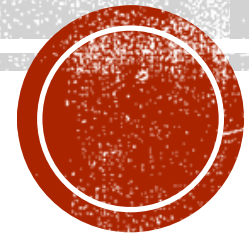


MAPPING URLS TO RAZOR PAGES USING ROUTING

Kamal Beydoun

Lebanese University – Faculty of Sciences I

Kamal.beydoun@ul.edu.lb



ROUTING IN ASP.NET CORE

```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);

builder.Services.AddRazorPages();    ❶

var app = builder.Build();

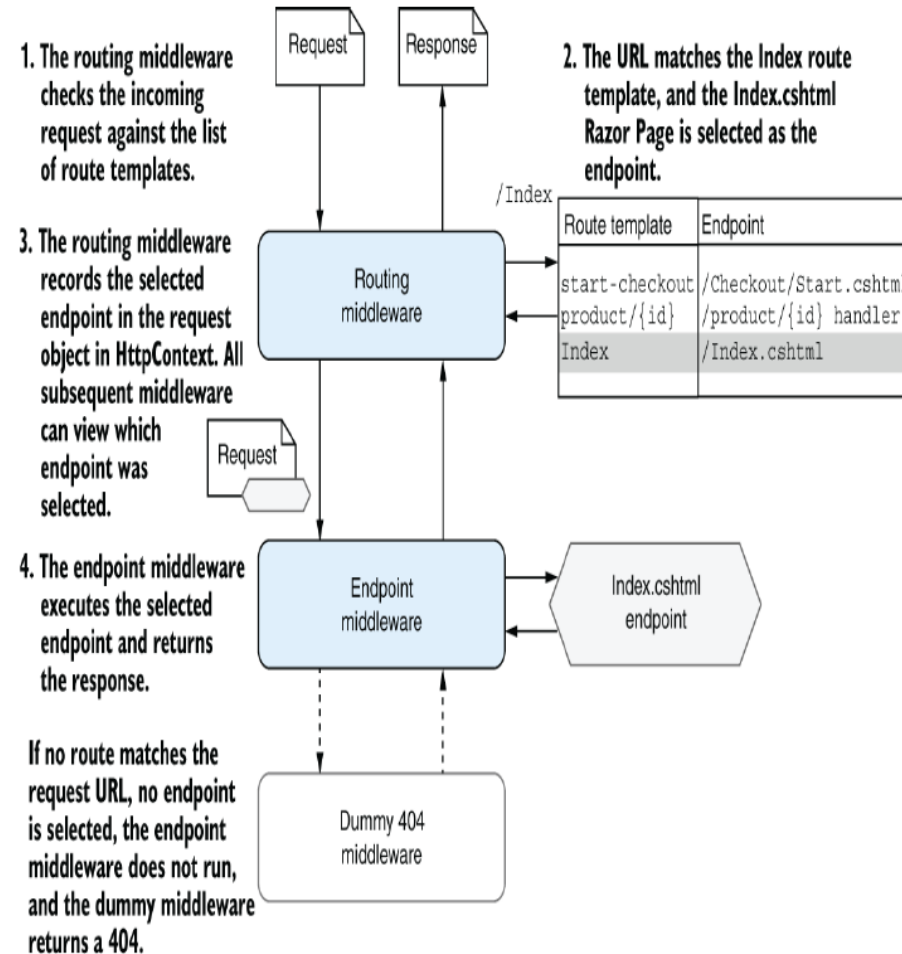
app.UseStaticFiles();
app.UseRouting();                    ❷
app.UseAuthorization();

app.MapRazorPages();                 ❸

app.Run();
```

ROUTING IN ASP.NET CORE

Routing in ASP.NET Core uses the same infrastructure and middleware whether you're building minimal APIs, Razor Pages, or MVC controllers, **but** there are **some differences in how you define the mapping** between your route templates and your handlers in each case.



CONVENTION-BASED ROUTING VS. EXPLICIT ROUTING

- Routing is a key part of ASP.NET Core, as it maps the incoming request's URL to a specific endpoint to execute.
- Two ways to define these URL–endpoint mappings in your application:
 - Using global, **convention-based routing**
 - Using **explicit routing**, where each endpoint is mapped with a single route template.

CONVENTION-BASED ROUTING VS. EXPLICIT ROUTING

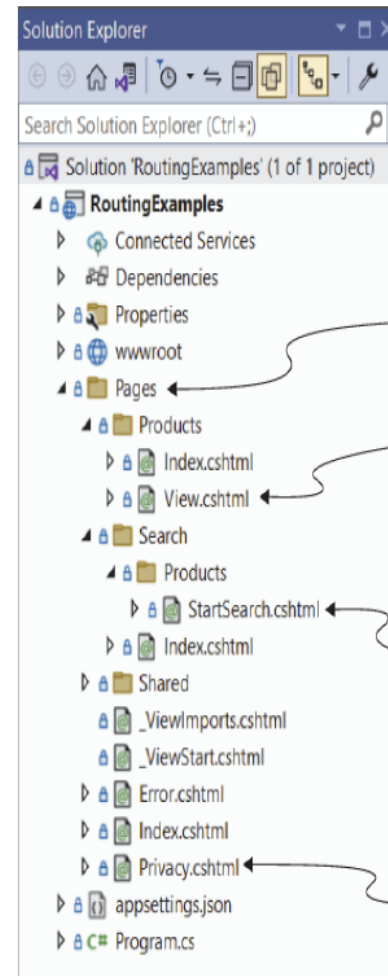
- **Convention-based routing** is defined globally for your application. You can use convention-based routes to map endpoints (MVC controller actions specifically) to URLs, but those MVC controllers must adhere strictly to the conventions you define.
- Alternatively, you can use **explicit routing** to tie a given URL to a **specific** endpoint.
 - You've seen this approach with minimal APIs, where each endpoint is **directly** associated with a route template.
- You can also use explicit routing with MVC controllers by placing **[Route]** attributes on the action methods themselves, hence explicit-routing is also often called attribute-routing.

ROUTING REQUESTS TO RAZOR PAGES

- Razor Pages use **explicit routing** by creating **route templates based on conventions**.
- For every Razor Page in your application, the framework uses the path of the Razor Page file relative to the Razor Pages root directory (Pages/), excluding the file extension (.cshtml).
- If you have a Razor Page located at the path Pages/Products/View.cshtml, the framework creates a route template with the value "Products/View"

ROUTING REQUESTS TO RAZOR PAGES

Remember that routing is not case-sensitive



Route templates are based on the file path relative to the Razor Pages root directory.

The Razor Pages root directory is called Pages.

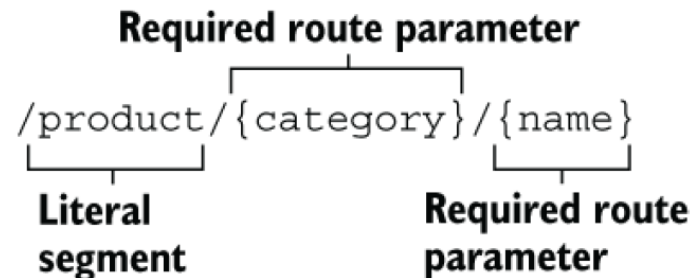
The Pages/Products/View.cshtml page has a route template of Products/View.

The Pages/Search/Products/StartSearch.cshtml page has a route template of Search/Products/StartSearch.

The Pages/Privacy.cshtml page has a route template of Privacy.

CUSTOMIZING RAZOR PAGE ROUTE TEMPLATES

- The route templates for a Razor Page are based on the file path by default, but you're also able to customize or replace the final template for each page.
- By default, Razor Pages have URLs consisting of a series of literal segments, such as "ToDo/Index".



ADDING A SEGMENT TO A RAZOR PAGE ROUTE TEMPLATE

- To customize the Razor Page route template, you update the `@page` directive at the top of the Razor Page's .cshtml file.
- To add an extra segment to a Razor Page's route template, add a space followed by the extra route template segment, after the `@page` statement.

```
@page "Extra"
```
- The default route template for the Razor Page at Pages/Privacy.xhtml, for example, is **"Privacy"**.
 - With the preceding directive, the new route template for the page would be **"Privacy/Extra"**.

ADDING A SEGMENT TO A RAZOR PAGE ROUTE TEMPLATE

- The most common reason for customizing a Razor Page's route template like this is to **add a route parameter**.
- `@page "{category}/{name}"` would match all the following URLs:
 - `/products/bags/white-rucksack`
 - `/products/shoes/black-size9`
 - `/Products/phones/iPhoneX`
- You can use the **same routing features** you learned about in chapter 6 with Razor Pages, including **optional** parameters, **default** parameters, and **constraints**.

REPLACING A RAZOR PAGE ROUTE TEMPLATE COMPLETELY

- To specify a custom route for a Razor Page, prefix the route with **/** in the `@page` directive.
 - This directive includes the `"/` at the start of the route, indicating that this is a **custom** route template, instead of an **addition**.
- `@page "{category}/{name}"`
- `@page "checkout"`

GENERATING URLS FOR RAZOR PAGES

- One of the benefits of using convention-based routing in Razor Pages is that your URLs can be somewhat fluid.
- If you rename a Razor Page, the URL associated with that page also changes.
- Renaming the Pages/Cart.cshtml page to Pages/Basket/View.cshtml, for example, causes the URL you use to access the page to change from /Cart to /Basket/View.

GENERATING URLS FOR A RAZOR PAGE

- The **Url** property is an instance of **IUrlHelper** that allows you to easily generate URLs for your application by referencing other Razor Pages by their file path.
- **IUrlHelper** has several different **overloads** of the **Page()** method.
 - Some of these methods allow you to specify a specific page handler, others let you generate an absolute URL instead of a relative URL, and some let you pass in additional route values.

```
public class IndexModel : PageModel           ❶
{
    public void OnGet()
    {
        var url = Url.Page("Currency/View", new { code = "USD" }); ❷
    }
}
```

❶ Deriving from **PageModel** gives access to the **Url** property.

❷ You provide the relative path to the Razor Page, along with any additional route values.

GENERATING URLS FOR AN MVC CONTROLLER

- Similar to Razor Pages.
- The main difference is that you use the **Action** method on the **IUrlHelper**, and you provide an MVC **controller** name and action name instead of a **page path**.

GENERATING URLS FOR AN MVC CONTROLLER

```
public class CurrencyController : Controller ❶
{
    [HttpGet("currency/index")] ❷
    public IActionResult Index()
    {
        var url = Url.Action("View", "Currency", ❸
            new { code = "USD" }); ❸
        return Content($"The URL is {url}"); ❹
    }

    [HttpGet("currency/view/{code}")]
    public IActionResult View(string code) ❺
    {
        /* method implementation*/
    }
}
```

- ❶ Deriving from Controller gives access to the Url property.
- ❷ Explicit route templates using attributes
- ❸ You provide the action and controller name to generate, along with any additional route values.
- ❹ Returns "The URL is /Currency/View/USD"
- ❺ The URL generated a route to this action method.

GENERATING URLS WITH LINKGENERATOR

- LinkGenerator has various analogous methods for generating URLs for Razor Pages and MVC actions, such as `GetPathByPage()`, `GetPathByAction()`, and `GetUriByPage()`, as shown in the following listing.

GENERATING URLS WITH LINKGENERATOR

```
public class CurrencyModel : PageModel
{
    private readonly LinkGenerator _link;
    public CurrencyModel(LinkGenerator linkGenerator)
    {
        _link = linkGenerator;
    }

    public void OnGet ()
    {
        var url1 = Url.Page("Currency/View", new { id = 5 });
        var url3 = _link.GetPathByPage(
            HttpContext,
            "/Currency/View",
            values: new { id = 5 });
        var url2 = _link.GetPathByPage(
            "/Currency/View",
            values: new { id = 5 });
        var url4 = _link.GetUriByPage(
            page: "/Currency/View",
            handler: null,
            values: new { id = 5 },
            scheme: "https",
            host: new HostString("example.com"));
    }
}
```

CUSTOMIZING CONVENTIONS WITH RAZOR PAGES

```
WebApplicationBuilder builder = WebApplication.CreateBuilder(args);  
builder.Services.AddRazorPages();
```

```
builder.Services.Configure<RouteOptions>(o => ❶  
{ ❶  
    o.LowercaseUrls = true; ❶  
    o.LowercaseQueryStrings = true; ❶  
    o.AppendTrailingSlash = true; ❶  
});
```

```
WebApplication app = builder.Build();
```

```
app.MapRazorPages();
```

```
app.Run();
```

- ❶ Changes the conventions used to generate URLs. By default, these properties are false.

ADVICE

- Avoid replacing the route template with an absolute path in a page's `@page` directive.
- Avoid adding literal segments to the `@page` directive. Rely on the file hierarchy instead.
- Avoid adding additional route templates to a Razor Page with the `AddPageRoute()` convention. Having multiple URLs to access a page can often be confusing.
- Do add route parameters to the `@page` directive to make your routes dynamic, as in `@page "{name}"`.
- Do consider using global conventions when you want to change the route templates for all your Razor Pages, such as using kebab-case, as you saw earlier.