**Capabilities Covered:**

* Implement MVC framework using .Net MVC 5

**Assignments on Basics of MVC 5**

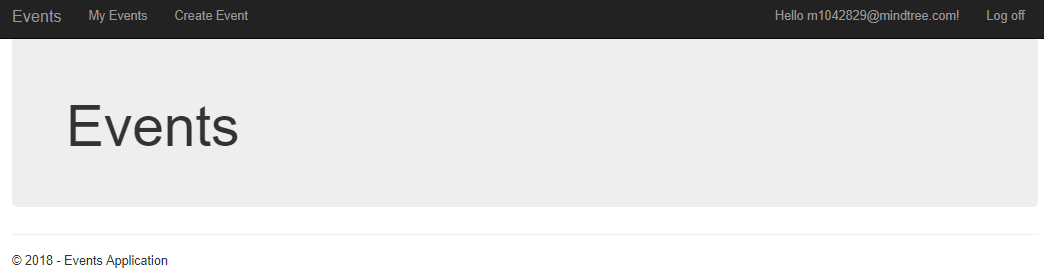
Develop an event management system where users should be able to view their own events, to create new events, edit their own events and delete their own events.

Assignment

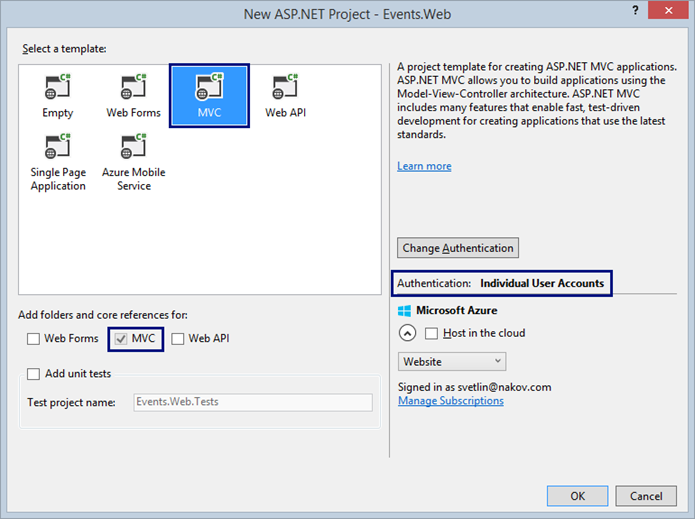
**Assignment 1: Register, Login and Layout change**

**pe**

**Problem description: Change the layout of the home page for the logged in user like the below screenshot:**



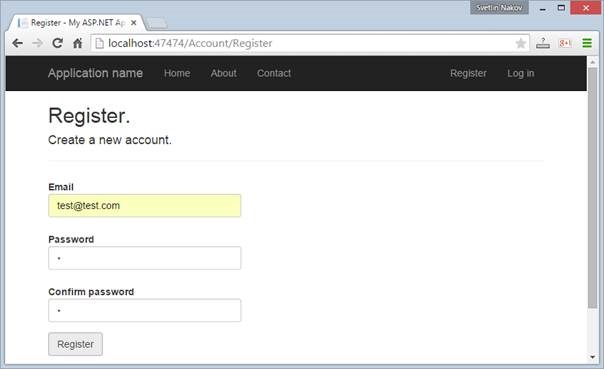
Step 1- Create mvc web application with authentication “Individual User Accounts”. Follow the screenshot:



**Note – After creating, check the Model, Controller and View folder, there some files are automatically incorporated. Debug and explore how these files are working.**

Step 2- After creating the application do the necessary changes in \_Layout.cshtml.

Step 3- Run the application, first register yourself with necessary details:



Step 4- Login with the correct credential.

Step 5- for logged in user, screen should be displayed like screenshot given above in the problem statement.

**Note: Keyfocus area is \_Layout.cshtml, model, view, controller folders.**

**Assignment 2: CRUD operation**

**pe**

In this section, focus will be on **CRUD** operation and some main MVC features like data passing through viewbag/viewdata/tempdata, Html.ActionLink,dataannotations.

* Create New Event:
* Event details will be below:

|  |  |  |  |
| --- | --- | --- | --- |
| Field Name | DataType | Validation | ErrorMessage |
| Title | String | 1. Maxlength – 200 ,minimum length -1 2. Displayname should ne   “EventTitle” | Title should be between 0 and 200 characters |
| StartDateTime | Datetime(It should be a datepickerfield) | Display name should be “Date and Time” |  |
| Duration | TimeSpan |  |  |
| Description | String |  |  |
| Location | String | Maxlength should be 200 |  |
| IsPublic | Bool | Display name should be “Is Public?” |  |

All the fields are required fields.

**Note: For all the validations, use DataAnnotation feature. These are called model validations.**

* create the “New Event“ form. The easiest way to start is by using the Razor view generator in Visual Studio. Create a folder “\Views\Events“ and under Events folder create a view named “Create” for event creation.

**In the Create form,**

1. Title should be ‘Create Event’
2. Remove “Back to List” link.
3. Add [Cancel] link to “My Events”, just after the [Create] button

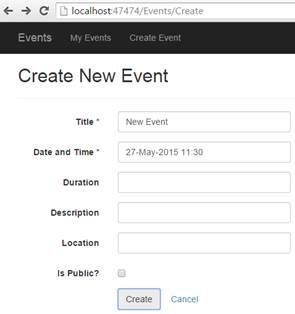
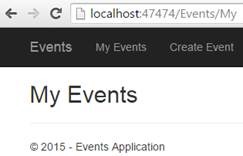
**Note: Analyse the view that has been created for event creation, analyse what are the lines of code has been auto generated.**

* After creating the ‘Create’ view, create Event controller, there need to incorporate create, edit, delete logic.

1. For Creating events, use [HttpPost] attribute.
2. For editing events, use [HttpPut] attribute.
3. To delete the event use [HttpDelete] attribute.
4. To retrieve the added events data, use [HttpGet] attribute.

* After creating events, page should be redirected to other page where user can see the added events details and can edit/delete the events.

1. For passing the data, Viewbag/viewdata/tempdata must be used.
2. To display, edit, delete create corresponding actions and view named “My Events”.

🡪 

* For CRUD operation, use ado.net to connect to SQL Server database.

**Assignment 3: Routing**

**pe**

**Convention Based Routing:**

Go to App\_Start 🡪RouteConfig.cs

|  |
| --- |
|  |

* Debug and analyse functionality of RegisterRoutes method.
* Change controller name to “Events” from “Home” in “defaults” property of maproute method and analyse the result. Change other properties and run in the browser and check result.

**Attribute based Routing:**

* Enable attribute based routing need to add the following code in RouteConfig.cs.

tes.MapMvcAttributeRoutes(); //Enables Attribute Based Routing

* Then add attribute based routing in “Event” controller class for action methods like below example code snippets:

|  |
| --- |
| 1. [Route("products/{id?}")] 2. **public** ActionResult Details(**string** id) 3. { 4. **if** (**string**.IsNullOrEmpty(id)) 5. { 6. **return** View("List", GetProductList()); 7. } 9. **return** View("Details", GetProductDetails()); 10. } |

**In convention based routing:**

|  |
| --- |
| 1. **public** **static** **void** RegisterRoutes(RouteCollection routes) 2. { 3. routes.IgnoreRoute("{resource}.axd/{\*pathInfo}"); 5. tes.MapMvcAttributeRoutes(); //Enables Attribute Based Routing 7. routes.MapRoute( 8. name: "Default", 9. url: "{controller}/{action}/{id}", 10. defaults: **new** { controller = "Product", action = "List", id = UrlParameter.Optional } 11. );  14. } |

As shown in the method above, the Route is defined on a Details action method that lets users access the product details page either by of these paths:

/Product/Details/Id or /products/id

Refer the example code and implement both routing in your code.

**KeyFocus area of this assignment:**

* How “Separation of concern” is implemented in MVC.
* Need to be aware of functionality of each and every folder of a MVC solution especially

focus on App\_Start folder as it contains all the configuration files of the application.

* Functionality of \_Layout.cshtml
* Difference on implementation of Viewdata, viewbag, tempdata
* Session management
* Scaffolding
* Convention based Routing, attribute based routing
* Razor syntax
* Different return types of controller action methods like ‘ActionResult’, ‘JsonResult’.
* Partial view

**Future Scope:**

* Apply basic authentication and token based authentication in the application.
* Three layered architecture
* Instead of ADO.Net, entity framework will be used to interact with Database.
* Dependency injection will be implemented.
* Whole CRUD operation will be managed from MVC web API, only need to consume that Web API from MVC application.

---- Thanks----