

## Conditions and Terms of Use

### Microsoft Confidential

This training package is proprietary and confidential, and is intended only for uses described in the training materials. Content and software is provided to you under a Non-Disclosure Agreement and cannot be distributed. Copying or disclosing all or any portion of the content and/or software included in such packages is strictly prohibited.

The contents of this package are for informational and training purposes only and are provided "as is" without warranty of any kind, whether express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, and non-infringement.

Training package content, including URLs and other Internet Web site references, is subject to change without notice. Because Microsoft must respond to changing market conditions, the content should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred.

© 2020 Microsoft Corporation. All rights reserved.

## Copyright and Trademarks

© 2020 Microsoft Corporation. All rights reserved.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

For more information, see Use of Microsoft Copyrighted Content at <http://www.microsoft.com/en-us/legal/intellectualproperty/Permissions/default.aspx>

Microsoft and Windows are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. Other Microsoft products mentioned herein may be either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are property of their respective owners.

# Lab 08: Routing

## Introduction

Unlike previous labs, this one does not actually result in additional features or functionality being added to MyShuttle. Routing is already a part of the application as well as the API, all be it that we have largely left the standard MVC routing in place. This lab will therefore leave the solution in the same state that we started, but we will use the project to experiment with various options that are available with routing in MVC.

## Objectives

After completing this lab, you will be able to:

- Change the default routing behavior for MVC actions and APIs.
- Understand how the routing table can be used to configure default paths, and how attributes can be used to control specific routing end points.

## Prerequisites (if applicable)

This lab will use the output from Lab 07. You can choose either to continue with the solution you got by following the previous lab, or start with the solution included in the folder **Labs\Module 08 - Routing\Begin**. No additional SDKs or tools are required for this lab.

## Scenario

During this exercise, you will try out various different ways to configure the MVC routing for your API calls, and focus on the **GetCount()** action within the **Drivers** controller.

## System Requirements

To complete this lab, you need:

- Microsoft Visual Studio 2019 or higher
- Microsoft SQL Server (any edition)

## Estimated time to complete this lab

15 minutes

# Exercise 1: Customizing Routing

## Objectives

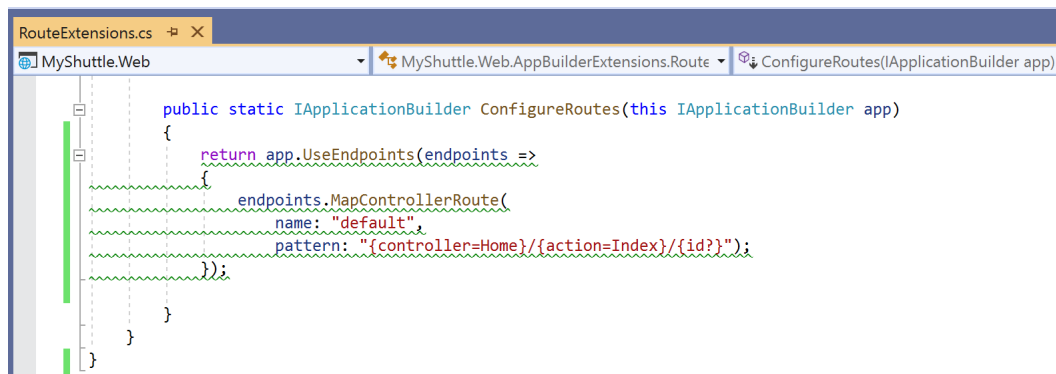
In this exercise, you will:

- Revert the API call to its "unmodified" state and verify that the existing MapRoute calls the endpoint as you would expect.
- Apply various other routing configurations and validate that the API call is being called from the new URLs.

## Task 1: Experiment with Routing

1. Open the **MyShuttle** solution, locate and open the **MyShuttle.Web / AppBuilderExtensions / RouteExtensions.cs** file.

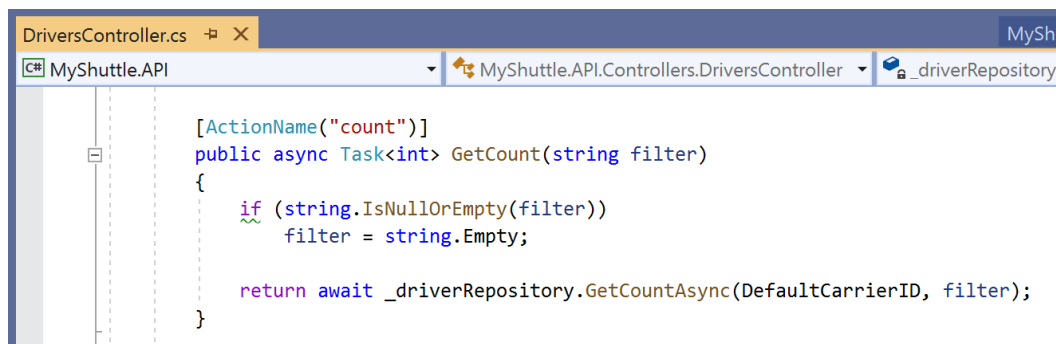
This is where the default route mapping is defined, and how the MVC routing engine will translate a URL into a method on a controller within your application.



```
RouteExtensions.cs
MyShuttle.Web
MyShuttle.Web.AppBuilderExtensions.Route
ConfigureRoutes(IApplicationBuilder app)

public static IApplicationBuilder ConfigureRoutes(this IApplicationBuilder app)
{
    return app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

2. Locate and open the **MyShuttle.API / Controllers / DriversController.cs** file.



```
DriversController.cs
MyShuttle.API
MyShuttle.API.Controllers.DriversController
_driverRepository

[ActionName("count")]
public async Task<int> GetCount(string filter)
{
    if (string.IsNullOrEmpty(filter))
        filter = string.Empty;

    return await _driverRepository.GetCountAsync(DefaultCarrierID, filter);
}
```

3. In **DriversController.cs**, we will setup our `GetCount()` method to test how our count filter works:

- Add a breakpoint to the **GetCount()** method, on the line beginning with `if (String...)`
- Build and run the application
- Then use your web browser to navigate to: `http://localhost:[YOUR LOCALHOST PORT]/Drivers/count`

4. Verify that the breakpoint is hit, and `filter` is null. Press **F5** to continue running the app.

5. Add `?filter=xyz` to the URL and refresh the browser, verify that the breakpoint is hit and that the filter value is **"xyz"**. Press **F5** to continue.

You should end up with a URL and the code break result below:

```
https://localhost:[YOUR LOCALHOST PORT]/Drivers/count?filter=xyz
```

6. Stop the debugger, comment out the line:

```
//[ActionName("count")]
```

7. Run the debugger, and notice navigation to the **/count** url no longer works – the default Route map will work however. So instead browse to:

```
http://localhost:[YOUR LOCALHOST PORT]/Drivers/GetCount
```

8. Stop the debugger, reinstate the `count` **ActionName** attribute (i.e. uncomment).

9. Now in **RouteExtensions.cs**, add a new route to the **ConfigureRoutes** map that will allow callers to use **"api"** as a prefix in front of the URL:

```
return app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "defaultWithPrefix",
        template: "api/{controller}/{action}/{id?}");

    routes.MapRoute(
        name: "default",
        template: "{controller}/{action}/{id?}",
        defaults: new { controller = "Home", action = "Index" });
});
```

10. Run the app again, now browse to:

```
http://localhost:[YOUR LOCALHOST PORT]/api/Drivers/count
```

11. However, also note that you do not have to include the **api/** prefix – the default route is still in place, and will work equally well.
12. Stop the debugger once more, now try a more constrained routing map:

```
return app.UseEndpoints(endpoints =>
{
    endpoints.MapControllerRoute(
        name: "DriverCountWithFilter",
        pattern: "api/Drivers/{filter}",
        defaults: new { controller = "Drivers", action = "count" });

    endpoints.MapControllerRoute(
        name: "defaultWithPrefix",
        pattern: "api/{controller}/{action}/{id?}");

    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");
});
```

```
//keeeeeep
return app.UseMvc(routes =>
{
    routes.MapRoute(
        name: "DriverCountWithFilter",
        template: "api/Drivers/{filter}",
        defaults: new { controller = "Drivers", action = "count" });

    routes.MapRoute(
        name: "defaultWithPrefix",
```

```
        template: "api/{controller}/{action}/{id?}");

routes.MapRoute(
    name: "default",
    template: "{controller}/{action}/{id?}",
    defaults: new { controller = "Home", action = "Index" });
});

```

13. Now the URL `http://localhost:[YOUR LOCALHOST PORT]/api/Drivers/HelloWorld` will route to our **GetCount()** method, but the filter will be set to "HelloWorld" – we have defaulted the action, but this will only apply to the Drivers controller. You may use any other string/name you like.

14. **Now remove or comment out the two new route maps we have added, before starting the next lab.**