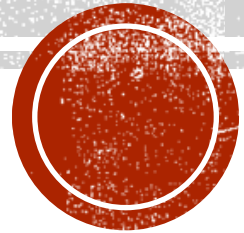
GENERATING RESPONSES WITH PAGE HANDLERS IN RAZOR PAGES



Kamal Beydoun
Lebanese University – Faculty of Sciences I
Kamal.beydoun@ul.edu.lb

RAZOR PAGES AND PAGE HANDLERS

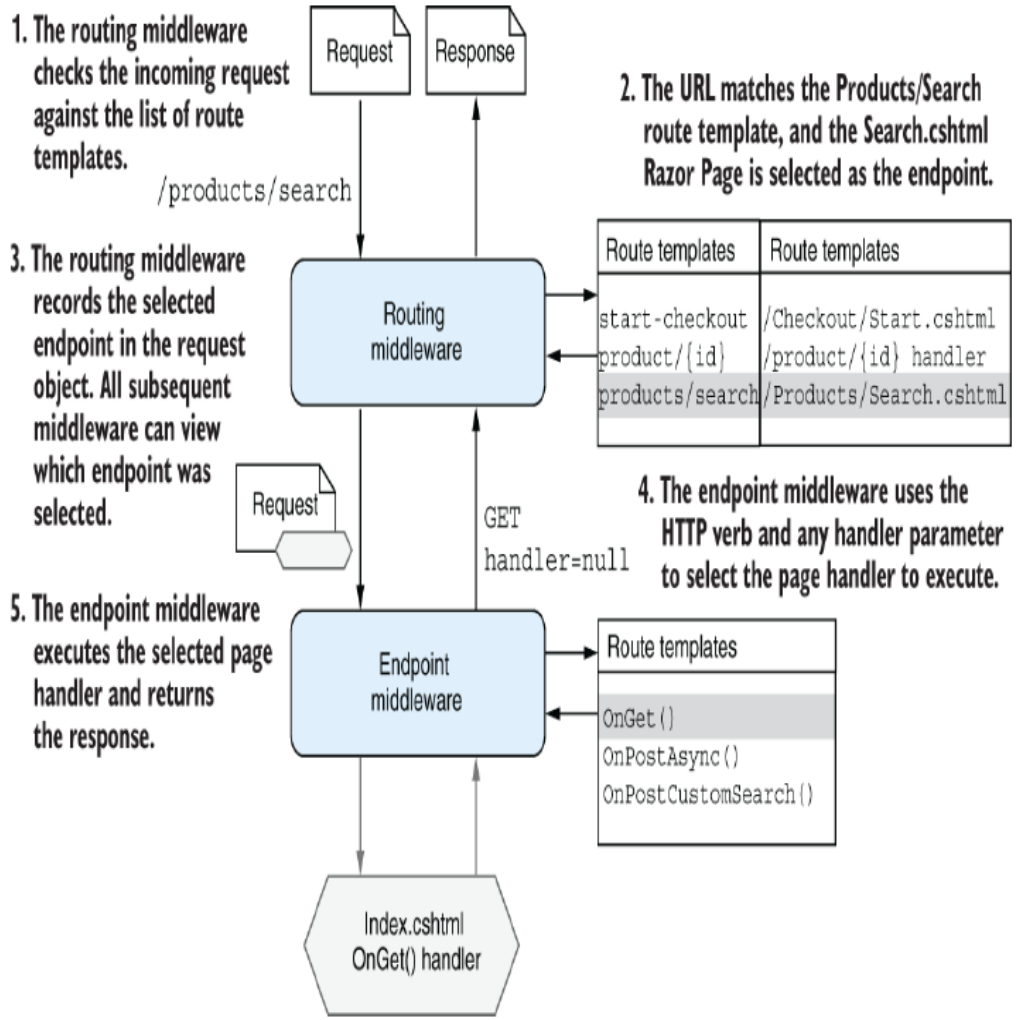
- For Razor Pages, the **entry point** is the page handler that resides in a Razor Page's **PageModel**.
- A page handler is a **method** that runs in response to a request.
- The responsibility of a page handler is generally threefold:
  - **Confirm** that the incoming request is valid.
  - **Invoke** the appropriate business logic corresponding to the incoming request.
  - **Choose** the appropriate kind of response to return.

RAZOR PAGES AND PAGE HANDLERS

Page handlers typically **return** one of three things:

- A **PageResult object**—This causes the associated Razor view to generate an HTML response.
- **Nothing** (the handler returns void or Task)—This is the same as the previous case, causing the Razor view to generate an HTML response.
- A **RedirectToPageResult**—This indicates that the user should be redirected to a different page in your application.

SELECTING A PAGE HANDLER TO INVOKE



SELECTING A PAGE HANDLER TO INVOKE

```
public class SearchModel : PageModel
{
    public void OnGet()                 ❶
    {
        // Handler implementation
    }

    public Task OnPostAsync()           ❷
    {
        // Handler implementation
    }

    public void OnPostCustomSearch()    ❸
    {
        // Handler implementation
    }
}
```

❶ Handles GET requests

❷ Handles POST requests. The async suffix is optional and is ignored for routing purposes.

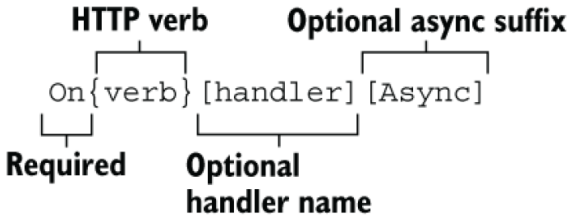
❸ Handles POST requests where the handler route value has the value CustomSearch

SELECTING A PAGE HANDLER TO INVOKE

EndpointMiddleware executes a selected Razor Page, **it selects a page handler** to invoke based on two variables:

- The **HTTP verb** used in the request (such as GET, POST, or DELETE)
- The value of the **handler route value** that comes from :
  - a **query string value** in the request URL, such as /Search?**handler**=CustomSearch.
  - a **route parameter**. You can include the {handler} route parameter in your Razor Page's route template.

@page "{handler?}"



SELECTING A PAGE HANDLER TO INVOKE

What happens if you get a request that doesn't match handlers, such as

- a request using the DELETE verb (in our example),
- a GET request with a nonblank handler value,
- or a POST request with an unrecognized handler value?

➡ The EndpointMiddleware **executes** an **implicit** page handler instead. Implicit page handlers contain **no logic**; they simply render the Razor view.

```
public void OnDelete() { }
```

ACCEPTING PARAMETERS TO PAGE HANDLERS

ASP.NET Core can bind two different targets in Razor Pages:

- **Method arguments**—If a page handler has method parameters, the arguments are bound and created from values in the request.
- **Properties** marked with a [**BindProperty**] attribute—Any properties on the PageModel marked with this attribute are bound to the request.
  - By default, this attribute does nothing for GET requests.
  - To bind properties for GET requests too, use the SupportsGet property of the attribute, as in [**BindProperty(SupportsGet = true)**].

ACCEPTING PARAMETERS TO PAGE HANDLERS

```
public class SearchModel : PageModel
{
    private readonly SearchService _searchService;                    ❶
    public SearchModel(SearchService searchService)                   ❶
    {                                                                 ❶
        _searchService = searchService;                               ❶
    }                                                                 ❶

    [BindProperty]                                                    ❷
    public BindingModel Input { get; set; }                           ❷
    public List<Product> Results { get; set; }                        ❸

    public void OnGet()                                               ❹
    {                                                                 ❹
    }                                                                 ❹

    public IActionResult OnPost(int max)                              ❺
    {
        if (ModelState.IsValid)                                       ❻
        {                                                             ❻
            Results = _searchService.Search(Input.SearchTerm, max);   ❻
            return Page();                                            ❻
        }                                                             ❻
        return RedirectToPage("./Index");                             ❻
    }
}
```

❶ The SearchService is injected from DI for use in page handlers.

❷ Properties decorated with the [BindProperty] attribute are model-bound.

❸ Undecorated properties are not model-bound.

❹ The page handler doesn't need to check if the model is valid. Returning void renders the view.

❺ The max parameter is model-bound using values in the request.

❻ If the request was not valid, the method indicates the user should be redirected to the Index page.

RETURNING IACTIONRESULT RESPONSES

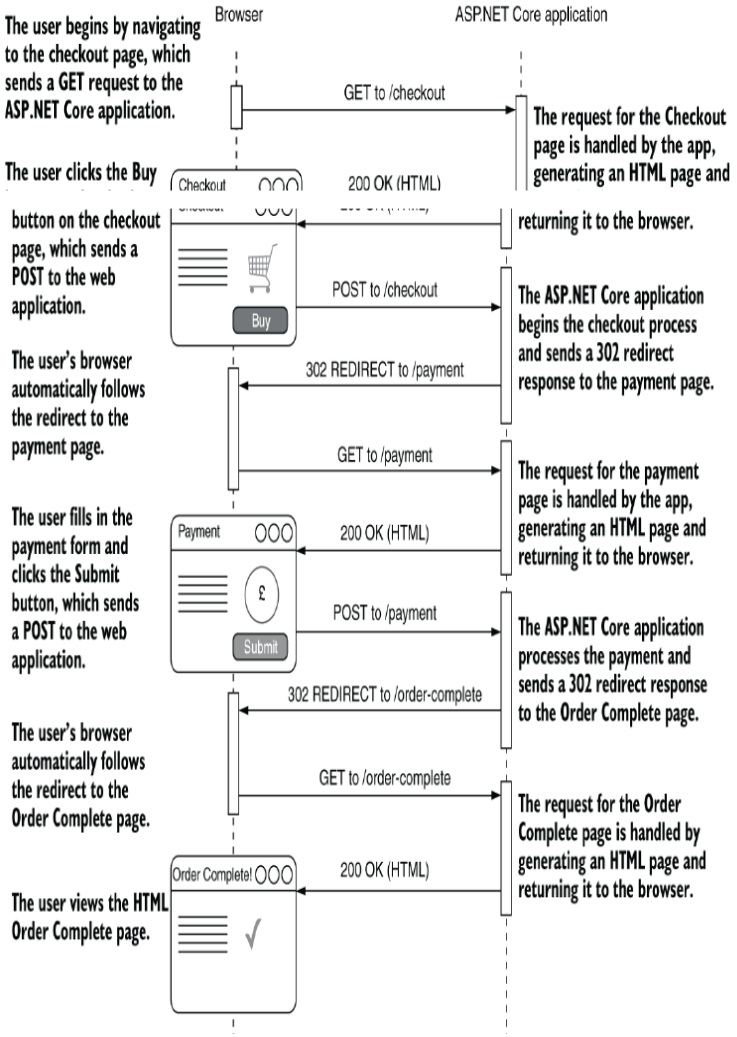
ASP.NET Core has many types of **IActionResult** :

- **PageResult**—Generates an HTML view for the associated page in Razor Pages and returns a 200 HTTP response.
- **ViewResult**—Generates an HTML view for a given Razor view when using MVC controllers and returns a 200 HTTP response.
- **PartialViewResult**—Renders part of an HTML page using a given Razor view and returns a 200 HTTP result; typically used with MVC controllers and AJAX requests.
- **RedirectToPageResult**—Sends a 302 HTTP redirect response to automatically send a user to another page.
- **RedirectResult**—Sends a 302 HTTP redirect response to automatically send a user to a specified URL (doesn't have to be a Razor Page).

RETURNING IACTIONRESULT RESPONSES

- **FileResult**—Returns a file as the response. This is a base class with several derived types:
  - **FileContentResult**—Returns a byte[] as a file response to the browser
  - **FileStreamResult**—Returns the contents of a Stream as a file response to the browser
  - **PhysicalFileResult**—Returns the contents of a file on disk as a file response to the browser
- **ContentResult**—Returns a provided string as the response.
- **StatusCodeResult**—Sends a raw HTTP status code as the response, optionally with associated response body content.
- **NotFoundResult**—Sends a raw 404 HTTP status code as the response.

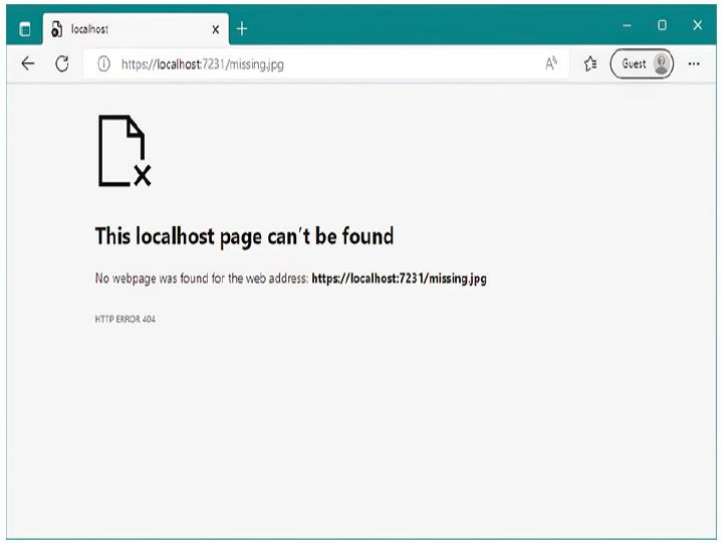
PAGERESULT AND REDIRECTTOPAGERESULT



In this flow, whenever you return HTML you use a **PageResult**; when you redirect to a new page, you use a **RedirectToPageResult**.

NOTFOUNDRESULT AND STATUSCODERESULT

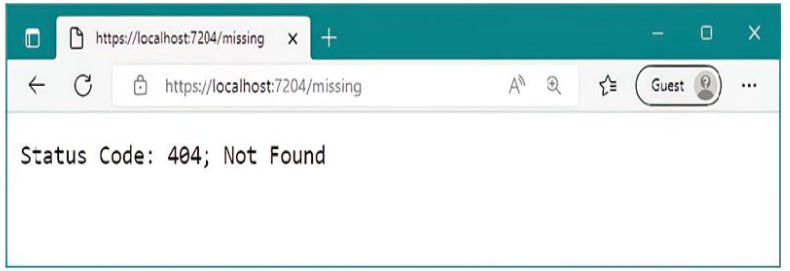
- Razor Pages can achieve this behavior of non found resource by returning a **NotFoundResult**, which returns a raw 404 HTTP status code.
- You could achieve a similar result using **StatusCodeResult** and setting the status code returned explicitly to 404.



HANDLER STATUS CODES WITH STATUSCODEPAGESMIDDLEWARE

- Razor Pages application can return a wide range of HTTP status codes that indicate some sort of error state.
- Returning **"raw"** status codes without additional content is generally OK if you're building a **minimal API or web API application**.
- For apps consumed directly by **users** such as **Razor Pages apps**, this can result in **a poor user experience**.
- Microsoft provides StatusCodePagesMiddleware for handling this use case.

```
app.UseStatusCodePages();
```

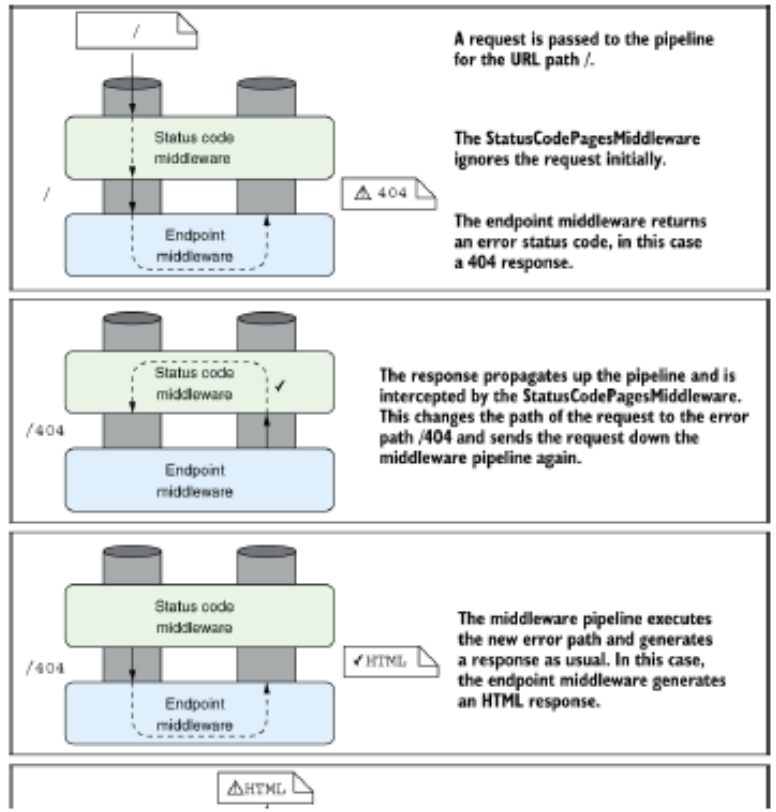


HANDLER STATUS CODES WITH STATUSCODEPAGESMIDDLEWARE

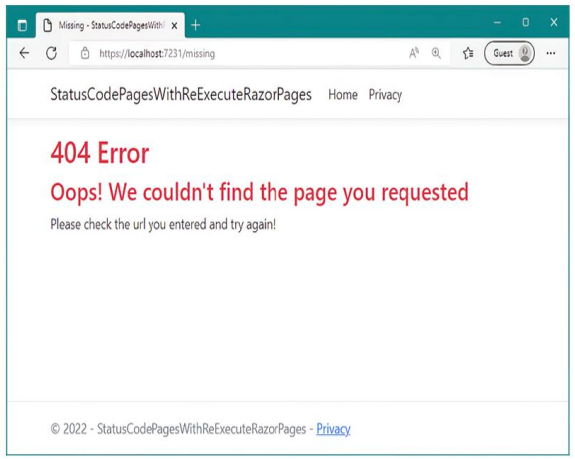
A more typical approach to using StatusCodePagesMiddleware in production is to **reexecute** the pipeline when an error is captured, using a similar technique to the ExceptionHandlerMiddleware.

```
app.UseStatusCodePagesWithReExecute("/{0}");
```

HANDLER STATUS CODES WITH STATUSCODEPAGESMIDDLEWARE



Note that the error handling path "/{0}" contains a format string token, {0}. When the path is reexecuted, the middleware **replaces** this **token** with the **status code number**.



HANDLER STATUS CODES WITH STATUSCODEPAGESMIDDLEWARE

- Instead of reexecuting the pipeline to generate the error response, you can **redirect the browser to the error page instead**, by calling

```
app.UseStatusCodePagesWithRedirects("/{0}");
```