ساختار پروژه



docs (Solution Folder and Physical Folder)

LICENSE (Solution Folder and Physical Folder)

src (Solution Folder and Physical Folder)

tests (Solution Folder and Physical Folder)

docs

ReadMe.docs

…

src

Core (Solution Folder and Physical Folder)

Framework (Solution Folder and Physical Folder)

Infrastructure (Solution Folder and Physical Folder)

Persistence (Solution Folder and Physical Folder)

Presentation (Solution Folder and Physical Folder)

Core

Application (Class Library)

Constants (Class Library)

Domain (Class Library)

Resources (Class Library)

**ViewModels** / Dtos (Class Library)

Framework

Dtat (Class Library)

Dtat.Net.Mail (Class Library)

Dtat.Net.Mail.Abstractions (Class Library)

Dtat.Seedwork.Abstractions (Class Library)

Dtat.StateMachine (Class Library)

Dtat.StateMachine.Abstractions (Class Library)

Infrastructure

Infrastructure (Class Library)

Persistence

Persistence (Class Library)

Persistence.Sqlite (Class Library)

Persistence.SqlServer (Class Library)

Presentation

Api (ASP.NET Core Web API)

Client (Blazor Web Assembly)

**Server** (**ASP.NET Core Razor Pages** / MVC)

**Project Dependencies:**

Dtat

Dtat.Seedwork.Abstractions

Dtat.StateMachine.Abstractions

Dtat.StateMachine

Dtat.StateMachine.Abstractions

Resources

Constants

Domain

Dtat

Constants

Resources

Dtat.Net.Mail.Abstractions

Dtat.Seedwork.Abstractions

Application

Domain

ViewModels

Domain

Constants

Resources

Infrastructure

Domain

Persistence

Dtat

Domain

Constants

**NuGets:**

Microsoft.EntityFrameworkCore.Tools

Microsoft.EntityFrameworkCore.Proxies

Microsoft.EntityFrameworkCore.Sqlite

Microsoft.EntityFrameworkCore.SqlServer

Server

Dtat

Dtat.Net.Mail

Domain

Constants

Resources

ViewModels

Persistence

Persistence.Sqlite

Persistence.SqlServer

**NuGets:**

System.Drawing.Common

Microsoft.EntityFrameworkCore.Design

Microsoft.AspNetCore.Authentication.Google

**نحوه تعریف یک Enum**

فرض کنید که می‌خواهیم یک Enum به نام Gender ایجاد نماییم، از آنجایی که این Enum مربوط به کاربر و Identity می‌باشد، در پوشه Features و در پوشه Identity، پوشه‌ای به نام Enums ایجاد کرده و یک Enum به نام GenderEnum در داخل آن ایجاد می‌کنیم:

public enum GenderEnum : int

{

Unspecified = 0,

Male = 1,

Female = 2,

}

برای این‌که روی عناوین Enum و فعال و غیرفعال کردن آن‌ها کنترل داشته باشیم، معادل آن، یک کلاس و به طبع آن، یک جدول در بانک اطلاعاتی ایجاد می‌کنیم.

نام کلاس را Gender نام‌گذاری می‌کنیم.

چون می‌خواهیم پروژه ما چند زبانه باشد، در کلاس Gender دو Property به نام‌های CultureId و Culture تعریف می‌کنیم.

در کلاس Gender، یک Property به نام Code از جنس GenderEnum تعریف می‌کنیم.

در کلاس Gender، یک Property به نام Title از جنس string، برای ذخیره‌سازی عنوان هر Enum تعریف می‌کنیم.

**ایجاد روابط بین مدل‌ها**

در این مثال خاص، یک رابطه یک به چند بین Culture و Gender‌ داریم و یک رابطه یک به چند بین Gender و UserProfile. این روابط را ایجاد می‌کنیم:

در کلاس Culture:

public Culture(…) : base()

{

Genders =

new System.Collections.Generic.List<Identity.Gender>();

}

public virtual System.Collections.Generic.IList

<Identity.Gender> Genders { get; private set; }

در کلاس UserProfile:

public System.Guid GenderId { get; set; }

public virtual Gender? Gender { get; set; }

در کلاس Gender:

public Gender(…) : base()

{

UserProfiles =

new System.Collections.Generic.List<UserProfile>();

}

public virtual System.Collections.Generic.IList

<UserProfile> UserProfiles { get; private set; }

**ایجاد و ویرایش کلاس‌های Configuration**

حال در پروژه Persistence، و در داخل پوشه Configurations، و سپس در پوشه Features و بعد در پوشه Identity، اقدام به ایجاد کلاس GenderConfiguration می‌کنیم:

builder

.HasIndex(current => new { current.CultureId, current.Code })

.Is Unique(unique: true)

;

builder

.HasIndex(current => new { current.CultureId, current.Title })

.IsUnique(unique: true)

;

builder

.HasMany(current => current.UserProfiles)

.WithOne(other => other.Gender)

.IsRequired(required: true)

.HasForeignKey(other => other.GenderId)

.OnDelete(deleteBehavior:

Microsoft.EntityFrameworkCore.DeleteBehavior.NoAction)

;

در کلاس Culture Configuration تغییرات ذیل را اعمال می‌کنیم:

builder

.HasMany(current => current.Genders)

.WithOne(other => other.Culture)

.IsRequired(required: true)

.HasForeignKey(other => other.CultureId)

.OnDelete(deleteBehavior:

Microsoft.EntityFrameworkCore.DeleteBehavior.NoAction)

;

**ایجاد Property در DatabaseContext**

در کلاس DatabaseContext یک Property به نام Genders و به شکل ذیل ایجاد می‌کنیم:

public Microsoft.EntityFrameworkCore.DbSet

<Domain.Features.Identity.Gender> Genders { get; set; }

**ایجاد Seed Data**

در پروژه Persistence و در پوشه Extensions و در داخل کلاس ModelBuilderExtensions اقدام به ایجاد Seed Data برای Gender می‌کنیم:

**تست پروژه**

یک‌بار پروژه را اجرا می‌کنیم که از ایجاد جداول و اطلاعات پایه اطمینان حاصل کنیم.

احتمالا به خطا برخورد می‌کنیم! چرا که برای UserProfile و در داخل Constructor آن، پس از اعمال تغییرات، GenderId را نیز به عنوان پارامتر الزامی مشخص کرده‌ایم، لذا در زمان Seed Data برای UserProfile با مشکل مواجه می‌شویم.

این مشکل را برطرف کرده و مجددا پروژه را اجرا می‌کنیم.

نحوده تعریف یک اطلاعات پایه، ماننده Post Type

فرض کنید که می‌خواهیم یک اطلاعات پایه به نام Post Type ایجاد نماییم، از آنجایی که این مورد مربوط به CMS می‌باشد، در پوشه ذیل، اقدام به ایجاد مدل آن می‌کنیم:

[src] 🡪 [Core] 🡪 [[Domain]] 🡪 [Features] 🡪 [Cms] 🡪 PostType.cs

**ایجاد روابط بین مدل‌ها**

در این مثال:

یک رابطه یک به چند بین Culture و PostType وجود دارد:

یک رابطه یک به چند بین PostType و Post وجود دارد:

این روابط را در کلاس‌های مربوطه به شکل فوق ایجاد می‌کنیم.

باید دقت داشته باشیم که PostType برای Post یک فیلد الزامی بوده و لذا باید در Constructor مربوط به Post آن‌را (PostTypeId) دریافت نماییم!

ایجاد و به روز رسانی فایل‌های Configuration در نشانی ذیل:

[src] 🡪 [Persistence] 🡪 [[Persistence]] 🡪 [Configurations] 🡪 [Features]

[Cms] 🡪 PostTypeConfiguration.cs 🡪 ایجاد می‌کنیم

[Common] 🡪 CultureConfiguration.cs 🡪 ویرایش می‌کنیم

ایجاد Property برای PostType در داخل کلاس DatabaseContext:

ایجاد Seed Data

از طریق نشانی ذیل، اقدام به ایجاد Seed Data می‌کنیم:

[src] 🡪 [Persistence] 🡪 [[Persistence]] 🡪 [Extensions]:

ModelBuilderExtensions.cs

ابتدا یک منو برای نمایش فهرست دسته‌بندی‌ها (به فارسی و انگلیسی) ایجاد می‌کنیم.

سپس در صورتی که این جدول نیاز به اطلاعات اولیه داشت، در این‌جا اقدام به درج اطلاعات اولیه می‌کنیم:

* خبر
* مقاله

یک‌بار برنامه را اجرا می‌کنیم تا اطمینان حاصل کنیم که هیچ مشکلی در طراحی مدل‌ها وجود ندارد.

صفحه مربوط به نمایش دسته‌بندی‌ها را در مسیر ذیل ایجاد می‌کنیم:

[src] 🡪 [Presentation] 🡪 [[Server]] 🡪 [Pages] 🡪 [Features] 🡪 [Cms]:

Types.cshtml & Types.cshtml.cs

پروژه را یک‌بار اجرا می‌کنیم تا نتیجه منوی بالای صفحه (دسته‌بندی‌ها) و نیز صفحه مربوط به آن را مشاهده نماییم.

صفحه مربوط به نمایش دسته‌بندی را در مسیر ذیل ایجاد می‌کنیم:

[src] 🡪 [Presentation] 🡪 [[Server]] 🡪 [Pages] 🡪 [Features] 🡪 [Cms]:

Types.cshtml & Types.cshtml.cs

برای انجام این کار، نیاز به ایجاد ViewModel در نشانی ذیل داریم:

[src] 🡪 [Core] 🡪 [[ViewModels]] 🡪 [Pages] 🡪 [Features] 🡪 [Cms]:

TypeViewModel.cs

با ایجاد این ViewModel اقدام به ایجاد / ویرایش فایل Type.cshtml.cs می‌کنیم:

حال فایل Type.cshtml را ایجاد / ویرایش می‌کنیم:

Role 1…N Localized Role

**Culture**

**Id Name**

1 فارسی

2 English

**Role**

**Id Name**

10 User

11 Admin

**Localized Role**

**Id Culture Id Role Id Title**

100 1 10 کاربر معمولی

101 1 11 مدیر سامانه

102 2 10 Simple User

103 2 11 Administrator

**1 10** !هر چی!

**1** !هر چی! **کاربر معمولی**

اسلایدهای آماده تستی:

<https://www.tutorialrepublic.com/examples/images/slide1.png>

<https://www.tutorialrepublic.com/examples/images/slide2.png>

<https://www.tutorialrepublic.com/examples/images/slide3.png>

<https://smartslider3.com/wp-content/uploads/2018/11/WordPresscarousel-840x441.png>