

# Entity Framework


Leonardo Sibela  
Maurício das Neves  
Myrna Bonturi

# O que é o Entity

O Entity é uma ferramenta classificada como ORM (Mapeamento Objeto-Relacional)

Ele permite que os desenvolvedores trabalhem com dados na forma de objetos específicos de domínio (e as propriedades), como clientes e endereços de clientes, sem precisar se preocupar com as tabelas de banco de dados e colunas onde esses dados são armazenados de base .

Com o Entity Framework, os desenvolvedores podem trabalhar em um nível mais alto de abstração quando eles lidam com dados e podem criar e manter aplicativos orientados objetos com menos código do que em aplicativos tradicionais.



# O que é o Entity

De forma resumida, é o Entity quem faz com que as operações de banco (select, insert, update, delete) sejam realizadas de forma fácil de desenvolver

Com isso, não podemos nos preocupar muito com queries sql

Além disso, a questão de segurança também fica por conta do Entity

**OBS:** Um dos problema básicos de segurança é o SQL-Injection





# Criando um Banco de Dados

# Criando um banco de dados no Visual Studio

O Visual Studio possui uma **versão simplificada do SQL Server** embutida

Nela podemos criar bases e tabelas de forma bem simples

Para acessar essa ferramenta, precisamos ir em **view -> SQL Server Explorer**

Nesta tela iremos encontrar um servidor onde iremos poder criar uma nova base

Para isso, devemos clicar com o **botão direito** no **diretório Databases**

Em seguida, basta clicar em **Add New Database**, escolher um **nome** e clicar no **botão Ok**



# Criando as tabelas

Em seguida, podemos criar as tabelas nesse banco de dados

Basta clicar com o **botão direito** no **diretório Tables** da base recém criada e clicar na opção “**Add New Table...**”

Na aba T-SQL, altere o nome da tabela → CREATE TABLE [dbo].[**altrar\_nome**]

Em seguida, crie as colunas que desejar informando Nome, tipo e se pode ser nula

No Painel de Propriedades escolha qual coluna da tabela será a Primary Key selecionando essa coluna na propriedade **Identity Column**




# T-SQL?

É comum estranhar o script gerado pelo SQL Server, pois ele não é um SQL normal, mas sim uma implementação do SQL da Microsoft (Transact-SQL)

Por isso, ele irá ficar algo parecido com isso:

```
CREATE TABLE [dbo].[User] (  
    [Id] INT NOT NULL PRIMARY KEY IDENTITY,  
    [mail] VARCHAR(50) NOT NULL,  
    [password] VARCHAR(50) NOT NULL  
)
```

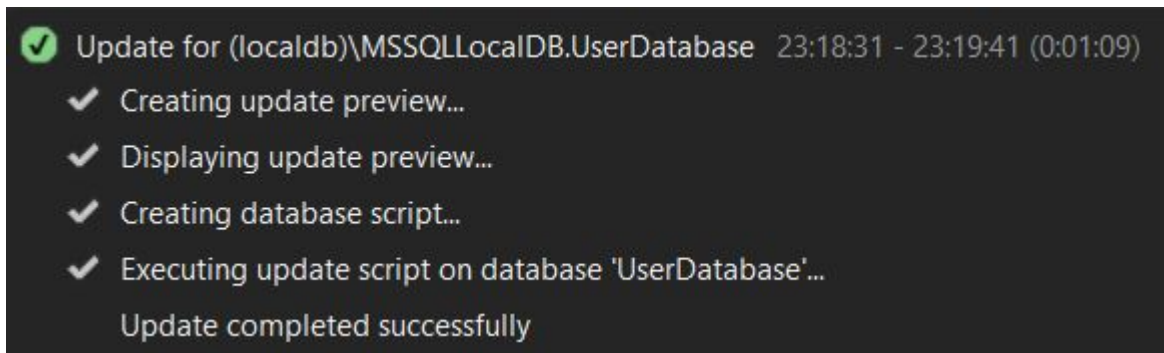


# Criando as tabelas

Após ter definido nome, colunas e propriedades da table, clicar em  Update

O Visual Studio irá abrir a tela “**Preview Database Updates**”

Para concluir a criação da tabela, basta clicar no botão “**Update Database**”







Criando uma conexão

# Como criar uma conexão

Crie um novo item dentro do pacote model do tipo **ADO.NET** e coloque o nome do arquivo de **Conexao**

Escolha a opção “**Code First from database**”, clique em “**Next**”

Isso irá permitir criar a nossas classes de modelo baseadas em nossas tabelas

Na próxima tela, escolha a opção “**New Connection**”



# Como criar uma conexão

**Data Source:** Microsoft SQL Server (SqlClient)

**Server Name:** (localdb)\MSSQLLocalDB ← [DIGITAR]

Clique em **“Refresh”**

Em **“Connect to a Database”**, escolha a opção **“Select or enter a database name”** e escolha a base de dados desejada

Se quiser, clique em **“Test Connection”** para ver se está tudo funcionando

Em seguida, clique em **OK**



# Como criar uma conexão


Vemos que o Visual Studio gerou a nossa String de Conexão

Feito isso, podemos clicar em “**Next**” e ir para a próxima tela

Por fim, podemos escolher as tabelas que desejamos gerar nossas classes

Ao seleccionar os itens desejados, basta clicar no botão “**Finish**”

Esse processo irá criar uma classe de conexão e, caso tenha sido selecionada apenas uma tabela, será criada uma classe de modelo que irá representar (no C#) a tabela existente no banco de dados



An illustration featuring a person with dark hair and a beard, wearing a brown jacket, sitting at a desk and looking at a laptop. The laptop screen displays a simplified version of the server racks shown in the background. The background consists of three teal server racks, each with three indicator lights (two dark, one white). The scene is set against a light brown background with faint circuit-like patterns. A large, semi-transparent blue circle is centered behind the text.

Fazendo um CRUD

# Usando a classe de conexão

Para trabalhar com nossa classe de conexão, teremos que chamar a classe, pegar o atributo do objeto que queremos persistir e chamar o método que realiza a ação (update, delete, insert, select) desejada

Dessa forma, se eu criei minha classe de conexão (chamada **Conexao**) e o objeto que eu quero trabalhar é o **livro**, eu posso **selecionar todos** os livros da seguinte forma:

```
Conexao conexao = new Conexao();  
conexao.Livros.All();
```



# Insert

Para inserir dado (do tipo livro) na base, basta eu chamar o método Add do meu objeto conexão e passar o livro como parâmetro.

Além disso, é necessário chamar o método SaveChanges, para commitar o insert

```
Livro livro = new Livro("Eu, Robô", "Isaac Asimov", "Ficção científica");  
conexao.Livros.Add(livro );  
conexao.SaveChanges();
```




# Update

Existem várias formas de atualizar um dado, sendo a mais simples usando o método `AddOrUpdate`, que pode também ser usado para adicionar

Para fazer uso dele, é necessário usar o pacote `Migrations` do Entity

```
using System.Data.Entity.Migrations;
```

```
livro.Autor = "Dom Casmurro";  
conexao.Livros.AddOrUpdate(livro );  
conexao.SaveChanges();
```





# Delete

Para remover um dado na base, basta eu chamar o método Remove do meu objeto conexão e passar o livro como parâmetro

Além disso, é necessário chamar o método SaveChanges, para commitar o delete

```
conexao.Livros.Remove(livro );  
conexao.SaveChanges();
```



# Select \*

Para selecionar todos os elementos de uma tabela (livros), basta eu pegar meu objeto de conexao, pegar o atributo Livros e chamar o método All

```
Conexao conexao = new Conexao();
```

```
conexao.Livros.All();
```



# select por ID

Podemos selecionar um único dado (linha) de uma tabela através do método find

Ele receberá o ID (primary key) do banco e irá devolver o objeto que tenha esse id

```
Conexao conexao = new Conexao();  
Livro livro = conexao.Livros.Find(1);
```

Obs: O id deve ser do tipo int, portanto, caso esse número venha de uma TextView, ele deve ser convertido de string para int

