Desenvolvimento de aplicações web com Asp.Net WebApi

Aula 21

EntityFramework

Tecnologia .NET para acesso a bases de dados utilizando o padrão **ORM (Mapeamento Objeto Relacional)**, ou seja, o EntityFramework é capaz de mapear as classes de entidade do projeto de forma que estas sejam interpretadas como tabelas do banco de dados.

O/R Mapping Object Object Object Object Objects in Memory Relational Database

Para mapear as classes de entidade no projeto DAL, criamos a estrutura abaixo:

- 1.3 Camada de Acesso a Dados
 - ▲ C# Projeto.DAL
 - Properties
 - ▶ ■ Referências
 - Configurations

 - ▶ C* FuncaoConfiguration.cs
 - ▶ C# FuncionarioConfiguration.cs
 - D Context
 - Contracts
 - Repositories
 - App.config
 - packages.config

^{**} Cada classe "Configuration" criada acima está mapeando uma entidade do sistema.



using System;

Treinamento em C# WebDeveloper Terça-feira, 30 de Maio de 2019

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity.ModelConfiguration; //importando
using Projeto.Entities; //entidades
namespace Projeto.DAL.Configurations
   public class DependenteConfiguration
        : EntityTypeConfiguration<Dependente>
   {
        //construtor -> ctor + 2x[tab]
        public DependenteConfiguration()
            //nome da tabela
            ToTable("DEPENDENTE");
            //chave primária
            HasKey(d => new { d.IdDependente });
            //demais campos
            Property(d => d.IdDependente)
                .HasColumnName("IDDEPENDENTE");
            Property(d => d.Nome)
                .HasColumnName("NOME")
                .HasMaxLength(150)
                .IsRequired();
            Property(d => d.DataNascimento)
                .HasColumnName("DATANASCIMENTO")
                .IsRequired();
            //Mapeamento do relacionamento
            HasRequired(d => d.Funcionario)
                .WithMany(f => f.Dependentes)
                .Map(map => map.MapKey("IDFUNCIONARIO"))
                .WillCascadeOnDelete(false);
        }
   }
}
-----
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando
using System.Data.Entity.ModelConfiguration; //importando
namespace Projeto.DAL.Configurations
    //classe de mapeamento para a entidade Funcao
   public class FuncaoConfiguration
        : EntityTypeConfiguration<Funcao>
```



```
{
        //construtor -> ctor + 2x[tab]
        public FuncaoConfiguration()
            //nome da tabela
            ToTable("FUNCAO");
            //chave primária
            HasKey(f => new { f.IdFuncao });
            //mapear os campos
            Property(f => f.IdFuncao)
                .HasColumnName("IDFUNCAO");
            Property(f => f.Nome)
                .HasColumnName("NOME")
                .HasMaxLength(150)
                .IsRequired();
            //mapeamento do relacionamento..
            //muitos para muitos
            HasMany(f => f.Funcionarios)
                .WithMany(f => f.Funcoes)
                .Map(map =>
                {
                    //nome da tabela associativa
                    map.ToTable("FUNCAOFUNCIONARIO");
                    //chave estrangeira para a entidade 'Funcao'
                    map.MapLeftKey("IDFUNCAO");
                    //chave estrangeira para a entidade 'Funcionario'
                    map.MapRightKey("IDFUNCIONARIO");
                });
        }
   }
}
______
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity.ModelConfiguration; //ORM
using Projeto.Entities; //importando
namespace Projeto.DAL.Configurations
    //REGRA 1) Herdar EntityTypeConfiguration
   public class FuncionarioConfiguration
        : EntityTypeConfiguration<Funcionario>
        //REGRA 2) Construtor -> ctor + 2x[tab]
        public FuncionarioConfiguration()
            //nome da tabela
            ToTable("FUNCIONARIO");
```

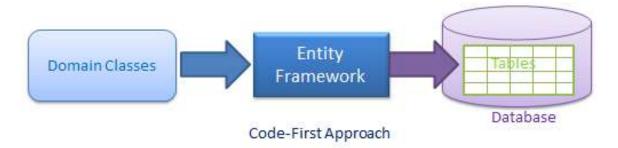


Desenvolvimento de aplicações web com Asp.Net WebApi

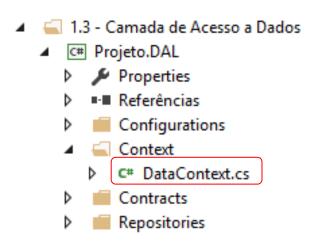
```
//chave primária
            HasKey(f => new { f.IdFuncionario });
            //demais campos
            Property(f => f.IdFuncionario)
                .HasColumnName("IDFUNCIONARIO");
            Property(f => f.Nome)
                .HasColumnName("NOME")
                .HasMaxLength(150)
                .IsRequired();
            Property(f => f.Salario)
                .HasColumnName("SALARIO")
                .HasPrecision(18,2)
                .IsRequired();
            Property(f => f.DataAdmissao)
                .HasColumnName("DATAADMISSAO")
                .IsRequired();
        }
    }
}
```

CodeFirst

Técnica utilizada pelo EntityFramework para gerar e atualizar a estrutura do banco de dados (tabelas) conforme o mapeamento das classes de entidade.



Para que o EntityFramework possa conectar-se a uma base de dados é necessário criarmos uma classe que HERDE a classe **DbContext**





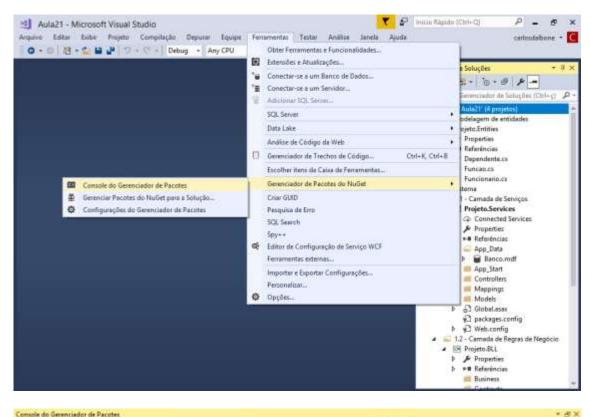
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration; //connectionstring
using System.Data.Entity; //entityframework
using Projeto.Entities; //classes de entidade
using Projeto.DAL.Configurations; //classes de mapeamento
namespace Projeto.DAL.Context
{
    //REGRA 1) HERDAR DbContext
   public class DataContext : DbContext
   {
        //REGRA 2) Criar um construtor que envie para a DbContext
        //o caminho da connectionstring do banco de dados
        public DataContext()
            : base(ConfigurationManager.ConnectionStrings
                    ["projeto"].ConnectionString)
        {
        }
        //REGRA 3) Sobrescrever o método OnModelCreating
        protected override void OnModelCreating(DbModelBuilder modelBuilder)
        {
            //adicionar as classes de mapeamento..
            modelBuilder.Configurations.Add(new FuncaoConfiguration());
            modelBuilder.Configurations.Add(new FuncionarioConfiguration());
            modelBuilder.Configurations.Add(new DependenteConfiguration());
        }
        //REGRA 4) Declarar uma propriedade (prop + 2x[tab])
       //DbSet para cada entidade
        public DbSet<Funcao> Funcao { get; set; }
        public DbSet<Funcionario> Funcionario { get; set; }
        public DbSet<Dependente> Dependente { get; set; }
   }
}
```

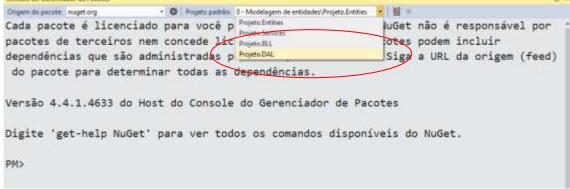
Desenvolvimento de aplicações web com Asp.Net WebApi

Aula 21

Migrations

Recurso do EntityFramework capaz de alterar a estrutura do banco de dados conforme o mapeamento das classes de entidade. Podemos por meio do Migrations criar as tabelas no banco de dados baseado no mapeamento e também atualizar a estrutura do banco de dados conforme o mapeamento for alterado.





Comando para habilitar o recurso de migrations

PM> enable-migrations

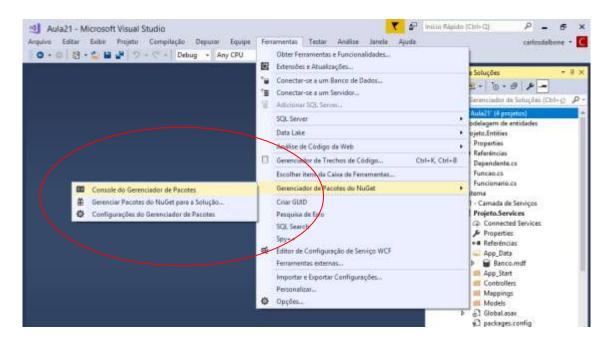
Checking if the context targets an existing database... Code First Migrations enabled for project Projeto.DAL.



A classe abaixo foi criada no projeto:

```
namespace Projeto.DAL.Migrations
{
   using System;
   using System.Data.Entity;
   using System.Data.Entity.Migrations;
   using System.Linq;
   internal sealed class Configuration : DbMigrationsConfiguration
                                        <Projeto.DAL.Context.DataContext>
   {
       public Configuration()
           //flag que dá permissão ao EntityFramework para
           //criar ou atualizar o conteudo do banco de dados
           AutomaticMigrationsEnabled = true;
           //flag que dá permissão ao EntityFramework para
           //excluir o conteudo do banco de dados
           AutomaticMigrationDataLossAllowed = true;
       }
       protected override void Seed(Projeto.DAL.Context.DataContext context)
           // This method will be called after migrating to the latest version.
           // You can use the DbSet<T>.AddOrUpdate() helper extension method
           // to avoid creating duplicate seed data.
   }
}
```

Atualizando o conteudo do banco de dados



Desenvolvimento de aplicações web com Asp.Net WebApi

Aula 21

PM> update-database

Specify the '-Verbose' flag to view the SQL statements being applied to the target database.

No pending explicit migrations.

Applying automatic migration: 201906042227394_AutomaticMigration.

Running Seed method.

PM>

Conteúdo gerado no banco de dados:

- △ Conexões de Dados
 - Banco.mdf
 - Tabelas
 - ▲ III _MigrationHistory
 - MigrationId
 - → ContextKey
 - Model
 - ☐ ProductVersion
 - ■ DEPENDENTE
 - **→** IDDEPENDENTE
 - NOME
 - ☐ DATANASCIMENTO
 - ☐ IDFUNCIONARIO
 - FUNCAO
 - **→** IDFUNCAO
 - ☐ NOME
 - FUNCAOFUNCIONARIO
 - → IDFUNCAO
 - → IDFUNCIONARIO
 - FUNCIONARIO
 - → IDFUNCIONARIO
 - NOME
 - ☐ SALARIO
 - ☐ DATAADMISSAO
 - Modos de Exibição
 - Procedimentos armazenados
 - Funções
 - Sinônimos



Desenvolvimento de aplicações web com Asp.Net WebApi

Contratos

Interfaces para definir os métodos que serão desenvolvidos na camada de repositório de dados.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Projeto.DAL.Contracts
    //<T> Tipo de dado genérico
    public interface IBaseRepository<T> where T : class
        void Insert(T obj);
        void Update(T obj);
        void Remove(T obj);
        List<T> FindAll();
        T FindById(int id);
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando
namespace Projeto.DAL.Contracts
    public interface IDependenteRepository
        : IBaseRepository<Dependente>
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando
namespace Projeto.DAL.Contracts
{
    public interface IFuncaoRepository
        : IBaseRepository<Funcao>
    {
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando

namespace Projeto.DAL.Contracts
{
    public interface IFuncionarioRepository
        : IBaseRepository<Funcionario>
        {
        }
    }
}
```

Repositório Genérico

or mati

Através do EntityFramework podemos criar uma classe de repositorio que possa inserir, atualizar, excluir, consultar todos e consultar por id qualquer tipo de entidade.

```
M Aula21

    ◆ Frepets DAL Repositiones Resellepository • T >

        4
                 using System.Text;
                using System. Threading. Tasks;
        5
        6
                 using System.Data.Entity; //entityframework
        7
                using Projeto.Entities; //importando
        8
                using Projeto.DAL.Context; //importando
                                                                                      CS0535 "BaseRepository<T>" não implementa membr
"BaseRepository<T>.insert(T)"
        9
                using Projeto.DAL.Contracts; //importando
      10
                                                                                         public void Insert(7 cbj)
              - namespace Projeto.DAL.Repositories
      11
                                                                                            throw new melloplementedburgthin();
      12
      13
                       public class BaseRepository<T> : IBaseRep
                                                                                         public void update(1 00])
      14
                            where T : class
                                                                                            throw new wortinglementsoficception();
      15
                                                        Implementar interface
                                                        Implementar interface explicitamente
                                                                                         public void Pemove(T.obj)
      17
                      }
                                                                                            throw new wolfoulnesstandischarge ();
      18
               }
                                                                                         public Listers FindAll()
      19
                                                                                            throw new muticulescentedEsception();
                                                                                         public * findeyle(int is)
                                                                                            throw new notinglementedDocuption();
                                                                                      Visualizor alterações
                                                                                      Comigir todas as ocomências em: Documento | Projeto
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Data.Entity; //entityframework
using Projeto.Entities; //importando
using Projeto.DAL.Context; //importando
using Projeto.DAL.Contracts; //importando
```



```
namespace Projeto.DAL.Repositories
    public class BaseRepository<T> : IBaseRepository<T>
        where T : class
    {
        public virtual void Insert(T obj)
            using (var ctx = new DataContext())
                ctx.Entry(obj).State = EntityState.Added;
                ctx.SaveChanges();
        }
        public virtual void Update(T obj)
            using (var ctx = new DataContext())
                ctx.Entry(obj).State = EntityState.Modified;
                ctx.SaveChanges();
        }
        public virtual void Remove(T obj)
            using (var ctx = new DataContext())
                ctx.Entry(obj).State = EntityState.Deleted;
                ctx.SaveChanges();
        }
        public virtual List<T> FindAll()
            using (var ctx = new DataContext())
                return ctx.Set<T>().ToList();
        }
        public virtual T FindById(int id)
            using (var ctx = new DataContext())
                return ctx.Set<T>().Find(id);
            }
        }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities; //importando
using Projeto.DAL.Contracts; //importando
```



Desenvolvimento de aplicações web com Asp.Net WebApi

```
namespace Projeto.DAL.Repositories
    public class DependenteRepository
        : BaseRepository<Dependente>, IDependenteRepository
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
using Projeto.DAL.Contracts;
namespace Projeto.DAL.Repositories
    public class FuncaoRepository
        : BaseRepository<Funcao>, IFuncaoRepository
    {
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
using Projeto.DAL.Contracts;
namespace Projeto.DAL.Repositories
{
    public class FuncionarioRepository
        : BaseRepository<Funcionario>, IFuncionarioRepository
    }
```

Camada de Regras de Negócio:

Criando as interfaces para a camada BLL

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

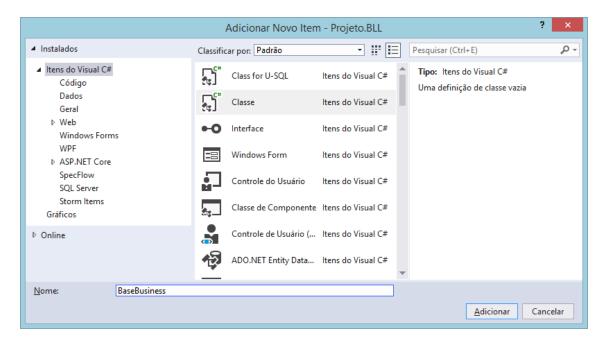


```
namespace Projeto.BLL.Contracts
    public interface IBaseBusiness<T> where T : class
        void Cadastrar(T obj);
        void Atualizar(T obj);
        void Excluir(T obj);
        List<T> ConsultarTodos();
        T ConsultarPorId(int id);
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
namespace Projeto.BLL.Contracts
    public interface IDependenteBusiness
        : IBaseBusiness<Dependente>
    {
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
namespace Projeto.BLL.Contracts
    public interface IFuncaoBusiness
        : IBaseBusiness<Funcao>
    {
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.Entities;
namespace Projeto.BLL.Contracts
{
    public interface IFuncionarioBusiness
        : IBaseBusiness<Funcionario>
    {
    }
}
```



Desenvolvimento de aplicações web com Asp.Net WebApi

Criando as classes para implementar cada interface:



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Projeto.DAL.Contracts; //importando
using Projeto.BLL.Contracts; //importando
namespace Projeto.BLL.Business
{
    public class BaseBusiness<T> : IBaseBusiness<T>
        where T : class
    {
        //atributo
        private IBaseRepository<T> repository;
        //construtor para entrada de argumentos
        public BaseBusiness(IBaseRepository<T> repository)
            this.repository = repository;
        }
        public virtual void Cadastrar(T obj)
        {
            repository.Insert(obj);
        }
        public virtual void Atualizar(T obj)
            repository.Update(obj);
        }
        public virtual void Excluir(T obj)
        {
            repository.Remove(obj);
        }
```

Aula 21

Desenvolvimento de aplicações web com Asp.Net WebApi

```
public virtual List<T> ConsultarTodos()
           return repository.FindAll();
       }
       public virtual T ConsultarPorId(int id)
           return repository.FindById(id);
       }
   }
}
Solução 'Aula21' (4 projetos)
   0 - Modelagem de entidades
   1 - Sistema
   1.1 - Camada de Serviços

    1.2 - Camada de Regras de Negócio

▲ C# Projeto.BLL

         Properties
         ▶ ■ ■ Referências
           Business
                C# BaseBusiness.cs
                C# DependenteBusiness.cs
            C# FuncaoBusiness.cs
               C# FuncionarioBusiness.cs
         Contracts
            C# IBaseBusiness.cs
            C* IDependenteBusiness.cs
                C# IFuncaoBusiness.cs
```

C# IFuncionarioBusiness.cs

1.3 - Camada de Acesso a Dados

Continua...