Build an ASP.NET Core Service and App with .NET (Core) 5.0 Two-Day Hands-On Lab

Lab 15

This lab walks you through creating a View Component. Prior to starting this lab, you must have completed Lab 14.

Part 1: Adding the Menu View Component

Step 1: Create the View Component Server-Side Code

- Create a new folder in the MVC project named `ViewComponents` and add a new class named `Menu.cs`.
- Add the following using statements:

```
using System.Linq;
using System.Threading.Tasks;
using AutoLot.Dal.Repos.Interfaces;
using AutoLot.Services.ApiWrapper;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.ViewComponents;
```

- Make the class `public` and inherit from `ViewComponent`:

```
namespace AutoLot.Mvc.ViewComponents
{
  public class MenuViewComponent : ViewComponent
  {
  }
}
```

- Add a constructor that takes an instance of the `IMakeRepo` and a `private` variable to hold the instance.

```
private readonly IMakeRepo _makeRepo;
private readonly IApiServiceWrapper _serviceWrapper;
public MenuViewComponent(IApiServiceWrapper serviceWrapper, IMakeRepo makeRepo)
{
  _makeRepo = makeRepo;
  _serviceWrapper = serviceWrapper;
}
```

- Note: Only implement the `Invoke` **or** the `InvokeAsync method, not both (or comment one out)`
- Implement the `Invoke` method (using Make Repository):

```
public IViewComponentResult Invoke()
{
  var makes = _makeRepo.GetAll();
  if (makes == null)
  {
    return new ContentViewComponentResult("Unable to get the makes");
  }
  return View("MenuView", makes);
}
```

- Implement the `InvokeAsync` method (using API Service Wrapper):

```
public async Task<IViewComponentResult> InvokeAsync()
{
  var makes = await _serviceWrapper.GetMakesAsync();
  if (makes == null)
  {
    return new ContentViewComponentResult("Unable to get the makes");
  }
  return View("MenuView", makes);
}
```

Step 2: Update the ViewImports.cshtml File

- To use the `ViewComponent` as a Tag Helper, the assembly must be registered in the `_ViewImports.cshtml` file. Add the following to the end of the file:

```
@addTagHelper *, SpyStore.Hol.MVC
```

Step 3: Create the MenuView partial view

- Add a new folder named `Components` under the `Views\Shared` folder. Add a new folder named `Menu` under the `Components` folder. Add a new partial view named `MenuView.cshtml` in the new folder.
- Update the code to match the following:

```
@model IEnumerable<Make>
<div class="dropdown-menu">
<a class="dropdown-item text-dark" asp-area="" asp-controller="Cars" asp-action="Index">All</a>

@foreach (var item in Model)
{
    <a class="dropdown-item text-dark" asp-controller="Cars" asp-action="ByMake" asp-route-
makeId="@item.Id" asp-route-makeName="@item.Name">@item.Name</a>
}
</div>
```

Step 4: Update the _Menu.cshtml Partial View

- Open the `_Menu.cshtml` file in `Views\Shared\Partials` folder and add the view component as a tag helper before each of the Privacy menu items:

```
<ul class="navbar-nav flex-grow-1">
  <li class="nav-item dropdown">
    <a class="nav-link dropdown-toggle text-dark" data-toggle="dropdown">
      Inventory <i class="fa fa-car"></i>
    </a>
    <vc:menu/>
  </li>
...
</ul>
```

Part 2: Adding the Custom Tag Helpers

Step 1: Stub out the Cars Controller

- Add a new file named `CarsController.cs` in the Controllers directory. Add the following using statements to the top of the file:

```
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
```

- Stub out the base methods on the controller:

```
namespace AutoLot.Mvc.Controllers
{
  [Route("[controller]/[action]")]
  public class CarsController : Controller
  {
    public IActionResult Index()
    {
      return View();
    }
    public IActionResult Details(int? id)
    {
      return View();
    }
    public async Task<IActionResult> Create()
    {
      return View();
    }
    public async Task<IActionResult> Edit(int? id)
    {
      return View();
    }
    public IActionResult Delete(int? id)
    {
      return View();
    }
  }
}
```

- **Note:** This will be completed in the next lab. The controller class and action methods are needed for the Tag Helpers.

Step 2: Create the Base TagHelper

- Create a new folder in the MVC project named `TagHelpers` and add another folder named `Base` under the `TagHelpers` folder. In the `Base` folder, add a new class named `ItemLinkTagHelperBase.cs`. Update the using statements to the following:

```
using AutoLot.Mvc.Controllers;
using AutoLot.Services.Utilities;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Razor.TagHelpers;
```

- Make the class public and abstract, and inherit from `TagHelper`:

```
namespace AutoLot.Mvc.TagHelpers.Base
{
  public abstract class ItemLinkTagHelperBase : TagHelper
  {
  }
}
```

- Add a protected constructor that accepts an instance of `IActionContextAccessor` and `IUrlHelperFactory`. Use them to create an instance of `IUrlHelper`:

```
protected readonly IUrlHelper UrlHelper;
protected ItemLinkTagHelperBase(
  IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
{
  UrlHelper = urlHelperFactory.GetUrlHelper(contextAccessor.ActionContext);
}
```

- Add a public property to hold the item id:

```
public int? ItemId { get; set; }
```

- Implement the `BuildContent` method:

```
protected void BuildContent(TagHelperOutput output,
  string actionName, string className, string displayText, string fontAwesomeName)
{
  output.TagName = "a"; // Replaces <email> with <a> tag
  var target = (ItemId.HasValue)
    ? UrlHelper.Action(actionName, nameof(CarsController).RemoveController(), new {id = ItemId})
    : UrlHelper.Action(actionName, nameof(CarsController).RemoveController());
  output.Attributes.SetAttribute("href", target);
  output.Attributes.Add("class",className);
  output.Content.AppendHtml($@"{displayText} <i class=""fas fa-{fontAwesomeName}""></i>");
}
```

Step 3: Create the Create Item TagHelper

- In the `TagHelpers` folder, add a new class named `ItemCreateTagHelper.cs` and update the usings:

```
using AutoLot.Mvc.Controllers;
using AutoLot.Mvc.TagHelpers.Base;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Razor.TagHelpers;
```

- Make the class public and inherit from `ItemLinkTagHelperBase`:

```
namespace AutoLot.Mvc.TagHelpers
{
  public class ItemCreateTagHelper : ItemLinkTagHelperBase
  {
    public ItemCreateTagHelper(
      IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
         : base(contextAccessor, urlHelperFactory) { }
  }
}
```

- Override Process and call into the base `BuildContent` method:

```
public override void Process(TagHelperContext context, TagHelperOutput output)
{
  BuildContent(output,nameof(CarsController.Create),"text-success","Create New","plus");
}
```

Step 4: Create the Delete Item TagHelper

- In the `TagHelpers` folder, add a new class named `ItemDeleteTagHelper.cs` and update the usings:

```
using AutoLot.Mvc.Controllers;
using AutoLot.Mvc.TagHelpers.Base;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Razor.TagHelpers;
```

- Make the class public and inherit from `ItemLinkTagHelperBase`:

```
namespace AutoLot.Mvc.TagHelpers
{
  public class ItemDeleteTagHelper : ItemLinkTagHelperBase
  {
    public ItemDeleteTagHelper(
      IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
         : base(contextAccessor, urlHelperFactory) { }
  }
}
```

- Override Process and call into the base `BuildContent` method:

```
public override void Process(TagHelperContext context, TagHelperOutput output)
{
  BuildContent(output,nameof(CarsController.Delete),"text-danger","Delete","trash");
}
```

Step 5: Create the Details Item TagHelper

- In the `TagHelpers` folder, add a new class named `ItemDetailsTagHelper.cs` and update the usings:

```
using AutoLot.Mvc.Controllers;
using AutoLot.Mvc.TagHelpers.Base;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Razor.TagHelpers;
```

- Make the class public and inherit from `ItemLinkTagHelperBase`:

```
namespace AutoLot.Mvc.TagHelpers
{
  public class ItemDetailsTagHelper : ItemLinkTagHelperBase
  {
    public ItemDetailsTagHelper(
      IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
         : base(contextAccessor, urlHelperFactory) { }
  }
}
```

- Override Process and call into the base `BuildContent` method:

```
public override void Process(TagHelperContext context, TagHelperOutput output)
{
  BuildContent(output,nameof(CarsController.Details),"text-info","Details","info-circle");
}
```

Step 6: Create the Edit Item TagHelper

- In the `TagHelpers` folder, add a new class named `ItemEditTagHelper.cs` and update the usings:

```
using AutoLot.Mvc.Controllers;
using AutoLot.Mvc.TagHelpers.Base;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Razor.TagHelpers;
```

- Make the class public and inherit from `ItemLinkTagHelperBase`:

```
namespace AutoLot.Mvc.TagHelpers
{
  public class ItemEditTagHelper : ItemLinkTagHelperBase
  {
    public ItemEditTagHelper(
      IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
         : base(contextAccessor, urlHelperFactory) { }
  }
}
```

- Override Process and call into the base `BuildContent` method:

```
public override void Process(TagHelperContext context, TagHelperOutput output)
{
  BuildContent(output,nameof(CarsController.Edit),"text-warning","Edit","edit");
}
```

Step 7: Create the List Items TagHelper

- In the `TagHelpers` folder, add a new class named `ItemListTagHelper.cs` and update the usings:

```
using AutoLot.Mvc.Controllers;
using AutoLot.Mvc.TagHelpers.Base;
using Microsoft.AspNetCore.Mvc.Infrastructure;
using Microsoft.AspNetCore.Mvc.Routing;
using Microsoft.AspNetCore.Razor.TagHelpers;
```

- Make the class public and inherit from `ItemLinkTagHelperBase`:

```
namespace AutoLot.Mvc.TagHelpers
{
  public class ItemListTagHelper : ItemLinkTagHelperBase
  {
    public ItemListTagHelper(
      IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
         : base(contextAccessor, urlHelperFactory) { }
  }
}
```

- Override Process and call into the base `BuildContent` method:

```
public override void Process(TagHelperContext context, TagHelperOutput output)
{
  BuildContent(output,nameof(CarsController.Index),"text-default","Back to List","list");
}
```

Summary

The lab created the Menu view component and the custom tag helpers.

Next steps

In the next part of this tutorial series, you will complete the `CarsController`.