Lab Exercise 3: C# .NET 6 Class Library --- CONTINUATION OF LAB EXERCISE 2

Setting up the UI

1. In BlogTestUI, right click Dependencies, and click Add Project Reference
2. Check BlogDataLibrary.

3. In BlogTestUI, right click > Add > New Item
4. In the search bar, search for JSON.
5. Select JavaScript JSON Configuration file, and name it as appsettings.json
6. After the “exclude” key, add the following configuration:

```
"exclude": [
    "**/bin",
    "**/bower_components",
    "**/jspm_packages",
    "**/node_modules",
    "**/obj",
    "**/platforms"
  ],
"ConnectionStrings": {
    "SqlDb": "<connection string here>"
  }
```

Ensure that the “SqlDb” keyword is the same as the constant connectionStringName in BlogDataLibrary’s SqlData.cs

7. To get the connection string, go to SQL Server Object Explorer, and expand the database.
8. Right click the database and select Properties.
9. Under properties, in the connection string key, copy the value and paste it in the code.

10. In Program.cs, add a new static method called GetConnection() and add the following code inside:

```
static SqlData GetConnection()
    {
        var builder = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("appsettings.json");

        IConfiguration config = builder.Build();
        ISqlDataAccess dbAccess = new SqlDataAccess(config);
        SqlData db = new SqlData(dbAccess);

        return db;
    }
```

11. In the main method, add the following code:

```
static void Main(string[] args)
    {
        SqlData db = GetConnection();
    }
```

App functionality: Login

1. In DB Project, right click the root and add a new folder named Stored Procedures
2. Right click Stored Procedures and add Stored Procedure
3. Name it spUsers_Authenticate
4. Type in the following SQL code:

```sql
CREATE PROCEDURE [dbo].[spUsers_Authenticate]
    @username nvarchar(16),
    @password nvarchar(16)
AS
begin
    set nocount on;

    SELECT [Id], [UserName], [FirstName], [LastName], [Password]
    FROM dbo.Users
    WHERE UserName = @username
    AND Password = @password;

end
```

*You can use SELECT *, and right click the asterisk > refactor > expand wildcards and it will automatically list out the columns.*
Ensure that the parameters (@username, @password) has the same datatype as the configuration in the database

5. In BlogDataLibrary, add a new method to SqlData named Authenticate(). Type in the following code:

```csharp
public UserModel Authenticate(string username, string password)
    {
        UserModel result = _db.LoadData<UserModel, dynamic>("dbo.spUsers_Authenticate",
                                                            new { username, password },
                                                            connectionStringName,
                                                            true).FirstOrDefault();

        return result;
    }
```

FirstOrDefault() returns the first row of a set of results, or null.

6. In BlogTestUI, add a new method GetCurrentUser() after main, and add the following code:

```
private static UserModel GetCurrentUser(SqlData db)
    {
        Console.Write("Username: ");
        string username = Console.ReadLine();

        Console.Write("Password: ");
        string password = Console.ReadLine();

        UserModel user = db.Authenticate(username, password);

        return user;
    }
```

7. Do the same for Authenticate():

```
public static void Authenticate(SqlData db)
    {
        UserModel user = GetCurrentUser(db);

        if (user == null)
        {
            Console.WriteLine("Invalid credentials.");
        }
        else
        {
            Console.WriteLine($"Welcome, {user.UserName}");
        }
    }
```

8. Call Authenticate() in the main method and pass the db parameter:

```
static void Main(string[] args)
    {
        SqlData db = GetConnection();

        Authenticate(db);

        Console.WriteLine("Press Enter to exit...");
        Console.ReadLine();
    }
```

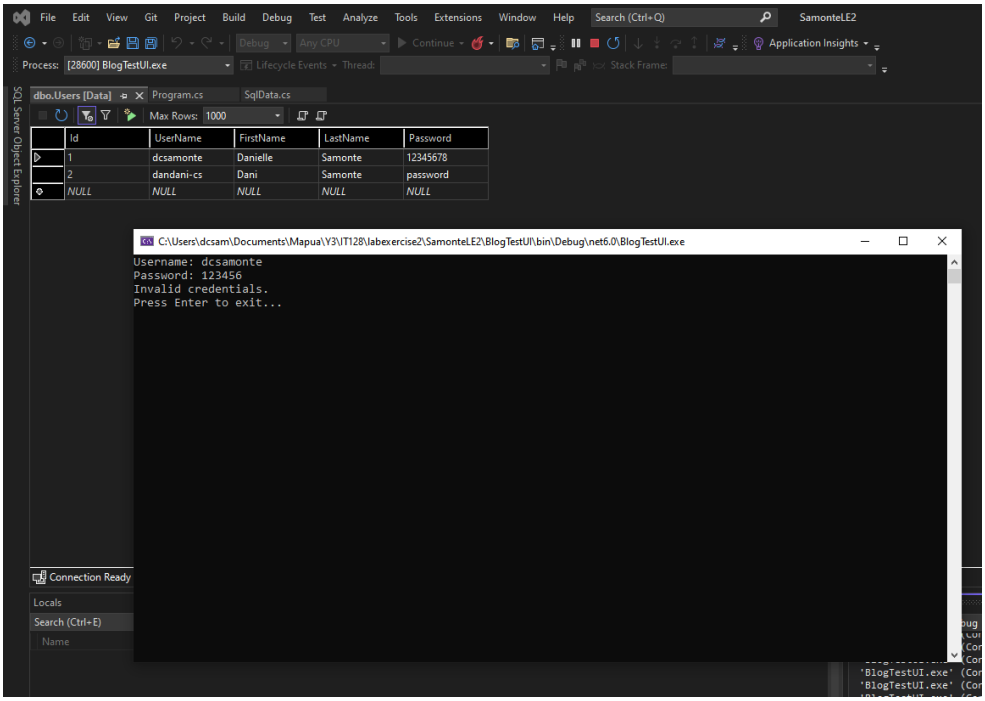
Publishing changes in DB

Because a change was made in the database by adding a stored procedure. The following steps will be reused every time a stored procedure is added.

1. Right click the DB project > Publish
2. Browse for the MSSQLLocal DB for the target database connection
3. Click Load Profile, and select the profile made when initially publishing the database.
4. Click Publish.

App functionality: Login Test

1. Right click BlogTestUI and set it as startup project.
2. Run the project by clicking the play button on the ribbon. Ensure it says “BlogTestUI”
3. Screenshot the login process using the given credentials in the Users Table. Try to give incorrect and correct credentials. Example:



App functionality: Register

1. Create a new stored procedure in the DB Project, and name it as "spUsers_Register."
2. Type in the following code:

```
CREATE PROCEDURE [dbo].[spUsers_Register]
    @userName nvarchar(16),
    @firstName nvarchar(50),
    @lastName nvarchar(50),
    @password nvarchar(16)
AS
begin
    set nocount on;

    INSERT INTO dbo.Users
    (UserName, FirstName, LastName, Password)
    VALUES (@userName, @firstName, @lastName, @password)
end
```

3. Save and publish the database. Refer to the section Publishing changes in DB.
4. In BlogDataLibrary, in SqlData.cs, add a new method called Register and type in the following code:

```
public void Register(string username, string firstName, string lastName, string password)
    {
        _db.SaveData<dynamic>(
            "dbo.spUsers_Register",
            new { username, firstName, lastName, password },
            connectionStringName,
            true);
    }
```

5. In BlogTestUI, add another method called Register with the following code:

```
public static void Register(SqlData db)
        {
            Console.Write("Enter new username: ");
            var username = Console.ReadLine();

            Console.Write("Enter new password: ");
            var password = Console.ReadLine();

            Console.Write("Enter first name: ");
            var firstName = Console.ReadLine();

            Console.Write("Enter last name: ");
            var lastName = Console.ReadLine();

            db.Register(username, firstName, lastName, password);
        }
```

6. Call the Register method in main, and pass in the db parameter.
7. Run the program, and add a new set of credentials for a new user. Ensure to use your first name and last name. Screenshot the process, and include the data view of users.
   There is a refresh button in the data view.

App functionality: Adding posts

1. In the DB Project, add a new stored procedure named spPosts_Insert.
2. Add the following code:

```sql
CREATE PROCEDURE [dbo].[spPosts_Insert]
    @userId int,
    @title nvarchar(150),
    @body text,
    @dateCreated datetime2
AS
begin
    INSERT INTO dbo.Posts
    (UserId, Title, Body, DateCreated)
    VALUES
    (@userId, @title, @body,
@dateCreated)
end
```

3. Make sure to publish the changes to the database before proceeding.
4. In BlogDataLibrary, add a new method called AddPost and write the following code:

```csharp
public void AddPost(PostModel post)
        {
            _db.SaveData("spPosts_Insert", new { post.UserId, post.Title, post.Body, post.DateCreated },
connectionStringName, true);
        }
```

5. In BlogTestUI, add the following code to a new method called AddPost:

```
private static void AddPost(SqlData db)
    {
        UserModel user = GetCurrentUser(db);

        Console.Write("Title: ");
        string title = Console.ReadLine();

        Console.WriteLine("Write body: ");
        string body = Console.ReadLine();

        PostModel post = new PostModel
        {
            Title = title,
            Body = body,
            DateCreated = DateTime.Now,
            UserId = user.Id
        };

        db.AddPost(post);
    }
```

6. Call the method in main, and pass in the db parameter.
7. Run the program, and add a new post. Set the following details:
   Title: First post of <lastname>
   Body: This is my first post, by <full name>
8. Screenshot the process, and include the data in SQL Server data view of the Posts Table.

App functionality: List of posts

1. In DB project, add a new stored procedure named “spPosts_List”
2. Add the following code:

```
CREATE PROCEDURE [dbo].[spPosts_List]
AS
begin
    set nocount on;

    SELECT [p].[Id], [p].[Title], [p].[Body], [p].[DateCreated], [u].[UserName], [u].[FirstName], [u].
[LastName]
    FROM dbo.Posts p
    INNER JOIN dbo.Users u
    ON p.UserId = u.Id
end
```

Joins collects the information from another table using the foreign key reference.

3. Make sure that the changes are published.
4. In BlogDataLibrary/SqlData.cs, add a new method called ListPosts().
5. Add the following code:

```
    public List<ListPostModel> ListPosts()
        {
            return _db.LoadData<ListPostModel, dynamic>("dbo.spPosts_List", new { },
connectionStringName, true).ToList();
        }
```

6. In BlogTestUI, add a new method called ListPosts() and code the following:

```
private static void ListPosts(SqlData db)
        {
            List<ListPostModel> posts = db.ListPosts();

            foreach (ListPostModel post in posts)
            {
                Console.WriteLine($"{post.Id}. Title: {post.Title} by {post.UserName}
[{post.DateCreated.ToString("yyyy-MM-dd")}]");
                Console.WriteLine($"{post.Body.Substring(0, 20)}...");
                Console.WriteLine();

            }
        }
```

7. Call the method in main and pass in the parameter db.
8. Screenshot the results when running the program. It should list the post id, title, username of author, date created, and the first 20 characters of the body.

App functionality: Showing post details

1. In DB Project, add a new stored procedure called “spPosts_Detail”
2. Add the following code:

```sql
CREATE PROCEDURE [dbo].[spPosts_Details]
    @id int
AS
begin
    set nocount on;

    SELECT [p].[Id], [p].[Title], [p].[Body], [p].[DateCreated], [u].[UserName], [u].[FirstName], [u].
[LastName]
    FROM dbo.Posts p
    INNER JOIN dbo.Users u
    ON p.UserId = u.Id
    WHERE p.Id = @id;

end
```

3. Publish the changes.
4. In BlogDataLibrary, add a new method in SqlData.cs called ShowPostDetails and add:

```csharp
public ListPostModel ShowPostDetails(int id)
        {
            return _db.LoadData<ListPostModel, dynamic>("dbo.spPosts_Details", new { id },
connectionStringName, true).FirstOrDefault();
        }
```

5. In BlogTestUI, add a new method called ShowPostDetails and type the following:

```csharp
    private static void ShowPostDetails(SqlData db)
        {
            Console.Write("Enter a post ID: ");
            int id = Int32.Parse(Console.ReadLine());

            ListPostModel post = db.ShowPostDetails(id);
            Console.WriteLine(post.Title);
            Console.WriteLine($"by {post.FirstName} {post.LastName} [{post.UserName}]");

            Console.WriteLine();

            Console.WriteLine(post.Body);

            Console.WriteLine(post.DateCreated.ToString("MMM d yyyy"));
        }
```

6. Call the method in main, and pass in the parameter db.
7. Run the program, and screenshot the details of the first post made.

Post-program requirements

1. Go to SqlData.cs in BlogDataLibrary.

2. Right click the declaration of “public class SqlData” > Quick Quick Actions and Refactoring > Extract Interface
3. Accept default values.
4. Add 2 more users and 2 more post. Screenshot the processes and the tables containing the data.

5. Screenshot the Solution Explorer. Ensure that all folders are expanded.

6. Copy and paste all the code in BlogDataLibrary/SqlData.cs and BlogTestUI/Program.cs

Put all screenshots of this activity with the link of your GitHub repository in a MS Word file and submit it in the LE3-1 Blackboard link provided. Filename should be LE3-1-SURNAME

Demonstrate your program and explain how it works. Discuss the functionalities and how did you do it. Submit an MP4 video in the LE3-VID Blackboard link provided. Filename should be LE3-VID-SURNAME.