Module 8

Using Layouts and CSS in ASP.NET Core

Solid Innovator

Module Overview

- Using Layouts
- Using CSS and JavaScript
- Using Bootstrap

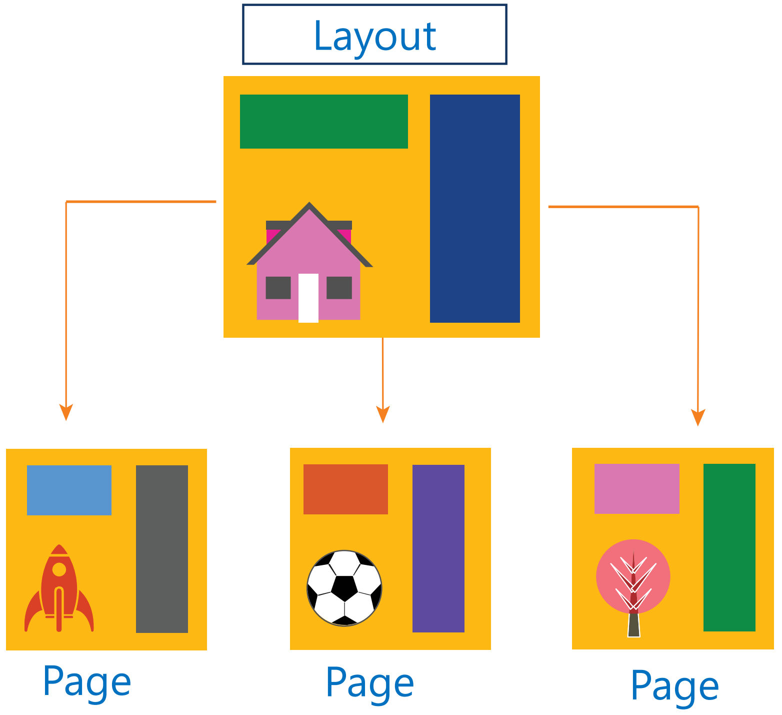
Lesson 1: Using Layouts

- What are Layouts?
- Creating a Layout
- Linking Pages and Layouts
- Using Sections in a Layout
- Demonstration: How to Create a Layout and Link it to a Page Content

What are Layouts?

You can use layouts to:

- Create a style template for a web application
- Define the content layout to share across multiple Pages



Creating a Layout

When you create a layout:

- You should store the layout files in the **\Pages\Shared** folder
- You can use the **@RenderBody** method to place the content of a page in the layout
- You can use the **ViewBag** property to pass information between a page and the layout

A Layout Example

An example of a layout:

```
<!DOCTYPE html>
 <html>
 <head>
     <title>@ViewBag.Title</title>
 </head>
 <body>
      <div>
          @RenderBody()
      </div>
 </body>
 </html>
```

Linking Pages and Layouts

To link pages and layouts:

- You can add the **Layout** directive at the top of the page file
- You can use the **_ViewStart.cshtml** file to define the layout
  - Add the **_ViewStart.cshtml** file to the **\Pages** folder of your project

Using Sections in a Layout

- Use the **@RenderSection** method in a layout

```
@RenderSection("section1")
@RenderBody()
@RenderSection("section2", false)
```

- Use the **@section** directive in a Page

```
@section section1 {
  <div>This is section1 content</div>
}
<div>This is the body</div>
```

Demonstration: How to Create a Layout and Link it to a Page

In this demonstration, you will learn how to:

- Add a **_ViewStart.cshtml** file to a Web application
- Add a layout to Web application
- Use the **@RenderBody** method in the layout
- Use the **@RenderSection** method in the layout
- Link Pages and a layout

Lesson 2: Using CSS and JavaScript

- Importing Styles
- Rendering and Executing JavaScript Code
- Using External Libraries
- Demonstration: How to Use npm to Add a Library

Importing Styles

After importing the CSS file:

- You should modify the layout of the web application by using the **<link>** element
- You can add CSS selectors to define how the styles should be applied:
    - CSS class selector helps specify a style for a group of elements
    - CSS id selector helps specify a style for any unique element in the HTML code

```
.menu
{
    font-weight:bold;
}
```

```
<p class="menu"> this is menu</p>
```

CSS Isolation

Isolate CSS styles to individual pages, views, and components to reduce or avoid:

- Dependencies on global styles that can be challenging to maintain.
- Style conflicts in nested content.

To add a scoped CSS file for a page, place the CSS styles in a companion .cshtml.css file matching the name of the .cshtml file.
The framework rewrites CSS selectors to match markup rendered by the app's pages.
A link to the bundled CSS styles is placed in the app's layout.

```
<link href="{ASSEMBLY NAME}.styles.css" rel="stylesheet">
```

Rendering and Executing JavaScript Code

- You can add JavaScript code to add interactive functionalities to webpages

```
<script>
    function helloWorld() {
        alert('Hello World');
    }
</script>
```

- You can add JavaScript code to web applications by:
  - Adding the JavaScript code to a page
  - Defining the JavaScript code in dedicated JavaScript files

Calling JavaScript Functions

- You can call JavaScript functions by using script blocks:
  - Define the JavaScript function in a script block

```
<script>
   helloWorld()
</script>
```

- You can also use events to trigger JavaScript functions:
  - Use the onclick event to initiate the JavaScript function

```
<input type="button" value="Hello" onclick="helloWorld();" />
```

Using External Libraries

To add a library to your application, you can:

- Download the source files from an official source
- Use a CDN (Content Delivery Network)
- Use a Package Manager
  - NuGet – For server-side libraries
  - Yarn
  - Webpack
  - Bower
  - npm

Using npm to Add Libraries

To start using NPM in an ASP.NET Core application, you should add a **package.json** file to your solution in your project’s root folder:

```
{
  "version": "1.0.0",
  "name": "asp.net",
  "private": true,
  "dependencies": {
    "jquery": "3.3.1"
  },
  "devDependencies": {

  }
}
```

Demonstration: How to Use npm to Add a Library

In this demonstration, you will learn how to:

- Add the jQuery package by using npm
- Add a link to a jQuery file from a layout
- Add a CSS file
- Add a link to the CSS file from a layout

Lab: Using Layouts and CSS in ASP.NET Core

- Exercise 1: Applying a Layout and Link Pages to it
- Exercise 2: Using CSS

Estimated Time: 30 minutes

Module Review and Takeaways

- Review Question
- Best Practices
- Common Issues and Troubleshooting Tips