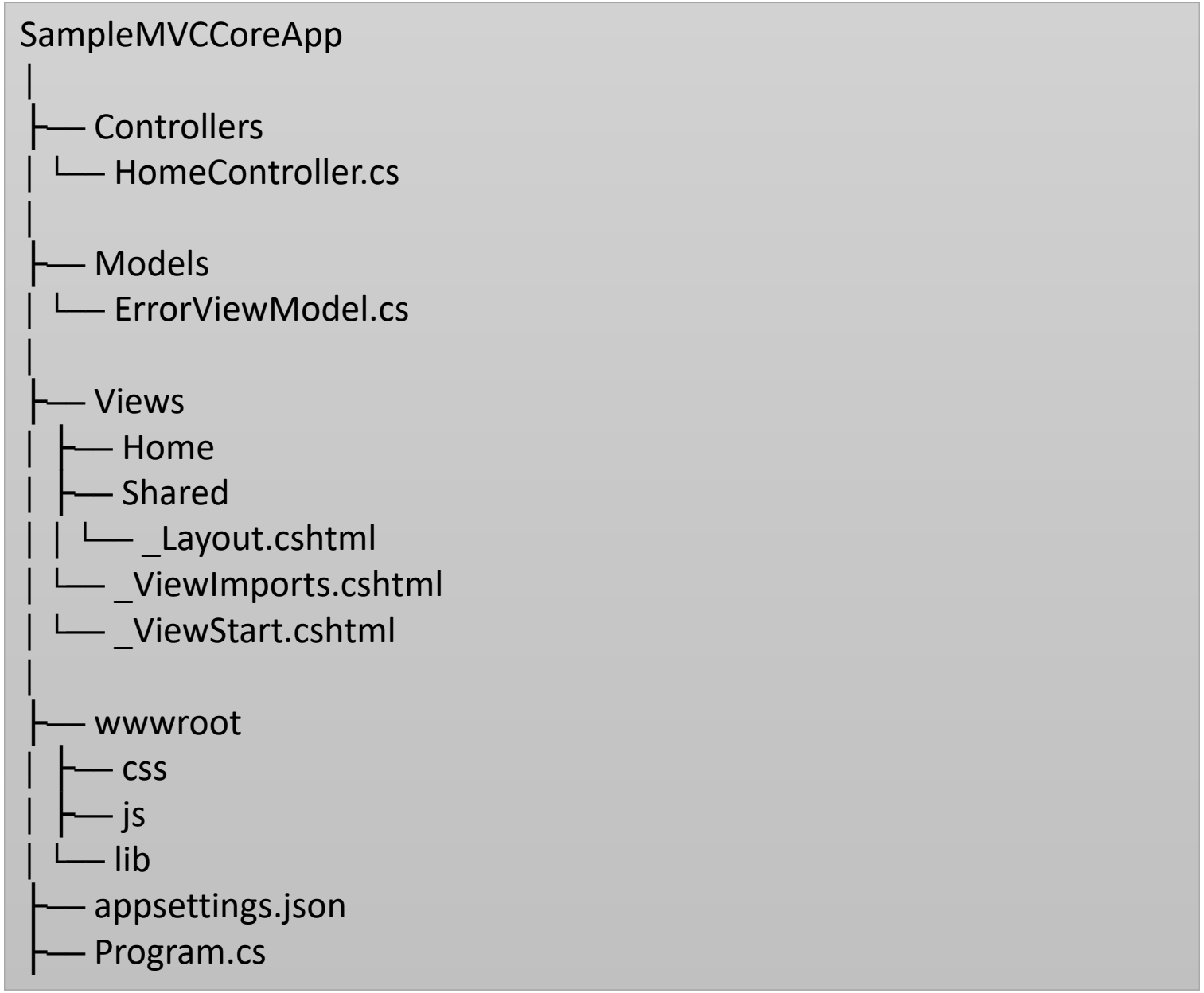
Asp.NET MVC Core

Trainer : Priyanka R Rangole

Email: priyanka.rangole@sunbeaminfo.com

Default MVC Project Structure



MVC Project Structure

Folder Purpose

- **Controllers**: Handles incoming requests
- **Models**: Contains business/data classes
- **Views**: UI layer (Razor files)
- **wwwroot**: Static files (CSS, JS, images)
- **Program.cs**: Application startup and middleware configuration

Web Server – Kestrel

- ASP.NET Core applications are **self-hosted** using **Kestrel web server**
- No need to use IIS explicitly
- Visual Studio allows running with **Kestrel or IIS Express**
- Default selection: **http (Kestrel)**

Running the Application

- Click **Run** (▶) button in Visual Studio
- Application opens in default browser
- Terminal window opens showing:
  - Application URL
  - Hosting environment

Stop Application

- Press **Ctrl + C** in terminal to stop Kestrel server

Output

- Default **responsive MVC web application**
- Automatically adjusts for desktop and mobile view
- Home page rendered using Razor View Engine

•ASP.NET Core project files are **synchronized with physical folders**
•Any file/folder added in project directory:
- Automatically appears in Solution Explorer

•No need to manually add files to project

Project File

•Located under the solution node
•Represents a **single ASP.NET Core MVC project**
•All files inside belong to this project
•Example:
- •SampleMVCCoreApp

1.Project Settings

•Open by double-clicking the project file
•Contains:
•Target .NET framework
•Project folders
•NuGet package references
•Used for project-level configuration

2.Connected Services

•Used to connect external services and APIs
•Supports integration with:
- Azure
- AWS
- Google Cloud
- Third-party services

•Empty by default if no service is added

3.Dependencies

•Shows all libraries used by the project

•Includes:

- NuGet packages
- Framework references
- Project references

•Helps manage install/uninstall of packages

Dependencies – Analyzers

•Used for **static code analysis**

•Helps:

- Enforce coding standards
- Detect code quality issues
- Identify potential bugs

•Can be built-in or from NuGet packages

Dependencies – Frameworks

•Shows target frameworks used by project

•ASP.NET Core MVC uses:

- Microsoft.NETCore.App
- Microsoft.AspNetCore.App

•Press **F4** to view:

- Version
- File path
- Framework details

4.Properties Folder

•Contains **launchSettings.json**

•Used to configure:

- Debug settings
- Launch profiles
- Environment (Dev, Staging, Production)

•Helps control how app runs in Visual Studio

5.wwwroot Folder

•Web root folder for the application

•Stores **static files**:

- CSS
- JavaScript
- Images
- External libraries

•Files are accessed using relative paths

6.Controllers Folder

- Contains controller classes
- Controllers:
  - Handle HTTP requests
  - Process user input
  - Return responses
- Acts as **brain of MVC application**

7.Models Folder

- Contains model classes
- Represents:
  - Business logic
  - Data structure
- Used to pass data between:
  - Controller and View

8.Views Folder

- Contains Razor files (.cshtml)
- Responsible for **UI rendering**
- Uses Razor syntax:
  - HTML + C#
- Includes Shared folder for layouts

9.appsettings.json

- Configuration file in JSON format
- Stores:
  - Application settings
  - Connection strings
  - Environment-based values
- Easy to modify without code changes

10.Program.cs

- Entry point of ASP.NET Core application
- Builds and starts the web application
- Configures:
  - Services
  - Middleware pipeline
- Executes from top to bottom

ASP.NET Core MVC – wwwroot Folder

What is wwwroot?

- Default web root directory in ASP.NET Core
- All static content must be placed inside wwwroot
- Files outside wwwroot **cannot be accessed** via HTTP

- wwwroot is the **web root folder**
- Used to store **static files**
- Static files are directly accessible via browser

Change from Classic ASP.NET

- Earlier ASP.NET:
  - Static files could be served from any folder
- ASP.NET Core:
  - Only files inside wwwroot are served
- Improves **security**

wwwroot Folder Structure

- Recommended subfolders:
  - css – stylesheets
  - js – JavaScript files
  - images – images
  - lib – external libraries
- Helps keep project organized

Static Files

- Static files include:
  - CSS files
  - JavaScript files
  - Images
  - Fonts
  - Library scripts
- These files do not change on server execution

Base URL + folder name + file name

Accessing Static Files

Static files are accessed using **relative URL**

Example:http://localhost:<port>/css/app.css

Why Only wwwroot is Allowed?

•Prevents access to:

- Configuration files
- Source code
- Sensitive data

•Enhances application **security**

UseStaticFiles()

•Must be added in Program.cs

•Enables static file access

```
app.UseStaticFiles();
```

Static Files Middleware

•Static files require middleware

•Middleware enables serving static content

•Without middleware, static files will not load

Program.cs in ASP.NET Core MVC

- **entry point** of the application
- Responsible for:
  - Starting the web server
  - Configuring the application
  - Handling incoming requests

Role of Program.cs

- Application starts execution from Program.cs
- ASP.NET Core apps:
  - Start as **console applications**
  - Then become **web applications**
- Executed when application is run (F5)

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

builder.Services.AddControllersWithViews();

var app = builder.Build();

if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}

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

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

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

// Register services

var app = builder.Build();

// Configure middleware

app.Run();
```

WebApplicationBuilder

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

- Creates WebApplicationBuilder object
- Sets up:
- Kestrel web server
- Content root
- appsettings.json
- Used for configuring services

Kestrel Web Server

- Internal web server of ASP.NET Core
- Automatically configured by `CreateBuilder()`
- No need to manually configure IIS