Razor Pages in ASP.NET Core

1. Introduction to Razor Pages

What Are Razor Pages?

- Razor Pages is a **page-based programming model** in ASP.NET Core.
- Introduced in **ASP.NET Core 2.0** as a **simplified alternative** to MVC (Model-View-Controller).
- Ideal for **page-focused web applications** (e.g., forms, CRUD operations).
- Encourages **separation of concerns** and **clean architecture**.

Key Benefits of Razor Pages:

- More **organized and maintainable** code.
- Each page has its own **.cshtml** file and **PageModel class**.
- Cleaner routing (based on file paths).
- No need for separate controllers.

2. Razor Page Structure

Basic Razor Page File

```
/Pages/
   Index.cshtml
   Index.cshtml.cs
```

- `Index.cshtml` → Razor markup (HTML + C#)
- `Index.cshtml.cs` → PageModel C# class (handles logic, bound to the Razor page)

🧪 3. Razor Programming Rules

Razor Syntax Overview

- Razor uses `@` to transition from HTML to C#.

```
<h1>Hello @Model.Name</h1>
```

Code Blocks:

```
@{
    var message = "Welcome!";
}
<p>@message</p>
```

Expressions:

```
<p>Today is @DateTime.Now.DayOfWeek</p>
```

Control Structures:

```
@if (Model.IsLoggedIn)
{
    <p>Welcome back, @Model.Username!</p>
}
else
{
    <p>Please log in.</p>
}
```

Looping:

```
@foreach (var item in Model.Products)
{
    <li>@item.Name - $@item.Price</li>
}
```

Comments:

```
@* This is a Razor comment *@
<!-- This is an HTML comment -->
```

📦 4. Model Data Sharing

PageModel Class

Each Razor Page has an associated PageModel:

```
public class IndexModel : PageModel
{
    public string Message { get; set; }

    public void OnGet()
    {
        Message = "Welcome to Razor Pages!";
    }
}
```

Binding Data to the View

Use `@Model` in the .cshtml to access data from PageModel.

```
<p>@Model.Message</p>
```

Handlers

- Razor Pages use **handler methods**: `OnGet()`, `OnPost()`, etc.

```
public void OnPost()
{
    // Handle form post
}
```

Model Binding

- Razor Pages support automatic **form value binding**:

```
public class ContactModel : PageModel
{
    [BindProperty]
    public string Email { get; set; }

    public void OnPost()
    {
        // Email has the posted value
    }
}
```

TempData & ViewData

- **ViewData**: One-way data transfer during a request.

```
ViewData["Title"] = "My Page";
```

```
<h1>@ViewData["Title"]</h1>
```

- **TempData**: Persists data across redirects.

```
TempData["Success"] = "Data saved!";
```

5. Routing in Razor Pages

- URL paths map directly to file paths.

```
/Pages/Products/List.cshtml → /Products/List
```

- You can override routes using `[@page]` directive:

```
@page "/custom-route"
```

6. Additional Features

Tag Helpers:

Razor uses **Tag Helpers** for clean HTML-like syntax.

```
<form method="post">
    <input asp-for="Email" />
</form>
```

Partial Pages:

Reusable Razor components:

```
@await Html.PartialAsync("_MyPartial")
```

7. Summary

- Razor Pages are great for page-centric apps.
- Keep logic in `PageModel`, UI in `.cshtml`.
- Razor syntax makes combining HTML + C# seamless.
- Supports clean routing, model binding, and dependency injection.

Q & A