Webtech Exercise 3
Team Yellow

For our project we chose MVC 4 with a Razor View Engine. We decided to use a predefined user management system called „SimpleMemberShipProvider“. We use only one DbContext that handles our database called „UserContext“, which can be found in our only model file. To setup your project you may choose a different Sql connection, we use one connection throughout the project which can be easily altered in our Web.config. The entry we use is:

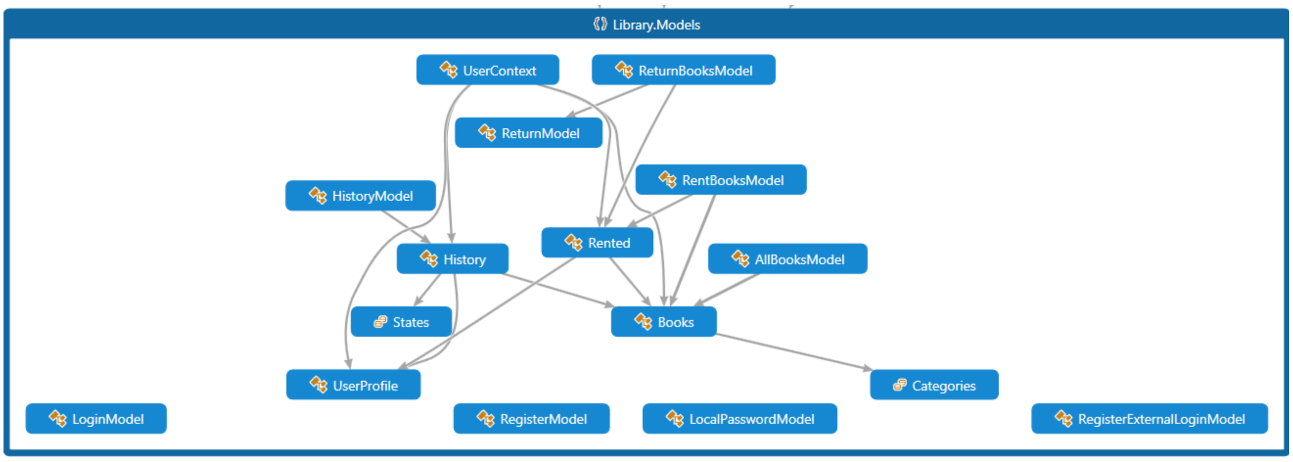
“`<add name="MyCon" connectionString="Data Source=MARCEL-PC;Initial Catalog=Lekram;Integrated Security=True" providerName="System.Data.SqlClient" />` “

When setting up our project and running it multiple times you should be aware of the following:
We use a Delete and Create on our Database, which means that every time you compile our project the database will be deleted and rebuild at the same Location. Unfortunately not everything in the Database will always be dropped (or it won't be dropped because there are still connections open). In this case you should drop the whole database manually and run the project again. Additionally we do not have seeding in place in this version of our Code. This means that you will need to manually create a new account and enter book data every time you run the application.

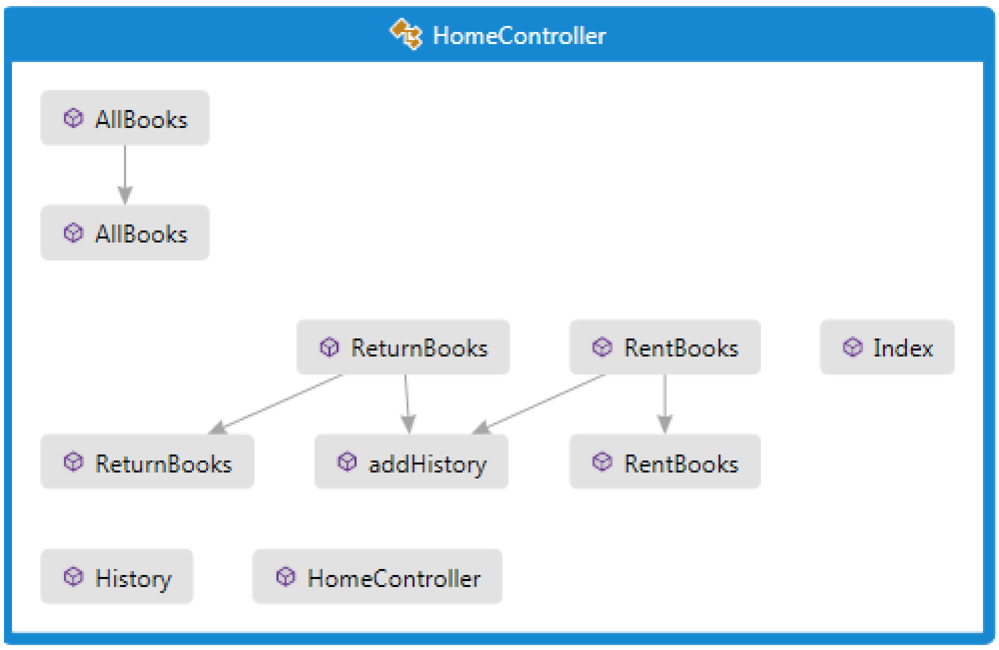
The structure of our Solution is very simple. All models and database definitions are contained in AccountModel.cs. Everything related to user control is in the AccountController while all library logic is in the HomeController. A similar seperation is in place for Views: All library related Views are in Views/Home while all User related Views are in Views/Account.

Information between Views and Controllers or Controllers and the Database is strictly passed through models. Each connection between a View and a Controller has its own model and each connection to the Database is handled by the DbContext.
We do not use any raw SQL or Javascript to pass information. We do use LINQ/Lambda function to query our database for certain results.

Connections in our models can be seen in the following Graph:



Connections between the Controllers in HomeController can be seen here (note duplicate methods are overloaded:


HomeController
AllBooks
AllBooks
ReturnBooks
RentBooks
Index
ReturnBooks
addHistory
RentBooks
History
HomeController