Build an EF and ASP.NET Core 2 App HOL

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

Prior to starting this lab, you must have completed Lab 4.

All labs and files are available at https://github.com/skimedic/dotnetcore_hol.

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 Assets download folder into the wwwroot\Images folder.

Step 2: Update the CSS for the site

1) Open the site.css file from the wwwroot\css folder
2) 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;
}
```

```
.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 LibraryManager

Library manager will be part of the Visual Studio tooling with version 15.7. Until then, download (or fork) https://github.com/aspnet/LibraryManager, build the solution, and install Microsoft.Web.LibraryManager.vsix.

Step 1: Add and update the library.json file

1) Right click on the MVC project and select Manage Client-Side Libraries. This adds the library.json file.
2) Add the following JSON to the library.json file:

```
{
  "version": "1.0",
  "defaultProvider": "cdnjs",
  "packages": [
    //https://api.cdnjs.com/libraries/jquery?output=human
    //https://api.cdnjs.com/libraries?output=human
    {
      "id": "jquery@2.2.4",
      "path": "wwwroot/lib/jquery",
      "files": ["jquery.js", "jquery.min.js"]
    },
    {
      "id": "jquery-validate@1.17.0",
      "path": "wwwroot/lib/jquery-validate",
      "files": ["jquery.validate.js","jquery.validate.min.js","additional-methods.js","additional-
methods.min.js"]
    },
    {
      "id": "jquery-validation-unobtrusive@3.2.6",
      "path": "wwwroot/lib/jquery-validation-unobtrusive",
      "files": ["jquery.validate.unobtrusive.js",
        "jquery.validate.unobtrusive.min.js"]
    },
    {
      "id": "twitter-bootstrap@3.3.7",
      "path": "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"
      ]
    }
  ]
}
```

Step 2: Update the _Layout.cshml file

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

```
<link rel="stylesheet" href="~/lib/bootstrap/dist/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>
```

Step 2: 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/dist/jquery.validate.min.js"
    asp-fallback-test="window.jQuery && window.jQuery.validator"
    crossorigin="anonymous"
    integrity="sha384-Fnqn3nxp3506LP/7Y3j/25BlWeA3PXTyT1l78LjECcPaKCV12TsZP7yyMxOe/G/k">
</script>
```

Part 2a: Manage Client-Side Libraries with Bower

Bower is still supported and can be used until LibraryManager is live.

Step 1: Add the Bower.json and .bowerrc files

2) Right click on the MVC project and select Add -> New Item -> JSON file, and name it bower.json.
3) Right click on the project and select Add -> New Item -> JSON File and name if .bowerrc.
   Note: The period (.) in front of the name is required.

Step 2: Update the .bowerrc file

The .bowerrc file configures where Bower managed packages will go into your project.

1) Add the following JSON to the file:

```
{
  "directory": "wwwroot/lib"
}
```

Step 3: Update the bower.json file

1) Open the newly added bower.json file.
2) Add the following base JSON to the file:

```
{
  "name": "asp.net",
  "private": true,
  "dependencies": {
  }
}
```

3) For each installed library in the wwwroot/lib directory, open the bower.json file, find the version, and add the name and version into a new line in the dependencies node:
   Note: The name is the same as the folder name.

```
"bootstrap": "3.3.7",
"jquery": "2.2.0",
"jquery-validation": "1.14.0",
"jquery-validation-unobtrusive": "3.2.6",
```

4) Add the additional line in the dependencies node:

```
"jquery-ajax-unobtrusive": "3.2.4"
```

5) Right click on the bower.json file and select “Restore packages”

Part 3: Update the Validation Scripts Partial View

1) Open the _ValidationScriptsPartial.cshtml file from Views\Shared folder of the MVC project.
2) Add the following line to the include="Development" section of the view:

```
<script src="~/lib/jquery-ajax-unobtrusive/jquery.unobtrusive-ajax.js"></script>
```

3) Add the following line to the exclude="Development" section of the view:

```
<script src="~/lib/jquery-ajax-unobtrusive/jquery.unobtrusive-ajax.min.js"></script>
```

Part 4: Create the Shared Views and Templates

Step 1: 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.
3) 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 2: 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.

3) Clear out the existing code and replace it with the following:
Note: The Tag Helpers in this and all code listings are shown in **bold**.

```
@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>
```

Step 3: 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");
  }
}
```

Part 5: Create the Products Views and Templates

Step 1: Create the ProductAndCategoryBase DisplayTemplate

1) Create a new folder named Products under the Views folder.
2) Create a new folder named DisplayTemplates under the Views\Products folder.
3) Add a Partial View named ProductAndCategoryBase.cshtml in the new folder.

4) 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>
```

Part 6: Create the Orders Views

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>
}
```

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> </th>
        <th> </th>
        <th> </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>
```

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.
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
<form asp-controller="Cart" asp-action="Update" asp-route-customerId="@Model.CustomerId"
    asp-route-id="@Model.Id" id="updateCartForm" method="post" data-ajax="true"
    data-ajax-mode="REPLACE-WITH" data-ajax-update="#row_@Model.Id" data-ajax-method="POST"
    data-ajax-success="success" data-ajax-failure="error" data-ajax-complete="complete">
  <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="CategoryName" />
  <input type="hidden" asp-for="Description" />
  <input type="hidden" asp-for="UnitsInStock" />
  <input type="hidden" asp-for="ModelName" />
  <input type="hidden" asp-for="ModelNumber" />
  <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 type="hidden" asp-for="CurrentPrice" />
  <input type="hidden" asp-for="ProductImage" />
  <input type="hidden" asp-for="ProductImageLarge" />
  <input type="hidden" asp-for="ProductImageThumb" />
  <input asp-for="Quantity" />
  <button class="btn btn-link btn-sm">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>
```

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> </th>
        <th> </th>
        <th> </th>
        <th><span id="CartTotal">@Html.FormatValue(cartTotal, "{0:C2}")</span></th>
      </tr>
    </tfoot>
  </table>
</div>
```

```
@section Scripts
{
  @{
      await Html.RenderPartialAsync("_ValidationScriptsPartial");
  }
  <script language="javascript" type="text/javascript">
        function success(data, textStatus, jqXHR) {
            "use strict";
            updateCartPrice();
        }
        function error(jqXhr, textStatus, errorThrown) {
            "use strict";
            alert('An error occurred: Please reload the page and try again.');
        }
        function complete(jqXhr, textStatus) {
            "use strict";
            //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,");
        }
        $(function () {
            //updateCartPrice();
        });
  </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>
```

Part 8: Delete the Home Views

1) Delete the Home Views folder, as they are not used in this application.

Summary

The lab updated the CSS for the site, and created the Views and Templates.

Next steps

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