Build an EF and ASP.NET Core 2.1 App HOL

Lab 8

Welcome to the Build an Entity Framework Core and ASP.NET Core 2.1 Application in a Day Hands-On Lab. This lab walks you through managing client-side libraries and creating the Views for the application.

Prior to starting this lab, you must have completed Lab 7.

Part 1: Add the Images and CSS

Step 1: Add the images

1) Delete the images from the wwwroot\Images folder.
2) Add the images from the Code\Completed\Lab5 folder.

Step 2: Update the CSS for the site

1) Delete site.min.css (bundling and minification will be handled in the next lab)
2) Open the site.css file from the wwwroot\css folder
3) Add the following style information to the end of the file:

```
.jumbotron {
    padding-top: 30px;
    padding-bottom: 30px;
    margin-bottom: 30px;
    color: inherit;
    background-color: #eeeeee;
}
.jumbotron h1,
.jumbotron .h1 { color: inherit; }
.jumbotron p {
    margin-bottom: 15px;
    font-size: 21px;
    font-weight: 200;
}
.jumbotron > hr { border-top-color: #d5d5d5; }
.container .jumbotron,
.container-fluid .jumbotron {
    border-radius: 0px;
    padding-left: 15px;
    padding-right: 15px;
}
```

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

```
.jumbotron .container { max-width: 100%; }
@media screen and (min-width: 768px) {
    .jumbotron {
        padding-top: 48px;
        padding-bottom: 48px;
    }
    .container .jumbotron,
    .container-fluid .jumbotron {
        padding-left: 60px;
        padding-right: 60px;
    }
    .jumbotron h1,
    .jumbotron .h1 { font-size: 63px; }
}
.body-content > .jumbotron {
    background: url(../images/jumbo.png);
    -webkit-background-size: cover;
    -moz-background-size: cover;
    -o-background-size: cover;
    background-size: cover;
    min-height: 200px;
    margin: -15px;
    margin-bottom: 0px;
    text-align: center; }
.body-content > .jumbotron .btn-lg, .body-content > .jumbotron .btn-group-lg > .btn {
    margin-top: 100px; }
@media (min-width: 992px) {
    .body-content > .jumbotron {
        min-height: 380px; }
    .body-content > .jumbotron .btn-lg, .body-content > .jumbotron .btn-group-lg > .btn {
        margin-top: 180px; } }
.body-content .top-row {
    margin-top: 30px; }
```

Part 2: Manage Client-Side Libraries with Library Manager

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.

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 1: Add and update the libman.json file

1) Right click on the MVC project and select Manage Client-Side Libraries. This adds the libman.json file.
2) Right click on the libman.json file and select "Enable restore on build". This will prompt for you to allow another Nuget package (Microsoft.Web.LibraryManager.Build) to be restored into the project.

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

3) Add the following JSON to the libman.json file:

```
{
  //https://api.cdnjs.com/libraries/jquery?output=human
  //https://api.cdnjs.com/libraries?output=human
  "version": "1.0",
  "defaultProvider": "cdnjs",
  "libraries": [
    {
      "library": "jquery@3.3.1",
      "destination": "wwwroot/lib/jquery",
      "files": ["jquery.js", "jquery.min.js"]
    },
    {
      "library": "jquery-validate@1.17.0",
      "destination": "wwwroot/lib/jquery-validate",
      "files": ["jquery.validate.js", "jquery.validate.min.js", "additional-methods.js", 
"additional-methods.min.js"]
    },
    {
      "library": "jquery-validation-unobtrusive@3.2.10",
      "destination": "wwwroot/lib/jquery-validation-unobtrusive",
      "files": ["jquery.validate.unobtrusive.js", "jquery.validate.unobtrusive.min.js" ]
    },
    {
      "library": "twitter-bootstrap@3.3.7",
      "destination": "wwwroot/lib/bootstrap",
      "files": [
        "css/bootstrap.css",
        "css/bootstrap.min.css",
        "js/bootstrap.js",
        "js/bootstrap.min.js",
        "fonts/glyphicons-halflings-regular.eot",
        "fonts/glyphicons-halflings-regular.svg",
        "fonts/glyphicons-halflings-regular.ttf",
        "fonts/glyphicons-halflings-regular.woff",
        "fonts/glyphicons-halflings-regular.woff2"
      ]
    }
  ]
}
```

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

Step 2: Update the _Layout.cshml file

1) Update the following lines to remove “dist” from the path:

```
<link rel="stylesheet" href="~/lib/bootstrap/css/bootstrap.css" asp-append-version="true" />
<link rel="stylesheet"
    href="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/css/bootstrap.min.css"
    asp-fallback-href="~/lib/bootstrap/css/bootstrap.min.css"
    asp-fallback-test-class="sr-only" asp-fallback-test-property="position"
    asp-fallback-test-value="absolute" />
<script src="~/lib/jquery/jquery.js"></script>
<script src="~/lib/bootstrap/js/bootstrap.js"></script>
<script src="https://ajax.aspnetcdn.com/ajax/jquery/jquery-2.2.0.min.js"
    asp-fallback-src="~/lib/jquery/jquery.min.js" asp-fallback-test="window.jQuery"
    crossorigin="anonymous"
    integrity="sha384-K+ctZQ+LL8q6tP7I94W+qzQsfRV2a+AfHIi9k8z8l9ggpc8X+Ytst4yBo/hH+8Fk">
</script>
<script src="https://ajax.aspnetcdn.com/ajax/bootstrap/3.3.7/bootstrap.min.js"
    asp-fallback-src="~/lib/bootstrap/js/bootstrap.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.fn && window.jQuery.fn.modal"
    crossorigin="anonymous"
    integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa">
</script>
```

2) Remove/Comment out the CookieConsentPartial.

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

Step 3: Update the _ValidationScriptsPartial.cshml file

1) Update the following lines to remove “dist” from the path and change jquery-validation to jquery-validate:

```
<script src="~/lib/jquery-validate/jquery.validate.js"></script>
<script src="https://ajax.aspnetcdn.com/ajax/jquery.validate/1.14.0/jquery.validate.min.js"
    asp-fallback-src="~/lib/jquery-validate/jquery.validate.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.validator"
    crossorigin="anonymous"
    integrity="sha384-Fnqn3nxp3506LP/7Y3j/25BlWeA3PXTyT1l78LjECcPaKCV12TsZP7yyMxOe/G/k">
</script>
```

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

Part 4: Create the Shared Views and Templates

NOTE: These files can be copied from the Code\Completed\Lab8\SpyStore_HOL.MVC project in lieu of typing/copy-paste.

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 usings for the Views. Update the using statements to match the following:

```
@using SpyStore_HOL.MVC
@using SpyStore_HOL.Models.Entities
@using SpyStore_HOL.Models.ViewModels
@using SpyStore_HOL.Models.ViewModels.Base
@using System.Collections.Generic
@using Microsoft.AspNetCore.Mvc.Rendering
@using SpyStore_HOL.MVC.ViewModels
@using SpyStore_HOL.MVC.Models;
```

Step 2: Create the DateTime DisplayTemplate

1) Create a new folder named DisplayTemplates under the Views\Shared folder.
2) Add a Partial View named DateTime.cshtml in the new folder.

1) 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

1) Create a new folder named EditorTemplates under the Views\Shared folder.
2) Add a Partial View named AddToCartViewModel.cshtml in the new folder.

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

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

```
@model AddToCartViewModel
@if (Model.Quantity == 0)
{
  Model.Quantity = 1;
}
<h1 class="visible-xs">@Html.DisplayFor(x => x.ModelName)</h1>
<div class="row ">
  <div class="col-sm-6 "><img src="@Url.Content($"~/images/{Model.ProductImageLarge}")" /></div>
  <div class="col-sm-6">
    <h1 class="hidden-xs">@Html.DisplayFor(x => x.ModelName)</h1>
    <div>Price:</div><div>@Html.DisplayFor(x => x.CurrentPrice)</div>
    <div>Only @Html.DisplayFor(x => x.UnitsInStock) left.</div>
    <div></div>
    <div>@Html.DisplayFor(x => x.Description)</div>
    <ul>
      <li> <div>MODEL NUMBER:</div> @Html.DisplayFor(x => x.ModelNumber)</li>
      <li>
        <div>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="CustomerId" />
      <input type="hidden" asp-for="CategoryName" />
      <input type="hidden" asp-for="Description" />
      <input type="hidden" asp-for="UnitsInStock" />
      <input type="hidden" asp-for="UnitCost" />
      <input type="hidden" asp-for="ModelName" />
      <input type="hidden" asp-for="ModelNumber" />
      <input type="hidden" asp-for="TimeStamp" />
      <input type="hidden" asp-for="ProductId" />
      <input type="hidden" asp-for="LineItemTotal" />
      <input type="hidden" asp-for="CurrentPrice" />
      <input type="hidden" asp-for="ProductImage" />
      <input type="hidden" asp-for="ProductImageLarge" />
      <input type="hidden" asp-for="ProductImageThumb" />
      <div class="row">
        <label>QUANTITY:</label>
        <input asp-for="Quantity" class="form-control" />
        <input type="submit" value="Add to Cart" class="btn btn-primary" />
      </div>
      <div asp-validation-summary="ModelOnly" class="text-danger"></div>
      <span asp-validation-for="Quantity" class="text-danger"></span>
    </div>
  </div>
```

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

Step 4: Create the AddToCart View

This view doubles as the Product Details view

1) Add a View named AddToCart.cshtml view Shared folder
2) Update the code to the following:

```
@model AddToCartViewModel
@{
    ViewData["Title"] = @ViewBag.Title;
}
<h3>@ViewBag.Header</h3>
<form method="post"
      asp-controller="Cart"
      asp-action="AddToCart"
      asp-route-customerId="@ViewBag.CustomerId"
      asp-route-productId="@Model.Id">
  @Html.EditorForModel()
</form>
@{
    if (ViewBag.CameFromProducts != null && ViewBag.CameFromProducts)
    {
        <div>
          <a onclick="window.history.go(-1); return false;">Back to List</a>
        </div>
    }
}
@section Scripts {
  @{
      await Html.RenderPartialAsync("_ValidationScriptsPartial");
  }
}
```

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

Part 5: Create the LoginView

Step 1: Create the View

1) Add a new View named LoginView to the Shared Folder
2) Update the markup to the following:

```
<ul class="nav navbar-nav">
    <li><a asp-controller="Cart" asp-action="Index" asp-route-customerId="@ViewBag.CustomerId"
title="Shopping Cart"><span class="glyphicon glyphicon-shopping-cart"></span> Cart</a></li>
    <li><a asp-controller="Orders" asp-action="Index" asp-route-customerId="@ViewBag.CustomerId"
title="Order History"><span class="glyphicon glyphicon-tag"></span> Orders</a></li>
    <li>
        <a href="#" class="dropdown-toggle" data-toggle="dropdown"><span class="glyphicon
glyphicon-search"></span> SEARCH</a>
        <div class="dropdown-menu search-dropdown-menu">
            <form asp-controller="Products" asp-action="Search" class="navbar-form navbar-left"
role="search">
                <div class="input-group">
                    <label class="sr-only" for="searchString">Search</label>
                    <input type="text" id="searchString" name="searchString" class="form-control"
placeholder="SEARCH">
                    <span class="input-group-btn">
                        <button class="btn btn-default" type="submit"><span class="glyphicon
glyphicon-search"></span></button>
                    </span>
                </div>
            </form>
        </div>
    </li>
</ul>
```

Step 2: Update _Layout.cshtml

1) Add the login partial view to the layout using the new Partial View Tag Helper:

```
<div class="navbar-collapse collapse">
<ul class="nav navbar-nav">
  <li><a asp-area="" asp-controller="Products" asp-action="Index">Home</a></li>
</ul>
  <partial name="LoginView"/>
</div>
```

Part 6: Create the Products Views and Templates

NOTE: These files can be copied from the Code\Completed\Lab8\SpyStore_HOL.MVC project in lieu of typing/copy-paste.

Step 1: Create the ProductAndCategoryBase DisplayTemplate

2) Create a new folder named Products under the Views folder.
3) Create a new folder named DisplayTemplates under the Views\Products folder.
4) Add a Partial View named ProductAndCategoryBase.cshtml in the new folder.
5) Clear out the existing code and replace it with the following:

```
@model ProductAndCategoryBase
<div class="col-xs-6 col-sm-4 col-md-3">
  <div>
    <img src="@Url.Content($"~/images/{Model.ProductImage}")" />
    <div>@Html.DisplayFor(x => x.CurrentPrice)</div>
    <div>
      <h5><a asp-controller="Products" asp-action="Details" asp-route-id="@Model.Id">
                @Html.DisplayFor(x => x.ModelName)</a></h5>
    </div>
    <div><span class="text-muted">Model Number:</span> @Html.DisplayFor(x => x.ModelNumber)</div>
    @if (ViewBag.ShowCategory)
    {
        <a asp-controller="Products"
           asp-action="ProductList"
           asp-route-id="@Model.CategoryId">@Model.CategoryName</a><br />
    }
    <a asp-controller="Cart"
       asp-action="AddToCart"
       asp-route-customerId="@ViewBag.CustomerId"
       asp-route-productId="@Model.Id"
       asp-route-cameFromProducts="true"
       class="btn btn-primary"><span class="glyphicon glyphicon-shopping-cart"></span>Add To
Cart</a>
  </div>
</div>
```

Step 2: Create the ProductList View

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

```
@model IList<ProductAndCategoryBase>
@{
    ViewData["Title"] = ViewBag.Title;
}
<div class="jumbotron">
  @if (ViewBag.Featured != 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>
```

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

Part 7: Create the Orders Views

NOTE: These files can be copied from the Code\Completed\Lab8\SpyStore_HOL.MVC project in lieu of typing/copy-paste.

Step 1: Create the Index View

1) Create a new folder named Orders under the Views folder.
2) Add a View named Index.cshtml in the new folder.
3) Clear out the existing code and replace it with the following:

```
@model IList<Order>
<h3>@ViewBag.Header</h3>
@for (int x = 0; x < Model.Count; x++)
{
  var item = Model[x];
    <div>
      <div>
        <div class="row">
          <div class="col-sm-2">
            <label>Order Number</label>
            <a asp-action="Details" asp-route-customerId="@item.CustomerId" asp-route-
orderId="@item.Id">
              @Html.DisplayFor(model => item.Id)
            </a>
          </div>
          <div class="col-sm-2">
            <label asp-for="@item.OrderDate"></label>
            @Html.DisplayFor(model => item.OrderDate)
          </div>
          <div class="col-sm-3">
            <label asp-for="@item.ShipDate"></label>
            @Html.DisplayFor(model => item.ShipDate)
          </div>
          <div class="col-sm-3">
            <label asp-for="@item.OrderTotal"></label>
            @Html.DisplayFor(model => item.OrderTotal)
          </div>
          <div class="col-sm-2">
            <a asp-action="Details" asp-route-customerId="@item.CustomerId" asp-route-
orderId="@item.Id"
               class="btn btn-primary">Order Details</a>
          </div>
        </div>
      </div>
    </div>
}
```

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

Step 2: Create the Details View

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

```
@model OrderWithDetailsAndProductInfo
@{
    ViewData["Title"] = "Details";
}
<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
    </address>
  </div>
</div>
<div class="table-responsive">
  <table class="table table-bordered">
    <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>
                <img src="@Url.Content($"~/images/{item.ProductImageThumb}")" class="pull-left" />
                <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 hidden-xs">
                  @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" asp-route-customerid="@Model.CustomerId" class="btn btn-primary">Back to
Order History</a>
</div>
```

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

Part 8: Create the Cart Views and Templates

NOTE: These files can be copied from the Code\Completed\Lab8\SpyStore_HOL.MVC project in lieu of typing/copy-paste.

Step 1: Create the CartRecordViewModel EditorTemplate

1) Create a new folder named Cart under the Views folder.
2) Create a new folder named EditorTemplates under the Views\Cart folder.
3) Add a Partial View named CartRecordViewModel.cshtml in the new folder.
4) 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-customerId="@Model.CustomerId"
      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="CustomerId" />
    <input type="hidden" asp-for="UnitsInStock" />
    <input type="hidden" asp-for="ProductId" />
    <input type="hidden" asp-for="LineItemTotal" />
    <input type="hidden" name="@nameof(Model.TimeStampString)" id="@nameof(Model.TimeStampString)"
value="@Model.TimeStampString" />
    <input type="hidden" name="@nameof(Model.LineItemTotal)" id="@nameof(Model.LineItemTotal)"
value="@Model.LineItemTotal" />
    <input asp-for="Quantity" />
    <button class="btn btn-link btn-sm"
onClick="updateCart($('#@formName')[0],'@Url.Action("Update","Cart",new {customerId =
@Model.CustomerId,id = @Model.Id})',@Model.Id);return false;">Update</button>
</form>
<form asp-controller="Cart" asp-action="Delete" asp-route-customerId="@Model.CustomerId"
      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>
```

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

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>
  <table class="table table-bordered">
    <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++)
    {
      var item = Model.CartRecords[x];
      cartTotal += item.LineItemTotal;
        @Html.Partial("Update", 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>
</div>
```

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

```
@section Scripts
{
  @{
    await Html.RenderPartialAsync("_ValidationScriptsPartial");
  }
  @*//https://developercommunity.visualstudio.com/content/problem/314560/visual-studio-crashes-
since-the-last-update.html#reply-316171*@
  <script language="javascript" type="text/javascript">
    function updateCart(form, url, id) {
      var quantity = form.elements["Quantity"].value;
      var timeStamp = form.elements["TimeStampString"].value;
      var token = $('input[name="__RequestVerificationToken"]', form).val();
      var myData = {TimeStampString: timeStamp, Quantity: quantity};
      var dataWithAntiforgeryToken = $.extend(myData, { '__RequestVerificationToken': token });
      $.ajax({
        url: url,
        type: "POST",
        data: dataWithAntiforgeryToken,
        success: function (data) {
          $("#row_"+id).replaceWith(data);
        },
        error: function () {
          alert('An error occurred: Please reload the page and try again.');
        }
      })
      .done(updateCartPrice);
    }

    function getSum(total, num) {
      "use strict";
      return total + Math.round(num.innerText * 100)/100;
    }

    function updateCartPrice() {
      "use strict";
      var list = $('span[id^="rawTotal"]');
      var total = $.makeArray(list).reduce(getSum, 0);
      $('#CartTotal')[0].innerText = '$' +
parseFloat(total).toFixed(2).toString().replace(/(\d)(?=(\d\d\d)+(?!\d))/g, "$1,");
      }
  </script>
}
```

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">
      <img src="@Url.Content($"~/images/{Model.ProductImageThumb}")" class="pull-left" />
      <a class="h5" asp-controller="Products" asp-action="Details"
         asp-route-id="@Model.ProductId">@Html.DisplayFor(model => model.ModelName)</a>
        <div class="small">@Html.DisplayFor(model => model.CategoryName)</div>
        <div class="small text-muted">@Html.DisplayFor(model => model.Description)</div>
      </div>
    </td>
  <td class="text-right">
    @Html.DisplayFor(model => model.CurrentPrice)
  </td>
  <td class="text-right">
    @Html.EditorForModel()
  </td>
  <td class="text-right">
    @Html.DisplayFor(model => model.UnitsInStock)
  </td>
  <td class="text-right">
    <span id="rawTotal_@Model.ProductId" class="hidden">@Model.LineItemTotal</span>
    <span id="total_@Model.ProductId">@Html.DisplayFor(model => model.LineItemTotal)</span>
  </td>
</tr>
```

NOTE: If Visual Studio hangs and then restarts when editing JavaScript, there is a known bug with the TypeScript tooling that can be resolved with this workaround:
https://developercommunity.visualstudio.com/content/problem/314560/visual-studio-crashes-since-the-last-update.html#reply-316171

Part 9: Delete the Home Controller and Views

1) Delete the HomeController as it is not used in this application.
2) Delete the Home Views folder, as those views are not used in this application.

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

Part 10: Bundle and Minify the JavaScript

Step 1: Add LigerShark.WebOptimizer.Core

The WebOptimizer .NET Core package plugs into the ASP.NET Core pipeline for bundling and minification.

1) Right click on the SpyStore_HOL.MVC project and select Manage Nuget Packages
2) Enter LigerShark.WebOptimizer.Core into the search box
3) Install the package (current version 1.0.215 or greater)

Step 2: Update the _ViewImports.cshtml file

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

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

Step 3: Update the Startup.cs file

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

Step 4: Update the _Layout.cshtml file

1) Remove “min” from the site.js file:

```
<script src="~/js/site.js" asp-append-version="true"></script>
```

2) Remove “min” from the site.css file:

```
<link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
```

Step 5: 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. services.AddWebOptimizer(options =>

```
{
  options.MinifyCssFiles(); //Minifies all CSS files
  //options.MinifyJsFiles(); //Minifies all JS files
  options.MinifyJsFiles("js/site.js"); //Minifies just one file
});
```

3) Bundling can be accomplished by using AddJavaScriptBundle to bundle JavaScript files and AddCssBundle to bundle CSS files. The first argument is the bundle name, then the files to be bundled (with globbing support). You will do this in Lab 11:

Summary

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

Next steps

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

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