

***Entity Framework Core***

**Training Problem**

|  |  |
| --- | --- |
| Document Code | 25e-BM/HR/HDCV/FSOFT |
| Version | 1.1 |
| Effective Date | 20/11/2012 |

**Hanoi, 09/2019**

RECORD OF CHANGES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Effective Date | Change Description | Reason | Reviewer | Approver |
| 1 | 25/09/2019 | Create Problems |  | TuTB | VinhNV |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Technologies 4](#_Toc29303641)

[Working Environment 4](#_Toc29303642)

[Database Relationship 5](#_Toc29303643)

[Problem\_01 6](#_Toc29303644)

[Create solution and setup environment 6](#_Toc29303645)

[Problem\_02 6](#_Toc29303646)

[Create models 6](#_Toc29303647)

[Problem\_03 7](#_Toc29303648)

[Create context 7](#_Toc29303649)

[Problem\_04 7](#_Toc29303650)

[Configure Fluent API for the Many-To-Many relationship 7](#_Toc29303651)

[Problem\_05 8](#_Toc29303652)

[Setup Database Initialization Strategy 8](#_Toc29303653)

[Problem\_06 8](#_Toc29303654)

[Create repositories 8](#_Toc29303655)

[Problem\_07 10](#_Toc29303656)

[Unit Test 10](#_Toc29303657)

[Problem\_08 11](#_Toc29303658)

[Code-first migration 11](#_Toc29303659)

|  |  |
| --- | --- |
|  | **CODE: NEFW.A.L001**  **TYPE: Long**  **LOC: NA**  **DURATION: Complete in 3 days** |

**Purpose of Problems**

Use Entity Framework Core to manage a simple blog named **JustBlog**.

The blog allows owner post new blog, review, update or delete an existing blog post. Each blog post is belong to on category and can have multiple tags. The post also have some comments.

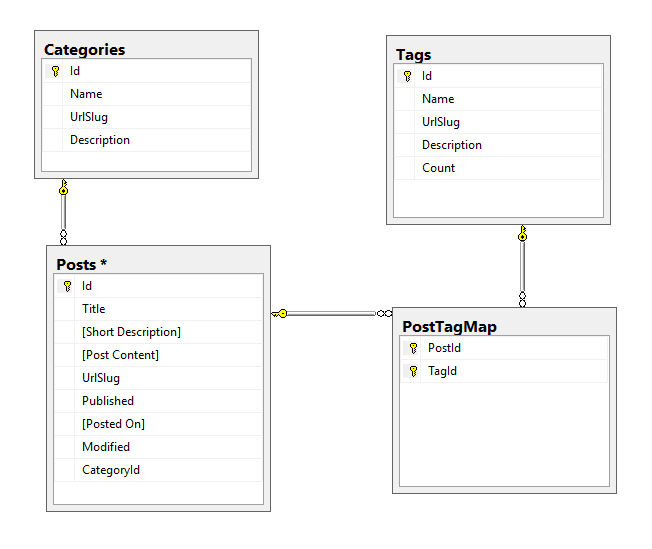
## Technologies

* Entity Framework Core
* MS .Net 6.0
* MS C# 10
* Linq
* Repository Design pattern.

## Working Environment

* MS SQL Server 2018,
* Visual Studio 2022,

## Database Relationship



* ***URL Slug:*** *is a SEO- and user-friendly string-part in a URL to identify, describe and access a resource. Often the title of a page/article is a valid candidate.*

In this series of assingments, student will use technical points in Entity Framework Core to create, manage a database for the blog.

* Based on database design, create models and context to generate database
* Create repositories to manipulate data
* Use migration to apply changes
* Create unit test project and test

*These Problems should be performed by individuals.*

# Problem\_01

## Create solution and setup environment

### Pre-condition

<none>

### Functional specification

<none>

### Technical needed

<none>

### Tool used

Visual Studio 2022

### Post condition

Create blank solution name: **FA.JustBlog**

Create project type Class Library into the solution. Project name: **FA.JustBlog.Core**

Install **Entity Framework Core** from Nuget

### Estimated time

10 minutes

# Problem\_02

## Create models

### Pre-condition

Finish **Problem\_01**

Inside project **FA.JustBlog.Core**, add new folder named **Models**

Push all models in side this folder, name space for all models is **FA.JustBlog.Core.Models**

### Functional specification

Based on database schema provided, student design appropriate data type for each field then create model for each entity

### Technical needed

EF Core Code-First Conventions

EF Core Data Annotations

### Tool used

Visual Studio 2022

### Post condition

Create model class for: Category, Post, Tag.

Choose appropriate data type for each field

Follow EF Core Code-First Conventions

All name need to follow naming convention

### Example

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace FA.JustBlog.Core.Models

{

public class Category

{

[Key]

public int Id { get; set; }

[Required(ErrorMessage = "Category name is required.")]

[StringLength(255)]

public string Name { get; set; }

[StringLength(255)]

public string UrlSlug { get; set; }

[StringLength(1024)]

public string Description { get; set; }

public virtual IList<Post> Posts { get; set; }

}

}

### Estimated time

60 minutes

# Problem\_03

## Create context

### Pre-condition

Finish **Problem\_02**

### Functional specification

Inside Models folder, create new class **JustBlogContext** for the **DBContext** and add appropriate **DbSet** properties.

### Technical needed

EF Core DBContext

EF Core Database Initialization

### Tool used

Visual Studio 2022

### Post condition

Create **JustBlogContext** for the application

### Estimated time

10 minutes

# Problem\_04

## Configure Fluent API for the Many-To-Many relationship

### Pre-condition

Finish **Problem\_03**

### Functional specification

Configure Fluent API for the Many-To-Many relationship between **Post** and **Tag**, applied into **PostTagMap**.

### Technical needed

EF Core DBContext

EF Core Fluent API

### Tool used

Visual Studio 2022

### Post condition

Configure Fluent API for the Many-To-Many relationship between **Post** and **Tag**, applied into **PostTagMap** successfully.

### Estimated time

10 minutes

# Problem\_05

## Setup Database Initialization Strategy

### Pre-condition

Finish **Problem\_04**

### Functional specification

Add at least 3 objects in seed data for all entities.

### Technical needed

EF Core DBContext

EF Core Code-first: Database Initialization Strategies

### Tool used

Visual Studio 2022

### Post condition

Override **Seed** method, add 3 objects for each entity.

Update **JustBlogContext** by call the database initialization strategy

### Estimated time

30 minutes

# Problem\_06

## Create repositories

### Pre-condition

Finish **Problem\_05**

Inside project **FA.JustBlog.Core**, add new folder named **Repositories**

Push all interfaces/class for repositories in side this folder, name space for all models is **FA.JustBlog.Core.Repositories**

### Functional specification

Based on database schema provided, student design appropriate data type for each field then create model for each entity

### Technical needed

EF Core Code-First

Repository Design Pattern

### References

**Repository Design Pattern** at <https://dotnettutorials.net/lesson/repository-design-pattern-csharp/>

### Tool used

Visual Studio 2022

### Post condition

Create repositories follow signatures.

* **ICategoryRepository/CategoryRepository**
  + Category Find(int categoryId);
  + void AddCategory(Category category);
  + void UpdateCategory(Category category);
  + void DeleteCategory(Category category);
  + void DeleteCategory(int categoryId);
  + IList<Category> GetAllCategories();
* **ITagRepository/TagRepository**
  + Tag Find(int TagId);
  + void AddTag(Tag Tag);
  + void UpdateTag(Tag Tag);
  + void DeleteTag(Tag Tag);
  + void DeleteTag(int TagId);
  + IList<Tag> GetAllTags();
  + Tag GetTagByUrlSlug(string urlSlug);
* **IPostRepository/PostRepository**
  + Post FindPost(int year, int month, string urlSlug);
  + Post FindPost(int postId);
  + void AddPost(Post post);
  + void UpdatePost(Post post);
  + void DeletePost(Post post);
  + void DeletePost(int postId);
  + IList<Post> GetAllPosts();
  + IList<Post> GetPublisedPosts();
  + IList<Post> GetUnpublisedPosts();
  + IList<Post> GetLatestPost(int size);
  + IList<Post> GetPostsByMonth(DateTime monthYear);
  + int CountPostsForCategory(string category);
  + IList<Post> GetPostsByCategory(string category);
  + int CountPostsForTag(string tag);
  + IList<Post> GetPostsByTag(string tag);

### Example

using FA.JustBlog.Core.Models;

namespace FA.JustBlog.Core.Repositories

{

public class CategoryRepository : ICategoryRepository

{

private readonly JustBlogContext db;

public CategoryRepository()

{

db = new JustBlogContext();

}

public void AddCategory(Category category)

{

db.Categories.Add(category);

db.SaveChanges();

}

public void Dispose()

{

db.Dispose();

}

}

}

### Estimated time

120 minutes

# Problem\_07

## Unit Test

### Pre-condition

Finish **Problem\_06**

Create new Unit Test Project (.NET 6.0) named **FA.JustBlog.UnitTest**

Add reference to **FA.JustBlog.Core** project

Add Application Configuration File in to the project

Configure connection string for **JustBlogContext**

### Functional specification

Create Unit test case for all methods in all repositories. Test cases need to cover:

* Normal cases: add/update/get items successfully
* Abnormal cases:
  + Invalid for required any field
  + Invalid format for date time field
  + Invalid format for number field
  + Exceed max length of text field
  + Any special characters (such as: < > ~ ! @ # $ % ^ & \* / \ ….) in any field
  + Get item not exist

### Technical needed

EF Core

Unit test or NUnit

### References

<https://docs.microsoft.com/en-us/visualstudio/test/getting-started-with-unit-testing?view=vs-2017>

### Tool used

Visual Studio 2022

### Post condition

Create test cases to cover requirement.

Run test case and check result.

Update code to fix bug if there.

Finish when all test case are passed.

Generate test report.

### Estimated time

180 minutes

# Problem\_08

## Code-first migration

### Pre-condition

Finish **Problem\_06**

### Functional specification

Update Post model to add new fields:

* ViewCount – int
* RateCount – int
* TotalRate – int
* Rate = TotalRate / Rate Count – decimal

Update Post repository, add new methods:

* IList<Post> GetMostViewedPost(int size)
* IList<Post> GetHighestPosts(int size)

Add new entity named **Comment**

* Name – string
* Email – string
* Post – Post class
* CommentHeader – string
* CommentText – string
* CommentTime – DateTime

Add repository for **Comment**

* Comment Find(int commentId);
* void AddComment(Comment comment);
* void AddComment(int postId, string commentName, string commentEmail, string commentTitle, string commentBody);
* void UpdateComment(Comment comment);
* void DeleteComment(Comment comment);
* void DeleteComment(int commendId);
* IList<Comment> GetAllComments();
* IList<Comment> GetCommentsForPost(int postId);
* IList<Comment> GetCommentsForPost(Post post);

### Technical needed

Entity Framework Core

EF Core Code-first migration

### References

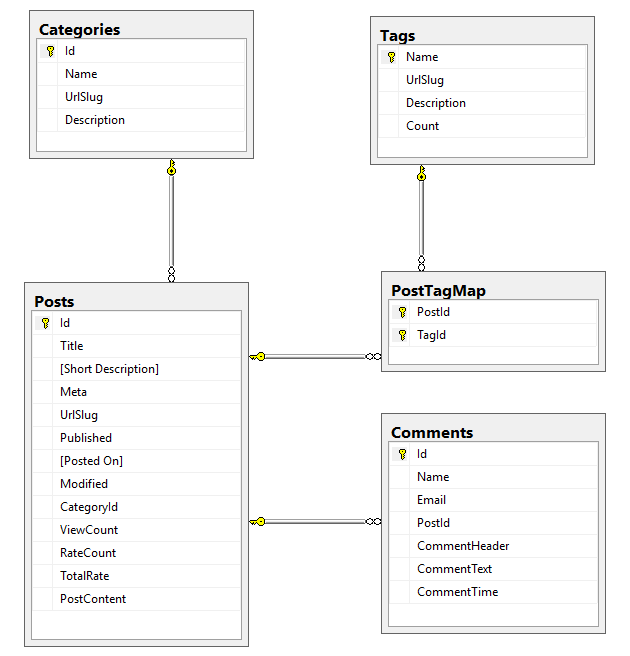
<https://www.entityframeworktutorial.net/code-first/code-based-migration-in-code-first.aspx>

### Tool used

Visual Studio 2022

### Post condition

All changes are applied successfully.

The Database Relationship is updated to:

Re-run all unit test cases and check result

### Estimated time

60 minutes

**-- THE END --**