Table of Contents

Demo: Shop MVC 3, dynamic content, node packages, client-side files

Author: Baifan Zhou
Some explanations are created with the help of ChatGPT.

Introduction

This demo will walk you through the process of adding a dynamic carousel to your Home Page, managing static client-side files, integrating client-side libraries like Bootstrap and jQuery, and configuring middleware for serving static files.

This demo is designed to help you integrate dynamic features into your ASP.NET Core MVC application. You will learn how to:

1. **Add a Dynamic Carousel**:
    - Enhance your Home Page by incorporating a carousel that displays multiple images, creating a more engaging user experience.

2. **Add Static Client-Side Files**:
    - Serve static files like CSS and JavaScript from the `wwwroot` folder, following best practices for structuring your ASP.NET Core MVC application.
    - Create the necessary folders and add images to be displayed in the carousel.

3. **Client-side Library**:
    - Use npm to manage client-side libraries such as Bootstrap and jQuery.
    - Understand the importance of the `package.json` and `package-lock.json` files:
        - **`package.json`** lists the dependencies required by your project.
        - **`package-lock.json`** locks the exact versions of these dependencies, ensuring consistent and secure installations.

4. **Add Middleware for Static Files**:
    - Configure your application to serve static files using `app.UseStaticFiles()`.
    - Update your layout file (`_Layout.cshtml`) to include references to Bootstrap and jQuery, enhancing the functionality and appearance of your web pages.

By following the steps in this demo, you'll learn how to effectively integrate dynamic content and client-side libraries into your ASP.NET Core MVC application, resulting in a more dynamic and user-friendly web application. Happy coding!

Dynamic Carousel

Now we want to add some dynamics to our Home Page.
Add the division code (Line10 – Line27) into the Index.cshtml file.

Index.cshtml

MyShop > Views > Home > Index.cshtml

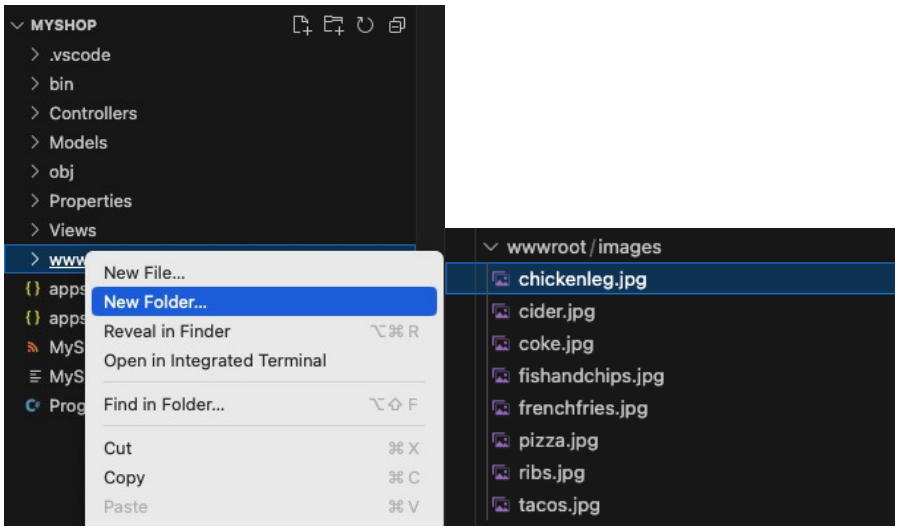
```
1  @{
2      ViewData["Title"] = "Home Page";
3  }
4
5  <div class="text-center">
6      <h1 class="display-4">Welcome to My Shop</h1>
7  </div>
8
9
10 <div id="carouselImages" class="carousel slide" data-bs-ride="true">
11     <div class="carousel-indicators">
12         <button type="button" data-bs-target="#carouselImages" data-bs-slide-to="0" class="active" aria-current="true"
           aria-label="Slide 1"></button>
13         <button type="button" data-bs-target="#carouselImages" data-bs-slide-to="1" aria-label="Slide 2"></button>
14         <button type="button" data-bs-target="#carouselImages" data-bs-slide-to="2" aria-label="Slide 3"></button>
15     </div>
16     <div class="carousel-inner">
17         <div class="carousel-item">
18             <img src="~/images/pizza.jpg" class="d-block w-100" alt="Pizza">
19         </div>
20         <div class="carousel-item active">
21             <img src="~/images/fishandchips.jpg" class="d-block w-100" alt="Fried Chicken Leg">
22         </div>
23         <div class="carousel-item">
24             <img src="~/images/tacos.jpg" class="d-block w-100" alt="Tacos">
25         </div>
26     </div>
27 </div>
```

This will add a carousel that displays three images.
To make this work, we need to add some client-side files and JavaScript libraries.

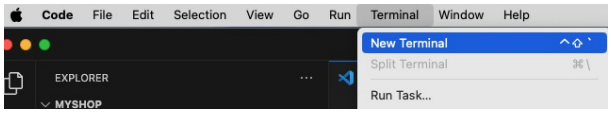
Add Static Client-Side Files and Client-Side library

In a typical ASP.NET Core MVC application, static files like CSS and JavaScript should be served from the `wwwroot` folder.

We begin with adding the images. Create the folder wwwroot under the upper folder. Creat a subfolder “images” under the wwwroot folder. You can also add folder or file by right-clicking the parent folder. Add images (in GitHub repo) to the folder:



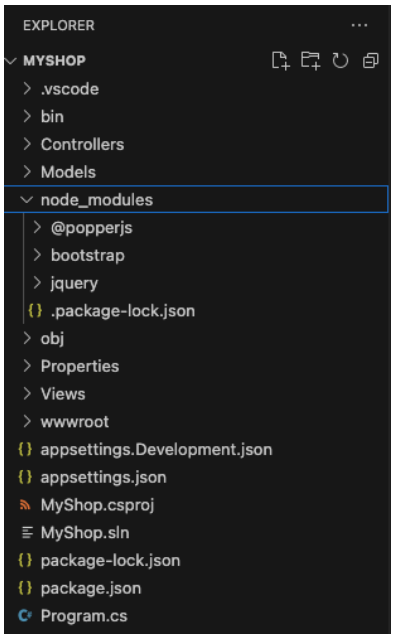
Go the the terminal



Add Client-Side Library (node packages) bootstrap and jquery using npm

```
baifanz@Baifans-MacBook-Pro MyShop % npm install bootstrap

added 2 packages in 441ms

2 packages are looking for funding
  run `npm fund` for details
baifanz@Baifans-MacBook-Pro MyShop % npm install jquery

added 1 package, and audited 4 packages in 472ms

2 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

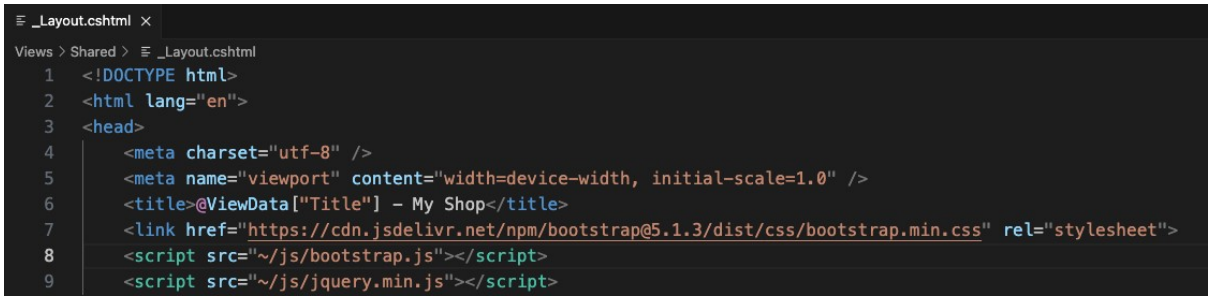
You will see a new folder is created named as “node_modules”.
Also some json files are created: package.json and package-lock.json



The `package.json` file important for Node.js project. For now, it lists the libraries (packages) that your project depends on. These dependencies are installed with specific versions or version ranges.
The `package-lock.json` file is automatically generated for any operations where npm modifies either the `node_modules` tree or `package.json`. Key Roles of `package-lock.json`:
Exact Versions: Locks the versions of the installed packages. This ensures that your project will install the exact same versions of dependencies and sub-dependencies in the future.
Integrity: Contains integrity hashes to ensure the security and integrity of the installed packages.
Performance: Helps npm to optimize the installation process by providing detailed dependency resolution information.

We only need two files from the node_modules. Create a new folder js under wwwroot, and copy the two files `node_modules/bootstrap/dist/js/bootstrap.js` and `node_modules/jquery/dist/jquery.min.js` to `wwwroot/js/`

Add these two files to _Layout.csthml by adding the two script lines (Line8 – Line9). Where ~ denotes the home address of the static files, namely wwwroot.



Add Middleware for Static Files

In the Program.cs, add a middleware of app.UseStaticFiles (Line12), this allows using the files in the folder wwwroot.

```
Program.cs
Program.cs
1   var builder = WebApplication.CreateBuilder(args);
2
3   builder.Services.AddControllersWithViews();
4
5   var app = builder.Build();
6
7   if (app.Environment.IsDevelopment())
8   {
9       app.UseDeveloperExceptionPage();
10  }
11
12  app.UseStaticFiles();
13
14  app.MapDefaultControllerRoute();
15
16  // app.MapControllerRoute(
17  //     name: "default",
18  //     pattern: "{controller=Home}/{action=Index}/{id?}");
19
20  app.Run();
```

Now Start Debugging, and see that the home page is dynamic