Testing CRUD Functions

In this lesson, the whole project will be compiled to test all the functions discussed in the previous lessons.

WE'LL COVER THE FOLLOWING

In the previous lessons, you saw the implementation for the *create*, *read*, *update* and *delete* operations.

In this lesson, we will test these functions by running the test case file, *MongoDb.dll*, that is created when the Github project compiles.

> All the test cases should pass if the implementations are correct.

You can go through the file *MongoDbRepositoryTests.cs*, given below, in order to view the different test cases written for the functions.

Running the Test File

Run the code below in order to test the functions discussed earlier.

The output of this executable will be displayed in the terminal tab.

```
using MongoDB.Bson;
using System.Linq;
using System.Threading.Tasks;
using Xunit;

namespace mongonetcore
{
    /// <summary>
    /// Testing MongoDbRepository class.
    /// </summary>
    /// <notes>
    /// In order for these tests to pass, you must have mongo server runing on localhost:2701
    /// If you need more info on how to do so check this blog post:
```

```
/// If you need more info on how to do so check this blog post:
/// https://rubikscode.net/2017/07/24/mongo-db-basics-part-1/
/// </notes>
public class MongoDbRepositoryTests
{
    UsersRepository _mongoDbRepo = new UsersRepository("mongodb://127.0.0.1:27017");

    private async Task Initialize()
    {
        var user = new User()
        {
            Name = "Nikola",
            Age = 30,
            Blog = "rubikscode.net",
            Location = "Beograd"
        };
        await _mongoDbRepo.InsertUser(user);

        user = new User()
        {
            Name = "Vanja",
            Age = 27,
            Blog = "eventroom.net",
            Location = "Beograd"
        };
        await _mongoDbRepo.InsertUser(user);
    }

    private async Task Cleanup()
    {
        await _mongoDbRepo.DeleteAllUsers();
    }

    [Fact]
    public async void CheckConnection_DbAvailable_ConnectionEstablished()
    {
        await Initialize();

        var connected = _mongoDbRepo.CheckConnection();
        Assert.True(connected);

        await Cleanup();
    }

    /// <summary>
    /// Test is ignored, because it lasts 30 seconds.
    /// </summary>
    [Fact]
    public void CheckConnection_DbNotAvailable_ConnectionFailed()
    {
        var mongoDbRepo = new UsersRepository("mongodb://127.0.0.1:27016");
        var connected = mongoDbRepo.CheckConnection();
        Assert.False(connected);
    }

    [Fact]
    public async Task GetAllUsers_ReadAllUsers_CountIsExpected()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetAllUsers();
        Assert.Equal(2, users.Count);
```

```
        await Cleanup();
    }

    [Fact]
    public async Task GetUserByField_GetUserByNameAndUserExists_UserReturned()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("name", "Nikola");
        Assert.Equal(1, users.Count);

        await Cleanup();
    }

    [Fact]
    public async Task GetUserByField_GetUserByBlogAndUserExists_UserReturned()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("blog", "rubikscode.net");
        Assert.Equal(1, users.Count);

        await Cleanup();
    }

    [Fact]
    public async Task GetUserByField_GetUserByNameAndUserDoesntExists_UserNotReturned()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("name", "Napoleon");
        Assert.Equal(0, users.Count);

        await Cleanup();
    }

    [Fact]
    public async Task GetUserByField_WrongField_UserNotReturned()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("badFieldName", "value");
        Assert.Equal(0, users.Count);

        await Cleanup();
    }

    [Fact]
    public async Task GetUserCount_JustFirstElement_Success()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsers(0, 1);
        Assert.Equal(1, users.Count);

        await Cleanup();
    }

    [Fact]
    public async Task InsertUser_UserInserted_CountIsExpected()
    {
        await Initialize();
```

```
        var user = new User()
        {
            Name = "Simona",
            Age = 0,
            Blog = "babystuff.com",
            Location = "Beograd"
        };

        var users = await _mongoDbRepo.GetAllUsers();
        var countBeforeInsert = users.Count;

        await _mongoDbRepo.InsertUser(user);

        users = await _mongoDbRepo.GetAllUsers();
        Assert.Equal(countBeforeInsert + 1, users.Count);

        await Cleanup();
    }

    [Fact]
    public async Task DeleteUserById_UserDeleted_GoodReturnValue()
    {
        await Initialize();

        var user = new User()
        {
            Name = "Simona",
            Age = 0,
            Blog = "babystuff.com",
            Location = "Beograd"
        };

        await _mongoDbRepo.InsertUser(user);

        var deleteUser = await _mongoDbRepo.GetUsersByField("name", "Simona");
        var result = await _mongoDbRepo.DeleteUserById(deleteUser.Single().Id);

        Assert.True(result);

        await Cleanup();
    }

    [Fact]
    public async Task DeleteUserById_UserDoesntExist_NothingIsDeleted()
    {
        await Initialize();

        var result = await _mongoDbRepo.DeleteUserById(ObjectId.Empty);

        Assert.False(result);

        await Cleanup();
    }

    [Fact]
    public async Task DeleteAllUsers_DelitingEverything_Sucess()
    {
        await Initialize();

        var result = await _mongoDbRepo.DeleteAllUsers();
```

```
        Assert.Equal(2, result);

        await Cleanup();
    }

    [Fact]
    public async Task UpdateUser_UpdateTopLevelField_UserModified()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("name", "Nikola");
        var user = users.FirstOrDefault();

        await _mongoDbRepo.UpdateUser(user.Id, "blog", "Rubik's Code");

        users = await _mongoDbRepo.GetUsersByField("name", "Nikola");
        user = users.FirstOrDefault();

        Assert.Equal("Rubik's Code", user.Blog);

        await Cleanup();
    }

    [Fact]
    public async Task UpdateUser_UpdateTopLevelField_GoodReturnValue()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("name", "Nikola");
        var user = users.FirstOrDefault();

        var result = await _mongoDbRepo.UpdateUser(user.Id, "blog", "Rubik's Code");

        Assert.True(result);

        await Cleanup();
    }

    [Fact]
    public async Task UpdateUser_TryingToUpdateNonExistingUser_GoodReturnValue()
    {
        await Initialize();

        var result = await _mongoDbRepo.UpdateUser(ObjectId.Empty, "blog", "Rubik's Code"

        Assert.False(result);

        await Cleanup();
    }

    [Fact]
    public async Task UpdateUser_ExtendingWithNewField_GoodReturnValue()
    {
        await Initialize();

        var users = await _mongoDbRepo.GetUsersByField("name", "Nikola");
        var user = users.FirstOrDefault();

        var result = await _mongoDbRepo.UpdateUser(user.Id, "address", "test address");

        Assert.True(result);
```

```
            await Cleanup();
        }

    }
}
```

Now that you have learned how to use MongoDB in C#, it’s time for a short quiz!!