How to Use the Identity Template

# Install Identity into your existing project. You need to add the following NuGet packages to your project:

## Microsoft.AspNet.Identity.OWIN.EntityFramework

## Microsoft.OWIN.Host.SystemWeb

## Microsoft.AspNet.Web.Optimization

# Update your connection string in your web.config file

You used to have:

<connectionStrings>

<add name="MyDBConnection"

connectionString="Server=tcp:fa16inclassdemos.database.windows.net,1433;Initial Catalog=fa16IdentityTemplate;Persist Security Info=False;User ID=Admin333K@fa16inclassdemos;Password=Password123;MultipleActiveResultSets=**False**;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;" providerName="System.Data.SqlClient" />

</connectionStrings>

But you need:

<connectionStrings>

<add name="MyDBConnection"

connectionString="Server=tcp:fa16inclassdemos.database.windows.net,1433;Initial Catalog=fa16IdentityTemplate;Persist Security Info=False;User ID=Admin333K@fa16inclassdemos;Password=Password123;MultipleActiveResultSets=**True**;Encrypt=True;TrustServerCertificate=False;Connection Timeout=30;" providerName="System.Data.SqlClient" />

</connectionStrings>

# Add a bunch of stuff from the template into your project

In the App\_Start Folder:

Add IdentityConfig.cs

Add Startup.Auth.cs

In the Controllers Folder:

AccountController.cs

RoleAdminController.cs

If you don’t already have a home controller, you can add the one from the template

In the Models Folder:

IdentityModels.cs

IdentityRoles.cs

RoleViewModels.cs

UserViewModels.cs

In the Views Folder:

In the Account Folder:

ChangePassword.cshtml

Index.cshtml

Login.cshtml

Register.cshtml

In the RoleAdmin Folder:

Create.cshtml

Edit.cshtml

Index.cshtml

In the Shared Folder:

\_LoginPartial.cshtml

Error.cshtml

If you don’t already have a \_Layout.cshtml, you can add the one from the template. If you have one, be sure to add the code @Html.Partial("\_LoginPartial") to the navbar.:

<div class="navbar-collapse collapse">

<ul class="nav navbar-nav">

<li>@Html.ActionLink("Home", "Index", "Home")</li>

</ul>

**@Html.Partial("\_LoginPartial")**

</div>

# View the Task List to See What to Change

## Open the Task List in Visual Studio by Selecting View 🡪 Task List. This will open a window that will show you the places in the code that need to be edited

## Add the “NOTE” flag to the task list to be able to see the additional notes in the project. Here are the steps:

1. Go to Tools 🡪Options…
2. Select Environment from the listbox on the left hand side.
3. If necessary, click on the small triangle just to the left of “Environment” to expand the list
4. Select “Task List” from the expanded list
5. Type “NOTE” (without the quotes) into the “Name” box
6. Press “Add”
7. Press “OK” to close the Options menu

## The task list should now show both Notes and TODOs so you can see what needs to be changed in the files you added.

# Make all of the changes in the template

Like so many things in MVC, everything works together, so it almost doesn’t matter where you start. Make sure you have addressed every TODO in the files you imported from the template. My suggestion is to remove the TODO comments as you complete them. The changes fall into a few broad categories:

## **Namespaces/Using Statements**:

The files were created in a project called IdentityTemplate. Several of the files you added will need to be changed to make the namespace of the file match the namespace of your project.

## **AppUser Model**:

Changes related to the fields you need on the user class. Identity uses a built-in User class that already has several fields that all applications need (UserID, Email, Phone Number). If you need additional information on your user class (First Name, Orders, etc.), you will need to add these fields to your AppUser class that inherits from the IdentityUser class. You will also need to change the code that references the AppUser class.

### Models 🡪 IdentityModels.cs – This is the file where the AppUser class lives. Add the fields you need here first. There are examples on lines 15 & 16 of the IdentityModels.cs file.

### Models 🡪 UserViewModels.cs – This file contains all the classes for the various view models required to get Identity to manage user accounts. You will need to add code to the RegisterViewModel to include the fields on your AppUser class that the user can change directly (e.g. First name)

### Views 🡪 Account 🡪 Register.cshtml – You will need to add code to the view to capture the additional fields that you added to AppUser.

### Controllers 🡪 AccountController.cs – You will need to modify the code on the Register action method (line 123) to capture the additional field methods.

IMPORTANT NOTE ABOUT REGISTERING NEW USERS: The template is set up so that all users can register themselves. Obviously, this isn’t the case for everyone. You will probably need to create a duplicate, modified version of the register files mentioned above for cases where a user can’t create their own profile (e.g. only managers can create new employees). These variations might be called RegisterCustomer, RegisterEmployee, etc.

## **DbContext**:

Identity has its own DbContext class that you MUST use for Identity to work.

### You need to DELETE any existing DbContext files that you’ve already created

### Change references from your DbContext to the new Identity DbContext.

### Add DbSets for your other model classes

## **Roles**:

Identity uses role-based authentication to allow/prevent users access to various pages. The Identity template contains both the models and the views/controllers for managing roles. You shouldn’t have to change anything in the code to enable roles. You WILL need to use the code listed on lines 129/131 of Controllers 🡪 AccountController.cs to add people to roles when you create them.

## **Configuring Identity**:

You have some choices in how you configure Identity – you can decide password requirements, assign a username, etc. These configuration choices live in the following files:

### App\_Start 🡪 Startup.Auth.cs

### App\_Start 🡪IdentityConfig.cs

# Add Users to Roles

Now, you should add your users to roles. Navigate to the RoleAdmin/Index page to add roles using the UI. By default, the RoleAdmin/Index page is available to all users. It needs to be this way for the initial setup, but you should change it as soon as you have authorized admins/managers in your system. To set up roles, complete the following steps:

## Make sure you have completed all of the steps above this one in the document. If the rest of the template isn’t working, this isn’t going to function either ☹

## Run the project and navigate to the RoleAdmin/Index page.

## Add a role for each of the required roles in your project (Customer, Employee, Manager, etc.)

## Create (Register) a user that will be in the role responsible for managing other users (this is probably “Admin” or “Manager”

## Add this new user to the Admin/Manager role by going to RoleAdmin/Index and selecting edit for the appropriate role.

## Stop your project and add in the tags to the controllers and/or action methods within the controllers.

## [AllowAnonymous] – anybody can access this controller/action, whether or not they are logged in

[Authorize] – any authenticated user can access this page

[Authorize(Roles = "Manager")] – only managers can access this page. Everyone else will be re-directed to login or given an error message