Walkthrough 16 - Identity Customization and Role-Based Authorization

Setup

This lab will explore further ASP.NET Core Identity by customizing the User Identity and using Role-based authorization.

1. Start SQL Server.
2. Open SecureSite from the end of the previous lab.

Create a User

1. If the database already exists and you have created a user, skip this section.
2. Otherwise, open the Package Manager Console (PMC) and issue the following command.
3. **Update-Database**
4. Run the site.
5. Register a new user.

ApplicationUser.cs

1. Add a new class to the Models folder named **ApplicationUser**.
2. Have the class inherit from IdentityUser, add the **using Microsoft.AspNetCore.Identity;** directive. Add two properties for user name.
3.
```
public class ApplicationUser : IdentityUser
{
    [PersonalData]
    public string FirstName { get; set; }
    [PersonalData]
    public string LastName { get; set; }
}
```
4. Save and close the file.
5. IdentityUser will now need to be replaced throughout the application with ApplicationUser.

Login.cshtml.cs

1. Open Areas / Identity / Pages / Account / Login.cshtml.cs.
2. Replace IdentityUser with ApplicationUser and add the **using SecureSite.Models;** namespace.
3.
```
namespace SecureSite.Areas.Identity.Pages.Account
{
    [AllowAnonymous]
    public class LoginModel : PageModel
    {
        private readonly UserManager<~~IdentityUser~~ApplicationUser> _userManager;
        private readonly SignInManager<~~IdentityUser~~ApplicationUser> _signInManager;
        private readonly ILogger<LoginModel> _logger;

        public LoginModel(SignInManager<~~IdentityUser~~ApplicationUser> signInManager,
            ILogger<LoginModel> logger,
            UserManager<~~IdentityUser~~ApplicationUser> userManager)
        {
            _userManager = userManager;
            _signInManager = signInManager;
            _logger = logger;
        }
        ...
```
4. Save the file.

Register.cshtml.cs

1. Open Areas / Identity / Pages / Account / Register.cshtml.cs.
2. Replace IdentityUser with ApplicationUser and add the **using SecureSite.Models;** namespace.
3.
```
namespace SecureSite.Areas.Identity.Pages.Account
{
    [AllowAnonymous]
    public class RegisterModel : PageModel
    {
        private readonly SignInManager<~~IdentityUser~~ApplicationUser> _signInManager;
        private readonly UserManager<~~IdentityUser~~ApplicationUser> _userManager;
        private readonly ILogger<RegisterModel> _logger;
        private readonly IEmailSender _emailSender;

        public RegisterModel(
            UserManager<~~IdentityUser~~ApplicationUser> userManager,
            SignInManager<~~IdentityUser~~ApplicationUser> signInManager,
            ILogger<RegisterModel> logger,
            IEmailSender emailSender)
        {
            _userManager = userManager;
```

```
            _signInManager = signInManager;
            _logger = logger;
            _emailSender = emailSender;
        }
        ...
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl ??= Url.Content("~/");
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
    if (ModelState.IsValid)
    {
        var user = new ~~IdentityUser~~ApplicationUser { UserName = Input.Email, Email = Input.Email };
        var result = await _userManager.CreateAsync(user, Input.Password);
        if (result.Succeeded)
        ...
```

4. Save the file.

_LoginPartial.cshtml

1. Open Views / Shared / _LoginPartial.cshtml.
2. Replace IdentityUser with ApplicationUser.
3.
```
@using Microsoft.AspNetCore.Identity
@inject SignInManager<~~IdentityUser~~ApplicationUser> SignInManager
@inject UserManager<~~IdentityUser~~ApplicationUser> UserManager

<ul class="navbar-nav">
...
```

4. Save the file.

Startup.cs

1. Open Startup.cs.
2. Replace IdentityUser with ApplicationUser in ConfigureServices and add the **using SecureSite.Models;** namespace.
3.
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
    services.AddDefaultIdentity<~~IdentityUser~~ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddEntityFrameworkStores<ApplicationDbContext>();

    // Configure Identity
    services.Configure<IdentityOptions>(options =>
    {
        options.Password.RequiredLength = 8;
        options.User.RequireUniqueEmail = true;
    });

    services.AddControllersWithViews();
    services.AddRazorPages();
}
```

4. Save the file.

ApplicationDbContext.cs

1. Open Data / ApplicationDbContext.cs.
2. Update the class to use the ApplicationUser, add the **using SecureSite.Models;** directive.
3.
```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
{
    public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)
        : base(options)
    {
    }
}
```

4. Build the project (Ctrl+B) to ensure no errors.

Updating the Database

1. From PMC, enter the command **Add-Migration FirstAndLastName**.
2. Enter the command **Update-Database**.
3. In SQL Server Object Explorer, view the AspNetUsers table to see the new columns.

Register.cshtml.cs

1. Open Areas / Identity / Pages / Account / Register.cshtml.cs.
2. Add first name and last name to the InputModel.
3.
```
public class InputModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} and at max {1} characters long.", MinimumLength = 8)]
```

```
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    public string ConfirmPassword { get; set; }

    [Required]
    [Display(Name = "First Name")]
    public string FirstName { get; set; }

    [Required]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }
}
```

4. Update the OnPostAsync method with FirstName and LastName.
5.
```
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl ??= Url.Content("~/");
    ExternalLogins = (await _signInManager.GetExternalAuthenticationSchemesAsync()).ToList();
    if (ModelState.IsValid)
    {
        var user = new ApplicationUser
            {
                UserName = Input.Email,
                Email = Input.Email,
                FirstName = Input.FirstName,
                LastName = Input.LastName
            };
        var result = await _userManager.CreateAsync(user, Input.Password);
        ...
```

6. Save the file.

Register.cshtml

1. Open Areas / Identity / Pages / Account / Register.cshtml.
2. Add first name and last name to the view.
3.
```
...
<form asp-route-returnUrl="@Model.ReturnUrl" method="post">
    <h4>Create a new account.</h4>
    <hr />
    <div asp-validation-summary="All" class="text-danger"></div>
    <div class="form-group">
        <label asp-for="Input.Email"></label>
        <input asp-for="Input.Email" class="form-control" autofocus />
        <span asp-validation-for="Input.Email" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.FirstName"></label>
        <input asp-for="Input.FirstName" class="form-control" />
        <span asp-validation-for="Input.FirstName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.LastName"></label>
        <input asp-for="Input.LastName" class="form-control" />
        <span asp-validation-for="Input.LastName" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.Password"></label>
        <input asp-for="Input.Password" class="form-control" />
        <span asp-validation-for="Input.Password" class="text-danger"></span>
    </div>
    <div class="form-group">
        <label asp-for="Input.ConfirmPassword"></label>
        <input asp-for="Input.ConfirmPassword" class="form-control" />
        <span asp-validation-for="Input.ConfirmPassword" class="text-danger"></span>
    </div>
    <button type="submit" class="btn btn-primary">Register</button>
</form>
...
```

4. Save the file.
5. Run the site, register a new user.

_LoginPartial.cshtml

1. Open Views / Shared / _LoginPartial.cshtml.
2. Change the login greeting to show the user's first name instead of their user name.
3.
```
@using Microsoft.AspNetCore.Identity
@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager

<ul class="navbar-nav">
@if (SignInManager.IsSignedIn(User))
{
    var user = await UserManager.GetUserAsync(User);
    var firstName = user.FirstName;
    <li class="nav-item">
        <a  class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Hello @User.Identity.Name!@firs
tName</a>
    </li>
```

4. Save the file.

5. Run the site, login with the new user if necessary. Notice the greeting.

Scaffold Identity

1. Right-click the project and select Add / New Scaffolded Item... .
2. In the list of Installed items, click Identity, select Identity, click Add.
3. Select **Account\Manage\Index**.
4. Set the Data context class to **ApplicationDbContext (SecureSite.Data)**.
5. Click Add.
6. Close the read me file.

Index.cshtml.cs

1. Open Areas / Identity / Pages / Account / Manage / Index.cshtml, select Index.cshtml.cs.
2. Add first name and last name to the InputModel.
3.
```
public class InputModel
{
    [Phone]
    [Display(Name = "Phone number")]
    public string PhoneNumber { get; set; }

    [Required]
    [Display(Name = "First Name")]
    public string FirstName { get; set; }

    [Required]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }
}
```

4. Update the LoadAsync method to retrieve first name and last name and put them in the input model.
5.
```
private async Task LoadAsync(ApplicationUser user)
{
    var userName = await _userManager.GetUserNameAsync(user);
    var phoneNumber = await _userManager.GetPhoneNumberAsync(user);
    var firstName = user.FirstName;
    var lastName = user.LastName;

    Username = userName;

    Input = new InputModel
    {
        PhoneNumber = phoneNumber,
        FirstName = firstName,
        LastName = lastName
    };
}
```

6. Update the OnPostAsync method to update first name and last name.
7.
```
public async Task<IActionResult> OnPostAsync()
{
    var user = await _userManager.GetUserAsync(User);
    if (user == null)
    {
        return NotFound($"Unable to load user with ID '{_userManager.GetUserId(User)}'.");
    }

    if (!ModelState.IsValid)
    {
        await LoadAsync(user);
        return Page();
    }

    var phoneNumber = await _userManager.GetPhoneNumberAsync(user);
    if (Input.PhoneNumber != phoneNumber)
    {
        var setPhoneResult = await _userManager.SetPhoneNumberAsync(user, Input.PhoneNumber);
        if (!setPhoneResult.Succeeded)
        {
            StatusMessage = "Unexpected error when trying to set phone number.";
            return RedirectToPage();
        }
    }

    // Update custom properties
    user.FirstName = Input.FirstName;
    user.LastName = Input.LastName;
    await _userManager.UpdateAsync(user);

    await _signInManager.RefreshSignInAsync(user);
    StatusMessage = "Your profile has been updated";
    return RedirectToPage();
}
```

8. Save the file.

Index.cshtml

1. Open Areas / Identity / Pages / Account / Manage / Index.cshtml.
2. Add first name and last name to the view.

3.
```
...
<div class="form-group">
    <label asp-for="Username"></label>
    <input asp-for="Username" class="form-control" disabled />
</div>
<div class="form-group">
    <label asp-for="Input.FirstName"></label>
    <input asp-for="Input.FirstName" class="form-control" autofocus />
    <span asp-validation-for="Input.FirstName" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="Input.LastName"></label>
    <input asp-for="Input.LastName" class="form-control" />
    <span asp-validation-for="Input.LastName" class="text-danger"></span>
</div>
<div class="form-group">
    <label asp-for="Input.PhoneNumber"></label>
    <input asp-for="Input.PhoneNumber" class="form-control" />
    <span asp-validation-for="Input.PhoneNumber" class="text-danger"></span>
</div>
...
```

4. Save the file.

_ViewImports.cshtml

1. Build the project, it will fail due to a namespace issue in Areas / Identity / Pages / Account / Manage / _ViewImports.cshtml, open the file and fix the namespace issue.
2.
```
@using SecureSite.Areas.Identity.Pages.Account.Manage
@using SecureSite.Models
```

3. Build the project.
4. Run the site, access the profile of a user and note the first name and last name properties are available.
5. Click the Hello Firstname link to access account management. Click the Personal data button, then the Download button to download personal data. Open the JSON file to see what personal data is stored for the current user.

Startup.cs

1. To enable Role based authorization, modify Identity in the ConfigureServices method.
2.
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
    services.AddDatabaseDeveloperPageExceptionFilter();

    services.AddDefaultIdentity<ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddRoles<IdentityRole>()
        .AddEntityFrameworkStores<ApplicationDbContext>();

    // Configure Identity
    services.Configure<IdentityOptions>(options =>
    {
        options.Password.RequiredLength = 8;
        options.User.RequireUniqueEmail = true;
    });

    services.AddControllersWithViews();
}
```

3. Save the file.
4. Run the site.

Roles

1. Register 3 users with the following email addresses **admin@mail.com**, **manager@mail.com** and **clerk@mail.com**.
2. In SQL Server Object Explorer, right-click the localhost\sqlexpress database and select New Query... .
3. Copy the following SQL and run it to create 3 roles and to assign users to them.
4.
```
USE [SecureSite-Identity];
INSERT INTO AspNetRoles (Id, [Name]) VALUES(NEWID(), 'Admin');
INSERT INTO AspNetRoles (Id, [Name]) VALUES(NEWID(), 'Manager');
INSERT INTO AspNetRoles (Id, [Name]) VALUES(NEWID(), 'Clerk');

INSERT INTO AspNetUserRoles
  VALUES((SELECT Id FROM AspNetUsers WHERE UserName = 'admin@mail.com'),
         (SELECT Id FROM AspNetRoles WHERE [Name] = 'Admin'));
INSERT INTO AspNetUserRoles
  VALUES((SELECT Id FROM AspNetUsers WHERE UserName = 'admin@mail.com'),
         (SELECT Id FROM AspNetRoles WHERE [Name] = 'Manager'));
INSERT INTO AspNetUserRoles
  VALUES((SELECT Id FROM AspNetUsers WHERE UserName = 'manager@mail.com'),
         (SELECT Id FROM AspNetRoles WHERE [Name] = 'Manager'));
INSERT INTO AspNetUserRoles
  VALUES((SELECT Id FROM AspNetUsers WHERE UserName = 'clerk@mail.com'),
         (SELECT Id FROM AspNetRoles WHERE [Name] = 'Clerk'));
```

5. Run the following SQL to see the roles.
6.
```
SELECT u.Email, r.[Name] [Role]
FROM AspNetUserRoles ur
```

```
JOIN AspNetRoles r ON ur.RoleId = r.Id
JOIN AspNetUsers u ON ur.UserId = u.Id
```

7. Run the site and access the Privacy page as admin@mail.com, manager@mail.com and as clerk@mail.com.

HomeController.cs

1. Decorate the Privacy method with the Admin role.
2.
```
[Authorize(Roles = "Admin")]
public IActionResult Privacy()
{
    return View();
}
```

3. Save the file. Only members of Admin will be able to access this page now.
4. Run the site and attempt to access as admin@mail.com, manager@mail.com and as clerk@mail.com.
5. Change the attribute so that members of Admin or Manager have access.
6.
```
[Authorize(Roles = "Admin, Manager")]
public IActionResult Privacy()
{
    return View();
}
```

7. Run the site and attempt to access as admin@mail.com, manager@mail.com and as clerk@mail.com.
8. Change the attribute so that a user has to be a member of both Admin and Manager to have access.
9.
```
[Authorize(Roles = "Admin~~, Manager~~")]
[Authorize(Roles = "Manager")]
public IActionResult Privacy()
{
    return View();
}
```

10. Run the site and attempt to access as admin@mail.com, manager@mail.com and as clerk@mail.com.

UsersController.cs

1. Add a new empty MVC Controller named **UsersController**.
2. Require users to be authenticated to access the new controller, add the **using Microsoft.AspNetCore.Authorization;** directive.
3.
```
[Authorize]
public class UsersController : Controller
{
    public IActionResult Index()
    {
        return View();
    }
}
```

4. Declare a property for the user manager and set it in the constructor; add the **using Microsoft.AspNetCore.Identity;** and **using SecureSite.Models;** directives.
5.
```
[Authorize]
public class UsersController : Controller
{
    private UserManager<ApplicationUser> userManager;

    public UsersController(UserManager<ApplicationUser> userManager)
    {
        this.userManager = userManager;
    }

    public IActionResult Index()
    {
        return View();
    }
}
```

6. Only allow users with the admin role access to the index method. Get the users from the user manager and return the view with them.
7.
```
[Authorize(Roles = "Admin")]
public IActionResult Index()
{
    var users = userManager.Users;
    return View(users);
}
```

8. Attempt to scaffold the view.
9. Right-click in the Index method and select Add View..., select Razor View and click Add. Set View name to **Index** and Template to **List**. Click the drop-down list for Model class and note that ApplicationUser isn't there. The View Scaffolder doesn't support Identity. Click Cancel twice.

User.cs

1. Add a new class to the Models folder named **User**.
2.
```
public class User
{
    public string UserName { get; set; }
```

```
    public string FirstName { get; set; }
    public string LastName { get; set; }
}
```

3. Save the file.

UsersController.cs

1. Scaffold the view. Right-click in the **I**ndex method and select Add View..., select Razor View and click Add. Set View name to **Index**, Template to **List** and Model class to **User (SecureSite.Models)**, click Add.

_LoginPartial.cshtml

1. Open Views / Shared / _LoginPartial.cshtml.
2. Add a link to the new UsersController in the signed in section of the view.
3.

```
@using Microsoft.AspNetCore.Identity
@inject SignInManager<ApplicationUser> SignInManager
@inject UserManager<ApplicationUser> UserManager

<ul class="navbar-nav">
    @if (SignInManager.IsSignedIn(User))
    {
        var user = await UserManager.GetUserAsync(User);
        var firstName = user.FirstName;
        <li class="nav-item">
            <a> class="nav-link text-dark" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Hello @firstName</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-controller="Users" asp-action="Index">Users</a>
        </li>
        <li class="nav-item">
        ...
```

4. Save the file.
5. Run the site and login as a user with the Admin role.
6. Attempt to access the UsersController. The error message will indicate that an ApplicationUser model was expected, but that a User model was received.

Index.cshtml

1. Open Views / Users / Index.cshtml.
2. Change the model to **ApplicationUser**. Update the title and remove the Create and Edit links.
3.

```
@model IEnumerable<SecureSite.Models.~~User~~ApplicationUser>

@{
    ViewData["Title"] = "~~Index~~Users";
}

<h1>~~Index~~@@ViewBag.Title</h1>

~~<p>~~
    ~~<a asp-action="Create">Create New</a>~~
~~</p>~~
<table class="table">
    <thead>
        <tr>
            <th>
                @Html.DisplayNameFor(model => model.UserName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.FirstName)
            </th>
            <th>
                @Html.DisplayNameFor(model => model.LastName)
            </th>
            ~~<th></th>~~
        </tr>
    </thead>
    <tbody>
@foreach (var item in Model) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.UserName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.FirstName)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LastName)
            </td>
            ~~<td>~~
                ~~@Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ }) |~~
                ~~@Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ }) |~~
                ~~@Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })~~
            ~~</td>~~
        </tr>
}
    </tbody>
</table>
```

4. Save the file and access the Users. Logout and login as [manager@mail.com (mailto:manager@mail.com)](mailto:manager@mail.com) and attempt to access the Users.