Authentication/Authorization

By: Mosh Hamedani

ASP.NET Identity Classes

- **API**: UserManager, RoleManager, SignInManager
- **Domain**: IdentityUser, IdentityRole
- **Persistence**: UserStore, RoleStore

Restricting Access

Declaratively

[Authorize]: apply to an action, a controller or globally (in FilterConfig).

[Authorize(**Roles = "CanManageMovies"**)]

Programatically

In an action:

```
if (User.Identity.IsAuthenticated) { … }
```

```
if (User.IsInRole("CanManageMovies") { … }
```

Seeding Users and Roles

Populate your database with the default user(s) and role(s).

Create an empty migration.

Script the data in existing users and roles tables and add them to the migration.

Remove the records from your database.

Run the migration.

Assigning a User to a Role

```
var roleStore = new RoleStore<IdentityRole>(new
                    ApplicationDbContext());

var roleManager = new RoleManager<IdentityRole>(roleStore);


await roleManager.CreateAsync(new
                    IdentityRole("CanManageMovies"));


await UserManager.AddToRoleAsync(user.Id, "CanManageMovies");
```

Adding Profile Data

Always start with the domain. Add the new properties to **ApplicationUser**.

Create a migration and update the database.

Modify the views: **Register.cshtml** and **ExternalLoginConfirmation.cshtml.**

When the registration form is posted, set the properties of the **ApplicationUser** object using view model properties. In **AccountController**, you need to modify two actions: **Register** and **ExternalLoginConfirmation**.

Enabling Social Logins

Enable SSL: select the project, F4, set **SSL Enabled** to true. Copy **SSL Url**, select the project, go to **Properties**, in the **Web** tab, set the **Startup URL**.

Apply **RequireSsl** filter globally (in FilterConfig).

Register your app with an external authentication provider to get a key/secret. In **App_Start/Startup.Auth.cs**, remove the comment for the corresponding providers and add your key/secret.