Unidad 4 – Capitulo 2 - Laboratorio 1

Trabajando con el ORM Entity Framework Core

Objetivos

Comprender los fundamentos de los ORM (Object Relational Mapping) y experimentar utilizando Entity Framework Core para manipular datos del dominio de negocio (entidades de negocio) modelados en forma de clases / objetos.

Duración Aproximada

30 minutos

Consignas

Realizar el ABMC / CRUD de una entidad o concepto del dominio. En los pasos que se describen a continuación, se adopta como ejemplo la entidad `Alumno` de un hipotético caso de negocio en el contexto de una Universidad, pero el ejemplo es fácilmente trasladable a cualquier otro escenario de negocio.

Pasos sugeridos

1) Utilizando Visual Studio, crear un proyecto de tipo Consola.

2) Crear una clase que represente un `Alumno` en el dominio de una Universidad.

   a) Definir algunas propiedades dentro de la clase creada que representen sus datos asociados, estas propiedades podrían ser:
      - `Id` (`int`)
      - `Apellido` (`string`)
      - `Nombre` (`string`)
      - `Legajo` (`int`)
      - `Direccion` (`string`)

3) Agregar el paquete NuGet `Microsoft.EntityFrameworkCore.SqlServer`.

4) Crear una clase que herede de `DbContext` para representar una sesión con la base de datos donde se persistirá la información del `Alumno`, esta clase podría llamarse `UniversidadContext`. Dentro de esta clase:

   a) Definir una propiedad que represente la colección de alumnos, por ejemplo:
      - `Alumnos` (`DbSet<Alumno>`)

   b) Sobrescribir el método `OnConfiguring` de la clase base para configurar el uso de SQL Server como base de datos (adaptar la cadena de conexión según corresponda):

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuil
der)
{
    optionsBuilder.UseSqlServer(@"Server=(localdb)\MSSQLLocalDB;Initial
Catalog=Universidad;Integrated Security=true");
```

```
}
```

TIP: si se desea imprimir en la consola el código SQL que el ORM genera al interactuar con la base de datos, se puede configurar en el `optionsBuilder` de esta manera:

```
optionsBuilder.LogTo(Console.WriteLine, LogLevel.Information);
```

c) Establecer en el constructor de la clase que el ORM se asegure de crear la base de datos si la misma no existe:

```
public UniversidadContext()
{
    this.Database.EnsureCreated();
}
```

5) En el archivo Program.cs, crear 4 métodos para realizar cada una de las operaciones de ABMC / CRUD típicas sobre el `Alumno`, cada método deberá apoyarse sobre una instancia de `UniversidadContext` para interactuar con la base de datos:

   a) `Crear`: instanciar un `Alumno` y guardarlo

   b) `Leer`: recuperar un `Alumno` previamente creado a partir de algún criterio, como puede ser su nombre o legajo, e imprimir sus datos en la consola

   c) `Actualizar`: realizar modificaciones sobre un `Alumno` en particular y guardar los cambios

   d) `Eliminar`: eliminar un `Alumno` previamente guardado

   TIP: instanciar la clase `UniversidadContext` anteponiendo la palabra clave `using` a la declaración de la variable; de esta manera, nos aseguraremos de cerrar la conexión con la base de datos y liberar recursos cuando finalice su utilización.

6) Dentro del archivo Program.cs, en el flujo de ejecución principal de la aplicación de consola, invocar cada uno de los 4 métodos creados e ir verificando el resultado que producen en la base de datos:

   a) La verificación del resultado de cada método puede realizarse utilizando SSMS (SQL Server Management Studio) o también desde la ventana Explorador de Objetos SQL Server dentro de Visual Studio

   b) Como sugerencia, se puede colocar un punto de interrupción en la primera línea de código del flujo principal de la aplicación e ir ejecutando paso a paso dicho flujo para, en simultáneo, evaluar los cambios que se van produciendo en la base de datos