**BCIT**

**Comp 4952 HCI for Application Development**

**Technical Programming Option**

**Option Head Mirela Gutica**

**Fall 2022**

**Understanding Controllers and Routing in ASP.NET Core MVC Applications**

1. Readings:

<https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-5.0>

<https://docs.microsoft.com/en-us/aspnet/core/mvc/controllers/actions?view=aspnetcore-5.0>

<https://learn.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-6.0>

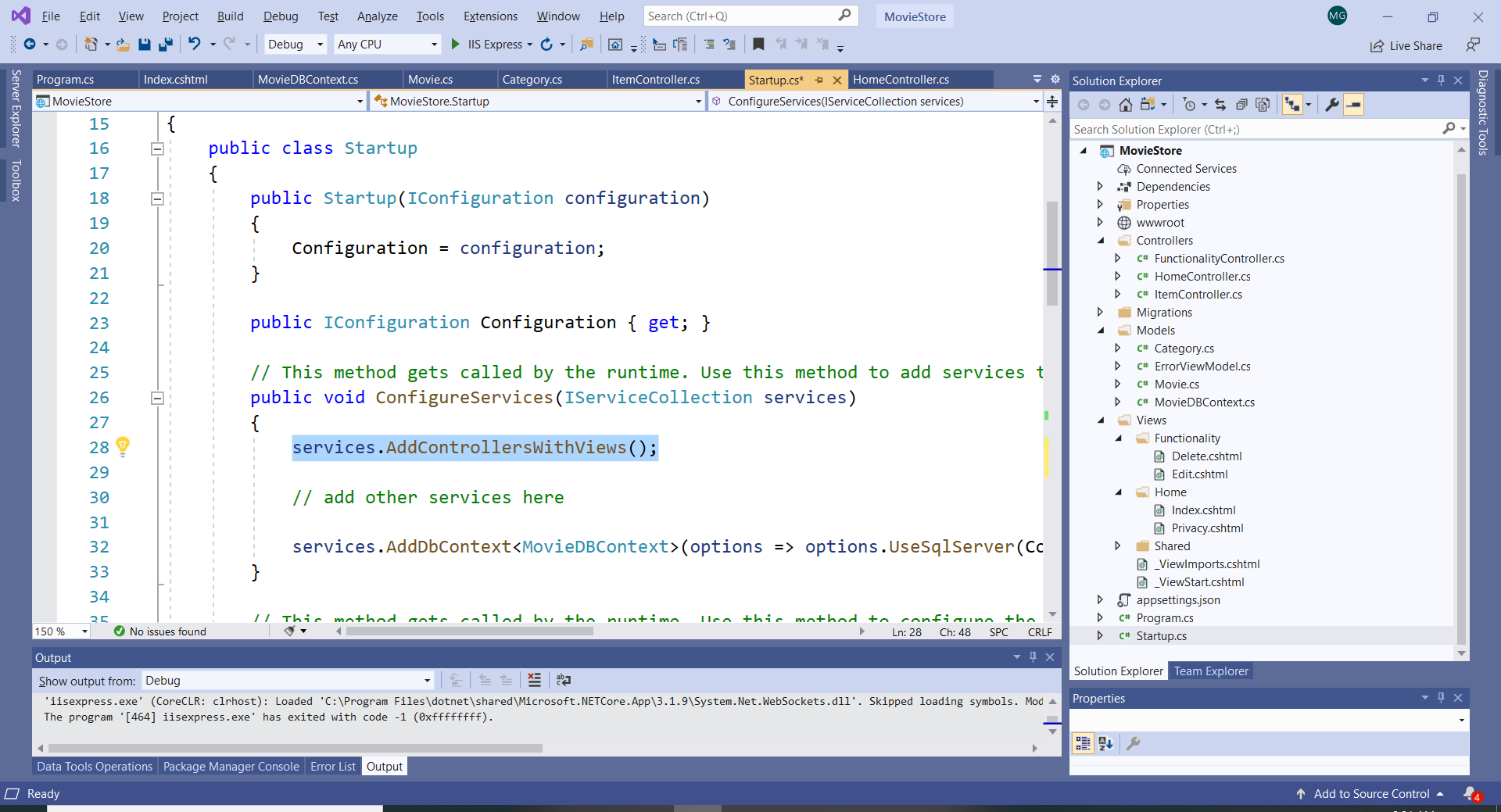
1. **Configuring the default route**

Endpoint Routing – routing system introduced in **ASP.NET Core MVC 2.2 and up;** this is the current recommended routing model.

1. Need to add routing services to your MVC application.

(Note: Previous ASP.NET MVC frameworks worked with AddMvc() method that added some services that were not needed.) **ASP.NET Core MVC 2.2 and up** uses the method AddControllersWithViews().

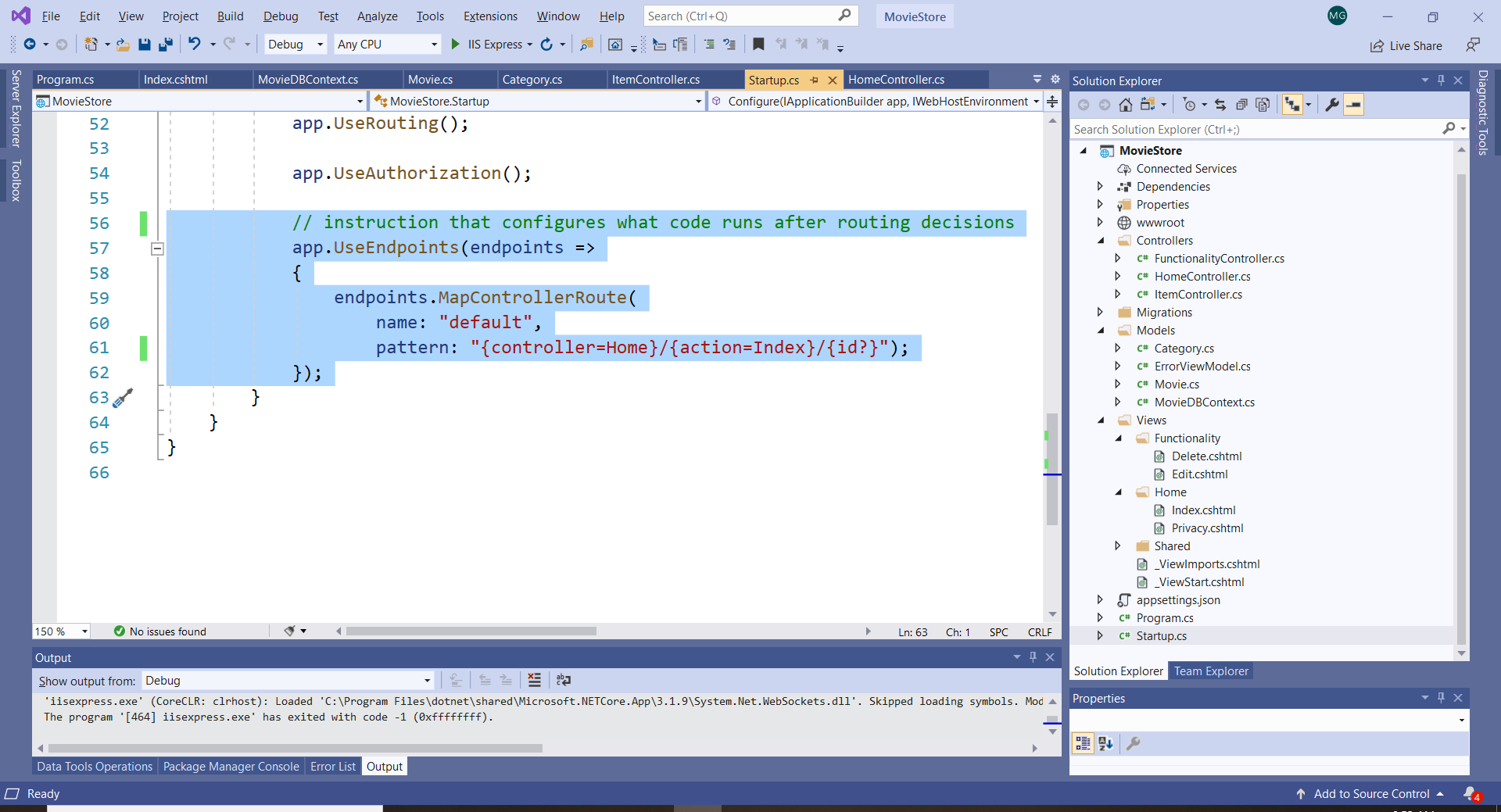
In the Startup.cs file, in the ConfigureServices() method, call AddControllersWithViews():



1. **Configuring the Routing**

In the Startup.cs file, the Configure() method marks where routing starts and what code in the middleware pipeline runs after the routing decision. This code can be updated (we will update this code later).

The code indicates that routing is directed to the Home controller.



Another was to indicate the routing to the Home controller is to specify in the endpoints section:

endpoints.MapDefaultControllerRoute();

1. **URL Format**

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

In ASP.NET Core MVC applications the URL is specified by 3 segments: the first segment is the controller; the second segment is the method in that particular controller and the third segment is the ID parameter of the action method. The “?” indicates that the ID is optional.

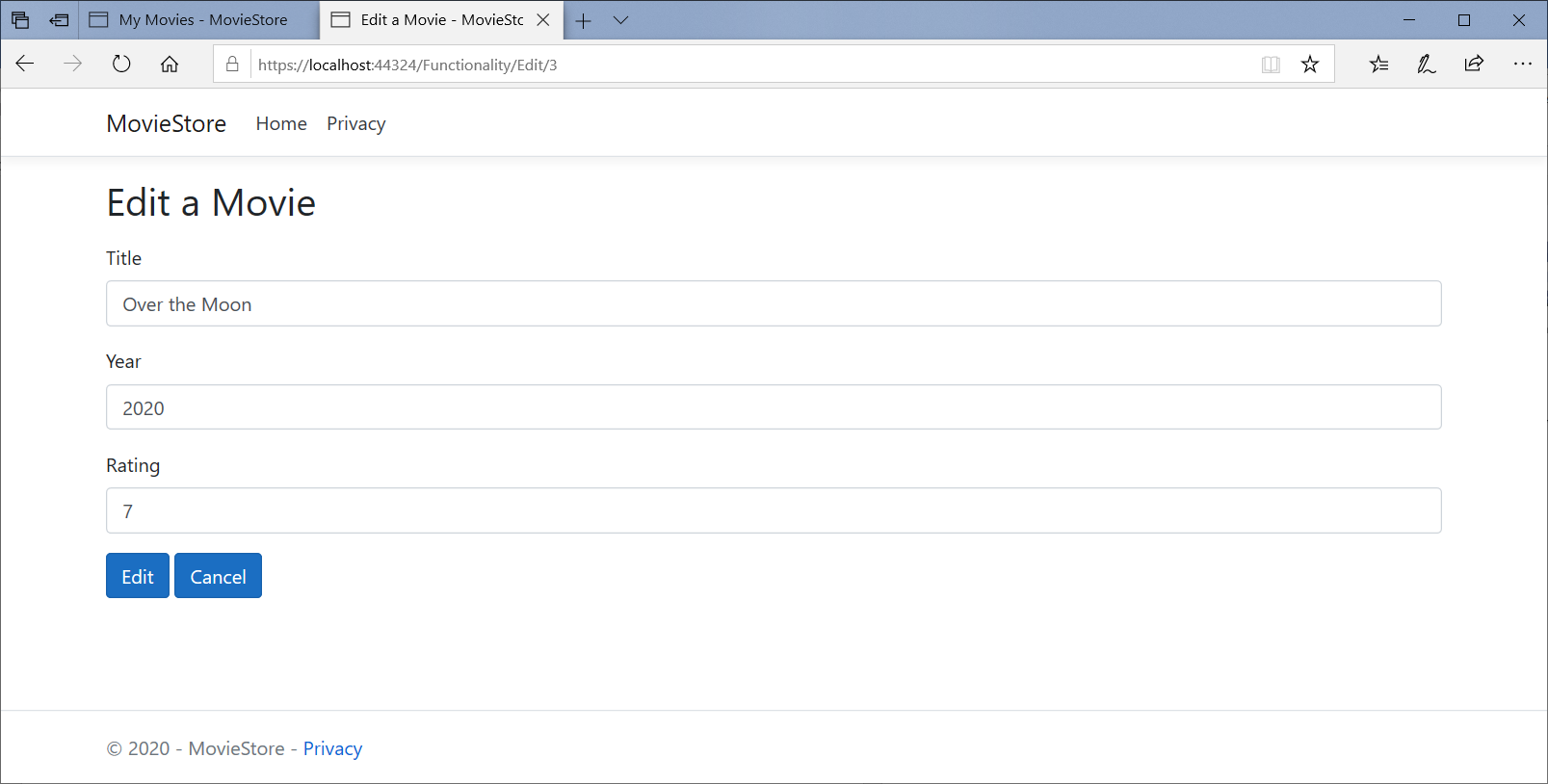
You can add more segments as we will discuss next.

The default route maps a request to an action method within a controller and can optionally pass a parameter.

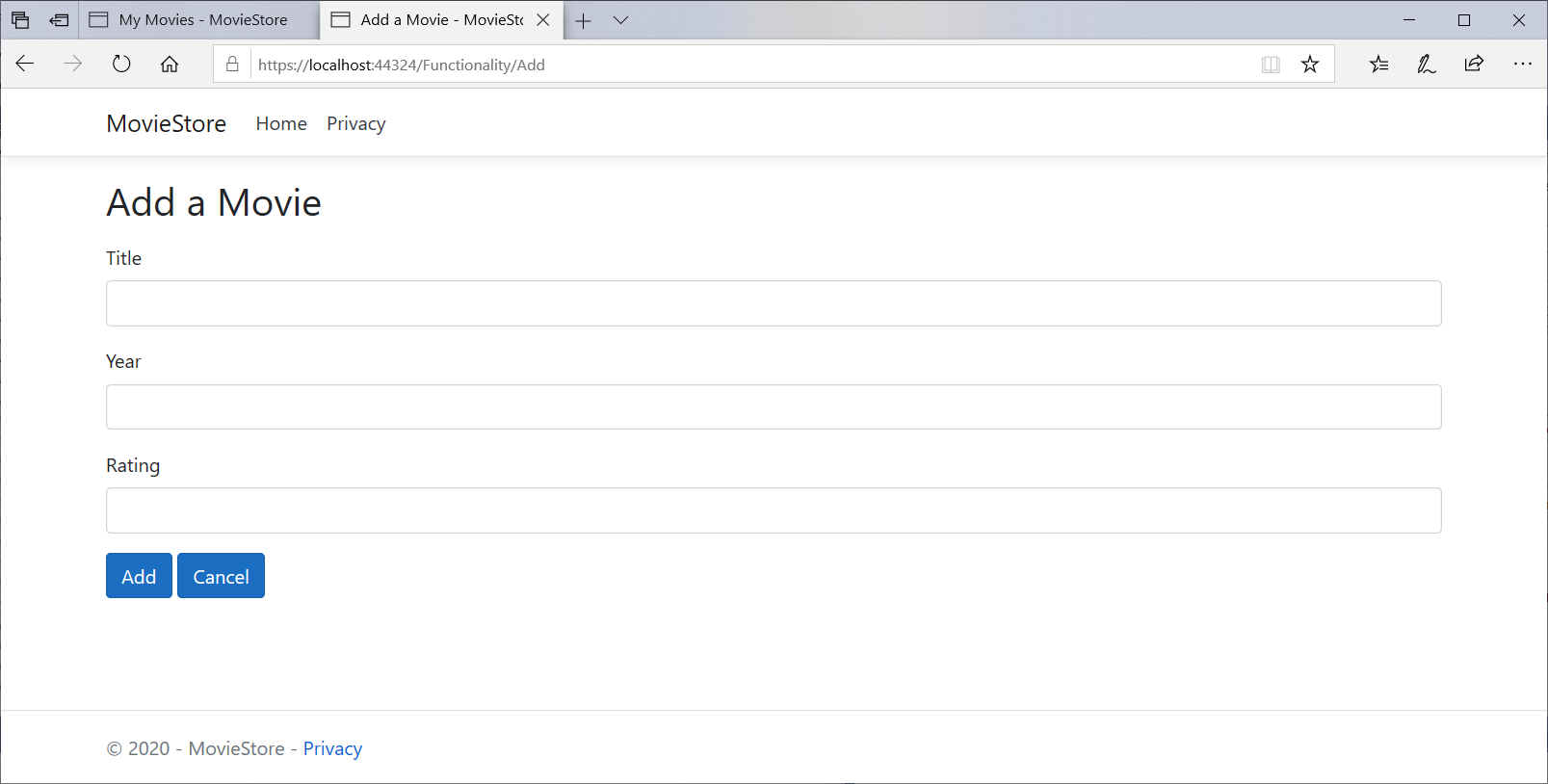
URL: “{controller}/{action}/{ID}

For example: the Movie “Over the Moon” had the ID = 3

<https://localhost:44324/Functionality/Edit/3>



“Add a New Movie” method doesn’t have an ID (doesn’t have a parameter):



1. **How Does a Controller Work?**

Let’s use a an ASP.NET Core MVC method called Content(). <https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.content?view=aspnetcore-5.0>

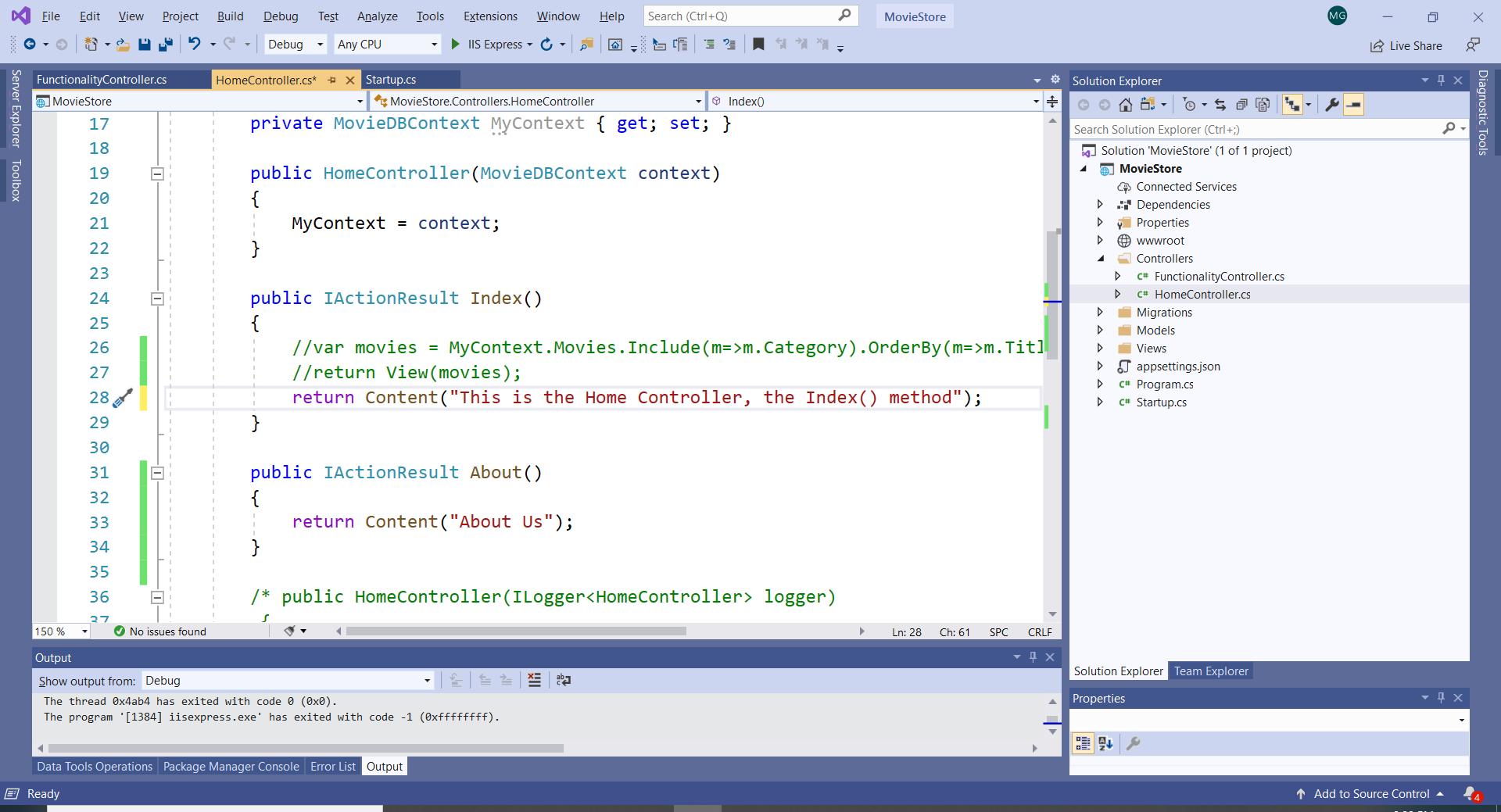
The method Content(“string”) creates a ContentResult without a View.

The next exercise demonstrates how a controller actions are executed.

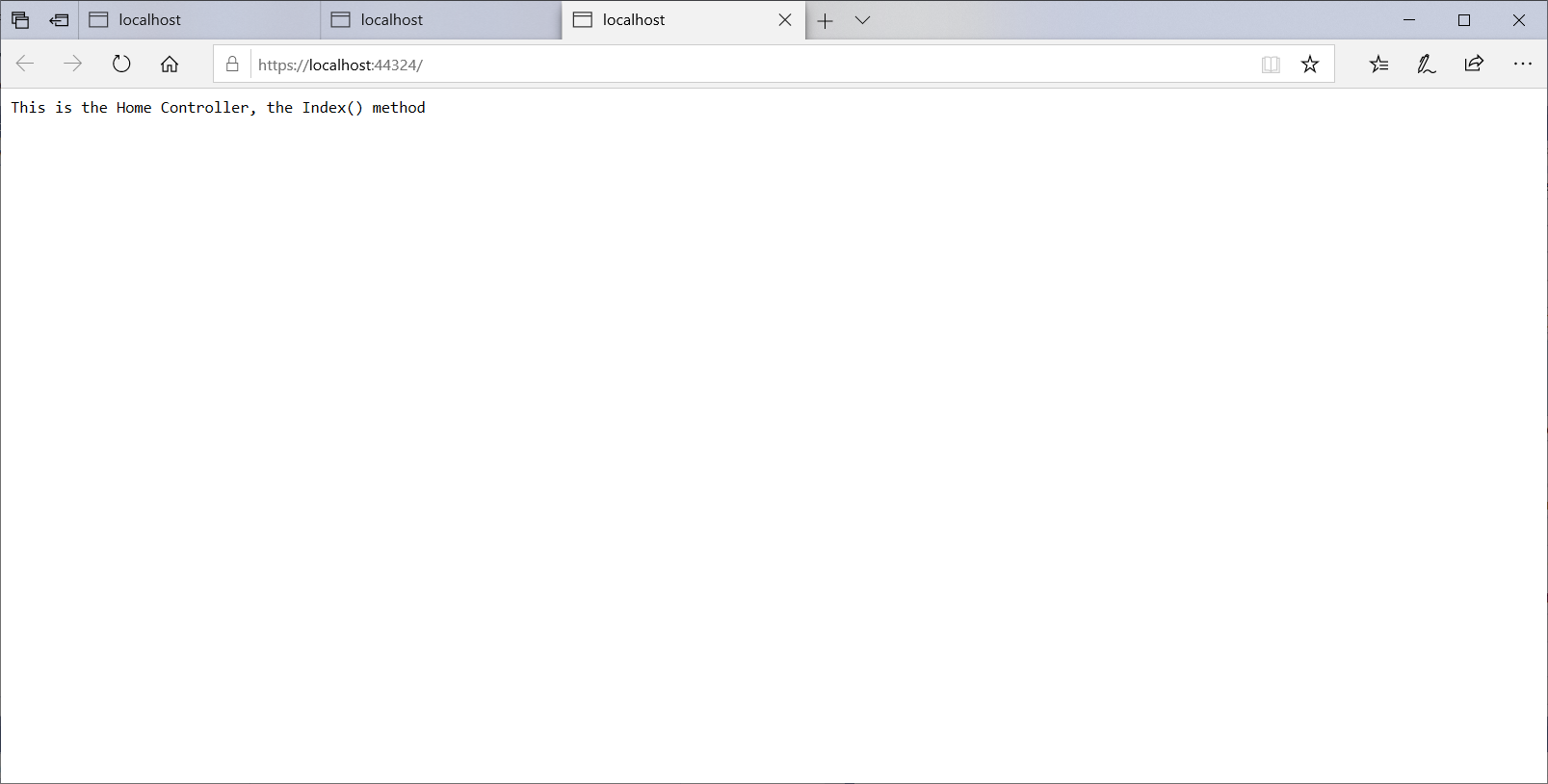
*Exercise:*

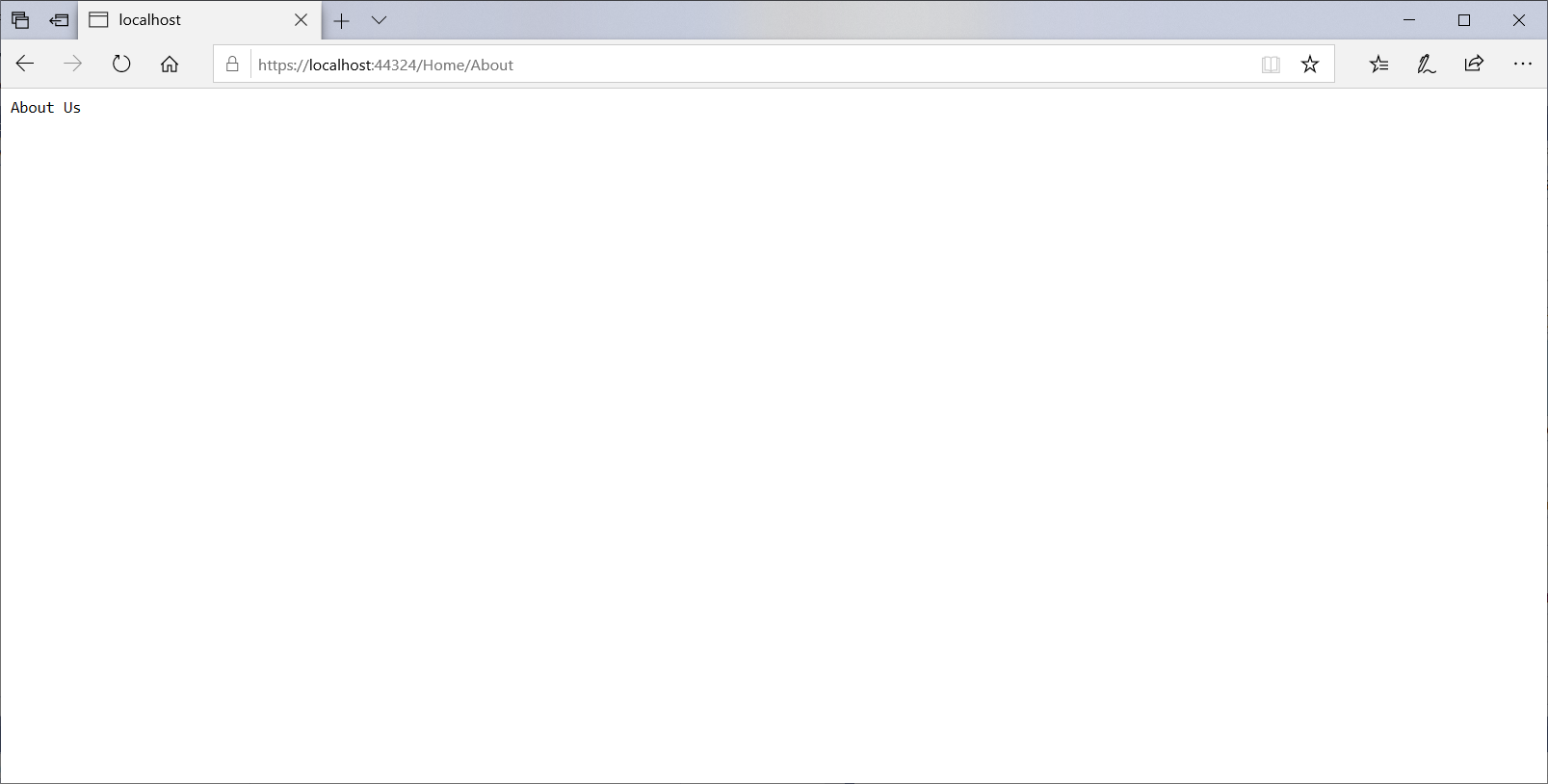
Let’s create an action method About() in the Home controller (without a view).

Let’s use the method Content() to return a string from both the Index() action method and the About() action method.



Run the application:



Now, type in the URL /Home/About and observe the result:

1. **Add more Paterns for Routing**

We learned that the Home controller has three segments as described in the Startup.cs. However, we can create other patters. For example, we intend to include a segment named “category” for a new controller named Item.

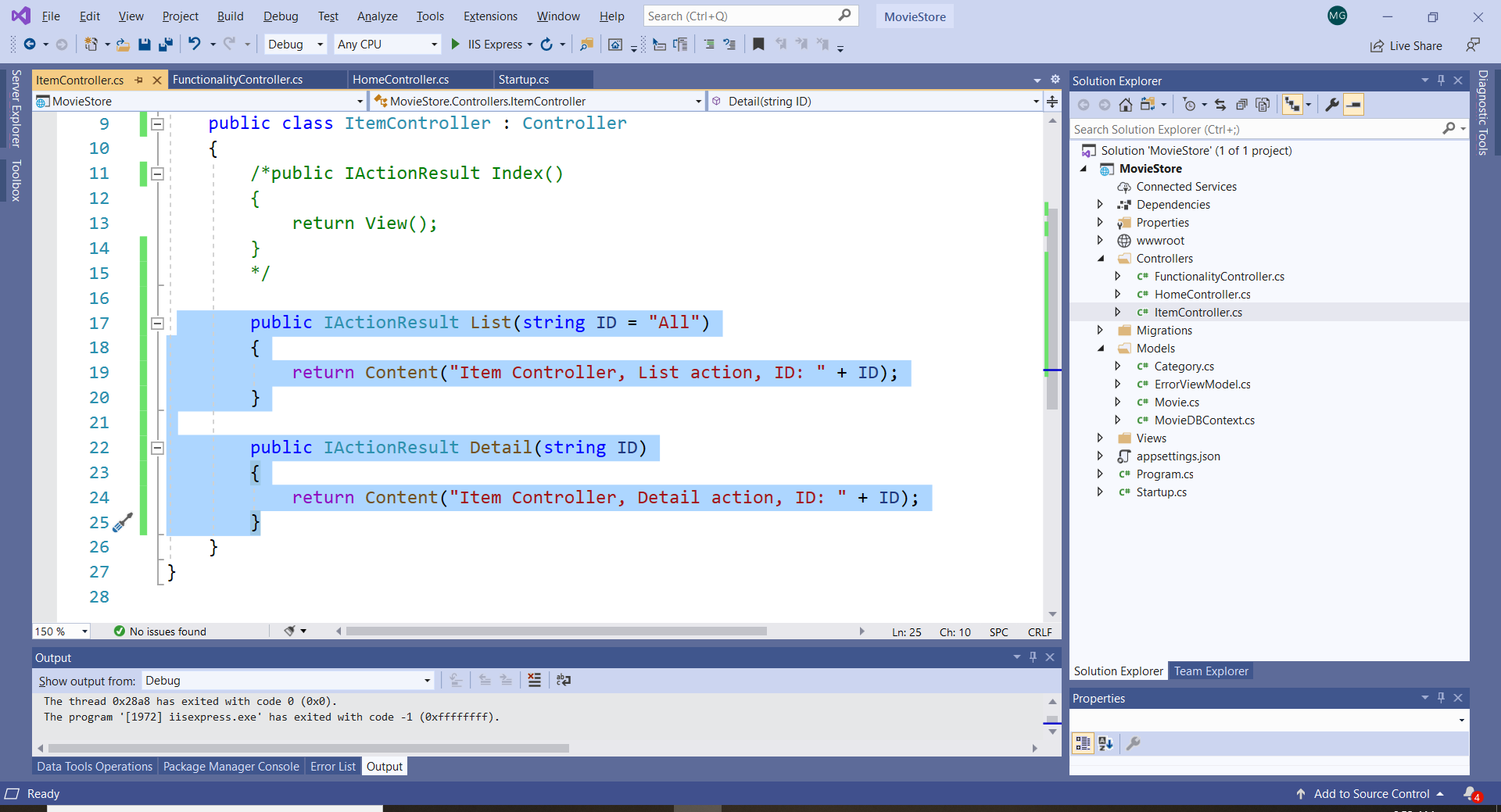
URL: “{controller}/{action}/{category}/{ID}

*Exercise*

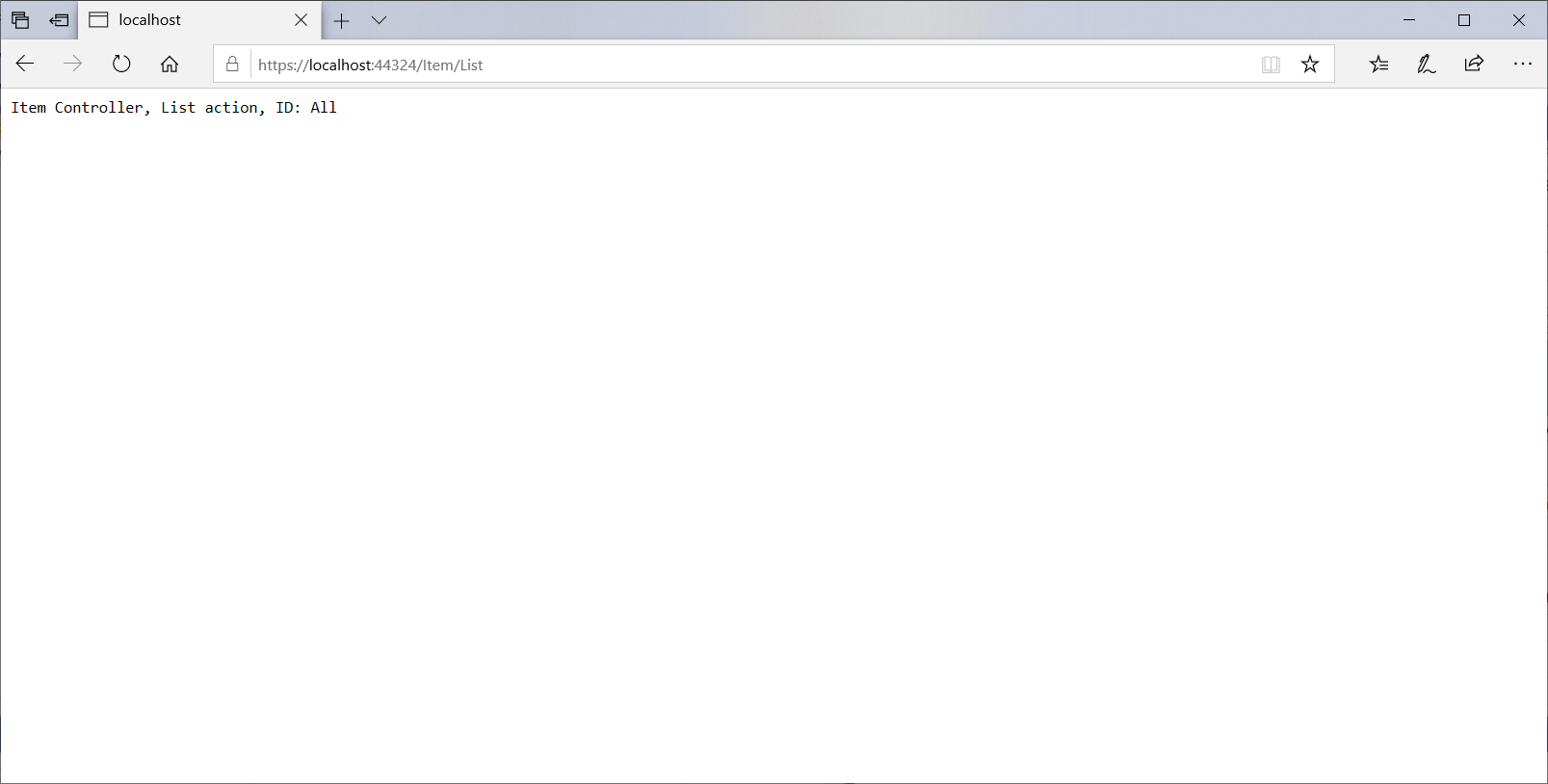
For this exercise, create a Controller that has two action methods: List() and Detail(): right click on the Controllers folder and select MVC Controller Empty.

Write the two methods that return a string and feature the Content() method.

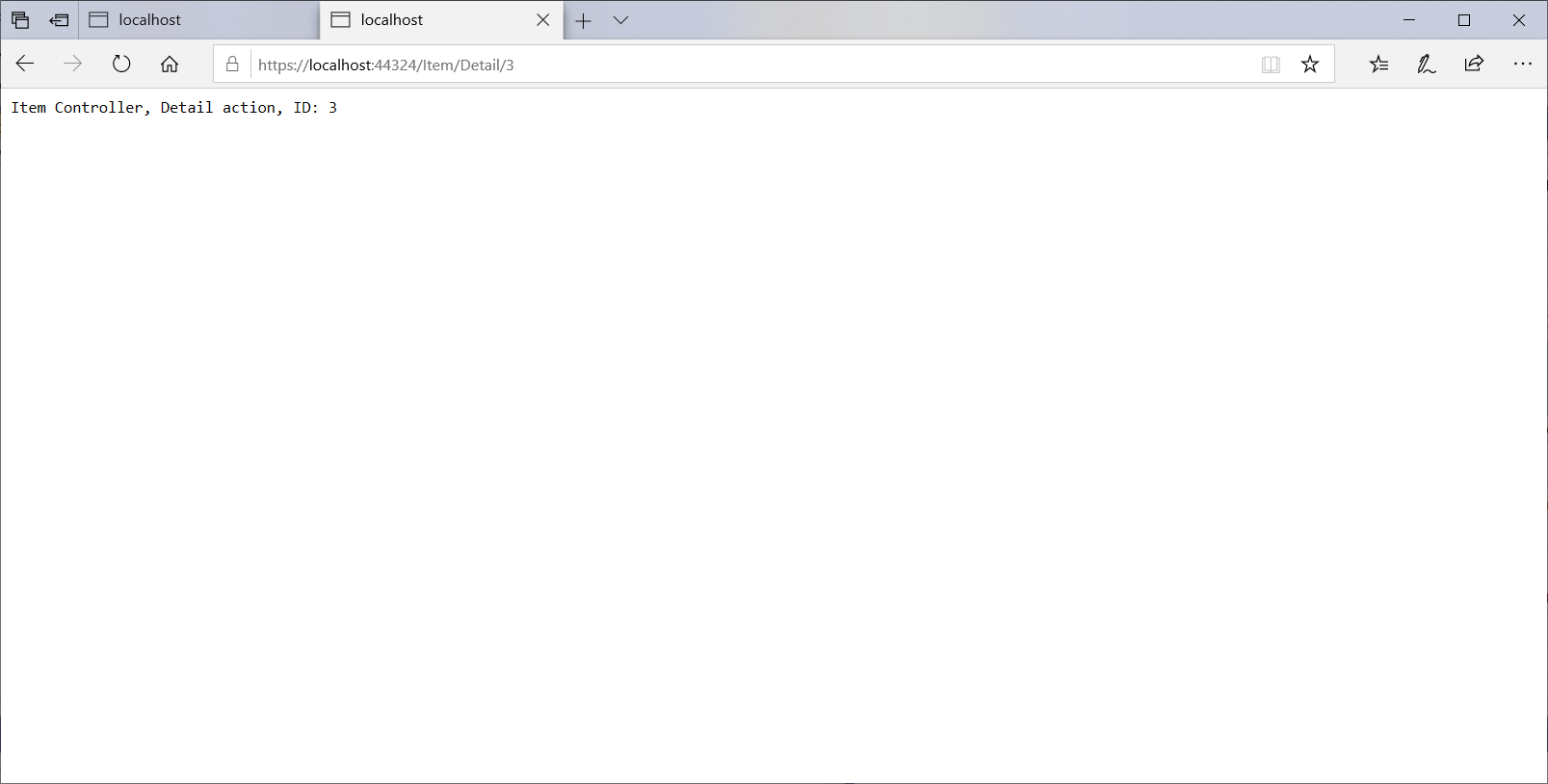
Note that the List() method provides a default value of “All” and the Detail() method doesn’t.



Run the application. Type in the URL: <https://localhost:44324/Item/List>



Type in the URL: <https://localhost:44324/Item/Detail/3>



To indicate static content as part of a segment use a string literal; to indicate a dynamic part use brackets.

You can mix a combination of static and dynamic data for segments.

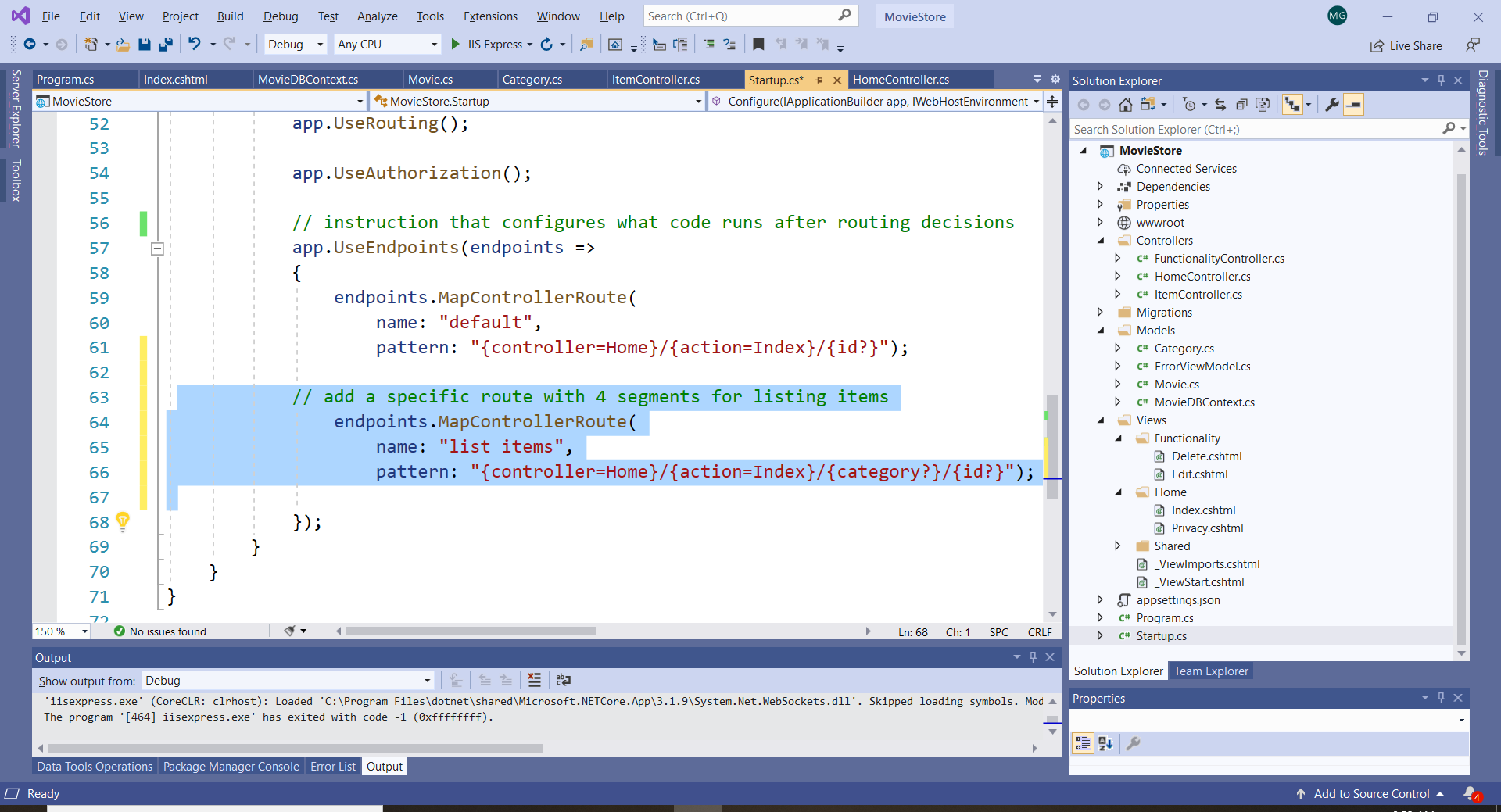
Add one more URL pattern (in the Startup.cs) to include a segment “category”.

// add a specific route with 4 segments for listing items

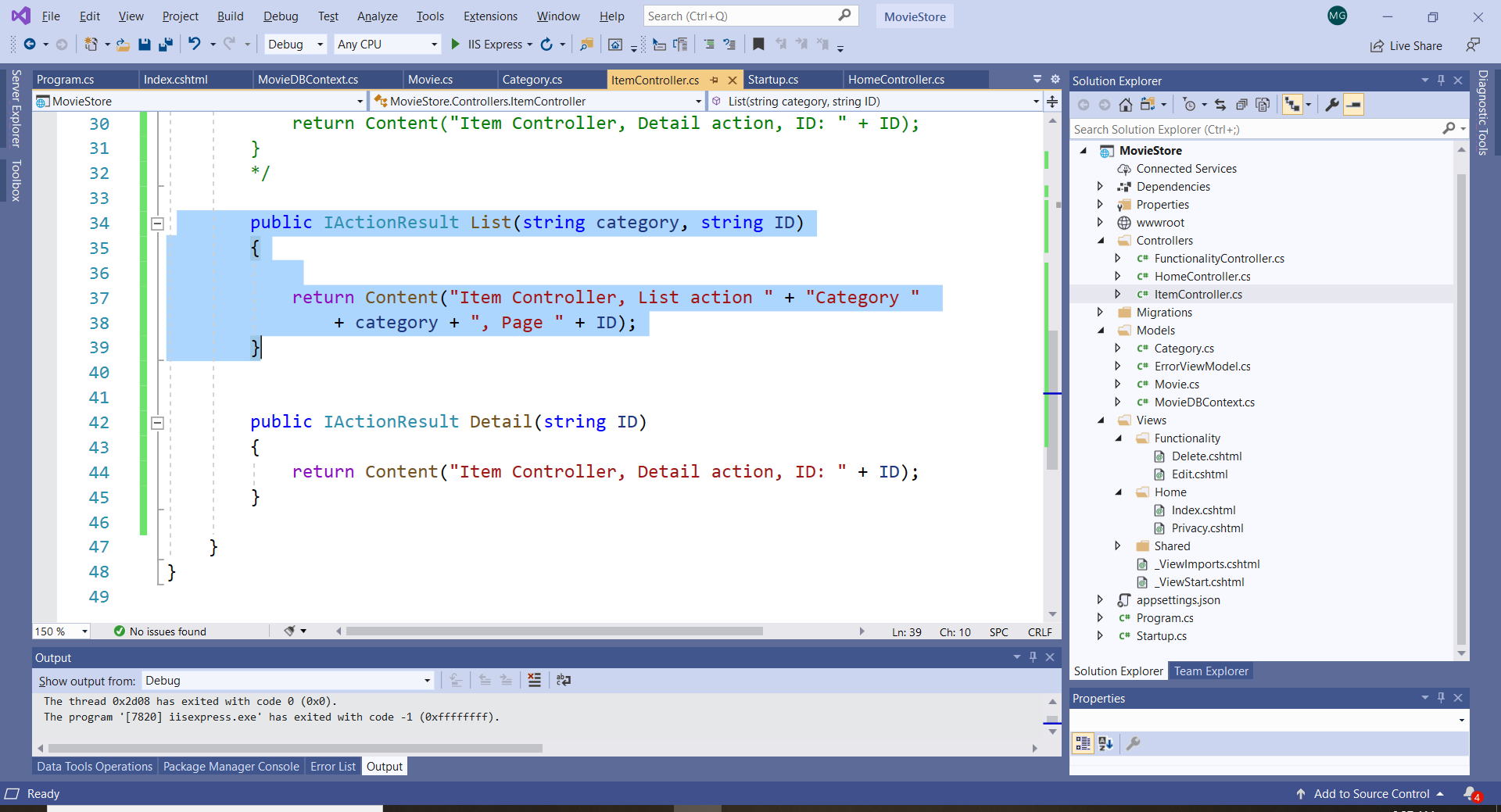
endpoints.MapControllerRoute(

name: "list items",

pattern: "{controller=Home}/{action=Index}/{category?}/{id?}");

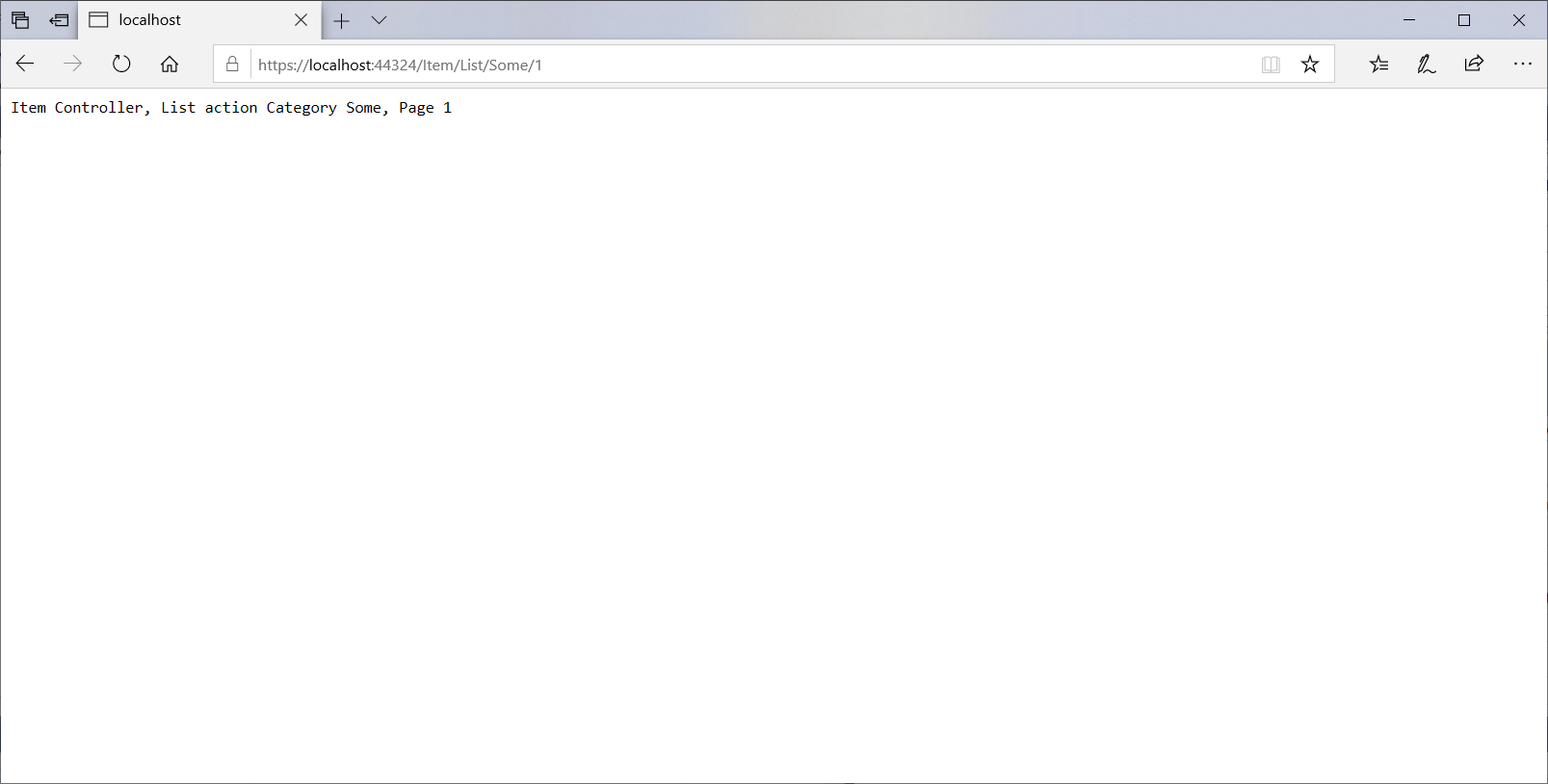


Modify the List() method to return a “category” and an “ID”:



Type in the URL: <https://localhost:44324/Item/List/Some/1>

Now the URL indicates the category (Some) and the ID (1):



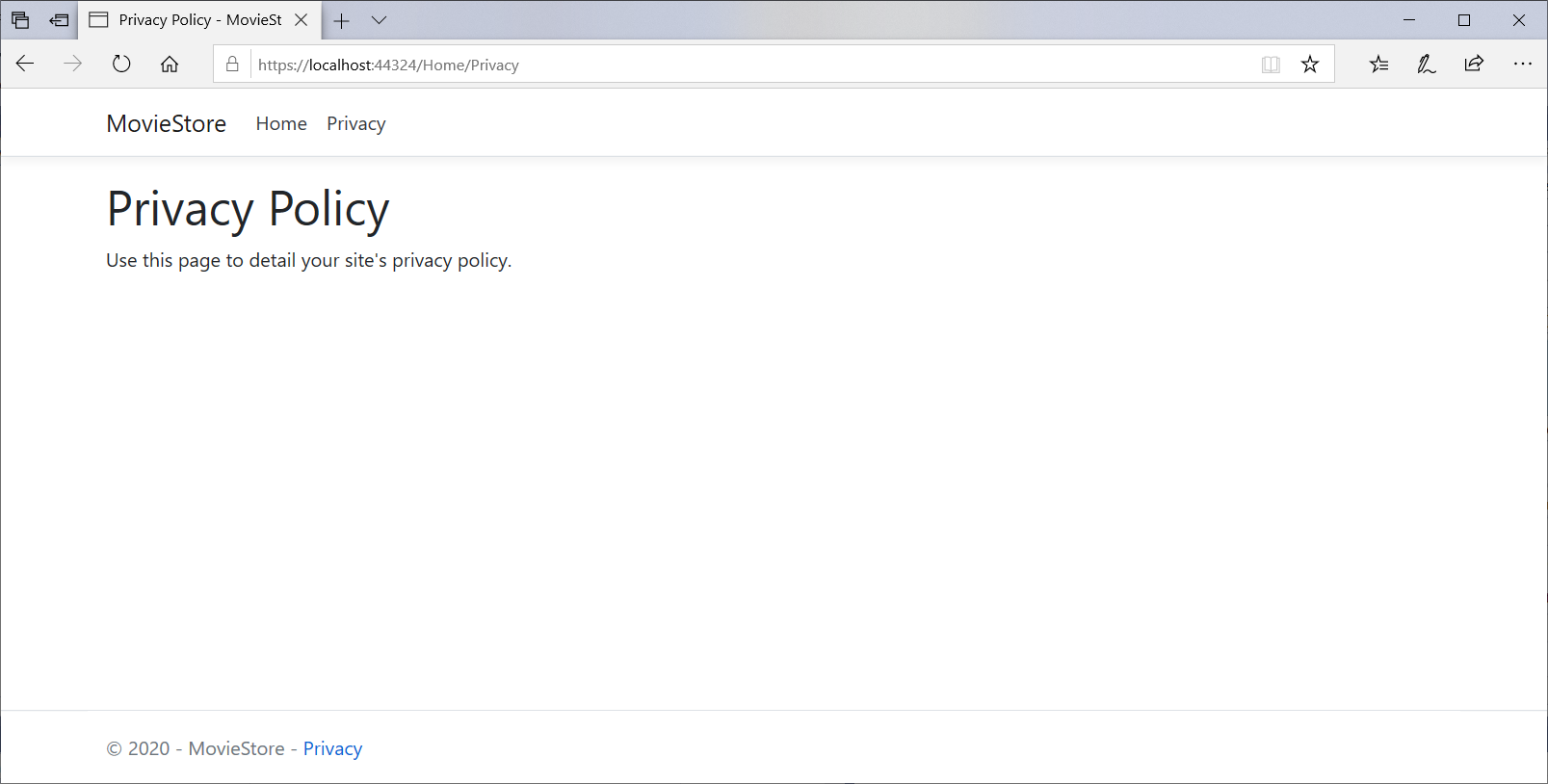
1. **Attribute Routing**

Attribute routing is used to override the patterns that are specified in the file Startup.ca. This method is called “attribute routing”.

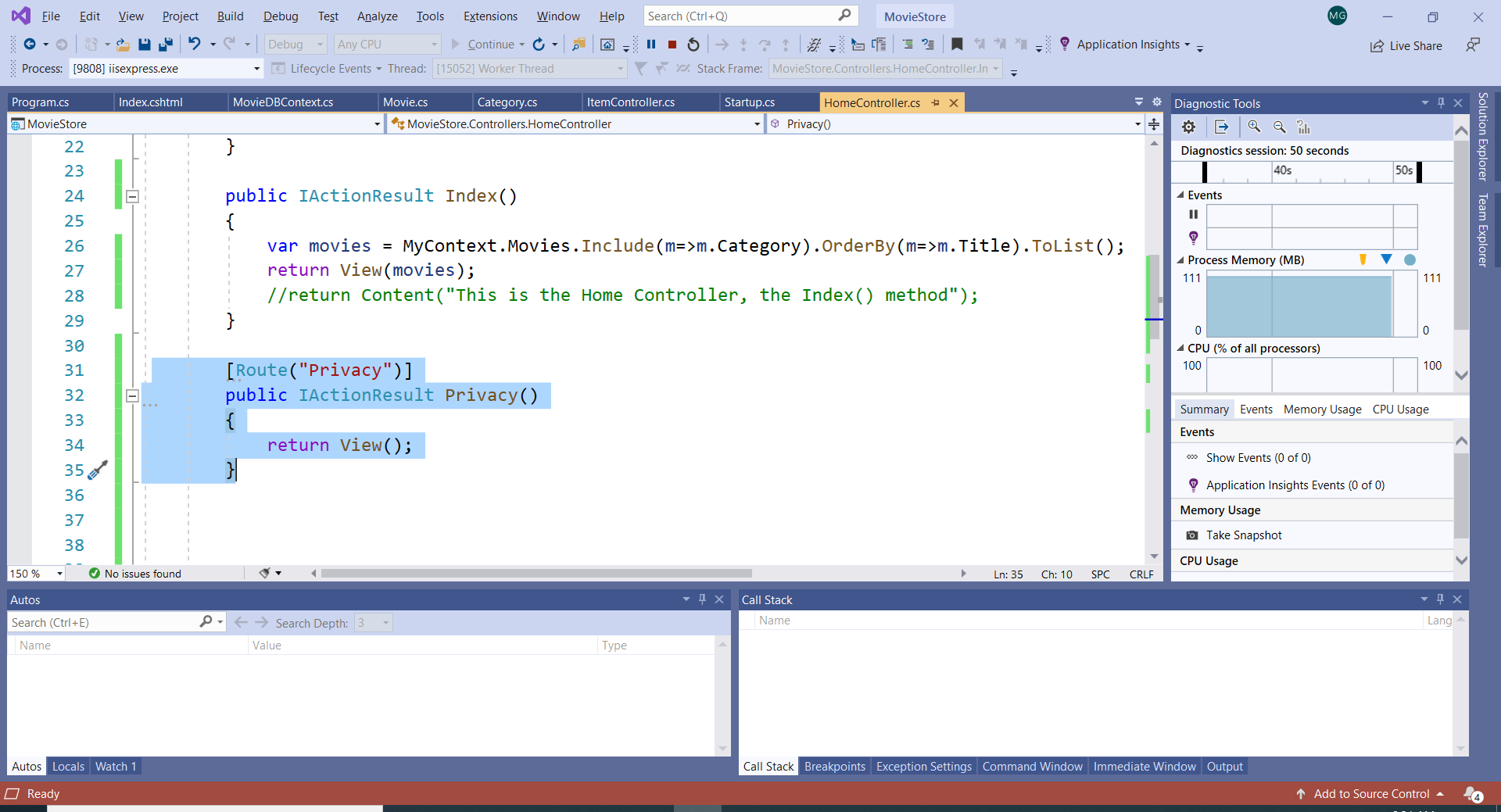
*Exercise 1*

Run the application and select “Privacy”. As per the endpoints pattern specified in Strattup.cs, note that the URL indicates the segments controller “Home” and action “Privacy”:

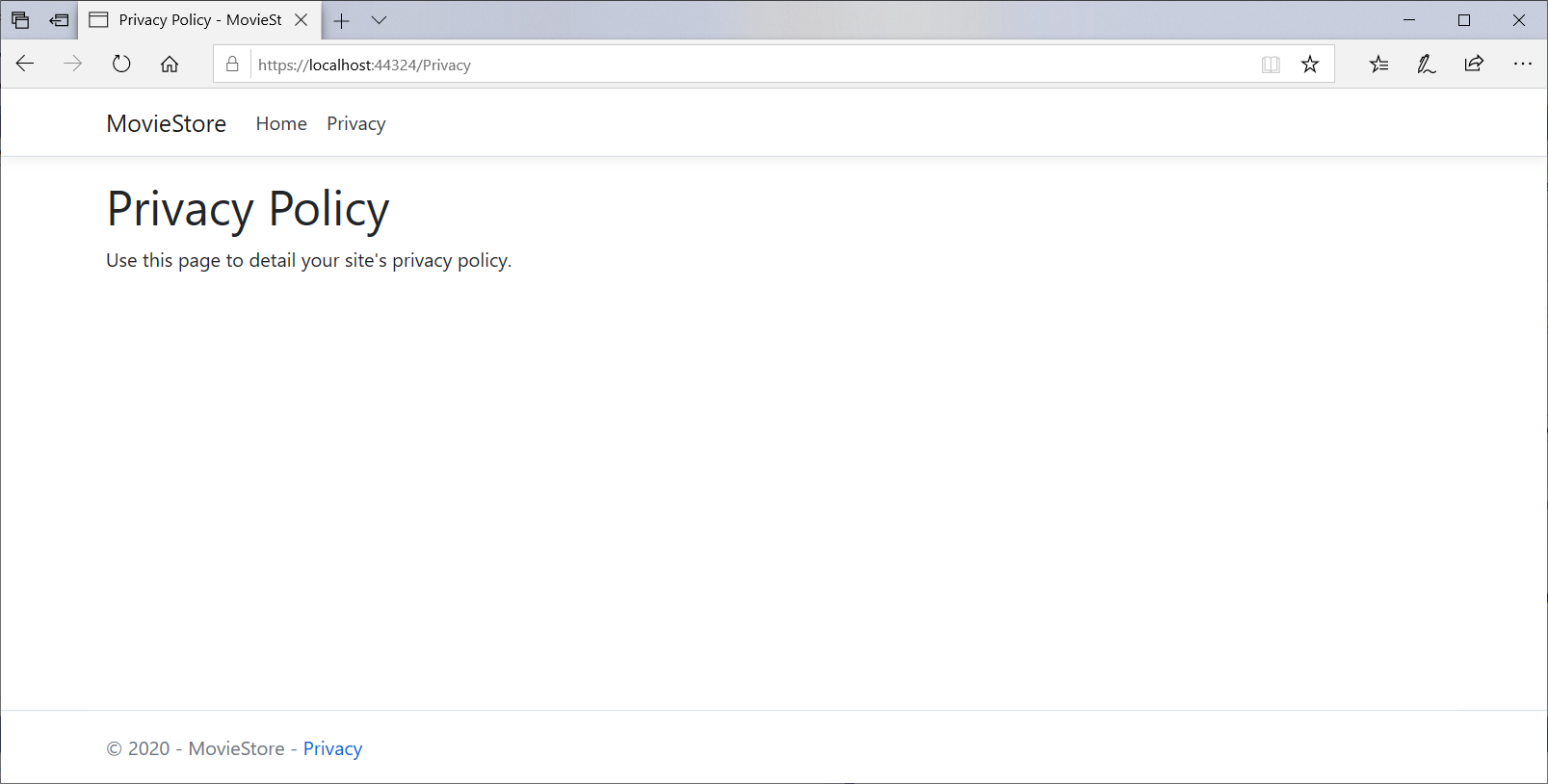
<https://localhost:44324/Home/Privacy>



We can override this rule by using attribute routing. In the Home Controller add the attribute [Route(”Privacy”)]:

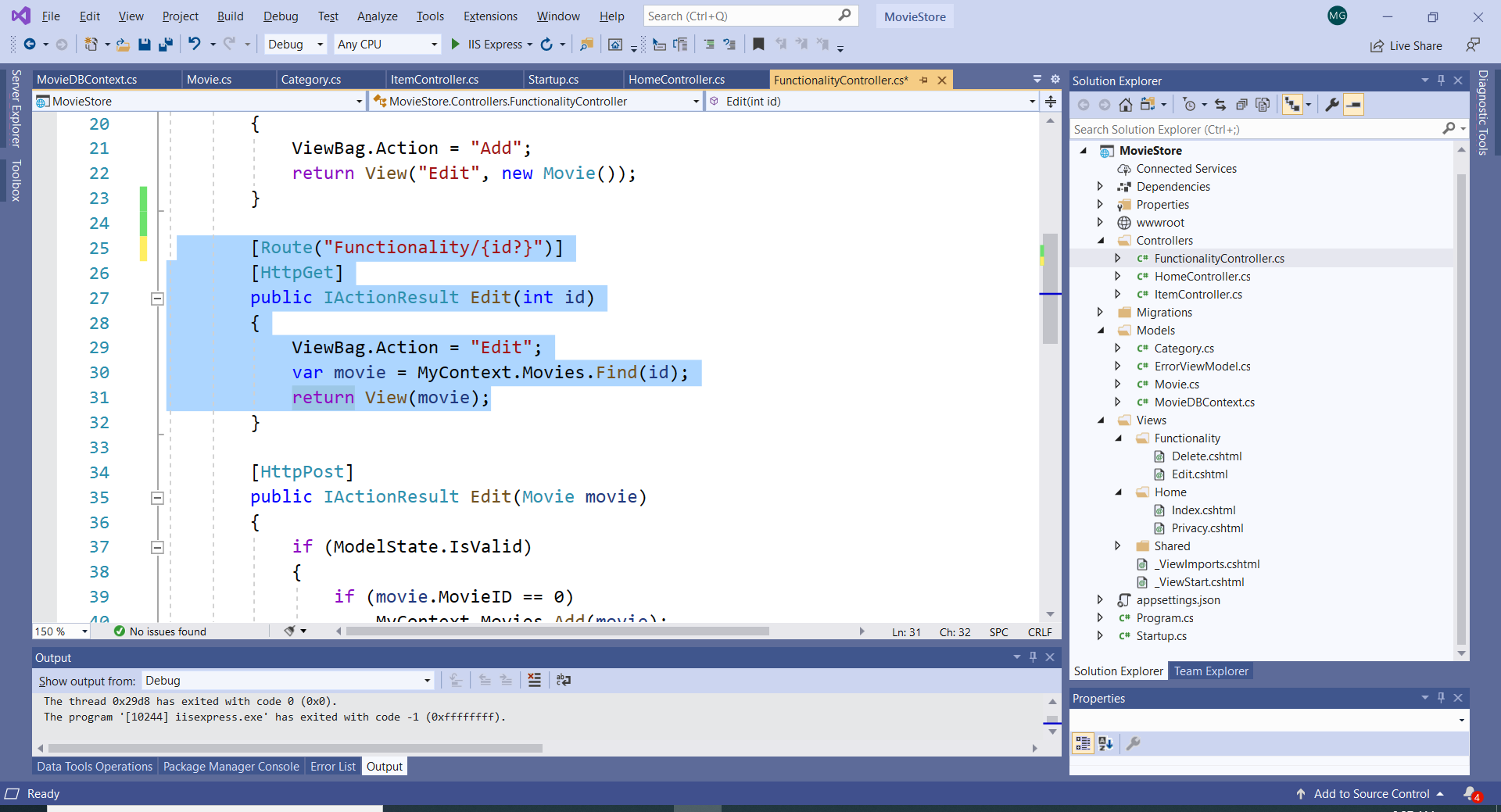


Observe that the controler name doesn’t appear anymore.



*Exercise 2*

In the Functionality controller, modify the route attribute to display only the controller and the ID:



The action “edit” is not displayed anymore (see the URL):

