Entity Framework Core – Models & DbContext

> ملخص عملي وبسيط لفهم **Entity Framework Core** بأسلوب مناسب لـ **Notion / LinkedIn** مع شرح عربي مختصر وقابل للتعديل.

1️⃣ تعريف DbContext

`DbContext` هو المسئول عن الربط بين الكود وقاعدة البيانات.

```
public class ApplicationDbContext : DbContext
{
}
```

🔹 Connection String

في المشاريع الحقيقية بنحطه في `appsettings.json`، لكن للتوضيح:

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer(
        "Data Source=.;Database=CarData;Integrated Security=True;");
}
```

2️⃣ Domain Models (Entities)

كل جدول في الداتابيز = Class

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

تسجيل الكلاسات في DbContext

```
public DbSet<Employee> Employees { get; set; }
```

⚠ أي كلاس **مش متسجل في DbContext** مش هيتحول لجدول.

3️⃣ Migrations

إنشاء Migration

```
Add-Migration InitialCreate
```

تطبيقها على الداتابيز

```
Update-Database
```

4️⃣ إضافة بيانات (Insert Data)

```
ApplicationDbContext context = new ApplicationDbContext();

Employee employee1 = new Employee
{
    Name = "Ahmed"
};

context.Employees.Add(employee1);
context.SaveChanges();
```

📌 لا نحدد الـ Id لأنه Identity.

5️⃣ Rollback Migration

```
Update-Database 0
```

➡ حذف كل الـ migrations

```
Update-Database MigrationName
```

➡ الرجوع لحد Migration معين

حذف ملف Migration

```
Remove-Migration
```

⚠ لازم تعمل rollback الأول.

6️⃣ Seeding باستخدام Migration

```
protected override void Up(MigrationBuilder migrationBuilder)
{
    migrationBuilder.Sql("INSERT INTO Employees VALUES('Mohamed')");
}

protected override void Down(MigrationBuilder migrationBuilder)
{
    migrationBuilder.Sql("DELETE FROM Employees WHERE Name='Mohamed'");
}
```

📌 البيانات دي بتظهر عند أي شخص يعمل `Update-Database`.

7️⃣ Constraints (القيود)

Data Annotations (سهل)

```
[Required]
public string Url { get; set; }
```

Fluent API (أفضل)

```
modelBuilder.Entity<Blog>()
    .Property(b => b.Url)
    .IsRequired();
```

8️⃣ التحكم في الجداول

تغيير اسم الجدول

```
[Table("Posts")]
```

أو

```
modelBuilder.Entity<Post>().ToTable("Posts");
```

استبعاد كلاس من Migration

```
modelBuilder.Entity<Blog>()
    .ToTable("Blogs", b => b.ExcludeFromMigrations());
```

9️⃣ Properties Configuration

- تغيير اسم العمود
- نوع البيانات
- الطول
- Default Value
- Computed Column

```
modelBuilder.Entity<Blog>()
    .Property(b => b.Url)
    .HasColumnType("varchar(50)")
    .HasMaxLength(50)
    .HasDefaultValue("N/A");
```

🔟 Primary Keys

```
modelBuilder.Entity<Blog>()
    .HasKey(b => b.Serial)
    .HasName("SerialKey");
```

Composite Key

```
modelBuilder.Entity<Blog>()
    .HasKey(b => new { b.Serial, b.Id });
```

1️⃣1️⃣ Relationships

One To One

- Navigation Property في الطرفين

One To Many

- List الـ في الطرف Many

Many To Many

- ينشئ جدول وسيط تلقائيًا EF Core
- أو نتحكم فيه يدويًا باستخدام `UsingEntity`

1️⃣2️⃣ Indexes

```
modelBuilder.Entity<Post>()
    .HasIndex(p => p.Title)
    .IsUnique()
    .HasDatabaseName("IX_Title");
```

✅ الخلاصة

- DbContext هو قلب EF Core
- Domain Models = Tables
- Migrations لإدارة التغييرات
- Fluent API أقوى من Data Annotations
- العلاقات لازم تتظبط صح عشان الداتابيز تطلع سليمة

📌 مناسب للنشر على: - LinkedIn - Notion - كمرجع مذاكرة سريع