Build an ASP.NET Core MVC App with EF Core One-Day Hands-On Lab

Lab 13

This walks you through creating and/or updating the Views for the Cars Controller. Prior to starting this lab, you must have completed Lab 12.

Note: Adjust any directory separators to your OS (e.g. \ for Windows, / for Mac/Linux)

Part 1: The RazorSyntax View

- Create a new view named `RazorSyntax` in the Home and update it to the following:

```
@model Car
@{
    ViewData["Title"] = "Razor Syntax";
}

<h1>Razor Syntax</h1>

@for (int i = 0; i < 15; i++)
{
  //do something
}
@{
    //Code Block
    var foo = "Foo";
    var bar = "Bar";
    var htmlString = "<ul><li>one</li><li>two</li></ul>";
}
@foo<br />
@htmlString<br />
@foo.@bar<br />
@foo.ToUpper()<br/>
@Html.Raw(htmlString)
<hr />
@{
    @:Straight Text
    <div>Value:@Model.Id</div>
    <text>
        Lines without HTML tag
    </text>
    <br />
}

<hr/>
@*
    Multiline Comments
    Hi.
*@
```

```
Email Address Handling:<br/>
foo@foo.com = foo@foo.com<br/>
@@foo<br/>
test@foo = test@foo<br/>
test@(foo) = testFoo<br/>
<hr/>
@functions {
    public static IList<string> SortList(IList<string> strings)  {
        var list = from s in strings orderby s select s;
        return list.ToList();
    }
}
-------------------------------------------------------------
@{
    var myList = new List<string> {"C", "A", "Z", "F"};
    var sortedList = SortList(myList); //MyFunctions.SortList(myList)
}
@foreach (string s in sortedList)
{
    @s@: 
}
<hr/>
@{
    Func<dynamic, object> b = @<strong>@item</strong>;
}
This will be bold: @b("Foo")

@Html.DisplayForModel()
@Html.DisplayForModel("CarWithColors")
<hr/>
@Html.EditorForModel()
<hr/>
<a asp-controller="Cars" asp-action="Details" asp-route-id="@Model.Id">@Model.PetName</a>
```

- Update the _Menu.cshtml partial view to include a menu item for the new view:

```
<li class="nav-item">
  <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="RazorSyntax">Razor
Syntax <i class="fas fa-cut"></i></a>
</li>
```

- Add a new action method named `RazorSyntax` in the `HomeController`:

```
[HttpGet]
public IActionResult RazorSyntax([FromServices] ICarRepo carRepo)
{
  var car = carRepo.Find(1);
  return View(car);
}
```

Part 2: Update the Index View

- Update the `Home/Index.cshtml` view to use the `DealerInfo` passed in from the action method. Update the view to the following:

```
@model AutoLot.Services.ViewModels.DealerInfo
@{
    ViewData["Title"] = "Home Page";
}

<div class="text-center">
    <h1 class="display-4">Welcome to @Model.DealerName</h1>
    <p class="lead">Located in @Model.City, @Model.State</p>
</div>
```

Part 3: The Car Views

Step 1: The Partial and Template Views

- Create a new folder named `Cars` under the `Views` folder. In this folder, create three new folders, `DisplayTemplates`, `EditorTemplates`, and `Partials`.
- Add a new view named `Car.cshtml` under the `View\Cars\DisplayTemplates` folder. Update the markup to the following:

```
@model AutoLot.Models.Entities.Car
<dl class="row">
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.MakeId)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.MakeNavigation.Name)
    </dd>
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.Color)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.Color)
    </dd>
    <dt class="col-sm-2">
        @Html.DisplayNameFor(model => model.PetName)
    </dt>
    <dd class="col-sm-10">
        @Html.DisplayFor(model => model.PetName)
    </dd>
</dl>
```


- Add a new view named `CarWithColors.cshtml` under the `View\Cars\DisplayTemplates` folder. Update the markup to the following:

```
@model Car
<hr />
<div>
  <dl class="row">
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.PetName)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.PetName)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.MakeNavigation)
    </dt>
    <dd class="col-sm-10">
      @Html.DisplayFor(model => model.MakeNavigation.Name)
    </dd>
    <dt class="col-sm-2">
      @Html.DisplayNameFor(model => model.Color)
    </dt>
    <dd class="col-sm-10" style="color:@Model.Color">
      @Html.DisplayFor(model => model.Color)
    </dd>
  </dl>
</div>
```

- Add a new view named `Car.cshtml` under the `View\Cars\EditorTemplates` folder. Update the markup to the following:

```
@model Car
<div asp-validation-summary="All" class="text-danger"></div>
<div >
    <label asp-for="PetName" class="col-form-label"></label>
    <input asp-for="PetName" class="form-control"/>
    <span asp-validation-for="PetName" class="text-danger"></span>
</div>
<div >
    <label asp-for="MakeId" class="col-form-label"></label>
    <select asp-for="MakeId" class="form-control" asp-items="ViewBag.LookupValues"></select>
    <span asp-validation-for="MakeId" class="text-danger"></span>
</div>
<div >
    <label asp-for="Color" class="col-form-label"></label>
    <input asp-for="Color" class="form-control"/>
    <span asp-validation-for="Color" class="text-danger"></span>
</div>
```


- Add a new view named `_CarListPartial.cshtml` under the `View\Cars\Partials` folder. Update the markup to the following:

```
@model IEnumerable<Car>
@{
  var showMake = true;
  if (bool.TryParse(ViewBag.ByMake?.ToString(), out bool byMake)) { showMake = !byMake; }
}
<p>
  <item-create></item-create>
</p>
<table class="table">
  <thead>
    <tr>
    @if (showMake)
    {
      <th>@Html.DisplayNameFor(model => model.MakeId) </th>
    }
      <th>@Html.DisplayNameFor(model => model.Color)</th>
      <th>@Html.DisplayNameFor(model => model.PetName)</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
  @foreach (var item in Model)
  {
    <tr>
    @if (showMake)
    {
      <td>@Html.DisplayFor(modelItem => item.MakeNavigation.Name)</td>
    }
      <td>@Html.DisplayFor(modelItem => item.Color)</td>
      <td>@Html.DisplayFor(modelItem => item.PetName)</td>
      <td>
        <item-edit item-id="@item.Id"></item-edit> |
        <item-details item-id="@item.Id"></item-details> |
        <item-delete item-id="@item.Id"></item-delete>
      </td>
    </tr>
  }
  </tbody>
</table>
```

Step 2: Create the Index and ByMake views

- Add a new view named `Index.cshtml` to the `Views\Cars` folder and update the markup to the following:

```
@model IEnumerable<Car>
@{
    ViewData["Title"] = "Index";
}
<h1>Vehicle Inventory</h1>
<partial name="Partials/_CarListPartial" model="@Model"/>
```

- Add a new view named `ByMake.cshtml` to the `Views\Cars` folder and update the markup to the following:

```
@model IEnumerable<Car>
@{
    ViewData["Title"] = "Index";
}
<h1>Vehicle Inventory for @ViewBag.MakeName</h1>
@{
    var mode = new ViewDataDictionary(ViewData) {{"ByMake", true}};
}
<partial name="Partials/_CarListPartial" model="Model" view-data="@mode"/>
```

Step 3: Create the Details view

- Add a new view named `Details.cshtml` to the `Views\Cars` folder and update the markup to the following:

```
@model Car
@{
  ViewData["Title"] = "Details";
}
<h1>Details for @Model.PetName</h1>
@Html.DisplayForModel()
@*@Html.DisplayForModel("CarWithColors")*@
<div>
  <item-edit item-id="@Model.Id"></item-edit>
  <item-delete item-id="@Model.Id"></item-delete>
  <item-list></item-list>
</div>
```

Step 4: Create the Create view

- Add a new view named `Create.cshtml` to the `Views\Cars` folder and update the markup to the following:

```
@model Car
@{ ViewData["Title"] = "Create"; }
<h1>Create a New Car</h1>
<hr/>
<form asp-controller="Cars" asp-action="Create">
<div class="row">
  <div class="col-md-4">
    <div asp-validation-summary="ModelOnly" class="text-danger"></div>
       @Html.EditorForModel()
    </div>
  </div>
  <div class="d-flex flex-row mt-3">
    <button type="submit" class="btn btn-success">Create <i class="fas fa-
plus"></i></button>  |  
    <item-list></item-list>
  </div>
</form>
@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```


Step 5: Create the Edit view

- Add a new view named `Edit.cshtml` to the `Views\Cars` folder and update the markup to the following:

```
@model Car
@{ ViewData["Title"] = "Edit"; }
<h1>Edit @Model.PetName</h1>
<hr />
<form asp-area="" asp-controller="Cars" asp-action="Edit" asp-route-id="@Model.Id">
  <div class="row">
    <div class="col-md-4">@Html.EditorForModel()</div>
  </div>
  <div class="d-flex flex-row mt-3">
     <input type="hidden" asp-for="Id" />
     <input type="hidden" asp-for="TimeStamp" />
     <button type="submit" class="btn btn-primary">Save <i class="fas fa-
save"></i></button>  |  
     <item-list></item-list>
  </div>
</form>
@section Scripts {
    <partial name="_ValidationScriptsPartial" />
}
```

Step 6: Create the Delete view

- Add a new view named `Delete.cshtml` to the `Views\Cars` folder and update the markup to the following:

```
@model Car
@{ ViewData["Title"] = "Delete";}
<h1>Delete @Model.PetName</h1>
<h3>Are you sure you want to delete this car?</h3>
<div>
    @Html.DisplayForModel()
    <form asp-action="Delete">
      <input type="hidden" asp-for="Id" />
      <input type="hidden" asp-for="TimeStamp" />
      <button type="submit" class="btn btn-danger">Delete <i class="fas fa-
trash"></i></button>  |  
      <item-list></item-list>
  </form>
</div>
```

Summary

The lab created the views for the `CarsController`.

Next Steps

The next lab creates an admin area for Make maintenance.