FEVEREIRO, 2019

ORM

Mapeamento objeto-relacional

Sprint 2

Escola SENAI de Informática

INTRODUÇÃO

Um ORM (Object-Relational Mapping), nada mais é do que um Framework ou um conjunto de classes que permite que você faça este trabalho sem precisar escrever códigos de conexão com o banco, querys de SQL a todo momento, preservando as características de orientação a objetos da linguagem face à natureza relacional dos bancos de dados atuais.

FRAMEWORKS ORM

ENTITY FRAMEWORK

http://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx
https://www.learnentityframeworkcore.com/

DAPPER

https://dapper-tutorial.net/dapper

NHIBERNATE

https://www.devart.com/entitydeveloper/nhibernate-designer.html
https://nhibernate.info/

ENTITY FRAMEWORK CORE

O EF Core é um mapeador objeto-relacional (O/RM) que permite aos desenvolvedores .NET trabalhar com um banco de dados usando objetos .NET. Ele elimina a necessidade da maior parte do código de acesso a dados que os desenvolvedores normalmente precisam escrever. O EF Core suporta muitos mecanismos de banco de dados.
O Entity Framework Core (EF Core) é uma versão leve, extensível e multiplataforma do Entity Framework.

ENTITY FRAMEWORK CORE

```
public List<EventoDomain> Listar()
{
    string QuerySelect = @"SELECT
                            E.ID AS ID_EVENTO,
                            E.TITULO AS TITULO_EVENTO,
                            E.DESCRICAO,
                            E.DATA_EVENTO,
                            E.ACESSO_LIVRE,
                            TE.ID AS ID_TIPO_EVENTO,
                            TE.TITULO AS TITULO_TIPO_EVENTO,
                            I.ID AS ID_INSTITUICAO,
                            I.NOME_FANTASIA,
                            I.RAZAO_SOCIAL,
                            I.CNPJ,
                            I.LOGRADOURO,
                            I.CEP,
                            I.UF,
                            I.CIDADE
                        FROM EVENTOS E
                        INNER JOIN TIPOS_EVENTOS TE ON  E.ID_TIPO_EVENTO = TE.ID
                        INNER JOIN INSTITUICOES I ON E.ID_INSTITUICAO = I.ID";

    List<EventoDomain> listaEventos = new List<EventoDomain>();

    using(SqlConnection con = new SqlConnection(StringConexao))
    {
        con.Open();

        using(SqlCommand cmd = new SqlCommand(QuerySelect, con))
        {
            SqlDataReader sdr = cmd.ExecuteReader();

            while (sdr.Read())
            {
                EventoDomain evento = new EventoDomain
                {
                    Id = Convert.ToInt32(sdr["ID_EVENTO"]),
                    Titulo = sdr["TITULO_EVENTO"].ToString(),
                    Descricao = sdr["DESCRICAO"].ToString(),
                    DataEvento = Convert.ToDateTime(sdr["DATA_EVENTO"]),
                    AcessoLivre = Convert.ToBoolean(sdr["ACESSO_LIVRE"]),
                    TipoEventoId = Convert.ToInt32(sdr["ID_TIPO_EVENTO"]),
                    TipoEvento = new TipoEventoDomain...,
                    InstituicaoId = Convert.ToInt32(sdr["ID_INSTITUICAO"]),
                    Instituicao = new InstituicaoDomain...
                };

                listaEventos.Add(evento);
            }
        }
    }

    return listaEventos;
}
```

```
public List<JogoDomain> Listar()
{
    using (GamesContext ctx = new GamesContext())
    {
        return ctx.Jogos.Include("Estudio").ToList();
        //Explicar o include
        //return ctx.Estudios.Include("Jogos").ToList();

    }
}
```

ENTITY FRAMEWORK CORE

- Ele foi reescrito a partir do zero;
- É multiplataforma (Windows, Mac e Linux);
- É Modular e suporta diversos provedores: SQL Server, MySQL, PostgreSQL, Oracle, SQLite, SQLCompact, DB2, InMemory, Azure Table Storage, etc;
- É open source e esta disponível no GitHub;
- Pode ser usado em aplicações Windows Forms, WPF, Console, ASP .NET, ASP .NET Core, WUP, Xamarin (em breve), etc;
- Suporta as abordagens : Code First, Database First, entre outras;
- Suporta a ferramenta de linha de comando : NET Core CLI
- Pode ser instalado via Nuget : Install-Package Microsoft.EntityFrameworkCore.SqlServer

ENTITY FRAMEWORK DATABASE FIRST

A abordagem DataBase First parte de um banco de dados existente e assim você deve ter um banco de dados e as tabelas já criadas e prontas para uso.

instalar os pacotes :

Microsoft.EntityFrameworkCore.SqlServer
Microsoft.EntityFrameworkCore.SqlServer.Design
Microsoft.EnityFrameworkCore.Tools

ENTITY FRAMEWORK DATABASE FIRST

No menu Tools clique em Package Manager Console e a seguir na janela do console digite o comando:

Scaffold-DbContext "Data Source=.\SqlExpress; Initial Catalog= InLock_Games_Manha; Integrated Security=True" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Domains -ContextDir Contexts -Context BlogContext

Nesta instrução temos :

O comando: Scaffold-DbContext

A string de conexão do banco de dados

O nome do provedor usado

-OutputDir - Nome da Pasta que fica as classes

-ContextDir - Nome da pasta que fica o Contexto

-Context - Nome do arquivo de Contexto

ENTITY FRAMEWORK DATABASE FIRST

A classe ...Context é a classe responsável pela interação com os objetos de dados sendo derivada da classe System.Data.Entity.DbContext (muitas vezes referida como o contexto).

A classe de contexto administra os objetos/entidades durante o tempo de execução, o que inclui preencher objetos com dados de um banco de dados, controlar alterações, e persistir dados para o banco de dados.

DbSet é o objeto que irá referenciar as Tabelas no banco de Dados.

ENTITY FRAMEWORK CODE FIRST

Em projetos onde a arquitetura é centralizada nas classes do domínio, a base de dados geralmente é criada no final da modelagem, por meio de um processo realizado automaticamente por ferramentas específicas ou frameworks.

O Code First funciona muito bem e é muito fácil de usar.

ENTITY FRAMEWORK CODE FIRST

Crie as classes de domínio

Defina os DataAnnotatios das Propriedades

Crie o Conexto do seu Banco de dados

Adicione o Migrations Inicial

Atualize o Banco de dados