# Data-Driven Apps

These are apps, which need to store data in a database to work properly.

## Book Library

The Book Library we’re going to create is a lot like the TODO List, with two main differences:

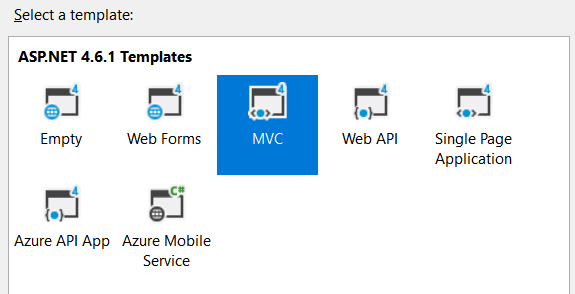
* We have users and use authorization for actions
* We can **edit books** (change title, etc...). Last time we could only create and delete them.

If you successfully complete all steps you will have a “**Book Library**” application with the following functionality:

* **Register User**
* **Login User**
* **Create New Book**
* **Edit Existing Book**
* **Delete Existing Book**
* **List All Books**

### Create a New Project

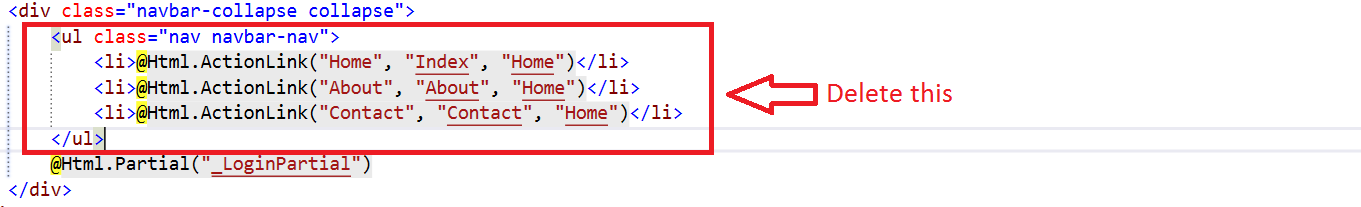
Creating the project is similar to the “**TODO List**” problem. This time leave the authentication to “**Individual User Authentication**”:



**Important: Don’t run the project before step 7.**

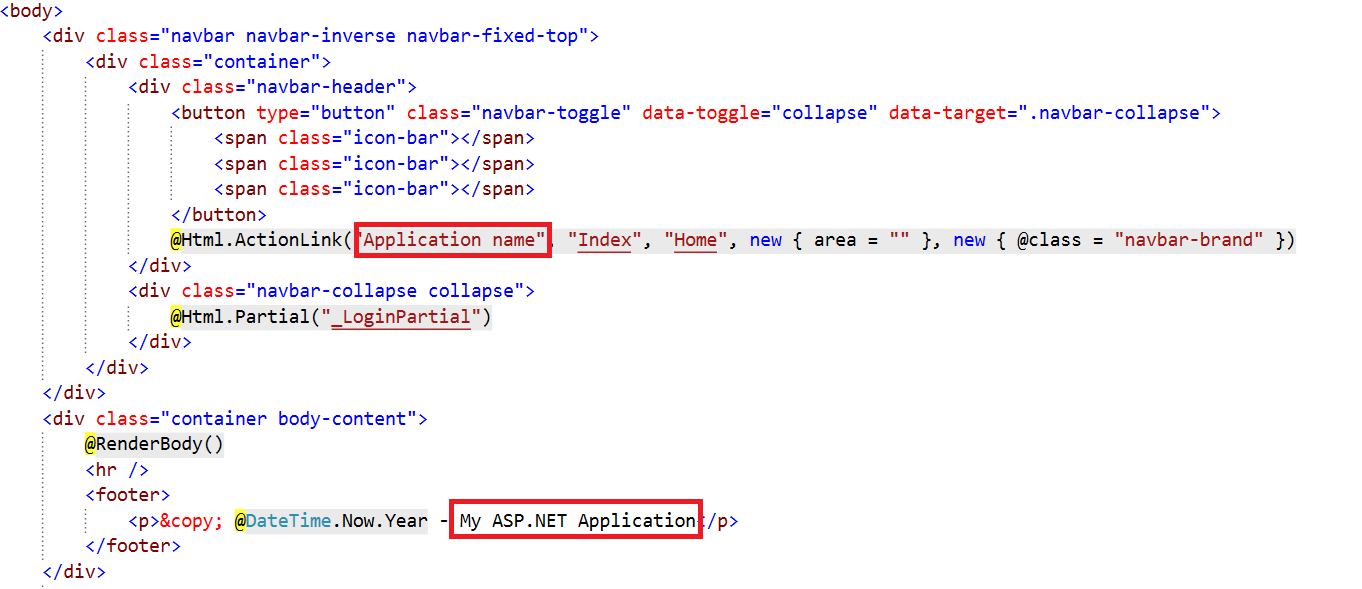
### Edit the Main Layout

Let’s start by deleting the unnecessary lines from „\_Layout.cshtml“.



### Edit the Application Name

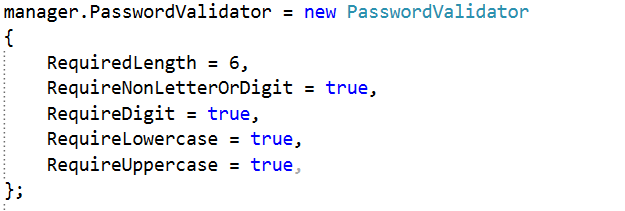
Go to your main layout and edit the following lines:



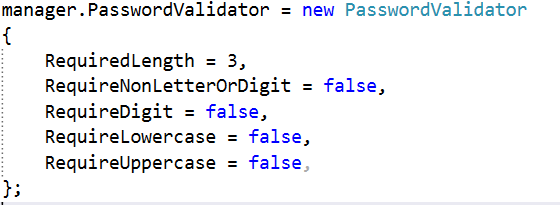
We are done with the layout for now.

### Simplify the Registration

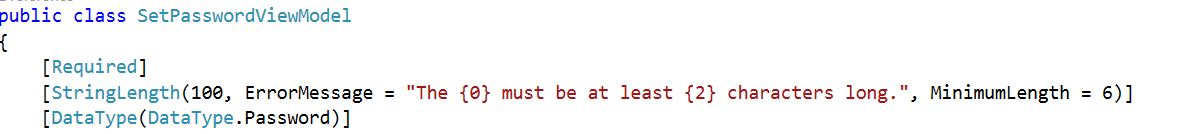
Go to your “App\_Start/IdentityConfig.cs” file. Find the following lines:



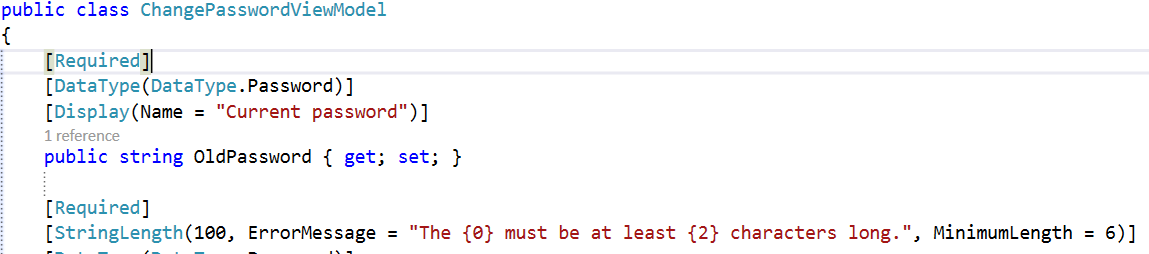
Edit them to receive this result:



This will simplify our password. Now we need to edit the models that validate our data. Go to the “ManageViewModels.cs”. Find this line:



Edit the minimum length to 3 symbols. You will find a similar line in the ChangePasswordViewModel class in the same file. Edit the **minimum length** there too:



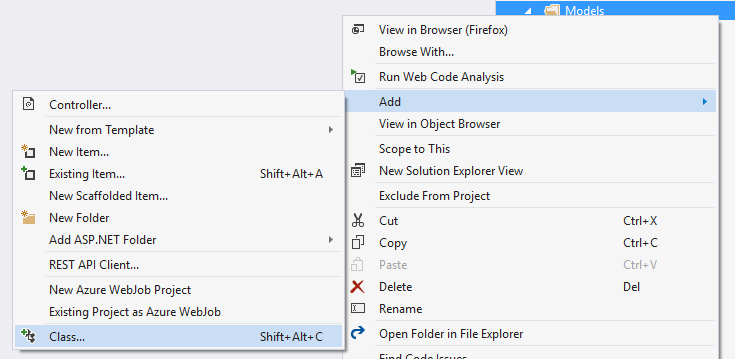
Use 3 symbols again.

Finally, we should go to the AccountViewModels.cs file. Find the lines that validate the password length, and change the value to 3 symbols. Here is how it should look like:

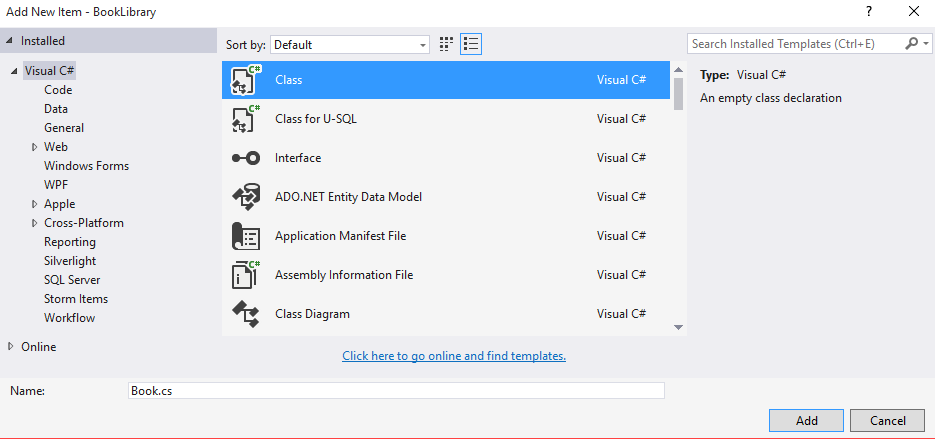


### Create the Book Model

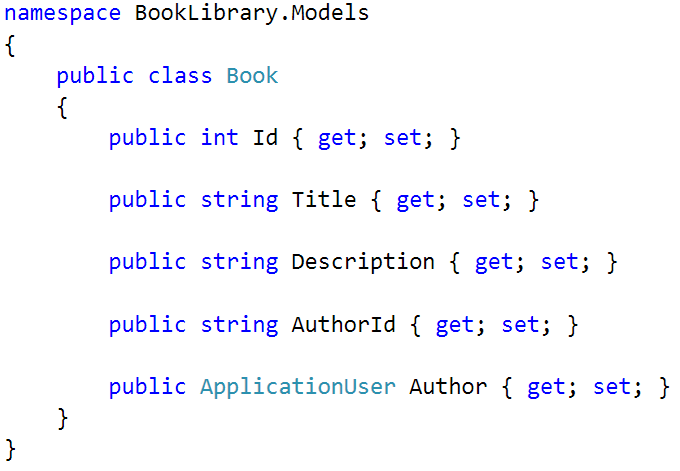
Create a new class in your **Models** folder:



Name the class Book.cs:



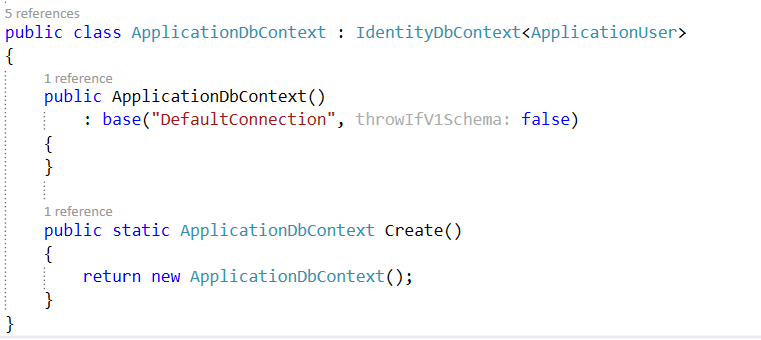
Add the following properties:



### Extract the DbContext

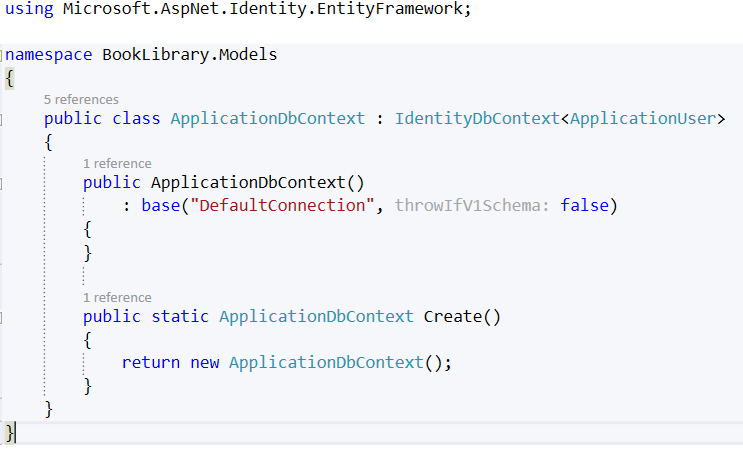
Create a new model and name it “ApplicationDbContext”.

Go to your IdentityModels.cs file and find the ApplicationDbContext class.

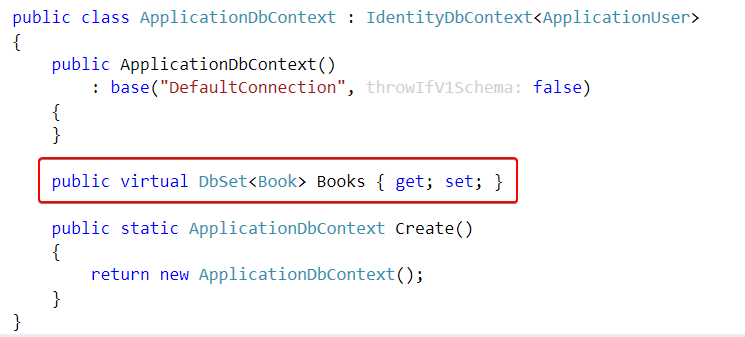


We need to **move** the class to a new file. We could either **cut** it and **paste** it into a file called ApplicationDbContext.cs.

After that, the ApplicationDbContext.csfile should look like this:

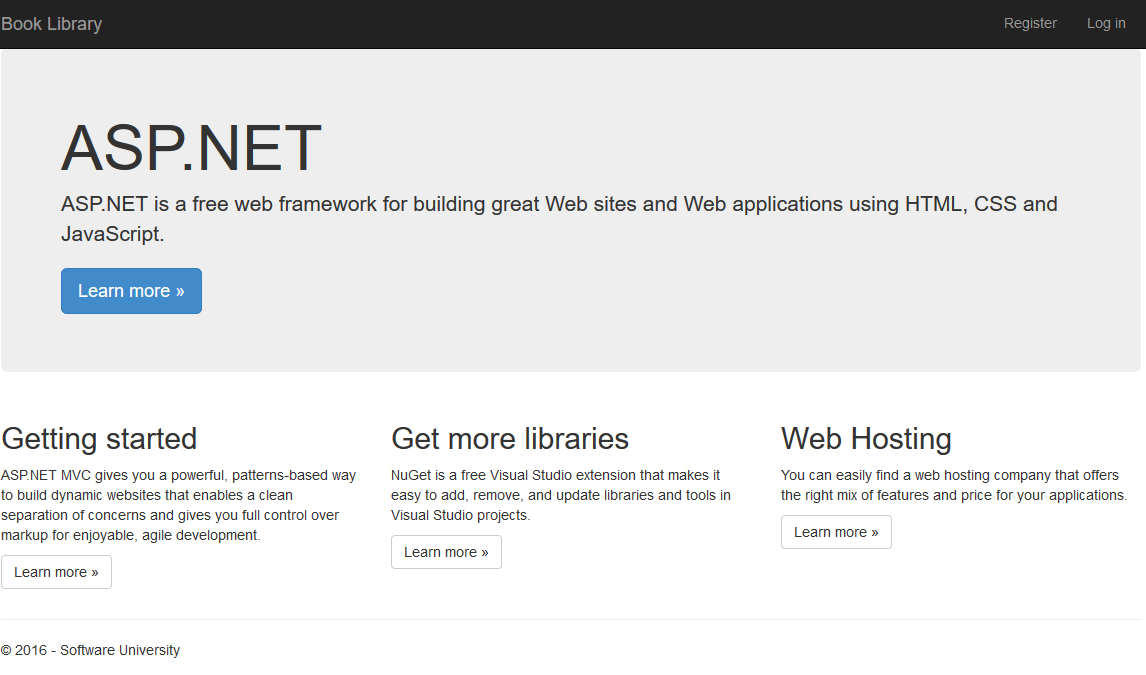


Add the following code:



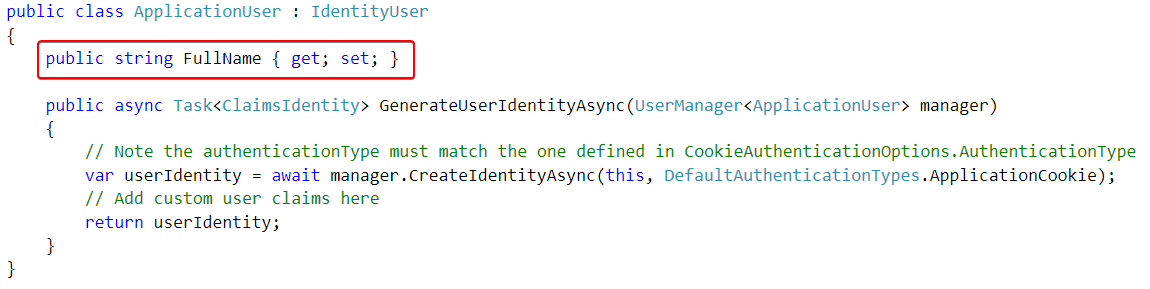
### Run the Application for the First Time

Don’t register a new user just yet! You should see this:

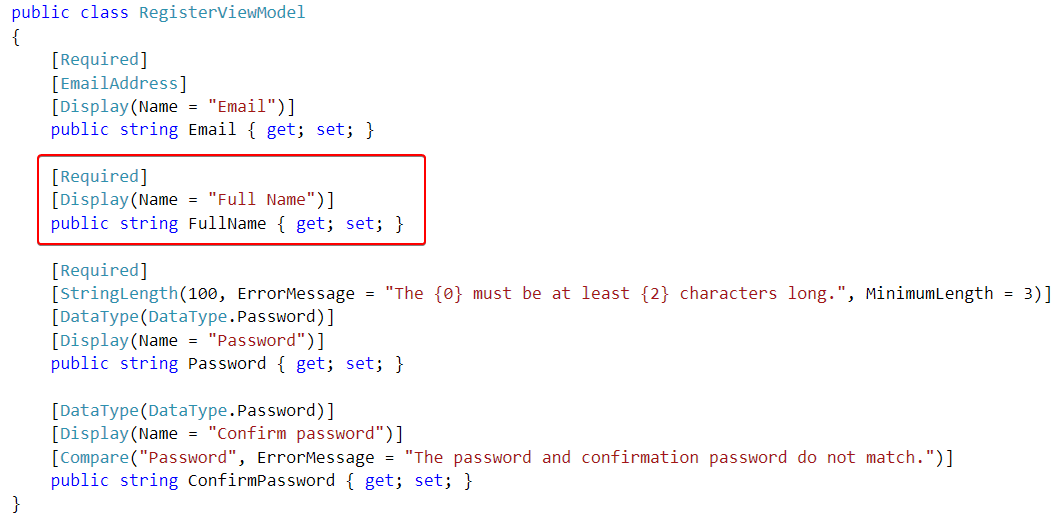


### Add Full Name to Our User

Go to your IdentityModels.cs file and write the following line:



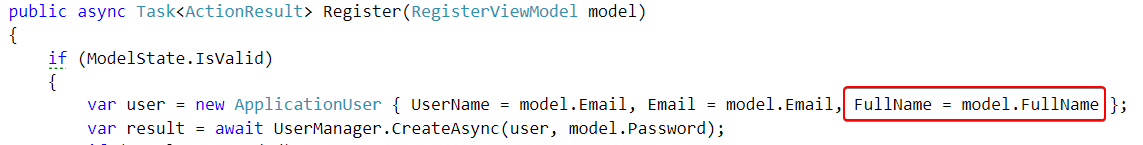
Now go to the AccountViewModels.cs and find the RegisterViewModel class. Add the following lines:



We are almost done. Go to your AccountController.cs. Find the Register **post** method:



Add this, to the existing line:

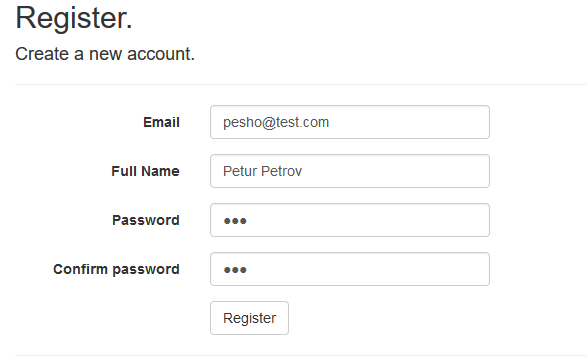


Finally go to the register view located in “Views/Account/Register.cshtml”. Write the following code:



### Register New User

Run the application and try to register a new user:

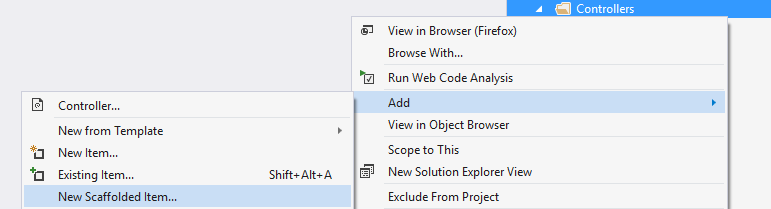


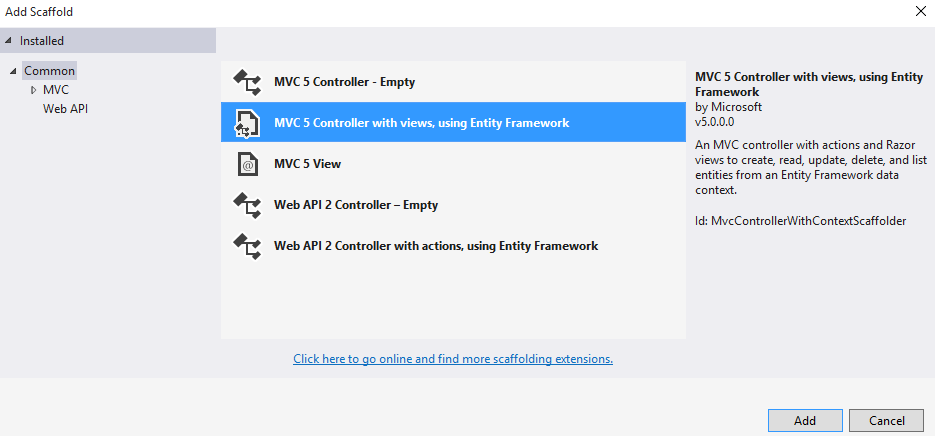
If it works you will be logged in, and you will see welcome message like this:



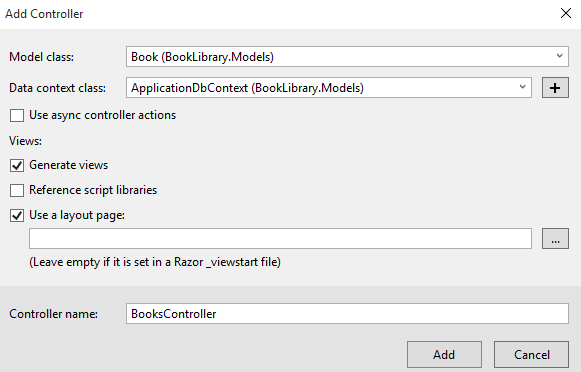
### Create the Book Controller

Add a **new scaffolded item** to your **Controllers** folder:



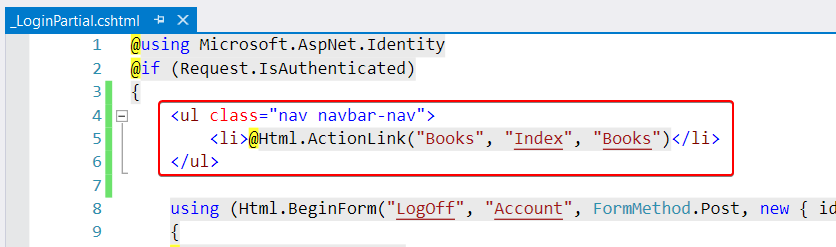


Use your Book **Model** and your ApplicationDbContext as shown in the image below:



### Edit the User Layout

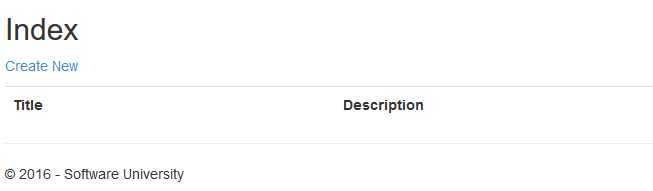
Go to the \_LoginPartial.cshtml layout and add the following code:



Now if you log in, you will see this button:



When you click on it, you will see this page:



**Don’t create new books yet!**

### Add Author Info

Right now, our **CRUD** operations **don’t** add the current user as an author of the books they created to our DB. We need to fix that. Find the Create method in your BooksController. It should look like this:



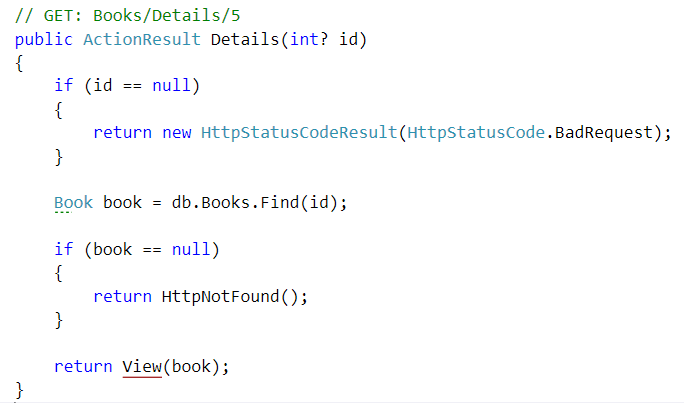
Add the following lines of code to the if statement:



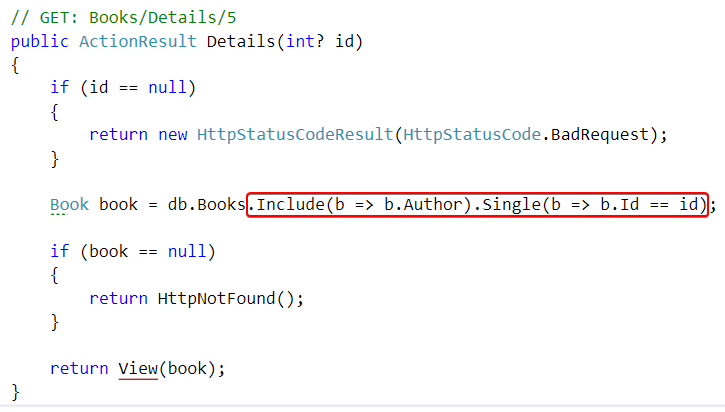
That will add the author to the book object.

### Show the Author

Our final step is to show the author of the book. We will do that by finding the Details method in our BookController. You should have something like this:

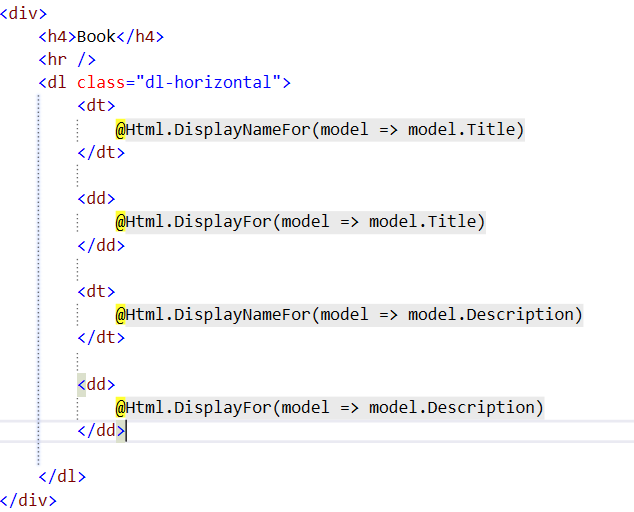


We will edit one of those lines, with LINQ query:

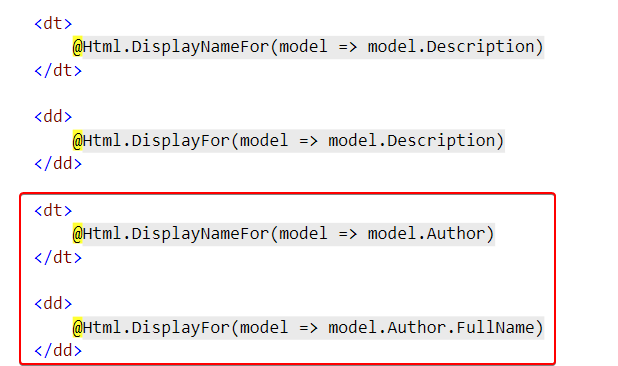


This will **explicitly request** the author’s information (full name, email, etc.) from the database alongside the book. We need to do this because by default Entity Framework doesn’t load properties from other entities. This is called **Lazy Loading**.

We are almost done. Let’s go to our “Views/Books/Details.cshtml” file. It should contain a div tag, that looks like this:



Add the following element:



You are ready to play with your Book Library now!