# مفاهیم پایه‌ای در معماری نرم‌افزار

## (Dependency Inversion Principle)

اصل وارونگی وابستگی یکی از اصول SOLID در برنامه‌نویسی شیءگرا است. این اصل دو قانون اساسی دارد: - ماژول‌های سطح بالا نباید به ماژول‌های سطح پایین وابسته باشند، بلکه هر دو باید به abstractions وابسته باشند - به جای وابستگی مستقیم به کلاس‌های concrete، باید از interface‌ها استفاده کنیم

## IOC (Inversion of Control)

یک پارادایم طراحی که در آن کنترل جریان برنامه معکوس می‌شود: - به جای اینکه کلاس خودش وابستگی‌هایش را ایجاد کند، این وابستگی‌ها از بیرون به آن تزریق می‌شوند - این رویکرد باعث کاهش coupling و افزایش testability می‌شود

## IOC Container

فریم‌ورکی برای مدیریت وابستگی‌ها با وظایف اصلی زیر: - ثبت type‌ها و interface‌ها - ایجاد instance از کلاس‌ها - تزریق وابستگی‌ها به صورت خودکار

نمونه‌های معروف: - Microsoft.Extensions.DependencyInjection در .NET - Autofac - Ninject

## Factory as IOC Container

الگوی Factory نوع ساده‌ای از IOC Container است که: - مسئولیت ایجاد object‌ها را بر عهده دارد - نسبت به IOC Container کامل قابلیت‌های محدودتری دارد - نیاز به کدنویسی بیشتری برای مدیریت وابستگی‌ها دارد - قابلیت مدیریت lifecycle‌ها را ندارد

## Dependency Injection

تکنیکی برای پیاده‌سازی IOC که شامل روش‌های زیر است:

### روش‌های اصلی تزریق وابستگی:

1. Constructor Injection: تزریق از طریق سازنده
2. Property Injection: تزریق از طریق property
3. Method Injection: تزریق از طریق متد

### مزایای استفاده:

* کاهش coupling بین کلاس‌ها
* افزایش قابلیت تست‌پذیری
* مدیریت بهتر و متمرکز وابستگی‌ها

## Service Lifetimes در .NET

در .NET سه نوع اصلی lifetime برای سرویس‌ها وجود دارد:

### Singleton

* یک instance در کل برنامه ایجاد می‌شود
* مناسب برای سرویس‌های stateless و پرهزینه

services.AddSingleton<IMyService, MyService>();

### Scoped

* یک instance در هر request ایجاد می‌شود
* مناسب برای سرویس‌های وابسته به context

services.AddScoped<IMyService, MyService>();

### Transient

* هر بار درخواست، یک instance جدید ایجاد می‌شود
* مناسب برای سرویس‌های lightweight و stateless

services.AddTransient<IMyService, MyService>();

## ارتباط مفاهیم با یکدیگر

* DIP اصل پایه‌ای است که IOC را ممکن می‌سازد
* IOC یک پارادایم است و Dependency Injection یک روش پیاده‌سازی آن
* IOC Container ابزاری است برای مدیریت Dependency Injection
* Factory یک نوع ساده‌تر از IOC Container است
* Service Lifetimes مکانیزمی است که IOC Container برای مدیریت چرخه حیات object‌ها استفاده می‌کند