CHAPTER 1

Getting Ready

ASP.NET Core Identity is the user management system for ASP.NET Core applications. It provides an API for managing users and roles and for signing users into and out of applications. Users can sign in with simple passwords, use two-factor authentication, or sign in using third-party platforms provided by Google, Facebook, and Twitter.

In this book, I explain how ASP.NET Core Identity is used and how it works behind the scenes. I describe the built-in features, from those that can be used for simple applications through to deep customizations for advanced projects.

What Do You Need to Know?

This is an advanced book written for experienced developers. You should not attempt to read this book unless you are familiar with web development using ASP.NET Core. You must understand HTML and how it is produced using Razor Pages and the MVC Framework. You must have at least a basic understanding of application security, although I introduce key concepts as I explain how they are implemented by ASP.NET Core Identity.

What Is the Structure of This Book?

This book is split into two parts, each of which covers a set of related topics.

Part 1: Using ASP.NET Core Identity

In Part 1 of this book, I show you how to apply ASP.NET Core Identity to an ASP.NET Core project. I show you how to set up and configure Identity, how to use the built-in Identity UI package to manage user accounts, and how to use the API to create custom workflows. By the end of this part of the book, you will be ready to use ASP.NET Core Identity in your projects.

Part 2: Understanding ASP.NET Core

In Part 2 of this book, I dig deep into the detail and explain how the features used in Part 1 are implemented, how they can be customized, and how they relate to the features provided by the ASP.NET Core platform. By the end of this part of the book, you will be ready to create custom implementations of the interfaces that shape ASP.NET Core Identity to suit the needs of the most advanced ASP.NET Core projects.

A. Freeman, *Pro ASP.NET Core Identity*, https://doi.org/10.1007/978-1-4842-6858-2_1

What Doesn't This Book Cover?

As noted, this book is for experienced ASP.NET Core developers who want to understand ASP.NET Core Identity and apply it in their ASP.NET Core project. It doesn't explain the basics of web applications or programming. It doesn't describe the ASP.NET Core platform, other than as it relates to how Identity works.

I cover the authentication scenarios that are used by the majority of Internet-facing ASP.NET Core projects. I don't describe the ASP.NET Core authentication features that are not related directly to ASP.NET Core Identity, such as integration with Active Directory or with third-party products, such as IdentityServer (which, despite its name, is not directly related to ASP.NET Core Identity and is more of an alternative).

What Software Do I Need for the Examples?

You need the same set of tools you use for ASP.NET Core development, including the .NET SDK and Visual Studio or Visual Studio Code. I describe the setup required for this book in Chapter 2.

How Do I Set Up the Development Environment?

Chapter 2 introduces ASP.NET Core Identity by creating a simple application, and, as part of that process, I tell you how to create a development environment for following the examples in this book.

What If I Have Problems Following the Examples?

The first thing to do is to go back to the start of the chapter and begin over. Most problems are caused by accidentally skipping a step or not fully applying the changes shown in a listing. Pay close attention to the emphasis in code listings, which highlights the changes that are required.

Next, check the errata/corrections list, which is included in the book's GitHub repository. Technical books are complex, and mistakes are inevitable, despite my best efforts and those of my editors. Check the errata list for the list of known errors and instructions to resolve them.

If you still have problems, then download the project for the chapter you are reading from the book's GitHub repository, `https://github.com/Apress/pro-asp.net-core-identity`, and compare it to your project. I create the code for the GitHub repository by working through each chapter, so you should have the same files with the same contents in your project.

If you still can't get the examples working, then you can contact me at `adam@adam-freeman.com` for help. Please make it clear in your email which book you are reading and which chapter/example is causing the problem. A page number or code listing is always helpful. Please remember that I get a lot of emails and that I may not respond immediately.

There are examples in Chapters 11, 22, and 23 that rely on third-party services from Google, Facebook, and Twitter. I am unable to provide support for these examples because problems can only be diagnosed using your private account credentials. Even if you are willing to share your credentials, I am not willing to use them. If you have problems with these examples, you should raise a support query with the authentication service provider.

What If I Find an Error in the Book?

You can report errors to me by email at `adam@adam-freeman.com`, although I ask that you first check the errata/corrections list for this book, which you can find in the book's GitHub repository at `https://github.com/Apress/pro-asp.net-core-identity`, in case it has already been reported.

I list errors that are likely to confuse readers, especially problems with example code, in the errata/corrections file on the GitHub repository, with a grateful acknowledgment to the first reader who reported it. I keep a list of less serious issues, which usually means errors in the text surrounding examples, and I correct them when I write a new edition.

Are There Lots of Examples?

There are *loads* of examples. The best way to learn is by example, and I have packed as many of them as I can into this book. To help make the examples easier to follow, I have adopted a simple convention, which I follow whenever possible. When I create a new file, I list the complete contents, as shown in Listing 1-1. All code listings include the name of the file in the listing's header, along with the folder in which it can be found.

Listing 1-1. The Contents of the Delete.cshtml File in the Pages/Identity/Admin Folder

```
@page "{id?}"
@model IdentityApp.Pages.Identity.Admin.DeleteModel
@{
    ViewBag.Workflow = "Delete";
}

<div asp-validation-summary="All" class="text-danger m-2"></div>

<form method="post">

    <h3 class="bg-danger text-white text-center p-2">Caution</h3>

    <h5 class="text-center m-2">
        Delete @Model.IdentityUser.Email?
    </h5>
    <input type="hidden" name="id" value="@Model.IdentityUser.Id" />
    <div class="text-center p-2">
        <button type="submit" class="btn btn-danger">Delete</button>
        <a asp-page="Dashboard" class="btn btn-secondary">Cancel</a>
    </div>
</form>
```

This listing is taken from Chapter 9. Don't worry about what it does; just be aware that this is a complete listing, which shows the entire contents of the file, and the header tells you what the file is called and its location in the project.

When I make changes to the code, I show the altered statements in bold, as shown in Listing 1-2.

Listing 1-2. Disabling a Button in the Delete.cshtml File in the Pages/Identity/Admin Folder

```
@page "{id?}"
@model IdentityApp.Pages.Identity.Admin.DeleteModel
@inject Microsoft.Extensions.Configuration.IConfiguration Configuration
@{
    ViewBag.Workflow = "Delete";
    string dashboardUser = Configuration["Dashboard:User"] ?? "admin@example.com";
}
```

```
<div asp-validation-summary="All" class="text-danger m-2"></div>

<form method="post">
    <h3 class="bg-danger text-white text-center p-2">Caution</h3>
    <h5 class="text-center m-2">
        Delete @Model.IdentityUser.Email?
    </h5>
    <input type="hidden" name="id" value="@Model.IdentityUser.Id" />
    <div class="text-center p-2">
        <button type="submit" class="btn btn-danger"
            disabled="@(Model.IdentityUser.Email == dashboardUser)">
                Delete
        </button>
        <a asp-page="Dashboard" class="btn btn-secondary">Cancel</a>
    </div>
</form>
```

This listing is taken from a later example, which requires changes to the file created in Listing 1-1. To help you make follow the example, the changes are marked in bold.

Some examples require a small change to a large file. So that I don't waste space listing the unchanged parts of the file, I just show the region that changes, as shown in Listing 1-3. You can tell this listing shows only part of a file because it starts and ends with an ellipsis (...).

Listing 1-3. Counting Users in the Dashboard.cshtml.cs File in the Pages/Identity/Admin Folder

```
...
public void OnGet() {
    UsersCount = UserManager.Users.Count();
    UsersUnconfirmed = UserManager.Users
        .Where(u => !u.EmailConfirmed).Count();
    UsersLockedout = UserManager.Users
        .Where(u => u.LockoutEnabled && u.LockoutEnd > System.DateTimeOffset.Now)
        .Count();
    UsersTwoFactor = UserManager.Users.Where(u => u.TwoFactorEnabled).Count();
}
...
```

In some cases, I need to make changes to different parts of the same file, in which case I omit some elements or statements for brevity, as shown in Listing 1-4. This listing adds new `using` statements and defines additional methods to an existing file, much of which is unchanged, and which has been omitted from the listing.

Listing 1-4. Supporting External Services in the SignIn.cshtml.cs File in the Pages/Identity Folder

```
using System.ComponentModel.DataAnnotations;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;
using SignInResult = Microsoft.AspNetCore.Identity.SignInResult;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Authentication;
using System.Net;
```

```
namespace IdentityApp.Pages.Identity {

    [AllowAnonymous]
    public class SignInModel : UserPageModel {

        // ...methods and properties omitted for brevity...

        public IActionResult OnPostExternalAsync(string provider) {
            string callbackUrl = Url.Page("SignIn", "Callback", new { ReturnUrl });
            AuthenticationProperties props =
               SignInManager.ConfigureExternalAuthenticationProperties(
                   provider, callbackUrl);
            return new ChallengeResult(provider, props);
        }

        public async Task<IActionResult> OnGetCallbackAsync() {
            ExternalLoginInfo info = await SignInManager.GetExternalLoginInfoAsync();
            SignInResult result = await SignInManager.ExternalLoginSignInAsync(
                info.LoginProvider, info.ProviderKey, true);
            if (result.Succeeded) {
                return Redirect(WebUtility.UrlDecode(ReturnUrl ?? "/"));
            } else if (result.IsLockedOut) {
                TempData["message"] = "Account Locked";
            } else if (result.IsNotAllowed) {
                TempData["message"] = "Sign In Not Allowed";
            } else {
                TempData["message"] = "Sign In Failed";
            }
            return RedirectToPage();
        }
    }
}
```

This convention lets me pack in more examples, but it does mean it can be hard to locate a specific technique. To this end, the chapters in this book begin with a summary table that describes the techniques it contains, and many of the chapters in Part 1 contain quick reference tables that list the methods used to implement a specific feature.

Where Can You Get the Example Code?

You can download the example projects for all the chapters in this book from https://github.com/Apress/pro-asp.net-core-identity.

How Do I Contact the Author?

You can email me at adam@adam-freeman.com. It has been a few years since I first published an email address in my books. I wasn't entirely sure it was a good idea, but I am glad I did it. I have received emails from around the world, from readers working or studying in every industry, and—for the most part, anyway—the emails are positive, polite, and a pleasure to receive.

I try to reply promptly, but I get many emails, and sometimes I get a backlog, especially when I have my head down trying to finish writing a book. I always try to help readers who are stuck with an example in the book, although I ask that you follow the steps described earlier in this chapter before contacting me.

While I welcome reader emails, there are some common questions for which the answers will always be "no." I am afraid that I won't write the code for your new startup, help you with your college assignment, get involved in your development team's design dispute, or teach you how to program.

What If I Really Enjoyed This Book?

Please email me at `adam@adam-freeman.com` and let me know. It is always a delight to hear from a happy reader, and I appreciate the time it takes to send those emails. Writing these books can be difficult, and those emails provide essential motivation to persist at an activity that can sometimes feel impossible.

What If This Book Has Made Me Angry and I Want to Complain?

You can still email me at `adam@adam-freeman.com`, and I will still try to help you. Bear in mind that I can only help if you explain what the problem is and what you would like me to do about it. You should understand that sometimes the only outcome is to accept I am not the writer for you and that we will have closure only when you return this book and select another. I'll give careful thought to whatever has upset you, but after 25 years of writing books, I have come to accept that not everyone enjoys reading the books I like to write.

Summary

In this chapter, I outlined the content and structure of this book. The best way to learn ASP.NET Core Identity is by example, so in the next chapter, I jump right in and show you how to set up your development environment and use it to create your first ASP.NET Core Identity project.