.NET App Dev Hands-On Lab

Razor Pages Lab 5 – View Components, Tag Helpers

This lab walks you through creating a View Component and custom Tag Helpers. Prior to starting this lab, you must have completed Razor Pages Lab 4.

Part 1: Adding the Menu View Component

Step 1: Update the Global Using Statements

- Add the following global using statements to the `GlobalUsings.cs` file in the `AutoLot.Web` project:

```
global using AutoLot.Models.Entities;
global using Microsoft.AspNetCore.Mvc.ViewComponents;
```

Step 2: Create the View Component Server-Side Code

- Create a new folder named `ViewComponents` in the `AutoLot.Web` project and add a new class named `MenuViewComponent.cs`. Update the class to the following:
  Note: Only implement the `Invoke` **or** the `InvokeAsync method, not both`

```
namespace AutoLot.Web.ViewComponents;
public class MenuViewComponent(IMakeRepo makeRepo) : ViewComponent
{
  public async Task<IViewComponentResult> InvokeAsync()
  {
    return await Task.Run<IViewComponentResult>(() =>
    {
      var makes = makeRepo.GetAll().ToList();
      if (!makes.Any())
      {
        return new ContentViewComponentResult("Unable to get the makes");
      }
      return View("MenuView", makes);
    });
  }
}
```

Step 3: Update the ViewImports.cshtml File

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

```
@addTagHelper *, AutoLot.Web
```

Step 4: Create the ViewComponent Partial View

- Add a new folder named `Components` under the `Pages\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-page="/Cars/Index" asp-route-makeId="" asp-
route-makeName="">All</a>

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

Step 5: Update the _Menu.cshtml Partial View

- Open the `_Menu.cshtml` file in `Pages\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>
```

Step 6: Stub out the Cars Index Page

- Add a new directory named `Cars` in the `Pages` directory. Add a new Razor Page – Empty named `Index.cshtml` to the `Cars` directory. Update the code behind to the following:

```
namespace AutoLot.Web.Pages.Cars;

public class IndexModel : PageModel
{
  public string MakeName { get; set; }
  public int? MakeId { get; set; }
  public void OnGet(int? makeId, string makeName)
  {
    MakeId = makeId;
    MakeName = makeName;
  }
}
```

- Update the `Index` view to the following:

```
@page
@model AutoLot.Web.Pages.Cars.IndexModel
@{
  if (Model.MakeId.HasValue)
  {
    <h1>@Model.MakeName</h1>
  }
  else
  {
    <h1>All Makes</h1>
  }
}
```

- **Note:** This page will be completed in the next lab. If you run the app now, the `Inventory` menu will show all the Makes in the drop-down, but none of the links will be functional.

Part 2: Adding the Custom Tag Helpers

Step 1: Update the GlobalUsings.cs file

- Add the following to the `GlobalUsings.cs file`:

```
global using Microsoft.AspNetCore.Mvc.Routing;
global using Microsoft.AspNetCore.Razor.TagHelpers;
```

Step 2: Create the ItemLinkTagHelperBase

- Create a new folder in the `AutoLot.Web` 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 class to the following:

```
namespace AutoLot.Web.TagHelpers.Base;

public abstract class ItemLinkTagHelperBase(
  IActionContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory) : TagHelper
{
  protected readonly IUrlHelper UrlHelper =
    urlHelperFactory.GetUrlHelper(contextAccessor.ActionContext);

  public int? ItemId { get; set; }

  private readonly string _pageName =
    contextAccessor.ActionContext.ActionDescriptor
        .RouteValues["page"]?.Split("/",StringSplitOptions.RemoveEmptyEntries)[0];

  protected string ActionName { get; set; }
  protected void BuildContent(TagHelperOutput output,
    string cssClassName, string displayText, string fontAwesomeName)
  {
    output.TagName = "a";
    var target = ItemId.HasValue
      ? UrlHelper.Page($"/{_pageName}/{ActionName}", new { id = ItemId })
      : UrlHelper.Page($"/{_pageName}/{ActionName}");
    output.Attributes.SetAttribute("href", target);
    output.Attributes.Add("class", cssClassName);
    output.Content.AppendHtml($@"{displayText} <i class=""fa-solid fa-{fontAwesomeName}""></i>");
  }
}
```

- Add the following to the `GlobalUsings.cs` file:

```
global using AutoLot.Web.TagHelpers;
global using AutoLot.Web.TagHelpers.Base;
```

Step 3: Create the ItemCreateTagHelper

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

```
namespace AutoLot.Web.TagHelpers;
public class ItemCreateTagHelper : ItemLinkTagHelperBase
{
  public ItemCreateTagHelper(IActionContextAccessor contextAccessor,
     IUrlHelperFactory urlHelperFactory) : base(contextAccessor, urlHelperFactory)
  {
    ActionName = "Create";
  }
  public override void Process(TagHelperContext context, TagHelperOutput output)
  {
    BuildContent(output, "text-success", "Create New", "plus");
  }
}
```

Step 4: Create the ItemDeleteTagHelper

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

```
namespace AutoLot.Web.TagHelpers;

public class ItemDeleteTagHelper : ItemLinkTagHelperBase
{
  public ItemDeleteTagHelper(IActionContextAccessor contextAccessor,
                             IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = "Delete";
  }
  public override void Process(TagHelperContext context, TagHelperOutput output)
  {
    BuildContent(output, "text-danger", "Delete", "trash");
  }
}
```

Step 5: Create the ItemDetailsTagHelper

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

```
namespace AutoLot.Web.TagHelpers;

public class ItemDetailsTagHelper : ItemLinkTagHelperBase
{
  public ItemDetailsTagHelper(IActionContextAccessor contextAccessor,
                              IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = "Details";
  }
  public override void Process(TagHelperContext context, TagHelperOutput output)
  {
    BuildContent(output, "text-info", "Details", "info-circle");
  }
}
```

Step 6: Create the ItemEditTagHelper

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

```
namespace AutoLot.Web.TagHelpers;

public class ItemEditTagHelper : ItemLinkTagHelperBase
{
  public ItemEditTagHelper(IActionContextAccessor contextAccessor,
                           IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = "Edit";
  }
  public override void Process(TagHelperContext context, TagHelperOutput output)
  {
    BuildContent(output, "text-warning", "Edit", "edit");
  }
}
```

Step 7: Create the ItemListTagHelper

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

```
namespace AutoLot.Web.TagHelpers;

public class ItemListTagHelper : ItemLinkTagHelperBase
{
  public ItemListTagHelper(IActionContextAccessor contextAccessor,
                           IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = "Index";
  }
  public override void Process(TagHelperContext context, TagHelperOutput output)
  {
    BuildContent(output, "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 build the `BasePageModel` and complete the `Cars` pages, which will use the custom tag helpers.