

.NET CORE

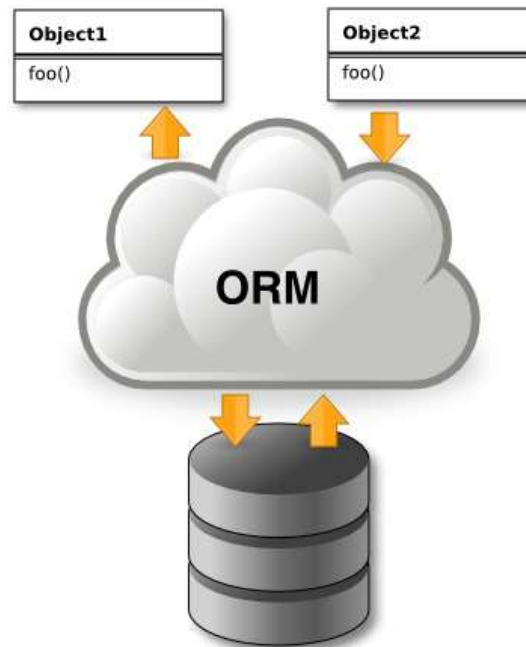
6. Capa de Acceso a Datos

- **Entity Framework Core**

Capa de acceso a Datos

¿Qué es un ORM?

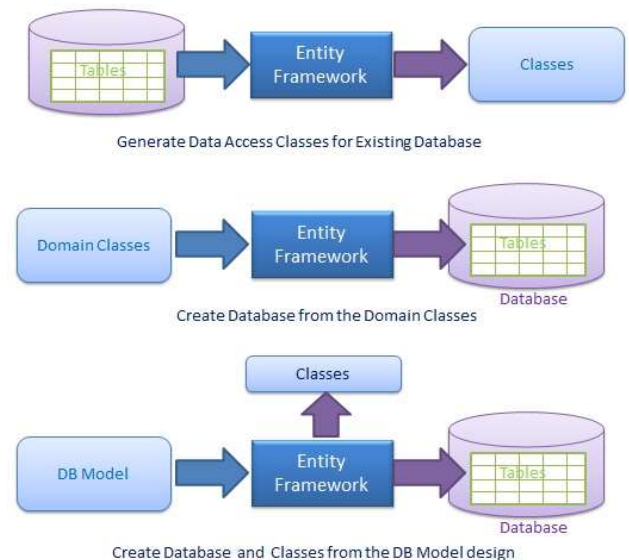
- ORM (Object Relational Mapping) es una biblioteca especializada en acceso a datos que genera un mapeo entre tus objetos de dominio con las tablas de Base de Datos
 - Nos abstrae de los tipos de datos que se usan en Base de Datos para que usemos los de nuestros objetos
 - Se pueden generar las clases a partir de las tablas o al revés, las tablas a partir de las clases.
 - Nos hace pensar en objetos en lugar de en tablas



Capa de acceso a Datos

¿Qué es un ORM?

- Ventajas
 - No tienes que escribir código SQL
 - Se saca todo el partido a la programación orientada a objetos
 - Nos permiten aumentar la reutilización del código y mejorar el mantenimiento del mismo
 - Mayor seguridad, ya que se ocupan automáticamente de higienizar los datos que llegan, evitando posibles ataques de inyección SQL
 - Hacen muchas cosas por debajo por nosotros como la conversión de tipos
 - Desventajas
 - Pueden llegar a ser muy complejos
 - No son ligeros por regla general: añaden una capa de complejidad a la aplicación que puede hacer que empeore su rendimiento
 - La configuración inicial que requieren se puede complicar dependiendo de la cantidad de entidades que se manejen y su complejidad
 - El hecho de que te aíse de la base de datos y no tengas casi ni que pensar en ella es un arma de doble filo.
- Si no sabes bien lo que estás haciendo y las implicaciones que tiene en el modelo relacional puedes construir modelos que generen consultas monstruosas y muy poco óptimas contra la base de datos, agravando el problema del rendimiento y la eficiencia



Capa de acceso a Datos

Entity Framework Core

- Entity Framework es el ORM de Microsoft. Tiene versiones tanto para .NET Framework como para .NET Core.
 - Con EF Core, el acceso a datos se realiza mediante un modelo. Un modelo se compone de clases de entidad y un objeto de contexto que representa una sesión con la base de datos, lo que permite consultar y guardar los datos.

```
public class Blog
{
    public int BlogId { get; set; }
    public string Url { get; set; }
    public int Rating { get; set; }
    public List<Post> Posts { get; set; }
}

public class Post
{
    public int PostId { get; set; }
    public string Title { get; set; }
    public string Content { get; set; }

    public int BlogId { get; set; }
    public Blog Blog { get; set; }
}
```

```
public class BloggingContext : DbContext
{
    public DbSet<Blog> Blogs { get; set; }
    public DbSet<Post> Posts { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        optionsBuilder.UseSqlServer(
            @"Server=(localdb)\mssqllocaldb;Database=Blogging;Integrated Security=true");
    }
}
```

Capa de acceso a Datos

Entity Framework Core: Guardar y Crear datos

- Para consultar, guardar o crear datos debemos crear una instancia del contexto y usarla.
 - Las consultas se pueden tirar contra el DbSet del objeto correspondiente que queremos consultar
 - La creación y modificación, además de usar el DbSet debemos usar el método SaveChanges después para que nos guarde todos los cambios que hemos ido trackeando.

```
using (var db = new BloggingContext())
{
    var blogs = db.Blogs
        .Where(b => b.Rating > 3)
        .OrderBy(b => b.Url)
        .ToList();
}
```

```
using (var db = new BloggingContext())
{
    var blog = new Blog { Url = "http://sample.com" };
    db.Blogs.Add(blog);
    db.SaveChanges();
}
```

Capa de acceso a Datos

Entity Framework Core: Añadir a ASP.NET Core

- Para añadir EF a nuestra aplicación ASP.NET Core hay que:
 - Instalar el paquete Nuget de Entity Framework: Microsoft.EntityFrameworkCore
 - Añadir EF en ConfigureServices con AddContext<ClaseContext> añadiéndole la cadena de conexión de nuestra BBDD.

```
public void ConfigureServices(IServiceCollection services)
{
    services.Configure<CookiePolicyOptions>(options =>
    {
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = SameSiteMode.None;
    });

    services.AddDbContext<SchoolContext>(options =>
        options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")))

    services.AddMvc();
}
```

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=Cc
  },
  "Logging": {
    "IncludeScopes": false,
    "LogLevel": {
      "Default": "Warning"
    }
  }
}
```

AppSettings.json

Capa de acceso a Datos

Entity Framework Core: Inyección en un Controlador

- Podemos inyectar el contexto de EF en cualquier sitio una vez configurado simplemente añadiéndolo en el Constructor de Controladores, Servicios, etc

```
public class StudentsController : Controller
{
    private readonly SchoolContext _context;

    public StudentsController(SchoolContext context)
    {
        _context = context;
    }
}
```

Capa de acceso a Datos

Entity Framework Core: Mappings

- Las clases Mappings definen el mapeo entre una Clase y una Tabla de Base de Datos.
- Son clases que se pueden definir aparte del contexto e incluirlas en éste.

```
0 references | Angel Carlos Lopez, 30 days ago | 1 author, 2 changes | 3 work items
public class CustomerMapping : IEntityTypeConfiguration<Customer>
{
    5 references | Angel Carlos Lopez, 30 days ago | 1 author, 2 changes | 3 work items
    public void Configure(EntityTypeBuilder<Customer> builder)
    {
        builder.ToTable("Customers");

        builder.Property(c => c.Name).IsRequired().HasMaxLength(200);
        builder.Property(c => c.Surname).IsRequired().HasMaxLength(200);
        builder.Property(c => c.Dni).IsRequired().HasMaxLength(10);
        builder.HasIndex(customer => customer.Dni).IsUnique();
    }
}
```

```
13 references | Angel Carlos López Quevedo, 164 days ago | 1 author, 1 change
public class OrdersAppContext : DbContext
{
    Publics Methods

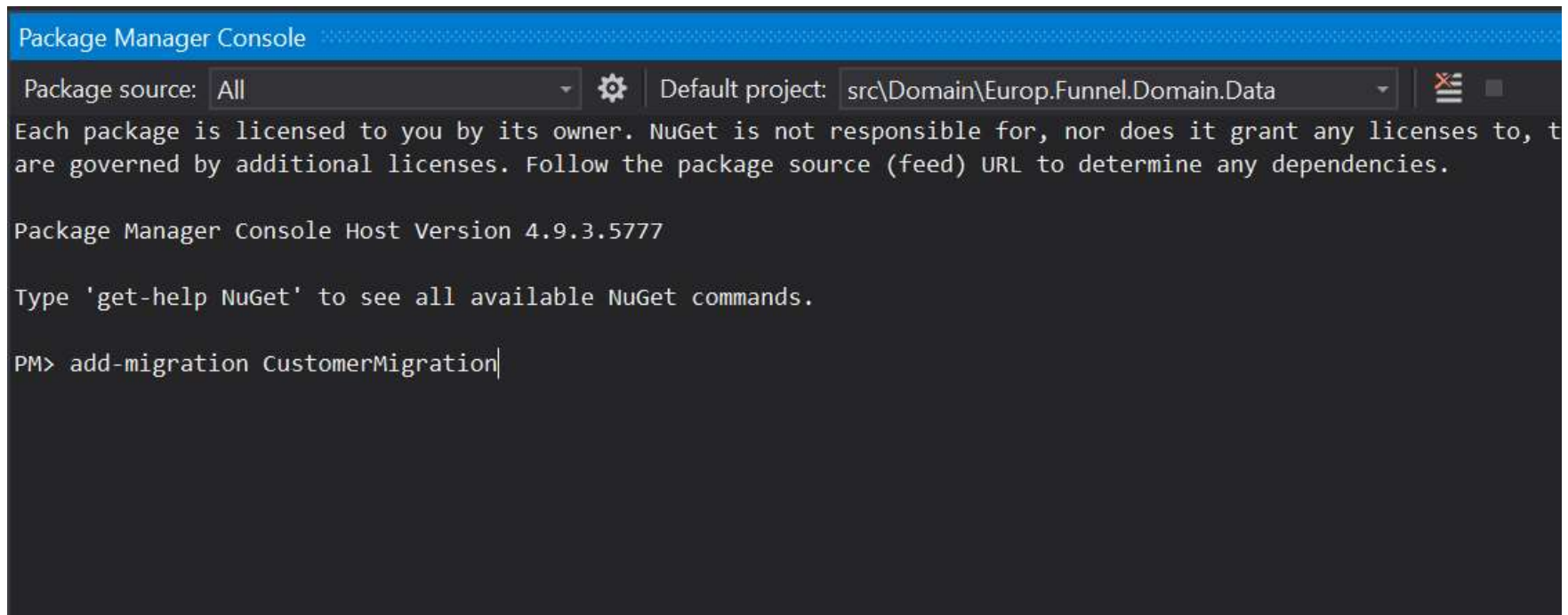
    0 references | Angel Carlos López Quevedo, 164 days ago | 1 author, 1 change | 0 exceptions
    protected override void OnModelCreating(ModelBuilder modelBuilder)
    {
        modelBuilder.ApplyConfiguration(new CustomerMapping());

        base.OnModelCreating(modelBuilder);
    }
}
```


Capa de acceso a Datos

Entity Framework Core: Migraciones

- EF Core tiene un sistema de Migraciones que podemos usar para evolucionar la base de datos
 - add-migration: añade una migración al código con las diferencias entre los mappings y la base de datos
 - update-database: nos actualiza la base de datos con las últimas migraciones añadidas al código.
- También se puede añadir métodos semilla para crear datos en la base de datos por defecto.



```
Package Manager Console
Package source: All [v] [g] Default project: src\Domain\Europ.Funnel.Domain.Data [v] [x] [h]
Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, the packages included in a package that you have installed. All packages included in a package that you have installed are governed by additional licenses. Follow the package source (feed) URL to determine any dependencies.

Package Manager Console Host Version 4.9.3.5777

Type 'get-help NuGet' to see all available NuGet commands.

PM> add-migration CustomerMigration
```

Capa de acceso a Datos

Dapper: MicroORM

- Dapper es un MicroORM ya que nos permite lanzar SQL contra la base de datos mapeando entidades pero no le hace falta tanta definición.
- Se suele usar para consultas y para crear nuestro código propio SQL cuando EF no es del todo bueno en rendimiento.

```
string sql = "SELECT TOP 10 * FROM OrderDetails";

using (var connection = new SqlConnection(FiddleHelper.GetConnectionStringSql)
{
    var orderDetails = connection.Query<OrderDetail>(sql).ToList();

    FiddleHelper.WriteTable(orderDetails);
})
```

Capa de acceso a Datos

CQS: Command-Query Segregation

- Lo ideal en las aplicaciones es tener separados por un lado Queries y Comandos.
 - Los comandos se suelen implementar usando ORM's como Entity Framework que nos ayuda mucho en la abstracción de Inserts y Updates complejos
 - Para tener más rendimiento y un código SQL más limpio en las consultas se suele emplear Dapper.

