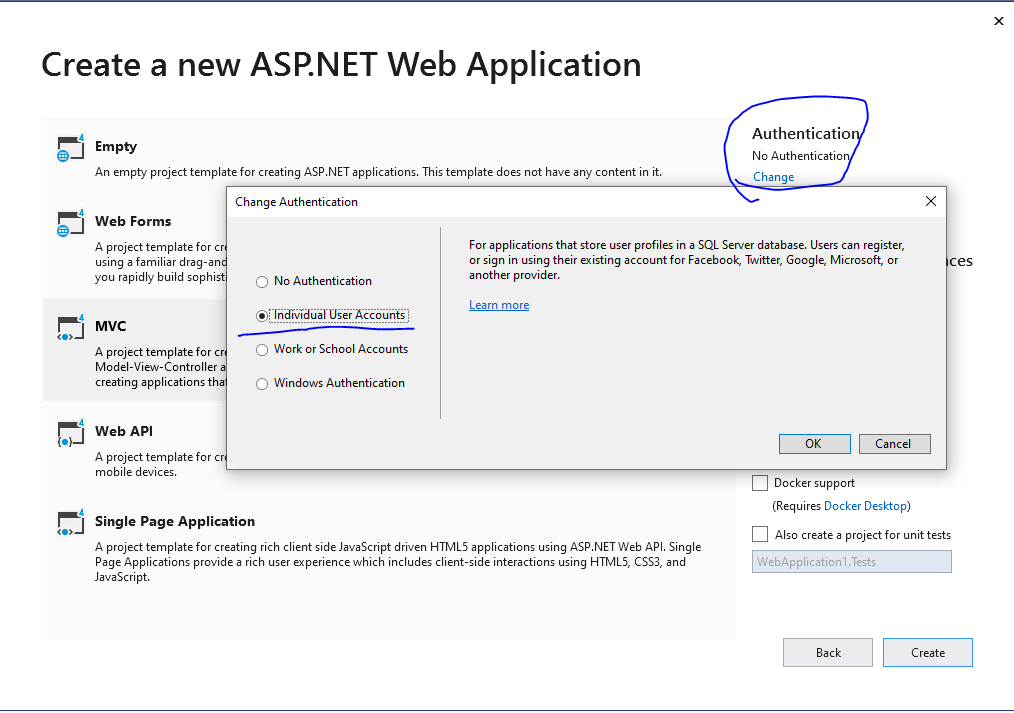
Create a new ASP.Net Web Application (.net Framework) Solution (Make sure its C# and not Visual Basic).

On the next screen when picking which packages to install, make sure Authentication is enabled by clicking change under Authentication and selecting Individual User Accounts. Uncheck host with Azure if it is checked.

  
  
This will generate a new EF context, and some preestablished user classes and controllers.

Take some time to look over some of the generated classes. Particularly:

/App\_start/Startup.Auth.cs

/models/Identitymodel.cs (This is where your ApplicationDbContext will be)

/models/AccountViewModels.cs (this contains all the viewmodels used in the views/controllers for accounts)

/Controllers/AccountController.cs

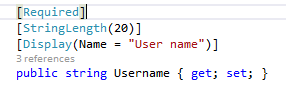
/Views/Account

Modify the existing viewmodels and controller actions to log in via a Username, instead of Email address. (ViewModels are special kind of Model that don’t represent an entity in the database, but rather a specific subset of properties, often from multiple Models, passed to views)

To do so, add a property Username to all relevant ViewModels.

(Hint, you can see which ViewModel each view uses at the top of each view, ie RegisterView.cshtml uses RegisterViewModel)

You may notice several Data Annotation attributes. We will learn about these in later lectures, for now you can put these ones. It will enforce view forms to Require the username field, a maximum length of 20, and use ‘User name’ for labels.



You will also have to update the Login and Register actions to reflect authenticating with the Username.

(ie, pass model.Username instead of model.Email

var result = await SignInManager.PasswordSignInAsync(model.Username, model.Password, model.RememberMe, shouldLockout: false);

)

And changing the create user in the Register Action to the new model.Username property created in the RegisterViewModel

var user = new ApplicationUser { UserName = model.Username, Email = model.Email };

Don’t forget the Views!

Update Register.cshtml and Login.cshtml to include a field for Username.

(Look at how other string fields, ie Email, are created. The mark up will be at the bottom of the exercise sheet if you are stuck)

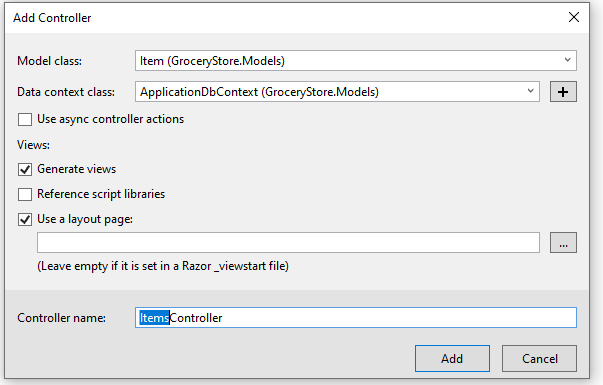
Create a model called ‘Item.cs’ and add the following 3 properties:

Public int ID (required for Entity Framework primary key by default)

Public double cost

Public string name  
  
Right click controllers, Add > New Scaffolded Item > MVC 5 Controllers with Views, using Entity Framework. Don’t forget to build after creating the Item.cs class!

Fill out the scaffolding wizard. You should have ApplicationDBContext as the data context class.



Using Attributes, require all the Actions of the ItemsController regarding CRUD (Create, replace, update, delete) to require the user to be logged in, and have “Admin” Role, while the normal index is open to unauthenticated users. This can be validating by navigating directly to Items/Details/1 without logging in.

(Hint: Attributes put on the controller class are applied to each action, and can be override at the action level. Refer to lecture slides and source code if you need a reminder on authorization attributes)

Create a new folder, Data, and a new class inside the folder DataInitializer.cs. Import the follow namespaces and begin creating an initializer. Create a testing user, an Admin role, and a sample Item (Don’t forget to add the sample Item to the context.Items DbSet)

using System.Data.Entity;

using [YourSolutionName].Models;

using Microsoft.AspNet.Identity;

using Microsoft.AspNet.Identity.EntityFramework;

Inherit the class DropCreateDatabaseIfModelChanges<ApplicationDbContext> or DropCreateDatabaseAlways<ApplicationDbContext>.

**Note: The latter class will always delete your database on start up, the former will only drop and recreate it if models have changed**

Below is the code for creating a new user and role “Admin”



Don’t forget to add the namespace System.Data.Entity and the Database.SetInitializer call in your Global.asax file! If you are having issues, refer to the Source Code from the lecture for a copy of an Initializer.

Modify your existing View/Items/Index.cshtml to only show the ACtionLinks if the user is authenticated and in the Admin role. (hint, you can use User.IsInRole to query if the current user has a role within a view)

Create a new Layout under Views/Shared, called \_AdminLayout.cshtml. Change some verbiage on it so we can recognize when it is loaded opposed to the default Layout.

Modify your existing \_ViewStart.cshtml file (located at the root of the Views folder) to set the Layout to the newly created Layout only if the user is an admin

Login.cshtml username markup

<div class="form-group">

@Html.LabelFor(m => m.Username, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.Username, new { @class = "form-control" })

@Html.ValidationMessageFor(m => m.Username, "", new { @class = "text-danger" })

</div>

</div>

Register.cshtml username markup (assuming property is called Username)

<div class="form-group">

@Html.LabelFor(m => m.Username, new { @class = "col-md-2 control-label" })

<div class="col-md-10">

@Html.TextBoxFor(m => m.Username, new { @class = "form-control" })

</div>

</div>