Module 4

Developing Pages

Solid Innovator

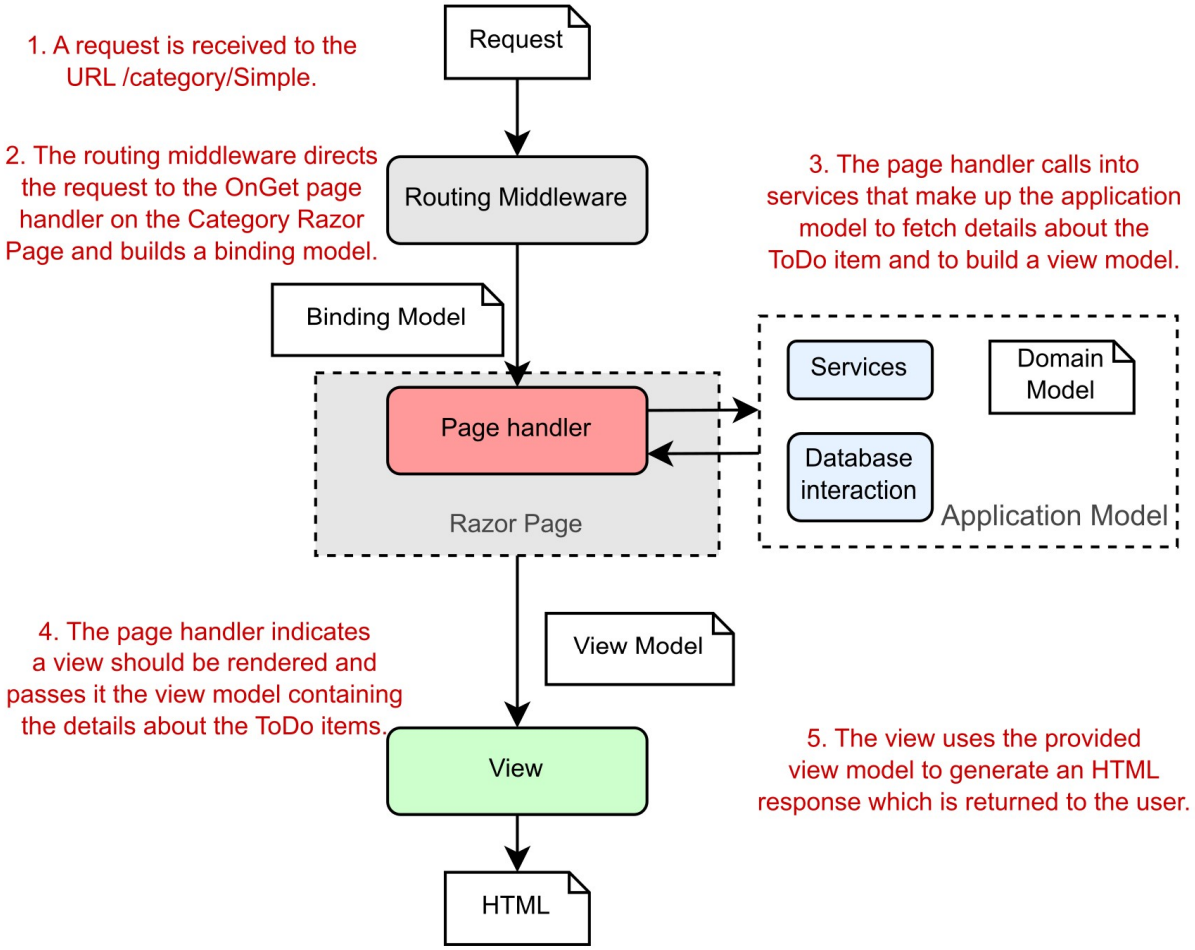
Module Overview

- Razor Pages
- Razor Page Handlers
- Configuring Routes

Lesson 1: Razor Pages

- Responding to User Requests
- Writing Page Handlers
- Using Parameters
- Using ViewBag and ViewData to Pass Information to Pages
- Demonstration: How to Write Pages and Handlers

Responding to User Requests



https://andrewlock.net/aspnetcore-in-action-2e-applying-the-mvc-design-pattern-to-razor-pages/

Razor Pages

- Combination of two files
  - –PageModel file: C# file ending in .cshtml.cs
    Class inheriting from PageModel containing handlers with server side logic
  - –Content Page: Razor file ending with .cshtml.
    Contains a mixture of client-side and server-side code, which, when processed, results in HTML being sent to the browser

Razor Page Handlers

- Methods automatically executed as a result of a request
- Naming convention used to select the appropriate handler method to execute
  - HTTP verb used for the request prefixed with "On": OnGet(), OnPost(), OnPut() etc.
- Optional asynchronous equivalents: OnPostAsync(), OnGetAsync() etc.
- Must be public
- Can have any return type
  - Typically void (or Task if asynchronous) or ActionResult.

Razor Pages

```
//index.cshtml
@page
<h3>@Message</h3>
<form method="post"><button class="btn btn-default">Click to post</button></form>
<p><a href="/" class="btn btn-default">Click to Get</a></p>
```

```
//Index.cshtml.cs
public class IndexModel : PageModel {
    public string Message { get; set; }
    public void OnGet() {
        Message = "Get used";
    }
    public void OnPost() {
        Message = "Post used";
    }
}
```

Named Handlers

- Specify multiple methods that can be executed for a single verb.
- The name of the handler is added to the form's action as a query string parameter

```
public void OnPost() {
    Message = "Form Posted";
}
public void OnPostDelete() {
    Message = "Delete handler fired";
}
public void OnPostEdit(int id) {
    Message = "Edit handler fired";
}
public void OnPostView(int id) {
    Message = "View handler fired";
}
```

```
<form method="post">
  <button>Submit</button>
  <button asp-page-handler="edit" >Edit</button>
  <button asp-page-handler=“delete" >Delete</button>
  <button asp-page-handler=“view" >View</button>
</form>
```

Parameters in Handlers Methods

- Handlers can accept parameters

```
public void OnPost (int id) {
    Message = $“Post handler fired for {id}";
}
```

In a POST handler, the parameter name must match a form field name for the incoming value to be automatically bound to the parameter:

```
<form method="post">
  <button>Submit</button>
  <input type="hidden" name="id" value="3" />
</form>
```

Action Result Types

Handlers can return different types of results:

- PageResult
- ContentResult
- NotFoundResult
- RedirectToPageResult
- BadRequestResult
- StatusCodeResult

Using Parameters

- The model binders obtain parameters from a user request and pass them to handlers
- There are several ways to retrieve parameters, including:
  - The Request property
  - The FormCollection object
  - Routing

Demonstration: How to Write Razor Pages and Handlers

In this demonstration, you will learn how to:

- Add a Razor Page to an ASP.NET Web application
- Add a handler that creates a model and passes it to a Page
- Add another handler that gets a parameter

Lesson 2: Configuring Routes

- The ASP.NET Core Routing Engine
- Discussion: Why Add Routes?
- What Is Search Engine Optimization?
- Configuring Routes by Using Convention-Based Routing
- Using Routes to Pass Parameters
- Configuring Routes by Using Attributes
- Demonstration: How to Add Routes

URL Matching

Virtual path of content page, removing root folder name and file extension.

- `/Pages/Privacy.cshtml -> yourdomain.com/privacy`
- `/Pages/Error.cshtml -> yourdomain.com/error`
- `/Pages/OneFolder/OnePage.cshtml -> yourdomain.com/OneFolder/OnePage`

`Index.cshtml` is considered the default document.
It has two routes defined

- file name without the extension -> `yourdomain.com/index`
- empty string -> `yourdomain.com`

Passing Parameters to Pages

- Supply value as a query string value
  - www.myblog.com/post?title=my-latest-post
- Supply Value as Form Input Field
  - <input type="hidden" name="title" value="my-latest-post">
- Supply value as Route Data
  - www.myblog.com/post/my-latest-post

Route Data

- Segment in the URL
- Plays no part in matching files on disk
- *Parameter* - arbitrary piece of data passed in the URL
- More readable
- More search engine-friendly

Route Templates

- Route Data parameters are defined in a **Route Template** as part of the **@page** directive in the .cshtml file.

```
@page "{title}"
```

make the parameter optional by adding a ? after it:

```
@page "{title?}"
```

provide a default value for the parameter

```
@page "{title=first post}"
```

Route Data Constraints

- Constrain route parameters values by data type and range

- `@page "{id:int}"`
- `@page "{id:min(10000)}"`
- `@page "{username:alpha:minlength(5):maxlength(8)}"`

Accessing Route Data Values

- RouteData.Values property

```
@RouteData.Values["title"]
```

- Add a parameter to a handler

```
public void OnGet(string title)
```

Lab: Developing Pages

- Exercise 1: Adding Pages and Handlers to a Web Application
- Exercise 2: Configuring Routes

Estimated Time: 30 minutes