.NET 10 App Dev Hands-On Lab

MVC Lab 5 – View Components, Tag Helpers

This lab walks you through creating a View Component and custom Tag Helpers. Before starting this lab, you must have completed MVC 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.Mvc` 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.Mvc` 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.Mvc.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 located in the `Views` folder. Add the following to the end of the file:

```
@addTagHelper *, AutoLot.Mvc
```

Step 4: Create the ViewComponent 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 5: 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>
```

Step 6: Stub out the Cars Controller

- Add a new class named `CarsController.cs` in the `Controllers` directory. Stub out the following methods on the controller:

```
namespace AutoLot.Mvc.Controllers;
[Route("[controller]/[action]")]
public class CarsController : Controller
{
  [Route("/[controller]")]
  [Route("/[controller]/[action]")]
  [HttpGet]
  public IActionResult Index() => View();
  [HttpGet("{makeId}/{makeName}")]
  public IActionResult ByMake(int makeId, string makeName)
  {
    return View();
  }
  [HttpGet("{id?}")]
  public IActionResult Details(int? id)
  {
    return View();
  }
  [HttpGet]
  public IActionResult Create()
  {
    return View();
  }
  [HttpGet]
  public IActionResult Edit(int? id)
  {
    return View();
  }
  [HttpGet]
  public IActionResult Delete(int? id)
  {
    return View();
  }
}
```

- **Note:** This controller will be completed in the next lab. The controller class and action methods are needed for the `MenuViewComponent` and the Tag Helpers. If you run the app now, you will see the drop-down menu of makes, but none of the menu items link to a working page because the views don't exist yet.

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

Step 2: Create the ItemLinkTagHelperBase

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

```
namespace AutoLot.Mvc.TagHelpers.Base;

public abstract class ItemLinkTagHelperBase : TagHelper
{
  private readonly string _controllerName;
  protected string ActionName { get; set; }
  protected readonly IUrlHelper UrlHelper;
  public int? ItemId { get; set; }

  protected ItemLinkTagHelperBase(IHttpContextAccessor contextAccessor,
    IUrlHelperFactory urlHelperFactory)
  {
    var httpContext = contextAccessor.HttpContext;
    var endpoint = httpContext?.GetEndpoint();
    var actionDescriptor = endpoint?.Metadata.GetMetadata<ActionDescriptor>()
      as ControllerActionDescriptor;
    ActionName = actionDescriptor?.ActionName;
    _controllerName = actionDescriptor?.ControllerName;
    UrlHelper = urlHelperFactory.GetUrlHelper(new ActionContext
    {
      HttpContext = httpContext,
      RouteData = httpContext.GetRouteData(),
      ActionDescriptor = actionDescriptor
    });
  }
  protected void BuildContent(TagHelperOutput output, string cssClassName,
    string displayText, string fontAwesomeName)
  {
    output.TagName = "a"; // Replaces <email> with <a> tag
    var target = (ItemId.HasValue)
      ? UrlHelper.Action(ActionName, _controllerName, new {id = ItemId})
      : UrlHelper.Action(ActionName, _controllerName);
    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.Mvc.TagHelpers;
global using AutoLot.Mvc.TagHelpers.Base;
```

Step 3: Create the ItemCreateTagHelper

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

```
namespace AutoLot.Mvc.TagHelpers;

public class ItemCreateTagHelper : ItemLinkTagHelperBase
{
  public ItemCreateTagHelper(
      IHttpContextAccessor contextAccessor,
      IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = nameof(CarsController.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 class to the following:

```
namespace AutoLot.Mvc.TagHelpers;

public class ItemDeleteTagHelper : ItemLinkTagHelperBase
{
  public ItemDeleteTagHelper(
      IHttpContextAccessor contextAccessor,
      IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = nameof(CarsController.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 class to the following:

```
namespace AutoLot.Mvc.TagHelpers;

public class ItemDetailsTagHelper : ItemLinkTagHelperBase
{
  public ItemDetailsTagHelper(
      IHttpContextAccessor contextAccessor,
      IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = nameof(CarsController.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 class to the following:

```
namespace AutoLot.Mvc.TagHelpers;

public class ItemEditTagHelper : ItemLinkTagHelperBase
{
  public ItemEditTagHelper(
      IHttpContextAccessor contextAccessor,
      IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = nameof(CarsController.Edit);
  }
  public override void Process(TagHelperContext context, TagHelperOutput output)
  {
    BuildContent(output,"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 code to the following:

```
namespace AutoLot.Mvc.TagHelpers;

public class ItemListTagHelper : ItemLinkTagHelperBase
{
  public ItemListTagHelper(
      IHttpContextAccessor contextAccessor, IUrlHelperFactory urlHelperFactory)
    : base(contextAccessor, urlHelperFactory)
  {
    ActionName = nameof(CarsController.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 `BaseCrudController` and complete the `CarsController`.