# *Web Programming VI (420-H60-HR)*

# *Lab 01 – MVC Review*

Date due: **Friday August 23**

**Learning Objectives**

Upon successful completion of this lab exercise, the student will have:

* Re-familiarized themselves with VS, MVC, C#
* Create Models using proper formatting
* Create views to read in models and add to them using HTML helpers and tag helpers

To Do

**Part A – MVC Review**

1. **(1 Mark)** Create a solution/project called *initials*\_H60L01 using the ASP.NET Core Web App (Model-View-Controller, C#) template. Make sure the Target Framework to .NET 8. and “Configure for HTTPS” is not checked.
2. **Note:** you must run your applications on IIS. The college does not grant you permissions to build your own executables.
3. **(2 Marks)** Update the default view for the Home controller and add the following paragraph in a <main> tag (or something like it):  
   “A list of people who exist somewhere in my imagination. They are all my friends.” Add more text/images that reflects your personality/interests.
4. **(2 Marks)** Update the \_Layout view. Add a banner with your name and some image floated right.
5. **(2 Marks)** Add the CSS as necessary to format the page to look good (sans serif font, margins, centering divs and maybe include a background image or gradient…do something creative).
6. **(5 Marks – 2 steps)** In the Models folder add a Person class with the following attributes (FOLLOW THE .NET EF NAMING STANDARD):  
   PersonId (long)  
   FirstName (string)  
   LastName (string)  
   Country (string)  
   Validated (bool)  
   Email (string)
7. Add four (4) constructors: one with no parameters, one with all the parameters, one with just the Id and one with all the parameters except the Id attribute.
8. **(3 Marks – 2 steps)** At the top of the class add a private static long variable called CurrentPersonNum and set the value to 1. In the constructor where everything but the Id is passed, set the Id field to be the value of the current number and increment it. That is, PersonId = CurrentPersonNum++; Note: Make sure
9. Also at the top of Person class add a private static List of Persons called PersonList as:  
   private static List<Person> PersonList = new List<Person>();

**NOTE: THESE STATIC VARIABLES WOULD NOT BE DONE AS THERE WOULD BE A DATABASE TO HOLD THE INFORMATION.**

There are a couple of hints at the bottom of the lab that you might want to check out if you are confused in this next part.

1. **(3 Marks)** Add a method to the Person class called GetPersons which simply returns the PersonList (method is of type List<Person>).
2. **(3 Marks)** Add a method to the Person class called AddPerson which adds the current object to the static list: PersonList.Add(this); and returns true (method is of type bool).

**ONCE AGAIN: THESE METHODS WOULD NORMALLY GO TO THE DATABASE AND RETRIEVE THE RECORDS, BUT WE DO NOT HAVE ONE AT THIS POINT.**

1. **(3 Marks)** Create a Person controller. In the Index method declare a list of Person objects and call the GetPersons method to get a list of the persons:  
   List<Person> PersonList = Person.GetPersons();

Return the list to the View (Return View(PersonList))

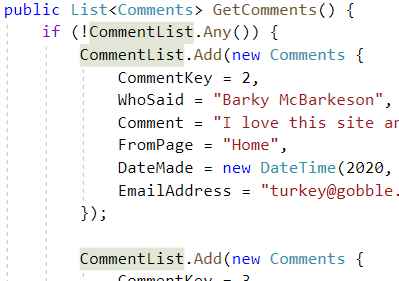
1. **(5 Marks)** Create the default Person view (Index) and have it (in a main div) display a header level 2 The Persons and the display the list of persons passed from the controller (you will need to check if the list is empty before displaying). Make sure you have a title, context, etc.
2. **(3 Marks)** In the Person Controller add a new route method called Add with the HttpGet command which returns the default View.
3. **(8 Marks)** Add a list, displayed horizontally, with the following items: Show Persons, Add Person.. Create the Add View using either HTML helpers or ASP Tag Helpers (you choose) that allows all the person information to be entered except the Id. Make sure you include a title, information, etc
4. **(4 Marks)** Back in the Person Controller, copy the Add method, change the command to HttpPost and add a parameter of type Person newPerson. Call newPerson.AddPerson to add the person entered to your list (newPerson.AddPerson()). You are now going to use a new call to return to the Index page of the Person Controller. To do this use: return RedirectToAction("Index", "Person");
5. **(2 Marks)** Update the \_Layout view to use ActionLinks to call the Person Index action or Person Add action as appropriate.
6. Test to make sure you can add person objects and that they are displayed in the list.
7. **(3 Marks – 2 steps)** At the top of the Person class add an enum called ValidCountry that contains the following (in this order): Canada, Albania, Denmark, France, Hungary, Spain, Turkey, Zaire.
8. Change the country field in the class (and the constructors) to be of enum type ValidCountry.
9. **(10 marks)** Use data annotations to make the fields in the form as specified:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attribute | Prompt  (Display Name) | Field type  on Form | Data Type  in Model | Validation |
| FirstName | First Name | Textbox | Text | Minimum length of 1,  Maximum length of 50 |
| LastName | Last Name | Textbox | Text | Minimum length of 2, maximum length of 50 |
| Country | Country | Drop down list with "Select Country" as first prompt |  | Not needed |
| Validated | Validated? | Checkbox | Boolean | Not needed |
| Email | Email | Email | Text | Email type |

1. **(3 marks)** Add appropriate tags to the view to display a summary of the errors at the top of the form and to display each error with the field.

1. **(5 marks)** Add CSS to display the errors such that the summary is in red italicized text, the input field is shaded in a light yellow (think opacity) and the error message is in red text with a light-yellow background on the line immediately following the field in error. You will need to add the CSS classes field-validation-error, input-validation-error and validation-summary-errors along with any other CSS you require.
2. **(3 marks)** Update the Controller such that, if the ModelState is not valid it returns to the view and if the model state is valid it adds the person to the list (AddPerson method) and calls the index method for person to display the list.
3. **(2 marks)** Add a message that says "Person Added Successfully" and display it in the Index page. If you want to make it fade after time you can, but that is not required.

HINT: You might want to add something like the following (but for Person) to your Get method that returns the list in the class. This negates having to deal with null lists.



HINT 2: Remember that you do not have to specify the Id for the Person as the constructor will take care of that for you automatically.

**To submit**

When you have completed the lab exercise show me your work and then copy the zip file to Moodle.