

COMP 2139

How to work with controllers and routing



Agenda

- Describe the three segments of the default route
- Describe how the segments of the default route map to a:
 - controller,
 - controllers action methods
 - controllers parameters
- For a custom route, describe the difference between:
 - specifying static and dynamic data
- When an application has multiple routes, describe the sequence in which they must be coded
- Distinguish between attribute routing and regular routing
- List three best practices for creating URLs
- List two benefits of well-designed URLs
- Describe how you can use areas to organize the folders and files of an application.

Routing



Routing in ASP.NET

- In ASP.NET Web, a controller is a class that handles HTTP requests.
- The public methods of the controller are called action methods or simply actions.
- When the Web API framework receives a request, it routes the request to an action.
- To determine which action to invoke, the framework uses **ASP.NET routing**.
- The **ASP.NET Routing module** is responsible for mapping incoming HTTP requests to particular MVC controller actions.

MVC Methods (Configurations in Program.cs)

The methods for adding the MVC routing service:

Method	Available with
<code>AddControllersWithViews()</code>	ASP.NET Core 2.2 and later
<code>AddMvc()</code>	Versions prior to ASP.NET.Core 2.2

Two methods for enabling and configuring routing

<code>UseRouting()</code>	selects the endpoint for the route if one is found
<code>UseEndpoints(endpoints)</code>	executes the endpoint selected by the routing

Methods that configure the default route

```
// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
app.UseDeveloperExceptionPage();
app.UseHttpsRedirection();
app.UseStaticFiles();

app.UseRouting();

//Added Endpoints
app.UseEndpoints(endpoints =>
{
    // most specific route - 5 required segments
    endpoints.MapControllerRoute(
        name: "paging_and_sorting",
        pattern: "{controller}/{action}/{cat}/page{num}/sort-by-{sortby}");

    // specific route - 4 required segments
    endpoints.MapControllerRoute(
        name: "paging",
        pattern: "{controller}/{action}/{cat}/page{num}");

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

- ASP.NET introduced a new approach to routing called endpoint routing (routing across differing middleware – MVC, Razor etc..). Its generally considered a best practice to use endpoint routing for all new ASP.NET development
- Before you can use endpoint routing , you need to add the necessary MVC services to the application.
- Then you need to mark where the routing decisions are made and configure the endpoint for each route.

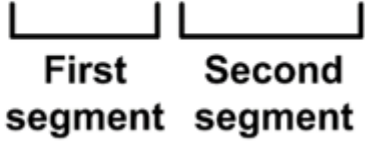
default route:

You can also use

`endpoints.MapDefaultControllerRoute()` ;

A URL that has two segments

`https://localhost:5001/Home/Index/`



First segment Second segment

The pattern for the default route

```
{controller=Home}/{action=Index}/{id?}
```

How the default route works

- The first segment specifies the controller. Since the pattern sets the Home controller as the default controller, this segment is optional.
- The second segment specifies the action method within the controller. Since the pattern sets the Index() method as the default action, this segment is optional.
- The third segment specifies an argument for the id parameter of the action method. The pattern uses a question mark (?) to specify that this segment is optional.

Request URL Mapping for the default route

`{controller=Home}/{action=Index}/{id?}`

Request URL	Controller	Action	Id
<code>http://localhost</code>	Home	Index	null
<code>http://localhost/Home</code>	Home	Index	null
<code>http://localhost/Home/Index</code>	Home	Index	null
<code>http://localhost/Home/About</code>	Home	About	null
<code>http://localhost/Product</code>	Product	Index	null
<code>http://localhost/Product/List</code>	Product	List	null
<code>http://localhost/Product/List/Guitars</code>	Product	List	Guitars
<code>http://localhost/Product/Detail</code>	Product	Detail	0
<code>http://localhost/Product/Detail/3</code>	Product	Detail	3

Controller returning Plain Text

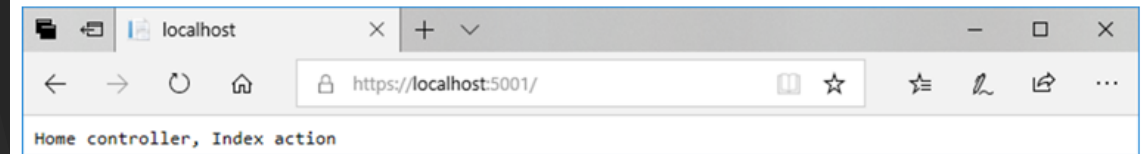
A method that a controller can use to return plain text to the browser

```
using Microsoft.AspNetCore.Mvc;

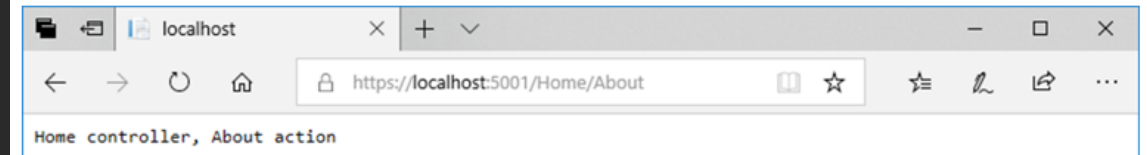
namespace GuitarShop.Controllers
{
    References
    public class HomeController : Controller
    {
        References
        public IActionResult Index()
        {
            return Content("Home controller, Index action");
        }

        References
        public IActionResult About()
        {
            return Content("Home controller, About action");
        }
    }
}
```

A browser after requesting the default page



A browser after requesting the Home/About page



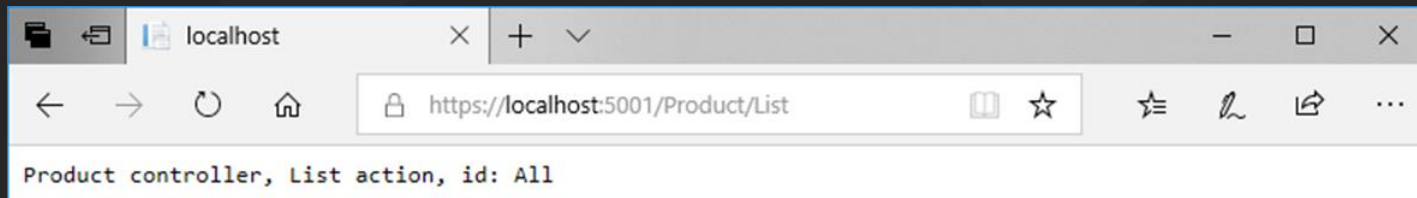
Product Controller

```
using Microsoft.AspNetCore.Mvc;

namespace GuitarShop.Controllers
{
    References
    public class ProductController : Controller
    {
        References
        public IActionResult List(string cat, int num)
        {
            return Content("Product controller, List action, " +
                "Category " + cat + ", Page " + num);
        }

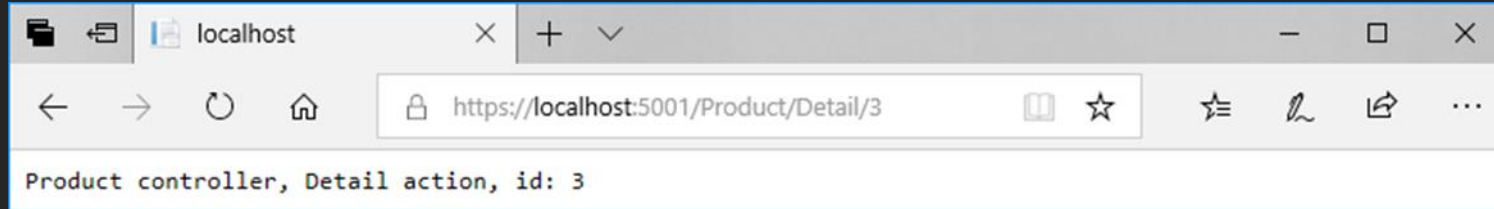
        References
        public IActionResult Detail(int id)
        {
            return Content("Product controller, Detail action, id: " + id);
        }
    }
}
```

A browser after requesting the Product/List action with no id segment

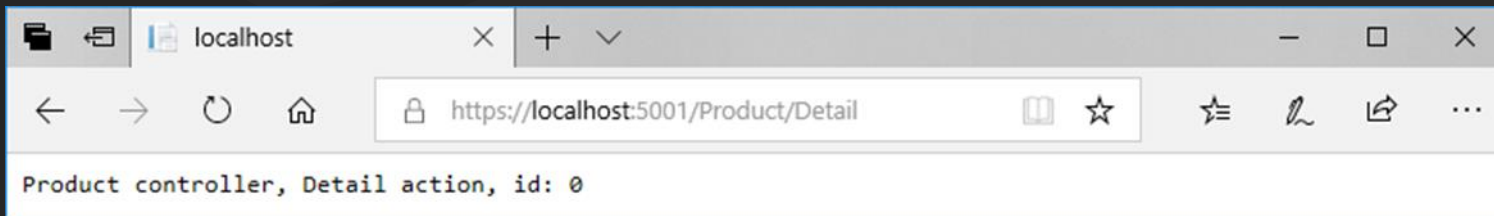


More Product Requests

A browser after requesting the Product/Detail action with an id segment



A browser after requesting the Product/Detail action with no id segment



Mixing static and dynamic data for URL

`{controller}/{action}/{cat}/Page{num}` // 4 segments



Example URLs with static and dynamic data

Request URL	Controller	Action	Parameters
<code>/Product/List/All/Page1</code>	Product	List	cat=All, num=1
<code>/Product/List/All/Page2</code>	Product	List	cat=All, num=2

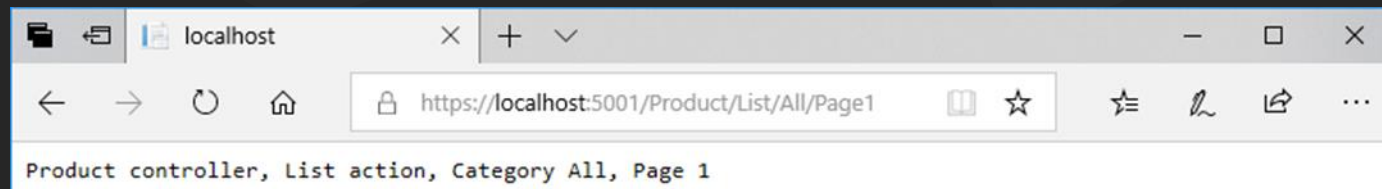
Product Controller: List method()

```
using Microsoft.AspNetCore.Mvc;

namespace GuitarShop.Controllers
{
    References
    public class ProductController : Controller
    {
        References
        public IActionResult List(string cat, int num)
        {
            return Content("Product controller, List action, " +
                "Category " + cat + ", Page " + num);
        }

        References
        public IActionResult Detail(int id)
        {
            return Content("Product controller, Detail action, id: " + id);
        }
    }
}
```

A URL that requests the Product/List action for page 1 of all categories



URL Pattern with one completely static segment

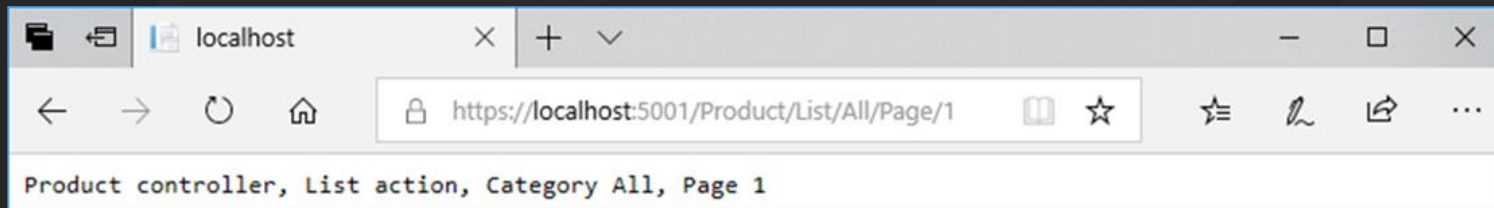
`{controller}/{action}/{cat}/Page/{num}`

// 5 segments

Example URLs with a completely static segment

Request URL	Controller	Action	Parameters
/Product/List/All/Page/1	Product	List	cat=All, num=1
/Product/List/All/Page/2	Product	List	cat=All, num=2

A URL that requests the Product/List action for page 1 of all categories



Three routing patterns mapped to controllers

```
app.UseEndpoints(endpoints =>
{
    // most specific route - 5 required segments
    endpoints.MapControllerRoute(
        name: "paging_and_sorting",
        pattern: "{controller}/{action}/{cat}/page{num}/sort-by-{sortby}");

    // specific route - 4 required segments
    endpoints.MapControllerRoute(
        name: "paging",
        pattern: "{controller}/{action}/{cat}/page{num}");

    // least specific route - 0 required segments
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    // endpoints.MapDefaultControllerRoute(); // another way to configure default route
});
```

most specific URL pattern matching

Custom routes

- most specific route listed first
- most general route listed last

optional parameter

The List() method of of the Product controller

```
using Microsoft.AspNetCore.Mvc;

namespace GuitarShop.Controllers
{
    0 references
    public class ProductController : Controller
    {
        0 references
        public IActionResult List(string cat, int num)
        {
            return Content("Product controller, List action, " +
                           "Category " + cat + ", Page " + num);
        }

        0 references
        public IActionResult Detail(int id)
        {
            return Content("Product controller, Detail action, id: " + id);
        }
    }
}
```

Sample URLs that use the default route

Request URL	Controller/Action	Parameters
/	Home/Index	id=null
/Home/About	Home/About	id=null
/Product/Detail/4	Product/Detail	id=4
/Product/List/Guitars	Product/List	id=Guitars

```
app.UseEndpoints(endpoints =>
{
    // most specific route - 5 required segments
    endpoints.MapControllerRoute(
        name: "paging_and_sorting",
        pattern: "{controller}/{action}/{cat}/page{num}/sort-by-{sortby}");

    // specific route - 4 required segments
    endpoints.MapControllerRoute(
        name: "paging",
        pattern: "{controller}/{action}/{cat}/page{num}");

    // least specific route - 0 required segments
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    // endpoints.MapDefaultControllerRoute(); // another way to configure default route
});
```

Sample URLs that use page routing

Request URL

/Product/List/All/Page3

/Product/List/Guitars/Page2

/Product/List/Guitars/Pg2

Parameters

id=All, page=3

id=Guitars, page=2

Not found because "Pg"
doesn't match static "Page".

```
app.UseEndpoints(endpoints =>
{
    // most specific route - 5 required segments
    endpoints.MapControllerRoute(
        name: "paging_and_sorting",
        pattern: "{controller}/{action}/{cat}/page{num}/sort-by-{sortby}");

    // specific route - 4 required segments
    endpoints.MapControllerRoute(
        name: "paging",
        pattern: "{controller}/{action}/{cat}/page{num}");

    // least specific route - 0 required segments
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    // endpoints.MapDefaultControllerRoute(); // another way to configure default route
});
```

```
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    app.UseHsts();
}
```

Sample URLs that use page_and_sorting routing

Request URL

/Product/List/Guitars/Page2/Sort-By-Name

Parameters

id=Guitars,
page=2,
sortby=Name

/Product/List/Guitars/Page2/By-Name

Not found because
"By-" doesn't
match "Sort-By-".

```
app.UseEndpoints(endpoints =>
{
    // most specific route - 5 required segments
    endpoints.MapControllerRoute(
        name: "paging_and_sorting",
        pattern: "{controller}/{action}/{cat}/page{num}/sort-by-{sortby}");


    // specific route - 4 required segments
    endpoints.MapControllerRoute(
        name: "paging",
        pattern: "{controller}/{action}/{cat}/page{num}");


    // least specific route - 0 required segments
    endpoints.MapControllerRoute(
        name: "default",
        pattern: "{controller=Home}/{action=Index}/{id?}");

    // endpoints.MapDefaultControllerRoute(); // another way to configure default route
});
```

Home Controller with attribute routing

```
using Microsoft.AspNetCore.Mvc;

namespace GuitarShop.Controllers
{
    0 references
    public class HomeController : Controller
    {
        [Route("/")] 
        0 references
        public IActionResult Index()
        {
            return Content("Home controller, Index action");
        }

        [Route("About")] 
        0 references
        public IActionResult About()
        {
            return Content("Home controller, About action");
        }
    }
}
```

Attributes Routing

- Overrides mapping in Program.cs
- Both of these show static route mapping
- Provides more granular control over routes

Request URL mappings that use attribute routing

Request URL	Maps to
/	the Home/Index action
/About	the Home/About action

Two default routes that are overridden by the attribute routing

/Home/Index

/Home/About

Two tokens for inserting variable data into a route

To insert the name of the **current controller or action** into a route, you can also use:

```
[controller]
```

```
[action]
```

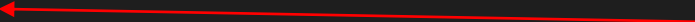
A more flexible way to code the attribute for the Home/About action

```
[Route("[action]")]
```

Mapping controllers that use attribute routing

```
//Added Endpoints
app.UseEndpoints(endpoints =>
{
    endpoints.MapControllers();

    // most specific route - 5 required segments
    endpoints.MapControllerRoute(
        name: "paging_and_sorting",
        pattern: "{controller}/{action}/{cat}/page{num}/sort-by-{sortby}");
}
```



Attribute routing is typically enabled by default. As a consequence you likely do not need to make any changes to the Program.cs to enable it.

If you do notice attribute routing is not working, you may need to enable it by calling MapControllers on your endpoints.

The Product controller with attribute routing and segments

```
using Microsoft.AspNetCore.Mvc;

namespace GuitarShop.Controllers
{
    References
    public class ProductController : Controller
    {
        [Route("[controller]s/{cat?}")] ←
        References
        public IActionResult List(string cat = "All")
        {
            return Content("Product controller, List action, Category: " + cat);
        }

        [Route("[controller]/{id}")] ←
        References
        public IActionResult Detail(int id)
        {
            return Content("Product controller, Detail action, ID: " + id);
        }

        [NonAction]
        References
        public string GetSlug(string name)
        {
            return name.Replace(' ', '-').ToLower();
        }
    }
}
```

Attributes Routing

- Overrides mapping in **Program.cs**
- Both these example utilize the name of **current controller**, and passed parameter

Sample Request URL mappings with attribute routing

Request URL	Description
/Products	Maps to the Product/List action and uses the default parameter value of "All".
/Products/Guitars	Maps to the Product/List action and passes an argument of "Guitars".
/Product/3	Maps to the Product/Detail action and supplies a valid int argument of 3.
/Product	This URL is not found. It does not map to the Product/Detail action because it does not supply the required id segment.

You can omit the NonAction attribute but then the method is still invokable as action method.

By default, the MVC framework treats all public methods of a controller class as action methods. If your controller class contains a public method and you do not want it to be an action method, you must mark that method with the NonActionAttribute attribute.

```
using Microsoft.AspNetCore.Mvc;

namespace GuitarShop.Controllers
{
    Oreferences
    public class ProductController : Controller
    {
        [Route("[controller]s/{cat?}")]
        Oreferences
        public IActionResult List(string cat = "All")
        {
            return Content("Product controller, List action, Category: " + cat);
        }

        [Route("[controller]/{id}")]
        Oreferences
        public IActionResult Detail(int id)
        {
            return Content("Product controller, Detail action, ID: " + id);
        }

        [NonAction]
        Oreferences
        public string GetSlug(string name)
        {
            return name.Replace(' ', '-').ToLower();
        }
    }
}
```

Two default routes that are overridden by attribute routing

```
/Product/List  
/Product/Detail
```

A more flexible way to code the attribute for the Product/List action

```
[Route ("[controller]s/{cat?}")]
```

A more flexible way to code the attribute for the Product/Detail action

```
[Route ("[controller]/{id}")]
```

Best practices for URLs

- Keep the URL as short as possible while still being descriptive and user-friendly.
- Use keywords to describe the content of a page, not implementation details.
- Make your URLs easy for humans to understand and type.
- Use hyphens to separate words, not other characters, especially spaces.
- Prefer names as identifiers over numbers.
- Create an intuitive hierarchy.
- Be consistent.
- Avoid the use of query string parameters if possible.

Example: URLs that identifies a products etc... (GOOD)

With a number

<https://www.domain.com/product/1307>

With a name

<https://www.domain.com/product/fender-special-edition-standard-stratocaster>

With a number and name (to keep it descriptive but short)

<https://www.domain.com/product/1307/fender-special>

Example: URLs that user query strings etc... (BAD)

```
https://www.domain.com/p/List?
```

```
/p/List?catId=1
```

```
/p/List?catId=1&pg=1
```

```
/p/Detail?id=1307
```

Example: Four URLs that follow best practices

1. `https://www.domain.com/product/list`
2. `https://www.domain.com/product/list/guitars`
3. `https://www.domain.com/product/list/guitars/page-1`
4. `https://www.domain.com/product/detail/1307/fender-stratocaster`

Example: Four shorter URLs that follow best practices

1. `https://www.domain.com/product`
2. `https://www.domain.com/product/guitars`
3. `https://www.domain.com/product/guitars/page-1`
4. `https://www.domain.com/product/1307/fender-stratocaster`



Areas

Project structure with an “Admin” area

```
GuitarShop
  /Areas
    /Admin
      /Controllers
        /HomeController.cs
        /ProductController.cs
      /Views
        /Home
          /Index.cshtml
        /Product
          /List.cshtml
          /AddUpdate.cshtml
          /Delete.cshtml
        /Shared
          /_AdminLayout.cshtml
        _ViewImports.cshtml
        _ViewStart.cshtml
```

```
/Controllers
  /HomeController.cs
  /ProductController.cs
/Models
  /Product.cs
  /Category.cs
/Views
  /Home
    /Index.cshtml
    /About.cshtml
  /Product
    /List.cshtml
    /Detail.cshtml
  /Shared
    /_Layout.cshtml
    _ViewImports.cshtml
    _ViewStart.cshtml
Program.cs
Startup.cs
```

Project structure with an Admin area....

The image shows a Visual Studio interface with two main windows. The top window displays the 'Add New Scaffolded Item' dialog, which is open to the 'Common' category under 'Installed'. The 'MVC Area' option is selected. The bottom window shows the project structure of 'GuitarShop' in the Solution Explorer, which includes 'Connected Services', 'Dependencies', 'Properties', 'launchSettings.json', 'Areas', 'Controllers', 'HomeController.cs', 'ProductController.cs', 'Program.cs', and 'Startup.cs'.

Add New Scaffolded Item

- Installed
 - Common
 - API
 - MVC
 - Razor Component
 - Razor Pages
 - Identity
 - Layout

MVC Area
by Microsoft
v1.0.0.0
An MVC area that can have its own models, views, controllers, and routes.
Id: AreaScaffolder

Solution 'Ch06bGuitarShop' (1 of 1 project)

- GuitarShop
 - Connected Services
 - Dependencies
 - Properties
 - launchSettings.json
 - Areas
 - Controllers
 - HomeController.cs
 - ProductController.cs
 - Program.cs
 - Startup.cs

Route that works with and (“Admin”) area

```
app.UseEndpoints(endpoints =>
{
    endpoints.MapAreaControllerRoute(
        name: "admin",
        areaName: "Admin",
        pattern: "Admin/{controller=Home}/{action=Index}/{id?}");

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


A token you can use to insert the area into a route

```
namespace GuitarShop.Areas.Admin.Controllers
{
    [Area("Admin")]
    public class ProductController : Controller
    {
        [Route("[area]/[controller]s/{id?}")]
        public IActionResult List(string id = "All") {...};

        public IActionResult Add() {...};
        public IActionResult Update(int id) {...};
        public IActionResult Delete(int id) {...};
    }
}
```

Request URL mappings associated with Areas

Request URL	Description
<code>/Admin/Products</code>	Maps to the Product/List action and uses the default parameter value of “All”.
<code>/Admin/Products/Guitars</code>	Maps to the Product/List action and passes an argument of “Guitars”.
<code>/Admin/Product/Add</code>	Maps to the Product/Add action.
<code>/Admin/Product/Update/3</code>	Maps to the Product/Detail action and passes an id argument of 3.

Questions?