Demo Bloggy ADO

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 about databases resp. SQL. None of this was any direct news to me, more to teach me the slightly more specific SQL that applies to MS SQL server.

We got access to a console application to handle various blog posts with some associated SQL scripts. After a bit of hassle and clumsiness on my part, I got it up so I could start the program. It made me realize that you make sure that the program verifies that the database exists locally, and that the associated tables exist with certain data.

I was also a little annoyed that there was no separate table 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 realistic.

From what I understand, we will not use so much of ADO in our professional lives, but it is the Entity Framework that mainly applies when I write this. Thanks to the code lines that we gained access to via the program, I gained a better understanding of this framework and how to work with it. Of course I could have looked on the internet for code examples, but I got it served, so to speak.

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!

Only follows the folder structure that we would work according to, in retrospect I have realized the benefit of working further with someone else's code. It is important to understand how everything is connected, etc. It was an old project they handed over to us when it was based on .NET Framework 2.1, but since there were no special functions in the project, it was easy to change to .NET Core 5/6

Domain

Folder that contains classes used in the service, I have chosen to add a new class in the form of Au BlogAuthor thor to manage writers in the service.

BlogAuthor

Class for dealing with different authors in the service.

BlogPost

Class for managing the various blog posts in the service.

SQLScripts

Folder with various SQL to set up the database and associated table for the blog posts but also to clear it.

Insert

SQL scripts that I may have modified to ensure that it also creates the associated table for data to be inserted

Recreate

SQL script for deleting and recreating the table for blog posts, has added that it should enter the data in the table also compared to "our" original.

Select

SQL script to present all the posts in the BlogPost table

App

File to handle everything presented to the user when this service!

Run ()

In parameter: No in parameters
Out parameter: Nothing is returned

Calls function to set up the database if necessary, then the function is called to present the basic menu in the program.

PageMainMenu ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function that pulls up the main menu for the program. As well as calling appropriate functions when the user requests this.

AddNewPost ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for requesting information about what is needed to add a blog post to the service.

PageUpdatePost ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for requesting information about what is needed to update a blog post in the service.

DeleteBlogPost ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for requesting information about what is needed to delete a blog post in the service.

ShowWriters ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function to present all bloggers in the service.

AddNewWriter ()

In parameter: No in parameters

Out parameter: Nothing is returned

Function for adding a new blogger to the service.

PageUpdatWriter ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for updating information about a blogger in the service.

DeleteWriter ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for deleting a blogger in the service.

manageWriterMenu ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for building a menu system for managing the various blogs in the service.

ShowAllBlogPostsBrief ()

In parameter: No in parameters
Out parameter: Nothing is returned

Function for presenting all blog posts that are in the database, presents data that comes from an SQL query that has been set against 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.

DataAccess

Script file with all functions that perform different types of work against the database

checkDatabase ()

In parameter: No in parameters

Out parameter: Nothing is returned

Function for building up the database and associated tables in the service.

GetAllBlogPostsBrief

In parameter: No in parameters

Out parameter: Nothing is returned

Function for retrieving posts from the database, however, there is no restriction or sorting around the retrieved posts even if I could have entered it in the SQL statement.

GetPostById ()

A parameter:

int postId: id of the post to be retrieved

Out parameter:

BlogPost returns a blog post all that is linked to the post.

Function for retrieving a blog post via its ID.

UpdateBlogpost ()

A parameter:

BlogPost blogPost Variable with the data to be updated in the table linked in the database.

Out parameter: Nothing is returned

Function that updates a blog post in the database.

ValidateAuthor ()

A parameter:

string Author, a string that either contains the username or its ID in the database

Out parameter:

bool exists true if the user exists

int authorId, the user's possible ID in the database

Function to validate whether the user exists in the database or not, search can be done via ID or username.

ListAuthors ()

A parameter: None a parameter

Out parameter:

List <BlogAuthor> A list of all bloggers in the service

Function for listing bloggers in the service.

AddBlogPost ()

A parameter:

BlogPost post, the post to be added to the database.

Out parameter: No out parameter

Function for adding a blog post to the database

GetAllWriters ()

A parameter: None a parameter

Out parameter:

List <BlogAuthor> A list of all bloggers

Function for retrieving all bloggers in the system.

CheckIfWriterExists ()

A parameter:

Object author, user name or its ID to be verified

Out parameters:

bool exists true if the user exists

BlogAuthor data, returns data about the blogger in the event of a match.

AddBlogWriter ()

A parameter:

string author, string with the username of the blogger

Out parameter: Nothing is returned

Function for adding bloggers to the service.

UpdateBlogWritert ()

A parameter:

BlogAuthor author, variable with info about the blogger to be updated in the database.

Out parameter: Nothing is returned

Function for updating information about a blogger.

DeleteBlogWriter ()

A parameter:

Int id: id of the blogger to be deleted from the system.

Out parameter: Nothing is returned

Function for deleting a blogger in the database.