Task Day8 MVC

1) How can we handle security issues related to hardcoded connection strings?

1. Store the connection string in appsettings.json instead of hardcoding it in the source code.

2. Use environment variables to keep sensitive data outside the project files.

3. Use a secure secrets manager like Azure Key Vault or User Secrets to encrypt and safely store the connection string.

----

5) (Bonus) What is AJAX? Mention 3 reasons to use it in MVC apps and implement a simple demo.

AJAX (Asynchronous JavaScript and XML) allows web pages to update parts of the page asynchronously without reloading the entire page.

3 Reasons to Use AJAX:

1. Enhances user experience by avoiding full-page reloads.

2. Reduces bandwidth usage and response time.

3. Improves performance and creates smoother interactions.

----

3) IServiceCollection

هي المجموعة (Collection) اللي بيتم فيها تسجيل كل الخدمات اللي التطبيق محتاجها.

يعني بتعتبر زي "قائمة التسجيل" لكل الـ Interfaces والـ Classes اللي عايز الـ Dependency Injection يتعامل معاها.

مثال:

builder.Services.AddScoped<IDepartmentRepository, DepartmentRepository>();

builder.Services.AddScoped<IEmployeeRepository, EmployeeRepository>();

كل ما تكتب .AddScoped() أو .AddSingleton() أو .AddTransient()،

أنت كده بتضيف ServiceDescriptor جديد داخل الـ IServiceCollection.

----

4) ServiceDescriptor

ده هو الوصف التفصيلي لكل خدمة داخل الـ IServiceCollection.

يعني لما تضيف خدمة زي:

builder.Services.AddScoped<IDepartmentRepository, DepartmentRepository>();

اللي بيحصل فعليًا إن النظام بيضيف كائن من نوع ServiceDescriptor جواه:

اسم الـ Interface (IDepartmentRepository)

الكلاس اللي بينفذه (DepartmentRepository)

ونوع عمر الخدمة (Scoped أو Singleton أو Transient)

----

5) ServiceProvider

بعد ما تسجّل كل الخدمات في الـ IServiceCollection،

بييجي دور الـ ServiceProvider — وده هو المسؤول عن إنشاء الكائنات فعليًا وقت التشغيل (runtime).

يعني لما تحتاج كائن معين (زي Repository أو DbContext)،

الـ ServiceProvider هو اللي "يُنشئه" و"يحقنه" في المكان الصح.

مثال:

var app = builder.Build();

لما تكتب builder.Build(),

بيتم تحويل كل الخدمات المسجلة في builder.Services (الـ IServiceCollection)

إلى كائن فعلي من نوع ServiceProvider.

وده هو اللي التطبيق بيستخدمه بعد كده لتوفير الـ Dependencies في الكنترولرز والصفحات.

----

6) Use Cases for Singleton and Transient

1-Singleton

is an object that is **created only once** during the entire lifetime of the application, and the same instance is reused whenever it's requested.

Use Cases:

1. Database Connection:

You usually need only one shared connection to save system resources.

2. Configuration Manager:

All parts of the program use the same settings, so one shared instance makes sense.

3. Logging System:

To ensure that all events are logged in one consistent place.

4. Cache Manager:

A single cache instance stores temporary data to avoid repeated data fetching.

2-Transient

A Transient object is **created anew every time** it's requested. It doesn't maintain any shared state or memory between calls.

Use Cases:

1. Short-lived operations:

Such as a calculation or a specific user request.

2. Stateless Services:

Like sending an email or validating data (no need to preserve state).

3. State Isolation:

When you want a clean object each time, with no previous data carried over.

----

7) Alpha Constraint

The Alpha Constraint ensures that the input contains letters only, with no numbers or special characters allowed.

In other words, only alphabetic values (A–Z or a–z) are permitted.

Example:

✅ Valid: "Ahmed"

❌ Invalid: "Ahmed123" or "A@hmed"

This constraint is typically used for fields that should contain names or text only, such as:

FirstName, LastName, or City.