Aline N. Favretto

Iniciando com Entity Framework

Aline N. Favretto

Desenvolvedora .NET
Estudante de Ciência da Computação na Unisinos
Cofundadora do .NET Caxias

Experiência com ASP.NET Web Forms e MVC, C#, DevExpress, Crystal Reports, Sql Server e Oracle.

.NET Caxias

Patrocinadores

Agenda

- O que é o EF?
- Como utilizar?
  - Modos de utilização
  - Mapeamento
- Loading Types
  - Lazy Loading
  - Eager Loading
  - Explicit Loading
- Vantagens e Desvantagens
- DEMO

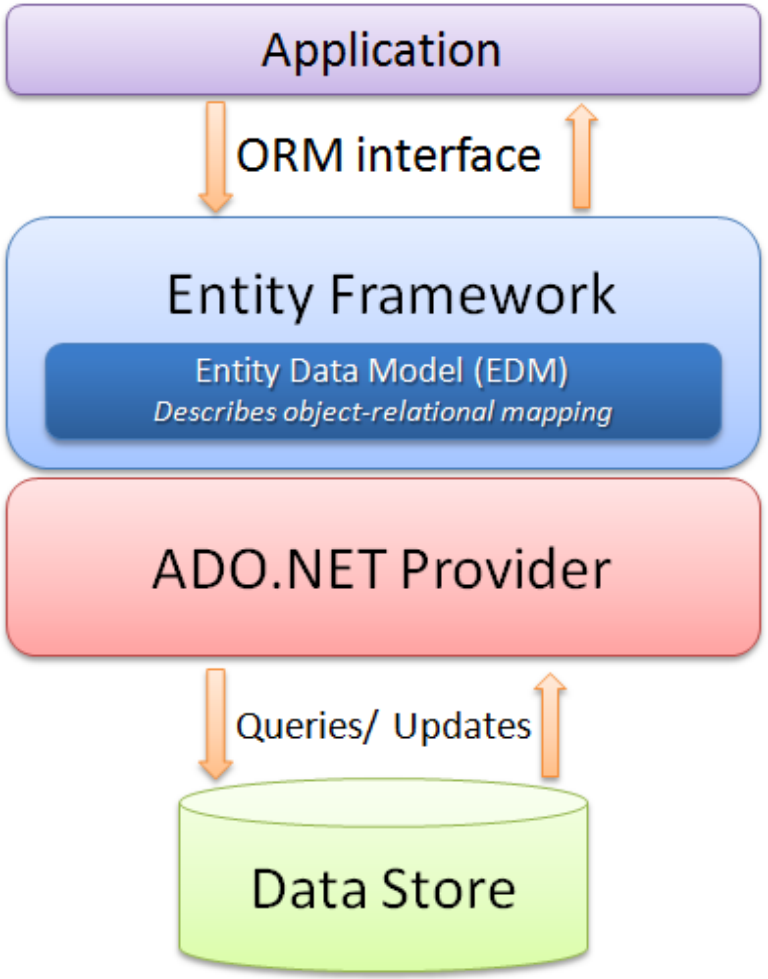
Entity Framework

O que é o EF?

ORM (Object Relational Mapping)

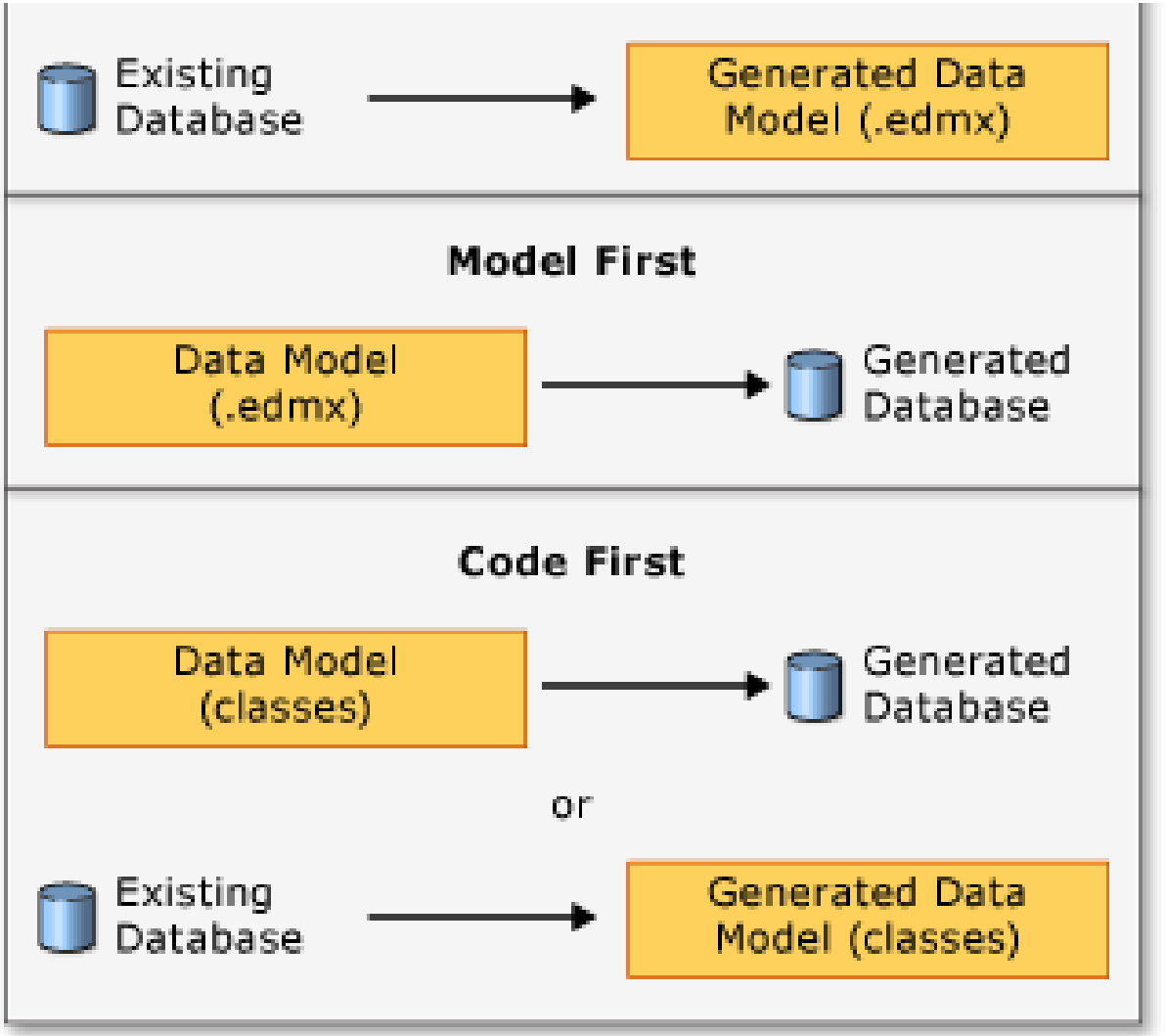
Suporta diversos bancos de dados relacionais

- SQL Server
- Oracle
- DB2
- MySQL



Como utilizar?

Modos de utilização



DbContext

```
public class EFContext : DbContext
{
    public EFContext()
                : base("Contexto")
    {
        Configuration.LazyLoadingEnabled = false;
        Configuration.ProxyCreationEnabled = false;
    }

    public DbSet<Cliente> Cliente { get; set; }
    public DbSet<Compra> Compra { get; set; }
    public DbSet<Faturamento> Faturamento { get; set; }
    public DbSet<Fornecedor> Fornecedor { get; set; }
```

DbContext

```
protected override void OnModelCreating(DbModelBuilder modelBuilder)
{
    modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();

    modelBuilder.Conventions.Remove<OneToManyCascadeDeleteConvention>();

    modelBuilder.Conventions.Remove<ManyToManyCascadeDeleteConvention>();

    modelBuilder.Properties<string>().Configure(p => p.HasColumnType("varchar"));

    modelBuilder.Properties<string>().Configure(p => p.HasMaxLength(150));

    modelBuilder.Properties()
        .Where(p => p.Name == "Id" + p.ReflectedType.Name)
        .Configure(p => p.IsKey());

    modelBuilder.Configurations.Add(new ClienteMap());
    modelBuilder.Configurations.Add(new CompraMap());
    modelBuilder.Configurations.Add(new FaturamentoMap());
    modelBuilder.Configurations.Add(new FornecedorMap());
```

Mapeamento

Fluent API

```
public class ClienteMap : EntityTypeConfiguration<Cliente>
{
    public ClienteMap()
    {
        ToTable("Cliente");
        HasKey(c => c.Id);
        Property(u => u.Id).HasDatabaseGeneratedOption(DatabaseGeneratedOption.Identity);
        Property(u => u.Nome).HasMaxLength(250).IsRequired();
        Property(u => u.Login).HasMaxLength(50).IsRequired();
        Property(u => u.Senha).HasMaxLength(200).IsRequired();
        Property(u => u.Email).HasMaxLength(150).IsRequired();
        Property(c => c.Cpf).IsRequired();
        Property(c => c.Rg).IsRequired();
    }
}
```

```
Property(n => n.Impostos).HasPrecision(10, 3).IsRequired();
Property(n => n.ValorTotal).HasPrecision(10, 3).IsRequired();
```

Data Annotations

```
public class Cliente
{
    public Cliente() { }

    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    public long Id { get; set; }

    [MaxLength(250)]
    [Required]
    public String Nome { get; set; }

    [MaxLength(200)]
    [Required]
    public String Senha { get; set; }
```

Loading Types

Lazy Loading

```
public class Endereco
{
    public long IdEndereco { get; set; }

    public string Rua { get; set; }

    public int Numero { get; set; }

    public string Bairro { get; set; }

    public virtual Cidade Cidade { get; set; }
    public long IdCidade { get; set; }

    public long CEP { get; set; }

    public string Complemento { get; set; }

}
```

Lazy Loading

```
public class EFContext : DbContext
{
    public EFContext(): base("Contexto") {
        Configuration.LazyLoadingEnabled = true;
        Configuration.ProxyCreationEnabled = true;
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }
```

Eager Loading

```
public class Compra
{
    public Compra() { }

    public long IdCompra { get; set; }
    public int NumeroNF { get; set; }
    public DateTime Data { get; set; }
    public StatusCompra Status { get; set; }
    public List<PedidoItemFornecedor> Pedidos { get; set; }
    public List<ExcecaoNF> Excecoes { get; set; }
}
```

Eager Loading

```
public class PedidoItemFornecedor
{
    public PedidoItemFornecedor() { }

    public long IdPedidoItemFornecedor { get; set; }
    public long IdProduto { get; set; }
    public Produto Produto { get; set; }
    public long IdFornecedor { get; set; }
    public Fornecedor Fornecedor { get; set; }
    public int Quantidade { get; set; }
    public DateTime DataPrevista { get; set; }
}
```

Eager Loading

```
public Compra BuscarCompra(long IdCompra)
{
    var res = contexto.Compra
        .Include("Pedidos")
        .Include("Pedidos.Produto")
        .Include("Pedidos.Fornecedor");

    return res.FirstOrDefault(c=>c.IdCompra == IdCompra);
}
```

Explicit Loading

```
public PedidoItemFornecedor BuscarPorId(long Id)
{
    var pedido = contexto.PedidoItemFornecedor
        .FirstOrDefault(c => c.IdPedidoItemFornecedor == Id);

    contexto.Entry(pedido).Reference(p => p.Produto).Load();

    return pedido;
}
```

Explicit Loading

```
public Compra BuscarPorId(long Id)
{
    var compra = contexto.Compra
        .FirstOrDefault(c => c.IdCompra == Id);

    contexto.Entry(compra).Collection(c => c.Pedidos).Load();
    return compra;
}
```

Vantagens e Desvantagens

Vantagens

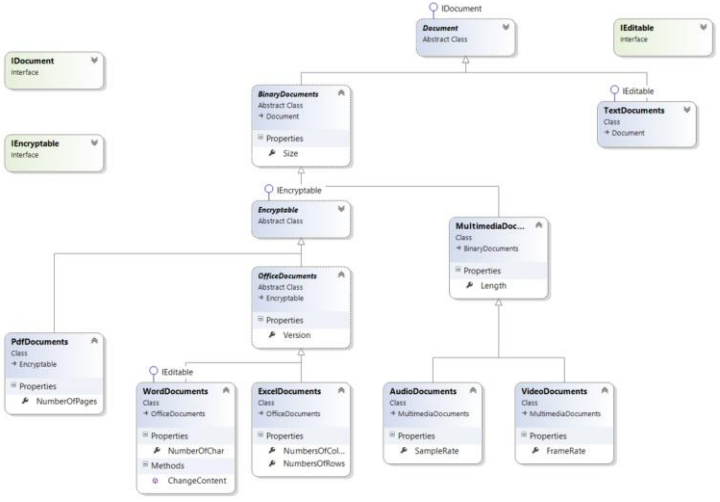
Evitar erros identificados apenas em tempo de execução, ocasionados por alterações nas tabelas.

Desvantagens

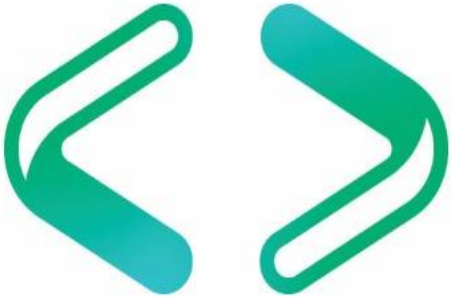
Criação de *stored procedures* mais complicada

Menor capacidade de otimizar as *queries* geradas

Criação de classes desnecessárias na importação de bancos de dados existentes



DEMO



PASS

SQLSATURDAY

BRASILIA | 19 AUG 2017

http://www.sqlsaturday.com/618

30/09 – São Paulo
21/10 – Rio de Janeiro
18/11 - Salvador

Apoio

Uniftec
centro universitário

Trino Polo
Polo de TI da Serra Gaúcha

SEPRORGS