Agenda: ASP.NET Core Application using Razor Pages

- First ASP.NET Core App using CLI
- Razor Pages Web App Project Layout
- Adding Model to Razor Pages App

First ASP.NET Core Application

1. Install Visual Studio Code and .NET Core SDK
2. Trust the HTTPS development certificate:

```
dotnet dev-certs https --trust
```

3. Create a new .NET Core project.

```
dotnet new razor -o aspnetcoreapp
```

4. Run the app.

```
cd aspnetcoreapp
dotnet run
```

5. Browse to http://localhost:5000
6. *Open Pages/Index.cshtml and edit its content.*
7. Browse the changes http://localhost:5000/

Understanding REST Protocol

Http Request

1) Request Line	Get URL
2) Request Headers	Name-Value Pairs
3) Posted Data	Form elements name-value pairs(POST)

Http Response

1) Status Line	Status Code / Description
2) Response Headers	Name-Value Pairs
3) Page Body	Page Content...

RESTful Request = HTTP (Protocol) + JSON / XML (Data Format)

Http Protocol Request

1) URL = http://www.demo.com/employee

2) Request method = GET / POST / PUT / MERGE / DELETE

3) Request Body = XML or JSON

Example

- **Get All Employees:** http://www.demo.com/employee + GET
- **Get one Employee:** http://www.demo.com/employee?Id=1 + GET
- **Add an Employee:** http://www.demo.com/employee + POST + {"empName":"Sandeep",.....}
- **Update an Employee:** http://www.demo.com/employee + PUT + id, {"empName":"Sandeep",.....}
- **Delete an Employee:** http://www.demo.com/employee + DELETE + id

Razor Pages Web App

Razor Pages is a server-side, **page-centric** programming model that makes building web UI easier and more productive.
Razor Pages makes coding page-focused scenarios easier and more productive than MVC application model.

Razor Pages use a simple markup syntax called ***Razor*** for embedding server-based code into webpages. Razor syntax is a combination of HTML and C# where the C# code defines the dynamic rendering logic for the page. In a webpage that uses the Razor syntax, there can be two kinds of content: **client content and server code**:
The *Pages* directory is where all Razor Pages are stored and organized in your ASP.NET Core application.

A Razor page has a *.cshtml* file extension. By convention, its associated *PageModel* C# class file uses the same name but with a *.cs* appended. For example, the Razor page *Index.cshtml* has an associated *PageModel* class file for *Index.cshtml.cs*.

Walkthrough:

1. Start Visual Studio 2019 → File → New Project → Visual C# → .NET Core → ASP.NET Core Web Application.
2. Name = FirstAspNetCoreWebApp
3. From dropdown choose .NET Core and ASP.NET Core 3.0, Web Application → OK

Note:

- .NET Core: It's web template for cross platform compatible project that runs on .NET Core framework.
- .NET Framework: This starts a new project that runs on the .NET Framework on Windows.

Project Layout:

The project structure of the ASP.NET Core empty template. The important files/folders in ASP.NET Core:

1. Dependencies

- **Microsoft.NETCore.All**: A set of .NET API's that are included in the default .NET Core application model.
- **Microsoft.ASPNETCore.App**: Provides a default set of APIs for building an ASP.NET Core application. This package requires the ASP.NET Core runtime. This runtime is installed by the .NET Core SDK
- **Microsoft.ASPNETCore.Razor.Design**: Razor is a markup syntax for adding server-side logic to web pages. This package contains MSBuild support for Razor.

2. Properties
   a. **launchSettings.json**: This json file holds project specific settings associated with each debug profile, Visual Studio is configured to use to launch the application, including any environment variables that should be used.
3. **wwwroot**: it stores all the StaticFiles in our project
   a. css
   b. js
   c. lib
   d. favicon.ico
4. **Pages**
   a. _Layout.cshtml
   b. _ValidationScriptsPartial.cshtml: Not included by default. Use the following to include:

```
@{await Html.RenderPartialAsync("_ValidationScriptsPartial"); }
```

   OR

```
<partial name="_ValidationScriptsPartial" />
```

   c. _ViewImports.cshtml
   d. _ViewStart.chtml
   e. Index.cshtml
   f. Error.cshtml
5. **appsettings.json**: is used to define application related settings like connection string, logging settings, or any other custom key which we used to define in web.config file
6. **bundleconfig.json**: Bundling and minifying JavaScript, CSS and HTML files in any project.
7. **Program.cs**: It's an entry point of an Application
8. **Startup.cs**: This is the entry point of every ASP.NET Core application, provides services that application requires.

Index.cshtml

```
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}
@Model.Message
```

Index.cshtml.cs: Is the Page Model object used with the Razor HTML Page.

```
public class IndexModel : PageModel
{
    public string Message { get; set; } = "Hello";
    public void OnGet()
    {
        Message += $" Server time is { DateTime.Now }";
    }
}
```

By convention, the `PageModel` class file has the same name as the Razor Page file with *.cs* appended.

File name and path	Matching URL
/Pages/Index.cshtml	/ or /Index
/Pages/Contact.cshtml	/Contact
/Pages/Store/Index.cshtml	/Store or /Store/Index
/Pages/Store/Contact.cshtml	/Store/Contact

Writing a Basic Form

1. To the project add the below class (Under **ViewModel** Folder)

```
public class Person
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

2. Right Click on Pages Folder → Add Razor Page → PageName = "**Person**" → OK
3. Edit Person.cshtml.cs

```
public class PersonModel : PageModel
{
//Note: Razor Pages, by default, bind properties only with non-GET verbs.
    [BindProperty]
    public Person person { get; set; }
    public string Message = "";
```

```
        public void OnGet()
        {
                person = new Person();
                person.Name = "Sandeep";
                person.Id = 1;
        }
        public void OnPost()
        {
                Message = person.Id + " " + person.Name;
        }
}
```

4. Edit Person.cshtml

```
<form method="POST">
        <div>Id: <input asp-for="person.Id" /></div>
        <div>Name: <input asp-for="person.Name" /></div>
        <input type="submit" value="Submit"/>
</form>
<hr />
@Model.Message
```

5. Visit http://localhost:5000/Person