30 Tricky .NET Core and SQL Server Interview Questions: Detailed Answers with Examples

.NET Core: Advanced & Tricky Questions

1. Difference Between Dependency Injection and Inversion of Control

Answer:
Inversion of Control (IoC) is a design principle in which the control of objects and their dependencies is transferred from the program to a container or framework.
Dependency Injection (DI) is one way to implement IoC by providing dependencies to an object instead of the object creating them itself.

Example:

```
// Using DI via constructor
public class DataService {
    private readonly IRepository _repo;
    public DataService(IRepository repo) {
        _repo = repo; // DI provides IRepository implementation.
    }
}
```

2. What is Middleware in .NET Core?

Answer:
Middleware are software components that form a pipeline to process HTTP requests and responses. Each middleware can perform operations before and after passing control to the next component.

Example:

```
public void Configure(IApplicationBuilder app) {
    app.Use(async (context, next) => {
        // Do something before next middleware
        await next.Invoke();
        // Do something after next middleware
    });
```

```
}
```

3. How to Secure Sensitive Config Data in .NET Core?

Answer:

Secure sensitive data using Azure Key Vault, environment variables, or User Secrets (for local development). Avoid putting secrets in `appsettings.json`.

Example:

```
// appsettings.json (Don't put secrets here)
{
  "ConnectionStrings": {
    "DefaultConnection": "<from Azure Key Vault>"
  }
}
```

4. Explain Model Binding and Model Validation

Answer:

Model Binding maps HTTP request data to action method parameters.

Model Validation ensures the received data matches validation rules (using data annotations).

Example:

```
public class User {
    [Required]
    public string Name { get; set; }
}

// Controller action
[HttpPost]
public IActionResult Register(User user) {
    if (!ModelState.IsValid) {
        return BadRequest();
    }
    // Proceed with valid user
}
```

5. AddSingleton vs AddScoped vs AddTransient

Answer:

- **AddSingleton:** One instance for the entire application's lifetime.
- **AddScoped:** One instance per HTTP request.
- **AddTransient:** New instance every time requested.

Example:

```
services.AddSingleton<IMyService, MyService>();   // Single global instance
services.AddScoped<IMyService, MyService>();      // New instance per request
services.AddTransient<IMyService, MyService>();   // New instance every injection
```

6. What is Kestrel?

Answer:
Kestrel is the default cross-platform web server for ASP.NET Core applications.

Example:
`dotnet run` launches your app using Kestrel by default, serving HTTP requests.

7. .NET Core vs .NET Framework

Answer:
.NET Core:

- Cross-platform (Windows, Linux, macOS)
- Modular and open source
- High performance and evolution

.NET Framework:

- Windows-only
- Monolithic, legacy

8. Purpose of UseStartup in Program.cs

Answer:

Specifies the Startup class to configure services and the HTTP request pipeline.

Example:

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder => {
            webBuilder.UseStartup<Startup>();
        });
```

9. IHostedService vs. BackgroundService

Answer:

- **IHostedService:** Interface to create background tasks.
- **BackgroundService:** Abstract class that implements IHostedService for long-running background tasks.

Example:

```
public class MyService : BackgroundService {
    protected override async Task ExecuteAsync(CancellationToken stoppingToken) {
        while (!stoppingToken.IsCancellationRequested) {
            // Do work
        }
    }
}
```

10. Purpose of appsettings.json

Answer:

Stores configuration settings in JSON format, can support environment-specific files (like `appsettings.Development.json`) and auto-reloading.

Example:

```
{ "Logging": { "LogLevel": { "Default": "Information" } } }
```

11. Can You Mix Different Languages in App_Code?

Answer:

No. Only one language per App_Code folder (C# or VB.net).

12. Difference Between 'int' and 'System.Int32'

Answer:

No technical difference; 'int' is a C# alias for System.Int32.

Example:

```
int a = 10;
System.Int32 b = 20;
// Both are the same
```

13. What's an Assembly in .NET?

Answer:

A compiled code library file (DLL or EXE) used for deployment, security, and versioning.

14. Managed vs Unmanaged Code

Answer:

- **Managed code:** Executed by CLR (.NET runtime). Handles memory, exceptions, GC.
- **Unmanaged code:** Runs outside CLR (e.g., C++), developer manages memory.

15. What is MSIL?

Answer:

Microsoft Intermediate Language; all .NET code is compiled to MSIL, which is further compiled to native code at runtime.

16. Caching in .NET Core

Answer:

Use In-Memory Cache, Distributed Cache (e.g., Redis), and custom strategies for frequently accessed data.

Example:

```
services.AddMemoryCache();
public class MyService {
    private readonly IMemoryCache _cache;
    public MyService(IMemoryCache cache) { _cache = cache; }

    public string GetData() {
        return _cache.GetOrCreate("key", entry => "cachedValue");
    }
}
```

17. Debug vs Trace

Answer:

- `Debug`: Used only in debug builds.
- `Trace`: Used in all builds (for diagnostics production/development).

Example:

```
Debug.WriteLine("Debug info");
Trace.WriteLine("Trace info");
```

18. In-Process vs Out-of-Process Hosting

Answer:

- In-Process: App runs inside IIS process for better performance.
- Out-of-Process: IIS acts as a reverse proxy; app runs in Kestrel.

19. Functions vs Stored Procedures in .NET Context

Answer:

- Functions: Return values, can be called in SELECT, no side-effects.
- Stored Procedures: Perform operations, can return multiple values, can modify data.

20. What is LINQ?

Answer:

Language Integrated Query; allows querying collections and databases using C# syntax.

Example:

```
var results = myList.Where(x => x.Age > 18).ToList();
```

SQL Server: Advanced & Tricky Questions

21. Find the Second Highest Salary

Answer and Example:

```
SELECT MAX(Salary) AS SecondHighest
FROM Employees
WHERE Salary < (SELECT MAX(Salary) FROM Employees);
```

22. Query to Detect Duplicate Rows

Answer and Example:

```
SELECT column, COUNT(*)
FROM table
GROUP BY column
HAVING COUNT(*) > 1;
```

23. What is a Correlated Subquery?

Answer:

A subquery that references a column from the outer query.

Example:

```
SELECT e1.Name
FROM Employees e1
WHERE e1.Salary > (
    SELECT AVG(Salary)
    FROM Employees e2
    WHERE e2.DepartmentId = e1.DepartmentId
```

```
);
```

24. Clustered vs Non-Clustered Indexes

Answer:

- **Clustered Index:** Determines physical order; only one per table.
- **Non-Clustered Index:** Separate from table data; many allowed.

25. Optimizing a Slow SQL Query

Answer:

- Add indexes
- Use SARGable predicates
- Analyze the execution plan

Example:

```
CREATE INDEX idx_salary ON Employees(Salary);
-- Now WHERE Salary > 50000 is fast
```

26. Recursive CTE

Answer:
A Common Table Expression that references itself.

Example:

```
WITH RecursiveCTE AS (
    SELECT Id, ParentId
    FROM Categories
    WHERE ParentId IS NULL
    UNION ALL
    SELECT c.Id, c.ParentId
    FROM Categories c
    INNER JOIN RecursiveCTE r ON c.ParentId = r.Id
)
```

```
SELECT * FROM RecursiveCTE
```

27. Find Employees Earning More Than Their Managers

Answer and Example:

```
SELECT e1.*
FROM Employees e1
JOIN Employees e2 ON e1.manager_id = e2.id
WHERE e1.salary > e2.salary;
```

28. Explain SQL Injection and Prevention

Answer:

SQL injection is malicious code execution through user input.

Prevent it via parameterized queries.

Example:

```
string sql = "SELECT * FROM Users WHERE Username = @Username";
cmd.Parameters.AddWithValue("@Username", username);
```

29. Find Top 10% Earners

Answer and Example:

```
SELECT *
FROM Employees
WHERE Salary >= (
   SELECT PERCENTILE_CONT(0.9) WITHIN GROUP (ORDER BY Salary)
   FROM Employees
);
```

30. Explain Normalization

Answer:

Organizing data to eliminate redundancy and maintain integrity, usually by dividing data into linked tables.

Example:

- Instead of storing employee details and department in one table, separate into Employees and Departments tables connected by DepartmentId.

Which question stumped you the most? Share more in the comments!
Repost To Help Others