Configurar uma aplicação .Net C# para conectar ao banco de dados Oracle

Autor(es)

Thiago Keller Torquato Vicco

@2TDS-2024

Sumário

Introdução

Nessa apostila iremos criar um projeto MVC utilizando a IDE do Visual Studio e iremos configurar com o Entity Framework Core. Lembrando que essa apostila é somente um passo a passo para a configuração, nas aulas teremos explicações mais profundas e outros conteúdos, **por isso é de grande importância a sua presença nas aulas.**

O que é o Entity Framework?

O Entity Framework é um framework ORM (Object-Relational Mapping) desenvolvido pela Microsoft. Ele permite que os desenvolvedores interajam com bancos de dados relacionais usando objetos e consultas em linguagens de programação, como C#. O Entity Framework abstrai as complexidades das consultas SQL e do acesso a dados, facilitando o mapeamento entre as estruturas de dados do banco de dados e as classes de objetos da aplicação. Isso simplifica o desenvolvimento de aplicativos ao permitir que os desenvolvedores manipulem os dados como objetos no código, enquanto o framework cuida da comunicação com o banco de dados.

Instalando Packages

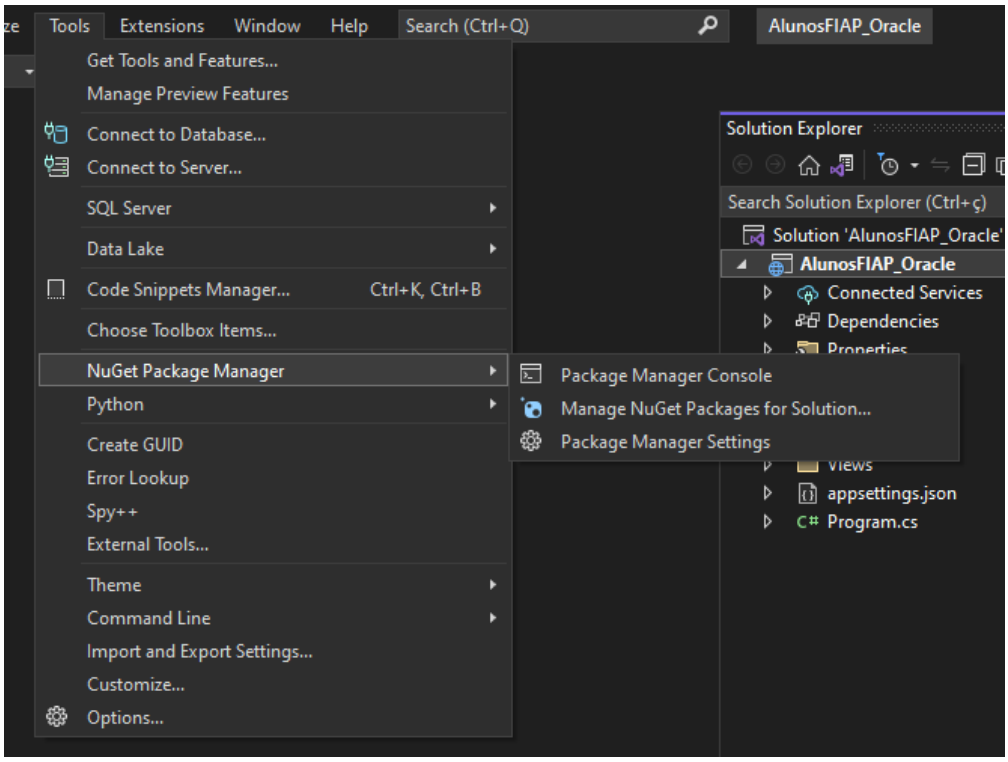
Para a conexão do banco de dados e uso do **EntityFrameworkCore** instalaremos os seguintes pacotes via nuget.

Pacotes Necessários[1]

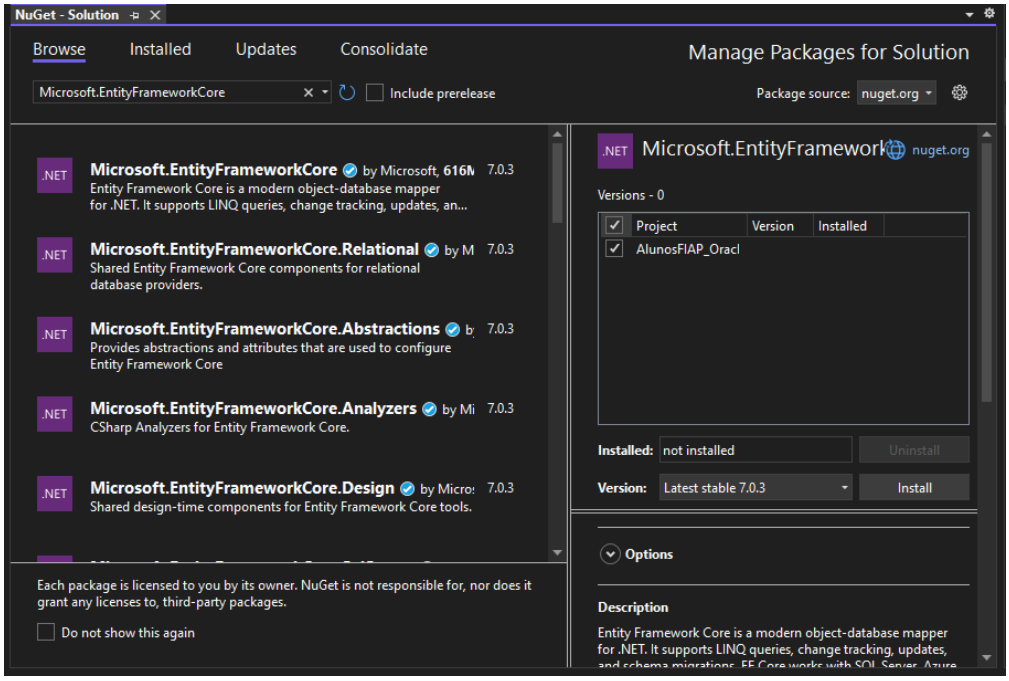
- Microsoft.EntityFrameworkCore
- Microsoft.EntityFrameworkCore.Tools
- Microsoft.EntityFrameworkCore.Design
- Oracle.EntityFrameworkCore

Instalando pacotes

Acesse o menu **"Tools/Nuget Package Manager/Manage Nuget Packages for Solution…"**



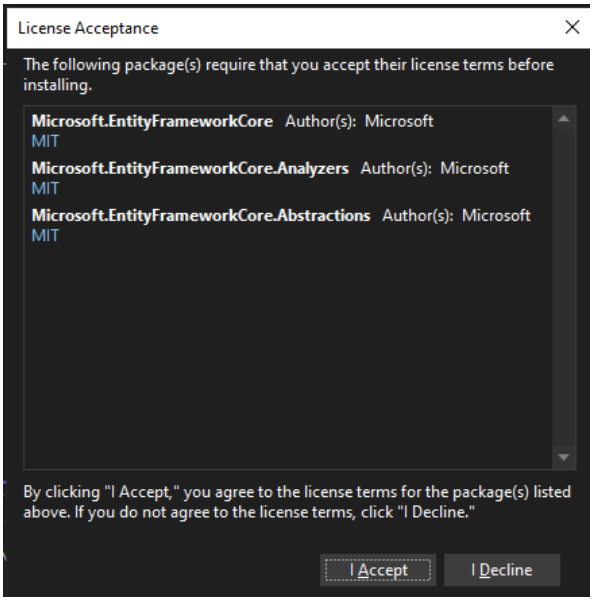
Clique na aba "**Browse**" e procure o nome do pacote conforme listagem anterior[1].



Siga os seguintes passos:

1. Selecione o pacote na listagem
2. Selecione o projeto que deseja instalar o pacote
3. Clique no botão “**Install**” ou “**Instalar**”

Obs.: Alguns pacotes poderão aparecer confirmação para instalação, aceitem todas.



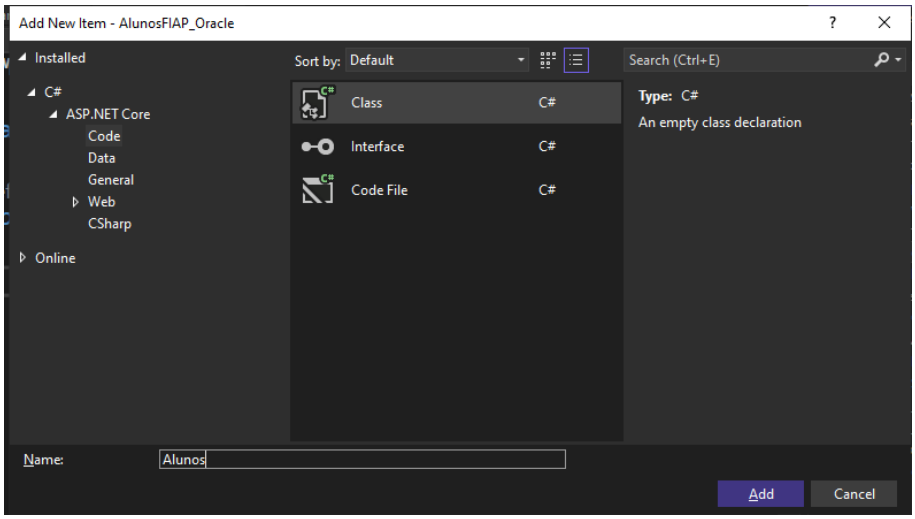
Repita isso para todos os pacotes na listagem anterior [1]

Criando uma classe

Dentro da pasta “**Models**” criaremos uma classe chamada Alunos.

Para criar a classe siga os passos:

- Clique com o botão direto do mouse sobre a pasta “**Models**”
- Selecione a opção “**Add**”
- Selecione a opção “**Class...**”
- Aparecerá uma nova tela, informe o nome e clique em “**Add**”




```
[Table("Alunos")]
public class Aluno
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public int AlunoId { get; set; }

    [Required(ErrorMessage = "O nome é obrigatório.")]
    public string Nome { get; set; }

    [Column("Email", TypeName = "varchar(255)")]
    [EmailAddress]
    public string Email { get; set; }

    [MaxLength(11, ErrorMessage = "O CPF deve conter no máximo 11
caracteres.")]
    public string CPF { get; set; }
```

```
    [Range(1, 8, ErrorMessage = "O semestre deve estar entre 1 e 8.")]
    public int Semestre { get; set; }

    [NotMapped]
    public string InformacaoTemporaria { get; set; }

    [ForeignKey("CursoId")]
    public int CursoId { get; set; }

    [DecimalPrecision(5, 2)]
    public decimal MediaNotas { get; set; }

}
```

No modelo que irão construir podemos ter quantos atributos acharem necessário. Os "**Data Annotations"** irão variar conforme o atributo. Para mais informações sobre acesso o link: System.ComponentModel.DataAnnotations Namespace | Microsoft Learn

Criando um DBContext

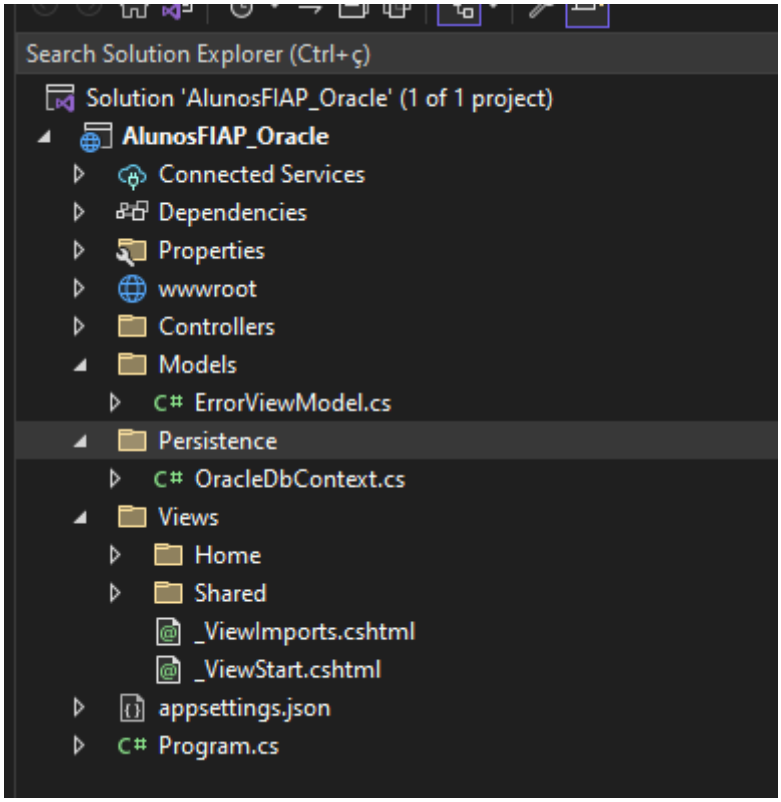
O DbContext é uma classe central no Entity Framework (EF) e Entity Framework Core (EF Core), atuando como um canal entre o código C# de sua aplicação e o banco de dados. É parte do namespace Microsoft.EntityFrameworkCore e serve para configurar o modelo de dados, realizar consultas, e salvar alterações no banco de dados. O DbContext encapsula uma sessão com o banco de dados, oferecendo uma API simplificada para executar operações CRUD (Criar, Ler, Atualizar, Deletar) nas entidades mapeadas.

Principais Características e Funcionalidades do DbContext:

- **Configuração de Modelo**: Define como as classes de entidade são mapeadas para as tabelas do banco de dados, incluindo chaves primárias, relações, índices, e convenções de nomenclatura. Essa configuração pode ser feita por meio de Data Annotations diretamente nas classes de entidade ou através do Fluent API no próprio DbContext.

- **Rastreamento de Mudanças**: O DbContext rastreia as alterações feitas nas instâncias das entidades desde a última vez que foram carregadas ou desde a última chamada de SaveChanges(). Esse rastreamento permite que o EF Core envie comandos SQL eficientes para o banco de dados, atualizando apenas as partes que realmente mudaram.
- **Execução de Consultas**: Permite a execução de consultas LINQ para selecionar, filtrar, ordenar e agrupar dados de maneira eficiente. As consultas são traduzidas pelo EF Core em comandos SQL otimizados para o banco de dados em uso.
- **Gerenciamento de Transações**: O DbContext usa transações para garantir a consistência dos dados. Ele pode gerenciar transações automaticamente ou permitir que você as controle manualmente para operações mais complexas.
- **Caching de Primeiro Nível**: O EF Core armazena instâncias de entidades em cache no contexto, o que pode melhorar o desempenho ao reutilizar entidades já carregadas em operações subsequentes dentro do mesmo contexto.
- **Migrações de Banco de Dados:** O DbContext trabalha com o sistema de migrações do EF Core para aplicar mudanças incrementais no esquema do banco de dados, facilitando o controle de versão e a evolução do modelo de dados.

Para ficar mais organizado criaremos uma pasta no projeto chamada "**Persistence**" e dentro dessa pasta criaremos uma classe **OracleDbContext** que herda de **DbContext** e incluiremos as definições das tabelas e relacionamentos, conforme necessário.



Mapeando a Classe

O DbSet<T> é uma classe fundamental no Entity Framework (EF) e no Entity Framework Core (EF Core), representando uma coleção de entidades de um determinado tipo T que podem ser consultadas ou atualizadas. Ele é parte do namespace Microsoft.EntityFrameworkCore e desempenha um papel crucial na interação entre o código da aplicação e o banco de dados, atuando como um ponto de entrada para as operações de CRUD (Criar, Ler, Atualizar, Deletar) nas entidades mapeadas.

```
0 references
public class OracleDbContext : DbContext
{
    0 references
    public DbSet<Alunos> Alunos{ get; set; }

}
```

Características Principais do DbSet<T>:

- **Consulta de Dados:** O DbSet<T> oferece suporte a operações de consulta LINQ, permitindo que desenvolvedores escrevam consultas de forma expressiva e em alto nível, que são traduzidas pelo EF Core para SQL. Isso inclui operações para filtrar, ordenar, agrupar e unir dados, facilitando o trabalho com dados relacionais de maneira eficiente.
- **Rastreamento de Entidades:** Quando uma entidade é obtida através de um DbSet, o EF Core por padrão rastreia suas alterações. Isso significa que, ao chamar o método SaveChanges() no contexto, o EF Core gera e executa comandos SQL para persistir quaisquer alterações feitas nas entidades desde o momento em que foram carregadas.
- **Operações CRUD:** Além das consultas, o DbSet<T> facilita a criação, atualização e exclusão de entidades. Métodos como Add(), Remove(), e Update() são usados para manipular as entidades na coleção. O EF Core então sincroniza estas alterações com o banco de dados ao salvar o contexto.
- **Carregamento de Relacionamentos:** O DbSet<T> suporta operações para carregar explicitamente relacionamentos entre entidades, como o carregamento preguiçoso (lazy loading), carregamento antecipado (eager loading) e carregamento explícito (explicit loading). Esses mecanismos ajudam a controlar como e quando os dados relacionados são carregados do banco de dados, otimizando o desempenho e a utilização de recursos.

Criando o Construtor

Na mesma classe **OracleDbContext** crie um construtor conforme imagem abaixo.

```
0 references
public OracleDbContext(DbContextOptions<OracleDbContext> options) : base (options)
{

}
```

O construtor do OracleDbContext, ou de qualquer classe que herde de DbContext no Entity Framework Core, é essencial para configurar como sua aplicação interage com o banco de dados. Ele é usado para passar configurações específicas para o DbContext através de uma instância de DbContextOptions. Este mecanismo permite configurar detalhes importantes como a string de conexão do banco de dados, o provedor de banco de dados (Oracle, SQL Server,

PostgreSQL, SQLite, etc.), comportamentos de logging, e outras opções de performance e comportamento.

Nossa classe **OracleDbContext** deverá ficar desta forma:

```
2 references
public class OracleDbContext : DbContext
{
    0 references
    public DbSet<Alunos> Alunos{ get; set; }

    0 references
    public OracleDbContext(DbContextOptions<OracleDbContext> options) : base (options)
    {

    }
}
```

Por que é importante?

O construtor e a passagem de **DbContextOptions** são cruciais porque permitem que sua aplicação configure e customize a conexão e o comportamento do **DbContext** com o banco de dados em tempo de execução. Essas opções são normalmente configuradas no início da aplicação, por exemplo, no método **ConfigureServices** em uma aplicação ASP.NET Core, onde você pode especificar qual banco de dados usar, a string de conexão, e outras opções específicas do provedor de banco de dados.

E as minhas configurações??

No desenvolvimento de aplicações .NET e .NET Core, o gerenciamento de configurações é um aspecto crucial que permite definir e acessar variáveis de configuração de maneira flexível e segura. O ASP.NET Core introduziu uma nova abordagem para lidar com configurações através de arquivos JSON, como appsettings.json e appsettings.<Environment>.json, onde

<Environment> representa o ambiente específico em que a aplicação está sendo executada, como Development, Staging ou Production.

appsettings.json

O arquivo appsettings.json é o arquivo de configuração padrão usado por uma aplicação .NET Core. Ele armazena configurações na forma de pares chave-valor e é carregado automaticamente pelo framework no início da aplicação. Este arquivo é utilizado para definir configurações que são comuns a todos os ambientes de execução da aplicação, como strings de conexão de banco de dados (em alguns casos), configurações de log, e outras preferências da aplicação.

appsettings.<Environment>.json

Para permitir configurações específicas de cada ambiente, como Development, Staging ou Production, o ASP.NET Core suporta arquivos de configuração nomeados como appsettings.<Environment>.json. Por exemplo, appsettings.Development.json e appsettings.Production.json. Esses arquivos sobrescrevem as configurações definidas em appsettings.json para o ambiente específico, permitindo que você personalize configurações, como strings de conexão ou níveis de log, para diferentes ambientes sem alterar o código.

String de Conexão

No arquivo **appsettings.development.json** do seu projeto, adicione a seguinte configuração para a conexão com a base de dados **Oracle**:

```
"ConnectionStrings": {
    "OracleConnection": "Data
    Source=(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=<HOST>)(P
    ORT=<PORT>)))(CONNECT_DATA=(SERVICE_NAME=< SERVICE_NAME >)));User
    ID=<USERID>;Password=<PASSWORD>;"
}
```

Obs.: Certifique-se de substituir <HOST>, <PORT>, <SERVICE_NAME>, <USERID> e <PASSWORD> com as informações da sua base de dados Oracle.

Mapeando o DbContext

Após a criação do **DbContext** e da classe que iremos persistir no banco de dados. Chegou a hora de informar na inicialização do projeto quais serão os bancos de dados que utilizaremos. **Sim, nós podemos ter vários bancos de dados na mesma aplicação**.

No arquivo **Program.cs** dentro da raiz do seu projeto, iremos adicionar um novo serviço.

```
builder.Services.AddDbContext<OracleDbContext>(options =>
{
    options.UseOracle(builder.Configuration.GetConnectionString("OracleConnection"));
});
```

Próximos passos

Nas próximas aulas nós criar nosso banco de dados.

Não percam!!!!

Links interessantes

Curso Online Entity Framework Core: banco de dados de forma eficiente | Alura

Visão geral do Entity Framework Core – EF Core | Microsoft Learn

Se chegaram a essa parte com sucesso, meus parabéns!! Em breve iremos melhorar essa aplicação.

LET’S ROCK THE FUTURE