# AutoMapper

## إيه هو AutoMapper؟

لو بتشتغل على تطبيقات .NET، أكيد واجهت مشكلة تحويل Objects من نوع لنوع تاني، زي تحويل Entity من الداتابيز إلى DTO عشان تبعته للـ API. هنا يجي دور AutoMapper، وهي مكتبة مفتوحة المصدر (Open Source) بتسمحلك تعمل تعيين تلقائي (Mapping) بين أنواع الكائنات بدون ما تكتب كود يدوي طويل وممل.

بدل ما تقعد تكتب destination.Property = source.Property لكل خاصية، AutoMapper بيعمل ده تلقائيًا بناءً على اتفاقيات (Conventions) زي تطابق الأسماء. ده بيوفر وقت، بيقلل الأخطاء، وبيحافظ على الكود نظيف.

AutoMapper مشروع شهير من Jimmy Bogard، وبيشتغل مع C#, VB.NET، وأي مشاريع .NET زي ASP.NET، Blazor، أو حتى .NET MAUI. النسخة الحالية (في 2025) هي حوالي x.13، وبتدعم .NET 8 وما فوق.

## 1. إزاي تبدأ تستخدم AutoMapper؟

البداية سهلة جدًا! خلينا نشوف الخطوات خطوة بخطوة:

- **خطوة 1: التثبيت عبر NuGet**

افتح مشروعك في Visual Studio، وركب الباقة عبر Package Manager Console:

Install-Package AutoMapper

أو من خلال الـ GUI بتاعت الـ Nuget

ده هيضيف AutoMapper لمشروعك.

- **خطوة 2: إنشاء الـ Mapping Configuration (خطوة مش هتبقي ضرورية لو عملت رقم 4)**

في مكان مركزي زي Program.cs أو Startup.cs (في ASP.NET Core)، أنشئ تكوين للـ Mapper:

```
using AutoMapper;

var config = new MapperConfiguration(cfg => {
    // هنا هتحدد الـ Mappings
    cfg.CreateMap<SourceClass, DestinationClass>();
});

IMapper mapper = config.CreateMapper();
```

الـ CreateMap ده بيقول لـ AutoMapper إزاي يحول بين نوعين. لو الأسماء متطابقة، هيعمل تلقائيًا.

- **خطوة 3: حقن الـ IMapper**

لو في تطبيق ASP.NET Core، ريجستر الـ AutoMapper في الـ Services:

```
services.AddAutoMapper(typeof(Program));
                                // أو اسم الـ Assembly اللي فيه الـ Profiles
```

وبعد كده، حقن IMapper في الـ Controllers أو Services:

```
private readonly IMapper _mapper;

public MyController(IMapper mapper) {
    _mapper = mapper;
}
```

- **خطوة 4: استخدم Profiles لتنظيم الـ Mappings**

بدل ما تحط كل الـ Mappings في مكان واحد (الخطوة 2) ، أنشئ كلاسات Profiles:

```
public class MyProfile : Profile {
    public MyProfile() {
        CreateMap<Source, Destination>();
    }
}
```

وبعد كده، AutoMapper هيكتشفها تلقائيًا لو استخدمت AddAutoMapper.

## 2. أمثلة على استخدام AutoMapper

يلا نشوف أمثلة عملية عشان تفهم الفرق!

- **مثال 1: Mapping بسيط (Automatic Mapping)**

تخيل عندك كلاسين:

```
public class UserEntity {
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
}

public class UserDto {
    public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
}
```

في الـ Configuration:

```
cfg.CreateMap<UserEntity, UserDto>();
```

الاستخدام:

```
UserEntity entity = new UserEntity { Id = 1, Name =
"Ahmed", Email = "ahmed@example.com" };

UserDto dto = _mapper.Map<UserDto>(entity);

// النتيجة: dto.Id = 1, dto.Name = "Ahmed", dto.Email =
"ahmed@example.com"
```

سهل، مش كده؟ لو الأسماء متطابقة، مش محتاج تكتب أي حاجة إضافية.

### 🔹 مثال 2: Mapping مع خصائص مختلفة الأسماء

لو الأسماء مش متطابقة، استخدم ForMember:

```
public class ProductEntity {
    public int ProductId { get; set; }
    public string ProductName { get; set; }
}

public class ProductDto {
    public int Id { get; set; }
    public string Name { get; set; }
}
```

في الـ Profile:

```
CreateMap<ProductEntity, ProductDto>()
    .ForMember(dest => dest.Id, opt => opt.MapFrom(src =>
src.ProductId))
    .ForMember(dest => dest.Name, opt => opt.MapFrom(src =>
```

```
src.ProductName));
```

الاستخدام:

```
ProductEntity entity = new ProductEntity { ProductId = 100,
ProductName = "Laptop" };

ProductDto dto = _mapper.Map<ProductDto>(entity);

// النتيجة: dto.Id = 100, dto.Name = "Laptop"
```

### 🔹 مثال 3: Mapping لقوائم (Collections)

AutoMapper بيدعم القوائم تلقائيًا:

```
List<UserEntity> entities = GetUsersFromDb(); // افترض ده جاي من الداتابيز

List<UserDto> dtos = _mapper.Map<List<UserDto>>(entities);
```

ده هيحول كل عنصر في القائمة تلقائيًا. مش محتاج تعمل حاجة زيادة خاصة بال collections بعد ما عملت الـ mappings علي الـ entities. الا في حالات معينة معقدة شوية هتحتاج شغل منك زي ان في nested collection مثلا.

### 🔹 مثال 4: Mapping مع تحويلات معقدة (Projections)

لو عايز تحول بيانات، زي دمج اتنين fields:

```
CreateMap<UserEntity, UserDto>()
    .ForMember(dest => dest.FullName, opt => opt.MapFrom(src =>
src.FirstName + " " + src.LastName));
```

## 3. نصائح مهمة للتعامل مع AutoMapper

✔️ **تجنب الـ Complex Mappings في البداية:** ابدأ بالبسيط، وبعد كده زد التعقيد زي استخدام Resolvers أو Value Converters لو محتاج تحويلات خاصة.

✔️ **تحقق من الـ Configuration:** استخدم AssertConfigurationIsValid() في الـ Config عشان تتأكد إن كل الـ Mappings صح:

```
config.AssertConfigurationIsValid();
```

✔️ **انتبه للأداء:** AutoMapper سريع، بس لو عندك ملايين السجلات، فكر في Projections مع Entity Framework عشان توفر الذاكرة.

✔️ **استخدم Reverse Mapping:** لو عايز تعيين ذهاب وعودة، استخدم ReverseMap():

```
CreateMap<UserEntity, UserDto>().ReverseMap();
```

✔️ **تعامل مع NULL:** لو في خصائص ممكن تكون NULL، حدد NullSubstitute أو تجاهلها بـ Ignore().

✔️ **اقرأ الـ Documentation:** الرسمي على GitHub عشان تفهم أكتر و تشوف الـ Features الجديدة.