**BCIT**

**Comp 4952 HCI for Application Development**

**Technical Programming Option**

**Option Head Mirela Gutica**

**Fall 2022**

**Implement Authentication and Authorization**

1. **Introduction**

So far, users had free access to a web application. We will discuss next how to authorize users for limited access to all or parts of a web application.

Authentication is the process of validating the identity of a user.

Authorization is the process of granting access to a user.

ASP.NET (all flavors Windows Forms, MVC, MVC Core, etc.) is provided with three types of authentication:

* Windows-based authentication
  + Causes a login dialog box
  + Supported by most browsers
  + Configured through the IIS management console
  + Uses Windows accounts
* Individual user authentication
  + Developers add a login page
  + Encrypts the username and password
  + Doesn’t use the Windows user account
* Third-party authentication (e.g., Facebook, Google, Twitter, etc.)
  + Provided by third-party services
  + Allows users to use their existing accounts
  + Can use two-factor authentication

ASP.NET Core Identity is an API that supports user interface (UI) login functionality.

Manages users, passwords, profile data, roles, claims, tokens, email confirmation, and more. Users can create an account with the login information stored in Identity or they can use an external login provider. Supported external login providers include Third-party authentication.

Readings: <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio>

1. **Authentication Cookie**

Authentication is using an “authentication cookie” in the request of the page. If the cookie is not found, the application is redirected in the browser to the login page (or de desired page, based on coding). After the user introduces the user name and password, the request is sent to the server. If the server finds the user in the DB, it will create an authentication cookie and send it back to the user. The user can decide that the authentication cookie to be stored permanently on his/hers machine (persistent cookie).

Important: the “authentication cookie” is a session cookie.

1. **Apply the Authorization**

Authorization can apply to a **controller**, or **to specific action method**, and it can identify a **role**.

Important: **Authorization can work even if you are not using the Identity package**.

Make sure that you include the namespace:

using Microsoft.AspNetCore.Authorization;

Example 1: Authorization for everyone:

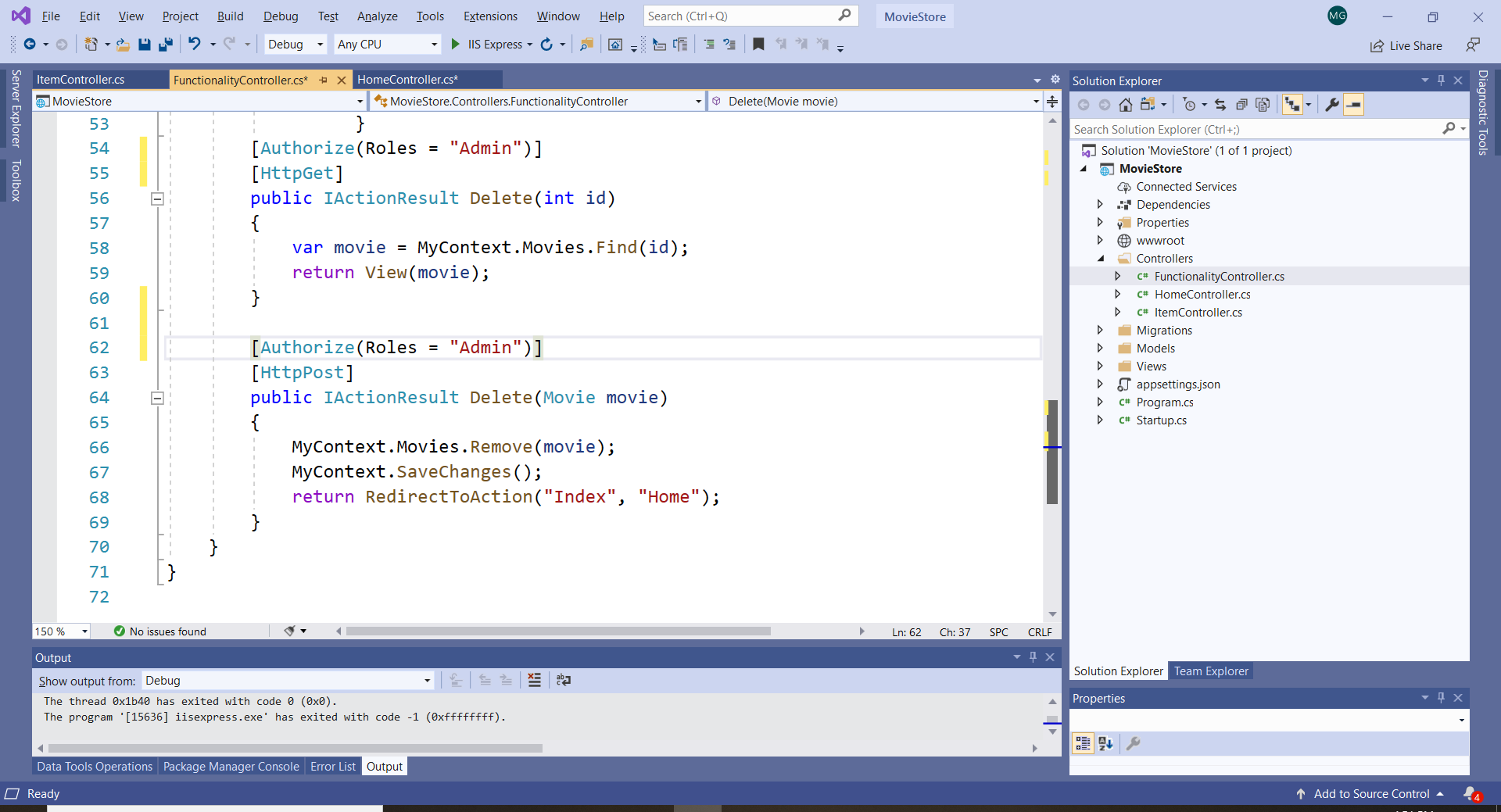
[Authorize]

public class HomeController : Controller

Example 2:

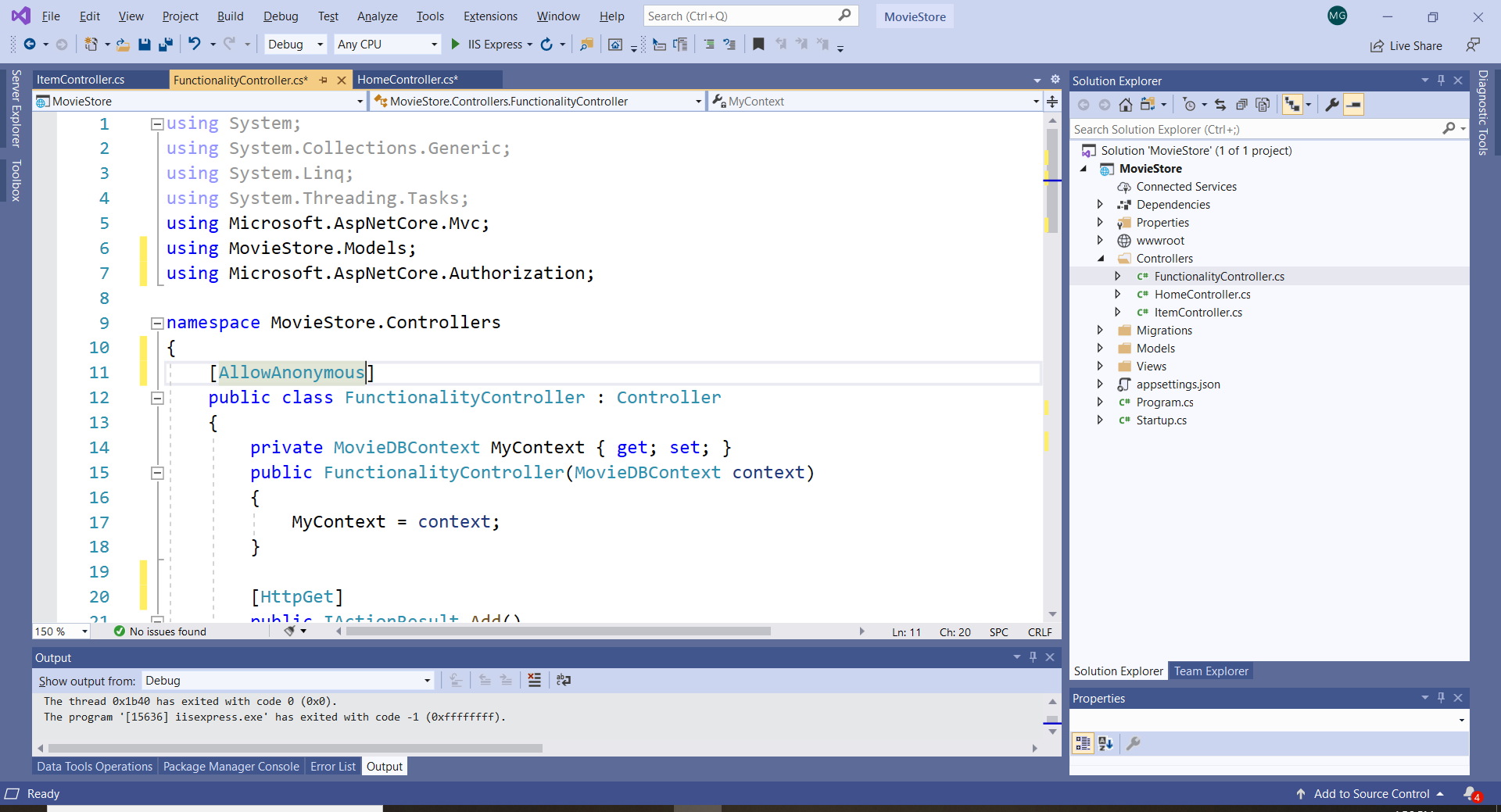
Here the whole controller is authorized to admin only:

Example 3: Here, only an admin and delete movies:



Example 4:

Here Anonymous access is allowed:



1. **Working with Identity**

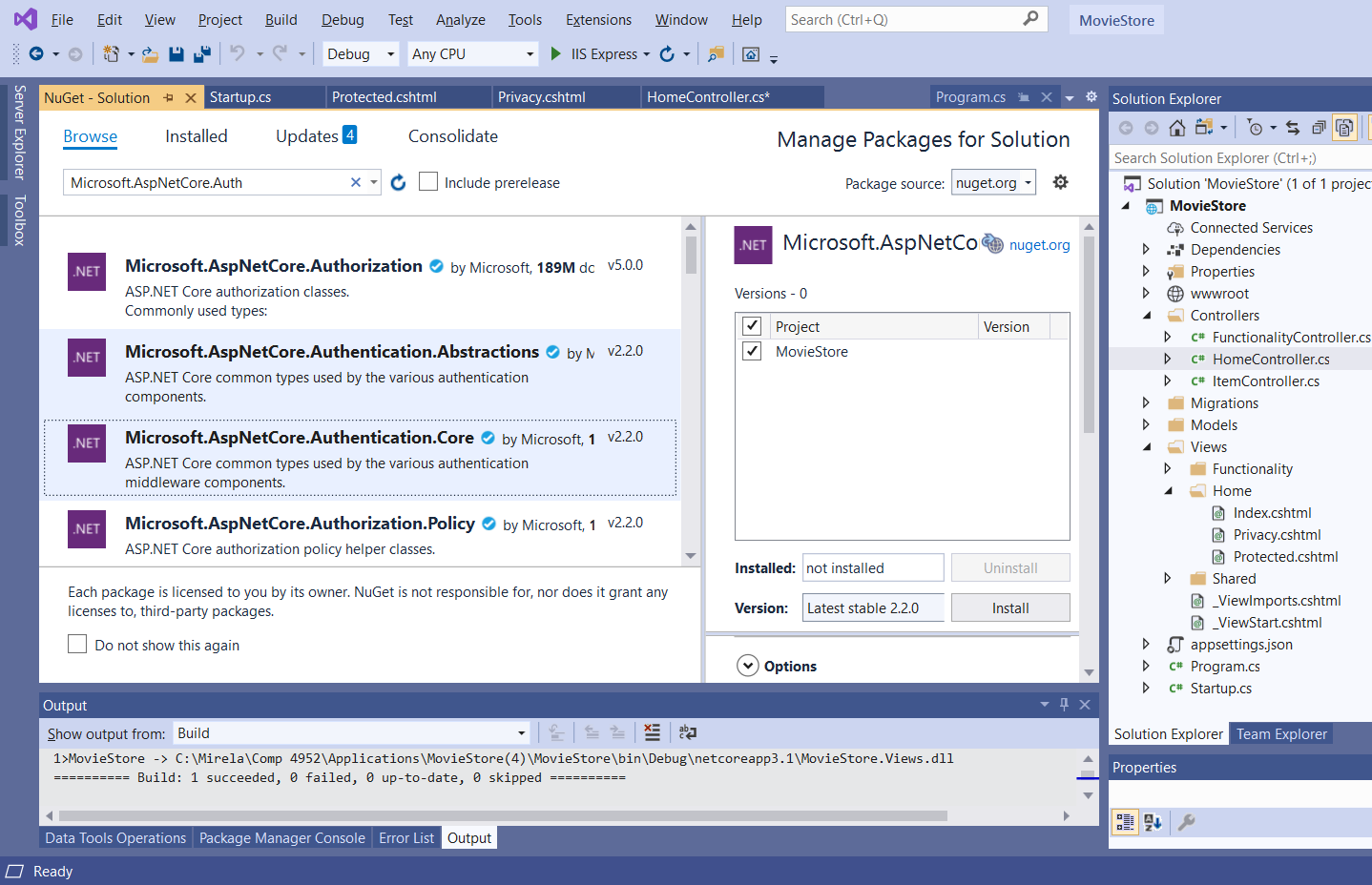
The User Identity class has all properties that you need for authentication: UserName, Email, Password, ConfirmPassword, PhoneNumber, PhoneNumberConfirmed. When you want to work with Identity, add the Package Identity (use NuGet Package Manager and the same steps as in the previous assignment: Implement an Entity Framework Code-First ASP.NET Core MVC Application). Make sure that the version is compatible with your application’s version (read the documentation). Errors occur if there are different versions. The Identity package is used to create user classes and generate user tables in the DB.

Read: <https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-5.0&tabs=visual-studio>

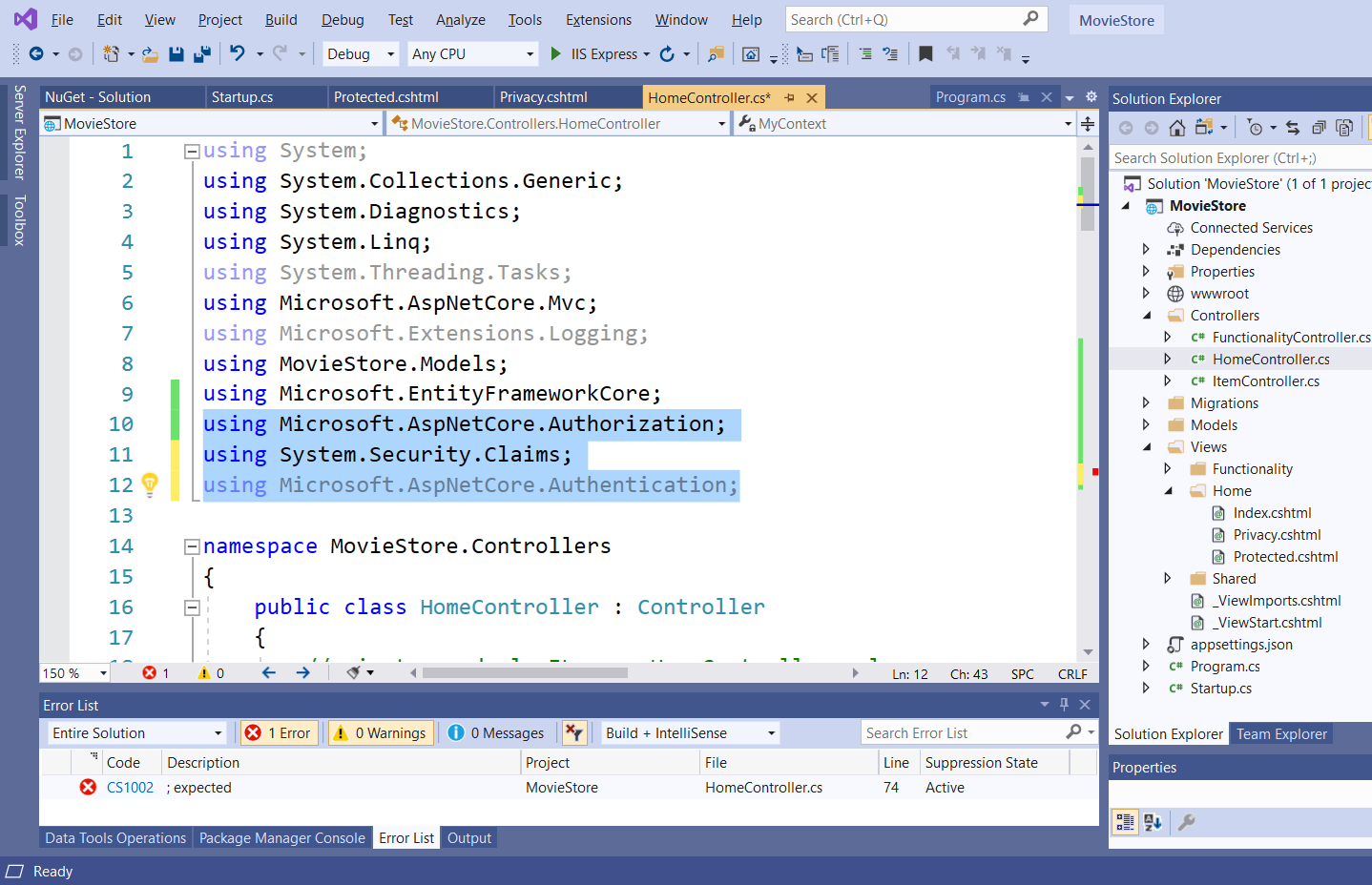
**Important:** You do not need the Identify Package for this assignment!

1. **Authentication Basics in ASP.NET Core**

Install the Microsoft packages that are highlighted below:

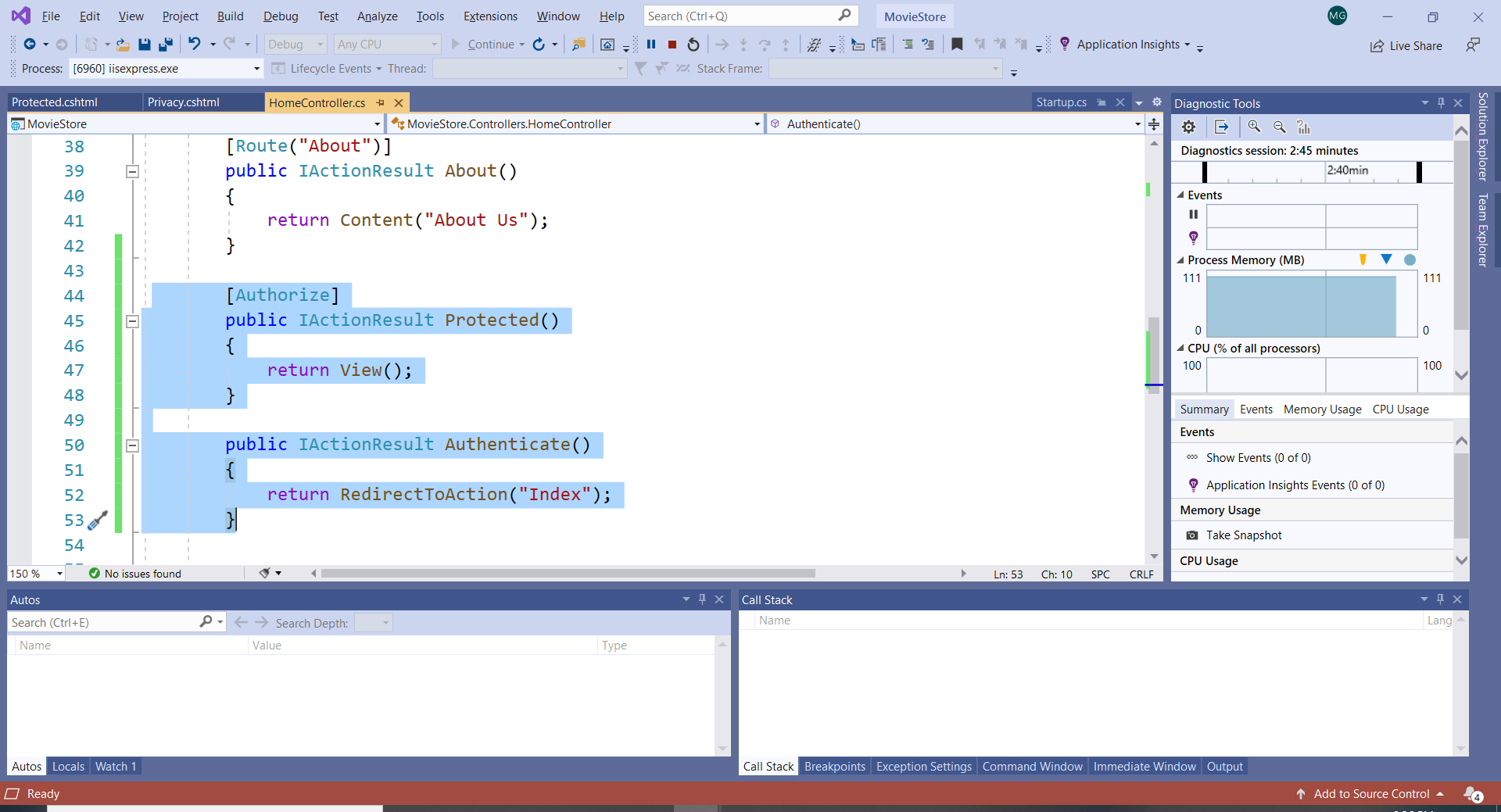


1. In the Home Controller you need to add the following name spaces:



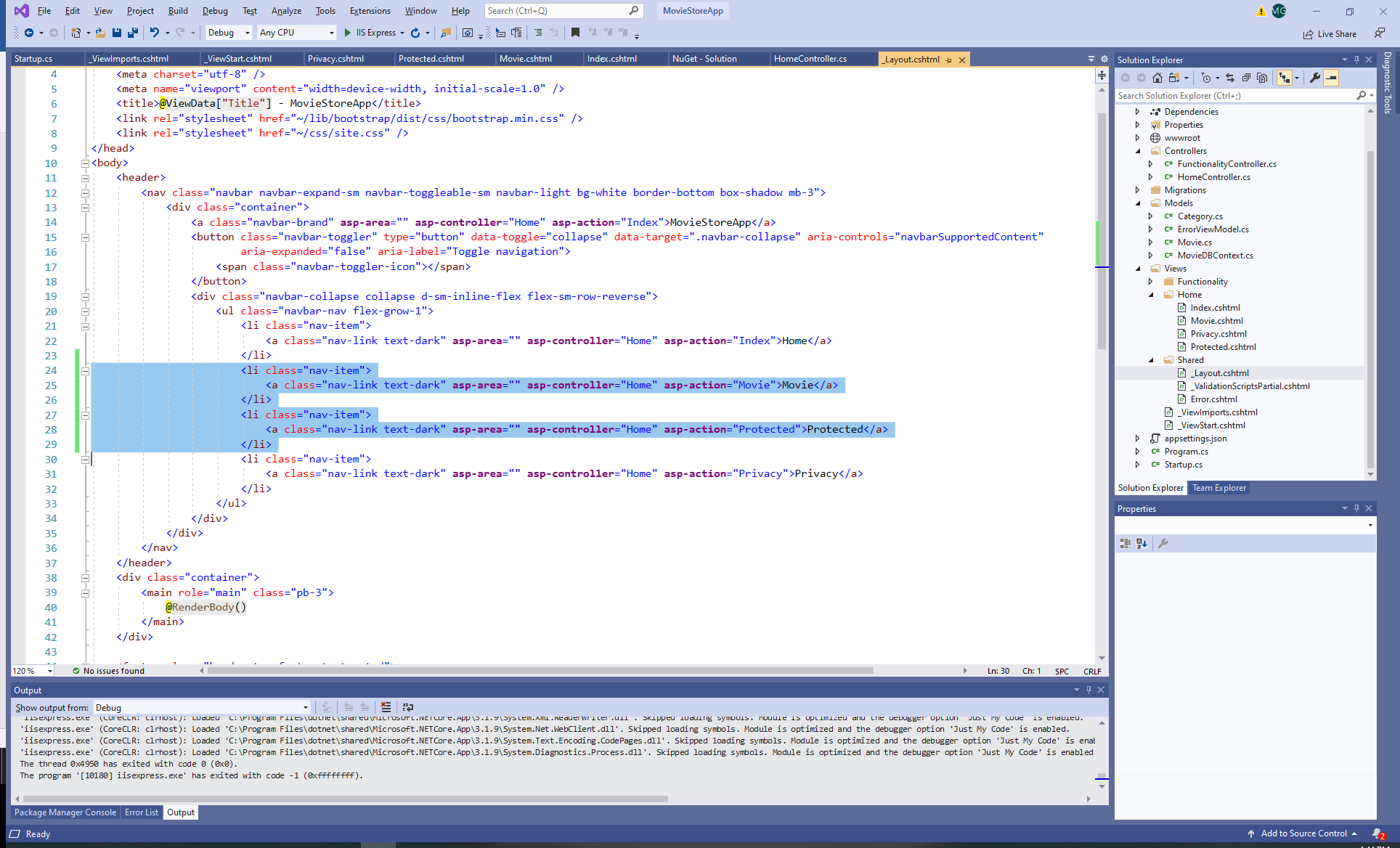
We will discuss System.Security.Claims later.

1. Create two more action methods in the Home Controller and a simple View() for the “Protected” controller :

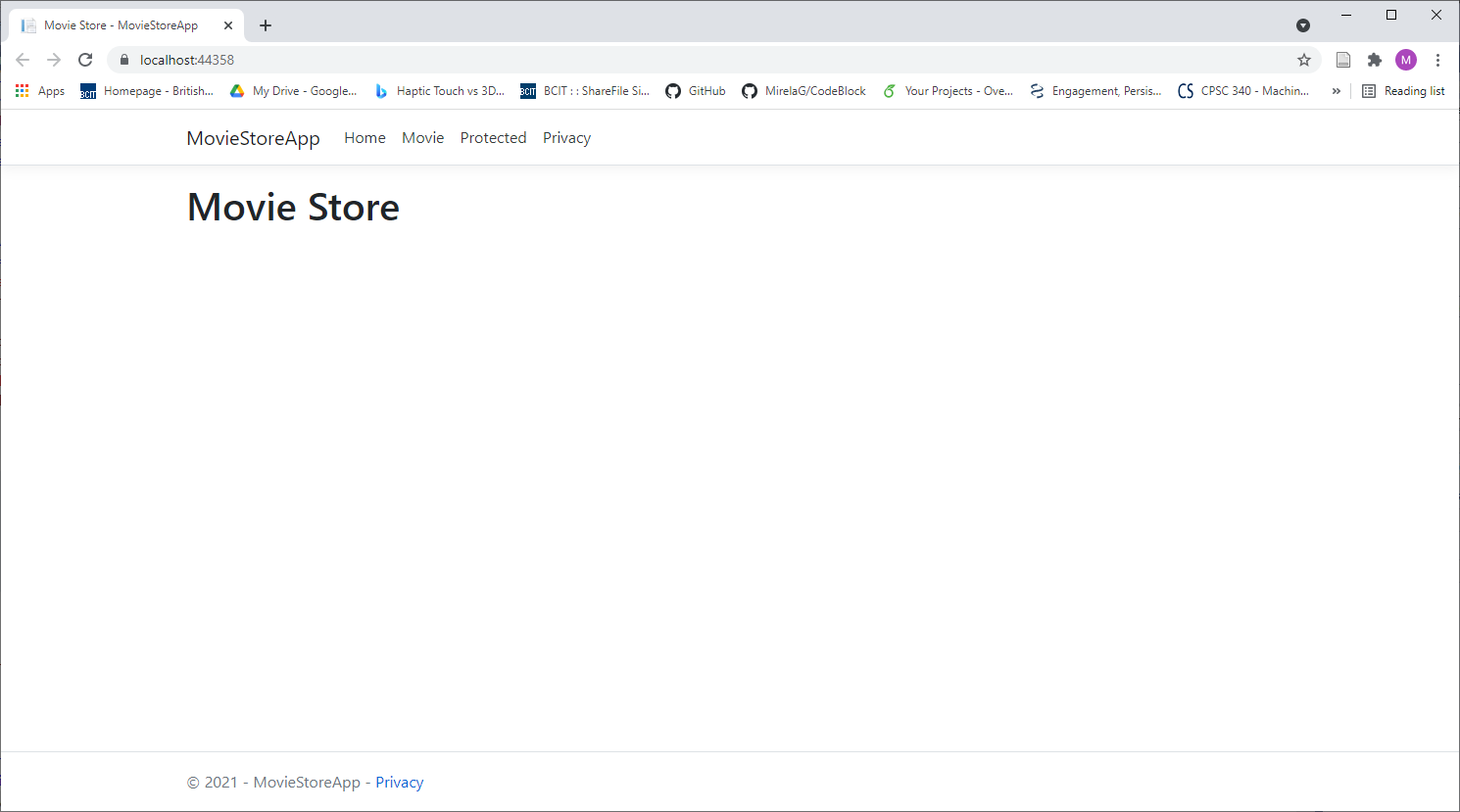


1. Create a new Movie Controller and copy the functionality from the Home Controller into the Movie Controller. Update the Home controller to return only a View(). Update the Views for Home and Movies accordingly.

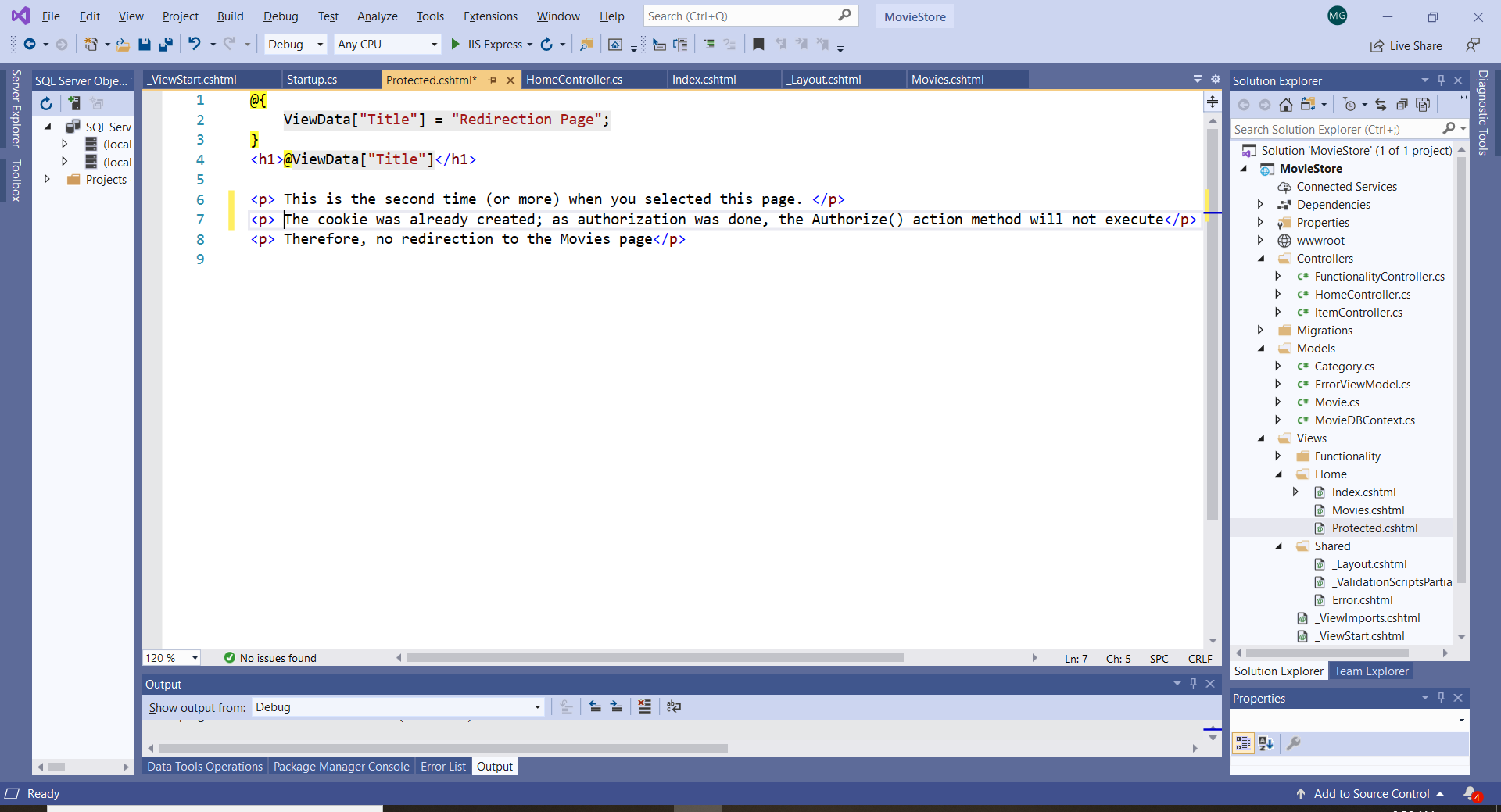




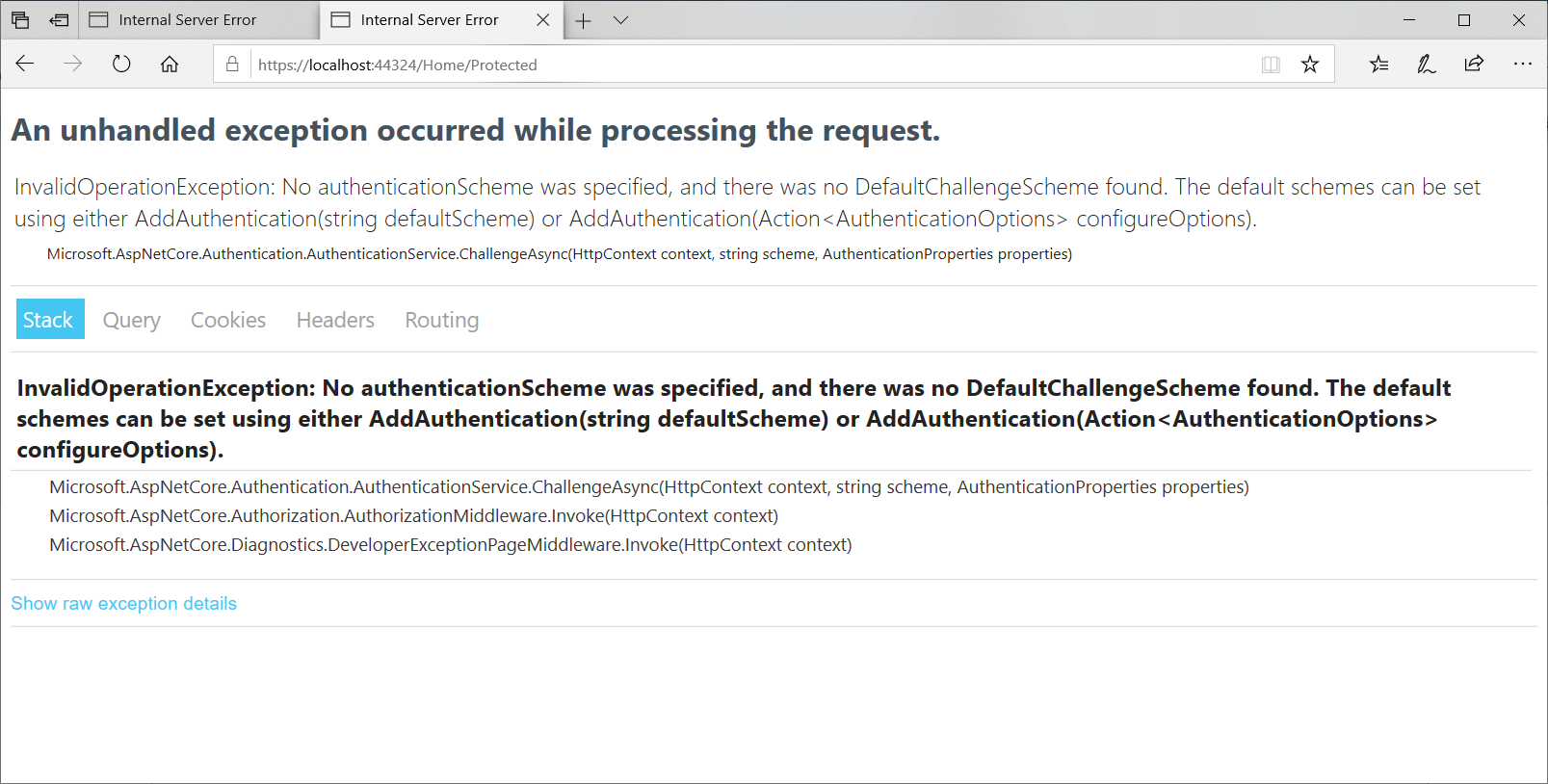
The Home Controller now looks like this:



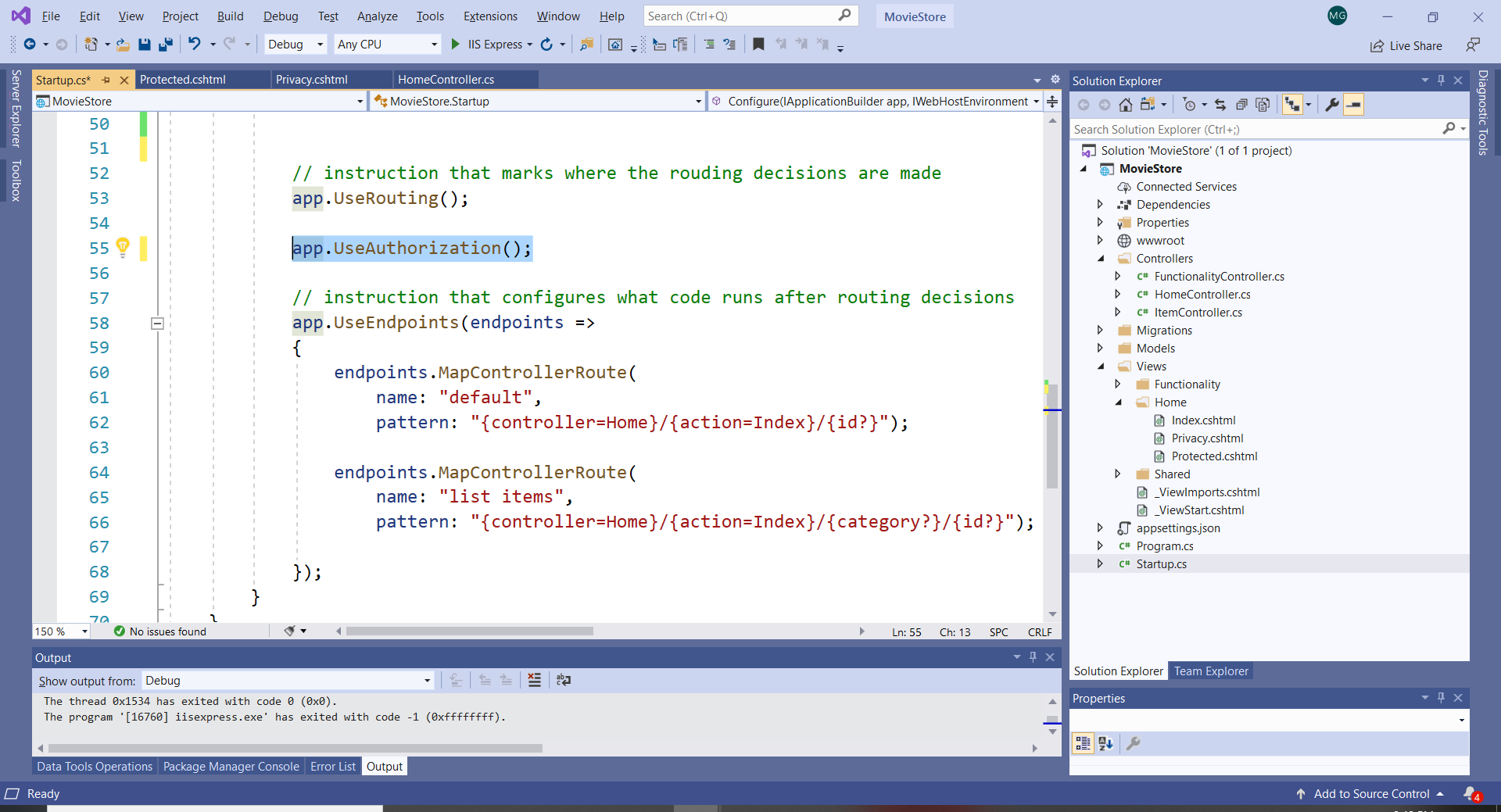
1. Code the View for Protected:



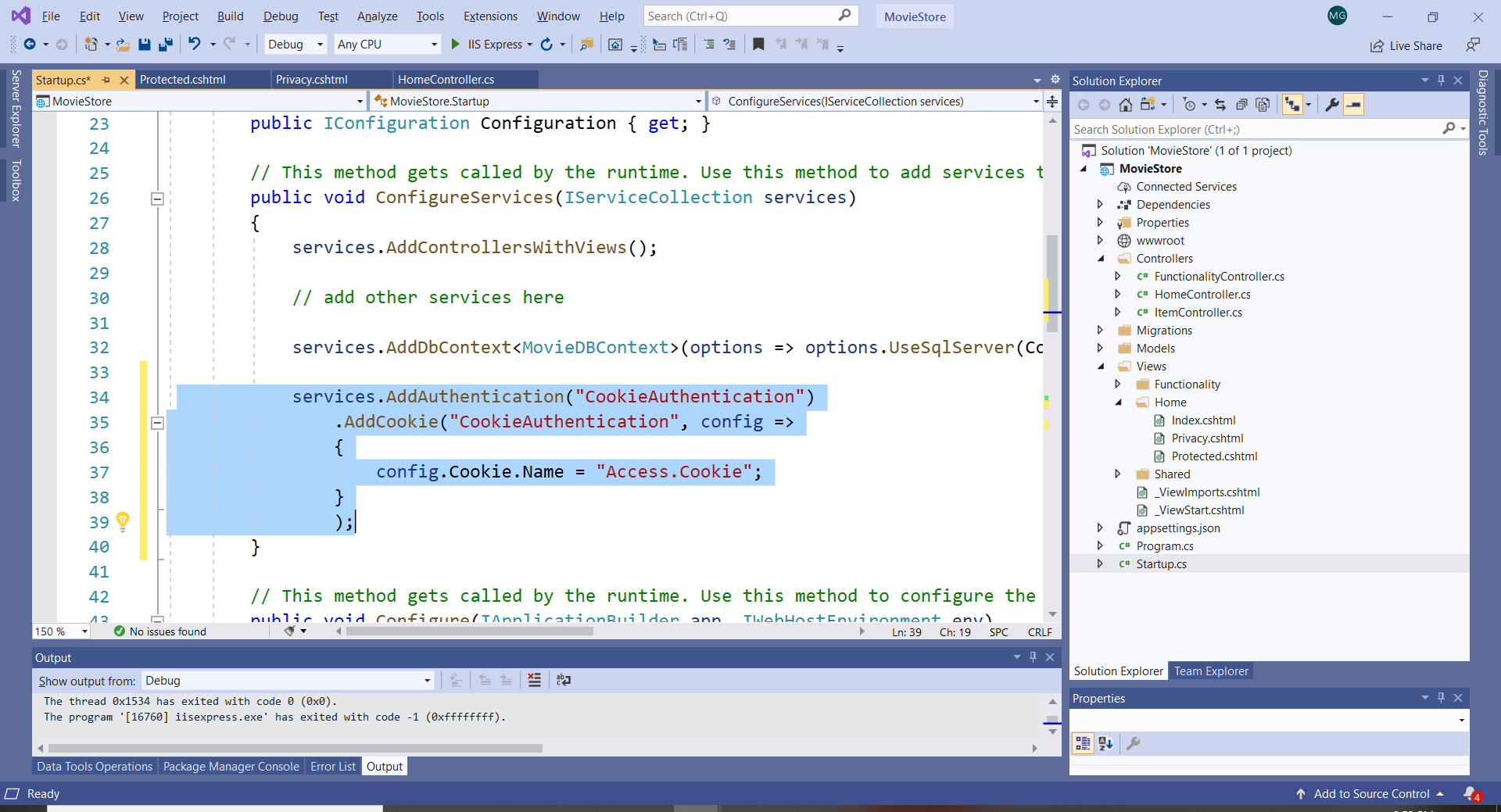
1. Try to access the page Protected. If we want to access now the Protected View: <https://localhost:44324/Home/Protected>, we get an error. This is because the action method has the attribute [Authorize].



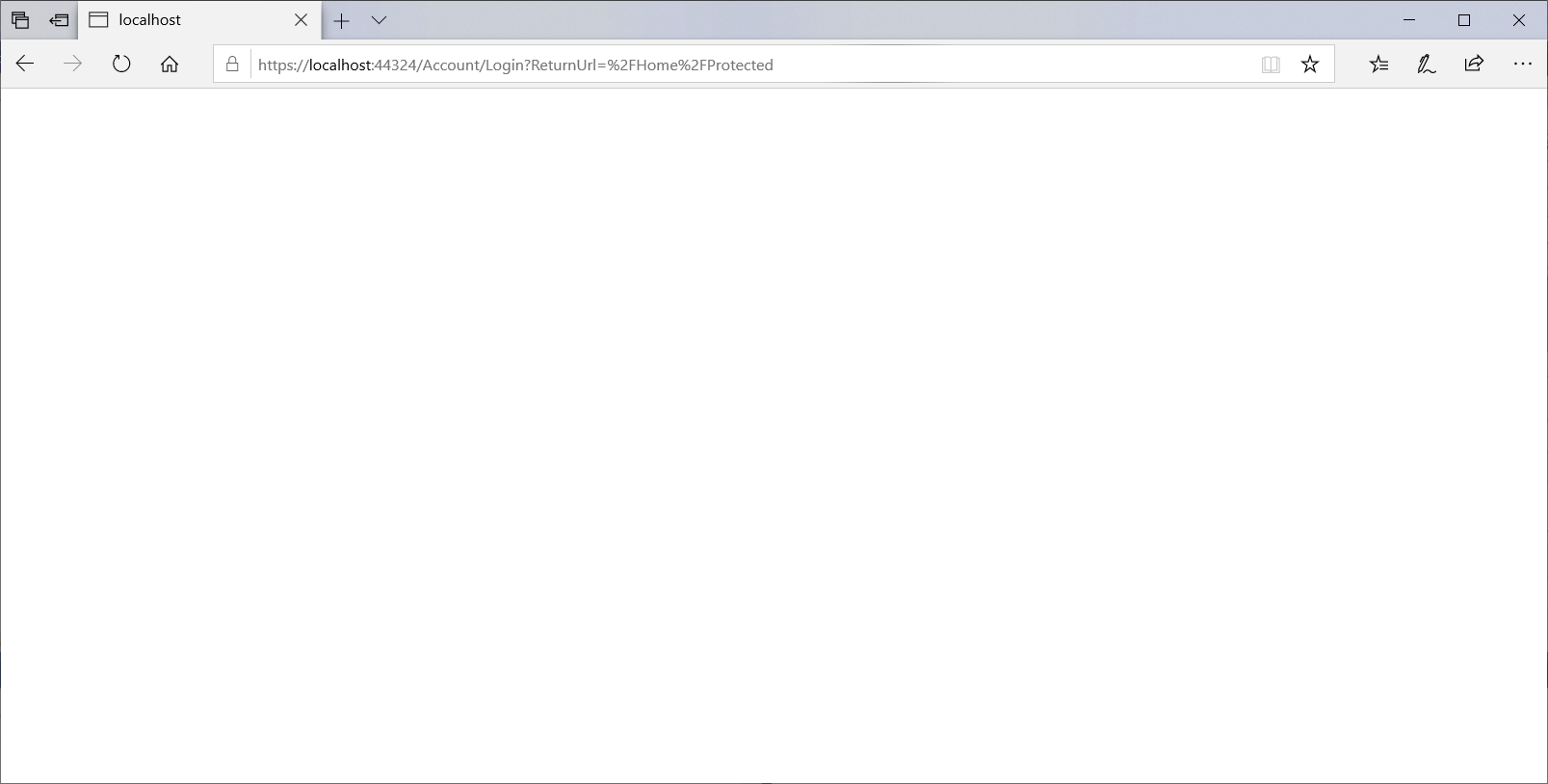
1. In the Startup.cs file we already have the statement app.UseAuthorization(); This statement indicate that authorization should happen after routing. This part of code (including routing) represents the **middleware execution** of an ASP.NET Core MVC application.



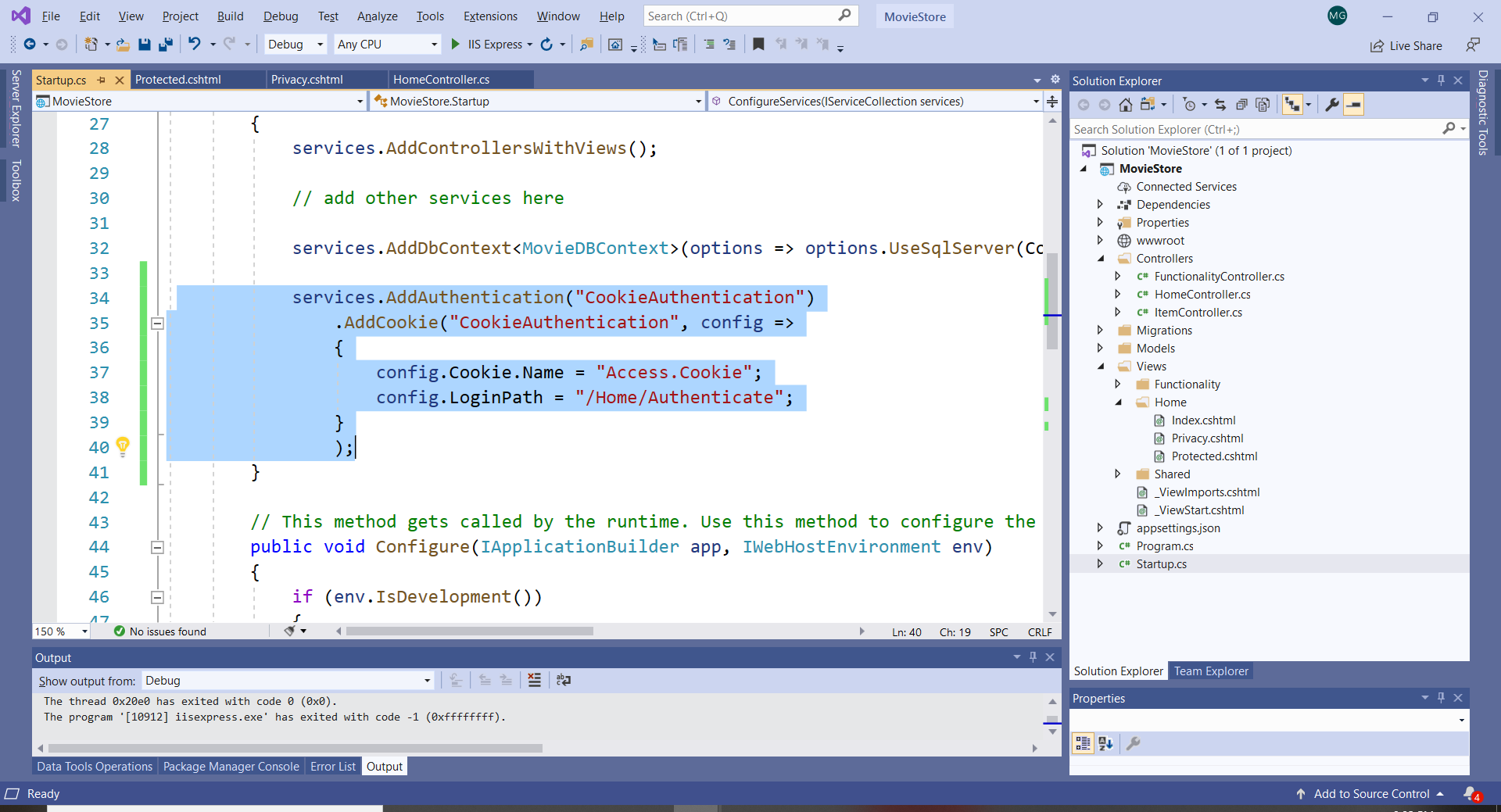
1. In Startup.cs file in ConfigureServices() add a session Cookie that will be used for authentication:

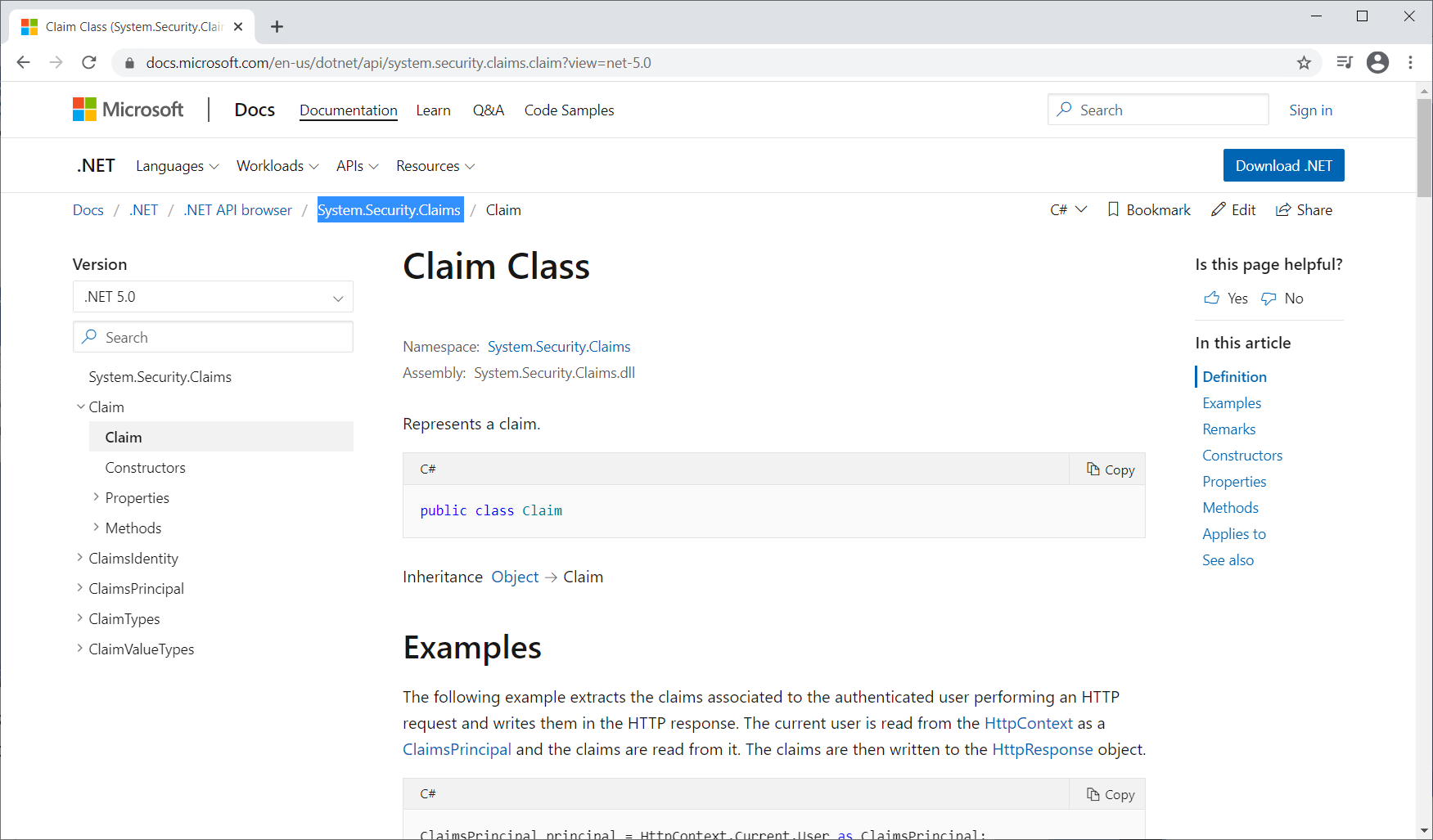


Now, run again and try to access <https://localhost:44324/Home/Protected>. There is no error; but there is no redirection specified.



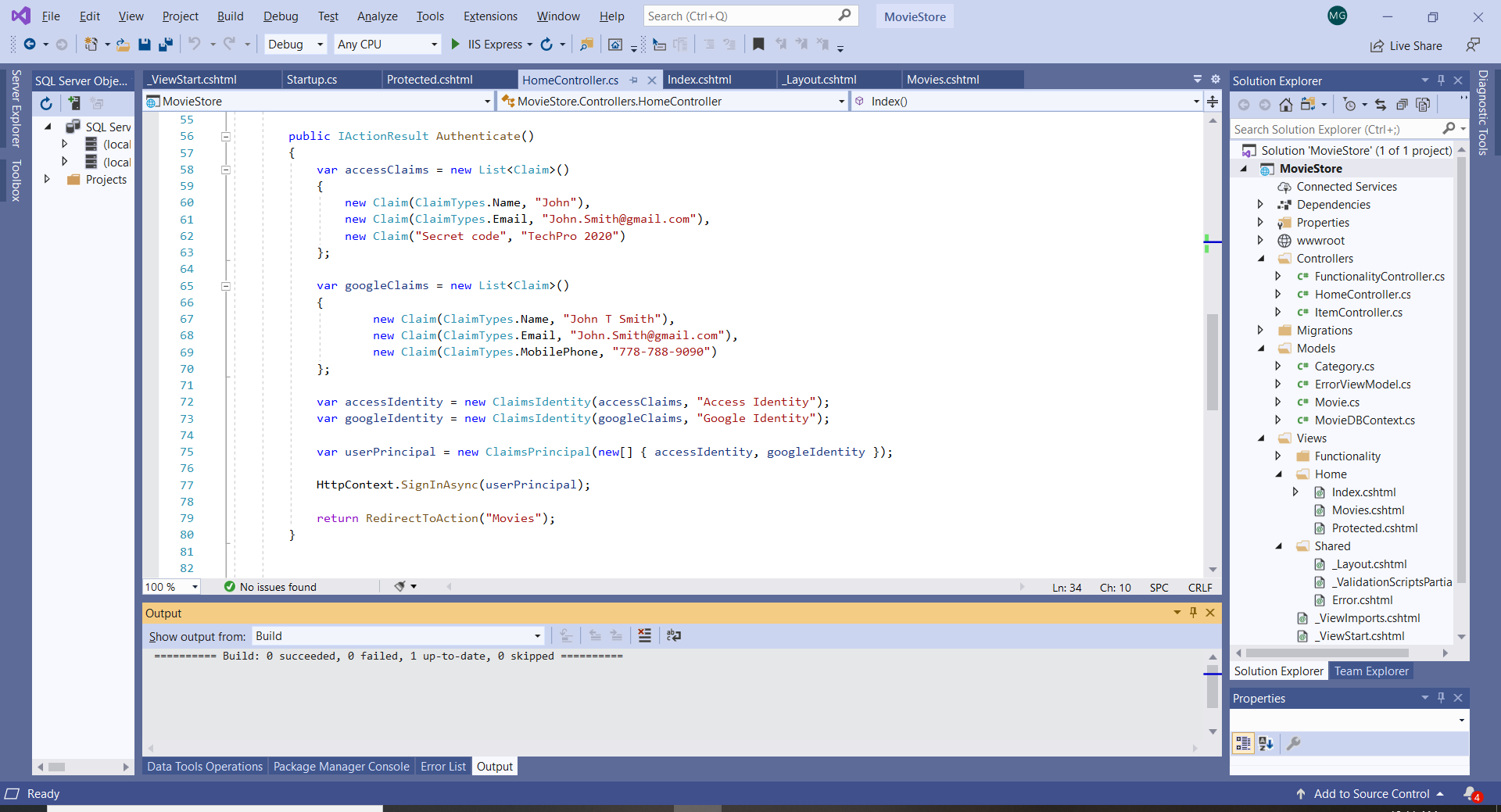
Let’s add the Login Path to include the Authenticate action method:



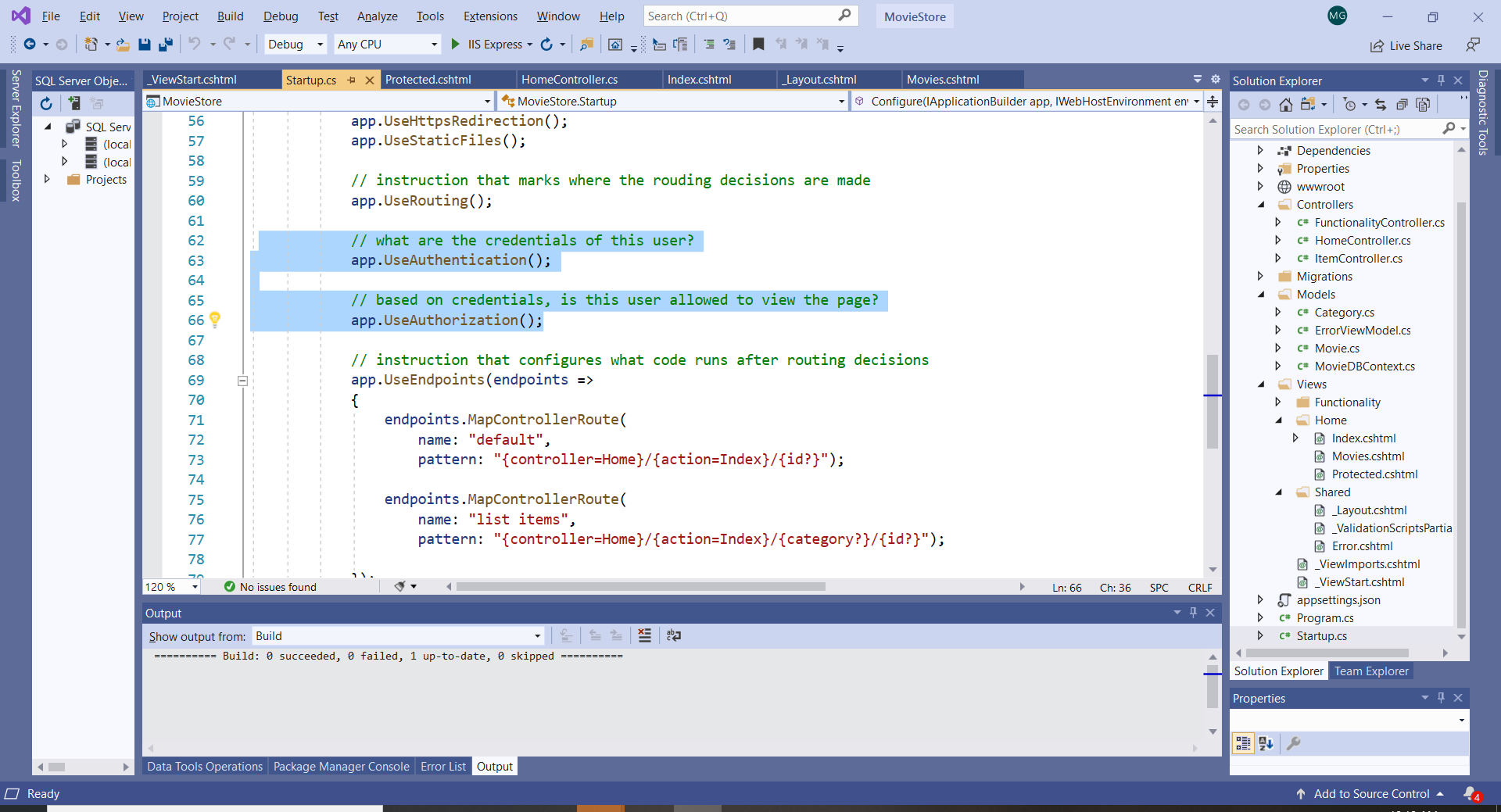
1. Run again. The action that will be executed is Authenticate() that currently redirects us to the Home page.
2. In C# there is a Claim class in the name space System.Security.Claims. Add the namespace to the Home Controller (if you did not do it already). This class works with HttpContext (HTTP request and response) and is used for authentication.
3. We will add manually specific claims, similar with how the Identity framework operates.

In the Home Controller write the following code in the Authenticate() action method.

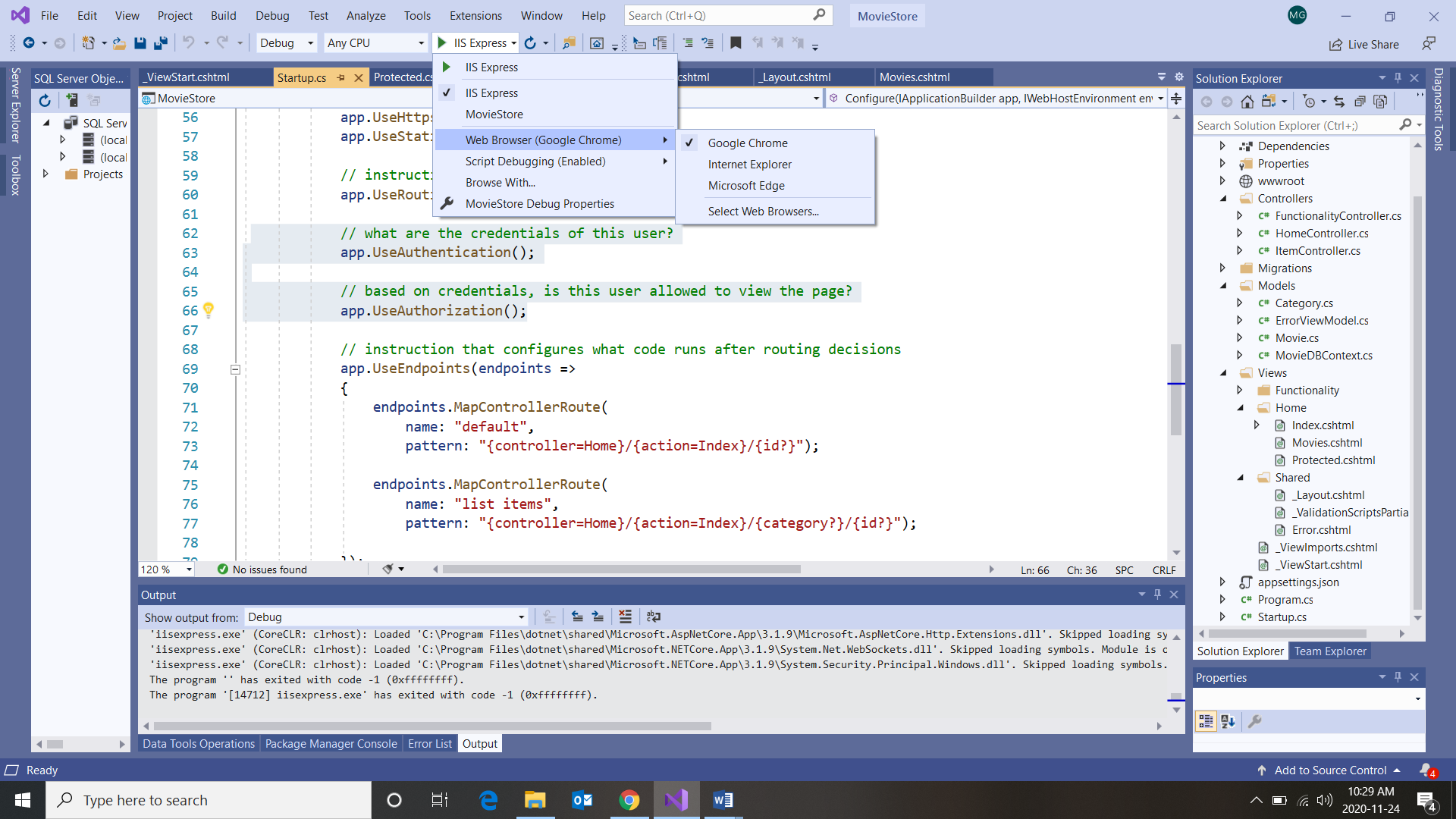
1. Two claims are created and they are used to create a cookie that allows the user to login.
2. The first claim consists of name, e-mail address, and a special string; and the second one is similar to Google authentication.
3. The HttpContext response to the server will contain the authentication cookie. In reality, we do not write this code manually, it is created from the DB Identity.
4. The user can authenticate with any of the two claims.
5. Note that the page is redirected now to Movies.



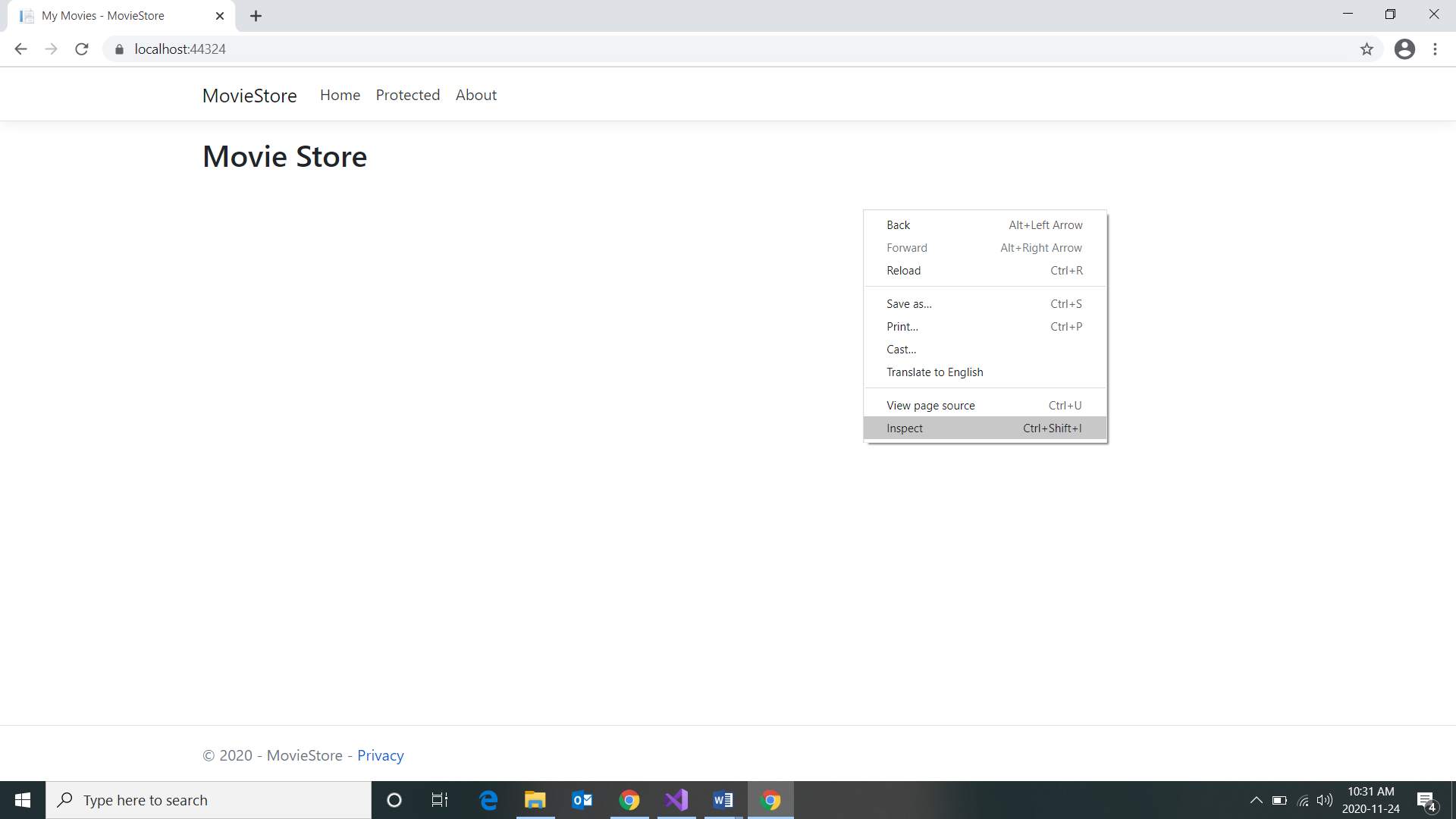
1. Now we need to update the middleware. Add Authentication() before Authorization() to the Startup.cs:



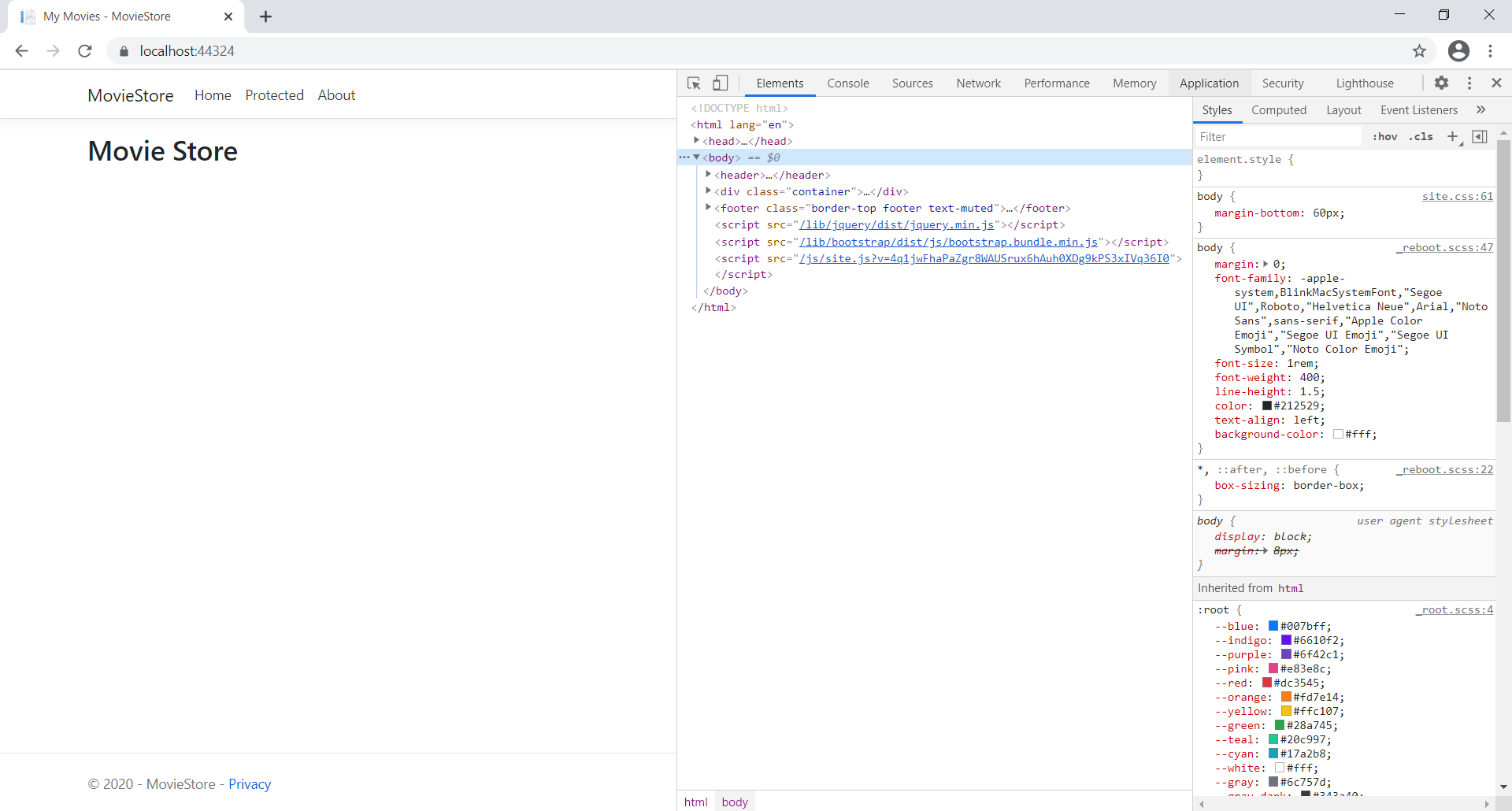
1. Before running the application, select Google Chrome as the target web browser.



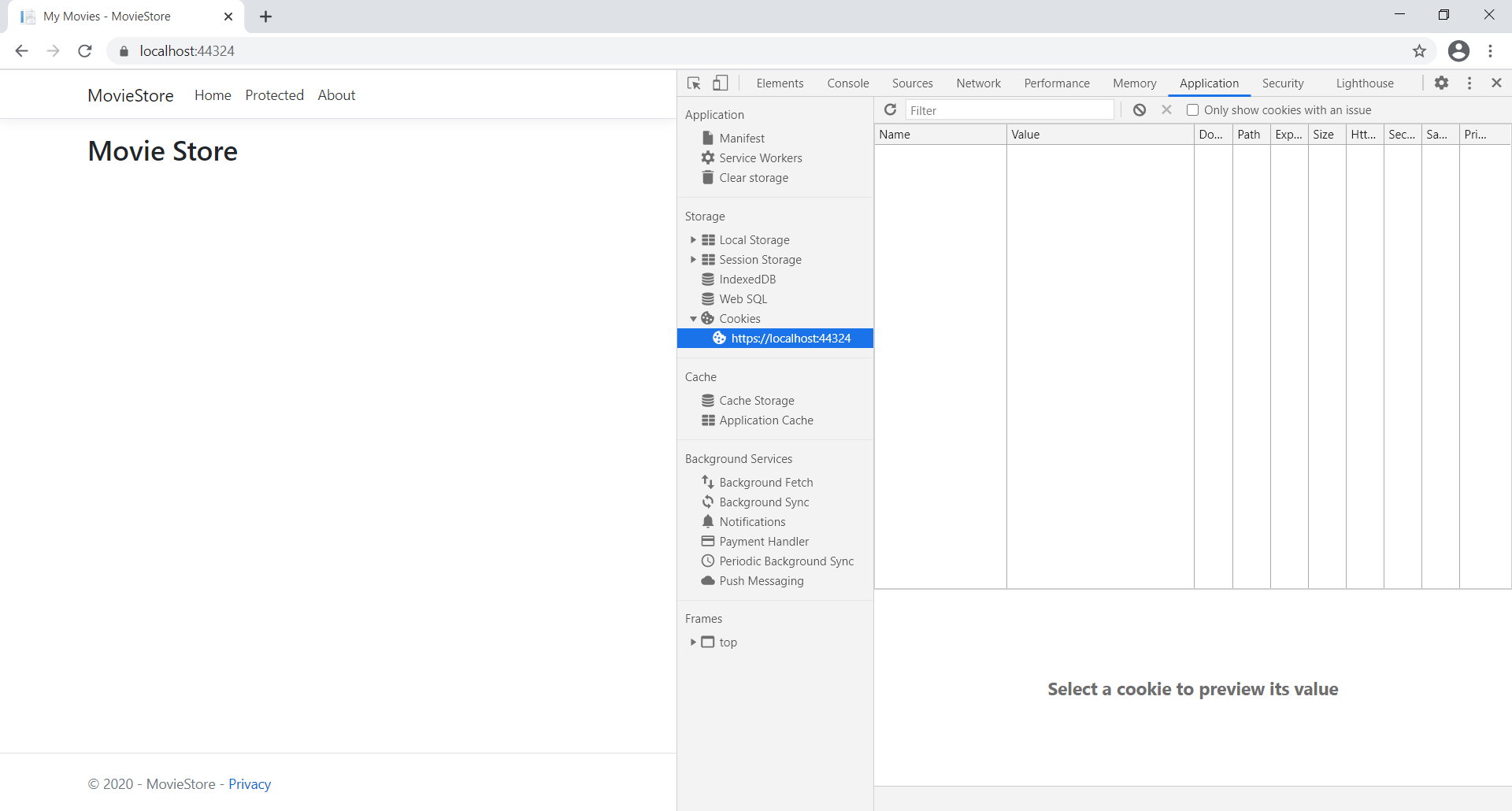
1. Run the application. Right click and select Inspect:



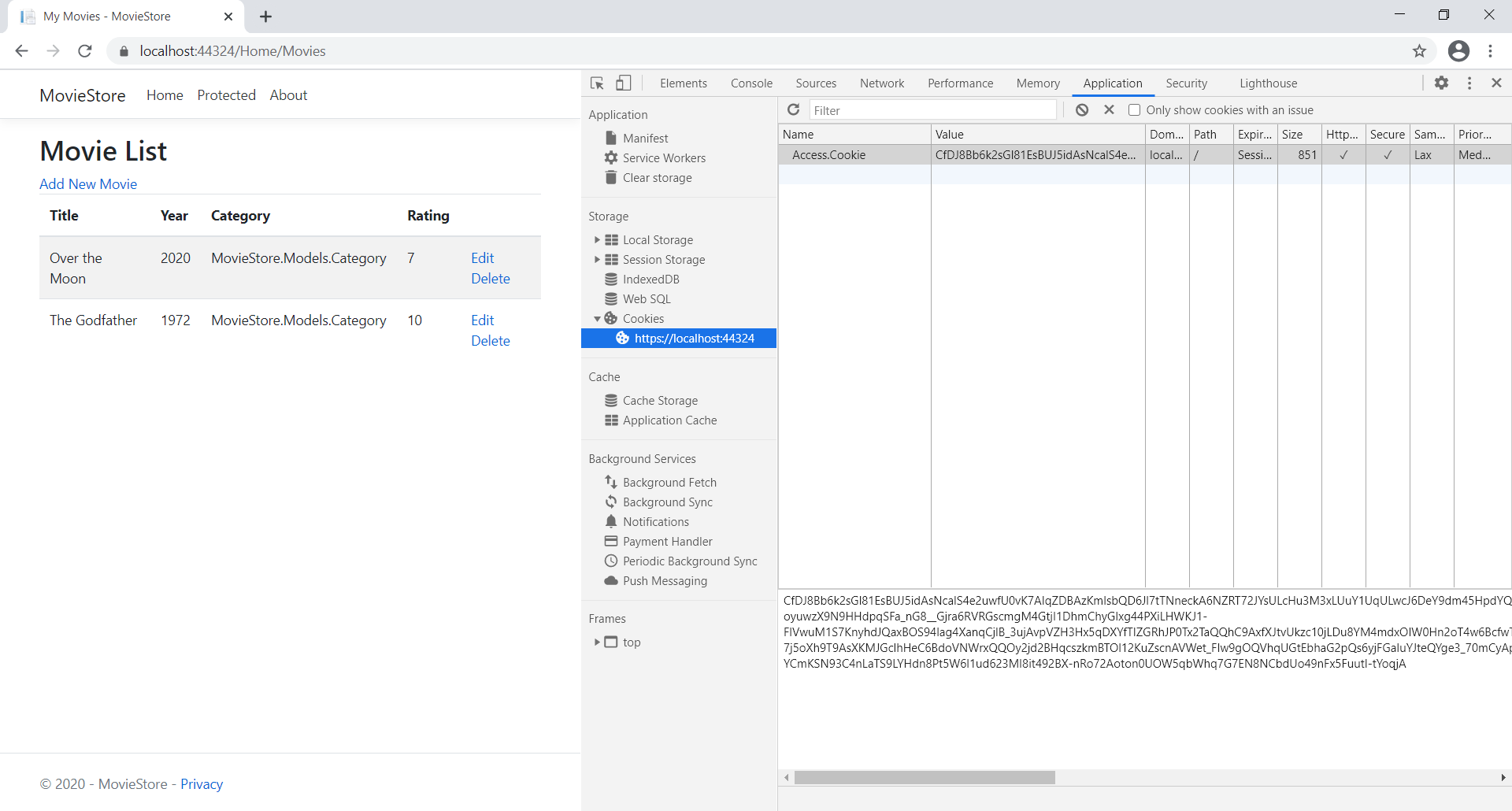
1. Select Application in the inspection window.



1. Select Cookies. Notice that there is no Cookie:



1. Now select Protected in the application. It will take us the to Movies page. Also notice that the AccessCookie was created:



1. Click again on Protected. This time the view associated with the “Protected” action Items is displayed.

