Aula 07 – ViewModel

Disciplina: Programação Web
Prof. Allbert Velleniche de Aquino Almeida
E-mail: allbert.almeida@fatec.sp.gov.br
Site: http://www.allbert.com.br
/allbert.almeida

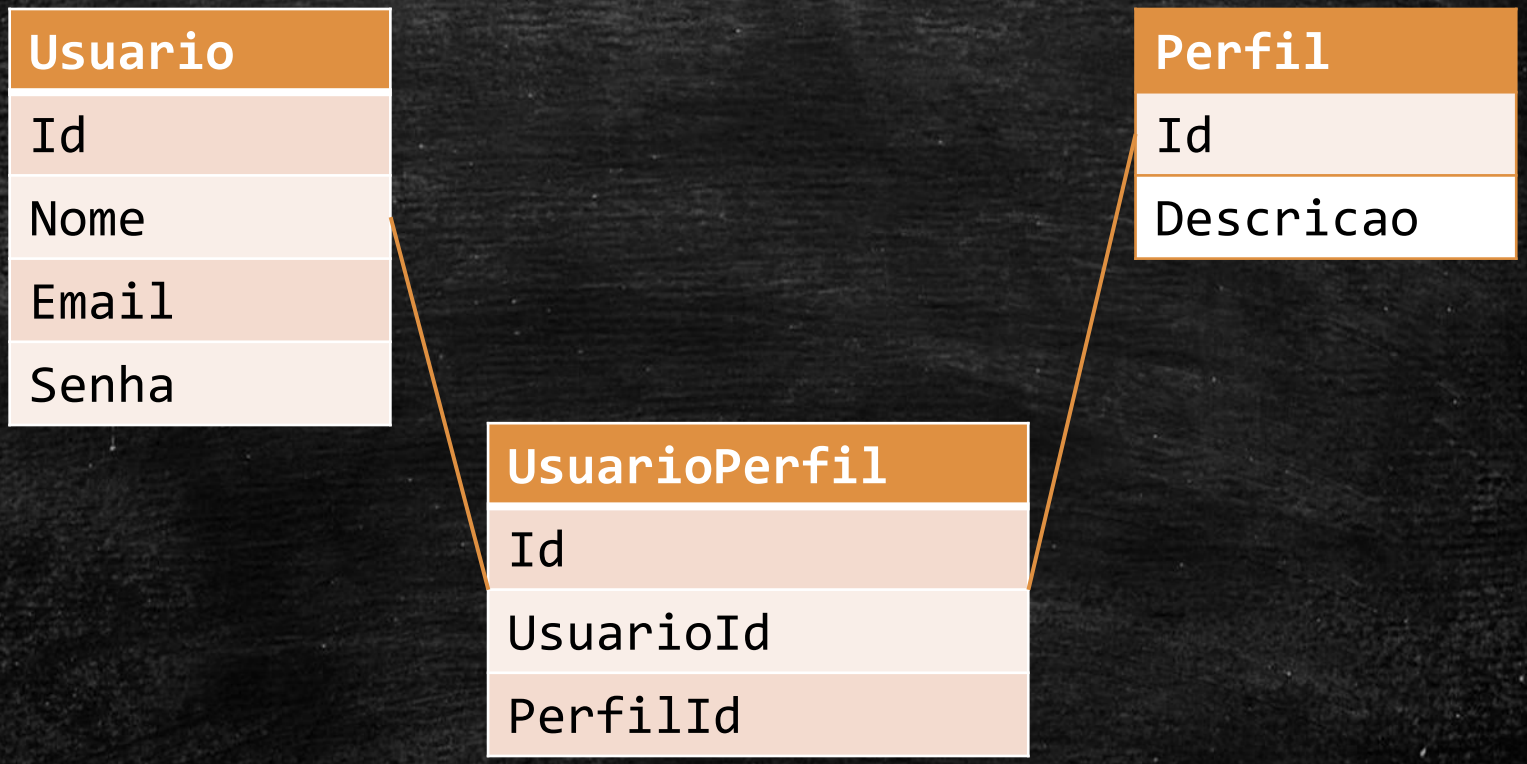
Objetivo

- O objetivo dessa aula é entender a necessidade de utilização das ViewsModels e construir uma aplicação que utilize essa referência.

Conceito

- View Model representa um conjunto de uma ou mais Models e outros dados que serão representados em uma View que necessita exibir determinado conjunto de informações;
- No ASP.Net MVC os View Models nos permitem modelar várias entidades a partir de um ou mais modelos em um único objeto.

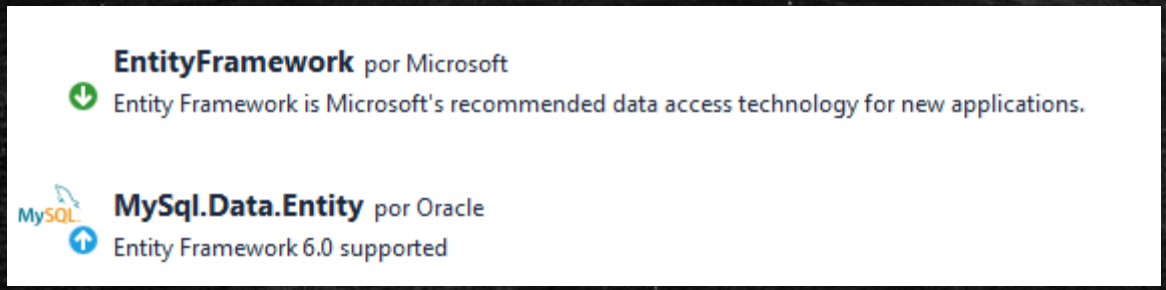
Modelo


Usuario
Id
Nome
Email
Senha
Perfil
Id
Descricao
UsuarioPerfil
Id
UsuarioId
PerfilId

Adicionar pacotes à solução

- MySql Data Entity 6.9.12

```
Install-Package MySql.Data.Entity
-Version 6.9.12
```



Alteração Classe Usuário

```
public class Usuario {
    public int Id { get; set; }
    [Required]
    [MaxLength(200)]
    public string Nome { get; set; }
    [Required]
    [MaxLength(200)]
    [EmailAddress]
    public string Email { get; set; }
    public string Senha { get; set; }
    public virtual ICollection<UsuarioPerfil>
UsuarioPerfil { get; set; }
}
```

Criar uma classe Perfil

```
public class Perfil
{
    public int Id { get; set; }
    [Required]
    [MaxLength(255)]
    [Display(Name ="Descrição")]
    public string Descricao { get; set; }
    public virtual
ICollection<UsuarioPerfil> UsuarioPerfil
{ get; set; }
}
```

Criar uma classe UsuarioPerfil

```
public class UsuarioPerfil {
    public int Id { get; set; }
    public int UsuarioId { get; set; }
    public int PerfilId { get; set; }
    public virtual Usuario Usuario { get; set; }
    public virtual Perfil Perfil { get; set; }
}
```

Criar uma classe Contexto

```
public class Contexto:DbContext {

    public Contexto() : base(nameOrConnectionString:
"StringConexao") { }

    public DbSet<Usuario> Usuario { get; set; }

    public DbSet<Perfil> Perfil { get; set; }

    public DbSet<UsuarioPerfil> UsuarioPerfil { get; set; }

}
```

Adicionar a String de Conexão no WebConfig

```
<connectionStrings>
    <add name="StringConexao"
providerName="MySql.Data.MySqlClient"
connectionString="server=localhost;dat
abase=aula;uid=root;pwd='root'" />
  </connectionStrings>
```

Contexto - ModelBuilder

- Na Classe Contexto iremos adicionar um método de sobrescrita (“override”) na criação do BD, ou seja, quando o modelo for criado usará as especificações contidas nesse método, sendo possível apontar como ficarão os nomes de tabelas e campos, entre outras ações que podem ser realizadas.

Alteração Classe Contexto

```
protected override void OnModelCreating(DbModelBuilder mb) {
        var usu = mb.Entity<Usuario>();
        usu.ToTable("usu_usuario");
        usu.Property(x => x.Id).HasColumnName("usu_codigo");
        usu.Property(x => x.Nome).HasColumnName("usu_nome");
        usu.Property(x => x.Email).HasColumnName("usu_email");
        usu.Property(x => x.Senha).HasColumnName("usu_senha");

        var per = mb.Entity<Perfil>();
        per.ToTable("per_perfil");
        per.Property(x => x.Id).HasColumnName("per_codigo");
        per.Property(x => x.Descricao).HasColumnName("per_descricao");

        var upe = mb.Entity<UsuarioPerfil>();
        upe.ToTable("upe_usuarioperfil");
        upe.Property(x => x.Id).HasColumnName("upe_codigo");
        upe.Property(x => x.PerfilId).HasColumnName("per_codigo");
        upe.Property(x => x.UsuarioId).HasColumnName("usu_codigo");
    }
```

Compilar a solução

- Compilar a solução para que os modelos sejam validados e seja possível criar o "CRUD" da classe Usuario; Perfil; UsuarioPerfil

Usar o comando

- Enable-Migrations
- Adicionar na classe Configuration

```
AutomaticMigrationsEnabled = true;

AutomaticMigrationDataLossAllowed = true;

SetSqlGenerator("MySql.Data.MySqlClient", new
MySql.Data.Entity.MySqlMigrationSqlGenerator());
```

Usar os comandos

- No console de gerenciador de pacotes NuGet:
  - –Add-Migration IntialDb
  - –Update-Database -Verbose

Classe Configuration

- Na classe "Configuration.cs" contém um método semente ("seed"), que permite a inclusão de registros iniciais para que tenhamos um conjunto de registros de testes.

Classe Configuration

- Para esses testes foram adicionados:
  - 3 usuários (Allbert, Bruno e Claudemir) com uma senha padrão "Abc123" já criptografada no padrão SHA512 (como veremos na próxima seção);
  - 3 perfis (Admin, Comum, Gerente);
  - 5 associações: O usuário 1 (Allbert) está associado a todos os perfis; o usuário 2 (Bruno) está associado ao perfil Comum; e o usuário 3 (Claudemir) está associado ao perfil Gerente;

Classe Configuration

- Adicionar na classe Seed (Configuration)

```
context.Usuario.AddOrUpdate(
            p => p.Email,
            new Models.Usuario { Id = 1, Nome = "Allbert Almeida", Email = "allbert@fatec.com.br", Senha =
"vDDsx1jGNpHGnmbYRjJmcJJL/5YJtf6/OcHobMqPtyeDrV5bcHY1nm1wm8wM03mt4UlZRfhZHph2yyY05DE5pg==" },
            new Models.Usuario { Id = 2, Nome = "Bruno Silva", Email = "bruno@fatec.com.br", Senha =
"vDDsx1jGNpHGnmbYRjJmcJJL/5YJtf6/OcHobMqPtyeDrV5bcHY1nm1wm8wM03mt4UlZRfhZHph2yyY05DE5pg==" },
            new Models.Usuario { Id = 3, Nome = "Claudemir Santos", Email = "claudemir@fatec.com.br", Senha =
"vDDsx1jGNpHGnmbYRjJmcJJL/5YJtf6/OcHobMqPtyeDrV5bcHY1nm1wm8wM03mt4UlZRfhZHph2yyY05DE5pg==" });
            context.Perfil.AddOrUpdate(
            p => p.Descricao,
            new Models.Perfil { Id = 1, Descricao = "Admin" },
            new Models.Perfil { Id = 2, Descricao = "Comum" },
            new Models.Perfil { Id = 3, Descricao = "Gerente" });
            context.UsuarioPerfil.AddOrUpdate(
            p => p.Id,
            new Models.UsuarioPerfil { Id = 1, UsuarioId = 1, PerfilId = 1 },
            new Models.UsuarioPerfil { Id = 1, UsuarioId = 1, PerfilId = 2 },
            new Models.UsuarioPerfil { Id = 1, UsuarioId = 1, PerfilId = 3 },
            new Models.UsuarioPerfil { Id = 1, UsuarioId = 2, PerfilId = 2 },
            new Models.UsuarioPerfil { Id = 1, UsuarioId = 3, PerfilId = 3 });
```

- Update-Database -Verbose

Classe Funcoes

```
public class Funcoes {
    /// <summary>
    /// Gera Hash SHA1, SHA256, SHA512
    /// </summary>
    /// <param name="texto"></param>
    /// <param name="nomeHash"></param>
    /// <returns></returns>
    public static string HashTexto(string texto, string nomeHash)
    {
HashAlgorithm algoritmo = HashAlgorithm.Create(nomeHash);
if (algoritmo == null)
{
    throw new ArgumentException("Nome de hash incorreto", "nomeHash");
}
byte[] hash = algoritmo.ComputeHash(Encoding.UTF8.GetBytes(texto));
    return Convert.ToBase64String(hash);
}
    }
```

Adicionar ViewModel

- Construir ViewModel utilizando as propriedades do formulário;
  - Cadastro;
  - Acesso;

Adicionar Classe VMUsuario

```
public class Cadastro
    {
    [Required]
    [MaxLength(255)]
    public string Nome { get; set; }
    [Required]
    [EmailAddress]
    public string Email { get; set; }
    [DataType(DataType.Password)]
    [RegularExpression("((?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{6,12})", ErrorMessage = "A senha deve conter aos menos
uma letra maiúscula, minúscula e um número. Deve ser no mínimo 6 caracteres")]
    public string Senha { get; set; }
    [DataType(DataType.Password)]
    [Compare("Senha")]
    [Display(Name ="Confirma Senha")]
    public string ConfirmaSenha { get; set; }
     }
     public class Acesso
     {
    [Required]
    [EmailAddress]
    public string Email { get; set; }
    [DataType(DataType.Password)]
    [RegularExpression("((?=.*\\d)(?=.*[a-z])(?=.*[A-Z]).{6,12})", ErrorMessage = "A senha deve conter aos menos
uma letra maiúscula, minúscula e um número. Deve ser no mínimo 6 caracteres")]
    public string Senha { get; set; }
     }
```

Adicionar HomeController

- Adicionar Action Cadastro a exibição (view) usando o modelo Create, da classe Cadastro;

Adicionar HomeController

```
private Contexto db = new Contexto();
[HttpPost]
[ValidateAntiForgeryToken]
public ActionResult Cadastro(Cadastro cad)
{
    if (ModelState.IsValid)
    {
        if (db.Usuario.Where(x => x.Email == cad.Email).ToList().Count > 0)
        {
            ModelState.AddModelError("", "E-mail já utilizado!");
            return View(cad);
        }
        UsuarioPerfil upe = new UsuarioPerfil();
        upe.Usuario = new Usuario();
        upe.Usuario.Nome = cad.Nome;
        upe.Usuario.Email = cad.Email;
        upe.Usuario.Senha = Funcoes.HashTexto(cad.Senha, "SHA512");
        upe.Perfil = db.Perfil.Find(2);
        if (upe.Perfil == null)
        {
            ModelState.AddModelError("", "Não existe o perfil para cadastro");
            return View(cad);
        }
        db.UsuarioPerfil.Add(upe);
        db.SaveChanges();
        return RedirectToAction("Index");
    }
    return View(cad);
}
```

Atividade prática

- Criar uma classe para Cadastro de Filmes (Título, Gênero, Duração, Resumo);
- Construir uma ViewModel com cadastro somente das propriedades Título e Gênero e inserir os dados no BD;
- Realize a trigamação do BD e adicione alguns registros no método Seed (semente)