IoC Container چیست و چگونه کار میکند

در برنامه‌نویسی شیءگرا، کلاس‌ها معمولاً برای انجام وظایف خود به سایر کلاس‌ها وابسته هستند. اگر هر کلاس خودش این وابستگی‌ها را بسازد، کد ما به‌شدت به پیاده‌سازی‌ها گره می‌خورد ( Couplingبالا) و تغییر یا تست‌ کردن آن دشوار می‌شود.  
برای حل این مشکل از**Inversion of Control (IoC)**  یا وارونگی کنترل استفاده می‌کنیم. در IoC ،کنترل ایجاد و مدیریت وابستگی‌ها از درون کلاس‌ها گرفته شده و به بخشی بیرونی سپرده می‌شود.

پرکاربردترین روش پیاده‌سازی IoC در دات‌نت،**Dependency Injection (DI)** است. در این روش، وابستگی‌ها از بیرون به کلاس تزریق می‌شوند (مثلاً از طریق سازنده‌ی کلاس). به این ترتیب، کلاس تنها به یک قرارداد (Interface) وابسته می‌شود و دیگر از جزئیات پیاده‌سازی بی‌نیاز خواهد بود

برای راحتی کار، به جای اینکه وابستگی‌ها را به‌صورت دستی بسازیم و تزریق کنیم، از**IoC Container** استفاده می‌کنیم.

کانتینر وظیفه دارد:

1. **ثبت سرویس‌ها :(Registration)**مشخص کنیم هر Interface با کدام کلاس پیاده‌سازی شود.
2. **ایجاد و تزریق(Resolution):** هنگام نیاز، کانتینر شیء مناسب را می‌سازد و وابستگی‌هایش را نیز به صورت بازگشتی پر می‌کند.
3. **مدیریت طول‌عمر :(Lifetime)**تعیین کند سرویس‌ها در چه محدوده‌ای زنده بمانند(Transient, Scoped, Singleto)
4. **مدیریت Dispose:** در پایان عمر سرویس‌ها، منابع آن‌ها را آزاد کند.

انواع الگوهای تزریق:

* **Constructor Injection:** ایمن، واضح، برای وابستگی‌های ضروری
* **Property Injection:** برای وابستگی‌های اختیاری
* **Method Injection**: وقتی فقط در یک عملیات خاص لازم است

انواع طول عمر سرویس ها:

* **Transient:** هر بار نمونه‌ی جدید ساخته می‌شود.
* **Scoped** **:** در هر محدوده‌ی کاری (مثلاً هر Request در وب) یک نمونه مشترک استفاده می‌شود.
* **Singleton:**  در کل طول عمر برنامه تنها یک نمونه ساخته و استفاده می‌شود.

از مزایای IoC Container میتوان به موارد زیر اشاره کرد:

* کاهش Coupling و افزایش Testability
* مدیریت طول‌عمر و Dispose خودکار
* امکان تعویض سریع پیاده‌سازی‌ها (مثلاً استفاده از Mock در تست)
* متمرکز شدن پیکربندی در یک نقطه (Composition Root)

برای رسیدن به BestPractice در IoC Container ها میتوان به **Constructor Injection**  و **کوچک نگه‌داشتن سازنده، ثبت در Composition Root**، انتخاب درست**Lifetime**  و دوری از**Captive Dependency** ، استفاده از Dispose خودکار(IDisposable) و جلوگیری ا زnewوdispose دستی وI**nterface Segregation اشاره کرد.**

IoC و به‌ویژه استفاده از IoC Container رویکردی قدرتمند در توسعه‌ی نرم‌افزارهای مدرن است که منجر به کدی تمیزتر، قابل نگهداری‌تر و تست‌پذیرتر می‌شود. در دات‌نت Core این قابلیت به‌صورت داخلی در دسترس است و در پروژه‌های بزرگ استفاده از آن تقریباً اجتناب‌ناپذیر است.