Escuelita Crombie 3er Edición

Entity Framework

¿Qué es un ORM en programación?

Desafios en la conexin a una base de datos

- Mantenimientos del esquema
- Integridad de los datos: Es importante garantizar la integridad de los datos almacenados en una base de datos. Esto incluye prevenir la corrupción de datos y garantizar la consistencia de los datos.
- Transformación de datos para ser usados en el Backend
- Seguridad: Es importante garantizar la seguridad de los datos almacenados en una base de datos. Esto incluye proteger los datos contra el acceso no autorizado y la manipulación indebida.

¿Que es un ORM?
Un ORM, o mapeador objeto-relacional, es una herramienta de software que permite a los desarrolladores trabajar con una base de datos relacional mediante objetos y entidades en lugar de tablas y columnas.

El objetivo de un ORM es simplificar el proceso de acceso y manipulación de datos en una base de datos, permitiendo a los desarrolladores trabajar con objetos y entidades en lugar de escribir consultas SQL.
Esto hace que el código sea más legible, mantenible y fácil de escribir, y también ayuda a prevenir errores comunes al trabajar con bases de datos, como las inyecciones SQL.

Instalación

Instalación de EF

- dotnet add package Microsoft.EntityFrameworkCore

Instalación para conectarnos con el motor SQL Server

- dotnet add package Microsoft.EntityFrameworkCore.SqlServer

Modelos Ejemplo

```
public class Categoria
{
        public Guid CategoriaId {get; set;}
        public string Nombre {get; set;}
        public string Descripcion {get; set;}
        public virtual ICollection<Tarea> Tarea  {get; set;}
}
```

```
public class Tarea
{
        public Guid TareaId {get; set;}
        public Guid CategoriaId {get; set;}
        public string Titulo {get; set;}
        public string Descripcion {get; set;}
        public Prioridad PrioridadTarea {get; set;}
        public DateTime FechaCreacion {get; set;}
        public virtual Categoria Categoria {get; set;}
}

public enum Prioridad
{
        Baja,
        Media,
        Alta
}
```

Configuración de EntityFramework

TareaContexto.cs

```
public class TareasContext : DbContext
{
        public DbSet<Categoria> Categorias  {get; set;}
        public DbSet<Tarea> Tareas  {get; set;}

        public TareaContext(DbContextOptions<TareasContext> options) : base(options) {}

}
```

Conexion con Base de datos

builder.Services.AddSqlServer<TareasContext
>(builder.Configuration.GetConnectionString(“NombreDeConnectionString”)”));

EnsureCreated() para asegurar la creación de la base de datos

```
using (var scope = app.Services.CreateScope())
{
        var dbContext = scope.ServiceProvider.GetRequiredService<TareaContext>();

           if (dbContext.Database.CanConnect())
           {
              Console.WriteLine("Conexión exitosa a la base de datos.");
              dbContext.Database.EnsureCreated();
           }
           else
           {
              Console.WriteLine("Error: No se pudo conectar a la base de datos.");
           }

}
```

Migraciones

Es una funcionalidad de Entity Framework que nos permite guardar de manera incremental los cambios realizados en la base de datos.

Nos permite construir un versionamiento de la
base de datos.

Instalar EntityFrameworkCore Design

- dotnet add package Microsoft.EntityFrameworkCore.Design --version 8.0.11

Instalación global EF Tools

- dotnet tool install –global dotnet-ef

Actualizar global EF Tools

- dotnet tool update –global dotnet-ef

Probar si esta correctamente instalada

- dotnet ef

Crear la migración

Esto creara una carpeta “Migration” con un version inicial

- dotnet ef migrations add InitalCreate
- Add-Migration InitialCreate

Creara una segunda version

- dotnet ef migrations add MyMigration
- Add-Migration MyMigration

Carpeta Migration

idMigracion_nombreMigracion.cs

funcion Up(){}

Son aquellos cambias que subira la migración al actualizar la base de datos

funcion Down()

Es la funcion determina lo cambios que se revertirán al hacer un rollback

Imapactar la migracion en base de datos

- dotnet ef database update
- Update-Database

Dentro de nuestra base de datos se creará una tabla

dbo.EFMigrationHistory, que será la tabla la cual lleva el registros de las migraciones

Data Annotation

Data Annotation puede darse cuenta cual es el atributo PK de cada clase, porque tiene el mismo nombre de la clase y/o porque termino Id, pero también se puede especificar con la annotation [Key]

[Required] = Define que la propiedad es requerida

[MaxLength(150)] = Define que registro tendrá con máximo 150 Caracteres

[ForeignKey(“CategoriaId”)] = Define el nombre de la clave foránea

[NotMapped] = Esta etiqueta define que este atributo no representara en la base de datos

[Range(1, 100, ErrorMessage = "El precio debe estar entre 1 y 100.")]

[RegularExpression(@"^[a-zA-Z]+$", ErrorMessage = "Solo se permiten letras.")]

[EmailAddress(ErrorMessage = "El correo electrónico no es válido.")]

[Column("Product_Name")] = Personaliza el nombre de la columna en la base de datos.

[Table] = Define el nombre de la tabla de la base de datos.

```
[Table("Products")]
public class Product
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```