

Figure 1 Creating a new project supporting user login

Contents

[Basics 3](#_Toc103630787)

[Model 3](#_Toc103630788)

[View 3](#_Toc103630789)

[Controller 3](#_Toc103630790)

[Database connection 4](#_Toc103630791)

[Entity Framework 4](#_Toc103630792)

[Migrations 4](#_Toc103630793)

[DbContext 4](#_Toc103630794)

# Basics

**Model**  : Classes containing their own properties and methods (ex. User, Book)

**View**  : A representation of one or multiple classes

**Controller** : The controller that decides what views to show

## Model

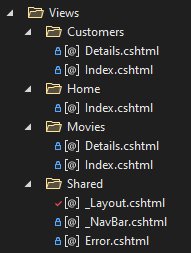
Contains a wide range of classes, each class represents an attribute used on your website. Attributes such as ‘User’, ‘Receipt’, ‘Menu’ or ‘Book’ to name a few



## View

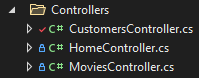
Classes can be visually represented using views. To put it simple: views are webpages created by the developer. Views can contain one or more classes; these classes will then be loaded into the view in places where the developer decided they would be.

Views can also be made for viewing one specific instance of a class (an object).  
The default view for each model is called Index.



## Controller

A controller decides what view and model to show to the user. It does this using the search bar. Once a user navigates to the website www.example.org/Movies/Details/1. The controller knows that it must display the Movies/Details using an object where the ID is 1. Controllers can take more advanced request such as filters.



# Database connection

## Entity Framework

This technique is used to talk to a database on a more abstract level.



Figure 2 Command flow from DbContext to Database

Our project will contain a Database Context class which manages Data Sets. These sets are classes that represent tables in our database. We use the Database Context and Linq to query a Data Set. The dataset will turn the Linq request into SQL and pass it through the database. The database then returns the results back to the dataset.

Entity Framework makes sure that everything done to a class will also be updated in the database.

## Migrations

To enable migrations, go to the NuGet package manager console and type:

enable-migrations

If the command isn’t found, execute the following command and try again (2022-05-16):

Install-Package EntityFramework -IncludePrerelease

Now there should be a new folder inside the project called ‘Migrations’. Meaning they were successfully enabled.

Every time you want to add a migration you use the following syntax:

Add-migration <unique\_migration\_key/id>

Whenever you want to sync your database, you use:

Update-database

When using code-first never just add data into the database, instead you want to use a migration.

### DbContext

Inside the file IdentityModels should be two classes: ApplicationUser & ApplicationDbContect. The second one is your DbContext and communicates with your database. Adding DbSets<> to this class as properties will let the DbContext know that is has to create a database table for that class.