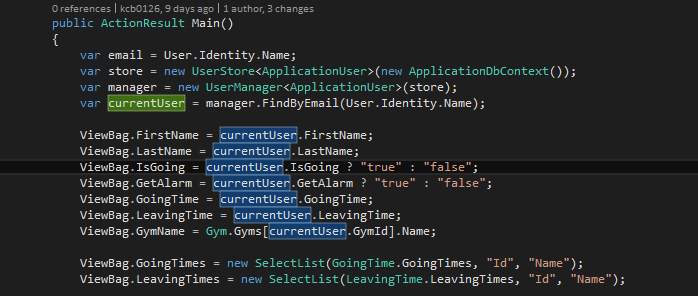
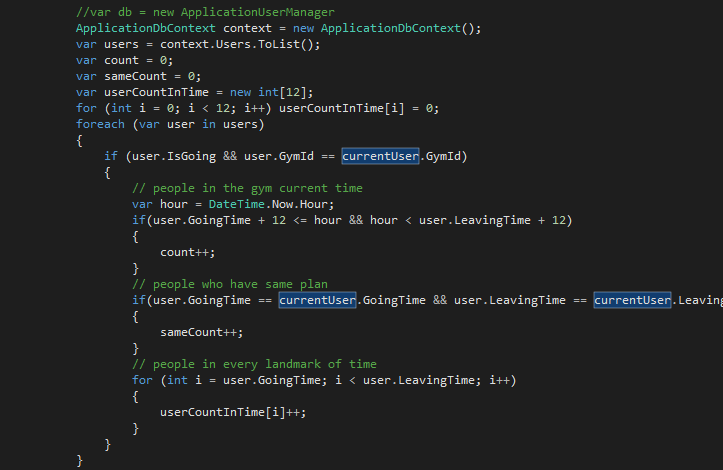
Gym Plan page:

To show and process data of page of gym plan, I used 2 actions: GET and POST method of Main.

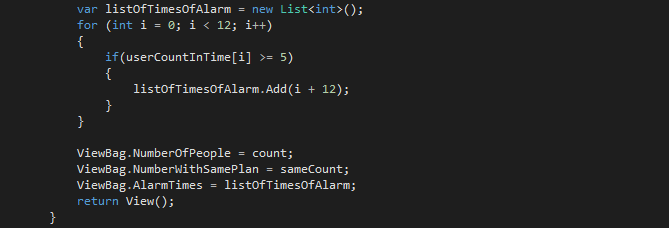
Main(GET): This action returns view of gym plan page.



Above code extracts information about gym plan from the user’s model and send them to the ViewBag to show in view page.

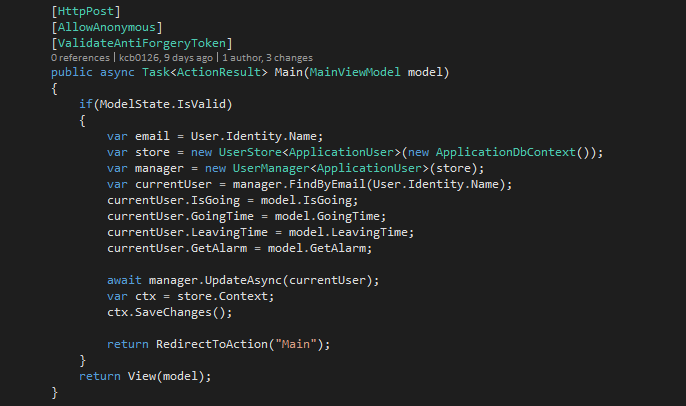


Here, list of all users are obtained from database and they are used to calculate number of people in the gym current time.



This code sends data related alarm so that view page can raise message if needed.

Main (POST): This action processes data submitted by user.



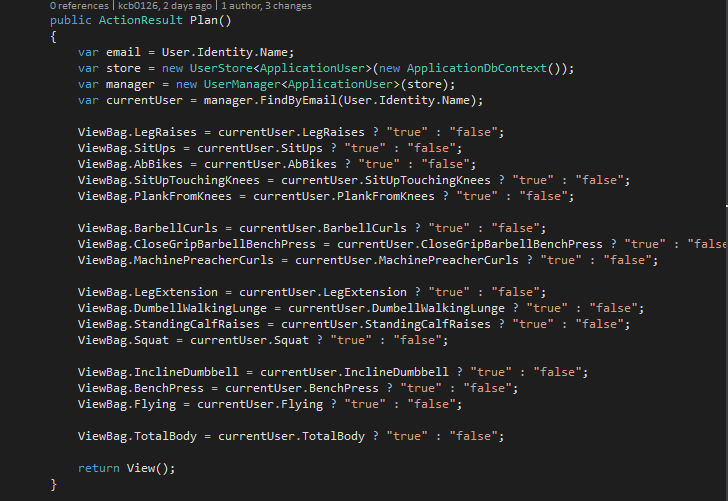
Here, using view model or MainViewModel, it defines data structure as a model. So data can be obtained from this model and saved to database.

After that, it redirects to above GET action to show gym plan page again.

Workout page:

To show workout page and process data, there are also GET and POST method for this action.

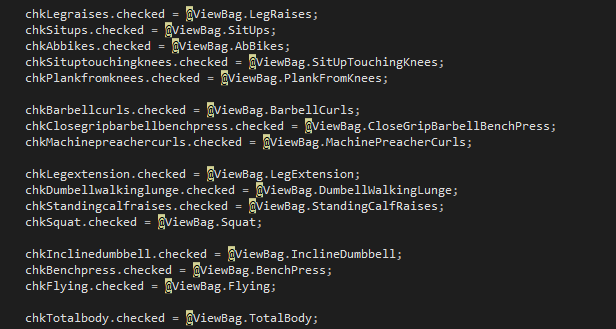
Plan (GET): It returns view of workout page.



All data for workout is Boolean type. And we should use JavaScript to set the checkboxes. To do this, Boolean variables are converted into string(“true”/”false”) by ternary operator. And they are sent to ViewBag to be used by JavaScript in view page.

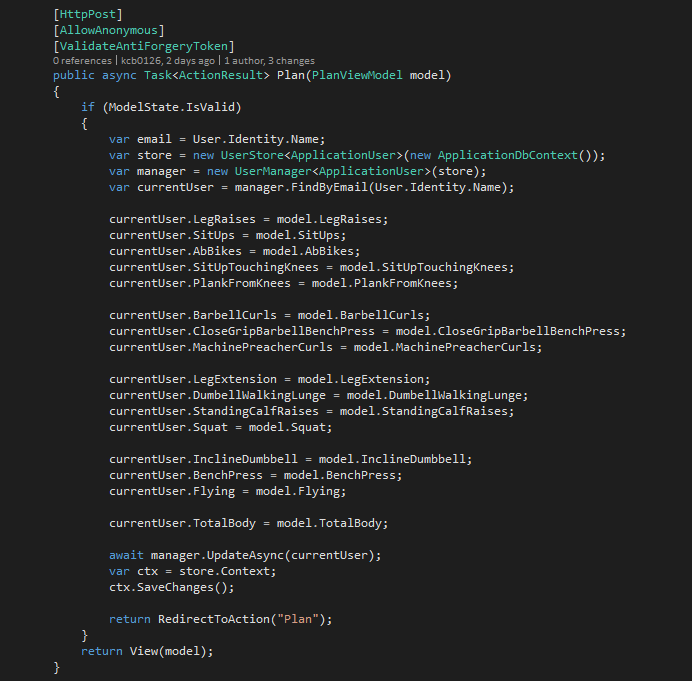
Part of JavaScript code in view page.





Here, JavaScript code uses above string(C#) variables as Boolean(JavaScript) ones. This is because a checkbox cannot be checked by static Html code.

Plan (POST): Save data submitted from the client and show workout page again.



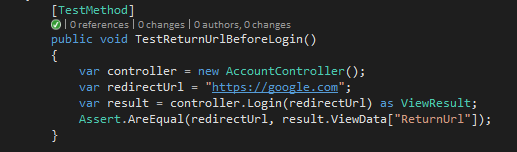
Here, data from client is stored in user’s profile, i.e. object of ApplicationUser for current user. And the information is saved in database. After that, it redirects to GET method of Plan action to show workout page again.

Unit Testing.

Unit test for AccountController

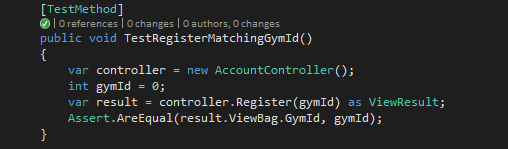
1. Test for return url before login.

If user not logged in calls other action in which authorization is needed, login page is shown first. After logged in, it redirects the first action. This test is to make sure the return url works fine.



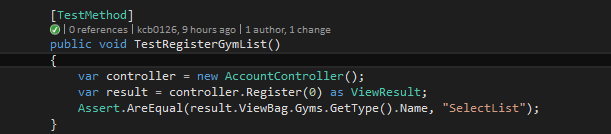
1. Test for gym id in register action.

When user tries to register, he or she already selected gym that represents gym id. So register action is called with gym id and this id is sent to the view through ViewBag. In this test, we can make sure the gym id is sent correctly.



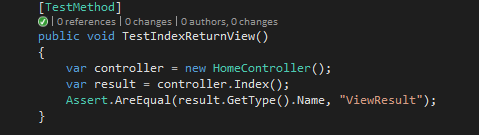
1. Test to show register action delivers list of names of gym.

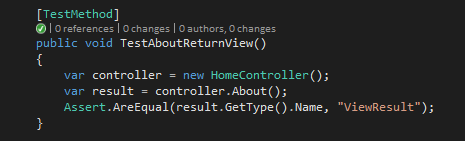
In the register action, names of gym are fetched from static data and sent to the view as SelectList. Comparing the type of above data with “SelectList” can make sure data is sent as SelectList variable.

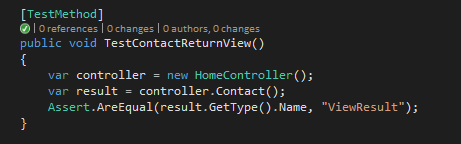


1. Test of return type of GET actions.

In this project, all GET actions return ViewResult, not a redirected action or Task for awaiting action result. Reading type of result of these actions can show results are in type of ViewResult.

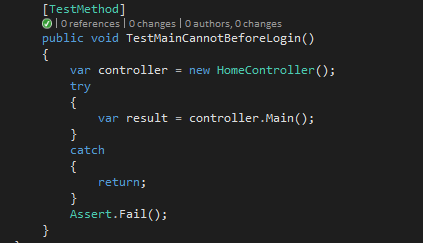






1. Test for main action before login.

Main action or gym plan action is available after login. If you try to create HomeController and call Main action of this, an exception is thrown. Following test if passed if exception is thrown while Main action is being called.



Result of above test cases:

