BAIS3110 Assignment 8 - Authentication #1 (2020)

This Lab is the first step in building a framework for securing ASP.NET Razor Page apps.

The goal is to setup an Authentication system that is totally separated from any other technologies such as Entity Framework. If you really want to work with EF there are many examples at docs.microsoft.com and elsewhere on the web.

We will be following this webpage but I am breaking it down into a series of steps for you to follow with less explanation, please refer to the Microsoft site for explanations and Details.

Part #1 is just going to configure the system for *cookie based authentication*, the account name and password is hardcoded so you shouldn't deploy this as a production level application. The next few labs will be adding secure password storage in a database and Role and Permissions.

1. Create a new Razor Pages App named BAIS3110Authentication
   - Select the Web Application Template
   - Clear the "Configure for HTTPS" Checkbox to not deal with Certs and setup issues in this lab
2. Right Click the Pages Folder and Add a new Folder named Admin
3. Right Click Admin and add 3 Pages (Scaffold with plain old Razor, no EF):
   - Forbidden.cshtml
   - Index.cshtml
   - Logout.cshtml
4. Open Startup.cs to configure Authentication.
   Add a namespace at the top of the page:

   ```
   using Microsoft.AspNetCore.Authentication.Cookies;
   ```

   Add the following code to the beginning of the ConfigureServices() method:

```
public void ConfigureServices(IServiceCollection services)
    {

    services.Configure<CookiePolicyOptions>(options =>
    {
        // This lambda determines whether user consent for non-essential cookies is needed for a given request.
        options.CheckConsentNeeded = context => true;
        options.MinimumSameSitePolicy = Microsoft.AspNetCore.Http.SameSiteMode.None;
    });
    services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
        .AddCookie(cookieOptions =>
        {
            cookieOptions.LoginPath = "/Login";
            cookieOptions.LogoutPath = "/Admin/Logout";
            cookieOptions.SlidingExpiration = true;
            cookieOptions.AccessDeniedPath = "/Admin/Forbidden";
            cookieOptions.ExpireTimeSpan = TimeSpan.FromDays(2); // Make this smaller for production
            cookieOptions.Cookie.HttpOnly = true;
        });
```

   In the Configure() method add app.UseAuthentication():

   ```
   app.UseRouting();

   app.UseAuthentication();
   app.UseAuthorization();
   ```

5. Add a Pages/Login.cshtml Razor Page:

```
<h1>Login</h1>
<div>
    @Model.Message
</div>
<form method="post">
    <div class="form-group">
        <label asp-for="Email" class="col-form-label col-md-2"></label>
        <div class="col-md-10">
            <input asp-for="Email" />
        </div>
    </div>
    <div class="form-group">
        <label asp-for="Password" class="col-form-label col-md-2"></label>
        <div class="col-md-10">
            <input asp-for="Password" />
        </div>
    </div>
    <div class="form-group">
        <button class="btn btn-outline-primary btn-sm">Log in</button>
    </div>
</form>
```

And the Login.cshml.cs:

```
using System.ComponentModel.DataAnnotations;
using System.Security.Claims;
using Microsoft.AspNetCore.Authentication.Cookies;
using Microsoft.AspNetCore.Authentication;

public class LoginModel : PageModel
    {
    [BindProperty]
    public string Email { get; set; }

    [BindProperty, DataType(DataType.Password)]
    public string Password { get; set; }
    public string Message { get; set; }

    public async Task<IActionResult> OnPost()
        {
        // Hardcoded User Credentials
        string UserEmail ="Bob@nait.ca";
        string UserName = "Bob";
        string UserPassword ="123";
        string UserRole ="Admin";

        if (Email == UserEmail)
            {
            if (Password == UserPassword)
                {
                var claims = new List<Claim>
                {
                    new Claim(ClaimTypes.Email, Email),
                    new Claim(ClaimTypes.Name, UserName)
                };
                var claimsIdentity = new ClaimsIdentity(claims,
                CookieAuthenticationDefaults.AuthenticationScheme);

                claimsIdentity.AddClaim(new Claim(ClaimTypes.Role, UserRole));

                AuthenticationProperties authProperties = new AuthenticationProperties
```

```
                    {
                    //AllowRefresh = <bool>,
                    // Refreshing the authentication session should be allowed.

                    //ExpiresUtc = DateTimeOffset.UtcNow.AddMinutes(10),
                    // The time at which the authentication ticket expires. A
                    // value set here overrides the ExpireTimeSpan option of
                    // CookieAuthenticationOptions set with AddCookie.

                    //IsPersistent = true,
                    // Whether the authentication session is persisted across
                    // multiple requests. When used with cookies, controls
                    // whether the cookie's lifetime is absolute (matching the
                    // lifetime of the authentication ticket) or session-based.

                    //IssuedUtc = <DateTimeOffset>,
                    // The time at which the authentication ticket was issued.

                    //RedirectUri = <string>
                    // The full path or absolute URI to be used as an http
                    // redirect response value.
                    };

                await HttpContext.SignInAsync(CookieAuthenticationDefaults.AuthenticationScheme,
                  new ClaimsPrincipal(claimsIdentity), authProperties);
                return RedirectToPage("/Admin/Index");
                }
            }
        Message = "Invalid attempt";
        return Page();
        }
```

6. Let's look at some Properties in the Pages/Index.cshtml file:

```
<div class="text-center">
    <h1 class="display-4">Authentication Example</h1>
    Is User Authenticated: @Model.User.Identity.IsAuthenticated <br />
    User Name: @Model.User.Identity.Name <br />
    @if (User.Identity.IsAuthenticated)
        {
        <div>Hello, @User.Identity.Name, the time is @DateTime.Now</div>
        }

    <br />
    Claims
    @foreach (System.Security.Claims.Claim claim in User.Claims)
        {
        <p>Claim Type: @claim.Type, Claim Value: @claim.Value <br /></p>
        }
</div>
```

7. Add some text to Pages/Admin/Forbidden.cstml:

```
<h1>Forbidden</h1>
You are not allowed to Access this resource
```

8. Add some text to Pages/Admin/Index.cstml:

```
<h1>Admin</h1>
Welcome to the Administration page
```

9. Add some text to Pages/Admin/Logout.cstml:

```
<h1>Logout</h1>
Logging you out.
```

And some code to Pages/Admin/Logout.cstml.cs:

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;

using Microsoft.AspNetCore.Authentication;
using Microsoft.AspNetCore.Authentication.Cookies;

namespace BAIS3110Authentication.Pages.Admin
{
    public class LogoutModel : PageModel
    {
        public async Task<IActionResult> OnGet()
            {
            await HttpContext.SignOutAsync(
                CookieAuthenticationDefaults.AuthenticationScheme);

            return Page();
            }
        }
}
```

10. Almost there. Edit the Pages/Privacy.cshtml.cs page:

```
using Microsoft.AspNetCore.Authorization;

namespace BAIS3110Authentication.Pages
    {
    [Authorize]
    public class PrivacyModel : PageModel
```

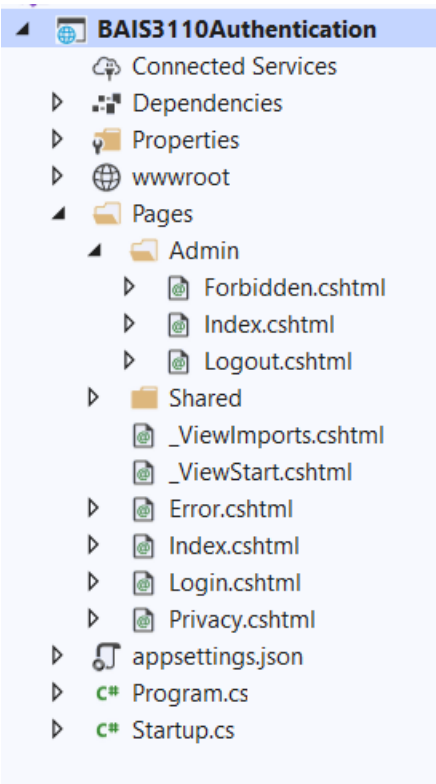
Adding the [Authorize] Attribute to a PageModel class requires the user to be authenticated in order to view the Page which forces the Application to redirect to the Login if you are not Authenticated.

11. Testing

Run your Program.	Note that the Home Page shows you aren't Logged in.
Click Privacy.	Note that you are redirected to Login.
Login.	Note that you are directed to Admin/Index (not the greatest, you should fix it)
Click Privacy again.	Note the page actually displayed this time.
Click Home	Note the Authentication Information is now being displayed
Close the App and rerun it.	Note you are still Logged in because you made the Cookie Persistent. UnComment `IsPersistent = true;` in Login.cshtml.cs, if required

Add /Admin/Logout to the end of the URL so you can Log out (Logout button in the menu would be nice)

Comments



The [Authorize] Attribute in the page models only tells if you are logged in at this point. We will remedy this in a future Lab.

Login should really Direct you back to the Last Page you were trying to Access, this is left to the student to play with (EFTS Exercise For The Student).

Your Name should be displayed in the upper right hand side of the menu bar along with a logout button, maybe even a Login button if you aren't logged in (EFTS).

Numerous commented blocks were left in the code to allow you to customize the functionality.

Submission

Change the Hardcoded UserCredentials in Login/OnPost() to your name and email and Submit a Screen shot of the Home page showing you are logged in.

It would be awesome if you just submitted the Screen Shot.