

Build an ASP.NET Core Service, and App with Core 2.2 Two-Day Hand-On Lab

Lab 13

This lab is the third in a series that creates the ASP.NET Core web application. This lab walks you through managing client-side libraries, and creating/updating the Views for the application. Prior to starting this lab, you must have completed Lab 12.

NOTE: All of the Views in this lab can be copied from \Lab13\Assets\Views.

Part 1: Add the Images and CSS

- 1) Delete the images from the wwwroot\Images folder.
- 2) Add the images from the 2.2\Code\Completed\Lab13\Assets\Images folder.
- 3) Delete site.min.css and site.css from the wwwroot\css folder
- 4) Add the site.css file from the 2.2\Code\Completed\Lab13\Assets\css folder.

Part 2: Manage Client-Side Libraries

Visual Studio:

Library Manager is installed with Visual Studio 2017 15.8 and later. Confirm the installation by opening Tools -> Extensions and Updates and searching for “Microsoft Library Manager”. If it’s not in the list of installed tools, search for it online in the Extensions and Updates dialog.

Command Line:

- 1) Install the Library Manager CLI Tooling as a global tool:

```
dotnet tool install -g Microsoft.Web.LibraryManager.Cli
```

Step 1: Delete the lib directory from the default template

- 1) Delete the wwwroot\lib folder. It will be replaced with files using library manager.

Step 2: Add the libman.json file

- 1) [Visual Studio] Right click on the SpyStore.Mvc project and select Manage Client-Side Libraries. This adds the libman.json file to the root of the project. Right click on the libman.json file and select “Enable restore on build”.

- 1) [Command Line] Create a new libman.json file with the following command:

```
libman init
```

All files copyright Phil Japikse (<http://www.skimedic.com/blog>)

Step 3: Update the libman.json file

- 1) Update the libman.json file to the following JSON (this file can be copied from 2.2\Code\Completed\Lab13\Assets):

```
{
  //https://api.cdnjs.com/libraries/jquery?output=human
  //https://api.cdnjs.com/libraries?output=human
  "version": "1.0",
  "defaultProvider": "cdnjs",
  "defaultDestination": "wwwroot/lib",
  "libraries": [
    {
      "library": "jquery@3.4.1",
      "destination": "wwwroot/lib/jquery",
      "files": ["jquery.js", "jquery.min.js"]
    },
    {
      "library": "jquery-validate@1.19.0",
      "destination": "wwwroot/lib/jquery-validation",
      "files": ["jquery.validate.js", "jquery.validate.min.js", "additional-methods.js",
"additional-methods.min.js"]
    },
    {
      "library": "jquery-validation-unobtrusive@3.2.11",
      "destination": "wwwroot/lib/jquery-validation-unobtrusive",
      "files": ["jquery.validate.unobtrusive.js", "jquery.validate.unobtrusive.min.js" ]
    },
    {
      "library": "font-awesome@4.7.0",
      "destination": "wwwroot/lib/fontawesome"
    },
    {
      "library": "twitter-bootstrap@4.3.1",
      "destination": "wwwroot/lib/bootstrap4",
      "files": [
        "css/bootstrap.css",
        "css/bootstrap.css.map",
        "css/bootstrap.min.css",
        "css/bootstrap.min.css.map",
        "js/bootstrap.bundle.js",
        "js/bootstrap.bundle.js.map",
        "js/bootstrap.bundle.min.js",
        "js/bootstrap.bundle.min.js.map",
        "js/bootstrap.js",
        "js/bootstrap.js.map",
        "js/bootstrap.min.js",
        "js/bootstrap.min.js.map"
      ]
    }
  ]
}
```

Part 3: Update the Layout and ValidationScriptsPartial views

Step 1: Update the _Layout.cshml file

1) Update the stylesheet links and environment tags to the following:

```
<environment include="Development,Local">
  <link rel="stylesheet" href="~/lib/bootstrap4/css/bootstrap.css" asp-append-version="true"/>
  <link rel="stylesheet" href="~/lib/fontawesome/css/font-awesome.css" asp-append-version="true"/>
</environment>
<environment exclude="Development, Local">
  <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
    asp-fallback-href="~/lib/bootstrap/dist/css/bootstrap.min.css"
    asp-fallback-test-class="sr-only" asp-fallback-test-property="position"
    asp-fallback-test-value="absolute" crossorigin="anonymous"
    integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"/>
  <link href="~/lib/fontawesome/css/font-awesome.min.css" rel="stylesheet" asp-append-
version="true" />
</environment>
<link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
```

2) Remove/Comment out the CookieConsentPartial.

```
@*<partial name="_CookieConsentPartial" />*@
```

3) Update the JavaScript environment tags to the following:

```
<environment include="Development, Local">
  <script src="~/lib/jquery/jquery.js" asp-append-version="true"></script>
  <script src="~/lib/bootstrap4/js/bootstrap.bundle.js" asp-append-version="true"></script>
</environment>
<environment exclude="Development, Local">
  <script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-3.3.1.min.js"
    asp-fallback-src="~/lib/jquery/jquery.min.js" asp-fallback-test="window.jQuery"
    crossorigin="anonymous"
    integrity="sha384-tsQFqpEReu7ZLhBV2VZ1Au7zcOV+rXbYlF2cqB8txI/8aZajjp4Bqd+V6D5IgvKT">
  </script>
  <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"
    asp-fallback-src="~/lib/bootstrap4/js/bootstrap.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.fn && window.jQuery.fn.modal"
    crossorigin="anonymous"
    integrity="sha384-ChfqquxZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/JmZQ5stwEULTy">
  </script>
</environment>
<script src="~/js/site.js" asp-append-version="true"></script>
```

- 4) Clear out the markup between the opening <body> tag and the opening <footer> tag. Replace the markup with the following:

```
<header class="navbar navbar-expand-sm navbar-light bg-primary navbar-static-top">
  <div class="container">
    <div class="navbar-header">
      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false"
aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
      </button>
      <a class="navbar-brand d-inline-block d-sm-none" asp-area="" asp-controller="Products" asp-
action="Index">SPY STORE</a>
    </div>
    <nav class="collapse navbar-collapse header-collapse" id="navbarSupportedContent">
      <ul class="nav navbar-nav ml-auto">
        <li class="nav-item dropdown categories-dropdown d-block d-lg-none">
          <a href="#" class="nav-link dropdown-toggle" data-toggle="dropdown">CATEGORIES <span
class="caret"></span></a>
          <ul class="dropdown-menu">
            <li class="nav-item"><a class="nav-link" asp-controller="Home" asp-
action="Privacy">Privacy</a></li>
          </ul>
        </li>
        <partial name="LoginView"/>
      </ul>
    </nav>
  </div>
</header>
<div class="page container">
  <div class="card">
    <nav class="card-header d-none d-sm-block">
      <div class="store-logo">
        <a asp-controller="Products" asp-action="Featured"></a>
      </div>
      <ul class="nav nav-pills d-none d-lg-flex">
        <li class="nav-item"><a class="nav-link" asp-controller="Home" asp-
action="Privacy">Privacy</a></li>
      </ul>
    </nav>
    <div class="card-body">
      @RenderBody()
    </div>
  </div>
</div>
```

Step 2: Update the _ValidationScriptsPartial.cshml file

- 1) Update the entire partial to the following:

```
<environment include="Development">
  <script src="~/lib/jquery-validation/jquery.validate.js"></script>
  <script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.js"></script>
</environment>
<environment exclude="Development">
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-
validate/1.19.0/jquery.validate.min.js"
    asp-fallback-src="~/lib/jquery-validation/jquery.validate.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.validator" crossorigin="anonymous"
    integrity="sha256-F6h55Qw6sweK+t7Si0JX+2bpSAa3b/fn1rVCJvMEj1A=">
</script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/jquery-validation-
unobtrusive/3.2.11/jquery.validate.unobtrusive.min.js"
    asp-fallback-src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.validator &&
window.jQuery.validator.unobtrusive" crossorigin="anonymous"
    integrity="sha256-9GycpJnliUjJDVDqP0UEu/bsm9U+3dnQUH8+3W10vkY=">
</script>
</environment>
```

Part 4: Create the Shared Views and Templates

Step 1: Update the _ViewImports.cshtml File

- 1) Open the _ViewImports.cshtml file in the Views folder. This file is loaded before any Views at or below the level of this file in the directory tree. This enables a central place to include all of the using statements for the Views. Update the using statements to match the following:

```
@using SpyStore.Hol.Mvc
@using SpyStore.Hol.Mvc.Models
@using SpyStore.Hol.Models.Entities
@using SpyStore.Hol.Models.ViewModels
@using System.Collections.Generic
@using Microsoft.AspNetCore.Mvc.Rendering
@using SpyStore.Hol.Mvc.Models.ViewModels
@addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

Step 2: Create the DateTime DisplayTemplate

- 1) Create a new folder named DisplayTemplates under the Views\Shared folder and add a Partial View named DateTime.cshtml in the new folder.
- 2) Clear out the existing code and replace it with the following:

```
@using System.Threading.Tasks
@model DateTime?
@if (Model == null)
{
    @:Unknown
}
else
{
    if (ViewData.ModelMetadata.IsNullableValueType)
    {
        @:@(Model.Value.ToString("d"))
    }
    else
    {
        @:@(((DateTime)Model).ToString("d"))
    }
}
```

Step 3: Create the AddToCartViewModel Editor Template

Views named for types will be rendered when an Html helper is used to edit or display a model (or model property) of that type.

- 1) Create a new folder named EditorTemplates under the Views\Shared folder and add a Partial View named AddToCartViewModel.cshtml in the new folder.

2) Clear out the existing code and replace it with the following:

```
@model AddToCartViewModel
@if (Model.Quantity == 0)
{
    Model.Quantity = 1;
}
<div class="card">
    <div class="card-body">
        <h1 class="d-block">@Html.DisplayFor(x => x.ModelName)</h1>
        <div class="row product-details-container">
            <div class="col-sm-6 product-images">
                
                <div class="key-label">PRODUCT IMAGES</div>
            </div>
            <div class="col-sm-6">
                <div class="price-label">PRICE:</div>
                <div class="price">@Html.DisplayFor(x => x.CurrentPrice)</div>
                <div class="units">Only @Html.DisplayFor(x => x.UnitsInStock) left.</div>
                <div class="product-description">@Html.DisplayFor(x => x.Description)</div>
                <ul class="product-details">
                    <li>
                        <div class="key-label">MODEL NUMBER:</div> @Html.DisplayFor(x => x.ModelNumber)
                    </li>
                    <li>
                        <div class="key-label">CATEGORY:</div> <a asp-controller="Products" asp-
action="ProductList" asp-route-id="@Model.CategoryId">@Model.CategoryName</a>
                    </li>
                </ul>
                <input type="hidden" asp-for="Id" />
                <input type="hidden" asp-for="TimeStamp" />
                <input type="hidden" asp-for="CategoryId" />
                <input type="hidden" asp-for="CategoryName" />
                <input type="hidden" asp-for="CustomerId" value="@ViewBag.CustomerId" />
                <input type="hidden" asp-for="ProductId" />
                <input type="hidden" asp-for="LineItemTotal" />
                <input type="hidden" asp-for="Description" />
                <input type="hidden" asp-for="ModelNumber" />
                <input type="hidden" asp-for="ModelName" />
                <input type="hidden" asp-for="ProductImage" />
                <input type="hidden" asp-for="ProductImageLarge" />
                <input type="hidden" asp-for="ProductImageThumb" />
                <input type="hidden" asp-for="UnitsInStock" />
                <input type="hidden" asp-for="CurrentPrice" />
                <div class="row cart-group">
                    <label for="qty">QUANTITY:</label>
                    <input asp-for="Quantity" class="cart-quantity form-control" />
                    <span asp-validation-for="Quantity" class="text-danger"></span>
                    <button class="btn btn-primary" type="submit">Add to Cart</button>
                </div>
                <script>
                    document.write('<a href="' + document.referrer + '">Go Back</a>');
                </script>
            </div>
        </div>
    </div>
</div>
```

Step 4: Create the AddToCart View

This view doubles as the Product Details view

1) Add a View named AddToCart.cshtml to the Shared folder and update the code to the following:

```
@model AddToCartViewModel
@{
    ViewData["Title"] = @ViewBag.Title;
}
<form method="post"
    asp-controller="Cart"
    asp-action="AddToCart"
    asp-route-customerId="@ViewBag.CustomerId"
    asp-route-productId="@Model.ProductId"
    asp-antiforgery="true"
    id="myForm">
    @Html.EditorForModel()
    <div asp-validation-summary="All" class="text-danger"></div>
</form>
@{
    if (ViewBag.CameFromProducts != null && ViewBag.CameFromProducts)
    {
        <div>
            <a href="#" onclick="window.history.go(-1); return false;">Back to List</a>
        </div>
    }
}
@section Scripts {
    @{
        <partial name="_ValidationScriptsPartial" />
    }
}
```


Step 5: Create the LoginView Partial

- 1) Add a new View named LoginView to the Views\Shared Folder and update the markup to the following (the LoginView partial is already referenced in the _Layout.cshtml view):

```
<li class="nav-item navbar-text">Hello <em>Super Spy</em></li>
<li class="nav-item"><a asp-controller="Cart" asp-action="Index" title="Cart" class="nav-
link"><span class="fa fa-shopping-cart"></span> Cart</a></li>
<li class="nav-item"><a class="nav-link" href="#"><span class="fa fa-lock"></span> LOGIN</a></li>
<li class="nav-item"><a asp-controller="Orders" asp-action="Index" title="Order History"
class="nav-link"><span class="fa fa-tag"></span> Orders</a></li>
<li class="nav-item dropdown search-dropdown">
  <a href="#" class="dropdown-toggle nav-link" data-toggle="dropdown"><span class="fa fa-
search"></span> SEARCH</a>
  <div class="dropdown-menu dropdown-menu-right bg-primary">
    <form asp-controller="Products" asp-action="Search" class="form-inline justify-content-
end" role="search">
      <div class="input-group md-4">
        <label class="sr-only" for="searchString">Search</label>
        <input type="text" id="searchString" name="searchString" class="form-control"
placeholder="SEARCH">
        <div class="input-group-append">
          <button class="btn btn-light" type="submit"><span class="fa fa-
search"></span></button>
        </div>
      </div>
    </form>
  </div>
</li>
```

Part 5: Create the Products Views and Templates

Step 1: Create the ProductViewModel Display Template

- 1) Create a new folder named Products under the Views folder. Create a new folder named DisplayTemplates under the Views\Products folder.

- 2) Add a Partial View named `ProductViewModel.cshtml` in the new folder. Clear out the existing code and replace it with the following:

```
@model ProductViewModel
<div class="col-6 col-sm-4 col-md-3">
    <div class="product">
        
        <div class="price">@Html.DisplayFor(x => x.CurrentPrice)</div>
        <div class="title-container">
            <h5>
                <a asp-controller="Products" asp-action="Details" asp-route-id="@Model.Id">
                    @Html.DisplayFor(x => x.Details.ModelName)</a>
            </h5>
        </div>
        <div><span class="model-number">Model Number:</span> @Html.DisplayFor(x =>
x.Details.ModelNumber)
        </div>
        @if (ViewBag.ShowCategory)
        {
            <a asp-controller="Products" asp-action="ProductList" asp-route-id="@Model.CategoryId"
class="category">@Model.CategoryName</a><br/>
        }
        <a asp-controller="Cart" asp-action="AddToCart" asp-route-productId="@Model.Id" asp-route-
cameFromProducts="true" class="btn btn-primary btn-cart">Add To Cart <span class="fa fa-shopping-
cart"></span></a>
    </div>
</div>
```

Step 2: Create the ProductList View

- 1) Add a View named `ProductList.cshtml` in the `Views\Products` folder. Clear out the existing code and replace it with the following:

```
@model IList<ProductViewModel>
@{
    ViewData["Title"] = ViewBag.Title;
}
<div class="jumbotron">
    @if (ViewBag.Featrued != true)
    {
        <a asp-controller="Products" asp-action="Featured" class="btn btn-info btn-lg">View Featured
Products &raquo;</a>
    }
</div>
<h3>@ViewBag.Header</h3>
<div class="row">
    @for (int x = 0; x < Model.Count; x++)
    {
        var item = Model[x];
        @Html.DisplayFor(model => item)
    }
</div>
```

Part 6: Create the Orders Views

Step 1: Create the Index View

- 1) Create a new folder named Orders under the Views folder, add a View named Index.cshtml in the new folder. clear out the existing code and replace it with the following:

```
@model IList<Order>
<div class="page container">
  <div class="card-body">
    <h3>@ViewBag.Header</h3>
    @for (int x = 0; x < Model.Count; x++)
    {
      var item = Model[x];
      <div class="order-card-heading">
        <div class="row">
          <div class="col-sm-2">
            <label>Order Number</label>
            <a asp-action="Details" asp-route-orderId="@item.Id">
              @Html.DisplayFor(model => item.Id)
            </a>
          </div>
          <div class="col-sm-2">
            <label asp-for="@item.OrderDate"></label><br />
            @Html.DisplayFor(model => item.OrderDate)
          </div>
          <div class="col-sm-2">
            <label asp-for="@item.ShipDate"></label><br />
            @Html.DisplayFor(model => item.ShipDate)
          </div>
          <div class="col-sm-2">
            <label asp-for="@item.OrderTotal"></label><br />
            @Html.DisplayFor(model => item.OrderTotal)
          </div>
          <div class="col-sm-2 order-actions">
            <a asp-action="Details" asp-route-orderId="@item.Id" class="btn btn-primary">Order
Details</a>
          </div>
        </div>
      </div>
    }
  </div>
</div>
```

Step 2: Create the Details View

- 1) Add a View named Details.cshtml in the Views\Orders folder. Clear out the existing code and replace it with the following:

```
@model OrderWithDetailsAndProductInfo
@{
    ViewData["Title"] = "Details";
}
<div class="card-body">
    <h3>@ViewBag.Header</h3>
    <div class="row top-row">
        <div class="col-sm-6">
            <label asp-for="OrderDate"></label>
            <strong>@Html.DisplayFor(model => model.OrderDate)</strong>
        </div>
        <div class="col-sm-6">
            <label asp-for="ShipDate"></label>
            <strong>@Html.DisplayFor(model => model.ShipDate)</strong>
        </div>
    </div>
    <div class="row">
        <div class="col-sm-6">
            <label>Billing Address:</label>
            <address>
                <strong>John Doe</strong><br>
                123 State Street<br>
                Whatever, UT 55555<br>
                <abbr title="Phone">P:</abbr> (123) 456-7890
            </address>
        </div>
        <div class="col-sm-6">
            <label>Shipping Address:</label>
            <address>
                <strong>John Doe</strong><br>
                123 State Street<br>
                Whatever, UT 55555<br>
                <abbr title="Phone">P:</abbr> (123) 456-7890
                <abbr title="Email">E:</abbr> John.Doe@domain.com
            </address>
        </div>
    </div>
    <div class="table-responsive">
        <table class="table table-bordered product-table">
            <thead>
                <tr>
                    <th style="width: 70%;">Product</th>
                    <th class="text-right">Price</th>
                    <th class="text-right">Quantity</th>
                    <th class="text-right">Total</th>
                </tr>
            </thead>
            <tbody>
```

```

@for (int x = 0; x < Model.OrderDetails.Count; x++)
{
    var item = Model.OrderDetails[x];
    <tr>
        <td>
            <div class="product-cell-detail">
                
                <a asp-controller="Products" asp-action="Details"
                    asp-route-id="@item.ProductId" class="h5">
                    @Html.DisplayFor(model => item.ModelName)
                </a>
                <div class="small text-muted">@Html.DisplayFor(model => item.Description)</div>
            </div>
        </td>
        <td class="text-right">@Html.DisplayFor(model => item.UnitCost)</td>
        <td class="text-right">@Html.DisplayFor(model => item.Quantity)</td>
        <td class="text-right">@Html.DisplayFor(model => item.LineItemTotal)</td>
    </tr>
}
</tbody>
<tfoot>
    <tr>
        <th>&nbsp;</th>
        <th>&nbsp;</th>
        <th>&nbsp;</th>
        <th class="text-right">@Html.DisplayFor(model => model.OrderTotal)</th>
    </tr>
</tfoot>
</table>
</div>
<div class="pull-right">
    <a asp-action="Index" class="btn btn-primary">Back to Order History</a>
</div>
</div>

```

Part 7: Create the Cart Views and Templates

Step 1: Create the CartRecordViewModel EditorTemplate

- 1) Create a new folder named Cart under the Views folder. Create a new folder named EditorTemplates under the Views\Cart folder.

- 2) Add a Partial View named `CartRecordViewModel.cshtml` in the new folder, clear out the existing code and replace it with the following:

```
@model CartRecordViewModel
@{
    var formName = "updateCartForm" + Model.Id;
}
<form asp-controller="Cart" asp-action="Update" asp-route-id="@Model.Id" id="@formName"
method="post">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
    <span asp-validation-for="Quantity" class="text-danger"></span>
    <input type="hidden" asp-for="Id" />
    <input type="hidden" asp-for="TimeStamp" />
    <input type="hidden" asp-for="CustomerId" />
    <input type="hidden" asp-for="UnitsInStock" />
    <input type="hidden" asp-for="ProductId" />
    <input type="hidden" asp-for="LineItemTotal" />
    <input asp-for="Quantity" class="cart-quantity text-right" />
    <button class="btn btn-link btn-sm" onClick="updateCart($('#@formName')[0],
'@Url.Action("Update", "Cart", new {id = @Model.Id})", @Model.Id);return false;">Update</button>
</form>
<form asp-controller="Cart" asp-action="Delete" asp-route-id="@Model.Id" id="deleteCartForm"
method="post">
    <input type="hidden" asp-for="Id" />
    <input type="hidden" asp-for="TimeStamp" />
    <button class="btn btn-link btn-sm">Remove</button>
</form>
```

Step 2: Create the Index View

- 1) Add a View named `Index.cshtml` in the `Views\Cart` folder.
- 2) Clear out the existing code and replace it with the following:

```

@model CartViewModel
@{
    ViewData["Title"] = "Index";
    var cartTotal = 0M;
}
<h3>@ViewBag.Header</h3>
<div>
    <div class="table-responsive">
        <table class="table table-bordered product-table">
            <thead>
                <tr>
                    <th style="width: 70%;">Product</th>
                    <th class="text-right">Price</th>
                    <th class="text-right">Quantity</th>
                    <th class="text-right">Available</th>
                    <th class="text-right">Total</th>
                </tr>
            </thead>
            @for (var x = 0; x < Model.CartRecords.Count; x++)
            {
                CartRecordViewModel item = Model.CartRecords[x];
                cartTotal += item.LineItemTotal;
                <partial name="Update" model="item" />
            }
            <tfoot>
                <tr>
                    <th></th>
                    <th>&nbsp;</th>
                    <th>&nbsp;</th>
                    <th>&nbsp;</th>
                    <th>
                        <span id="CartTotal">@Html.FormatValue(cartTotal, "{0:C2}")</span>
                    </th>
                </tr>
            </tfoot>
        </table>
        <form asp-controller="Cart" asp-action="Buy">
            <input asp-for="Customer.Id" type="hidden"/>
            <input asp-for="Customer.EmailAddress" type="hidden"/>
            <input asp-for="Customer.Password" value="hidden" type="hidden"/>
            <div class="pull-right">
                <button class="btn btn-primary">Checkout</button>
            </div>
        </form>
    </div>
</div>

```

3) Add the following JavaScript for the ajax call

@section Scripts

```
{
  <partial Name="_ValidationScriptsPartial"/>
  <script language="javascript" type="text/javascript">
    function updateCart(form, url, id) {
      "use strict";
      var quantity = form.elements["Quantity"].value;
      var timeStamp = form.elements["TimeStamp"].value;
      var productId = form.elements["ProductId"].value;
      var customerId = form.elements["CustomerId"].value;
      var unitsInStock = form.elements["UnitsInStock"].value;
      var token = $('input[name="__RequestVerificationToken"]', form).val();
      var myData = { TimeStamp: timeStamp, Quantity: quantity, ProductId:productId, CustomerId:
customerId, UnitsInStock:unitsInStock };
      var dataWithAntiforgeryToken = $.extend(myData, { '__RequestVerificationToken': token });
      $.ajax({
        url: url,
        type: "POST",
        data: dataWithAntiforgeryToken,
        success: function(data) {
          $("#row_" + id).replaceWith(data);
        },
        error: function(jqXHR, exception) {
          alert('An error occurred: Please reload the page and try again.');
```


Step 3: Create the Update View

- 1) Add a View named Update.cshtml in the Views\Cart folder.
- 2) Clear out the existing code and replace it with the following:

```
@model CartRecordViewModel
<tr id="row_@Model.Id">
  <td>
    <div class="product-cell-detail">
      
      <a class="h5" asp-controller="Products" asp-action="Details" asp-route-
id="@Model.ProductId">@Html.DisplayFor(model => model.ModelName)</a>
      <div class="small text-muted">@Html.DisplayFor(model => model.CategoryName)</div>
      <div class="small text-muted d-none d-sm-block">@Html.DisplayFor(model =>
model.Description)</div>
    </div>
  </td>
  <td class="text-right">@Html.DisplayFor(model => model.CurrentPrice)</td>
  <td class="text-right cart-quantity-row">@Html.EditorForModel()</td>
  <td class="text-right">@Html.DisplayFor(model => model.UnitsInStock)</td>
  <td class="text-right">
    <span id="rawTotal_@Model.ProductId" class="d-none">@Model.LineItemTotal</span>
    <span id="total_@Model.ProductId">@Html.DisplayFor(model => model.LineItemTotal)</span>
  </td>
</tr>
```

Part 8: Bundle and Minify the JavaScript

Step 1: Update the _ViewImports.cshtml file

- 1) Before the @addTagHelper for Microsoft.AspNetCore.Mvc.TagHelpers, add the following line:

```
@addTagHelper *, WebOptimizer.Core
```

Step 2: Update the Startup.cs file

- 1) In the Configure method, add app.UseWebOptimizer *before* the UseStaticFiles call.

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
  //omitted for brevity

  app.UseWebOptimizer();
  app.UseStaticFiles();
  //omitted for brevity
}
```

Step 3: Minimize the JavaScript and CSS files

To minimize specific files or to create bundles, add configuration options into the AddWebOptimizer() method.

- 1) In the `ConfigureServices` method, add `services.AddWebOptimizer`. If not configured any farther, this automatically minimizes all JS and CSS files.
- 2) Add `MinifyCssFiles` and `MinifyJsFiles` to minimize the files. Wrap the call in an if statement to disable bundling and minification in the Development and Local environments, but minify and bundle in any other environments.

```
if (!_env.IsDevelopment() || _env.EnvironmentName == "Local")
{
    services.AddWebOptimizer(false,false);
}
else
{
    services.AddWebOptimizer(options =>
    {
        options.MinifyCssFiles(); //Minifies all CSS files
        //options.MinifyJsFiles(); //Minifies all JS files
        options.MinifyJsFiles("js/site.js");
        //options.AddJavaScriptBundle("js/validations/validationCode.js", "js/validations/**/*.js");
    });
}
```

- 3) The validation scripts are going to be built in Lab 16:

Summary

The lab updated the CSS for the site, managed client-side libraries, added the Views and Templates, and minimized the CSS and JS files.

Next steps

In the next part of this tutorial series, you will create the menu using a View Components.