## Patrón de Diseño DTO (Data Transfer Object)

## Taller Práctico: DTO usando AutoMapper

## Objetivo del ejemplo

Crearemos una entidad llamada Empleado, un DTO llamado EmpleadoDTO, y una API que devuelva EmpleadoDTO, usando AutoMapper.

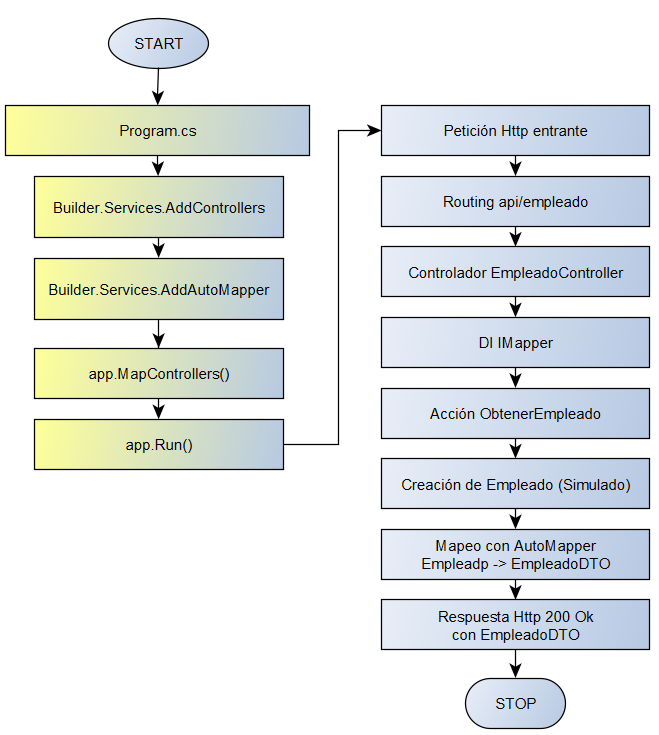
Tengamos en cuenta que El patrón DTO (Data Transfer Object) es un patrón de diseño usado para transportar datos entre procesos o capas de una aplicación, especialmente entre la capa de negocio (modelo) y la capa de presentación. También es posible lo contrario.

**Tabla 1.**

*Resumen de trazabilidad de ejecución del programa*

| **Paso** | **Descripción** |
| --- | --- |
| 1 | Se registra AutoMapper en el contenedor de servicios |
| 2 | El cliente hace una petición GET a /api/empleado |
| 3 | Se crea una nueva instancia de EmpleadoController |
| 4 | Se inyecta IMapper (singleton) al constructor |
| 5 | Se ejecuta ObtenerEmpleado() |
| 6 | Se crea un objeto Empleado manualmente |
| 7 | Se usa AutoMapper para mapearlo a EmpleadoDTO |
| 8 | Se retorna el DTO en una respuesta HTTP 200 |

**Figura 1.**

Flujo de ejecución del programa

## Crear el proyecto

Si usas Visual Studio Code:

dotnet new webapi -n NombreProyecto  
cd NombreProyecto

O Desde Visual Studio 2022:

### Abre Visual Studio 2022

* Haz clic en **"Crear un nuevo proyecto"**

### Selecciona la plantilla correcta

* Busca: ASP.NET Core Web API
* Asegúrate de que diga **.NET 8**
* Haz clic en **Siguiente**

### Configura el proyecto

* Nombre del proyecto: NombreProyecto
* Ubicación del proyecto
* Nombre de la solución: opcional
* Haz clic en **Siguiente**

### Configura el Framework

* **Framework:** .NET 8
* **Autenticación:** Ninguna (a menos que quieras autenticación)
* Marca **"Use controllers"** (IMPORTANTE: si quieres usar Controllers tradicionales y no solo el estilo minimalista)
* Marca **"Enable OpenAPI support"** (esto te activa Swagger)
* Desmarca "Enable Docker" (si no vas a usar contenedores ahora)
* Haz clic en **Crear**

## Estructura del proyecto

Creamos carpetas para organizarnos:  
/Models  
/DTOs  
/Profiles  
/Controllers

## Clase de Dominio (Entidad)

Archivo: Models/Empleado.cs  
namespace \_01\_Dto.Models  
{  
 public class Empleado  
 {  
 public int Id { get; set; }  
 public string FullName { get; set; } = string.Empty;  
 public string Role { get; set; } = string.Empty;  
 public decimal Salary { get; set; }  
 }  
}

## Clase DTO

Archivo: DTOs/EmpleadoDTO.cs  
namespace \_01\_Dto.DTOs  
{  
 public class EmpleadoDTO  
 {  
 public int Id { get; set; }  
 public string FullName { get; set; } = string.Empty;  
 public string Cargo { get; set;} = string.Empty;  
 }  
}

## Perfil de AutoMapper

Archivo: Profiles/EmpleadoProfile.cs  
using \_01\_Dto.DTOs;  
using \_01\_Dto.Models;  
using AutoMapper;  
  
namespace \_01\_Dto.Profiler  
{  
 public class EmpleadoProfile : Profile  
 {  
 public EmpleadoProfile()  
 {  
 CreateMap<Empleado, EmpleadoDTO>()  
 .ForMember(d => d.Cargo, o => o.MapFrom(s => s.Role));  
 }  
 }  
}

## Configurar AutoMapper en Program.cs

Archivo: Program.cs  
using \_01\_Dto.Profiler;  
  
var builder = WebApplication.CreateBuilder(args);  
  
// Registrar AutoMapper y escanear el assembly que contiene los perfiles  
builder.Services.AddAutoMapper(typeof(EmpleadoProfile));  
  
// Agregar controladores  
builder.Services.AddControllers();  
  
var app = builder.Build();  
  
app.MapControllers();  
  
app.Run();

## Crear el controlador para probar AutoMapper

Archivo: Controllers/EmpleadoController.cs  
using \_01\_Dto.DTOs;  
using \_01\_Dto.Models;  
using AutoMapper;  
using Microsoft.AspNetCore.Mvc;  
  
namespace \_01\_Dto.Controllers  
{  
 [ApiController]  
 [Route("api/[controller]")]  
 public class EmpleadoController : ControllerBase  
 {  
 private readonly IMapper \_mapper;  
  
 public EmpleadoController(IMapper mapper)  
 {  
 \_mapper = mapper;  
 }  
  
 [HttpGet]  
 public ActionResult<EmpleadoDTO> ObtenerEmpleado()  
 {  
 var empleado = new Empleado  
 {  
 Id = 1,  
 FullName = "Carlos Pérez",  
 Role = "Gerente de Proyectos",  
 Salary = 9500000  
 };  
  
 var dto = \_mapper.Map<EmpleadoDTO>(empleado);  
 return Ok(dto);  
 }  
 }  
}

## Probar en Postman o navegador

Levanta el proyecto en tu Visual Studio 2022:   
Abre el navegador o Postman:

Revisa en que puerto está escuchando el localhost, ejemplo:  
http://localhost:5000/api/empleado  
  
Respuesta esperada (JSON):  
{  
 "nombreCompleto": "Carlos Pérez",  
 "cargo": "Gerente de Proyectos"  
}

## ¿Qué aprendiste aquí?

• Crear un perfil de AutoMapper (EmpleadoProfile).  
• Registrar AutoMapper en Program.cs.  
• Usar AutoMapper dentro de un controlador.  
• Separar entidades y DTOs por carpetas para un mejor diseño.  
• Crear una API que devuelve DTOs automáticamente mapeados desde modelos.

¿Cuándo usar el patrón DTO?

Usa DTOs cuando:

* Estás construyendo una API
* Necesitas proteger o reducir los datos enviados
* Quieres mantener una arquitectura en capas
* Estás usando patrones como CQRS, Clean Architecture, DDD