go deploy

Screenshots

Module 11: Managing Security

Lab: Managing Security

Scenario

You have been asked to create a web-based library application for your organization's customers. The application should have a page showing recommended books, and support user login and registration. Logged in users should have the ability to add borrow books, and users with the 'admin' role should be able to add books to the library. The application should also have a demonstration of a cross-site request forgery attack.

Exercise 1: Use Identity

Scenario

In this exercise, you will first add an entity-framework-database context to the LibraryContext class. You will then enable using identity. After that, you will add sign-in, and register user logic. Finally, you will retrieve data from the identity property in the LendingBook.cshtml view.

The main tasks for this exercise are as follows:

- Add the Entity Framework database context
- Enable using Identity
- Add sign in and user registration
- Run the application

Task 1: Run the starter application

1. In File Explorer, navigate to **D:\Allfiles\Mod11\Labfiles\01_Library_begin**, and then double-click **Library.sln**.

   Note: If a **Security Warning for Library** dialog box appears, verify that the **Ask me for every project in this solution** check box is cleared, and then click OK.

2. In the **Library - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Debugging**.

   Note: The application starts in a browser showing the home page, with a list of

go deploy

- [ ] 3. In the menu bar, click **Books**, and click on one of the **Borrow a Book** links.

> **Note**: The **Books** page and the option to borrow a book is only intended for authorized users. In this exercise we're going to restrict access to this page, while allowing anonymous users to see the Home page.

- [ ] 4. In Microsoft Edge, click **Close**.
- [ ] 5. In the **Library - Microsoft Visual Studio** window, on the Tools menu, choose **NuGet Packager Manager**, and then **Manage NuGet Packages for Solution...**.
- [ ] 6. Make sure that the **Installed** tab is selected, and notice that the packages **Microsoft.AspNetCore.Identity.UI** and **Microsoft.AspNetCore.Identity.EntityFrameworkCore** have already been installed, in addition to **Microsoft.EntityFrameworkCore.Sqlite**.

Task 2: Enable using Identity

- [ ] 1. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, open **Program.cs**.
- [ ] 2. In the **Program.cs** code window, locate the following code:

```
using Microsoft.EntityFrameworkCore;
```

- [ ] 3. After the located code, add the following line of code:

```
using Microsoft.AspNetCore.Identity;
```

- [ ] 4. In the **Program.cs** code window, locate the following code:

```
builder.Services.AddDbContext<LibraryContext>(options =>
    options.UseSqlite("Data Source=library.db"));
```

- [ ] 5. After the located code, add the following code:

```
builder.Services.AddDefaultIdentity<IdentityUser>(
    options => options.SignIn.RequireConfirmedAccount = true)
    .AddEntityFrameworkStores<LibraryContext>();
```

- [ ] 6. In the **Program.cs** code window, locate the following code:

```
app.UseRouting();
```

- [ ] 7. After the located code, add the following code:

8. In the **Program.cs** code window, locate the following code:

```
app.Run();
```

9. **Before** the located code, add the following code:

```
app.MapRazorPages();
```

10. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, open the **Data** folder and open **LibraryContext.cs**.

11. In the **LibraryContext.cs** code window, note that the data context is currently derived from **DbContext**:

```
public class LibraryContext : DbContext
```

12. In the **LibraryContext.cs** code window, modify the data context so that it is derived from IdentityDbContext (rather than DbContext):

```
public class LibraryContext : IdentityDbContext
```

13. At the beginning of the **LibraryContext.cs** code window, note that the following namespaces have been added to use ASP.NET Core identity:

```
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
```

14. In the **LibraryContext.cs** code window, locate the line of code:

```
    base.OnModelCreating(modelBuilder);
```

> **Note**: we are seeding the database with book data in the **OnModelCreating** method override.

15. In the **LibraryContext.cs** code window, following the located code, press return twice, and add the following code to seed some initial identities (just as a convenience for the purposes of the lab):

go deploy

```
var users = new List<IdentityUser>();
var usernames = new List<string>() { "bill", "mary", "mike", "tom" };
foreach (var name in usernames)
{
    var user = new IdentityUser
    {
        Email = name + "@foo.com",
        NormalizedEmail = name.ToUpper() + "@FOO.COM",
        UserName = name + "@foo.com",
        NormalizedUserName = name.ToUpper() + "@FOO.COM",
        PhoneNumber = "+123456789",
        EmailConfirmed = true,
        PhoneNumberConfirmed = true
    };
    var passwordHash = password.HashPassword(user, "FooBar11!");
    user.PasswordHash = passwordHash;
    users.Add(user);
}

// Add the initial users
modelBuilder.Entity<IdentityUser>().HasData(users);

// Add a role "admin"
var adminRole = new IdentityRole
{
    Name = "admin",
    NormalizedName = "ADMIN"
};
modelBuilder.Entity<IdentityRole>().HasData(adminRole);

// Assign role to the first user in the list
modelBuilder.Entity<IdentityUserRole<string>>().HasData(
    new IdentityUserRole<string>
    {
        RoleId = adminRole.Id,
        UserId = users[0].Id
    }
);
```

- [ ] 16. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, open the **Views** folder, and then the **Shared** folder, and then open **_Layout.cshtml**.
- [ ] 17. In the **_Layout.cshtml** code window, locate the following code that represents the navigation menu:

go deploy

```
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-act
            </li>
            <li class="nav-item">
                <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-act
            </li>
        </ul>
    </div>
```

- [ ] 18. Before the final closing div of the selected code, insert the following code to provide the login control for ASP.NET Core identity:

```
<partial name="_LoginPartial" />
```

Task 3: Run the application

- [ ] 1. In the **Library - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.
- [ ] 2. In the **Library - Microsoft Visual Studio** window, in **Solution Explorer**, select the files **library.db**, **library.db-shm**, and **library.db-wal** (if present), right-click and choose **Delete**, and click **OK** on the warning.

> **Note**: This will delete the database. Next time we run the app, we'll regenerate it with a new schema.

- [ ] 3. In the **Library - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Debugging**.

> **Note**: The Library site now has *Register* and *Login* buttons. We can still access the **Books** page without logging in, though.

- [ ] 4. In the browser, click on the **Login** button, and log in with the email **mary@foo.com** and the password **FooBar11!**.

> **Note**: You have successfully logged in to the app, and your email appears at the top of the page, using the identity of one of the users that was used to seed the database.

- [ ] 5. In the browser, click on the **Logout** button.

> **Note**: You have now logged out, but notice that you still have access to the Books page.

> **Note**: Note that by default the password complexity requirements are enforced. This user will be created in the database.

- [ ] 7. In the browser, click on the **Click here to confirm your account** link.

> **Note**: For the demo, there is no real email verification. In a production application you would configure email, and the user would need to confirm the account by sending a response from their email account.

- [ ] 8. In the browser, click on the **Login** button, and log in with the email and password you just created.

> **Note**: You have successfully registered a user and used that identity to log in to the app. Your email appears at the top of the page, along with a logout button.

- [ ] 9. In the browser, click on the **Logout** button.

> **Note**: You have logged out again, but notice that you still have access to the Books page.

- [ ] 10. In Microsoft Edge, click **Close**.

Task 4: Restrict access to the book borrowing pages

- [ ] 1. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, open the **Controllers** folder and open **HomeController.cs**.
- [ ] 2. At the beginning of the **HomeController.cs** code window, add the following code:

```
using Microsoft.AspNetCore.Authorization;
```

- [ ] 3. In the **LibraryContext.cs** code window, locate the following code:

```
public IActionResult GetBooksByGenre()
{
    var booksGenreQuery = from b in _context.Books
                          orderby b.Genre.Name
                          select b;

    return View(booksGenreQuery);
}
```

go deploy

```
[Authorize]
```

5. In the **LibraryContext.cs** code window, locate the following code:

```
public IActionResult LendingBook(int id)
{
    Book book = _context.Books.FirstOrDefault(b => b.Id == id);
    if (book == null)
    {
        return NotFound();
    }
    return View(book);
}
```

6. Immediately before the method, add the following attribute:

```
[Authorize]
```

7. In the **LibraryContext.cs** code window, locate the following code:

```
[HttpPost, ActionName("LendingBook")]
public async Task<IActionResult> LendingBookPost(int id)
{
    var bookToUpdate = _context.Books.FirstOrDefault(b => b.Id == id);
    bookToUpdate.Available = false;
    if (await TryUpdateModelAsync<Book>(
        bookToUpdate,
        "",
        b => b.Available))
    {
        _context.SaveChanges();
        return RedirectToAction(nameof(Index));
    }
    return View(bookToUpdate);
}
```

8. Immediately before the method, add the following attribute:

```
[Authorize]
```

Task 5: Run the application

1. In the **Library - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.
2. In the **Library - Microsoft Visual Studio** window, on the **Debug** menu, click **Start Debugging**.

go deploy

navigate to the **Privacy** page from the link in the footer.

- [ ] 3. Click on the **Books** link in the navigation menu.

> **Note**: You should now see a login prompt.

- [ ] 4. Log in to the application with the username **mary@foo.com** and the password **FooBar11!**.

> **Note**: You should now be able to see the books page again because you are an authenticated user.

- [ ] 5. Click on one of the **Borrow a Book** links, and verify that you are allowed to borrow a book.
- [ ] 6. In Microsoft Edge, click **Close**.

> ✓ **Results**: After completing this exercise, you have configured identity in the application, and added the [Authorize] attribute to restrict access to certain content.

Exercise 2: Add Authorization

Scenario

In this exercise, you will add the Authorize attribute to the LibrarianController class. You will then configure role-based authorization by modifying the relevant attribute in the LibrarianController class.

The main tasks for this exercise are as follows:

- Enable Authorization
- Add the Authorize attribute to an action
- Add a specific Role to the Authorize attribute
- Run the application.

Task 1: Enable Authorization

- [ ] 1. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, click **Program.cs**.
- [ ] 2. In the **Program.cs** code window, locate the following code:

go deploy

3. Ensure that the cursor is at the end of the second line, press Enter, and then type the following code:

```
.AddRoles<IdentityRole>()
```

4. Verify that the edited code is as follows:

```
builder.Services.AddDefaultIdentity<IdentityUser>(
    options => options.SignIn.RequireConfirmedAccount = true)
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<LibraryContext>();
```

5. In Solution Explorer, open the file **_Layout.cshtml** in the Views/Shared folder:

6. In the **_Layout.cshtml** file, locate the following code:

```
<li class="nav-item">
    <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Get
</li>
```

7. After the located code, add the following code to conditionally show an additional menu item:

```
@if (User.IsInRole("admin"))
{
    <li class="nav-item">
        <a class="nav-link text-dark" asp-area="" asp-controller="Librarian" asp-ac
    </li>
}
```

8. In Solution Explorer, open the **LibrarianController.cs** file in the Controllers folder.

9. Notice that as currently configured, an anonymous user could invoke the **LibrarianController** actions if they were able to guess the URL **/Librarian/Index**. We will add Authorize attributes to these methods. Add the following line of code immediately before **public IActionResult Index()**:

```
[Authorize]
```

10. Add the following code immediately before **public IActionResult AddBook()**:

```
[Authorize]
```

11. Add the following code immediately before **public IActionResult AddBookPost(Book book)**:

go deploy

Task 2: Run the application

- [ ] 1. In the **Library - Microsoft Visual Studio** window, on the **FILE** menu, click **Save All**.
- [ ] 2. In the **Library - Microsoft Visual Studio** window, on the **DEBUG** menu, click **Start Without Debugging**.
- [ ] 3. In the menu bar in the browser, click **Login**.
- [ ] 4. On the **Login** page, log in with the username **mary@foo.com** and password **FooBar11!**.

> **Note**: This user does not see any additional menu items because they have not been given the admin role (only the user bill@foo.com was given this role as part of the database seeding).

- [ ] 5. In the browser, change the URL path from **https://localhost:7143/** to **https://localhost:7143/Librarian/Index**.

> **Note**: This user has not been given the admin role, but is still able to get to the admin page by 'hacking' the URL. We'll fix this in the next task.

- [ ] 6. In Microsoft Edge, click **Close**.

Task 3: Add the role to the Authorize attributes

- [ ] 1. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, expand **Controllers**, and then click **LibrarianController.cs**.
- [ ] 2. In the **LibrarianController.cs** code window, locate the following code:

```
[Authorize]
public IActionResult Index()
{
    return View();
}
```

- [ ] 3. Modify the Authorize attribute as follows:

```
[Authorize(Roles = "admin")]
public IActionResult Index()
{
    return View();
}
```

go deploy

Task 4: Run the application

- [ ] 1. In the **Library - Microsoft Visual Studio** window, on the **FILE** menu, click **Save All**.
- [ ] 2. In the **Library - Microsoft Visual Studio** window, on the **DEBUG** menu, click **Start Debugging**.
- [ ] 3. In the menu bar, click **Login**.
- [ ] 4. On the **Login** page, enter the email mary@foo.com and the password FooBar11!.
- [ ] 5. In the browser, change the URL path from **https://localhost:7143/** to **https://localhost:7143/Librarian/Index**.

> **Note**: Access to the controller action is blocked by the Authorize attribute, and the user sees an **Access denied** error page.

- [ ] 6. In the browser, click on the **Logout** link.
- [ ] 7. Log back in to the application with the username bill@foo.com and the password FooBar11!.

> **Note**: You should now see the Admin option in the menu because Bill has been seeded with the admin role.

- [ ] 8. Click on one of the **Admin** link in the menu, and verify that you can see the Admin Portal page.
- [ ] 9. Click on the **Add Book to Library** link.
- [ ] 10. On the **Add Book to Library** page, in the **Genre** list, select **Mystery**.
- [ ] 11. On the **Add Book to Library** page, in the **Name** box, type The Riverside Villas Murder.
- [ ] 12. On the **Add Book to Library** page, in the **Author** box, type Kingsley Amis.
- [ ] 13. On the **Add Book to Library** page, in the **Date Published** box, use the date picker to pick 1st September, 1973 (or you can just pick any date).
- [ ] 14. On the **Add Book to Library** page, in the **Photo** box, import an image from **D:\Allfiles\Mod11\Labfiles\Image\book.jpg**, and then click **Add a Book**.
- [ ] 15. Click **Back to Our Books**.

> **Note**: The book you added is in the library books list.

- [ ] 16. In the menu bar, click **Logout**.

go deploy

✓ **Results**: After completing this exercise, you have added role-based authorization to the application.

Exercise 3: Demonstrate Resistance to Cross-Site Request Forgery Attacks

Scenario

In this exercise, you will first write the Cross-Site Request Forgery attack in a separate project. You will then run the application and see the possible attack. Finally, you will avoid the Cross-Site Request Forgery attack by adding the ValidateAntiForgeryToken attribute in the AccountController class, run the application, and see that the attack is no longer possible.

The main tasks for this exercise are as follows:

- Write the Cross-Site Request Forgery attack
- Run the application - the attack succeeds
- Add the ValidateAntiForgeryToken attribute
- Run the application - the attack fails

Task 1: Write the Cross-Site Request Forgery attack

1. In Solution Explorer, right-click **CrossSiteRequestForgeryAttack**, point to **Add**, and then click **New Folder**.
2. In the **NewFolder** box, type **Controllers**, and then press Enter.
3. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, right-click the **Controllers** folder, point to **Add**, and then click **Controller...**.
4. In the **Add New Scaffolded Item** dialog box, click **MVC Controller - Empty**, and then click **Add**.
5. In the **Add New Item** dialog box, ensure that value in the **Name:** textbox is **HomeController.cshtml**, and then click **Add**.
6. In the **HomeController.cs** code window, right-click the following code, and then click **Add View...**.

```
public IActionResult Index()
```

7. In the **Add New Scaffolded Item** dialog box, ensure that the **Razor View** template is selected, and click **Add**.
8. In the **Add Razor View** dialog box, ensure that value in the **View name** textbox is **Index**, and that both the **Create as partial view** and **Use a layout page** checkboxes are unchecked, and then click

go deploy

9. In the **Index.cshtml** code window, locate the following code:

```
<title>Index</title>
```

10. Place the cursor at the end of the located code, press Enter, and then type the following code:

```
<link href="~/css/style.css" rel="stylesheet" />
```

11. In the **Index.cshtml** code window, in the **BODY** element, type the following code:

```
<h1>Cross-Site Request Forgery Attack</h1>
<h3>Click - Submit to Perform the Book Borrowing Attack</h3>
<form action="https://localhost:7143/Home/LendingBook" method="post">
    <input name="id" type="number" value="2" />
    <input type="submit" value="Attack">
</form>
```

Task 2: Run the application -- the attack is possible

1. In the **Library - Microsoft Visual Studio** window, on the **FILE** menu, click **Save All**.
2. In Solution Explorer, right-click **Library**, and on the **DEBUG** menu click **Start New Instance**.

> **Note**: The menu bar has a navigation tab for **Login**, meaning you are not logged in.

3. In the menu bar in the browser, click **Login**.
4. On the **Login** page, log in with the username **mary@foo.com** and password **FooBar11!**.

> **Note**: You are now able to navigate to the Books page because the user is logged in.

5. In Visual Studio, Solution Explorer, right-click **CrossSiteRequestForgeryAttack**, point to **Debug**, and then click **Start Without Debugging**.

> **Note**: You may need to make Solution Explorer visible by clicking on its tab in the navigation bar on the right of the panel.

6. On the **Cross-Site Request Forgery Attack** page, click **Attack**.

> **Note**: The browser takes you to the book borrowing confirmation page, and you can see that you are logged in as mary@foo.com, even though the attack was initiated from

go deploy

- [ ] 7. In Microsoft Edge, click **Close**.

Task 3: Avoid the Cross-Site Request Forgery attack

- [ ] 1. In the **Library - Microsoft Visual Studio** window, in Solution Explorer, under **Controllers**, click **HomeController.cs**.
- [ ] 2. In the **LendingBookPost** action code block, locate the following code:

```
[HttpPost, ActionName("LendingBook")]
public async Task<IActionResult> LendingBookPost(int id)
```

- [ ] 3. Place the cursor before the located code, and then insert the following code:

```
[ValidateAntiForgeryToken]
```

Task 4: Run the application -- The attack fails

- [ ] 1. In the **Library - Microsoft Visual Studio** window, on the **File** menu, click **Save All**.
- [ ] 2. In Solution Explorer, right-click **Library**, and on the **DEBUG** menu click **Start New Instance**.

> **Note**: The menu bar has a navigation tab for **Login**, meaning you are not logged in.

- [ ] 3. In the menu bar in the browser, click **Login**.
- [ ] 4. On the **Login** page, log in with the username **mary@foo.com** and password **FooBar11!**.

> **Note**: You are now able to navigate to the Books page because the user is logged in.

- [ ] 5. The **CrossSiteRequestForgeryAttack** should still be running. If not, in Solution Explorer, right-click the **CrossSiteRequestForgeryAttack** project, point to **Debug**, and then click **Start Without Debugging**.
- [ ] 6. On the **Cross-Site Request Forgery Attack** page, click **Attack** for the book borrowing attack.

> **Note**: The browser reports a 400 error, and the book is not borrowed. The attack has been thwarted.

- [ ] 7. In Microsoft Edge, click **Close**.

go deploy

- [ ] 8. On the **Library(Running) - Microsoft Visual Studio** window, on the Debug menu, click **Stop Debugging**.

✓ **Results**: After completing this exercise, you have demonstrated that the site has been hardened against a cross-site request forgery (XSRF) attack.