💡 Dependency Injection

הזרקת תלויות (Dependency Injection - DI) היא תבנית עיצוב בתוכנה המאפשרת ניהול תלויות בין רכיבים בצורה יעילה ומודולרית. ב-.NET, הזרקת תלויות היא חלק מובנה במסגרת העבודה, ומסייעת ביצירת יישומים גמישים וניתנים לתחזוקה.

מהי תלות? תלות היא אובייקט שאובייקט אחר מסתמך עליו. לדוגמה, אם מחלקה Worker משתמשת במחלקה MessageWriter כדי לכתוב הודעות, MessageWriter היא תלות של Worker.

הבעייתיות בתלויות מקודדות ישירות: כאשר מחלקה יוצרת בעצמה את התלויות שלה, נוצרות מספר בעיות:

- קשה להחליף את המימוש של התלות ללא שינוי בקוד המחלקה.
- אם לתלות יש תלויות נוספות, יש לנהל גם אותן בתוך המחלקה, מה שמסבך את הקוד.
- קשה לבדוק את המחלקה בבדיקות יחידה (Unit Tests) מכיוון שלא ניתן להחליף את התלות במימוש מדומה (Mock).

כיצד הזרקת תלויות פותרת את הבעיות הללו? הזרקת תלויות מאפשרת:

- שימוש בממשקים או מחלקות בסיס כדי להפריד בין המימוש לתלות.
- רישום התלויות במכולת שירותים (Service Container) המנהלת את יצירתן ומחזור החיים שלהן.
- הזרקת התלויות דרך בנאי המחלקה, כך שהמסגרת (Framework) אחראית על יצירתן וניהולן.

הגדרת ממשק: נגדיר ממשק IMessageWriter עם מתודה Write:

```
public interface IMessageWriter
{
	void Write(string message);
}
```

מימוש הממשק: ניצור מחלקה MessageWriter שממממשת את הממשק:

```
public class MessageWriter : IMessageWriter
{
	public void Write(string message)
	{
		Console.WriteLine($"MessageWriter.Write(message: \"{message}\")");
	}
}
```

ישום השירותים: במהלך אתחול היישום, נרשום את השירותים ב Service Container:

```
var builder = WebApplication.CreateBuilder(args);

builder.Services.AddScoped*<IMessageWriter, MessageWriter>();
builder.Services.AddHostedService<Worker>();
```

© איילת ויץ

```
 var app = builder.Build();
app.Run();
```

הזרקת התלות במחלקה: נשתמש בהזרקת תלויות דרך הבנאי במחלקה Worker:

```
public class Worker : BackgroundService
{
        private readonly IMessageWriter _messageWriter;
        public Worker(IMessageWriter messageWriter)
        {
                _messageWriter = messageWriter;
        }
        protected override async Task ExecuteAsync(CancellationToken stoppingToken)
        {
                while (!stoppingToken.IsCancellationRequested)
                {
                        _messageWriter.Write($"Worker running at:{DateTimeOffset.Now}");
                         await Task.Delay(1000, stoppingToken);
                }
        }
}
```

***מחזורי חיים של Services:**

בעת רישום שירותים, ניתן לקבוע את מחזור החיים שלהם:

Transient מופע חדש נוצר בכל פעם שהשירות נדרש

Scoped מופע נוצר לכל בקשה (ביישומי Web)

Singleton מופע יחיד נוצר ומשותף לאורך חיי היישום

יתרונות הזרקת תלויות:

- הפרדה ברורה בין ממשק למימוש.
- יכולת להחליף מימושים בקלות (לדוגמה, לצורך בדיקות).
- ניהול מרכזי של תלויות ומחזור החיים שלהן.
- הגברת גמישות ותחזוקתיות הקוד.

סיכום: הזרקת תלויות היא כלי רב עוצמה ב-.NET המאפשר בניית יישומים מודולריים, גמישים וניתנים לתחזוקה. באמצעות שימוש נכון בה, ניתן לנהל תלויות בין רכיבים בצורה יעילה ולהפחית את התלות במימושים ספציפיים.

© איילת ויץ