Walkthrough 15 - Authentication and Authorization

Setup

This lab will explore ASP.NET Core Identity.

1. Start SQL Server.
2. Start Visual Studio.
3. Click Create a new project.
4. Set language to **C#** and project type to **Web**.
5. Select the ASP.NET Core Web App (Model-View-Controller) template, click Next.
6. Set Project name to **SecureSite**.
7. Set Location to a **folder of your choosing**.
8. Ensure Place solution and project in the same directory is not selected, click Next.
9. Set version to **.NET 5.0**, set Authentication Type to **Individual Accounts**, leave Configure for HTTPS selected, click Create.

Data

1. Expand the Data folder and open ApplicationDbContext.cs.
2. Note that ApplicationDbContext inherits IdentityDbContext, which itself inherits from DbContext.
3. Expand the Migrations folder and examine the two generated files.

appsettings.json

1. Update the connection string for a new SQL Server Express database named SecureSite-Identity.
2.
```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=~~(localdb)~~localhost\\~~mssqllocaldb~~sqlexpress;Database=~~aspnet-SecureSite-7DAD6EFC-40A3-402E-832B-B923EEB34FC3~~SecureSite-Identity;Trusted_Connection=True;MultipleActiveResultSets=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "AllowedHosts": "*"
}
```
3. Save the file.

Update-Database

1. Open the Package Manager Console (PMC) and issue the following command.
2. **Update-Database**
3. Open SQL Server Object Explorer and examine the new database.

Create a User

1. Run the site.
2. Click the Privacy link to verify you can access it.
3. Notice the Register and Login links on the right.
4. Click Login, note the Use another service to log in message.
5. Click Register as a new user.
6. Click the Register button, note the error messages.
7. Enter an invalid email address and press the Tab key.
8. Enter a valid email address.
9. Set password and confirm password to **abc**.
10. Set passwords to **abcdef**, click the Register button. Note the error messages.
11. Set passwords to **Abcd5!** and Register.
12. In SQL Server Object Explorer, view the data of the AspNetUsers table. Notice that EmailConfirmed is False.
13. Back in the browser, on the Register confirmation page that appears, click the **Click here to confirm your account** link.
14. In SQL Server Object Explorer, click the Refresh button to see that EmailConfirmed is now True. Leave this window open.
15. Login with the newly created account.
16. Click the Hello email address link.
17. Explore the various account management options.
18. Click the Logout link.
19. Register and confirm another user with the same password **Abcd5!**.
20. Refresh the AspNetUsers data and notice that even though the passwords were the same, due to salting, the PasswordHash values are different.

_Layout.cshtml

1. Open Views / Shared / _Layout.cshtml
2. Notice in the navbar div, there is a partial view named _LoginPartial
3.
```
...
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
    <ul class="navbar-nav flex-grow-1">
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
        </li>
        <li class="nav-item">
            <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </li>
    </ul>
    <partial name="_LoginPartial" />
</div>
...
```

_LoginPartial.cshtml

1. Open Views / Shared / _LoginPartial.cshtml
2. Notice that it has a section that displays Hello and Logout if the user is signed in; and a section with Register and Login.

HomeController.cs

1. In the Controllers folder, open HomeController.
2. Decorate the Privacy method with an Authorize attribute, add the **using Microsoft.AspNetCore.Authorization;** directive.
3.
```
[Authorize]
public IActionResult Privacy()
{
    return View();
}
```
4. Save the file.
5. Run the site, click the Privacy link. You will be required to login, do so.
6. Logout.
7. Move the Authorize attribute to the class.
8.
```
[Authorize]
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;

    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }

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

    ~~[Authorize]~~
    public IActionResult Privacy()
    {
        return View();
    }

    [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
    public IActionResult Error()
    {
        return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
    }
}
```
9. Save the file.
10. Run the site, you will be required to login to access any endpoint of this Controller, do so.
11. Logout.
12. Decorate the Index method with the AllowAnonymous attribute.
13.
```
[AllowAnonymous]
public IActionResult Index()
{
    return View();
}
```
14. Save the file.
15. Run the site, access the home page.
16. Click the Privacy link, login.
17. Logout.

Startup.cs

1. Edit the ConfigureServices method to begin configuring Identity.
2.
```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<ApplicationDbContext>(options =>
        options.UseSqlServer(
            Configuration.GetConnectionString("DefaultConnection")));
    services.AddDatabaseDeveloperPageExceptionFilter();

    services.AddDefaultIdentity<IdentityUser>(options => options.SignIn.RequireConfirmedAccount = true)
        .AddEntityFrameworkStores<ApplicationDbContext>();

    // Configure Identity
    services.Configure<IdentityOptions>(options =>
    {

    });

    services.AddControllersWithViews();
}
```

3. Put the cursor on IdentityOptions and press F1.
4. Check the PasswordOptions, SignInOptions and UserOptions.
5. Configure a couple of options.
6.
```
...
// Configure Identity
services.Configure<IdentityOptions>(options =>
{
    options.Password.RequiredLength = 8;
    options.User.RequireUniqueEmail = true;
});
...
```
7. Save the file.
8. Run the site, begin registering a new user with the password **abc**, tab to the Confirm password textbox. The initial error message about password length will be incorrect. Provide the password **Abcd5!** and attempt to Register. This error message is correct.

Scaffold Identity

1. In Solution Explorer, expand Areas / Identity / Pages. This is where Identity customization will happen.
2. Right-click the project and select Add / New Scaffolded Item... .
3. In the list of Installed items, click Identity, select Identity, click Add.
4. Select **Account\Login** and **Account\Register**.
5. Set the Data context class to **ApplicationDbContext (SecureSite.Data)**.
6. Click Add.
7. Close the read me file.
8. In Solution Explorer, expand Areas / Identity / Pages / Account, select Login.cshtml.

Login.cshtml

1. Delete the local account heading and give the first input autofocus.
2.
```
@page
@model LoginModel

@{
    ViewData["Title"] = "Log in";
}

<h1>@ViewData["Title"]</h1>
<div class="row">
    <div class="col-md-4">
        <section>
            <form id="account" method="post">
                ~~<h4>Use a local account to log in.</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>
                    ...
```
3. Delete the Use another service to log in markup.
4.
```
    ...
    </div>
    ~~<div class="col-md-6 col-md-offset-2">~~
        ~~<section>~~
            ~~<h4>Use another service to log in.</h4>~~
            ~~<hr />~~
            ~~@{~~
                ~~if ((Model.ExternalLogins?.Count ?? 0) == 0)~~
                ~~{~~
                    ~~<div>~~
                        ~~<p>~~
                            ~~There are no external authentication services configured. See <a href="https://go.microsoft.com/fwlink/?LinkID~~
~~=532715">this article</a>~~
                            ~~for details on setting up this ASP.NET application to support logging in via external services.~~
                        ~~</p>~~
                    ~~</div>~~
                ~~}~~
                ~~else~~
                ~~{~~
                    ~~<form id="external-account" asp-page="./ExternalLogin" asp-route-returnUrl="@Model.ReturnUrl" method="post" class="for~~
~~m-horizontal">~~
                        ~~<div>~~
                            ~~<p>~~
```

```
                                ~~@foreach (var provider in Model.ExternalLogins)~~
                                ~~{~~
                                    ~~<button type="submit" class="btn btn-primary" name="provider" value="@provider.Name" title="Log in usi~~
~~ng your @provider.DisplayName account">@provider.DisplayName</button>~~
                                ~~}~~
                            ~~</p>~~
                        ~~</div>~~
                    ~~</form>~~
                ~~}~~
            ~~}~~
        ~~</section>~~
    ~~</div>~~
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

5. Save the file.

Register.cshtml

1. Give the first input autofocus.
2.

```
@page
@model RegisterModel
@{
    ViewData["Title"] = "Register";
}

<h1>@ViewData["Title"]</h1>

<div class="row">
    <div class="col-md-4">
        <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>
                ...
```

3. Delete the Use another service to register in markup.
4.

```
    ...
    </div>
    ~~<div class="col-md-6 col-md-offset-2">~~
        ~~<section>~~
            ~~<h4>Use another service to register.</h4>~~
            ~~<hr />~~
            ~~@{~~
                ~~if ((Model.ExternalLogins?.Count ?? 0) == 0)~~
                ~~{~~
                    ~~<div>~~
                        ~~<p>~~
                            ~~There are no external authentication services configured. See <a> href="https://go.microsoft.com/fwlink/?LinkI~~
~~D=532715">this article</a>~~
                            ~~for details on setting up this ASP.NET application to support logging in via external services.~~
                        ~~</p>~~
                    ~~</div>~~
                ~~}~~
                ~~else~~
                ~~{~~
                    ~~<form id="external-account" asp-page="./ExternalLogin" asp-route-returnUrl="@Model.ReturnUrl" method="post" class="for~~
~~m-horizontal">~~
                        ~~<div>~~
                            ~~<p>~~
                                ~~@foreach (var provider in Model.ExternalLogins)~~
                                ~~{~~
                                    ~~<button> type="submit" class="btn btn-primary" name="provider" value="@provider.Name" title="Log in us~~
~~ing your @provider.DisplayName account">@provider.DisplayName</button>~~
                                ~~}~~
                            ~~</p>~~
                        ~~</div>~~
                    ~~</form>~~
                ~~}~~
            ~~}~~
        ~~</section>~~
    ~~</div>~~
</div>

@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

5. Save the file.

Register.cshtml.cs

1. Expand Register.cshtml and select Register.cshtml.cs.
2. Find the InputModel class in the file.
3. Update the StringLength attribute of the Password property.

```
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 = ~~6~~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; }
}
```

1. Save the file.
2. Run the site, begin registering a new user with the password **abc**. The initial error message about password length will be correct now.