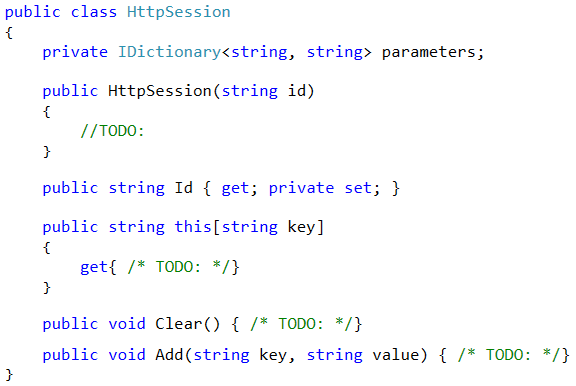
# Lab: Simple MVC Framework - Part II

Problems for exercises and homework for the [“C# Web Basics” course @ SoftUni](https://softuni.bg/courses/csharp-web-development-basics).

In this lab we wil further improve our HTTP Server to create sessions so it now will be much more easier to add features to our web application related with the current session of the user.

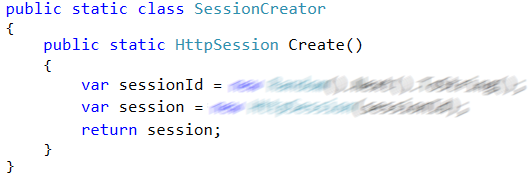
## HTTP Session

In our **HTTP Server** in the **Models folder** we need a HttpSession class. That class would have **Id** and **collection of key-value pairs** (which can be used to change the theme of the site, or language or some other user preferences). Also, the class should have methods for easily **adding key-value pairs** **and clear all the parameters** in the collection.



## Session Creator

To easily create sessions with unique Id we would create simple class SessionCreator in the Utilities folder. That class would have only 1 static method Create() that returns new HttpSession with random unique Id.

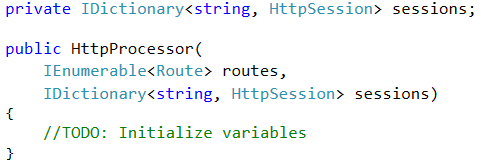


## HTTP Request

Our HttpRequest now should have property of type HttpSession. Add it to the class.

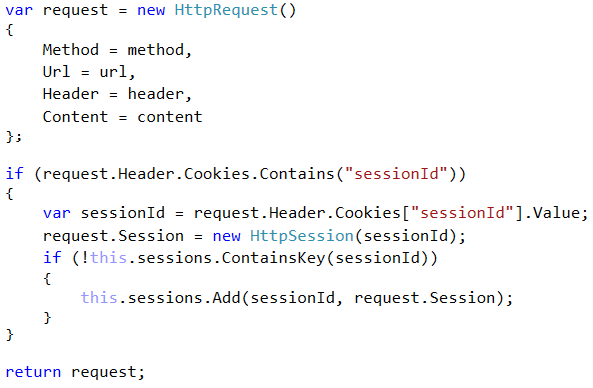
## HTTP Processor

Now our HTTP Processor must be now modified to work with the sessions. Sessions will be stored in a collection. We should receive that collection from the HttpServer. (but we will implement that in the next task)

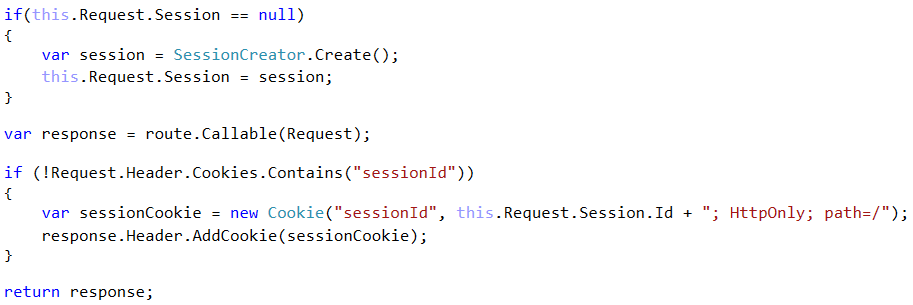


Every time the user opens some browser and send a request to our server we should check if there is a cookie with key “**sessionId**” if there is one we should create new session with the provided ID and add it to the request. And also, add that session to the collection of sessions.

[image]

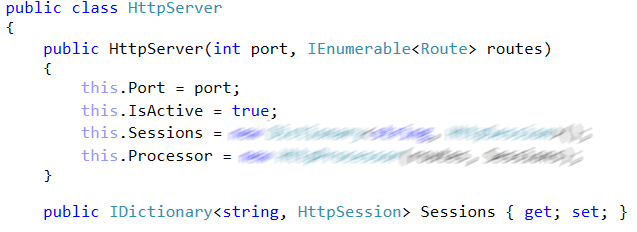


Just before we send the HttpResponse back to the client we should check if there is cookie in the request with key **sessionId**. If there is no such cookie that means that is the **first request to our server** from that user and we should **set a cookie with the session Id in the response.**



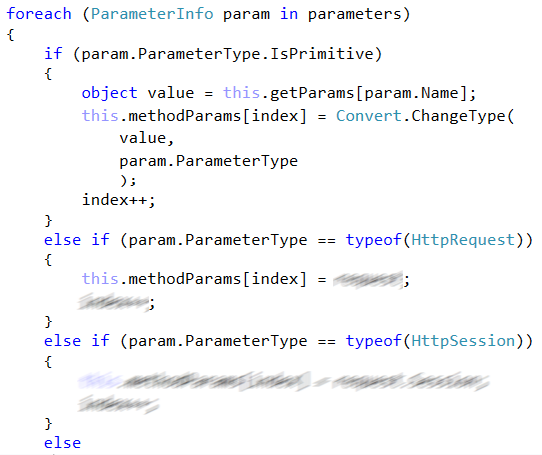
## HTTP Server

Final step in modifying our server will be adding collection of our sessions to the server and then pass it to the HttpProcessor

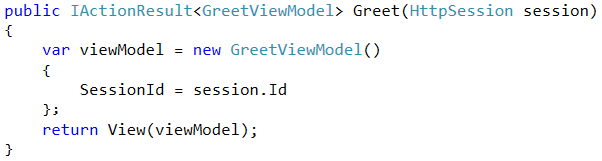


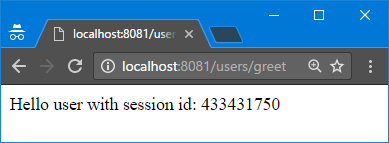
## MVC Binding Parameters

Tо be able to obtain the information in the session we need to modify the code that bind the parameters. If the parameter is of type HttpSession or HttpRequest we should bind the request or the session of the request to the method parameters.



Now we could use the session in our program logic (for example we could use it to log in user in the system and so on.)





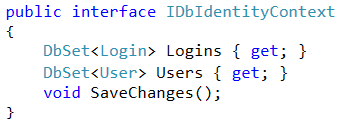
## Application Security - Introduction

Tо improve the security of our application we can implement user **registration, login and restriction to some pages** to be only reachable when the user is logged in.

We need **2 models** that would be saved in the **database**:

* **User** – username, password
* **Login** – session Id, User, IsActive

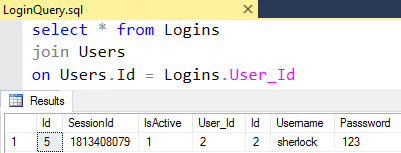
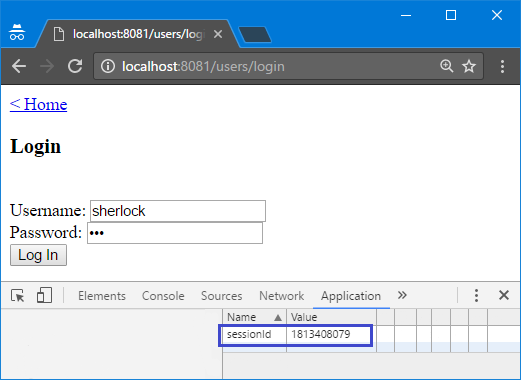
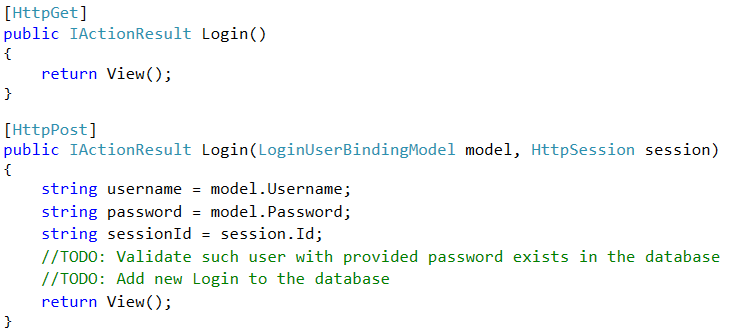
Create those models in the Models folder. Also, we need to create 1 more interface. In the MVC/Interfaces create new interface called IDbIdentityContext. That interface would define 2 getters and 1 method.



Assuming you are using the **NotesApp** from the previous Lab Exercise make the NotesAppContext to implement IDbIdentityContext.

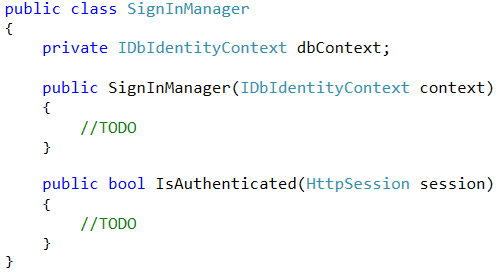
## Application Security - Login

We already implemented the process of registration new users at the **/users/register** page. Now we should implement the feature of users of our application to log in. Create page **/users/login** similar to the **/users/register** but this time the POST form will **log the user**. If the provided **username** and **password** exists in the database, his **session is saved to the database**.



## Application Security – Sign in Manager

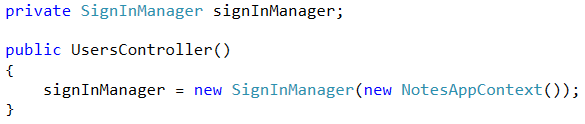
We might want to **restrict some pages to be accessible only to logged in users**. In that case, we would need a method that check whether the user is logged in and only in that case show him the page. In the MVC folder add new folder Security and inside of it add new class SignInManager.



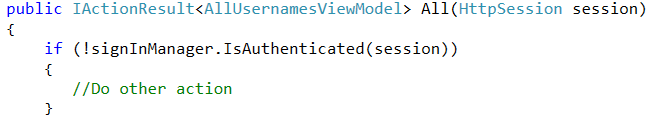
The IsAuthenticated() method should check in the database whether **there is session with provided session Id and if that session is active.**

## Application Security – Restrict Access

For example, we want to restrict the access to the users/all page only to registered users. We could use the SignInManager class in our UsersController



And at the beginning at the method we could check if user is logged in and only then show him the page. Otherwise we could do some other action (for example redirect him to the login page).

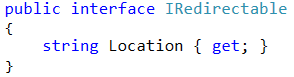


## Application Security – Redirect

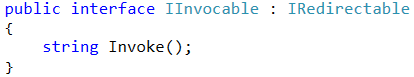
To redirect the user to other page we need to do some modifications in the Interfaces, ControllerRouter and the main Controller

#### IRedirectable

Create new interface IRedirectable that would contain only one string property Location.

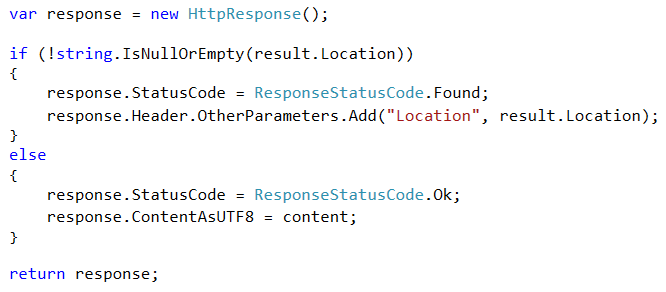


Make the IInvocable interface inherits IRedirectable. Then add Location property to the ActionResult classes (generic and non-generic version) and make it possible to be set from the constructor.



#### ControllerRouter

Here we should check whether the invoked method returned some location in it. If it is **not null or empty**, we should change the response to **redirect to the provided location**. Otherwise we should **return the standard response**.

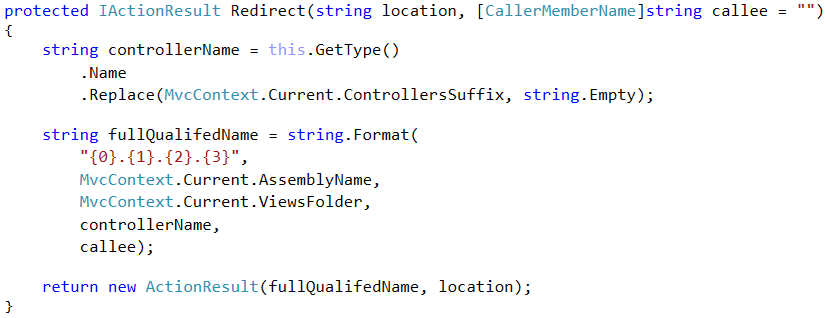


#### Controller

In the base Controller class, we would add 2 additional methods:

* IActionResult Redirect(location, callee)
* IActionResult<T> Redirect(model, location, callee)

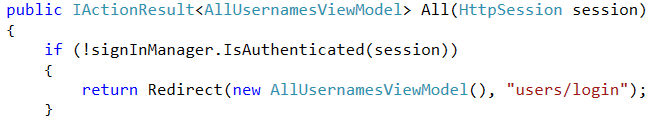
They are very similar to the View(callee) and View(model, callee) method but they also pass the provided location to the ActionResult classes. Here is the implementation of the first method.



Implement the other method by yourself.

## Application Security – Redirect Example

Assume we want to access **/**users/all page only if we are logged in and if not we should redirect to user to the login page. With the classes and methods, we already created it would be pretty simple.



## NotesApp – Login Redirect

After successful log in **redirect the user** to the **/home/index** page. Otherwise stay on the login page.

## NotesApp – Logout

On the home page **add Log out button** that logs out the user from the system. After logging out **redirect to the /home/index** page. Implement the log out process in the SignInManager class.