Session 9

Onion Architecture in ASP.NET Core – I

Session Overview

- Explain inversion of control, dependency inversion principle, and Onion Architecture
- Describe the Onion Architecture layers
- Explain advantages of the Onion Architecture

Onion Architecture Concepts

Major challenges of traditional Onion Architecture:

Tight Coupling

Loose Coupling

Onion Architecture is based on these basic design principles:

Inversion of Control

Dependency Inversion Principle

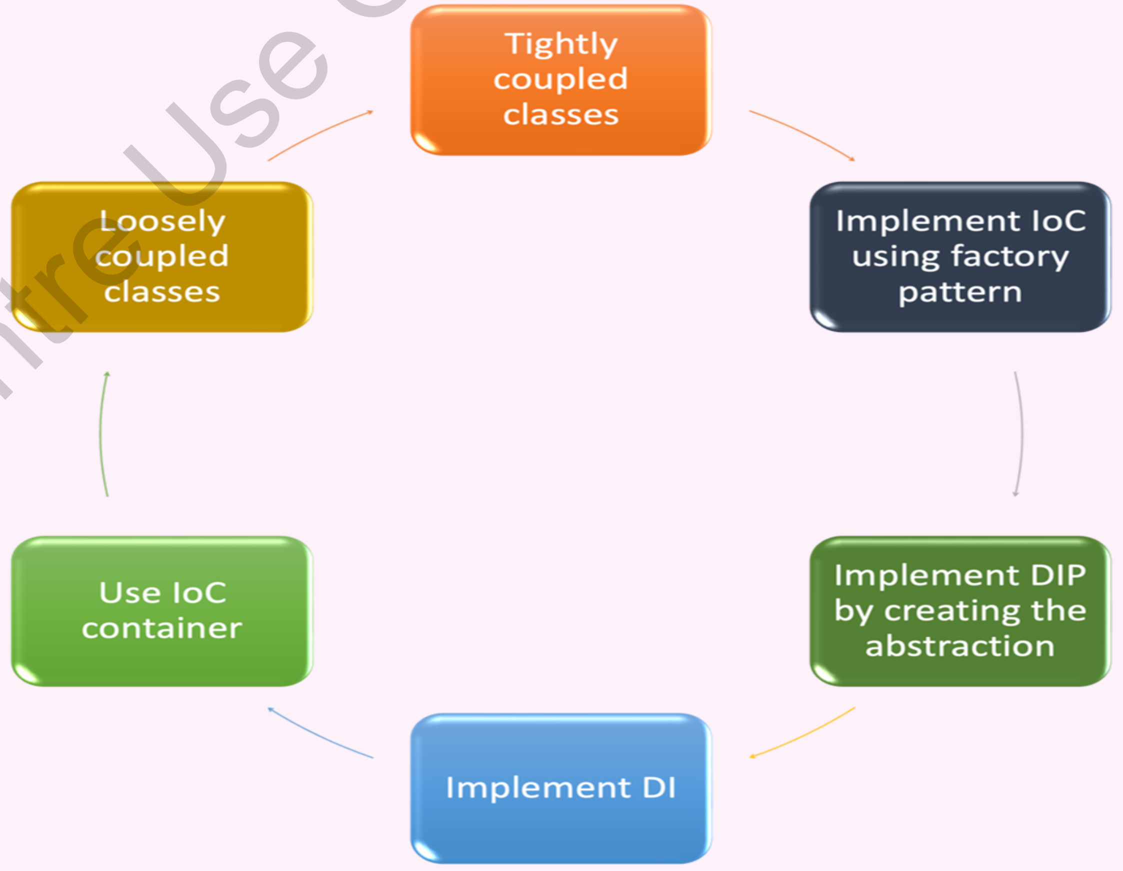
Tightly coupled classes
Implement IoC using factory pattern
Implement DIP by creating the abstraction
Implement DI
Use IoC container
Loosely coupled classes

Figure 9.1: IoC and DIP Implementation

Why Onion Architecture?

- Solves the problems related to N-layer architecture.
- Offers solution by providing several concentric layers that connect to the centre.
- Helps to build better applications in terms of testability, practicality, and consistency.
- Provides solutions to common design problems.

Layers of the Onion Architecture

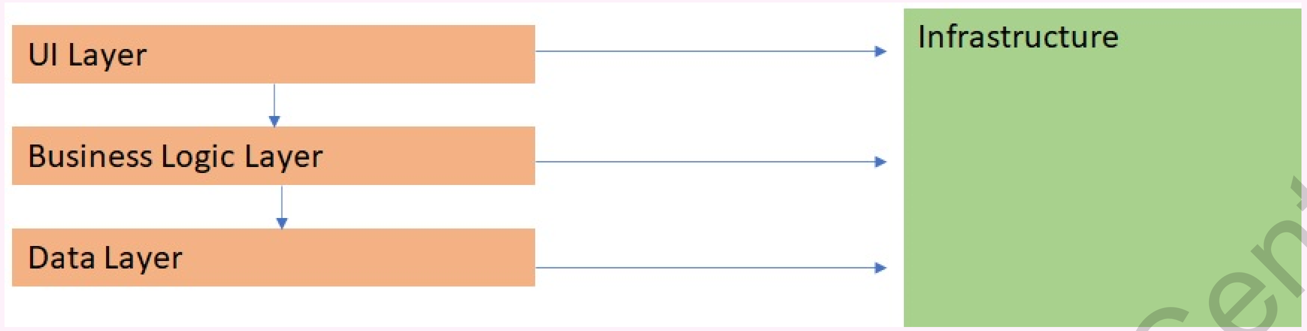

Figure 9.2: Traditional Architecture

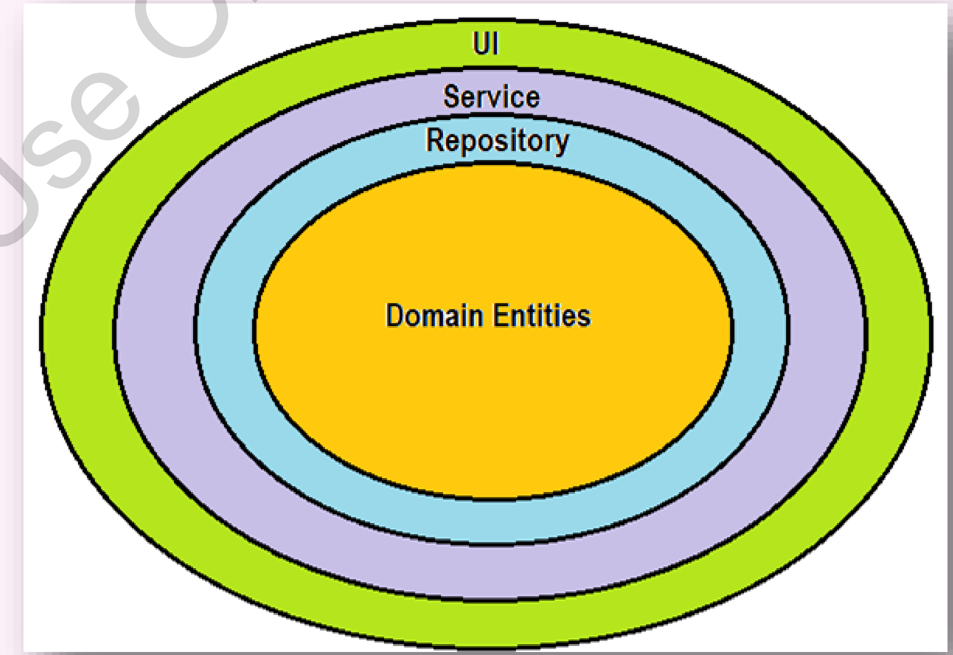

Figure 9.3: Layers of Onion Architecture

Advantages of the Onion Architecture

Easy Maintenance

- Easier to maintain because all the codes are based on layers or the center.

Improved Testing

- Allows generation of unit tests for independent levels without affecting other components of an application.

Loose Coupling

- It creates a loosely linked application as the program's outer layer always communicates with the inner layer through interfaces.

Easy Implementation

- For anything that is to be taken from an external service, an interface is created that is taken care of by higher layers of the Onion Architecture. Internal layers do not rely on external layers.

Summary

- Inversion of Control (IOC) inverts the flow of controls in an object-oriented design.
- Dependency Inversion Principle (DIP) removes the dependency between the high-level modules or classes and low-level modules or classes.
- Onion Architecture is based on the principle of IOC and DIP and addresses two major challenges of traditional architecture – tight coupling and concern partition.
- It has different layers, namely, Domain entities layer, Repository layer, Service Layer, and UI Layer.
- It creates a loosely linked application as the program's outer layer always communicates with the inner layer through interfaces.
- The Onion Architecture retains an application's business logic, data access logic, and model in the middle while pushing dependencies far outside.