6. What is Dependency Injection (DI)?

- **Definition:**
  DI is a design pattern where dependencies are provided to a class rather than being created inside the class.
- **Benefits:**
  - Reduces coupling
  - Improves testability
  - Promotes clean code architecture

7. How to Implement Dependency Injection in ASP.NET Core?

- ASP.NET Core has a **built-in IoC container**.
- Register services inside `Program.cs`:

```
builder.Services.AddTransient<IMyService, MyService>();
```

- Inject into constructors:

```
public class MyController : ControllerBase
{
    private readonly IMyService _service;
    public MyController(IMyService service)
    {
        _service = service;
    }
}
```

8. DI Lifetimes

- **Transient:**
  - New instance created each time.
  - Use for lightweight, stateless services.

```
services.AddTransient<IMyService, MyService>();
```

- **Scoped:**
  - One instance per request.
  - Good for per-request operations like repositories.

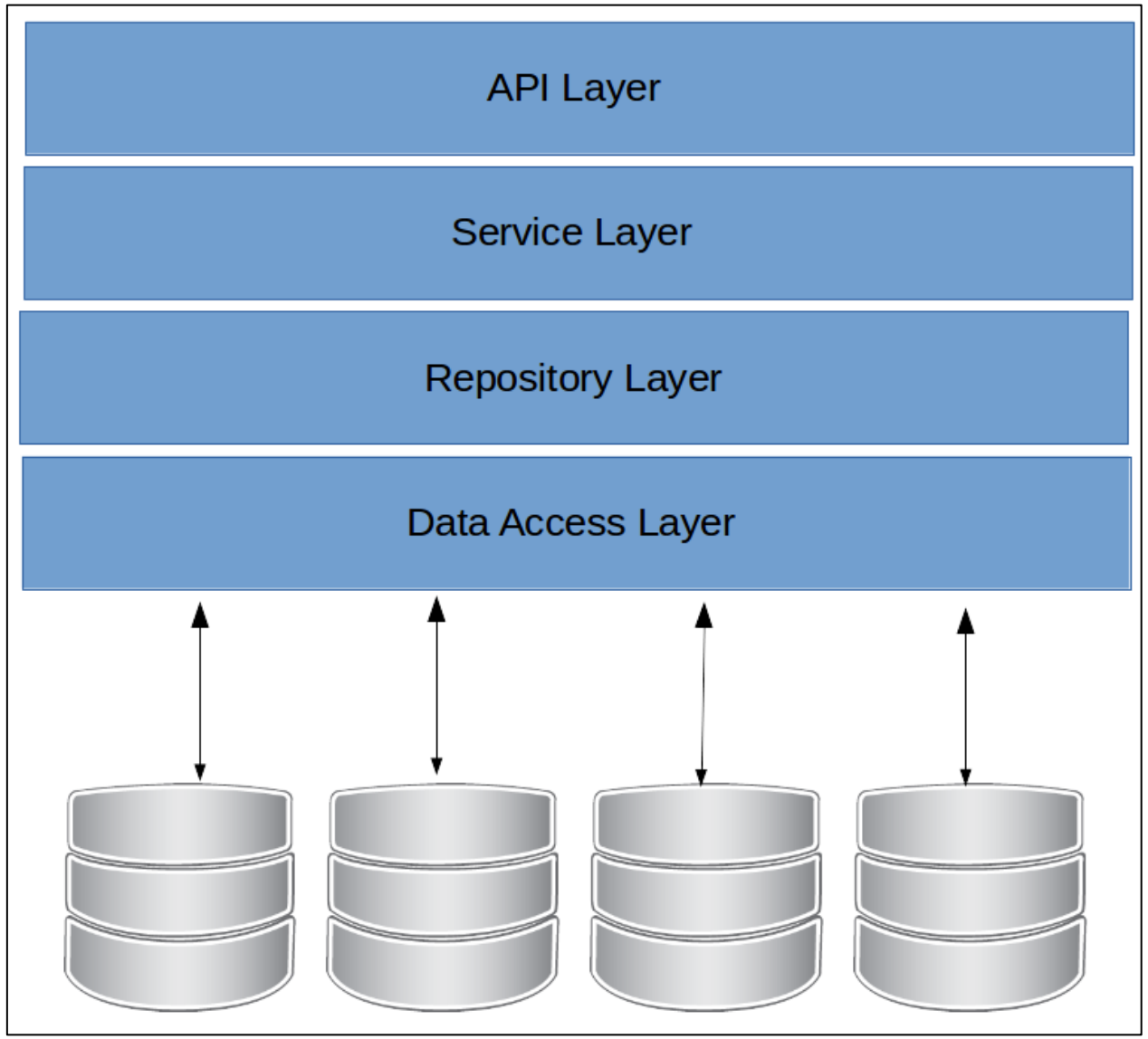
```
services.AddScoped<IMyService, MyService>();
```

- **Singleton:**
  - One instance for the entire application lifetime.
  - Use for heavy services like caching.

```
services.AddSingleton<IMyService, MyService>();
```

9. Repository and Service Pattern

- **Repository Pattern:**
  - Encapsulates data access logic.
  - Example: `IProductRepository` with methods like `GetAllProducts()`.
- **Service Pattern:**
  - Encapsulates business logic.
  - Uses repositories internally.


API Layer
Service Layer
Repository Layer
Data Access Layer

Example:

```
public interface IProductRepository
{
    IEnumerable<Product> GetAll();
}

public class ProductRepository : IProductRepository
{
    // Inject DbContext
}

public interface IProductService
{
    IEnumerable<Product> GetProducts();
}

public class ProductService : IProductService
{
    private readonly IProductRepository _repo;
    public ProductService(IProductRepository repo) => _repo = repo;

    public IEnumerable<Product> GetProducts() => _repo.GetAll();
}
```

Register with DI:

```
services.AddScoped<IProductRepository, ProductRepository>();
services.AddScoped<IProductService, ProductService>();
```