Build an EF and ASP.NET Core 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 View Models and the Controllers.

Prior to starting this lab, you must have completed Lab 6.

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

Part 1: Create the ViewModels

Step 1: Create the base ViewModel

1) Create a new folder name ViewModels in the MVC project. Create a new folder named Base under ViewModels.
2) Add a new class named CartViewModelBase.cs.
3) Add the following using statements:

```
using System.ComponentModel.DataAnnotations;
using Newtonsoft.Json;
using SpyStore_HOL.Models.ViewModels.Base;
```

4) Update the code to the following:

```
public class CartViewModelBase : ProductAndCategoryBase
{
  public int? CustomerId { get; set; }
  [DataType(DataType.Currency), Display(Name = "Total")]
  public decimal LineItemTotal { get; set; }
  public string TimeStampString =>
     TimeStamp != null ? JsonConvert.SerializeObject(TimeStamp).Replace("\"", "") : string.Empty;
}
```

Step 2: Create the AddToCartViewModel ViewModel

1) Add a new class named AddToCartViewModel.cs.
2) Add the following using statements:

```
using SpyStore_HOL.MVC.Validation;
using SpyStore_HOL.MVC.ViewModels.Base;
```

3) Update the code to the following:

```
public class AddToCartViewModel : CartViewModelBase
{
  [MustNotBeGreaterThan(nameof(UnitsInStock)), MustBeGreaterThanZero]
  public int Quantity { get; set; }
}
```

Step 3: Create the CartRecordViewModel ViewModel

1) Add a new class named CartRecordViewModel.cs.
2) Add the following using statements:

```
using SpyStore_HOL.MVC.Validation;
using SpyStore_HOL.MVC.ViewModels.Base;
```

3) Update the code to the following:

```
public class CartRecordViewModel : CartViewModelBase
{
  [MustNotBeGreaterThan(nameof(UnitsInStock))]
  public int Quantity { get; set; }
}
```

Step 4: Create the CartViewModel ViewModel

1) Add a new class named CartViewModel.cs.
2) Add the following using statements:

```
using System.Collections.Generic;
using SpyStore_HOL.Models.Entities;
```

3) Update the code to the following:

```
public class CartViewModel
{
  public Customer Customer { get; set; }
  public IList<CartRecordViewModel> CartRecords { get; set; }
}
```

Step 5: Update the _ViewImports.cshtml file

1) Open the _ViewImports.cshtml file in the Views folder
2) Add the following using statement:

```
@using SpyStore_HOL.MVC.ViewModels
```

Part 2: Create the Controllers

Step 1: Create the base controller

1) Create a new folder named Base under the Controllers folder
2) Add a class named BaseController to the Base folder
3) Add the following using statements:

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Filters;
```

4) Update the code to match the following:

```
public class BaseController : Controller
{
```

```
  public override void OnActionExecuting(ActionExecutingContext context)
  {
    ViewBag.CustomerId = 0;
  }
}
```

Step 2: Create the ProductsController controller

1) Add a class named ProductsController to the Controllers folder
2) Add the following using statements:

```
using Microsoft.AspNetCore.Mvc;
using SpyStore_HOL.DAL.Repos.Interfaces;
using SpyStore_HOL.MVC.Controllers.Base;
```

3) Update the code to match the following:

```
public class ProductsController : BaseController
{
  private readonly IProductRepo _productRepo;
  private readonly ICategoryRepo _categoryRepo;
  public ProductsController(IProductRepo productRepo, ICategoryRepo categoryRepo)
  {
    _productRepo = productRepo;
    _categoryRepo = categoryRepo;
  }
  [HttpGet]
  public ActionResult Error()
  {
    return View();
  }
  [HttpGet]
  public ActionResult Index()
  {
    return RedirectToAction(nameof(Featured));
  }
  public ActionResult Details(int id)
  {
    return RedirectToAction(nameof(CartController.AddToCart), nameof(CartController).Replace("Controller", ""), new {
customerId = ViewBag.CustomerId, productId = id, cameFromProducts = true });
  }
  [HttpGet]
  public IActionResult Featured()
  {
    ViewBag.Title = "Featured Products";
    ViewBag.Header = "Featured Products";
    ViewBag.ShowCategory = true;
    ViewBag.Featured = true;
    return View("ProductList", _productRepo.GetFeaturedWithCategoryName());
  }
  [HttpGet]
  public IActionResult ProductList(int id)
  {
```

```
    var cat = _categoryRepo.Find(id);
    ViewBag.Title = cat?.CategoryName;
    ViewBag.Header = cat?.CategoryName;
    ViewBag.ShowCategory = false;
    ViewBag.Featured = false;
    return View(_productRepo.GetProductsForCategory(id));
  }
  [Route("[controller]/[action]")]
  [HttpPost("{searchString}")]
  public IActionResult Search(string searchString)
  {
    ViewBag.Title = "Search Results";
    ViewBag.Header = "Search Results";
    ViewBag.ShowCategory = true;
    ViewBag.Featured = false;
    return View("ProductList", _productRepo.Search(searchString));
  }
}
```

Step 3: Create the OrdersController controller

1) Add a class named OrdersController to the Base folder
2) Add the following using statements:

```
using System.Collections.Generic;
using System.Linq;
using Microsoft.AspNetCore.Mvc;
using SpyStore_HOL.DAL.Repos.Interfaces;
using SpyStore_HOL.Models.Entities;
using SpyStore_HOL.Models.ViewModels;
using SpyStore_HOL.MVC.Controllers.Base;
```

3) Update the code to match the following:

```
[Route("[controller]/[action]/{customerId}")]
public class OrdersController : BaseController
{
  private readonly IOrderRepo _orderRepo;
  public OrdersController(IOrderRepo orderRepo)
  {
    _orderRepo = orderRepo;
  }
  [HttpGet]
  public IActionResult Index(int customerId)
  {
    ViewBag.Title = "Order History";
    ViewBag.Header = "Order History";
    IList<Order> orders = _orderRepo.GetOrderHistory(customerId).ToList();
    if (orders == null) return NotFound();
    return View(orders);
  }
  [HttpGet("{orderId}")]
  public IActionResult Details(int customerId, int orderId)
```

```
  {
    ViewBag.Title = "Order Details";
    ViewBag.Header = "Order Details";
    OrderWithDetailsAndProductInfo orderDetails = _orderRepo.GetOneWithDetails(customerId, orderId);
    if (orderDetails == null) return NotFound();
    return View(orderDetails);
  }
}
```

Step 4: Create the CartController controller

1) Add a class named CartController to the Base folder
2) Add the following using statements:

```
using System;
using System.Collections.Generic;
using AutoMapper;
using Microsoft.AspNetCore.Mvc;
using Newtonsoft.Json;
using SpyStore_HOL.DAL.Repos.Interfaces;
using SpyStore_HOL.Models.Entities;
using SpyStore_HOL.Models.ViewModels;
using SpyStore_HOL.Models.ViewModels.Base;
using SpyStore_HOL.MVC.Controllers.Base;
using SpyStore_HOL.MVC.ViewModels;
```

3) Update the code to match the following:

```
[Route("[controller]/[action]/{customerId}")]
public class CartController : BaseController
{
  private readonly IShoppingCartRepo _shoppingCartRepo;
  private readonly ICustomerRepo _customerRepo;
  private readonly IProductRepo _productRepo;
  readonly MapperConfiguration _config = null;
  public CartController(
      IShoppingCartRepo shoppingCartRepo,
      ICustomerRepo customerRepo,
      IProductRepo productRepo)
  {
    _shoppingCartRepo = shoppingCartRepo;
    _customerRepo = customerRepo;
    _productRepo = productRepo;
    _config = new MapperConfiguration(
        cfg =>
        {
          cfg.CreateMap<AddToCartViewModel, ShoppingCartRecord>()
              .AfterMap((s, t) =>
                  {
                    t.Id = 0;
                    t.TimeStamp = null;
                  });
```

```
        cfg.CreateMap<CartRecordViewModel, ShoppingCartRecord>();
        cfg.CreateMap<CartRecordWithProductInfo, CartRecordViewModel>();
        cfg.CreateMap<ProductAndCategoryBase, AddToCartViewModel>();
      });
}
[HttpGet]
public IActionResult Index(int customerId)
{
  ViewBag.Title = "Cart";
  ViewBag.Header = "Cart";
  var cartItems = _shoppingCartRepo.GetShoppingCartRecords(customerId);
  var customer = _customerRepo.Find(customerId);
  var mapper = _config.CreateMapper();
  var viewModel = new CartViewModel
  {
    Customer = customer,
    CartRecords = mapper.Map<IList<CartRecordViewModel>>(cartItems)
  };
  return View(viewModel);
}
[HttpGet("{productId}")]
public IActionResult AddToCart(int customerId, int productId, bool cameFromProducts = false)
{
  ViewBag.CameFromProducts = cameFromProducts;
  ViewBag.Title = "Add to Cart";
  ViewBag.Header = "Add to Cart";
  ViewBag.ShowCategory = true;
  var prod = _productRepo.GetOneWithCategoryName(productId);
  if (prod == null) return NotFound();
  var mapper = _config.CreateMapper();
  var cartRecord = mapper.Map<AddToCartViewModel>(prod);
  cartRecord.Quantity = 1;
  return View(cartRecord);
}
[ActionName("AddToCart"), HttpPost("{productId}"), ValidateAntiForgeryToken]
public IActionResult AddToCartPost(
    int customerId, int productId, AddToCartViewModel item)
{
  if (!ModelState.IsValid) return View(item);
  try
  {
    var mapper = _config.CreateMapper();
    var cartRecord = mapper.Map<ShoppingCartRecord>(item);
    cartRecord.DateCreated = DateTime.Now;
    cartRecord.CustomerId = item.CustomerId ?? 0;
    _shoppingCartRepo.Add(cartRecord);
  }
  catch (Exception ex)
  {
    ModelState.AddModelError(string.Empty, "There was an error adding the item to the cart.");
    return View(item);
  }
```

```
    return RedirectToAction(nameof(CartController.Index), new { customerId });
  }
  [HttpPost("{id}"), ValidateAntiForgeryToken]
  public IActionResult Update(int customerId, int id,
      string timeStampString, CartRecordViewModel item)
  {
    item.TimeStamp = JsonConvert.DeserializeObject<byte[]>($"\"{timeStampString}\"");
    if (!ModelState.IsValid) return PartialView(item);
    var mapper = _config.CreateMapper();
    var newItem = mapper.Map<ShoppingCartRecord>(item);
    try
    {
      newItem.DateCreated = DateTime.Now;
      _shoppingCartRepo.Update(newItem);
      var updatedItem = _shoppingCartRepo.GetShoppingCartRecord(customerId, item.ProductId);
      item = mapper.Map<CartRecordViewModel>(updatedItem);
      return PartialView(item);
    }
    catch (Exception ex)
    {
      ModelState.AddModelError(string.Empty, "An error occurred updating the cart.  Please reload the page and try
again.");
      return PartialView(item);
    }
  }
  [HttpPost("{id}"), ValidateAntiForgeryToken]
  public IActionResult Delete(int customerId, int id,
      ShoppingCartRecord item)
  {
    _shoppingCartRepo.Delete(id, item.TimeStamp);
    return RedirectToAction(nameof(Index), new { customerId });
  }
}
```

Step 5: Delete the Home controller

1) Delete the HomeController, as it's not used in this application.

Summary

The lab created the ViewModels and the Controllers

Next steps

In the next part of this tutorial series, you will create the Views for the application.