koi 1 Lakh+ kamata hai

▶ GroupBy

Data ko group mein organize karna

Example:

```
employees.GroupBy(e => e.DeptId)
```

✓ Department ke hisaab se group

💡 LINQ ke Fayde (Benefits):

✓ **Readable Code:** SQL jaisa syntax, par C# mein
✓ **Type-Safe:** Compile time par errors catch ho jaate hain
✓ **Flexible:** Collections, Databases, XML - sabke saath kaam karta hai

Authentication vs Authorization: Kaun aur Kya?

Yeh do alag concepts hain jo security ke liye zaroori hain.

🔐 Authentication

Kaun ho tum? Username aur Password se **Login** karna.

Server check karta hai ki tum kaun ho. Agar correct credentials ho, toh login ho jaate ho.

✓ Authorization

Tum kya kar sakte ho? Login ke baad **Admin** page access karna.

Server check karta hai ki tum kya permissions ho. Agar Admin role ho, toh Admin page access kar sakte ho.

```
[Authorize(Roles = "Admin")]
```

✓ Sirf Admin role wale hi is function ko access kar sakte hain.

🔑 **JWT (JSON Web Token):** Authentication ke liye use hota hai. Server ek token deta hai, aur client har request mein woh token bhejta hai.

Clustered vs Non-Clustered Index: Data Search Speed

Index database mein data ko jaldi dhundhne mein help karte hain.

Feature	Clustered Index	Non-Clustered Index
Max Count	Sirf **1** per table	**999** tak ho sakte hain
Data Storage	Actual data ko **sort** karta hai.	Data ki location ka **pointer** store karta hai.
Speed	Bahut **Fast** (Primary Key par banta hai)	**Thoda slow** (Pointer follow karna padta hai)
Default	Primary Key par automatically banta hai	Optional - manually create karna padta hai
Use Case	Frequently searched columns (ID, Date)	Other columns jo search karte ho (Name, Email)

📖 Analogy - Kitaab ka Example:

Clustered Index = Ek **kitaab** jo alphabetical order mein likhi hai. Jab aap kisi word ko dhundna ho, toh directly us page par ja sakte ho kyunki data sorted hai.

Non-Clustered Index = Kitaab ke peeche ka **Index Page** jo important topics ke page numbers batata hai. Pehle Index dekho, phir us page par jao.

🎯 **Key Takeaway:** Clustered Index bahut fast hai kyunki data directly sorted rehta hai. Non-Clustered Index thoda slow hai par flexible hai - multiple indexes bana sakte ho.

async-await: Application ki Capacity Badhana

Jab koi operation time leta hai (e.g., DB call), toh hum async-await use karte hain.

❌ Problem (Bina async-await)

Jab DB call hoti hai, toh woh **thread (worker) block** ho jaata hai aur doosri requests handle nahi kar pata.

- Thread busy rehta hai (sirf wait kar raha hai)
- Doosri requests ke liye thread available nahi hota
- Server ki capacity kam ho jaati hai
- Zyada users → Server slow ho jaata hai

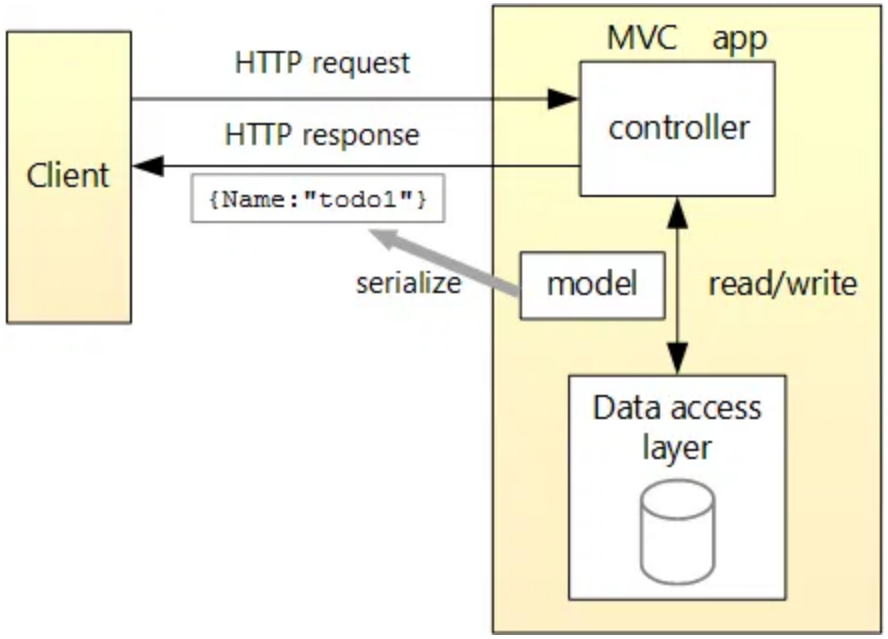
✓ Solution (async-await ke saath)

async-await use karne se, jab DB call ho rahi hoti hai, toh woh thread **free** ho jaata hai aur doosri requests handle karta hai.

- Thread block nahi hota
- Doosri requests ke liye thread available rehta hai
- Server ki capacity badh jaati hai
- Zyada users ko handle kar sakte ho

```
public async Task<IActionResult> Get() =>
    Ok(await context.Employees.ToListAsync());
```

🚀 **Fayda:** Server ki **capacity** badh jaati hai. Zyada requests handle ho sakti hain, aur application fast rehta hai.



Interview Tips aur Next Steps

Thodi si mehnat aur offer letter aapka!

💡 Interview Tips

Tip 1: Practice Code

Sirf padhne se nahi hoga. Ek **Mini CRUD Project** banao aur khud se run karke dekho.

Tip 2: Revise Loud

Har answer ko **zor se** bol kar practice karo. Confidence badhega aur memory mein bhi stick ho jayega.

Tip 3: Real Examples

Har concept ke liye **real-world example** sochna. "Yeh concept kahan use hota hai?" - yeh socho.

Tip 4: Code Review

GitHub par open-source projects dekho aur samjho ki **professional code** kaise likha jaata hai.

📊 Important HTTP Status Codes

200	201	400	401	403	404
OK - Success	Created - Resource banaya	Bad Request - Invalid data	Unauthorized - Login required	Forbidden - Permission denied	Not Found - Resource nahi mila

🚀 Next Steps (Aur Seekhne Ke Liye)

- ✓ **Microservices:** Ek bade application ko chhote services mein divide karna
- ✓ **Docker & Kubernetes:** Application ko containers mein run karna
- ✓ **Unit Testing:** Code ko test karna aur bugs dhundhna
- ✓ **Design Patterns:** Singleton, Factory, Observer patterns
- ✓ **Cloud (Azure):** Application ko cloud par deploy karna

🔥 Offer letters aa hi jayenge! 🔥

Mehnat karo, consistent raho, aur success zaroor milegi!