Aula 5

Programação I

Prof. Alan Matheus Pinheiro Araya

1

Conversa Inicial

2

Interações com banco de dados

- **Entity Framework (EF Core)**
- **Dapper**

3

Introdução ao Entity Framework

4

O Entity Framework

- **"Mini Framework": inspirado no Hibernate (Java)**
- **ORM (*object relational model*): traduz a modelagem de objetos (orientação a objetos) em modelos relacionais (bancos de dados)**
- **Pode trabalhar com uma ampla gama de bancos, até mesmo os não relacionais**

5

- **Utiliza seus modelos de objetos C# para criar uma "espelho" dessa modelagem no banco**
- **O EF faz isso convertendo:**
  - **Objetos -> Tabelas**
  - **Propriedades -> Colunas**
  - **Relacionamentos -> *Constraints***
  - ***Querys LINQ* -> *Query SQL***
- **O desenvolvedor escreve a *query* em LINQ e o EF traduz ela para o SQL correto do seu banco (independente de qual ele seja)**

6

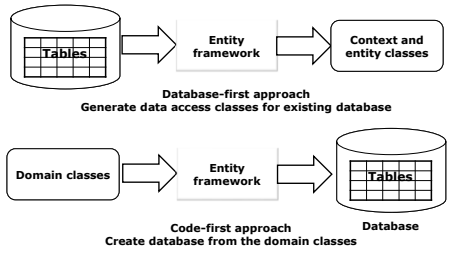
- O EF pode ser utilizado através do pacote Nuget "Microsoft.EntityFrameworkCore"
- Equivalência entre estruturas de banco de dados com os objetos em C# para o EF:

Base Relacional	.NET/C#
Tabela	Classe
Colunas de tabelas	Propriedades
Linhas de tabelas	Elementos de uma coleção, por ex: linhas de uma List<>
Chaves Primárias (Primary keys)	Uma única instância de uma classe
Chaves Estrangeiras (Foreign Key)	Referência para outra classe
Comandos SQL, como WHERE (por exemplo)	Operadores LINQ (Where(x=> .....))

7

- Modelos de utilização do EF:
  - Database First – cria-se primeiro o modelo relacional no banco de dados e depois os objetos C# manualmente
  - Code First – cria-se primeiro o modelo de orientação a objetos e, a partir dele, gera-se os modelos do banco de dados

8



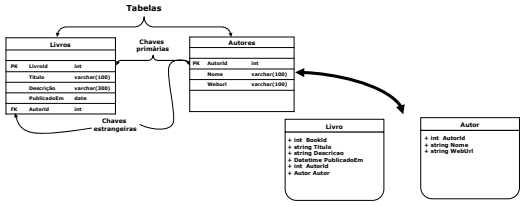
9

Modelagem de objeto *versus* Modelagem Relacional

- Modelagem de objetos != modelagem relacional
  - Polimorfismo
  - Herança
  - Isso não é exatamente um problema
- Objetos que podem ser mapeados são chamados de "Entidades"

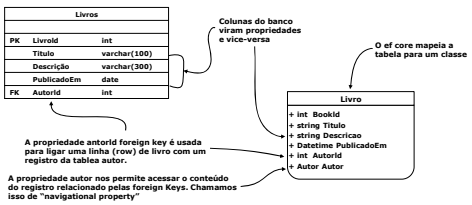
10

- Vamos modelar um cenário de exemplo para analisar as equivalências entre os modelos (relacional *versus* objetos):



11

- Vamos modelar um cenário de exemplo para analisar as equivalências entre os modelos (relacional *versus* objetos):



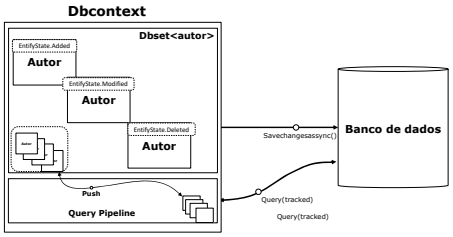
12

DbContext

- DbContext é a principal classe do EF para manipulação das entidades
  - É como um grande "Repositório"
- Alterações nas entidades devem ser "enviadas" ao DbContext
  - Converte em SQL
  - Traduz querys LINQ em SQL

13

DbContext



14

- O DbContext é uma classe abstrata. Logo, precisamos herdar dela e implementar alguns métodos para usá-lo em nossa aplicação

Seu contexto (herdando de DbContext):

```
public class ApplicationContext : DbContext
{
    private const string stringDeConexao = @"Server=database-aula.cg8lcumyqg6o.us-
east2.rds.amazonaws.com;Port=3306;Database=dbaula;User Id=admin;Password=***";

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
        var serverVersion = new MySqlServerVersion(new Version(8, 0, 25));

        optionsBuilder.UseMySql(stringDeConexao, serverVersion)
                      .EnableDetailedErrors();
    }
}
```

String de conexão

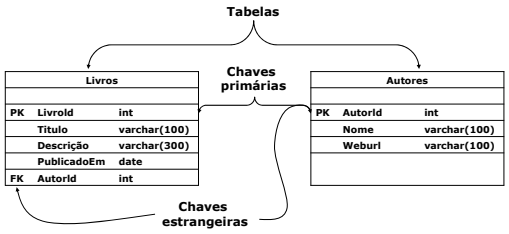
Inicializando conexão com MySql

15

Mapeando o modelo

- Deve-se modelar as entidades mais próximas do modelo Orientado a Objetos possível
- O EF se encarregará de traduzir esse modelo para SQL e vice-versa
  - EntityConfiguration

16



17

Operações de CRUD no EF

18

Operações de CRUD

- Todas as operações passam pelo DbContext
  - Comandos de Insert/Update/Delete (commands) somente são enviados ao banco via "SaveChangesAsync"
  - Querys (read) são realizadas "na hora" (a depender do operador LINQ)
    - Tracked
    - NotTracked

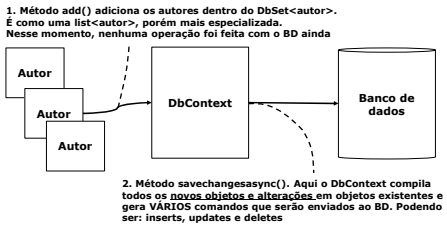
19

Insert/create

- Vamos realizar um procedimento de Create (insert) juntos?

20

Insert/create



21

Update

- Vamos realizar um procedimento de update juntos?

22

- O EF possuí vários estados para as entidades, são eles:
  - Added: as entidades são novas e ainda não foram inseridas no banco de dados
  - Unchanged: as entidades não foram alteradas desde que foram consultadas do banco de dados (estado inicial)

23

- Modified
  - As entidades foram alteradas depois que foram consultadas do banco de dados
- Deleted
  - As entidades existem no banco de dados, mas já estão marcadas para serem excluídas quando SaveChanges for chamado
- Detached
  - As entidades não estão sendo acompanhadas (tracked) pelo DbContext

24

Delete/remove

- Vamos realizar um procedimento de remove juntos?

25

Select/read

- O EF permite dois modelos básicos de consultas ao banco:
  - Querys LINQ
    - São traduzidas de C# para SQL pelo EF
  - Querys diretas
    - Pode-se escrever comandos SQL e o EF apenas converte seu retorno em objetos

26

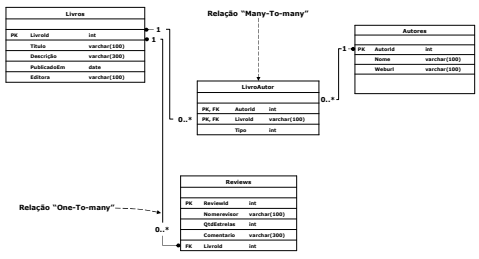
- Vamos ver um exemplo prático de query com EF?

27

Relações no EF

28

Melhorando o modelo



29

- Principais diferenças entre relações de OO e ER:
  - One-To-Many (um para muitos)
    - Relação entre uma entidade que sempre existirá e outra que pode existir. Sendo a segunda uma ou mais instâncias
  - Many-To-Many (muitos para muitos)
    - Utilizada para presentar uma relação onde uma instância da entidade "1" pode se relacionar a mais de uma instância da entidade "2" e vice-versa

30

- **One-To-One (um para uma)**
  - **Relação entre duas entidades onde as duas sempre existem**
  - **Não possui uma correspondência suportada diretamente pela modelagem ER**

31

32

Querys em modelos "complexos" no EF

33

Montando o cenário

- **1. Listar todos os autores baseado nas reviews de seus livros, ordenando do autor mais bem avaliado para o menor**
- **2. Listar os livros com as melhores reviews (de 3 a 5 estelas) e agrupar por ano**
- **3. Top 3 livros menos avaliados e seus comentários de avaliação (review)**

34

- **Vamos ver como construir essas as querys para essas perguntas?**

35

Dapper e o ADO.NET

36

Além do EF

- O EF não é a única forma de acesso a banco pelo C#
- O ADO.NET foi a primeira tecnologia para acesso de banco de dados
- Existem outros mini-frameworks que implementam melhorias e formas de acesso ao banco via ADO.NET

37

Dapper

- "Micro-ORM"
- Projeto opensource, gratuito e disponível no Github
- Foco em performance
  - Excelente alternativa para quem precisa escrever querys SQL diretamente no código
- Não suporta "Migrations" e não gera o banco a partir do código (modelo OO)

38

- Não suporta o ChangeTracking de entidades como o EF
- Mapeia entidade para uma query e vice-versa
- Pouca ou nenhuma necessidade de configuração inicial
  - Com o namespace do Dapper no projeto, já é possível utilizá-lo
  - Provê seus métodos por meio de métodos de extensão

39

- Principais métodos para uso do Dapper:
  - Execute / ExecuteAsync
    - Executa um comando no banco de dados e retorna a quantidade de linhas afetadas
    - Pode ser um insert/update ou delete
  - ExecuteScalar / ExecuteScalarAsync
    - Executa um comando no banco de dados e retorna um valor único de retorno
    - Pode ser, por exemplo, o ID gerado pelo Insert de uma IdentityColumn

40

- Query / QueryAsync
  - Executa um comando no banco de dados e retorna um IEnumerable<T>
- QueryMultiple / QueryMultipleAsync
  - Executa mais de um comando simultaneamente dentro da mesma query, como dois selects ou uma StoreProcedure
  - Possibilita o mapeamento de mais de um "result set" no retorno

41

- Vamos ver como utilizar o dapper para alguns comandos de CRUD em nosso modelo anterior?

42