EfBloggyImproved

It was not a direct assignment to perform, but we would do it more ourselves or in a group. The goal was that we would gain some knowledge entity framework (EF). In my opinion, this is an improved version of the basic code that we started working on " EfBloggy ". It is certainly deliberately done that these possibilities do not exist from the beginning, but you must realize this yourself a little later when we carry out a project where we can use previous tasks as a basis. May see what I will choose a little later in the course when it comes time to complete this project.

We got access to a console application to handle various blog posts with a default database. If you just read the instructions for the application, everything will go well… Have therefore added some code that checks if the database exists and if not, the user is prompted to run the Update-Database in the package manger console .

I was also a little annoyed that there was no separate table / class to handle the different writers for posts. Prefers to normalize data as far as possible! We had as an exercise task to add the possibility to add a new blog post or delete blog posts in the service. Also added a timestamp on the posts to make it a little more lifelike. Therefore, I developed this improved version of the code with one to many relationships in the database.

Of course, it may not be the smartest thing to work against a local data on the computer, but since the program will not work via the network, I see no direct problems with this! The code could have been written a little smarter and more general in some places, but to be a two-day job, I still feel quite happy with the result-

It also did not come with any code documentation for the demo project!

Migrations

Directory for Visual Studios' migration management for the database. It has to take care of itself, so to speak.

App

The "main program file" itself or how to express it, handles everything that has to do with the visual against the user. All database management takes place in a file called Functions .

Run ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called at startup of the program, first checks if the database exists. If not then the user is prompted to run the Update Database in the Package Mangager Console . Then it is verified that there is some basic data in the database for the user to test the program with. Then the function itself is called to render the menu itself.

ShowAllBlogPostsBrief ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that retrieves all blog posts, of course it would have been possible to limit the number of posts that are retrieved and sorted on the timestamp, etc. But since this will not be used in practice, it feels pointless to put energy into this. It's easy to just add this, etc., then you have put too much energy into something that will not be used.

ShowAllBlogers ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for presenting all bloggers in the service, here there is also no sorting by name or when these were added to the service. As I said, a balancing question about how much time you should spend on an exercise.

MainMenu ( )

A parameter: None a parameter

Out parameter: No out parameter

Function to present the main menu itself in the program, it calls a number of sub-functions based on which key the user presses.

AddPost ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for adding a new blog post to the program. First request that you indicate who the author is via either ID or name, and verify that this exists in the program Then request the actual title of the blog post. If all requirements are met, the post will be added to the database via another function in the code.

PageUpdatePost ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for updating a blog post in the program. First request that you enter the ID of the item you want to edit. If this is available, the title is presented and you are offered to update the content of the post.

DeletePost ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for deleting a blog post in the program. First request that you enter the ID of the item you want to delete. If this is available, the title itself is presented and you must confirm that you want to delete the post.

showEditerBlogerMenu ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that renders the menu for handling the various blogs in the service.

DeleteBloger ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is called when you want to delete a blogger in the service, is based on the ID or the name of the blogger you want to delete. Of course, you have to confirm that this is the blogger you want to delete.

UpdateBloger ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for updating a blogger's information. Right now just the name of this person. There is also a check that the new name is available in the database.

AddBloger ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for adding a blogger information. After the name has been entered, it is checked whether this is available in the database.

Header ( )

A parameter:

String text: Text to be formatted and presented to the user

Out parameter: Nothing is returned

Function for putting green color on the text that enters the function, and ensures that it is printed in capital letters.

WriteLine ( )

A parameter:

String text: Text to be formatted and presented to the user

Out parameter: Nothing is returned

Function for adding white color to the text that enters the function and adds a line break.

Write ( )

A parameter:

String text: Text to be formatted and presented to the user

Out parameter: Nothing is returned

Function to add white color to the text that enters the function.

BlogContext

Script file to create a "connection" to the local EF database

BlogPost

Script file that defines two classes used by the built-in tools in EF to create corresponding tables and relationships between them.

The two classes are:

BlogAuthor

A class that defines everything that applies to a blogger, has one to many relationships regarding the blogger and the posts.

BlogPost

Class that defines everything applies around a blog post, many have a relationship regarding the blog post and the blogger.

Functions

Script file that handles everything that concerns the database itself, such as CRUD.

ClearDatabase ( )

A parameter: None a parameter

Out parameter: No out parameter

Function that is used to delete the database in a smooth way, remains only to facilitate future development of the code.

AddSomeDefaultData ( )

A parameter: None a parameter

Out parameter: No out parameter

Function for inserting some basic data if the database is empty, just to give users a nice first impression.

AddPost ( )

A parameter:

string title , the title of the blog post itself
int id, id on the blogger itself to which the post is to be linked

Out parameter: No out parameter

Function for adding a blog post to the database via EF.

RemovePost ( )

A parameter:

int id, id on the post itself to be deleted.

Out parameter: No out parameter

Function for deleting a blog post based on its ID in the database

DeleteBloger ( )

A parameter:

int id, id of the blogger himself to be deleted.

Out parameter: No out parameter

Function for deleting a blogger from the database.

AddBloger ( )

A parameter:

BlogAuthor author , class with the blogger himself to be added to the database we EF.

Out parameter: No out parameter

Function for adding a blogger to the database.

Program

Script file that starts the program itself when it runs.

Main ( )

A parameter:

Common parameters that can be attached when starting the program, but it is not used by the program.

Out parameter: No out parameter

The state function itself for the program, does not do much more than call the start of the program, which checks some basic requirements and then requests a printout of the menu and we are up and running.

TestConnection

Script file that is used to check if we have a database on the server we are working against.

TestConnection ( )

A parameter:

this DbContext context , the "connection" that we are going to verify works as intended. We are therefore checking whether we can make a connection against the intended "database connection".

Out parameter:

Bool , true if we can make a connection when the database exists, otherwise false which means that the database is missing.