Code for Daryl’s Movie Emporium

Contents

[Notes about code 2](#_Toc53180039)

[3rd Party 2](#_Toc53180040)

[QR Code 2](#_Toc53180041)

[YouTube Modal 2](#_Toc53180042)

[SessionHelper 2](#_Toc53180043)

[Other 2](#_Toc53180044)

[Images 3](#_Toc53180045)

[Trailers 3](#_Toc53180046)

[Controllers (Controller code is a mixture of template code and my own) 3](#_Toc53180047)

[MoviesController 3](#_Toc53180048)

[HomeController 12](#_Toc53180049)

[Cart Controller 14](#_Toc53180050)

[Models (Model code is my own except for portions of the Movie class that came from a Microsoft tutorial) 22](#_Toc53180051)

[MvcMovie.Data (Database for Movies) 22](#_Toc53180052)

[Movie 23](#_Toc53180053)

[PurchaseModel 25](#_Toc53180054)

[Item 32](#_Toc53180055)

[SeedData 32](#_Toc53180056)

[Views (Views code is a mixture of template code and my own web page layouts) 35](#_Toc53180057)

[Home 35](#_Toc53180058)

[Index 35](#_Toc53180059)

[Privacy 36](#_Toc53180060)

[Movies 36](#_Toc53180061)

[Index 36](#_Toc53180062)

[MoviesByGenre 38](#_Toc53180063)

[AdminIndex 41](#_Toc53180064)

[Edit 44](#_Toc53180065)

[Details 46](#_Toc53180066)

[Delete 48](#_Toc53180067)

[Create 50](#_Toc53180068)

[Cart 52](#_Toc53180069)

[Index 52](#_Toc53180070)

[FinalizePurchase 55](#_Toc53180071)

[Invoice 63](#_Toc53180072)

[Shared 68](#_Toc53180073)

[EmptyCart (This is the only custom shared view) 68](#_Toc53180074)

[Layout (included in MVC template, but this is modified) 69](#_Toc53180075)

[CSS (Much of the CSS is from the template, but much is customized or added onto) 70](#_Toc53180076)

[JavaScript 76](#_Toc53180077)

[Helpers 77](#_Toc53180078)

[HelperClass 77](#_Toc53180079)

# Notes about code

## 3rd Party

### QR Code

The QR Code generator is a JavaScript library add on. It is from <https://github.com/jeromeetienne/jquery-qrcode> and released under the MIT License. This is used for 2 Factor Authentication

### YouTube Modal

The YouTube modal is a jQuery library. It is from <https://www.jqueryscript.net/lightbox/Youtube-Video-In-Modal-jQuery-GRT-Youtube-Popup.html> and released under the MIT License. This is used for modal popups for trailers.

### SessionHelper

The SessionHelper class is from an online lesson at <http://learningprogramming.net/net/asp-net-core-mvc/build-shopping-cart-with-session-in-asp-net-core-mvc/> and is licensed under their site policy at <http://learningprogramming.net/terms-of-use/>. This is used for making it easier to get complex data into and out of a session variable for tracking via cookies.

### Other

Many educational websites provided information about ASP.Net works, HTML, JavaScript, CSS, etc. and were used as references to learn various techniques. This application would not be possible without them. This document does not cover the numerous C# source files and CSHTML page generation files that were scaffolded in when adding the Identity service. While some very minor modifications were made (pragmas for disabling a warning and adding a value to display in the \_layout), the rest is essentially stock as the scaffold added it or modified only to complete the partially created code.

### Images

The various images in use are either downloaded to the site locally or live linked from other sites. In no way do I claim ownership of these images.

### Trailers

All trailers are linked from YouTube and are the property of their respective copyright holders.

# Controllers (Controller code is a mixture of template code and my own)

## MoviesController

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using MvcMovie.Data;

using MvcMovie.Models;

using System.Linq;

using System.Threading.Tasks;

namespace MvcMovie.Controllers

{

/// <summary>

/// The Movies controller handles the pages for browsing the site's web store.

/// </summary>

[Authorize]

public class MoviesController : Controller

{

private readonly MvcMovieContext \_context;

/// <summary>

/// The constructor takes in a database context to provide data to the website.

/// </summary>

/// <param name="context"></param>

public MoviesController(MvcMovieContext context)

{

\_context = context;

}

/// <summary>

/// The Index method returns the genre page, which is the start of the web store experience.

/// </summary>

/// <returns></returns>

// GET: Movies

public async Task<IActionResult> Index()

{

var movies = from m in \_context.Movie select m;

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

return View(await movies.ToListAsync());

}

/// <summary>

/// The AdminIndex method provides the admin user with access to pages for CRUD operations.

/// It takes in search criteria to narrow the view based on genre or title.

/// </summary>

/// <param name="criteria"></param>

/// <param name="searchString"></param>

/// <returns></returns>

// GET: Movies

public async Task<IActionResult> AdminIndex(string criteria, string searchString)

{

var movies = from m in \_context.Movie

select m;

if (criteria == "Title")

{

if (!string.IsNullOrEmpty(searchString))

{

movies = movies.Where(s => s.Title.Contains(searchString));

}

}

if (criteria == "Genre")

{

if (!string.IsNullOrEmpty(searchString))

{

movies = movies.Where(s => s.Genre.Contains(searchString));

}

}

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

return View(await movies.ToListAsync());

}

/// <summary>

/// The MoviesByGenre method returns the movie view filtered by the chosen genre.

/// It takes in an id which is the genre.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

// GET: Movies by Genre for Users

public async Task<IActionResult> MoviesByGenre(string id)

{

var movies = from m in \_context.Movie select m;

ViewBag.genre = id;

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

if (!string.IsNullOrEmpty(id))

{

movies = movies.Where(s => s.Genre.Contains(id));

return View(await movies.ToListAsync());

}

else

{

return View("Index");

}

}

/// <summary>

/// The Details method returns a detail view for CRUD operations.

/// It takes in an id parameter, which is the title of the movie.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

// GET: Movies/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var movie = await \_context.Movie

.FirstOrDefaultAsync(m => m.ID == id);

if (movie == null)

{

return NotFound();

}

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

return View(movie);

}

/// <summary>

/// The Create method returns a view for creating a new movie entry in the database CRUD operation.

/// </summary>

/// <returns></returns>

// GET: Movies/Create

public IActionResult Create()

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

return View();

}

/// <summary>

/// This entry for the Create method takes in the pertinent fields for the database to create the movie entry.

/// </summary>

/// <param name="movie"></param>

/// <returns></returns>

// POST: Movies/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("ID,Title,ReleaseDate,Genre,Price,ImageURL,TrailerURL,Description")] Movie movie)

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

if (ModelState.IsValid)

{

\_context.Add(movie);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(AdminIndex));

}

return View(movie);

}

/// <summary>

/// The Edit method returns a view for editing a movie entry via CRUD operation.

/// It takes in an id parameter, which is the movie title.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

// GET: Movies/Edit/5

public async Task<IActionResult> Edit(int? id)

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

if (id == null)

{

return NotFound();

}

var movie = await \_context.Movie.FindAsync(id);

if (movie == null)

{

return NotFound();

}

return View(movie);

}

/// <summary>

/// This entry for Edit takes in the movie id and binds to the model fields for the movie. It then updates the database context to save the edit.

/// </summary>

/// <param name="id"></param>

/// <param name="movie"></param>

/// <returns></returns>

// POST: Movies/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("ID,Title,ReleaseDate,Genre,Price,ImageURL,TrailerURL,Description")] Movie movie)

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

if (id != movie.ID)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(movie);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!MovieExists(movie.ID))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(AdminIndex));

}

return View(movie);

}

/// <summary>

/// The Delete method deletes an entry from the movie database.

/// It takes in an id, which is the movie title.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

// GET: Movies/Delete/5

public async Task<IActionResult> Delete(int? id)

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

if (id == null)

{

return NotFound();

}

var movie = await \_context.Movie

.FirstOrDefaultAsync(m => m.ID == id);

if (movie == null)

{

return NotFound();

}

return View(movie);

}

/// <summary>

/// The DeleteConfirmed method confirms the admin user wishes to delete a movie.

/// It takes in an id parameter, which is the movie id to be deleted.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

// POST: Movies/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

var movie = await \_context.Movie.FindAsync(id);

\_context.Movie.Remove(movie);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(AdminIndex));

}

/// <summary>

/// The MovieExists method confirms a movie exists in the database.

/// It takes in an id parameter, which is the private key index for the movie.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

private bool MovieExists(int id)

{

return \_context.Movie.Any(e => e.ID == id);

}

}

}

## HomeController

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using MvcMovie.Models;

using System.Diagnostics;

namespace MvcMovie.Controllers

{

/// <summary>

/// HomeController is the controller for the main home page of the site.

/// </summary>

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

/// <summary>

/// The constructor takes in a ILogger

/// </summary>

/// <param name="logger"></param>

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

/// <summary>

/// The Index method returns the view for the Home page.

/// </summary>

/// <returns></returns>

public IActionResult Index()

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0; // Total number of items in cart

return View();

}

/// <summary>

/// The Privacy method returns the view for the Privacy page.

/// </summary>

/// <returns></returns>

public IActionResult Privacy()

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

return View();

}

/// <summary>

/// The Error method returns an error page in the event that the controller cannot process an action.

/// </summary>

/// <returns></returns>

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems") != null ? HttpContext.Session.GetInt32("totalItems") : 0;

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

## Cart Controller

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using MvcMovie.Data;

using MvcMovie.Helpers;

using MvcMovie.Models;

using System;

using System.Collections.Generic;

using System.Linq;

namespace MvcMovie.Controllers

{

/// <summary>

/// The CartController serves the views for the shopping cart.

/// </summary>

[Authorize]

public class CartController : Controller

{

private readonly MvcMovieContext \_context;

/// <summary>

/// The constructor provides the shopping cart with access to the movie database.

/// </summary>

/// <param name="context"></param>

public CartController(MvcMovieContext context)

{

\_context = context;

}

/// <summary>

/// The Index method returns the shopping cart view.

/// </summary>

/// <returns></returns>

public IActionResult Index()

{

if (!HttpContext.Session.TryGetValue("cart", out byte[] value))

{

ViewBag.totalItems = 0;

return View("EmptyCart");

}

else

{

var cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

ViewBag.cart = cart.OrderBy(x => x.Movie.Title);

ViewBag.subtotal = cart.Sum(item => item.Movie.Price \* item.Quantity);

ViewBag.tax = Math.Round(ViewBag.subtotal \* (decimal)0.08, 2);

ViewBag.discount = ViewBag.subtotal / (decimal)50 >= (decimal)1.0 ? (decimal)5.00 \* Math.Truncate(ViewBag.subtotal / (decimal)50) : (decimal)0.0;

ViewBag.shipping = ViewBag.subtotal > (decimal)50.0 ? (decimal)0.00 : (decimal)4.99;

ViewBag.total = Math.Round(ViewBag.subtotal + ViewBag.tax + ViewBag.shipping - ViewBag.discount, 2);

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return View();

}

}

/// <summary>

/// The FinalizePurchase method returns the FinalizePurchase page for entering the billing address, shipping address and purchaser credit card inforamtion.

/// </summary>

/// <returns></returns>

public IActionResult FinalizePurchase()

{

var cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

ViewBag.cart = cart;

ViewBag.subtotal = cart.Sum(item => item.Movie.Price \* item.Quantity);

ViewBag.tax = Math.Round(ViewBag.subtotal \* (decimal)0.08, 2);

ViewBag.discount = ViewBag.subtotal / (decimal)50 >= (decimal)1.0 ? (decimal)5.00 \* Math.Truncate(ViewBag.subtotal / (decimal)50) : (decimal)0.0;

ViewBag.shipping = ViewBag.subtotal > (decimal)50.0 ? (decimal)0.00 : (decimal)4.99;

ViewBag.total = Math.Round(ViewBag.subtotal + ViewBag.tax + ViewBag.shipping - ViewBag.discount, 2);

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return View();

}

/// <summary>

/// The Invoice method returns the Invoice view.

/// This is a final summary of the purchase.

/// </summary>

/// <param name="model"></param>

/// <returns></returns>

public ActionResult Invoice(PurchaseModel model)

{

var cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

if (model.ShipName == null || model.ShipAdd1 == null || model.ShipState.ToString() == string.Empty || model.ShipZip == null)

{

ViewBag.ShipName = model.BillName;

ViewBag.ShipAdd1 = model.BillAdd1;

ViewBag.ShipAdd2 = model.BillAdd2;

ViewBag.ShipState = model.BillState;

ViewBag.ShipZip = model.BillZip;

}

else

{

ViewBag.ShipName = model.ShipName;

ViewBag.ShipAdd1 = model.ShipAdd1;

ViewBag.ShipAdd2 = model.ShipAdd2;

ViewBag.ShipState = model.ShipState;

ViewBag.ShipZip = model.ShipZip;

}

ViewBag.BillName = model.BillName;

ViewBag.BillAdd1 = model.BillAdd1;

ViewBag.BillAdd2 = model.BillAdd2;

ViewBag.BillState = model.BillState;

ViewBag.BillZip = model.BillZip;

ViewBag.cart = cart.OrderBy(x => x.Movie.Title);

ViewBag.subtotal = cart.Sum(item => item.Movie.Price \* item.Quantity);

ViewBag.tax = Math.Round(ViewBag.subtotal \* (decimal)0.08, 2);

ViewBag.discount = ViewBag.subtotal / (decimal)50 >= (decimal)1.0 ? (decimal)5.00 \* Math.Truncate(ViewBag.subtotal / (decimal)50) : (decimal)0.0;

ViewBag.shipping = ViewBag.subtotal > (decimal)50.0 ? (decimal)0.00 : (decimal)4.99;

ViewBag.total = Math.Round(ViewBag.subtotal + ViewBag.tax + ViewBag.shipping - ViewBag.discount, 2);

HttpContext.Session.Clear();

HttpContext.Session.SetInt32("totalItems", 0);

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return View();

}

/// <summary>

/// The Buy method adds a movie to the shopping cart.

/// It takes in an id parameter, which is the movie title, and a goBack parameter, which is the genre page the user was browsing.

/// </summary>

/// <param name="id"></param>

/// <param name="goBack"></param>

/// <returns></returns>

public IActionResult Buy(string id, string goBack)

{

var Movies = from m in \_context.Movie select m;

Movie Movie = new Movie();

if (SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart") == null)

{

var cart = new List<Item>();

cart.Add(new Item { Movie = Movies.Where(x => x.Title == id).FirstOrDefault(), Quantity = 1 });

SessionHelper.SetObjectAsJson(HttpContext.Session, "cart", cart);

HttpContext.Session.SetInt32("totalItems", cart.Select(x => x.Quantity).ToList().Sum());

}

else

{

List<Item> cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

int index = DoesExist(id);

if (index != -1)

{

cart[index].Quantity++;

}

else

{

cart.Add(new Item { Movie = Movies.Where(x => x.Title == id).FirstOrDefault(), Quantity = 1 });

}

HttpContext.Session.SetInt32("totalItems", cart.Select(x => x.Quantity).ToList().Sum());

SessionHelper.SetObjectAsJson(HttpContext.Session, "cart", cart);

}

// Redirecting back to page, so add to cart doesn't automatically go to the shopping cart. Need to hit the cart button in the menu bar to access the cart.

//return RedirectToAction("Index");

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return RedirectToAction("MoviesByGenre", "Movies", new { @id = goBack });

}

/// <summary>

/// The Remove method removes all instances of a movie from the shopping cart.

/// It takes in an id parameter, which is the movie title.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public IActionResult Remove(string id)

{

List<Item> cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

int index = DoesExist(id);

cart.RemoveAt(index);

SessionHelper.SetObjectAsJson(HttpContext.Session, "cart", cart);

HttpContext.Session.SetInt32("totalItems", cart.Select(x => x.Quantity).ToList().Sum());

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return RedirectToAction("Index");

}

/// <summary>

/// The Increment method adds one additional copy of a movie to the shopping cart.

/// It takes in an id parameter, which is the movie title.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public IActionResult Increment(string id)

{

List<Item> cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

int index = DoesExist(id);

cart[index].Quantity++;

SessionHelper.SetObjectAsJson(HttpContext.Session, "cart", cart);

HttpContext.Session.SetInt32("totalItems", cart.Select(x => x.Quantity).ToList().Sum());

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return RedirectToAction("Index");

}

/// <summary>

/// The Decrement method removes one copy of a movie from the shopping cart.

/// It takes in an id parameter, which is the movie title.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public IActionResult Decrement(string id)

{

List<Item> cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

int index = DoesExist(id);

cart[index].Quantity--;

if (cart[index].Quantity < 1)

{

cart.RemoveAt(index);

}

SessionHelper.SetObjectAsJson(HttpContext.Session, "cart", cart);

HttpContext.Session.SetInt32("totalItems", cart.Select(x => x.Quantity).ToList().Sum());

ViewBag.totalItems = HttpContext.Session.GetInt32("totalItems");

return RedirectToAction("Index");

}

/// <summary>

/// The DoesExist method verifies a movie is in the database.

/// It takes in an id parameter, which is the movie title.

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

private int DoesExist(string id)

{

List<Item> cart = SessionHelper.GetObjectFromJson<List<Item>>(HttpContext.Session, "cart");

for (int i = 0; i < cart.Count; i++)

{

if (cart[i].Movie.Title.Equals(id))

{

return i;

}

}

return -1;

}

}

}

# Models (Model code is my own except for portions of the Movie class that came from a Microsoft tutorial)

## MvcMovie.Data (Database for Movies)

using Microsoft.EntityFrameworkCore;

using MvcMovie.Models;

namespace MvcMovie.Data

{

/// <summary>

/// The MvcMovieContext class defines the database context for the Movies model.

/// </summary>

public class MvcMovieContext : DbContext

{

/// <summary>

/// The constructor takes in DbContextOptions.

/// </summary>

/// <param name="options"></param>

public MvcMovieContext(DbContextOptions<MvcMovieContext> options)

: base(options)

{

}

/// <summary>

/// The Movie field is the DbSet for the context.

/// </summary>

public DbSet<Movie> Movie { get; set; }

}

}

## Movie

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace MvcMovie.Models

{

/// <summary>

/// The Movie class contains the model definition for a movie object.

/// </summary>

public class Movie

{

/// <summary>

/// The ID field is the private key id.

/// </summary>

public int ID { get; set; }

/// <summary>

/// The Title field is the movie title.

/// </summary>

public string Title { get; set; }

/// <summary>

/// The ReleaseDate field is the release date for the movie.

/// </summary>

[Display(Name = "Release Date")]

[DataType(DataType.Date)]

public DateTime ReleaseDate { get; set; }

/// <summary>

/// The Genre field is the movie's genre.

/// In some cases multiple genres are assigned separated by commas.

/// </summary>

public string Genre { get; set; }

/// <summary>

/// The Price field is the movie's selling price.

/// </summary>

[Column(TypeName = "decimal(18,2)")]

public decimal Price { get; set; }

/// <summary>

/// The ImageURL field stores a link to a movie poster.

/// </summary>

[Display(Name = "Trailer")]

public string ImageURL { get; set; }

/// <summary>

/// The TrailerURL field stores the YouTube ID for a movie's trailer.

/// </summary>

public string TrailerURL { get; set; }

/// <summary>

/// The Description field stores the movie's plot description.

/// </summary>

public string Description { get; set; }

}

}

## PurchaseModel

#pragma warning disable 1591

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

namespace MvcMovie.Models

{

/// <summary>

/// PurchaseModel class

/// Models information for purchase. Includes billing and shipping information as well as credit card purchase information.

/// </summary>

public class PurchaseModel

{

/// <summary>

/// ShipName is the name of the shipee.

/// </summary>

[DisplayName("Name")]

public string ShipName { get; set; }

/// <summary>

/// ShipAdd1 is line 1 of the shipee's address.

/// This is typically a street address or PO Box.

/// </summary>

[DisplayName("Address 1")]

public string ShipAdd1 { get; set; }

/// <summary>

/// ShipAdd2 is line 2 of the shipee's address.

/// This usually is an apartment number of similar.

/// </summary>

[DisplayName("Address 2")]

public string ShipAdd2 { get; set; }

/// <summary>

/// ShipState is the state for the shipee's address.

/// It is of type StateList (enum of US States and districts)

/// </summary>

[DisplayName("State")]

public StateList ShipState { get; set; }

/// <summary>

/// ShipZip is the shipee's zip code.

/// </summary>

[DisplayName("Zip Code")]

public string ShipZip { get; set; }

/// <summary>

/// BillName is the name of the purchaser.

/// </summary>

[DisplayName("Name")]

public string BillName { get; set; }

/// <summary>

/// BillAdd1 is line 1 of the purchaser's address.

/// It is typically a street address or PO Box and is tied to the credit card.

/// </summary>

[DisplayName("Address 1")]

public string BillAdd1 { get; set; }

/// <summary>

/// BillAdd2 is line 2 of the purchaser's address.

/// It is typically an apartment number or similar.

/// </summary>

[DisplayName("Address 2")]

public string BillAdd2 { get; set; }

/// <summary>

/// BillState is the state of the purchaser's address.

/// It is of type StateList (enum of US States and districts).

/// </summary>

[DisplayName("State")]

public StateList BillState { get; set; }

/// <summary>

/// BillZip is the zip code of the purchaser's address.

/// </summary>

[DisplayName("Zip Code")]

public string BillZip { get; set; }

/// <summary>

/// Card is the purchaser's credit or debit card number.

/// </summary>

[DisplayName("Credit Card")]

public int Card { get; set; }

/// <summary>

/// Cvv is the CVV security code of the purchaser's credit or debit card.

/// </summary>

[DisplayName("CVV")]

public int Cvv { get; set; }

/// <summary>

/// Month is the expiration month of the purchaser's credit or debit card.

/// It is of type MonthList (enum of months of the year).

/// </summary>

[DisplayName("Month")]

public MonthList Month { get; set; }

/// <summary>

/// Year is the expiration year of the purchaser's credit or debit card.

/// It is of type YearList (enum of years spanning from 2020 to 2027).

/// </summary>

[DisplayName("Year")]

public YearList Year { get; set; }

}

/// <summary>

/// StateList is an enum of the US states and districts.

/// It is used for drop down lists.

/// </summary>

public enum StateList

{

Alabama,

Alaska,

Arizona,

Arkansas,

California,

Colorado,

Connecticut,

Delaware,

[Display(Name = "District Of Columbia")]

District\_of\_Columbia,

Florida,

Georgia,

Hawaii,

Idaho,

Illinois,

Indiana,

Iowa,

Kansas,

Kentucky,

Louisiana,

Maine,

Maryland,

Massachusetts,

Michigan,

Minnesota,

Mississippi,

Missouri,

Montana,

Nebraska,

Nevada,

[Display(Name = "New Hampshire")]

NewHampshire,

[Display(Name = "New Jersey")]

NewJersey,

[Display(Name = "New Mexico")]

NewMexico,

[Display(Name = "New York")]

NewYork,

[Display(Name = "North Carolina")]

NorthCarolina,

[Display(Name = "North Dakota")]

NorthDakota,

Ohio,

Oklahoma,

Oregon,

Pennsylvania,

[Display(Name = "Rhode Island")]

RhodeIsland,

[Display(Name = " South Carolina")]

SouthCarolina,

[Display(Name = "South Dakota")]

SouthDakota,

Tennessee,

Texas,

Utah,

Vermont,

Virginia,

Washington,

[Display(Name = "West Virginia")]

WestVirginia,

Wisconsin,

Wyoming

}

/// <summary>

/// MonthList is an enum of the months in the year.

/// It is used for drop down lists.

/// </summary>

public enum MonthList

{

Jan,

Feb,

Mar,

Apr,

May,

Jun,

Jul,

Aug,

Sep,

Oct,

Nov,

Dec

}

/// <summary>

/// YearList is an enum of the years 2020 to 2027.

/// It is used for drop down lists.

/// </summary>

public enum YearList

{

[Display(Name = "2020")]

one = 2020,

[Display(Name = "2021")]

two = 2021,

[Display(Name = "2022")]

three = 2022,

[Display(Name = "2023")]

four = 2023,

[Display(Name = "2024")]

five = 2024,

[Display(Name = "2025")]

six = 2025,

[Display(Name = "2026")]

seven = 2026,

[Display(Name = "2027")]

eight = 2027

}

}

## Item

namespace MvcMovie.Models

{

/// <summary>

/// The Item class is the model for the shopping cart.

/// </summary>

public class Item

{

/// <summary>

/// The Movie field is a movie object.

/// </summary>

public Movie Movie { get; set; }

/// <summary>

/// The Quantity field is the number of copies of a movie in the cart.

/// </summary>

public int Quantity { get; set; }

}

}

## SeedData

(This came directly from the tutorial, and I did use the class, however, I added new fields to the database and did not update this class to generate that data.).

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.DependencyInjection;

using MvcMovie.Data;

using System;

using System.Linq;

namespace MvcMovie.Models

{

/// <summary>

/// The SeedData class seeds an empty movie database.

/// </summary>

public static class SeedData

{

/// <summary>

/// The Initialize method adds some basic information to the database if it is empty.

/// </summary>

/// <param name="serviceProvider"></param>

public static void Initialize(IServiceProvider serviceProvider)

{

using (var context = new MvcMovieContext(

serviceProvider.GetRequiredService<

DbContextOptions<MvcMovieContext>>()))

{

// Look for any movies.

if (context.Movie.Any())

{

return; // DB has been seeded

}

context.Movie.AddRange(

new Movie

{

Title = "When Harry Met Sally",

ReleaseDate = DateTime.Parse("1989-2-12"),

Genre = "Romantic Comedy",

Price = 7.99M

},

new Movie

{

Title = "Ghostbusters ",

ReleaseDate = DateTime.Parse("1984-3-13"),

Genre = "Comedy",

Price = 8.99M

},

new Movie

{

Title = "Ghostbusters 2",

ReleaseDate = DateTime.Parse("1986-2-23"),

Genre = "Comedy",

Price = 9.99M

},

new Movie

{

Title = "Rio Bravo",

ReleaseDate = DateTime.Parse("1959-4-15"),

Genre = "Western",

Price = 3.99M

}

);

context.SaveChanges();

}

}

}

}

# Views (Views code is a mixture of template code and my own web page layouts)

## Home

### Index

@{

ViewData["Title"] = "Home Page";

}

<body class="home">

<div class="text-center">

<h1 class="display-4"><img src="~/img/coollogo\_com-12440471.png" /></h1>

<p>Your place to find all the hot movies in our rinky dink database!</p>

</div>

<div>

<center>

<table>

<tr>

<td>

<img src="~/img/fire.gif" />

</td>

<td style="vertical-align: bottom">

<img src="~/img/HotDeals.gif" />

</td>

<td>

<img src="~/img/fire.gif" />

</td>

</tr>

</table>

</center>

</div>

</body>

### Privacy

@{

ViewData["Title"] = "Privacy Policy";

}

<h1>@ViewData["Title"]</h1>

<body class="privacy">

<table>

<tr>

<td>

<p>We at Daryl's Movie Emporium take your privacy seriously. After 10 minutes of inactivity your session clears and the shopping cart empties.<br />We don't even keep a purchase history for you, so your choices are your own.</p>

</td>

</tr>

<tr>

<td style="text-align: center">

<img src="../img/online-privacy-clipart.png" width="400" height="400" style="vertical-align: middle" />

</td>

</tr>

</table>

</body>

## Movies

### Index

@model IEnumerable<MvcMovie.Models.Movie>

@{

ViewData["Title"] = "Genre";

}

<h1>Browse Popular Genres</h1>

<table class="table">

<tr>

<td>

<a asp-action="MoviesByGenre" asp-route-id="Comedy">

<img class="resize" alt="Comedy" src="https://www.filmsite.org/images/comedy-genre.jpg" />

</a>

</td>

<td style="text-align: center">

<a asp-action="MoviesByGenre" asp-route-id="Action">

<img class="resize" alt="Action" src="https://www.filmsite.org/images/action-genre.jpg" />

</a>

</td>

<td style="text-align: right">

<a asp-action="MoviesByGenre" asp-route-id="Sci Fi">

<img class="resize" alt="Sci Fi" src="https://www.filmsite.org/images/scifi-genre.jpg" />

</a>

</td>

</tr>

<tr>

<td>

<a asp-action="MoviesByGenre" asp-route-id="Romance">

<img class="resize" alt="Romance" src="https://www.filmsite.org/images/romance-genre.jpg" />

</a>

</td>

<td style="text-align: center">

<a asp-action="MoviesByGenre" asp-route-id="Horror">

<img class="resize" alt="Horror" src="https://www.filmsite.org/images/horror-genre.jpg" />

</a>

</td>

<td style="text-align: right">

<a asp-action="MoviesByGenre" asp-route-id="Fantasy">

<img class="resize" alt="Fantasy" src="https://www.filmsite.org/images/fantasy-genre.jpg" />

</a>

</td>

</tr>

<tr>

<td>

<a asp-action="MoviesByGenre" asp-route-id="Western">

<img class="resize" alt="Western" src="https://www.filmsite.org/images/westerns-genre.jpg" />

</a>

</td>

<td style="text-align: center">

<a asp-action="MoviesByGenre" asp-route-id="Animated">

<img class="resize" alt="Animation" src="https://www.filmsite.org/images/animated-genre.jpg" />

</a>

</td>

<td style="text-align: right">

<a asp-action="MoviesByGenre" asp-route-id="Adventure">

<img class="resize" alt="Adventure" src="https://www.filmsite.org/images/adventure-genre.jpg" />

</a>

</td>

</tr>

</table>

### MoviesByGenre

@model IEnumerable<MvcMovie.Models.Movie>

@using MvcMovie.Helpers;

@{

ViewData["Title"] = "Daryl's Movie Emporium";

}

<h1>@ViewBag.genre</h1>

<body>

<table class="table sales">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.ImageURL)

</th>

<th>

@Html.DisplayNameFor(model => model.Title)

</th>

<th>

@Html.DisplayNameFor(model => model.ReleaseDate)

</th>

<th>

@Html.DisplayNameFor(model => model.Price)

</th>

<th>

Description

</th>

<th>

Purchase

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.OrderBy(x => x.Title))

{

<tr class="mList" style="vertical-align:central">

<td style="vertical-align: middle">

<div class="player">

<a href="#" target="\_blank" class="youtube-link" youtubeid=@Html.DisplayFor(modelItem => item.TrailerURL)> <img class="thumbnail" src=@Html.DisplayFor(modelItem => item.ImageURL) alt="Trailer" /> </a>

</div>

</td>

<td style="vertical-align: middle">

@Html.DisplayFor(modelItem => item.Title)

</td>

<td style="vertical-align: middle">

@Html.DisplayFor(modelItem => item.ReleaseDate)

</td>

<td style="vertical-align: middle">

@Html.DisplayFor(modelItem => item.Price)

</td>

<td style="vertical-align: middle">

<button id=@HelperClass.Combine(Html.DisplayTextFor(modelItem => item.Title), string.Empty) title="Movie Description" class="btn-primary" onclick="displayModal(this.id)">Description</button>

<div id=@HelperClass.Combine(Html.DisplayTextFor(modelItem => item.Title), "\_modal") class="modal">

<div class="modal-content">

<span class="close">&times;</span>

<p>@Html.DisplayTextFor(modelItem => item.Description)</p>

</div>

</div>

</td>

<td style="vertical-align: middle">

<input type="button" class="btn-primary" title="Add to Cart" value="Add to Cart" onclick="location.href='@Url.Action("buy", "cart", new { id = item.Title, goBack = ViewBag.genre })'" />

</td>

<td>

</td>

</tr>

}

</tbody>

</table>

<!-- Jquery -->

<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>

<!-- Github button for demo page -->

<script async defer src="https://buttons.github.io/buttons.js"></script>

<!-- GRT Youtube Popup -->

<script src="~/js/grt-youtube-popup.js"></script>

<script>$(".youtube-link").grtyoutube({ autoPlay: true });</script>

</body>

### AdminIndex

@using Microsoft.AspNetCore.Identity

@inject SignInManager<IdentityUser> SignInManager

@inject UserManager<IdentityUser> UserManager

@model IEnumerable<MvcMovie.Models.Movie>

@if (SignInManager.IsSignedIn(User))

{

@if (UserManager.GetUserName(User) == "sonnierd3916@uhcl.edu")

{

{

ViewData["Title"] = "Movie Administration";

}

<h1>Movie Database Administration</h1>

<p>

<a asp-action="Create">Create New</a>

</p>

<form asp-controller="Movies" asp-action="Index">

<p>

<input type="radio" id="title" name="criteria" value="Title" checked="checked" />

<label for="criteria">Title</label>

<input type="radio" id="genre" name="criteria" value="Genre" />

<label for="criteria">Genre</label>

<input type="text" name="SearchString">

<input type="submit" value="Filter" />

</p>

</form>

<table class="table">

<thead>

<tr>

<th>

Thumbnail

</th>

<th>

@Html.DisplayNameFor(model => model.Title)

</th>

<th>

@Html.DisplayNameFor(model => model.ReleaseDate)

</th>

<th>

@Html.DisplayNameFor(model => model.Genre)

</th>

<th>

@Html.DisplayNameFor(model => model.Price)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model.OrderBy(x => x.Title))

{

<tr>

<td>

<a href=@Html.DisplayFor(modelItem => item.TrailerURL) target="\_blank"> <img src=@Html.DisplayFor(modelItem => item.ImageURL) alt="Trailer" width="35" height="55" /> </a>

</td>

<td>

@Html.DisplayFor(modelItem => item.Title)

</td>

<td>

@Html.DisplayFor(modelItem => item.ReleaseDate)

</td>

<td>

@Html.DisplayFor(modelItem => item.Genre)

</td>

<td>

@Html.DisplayFor(modelItem => item.Price)

</td>

<td>

<a asp-action="Edit" asp-route-id="@item.ID">Edit</a> |

<a asp-action="Details" asp-route-id="@item.ID">Details</a> |

<a asp-action="Delete" asp-route-id="@item.ID">Delete</a>

</td>

</tr>

}

</tbody>

</table>

}

else

{

<p>Access Denied!</p>

}

}

### Edit

@model MvcMovie.Models.Movie

@{

ViewData["Title"] = "Edit";

}

<h1>Edit</h1>

<h4>Movie</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Edit">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input type="hidden" asp-for="ID" />

<div class="form-group">

<label asp-for="Title" class="control-label"></label>

<input asp-for="Title" class="form-control" />

<span asp-validation-for="Title" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ReleaseDate" class="control-label"></label>

<input asp-for="ReleaseDate" class="form-control" />

<span asp-validation-for="ReleaseDate" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Genre" class="control-label"></label>

<input asp-for="Genre" class="form-control" />

<span asp-validation-for="Genre" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Price" class="control-label"></label>

<input asp-for="Price" class="form-control" />

<span asp-validation-for="Price" class="text-danger"></span>

</div>

<div class="form-group">

<label class="control-label">Thumbnail URL</label>

<input asp-for="ImageURL" class="form-control" />

<span asp-validation-for="ImageURL" class="text-danger"></span>

</div>

<div class="form-group">

<label class="control-label">YouTube ID</label>

<input asp-for="TrailerURL" class="form-control" />

<span asp-validation-for="TrailerURL" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Description" class="control-label"></label>

<input asp-for="Description" class="form-control" />

<span asp-validation-for="Description" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Save" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="AdminIndex">Back to List</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

### Details

@model MvcMovie.Models.Movie

@{

ViewData["Title"] = "Details";

}

<h1>Details</h1>

<div>

<h4>Movie</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Title)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Title)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ReleaseDate)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ReleaseDate)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Genre)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Genre)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ImageURL)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ImageURL)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Description)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Description)

</dd>

</dl>

</div>

<div>

<a asp-action="Edit" asp-route-id="@Model.ID">Edit</a> |

<a asp-action="AdminIndex">Back to List</a>

</div>

### Delete

@model MvcMovie.Models.Movie

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

<h3>Are you sure you want to delete this?</h3>

<div>

<h4>Movie</h4>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Title)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Title)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ReleaseDate)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ReleaseDate)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Genre)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Genre)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Price)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Price)

</dd>

</dl>

<form asp-action="Delete">

<input type="hidden" asp-for="ID" />

<input type="submit" value="Delete" class="btn btn-danger" /> |

<a asp-action="AdminIndex">Back to List</a>

</form>

</div>

### Create

@model MvcMovie.Models.Movie

@{

ViewData["Title"] = "Create";

}

<h1>Create</h1>

<h4>Movie</h4>

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Create">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="Title" class="control-label"></label>

<input asp-for="Title" class="form-control" />

<span asp-validation-for="Title" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="ReleaseDate" class="control-label"></label>

<input asp-for="ReleaseDate" class="form-control" />

<span asp-validation-for="ReleaseDate" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Genre" class="control-label"></label>

<input asp-for="Genre" class="form-control" />

<span asp-validation-for="Genre" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Price" class="control-label"></label>

<input asp-for="Price" class="form-control" />

<span asp-validation-for="Price" class="text-danger"></span>

</div>

<div class="form-group">

<label class="control-label">Thumnail URL</label>

<input asp-for="ImageURL" class="form-control" />

<span asp-validation-for="ImageURL" class="text-danger"></span>

</div>

<div class="form-group">

<label class="control-label">YouTube ID</label>

<input asp-for="TrailerURL" class="form-control" />

<span asp-validation-for="TrailerURL" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Description" class="control-label"></label>

<input asp-for="Description" class="form-control" />

<span asp-validation-for="Description" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Create" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="AdminIndex">Back to List</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

## Cart

### Index

@using System.Linq;

@{

ViewData["Title"] = "Index";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Shopping Cart</h1>

<table cellpadding="2" cellspacing="2" border="1" class="table mList sales">

<tr>

<th>Option</th>

<th>Title</th>

<th>Thumbnail</th>

<th>Price</th>

<th>Quantity</th>

<th>Sub Total</th>

<th>Add/Subtract</th>

</tr>

@foreach (var item in ViewBag.cart)

{

<tr>

<td style="vertical-align: middle"><a asp-controller="cart" asp-action="remove" asp-route-id="@item.Movie.Title" class="btn-danger btn-block center">Remove All</a></td>

<td style="vertical-align: middle">@item.Movie.Title</td>

<td style="margin: auto"><img src=@item.Movie.ImageURL width="60" /></td>

<td style="vertical-align: middle">@item.Movie.Price</td>

<td style="vertical-align: middle">@item.Quantity</td>

<td style="vertical-align: middle">@(item.Movie.Price \* item.Quantity)</td>

<td style="vertical-align: middle">

<table class="table-borderless">

<tr style="text-align: center">

<td></td>

<td></td>

<td>

<a href="@Url.Action("increment", "cart", new { id = item.Movie.Title })" title="Add One">

<img src="~/img/plus-orange.png" />

</a>

</td>

<td>

<a href="@Url.Action("decrement", "cart", new { id = item.Movie.Title })" title="Remove One">

<img src="~/img/minus-orange.png" />

</a>

</td>

</tr>

</table>

</td>

</tr>

}

<tr>

<td align="right" colspan="6">Subtotal</td>

<td>

@ViewBag.subtotal

</td>

</tr>

<tr>

<td align="right" colspan="6">Taxes</td>

<td>

@ViewBag.tax

</td>

</tr>

<tr colspan="7">

<td align="right" colspan="6">

Discount of $5.00 off your purchases for every $50.00 you spend!

</td>

<td>

@ViewBag.discount

</td>

</tr>

<tr>

<td align="right" colspan="6">Shipping</td>

<td>

@ViewBag.shipping

</td>

</tr>

<tr>

<td align="right" colspan="6">Total</td>

<td>

@ViewBag.total

</td>

</tr>

</table>

<br>

<table width="100%" class="table-borderless">

<tr>

<td>

<a asp-controller="movies" asp-action="MoviesByGenre" asp-route-id="@ViewBag.last" class="btn-info btn-lg">Continue Shopping</a>

</td>

<td>

<a asp-controller="cart" asp-action="FinalizePurchase" class="btn-success btn-lg">Purchase Now</a>

</td>

</tr>

</table>

### FinalizePurchase

@using MvcMovie.Models

@model PurchaseModel

@{ ViewData["Title"] = "FinalizePurchase";

Layout = "~/Views/Shared/\_Layout.cshtml"; }

<h1>Finalize Purchase</h1>

<table cellpadding="2" cellspacing="2" border="1" class="table mList">

<tr>

<th>Title</th>

<th>Price</th>

<th>Quantity</th>

<th>Sub Total</th>

</tr>

@foreach (var item in ViewBag.cart)

{

<tr>

<td>@item.Movie.Title</td>

<td>@item.Movie.Price</td>

<td>@item.Quantity</td>

<td>@(item.Movie.Price \* item.Quantity)</td>

</tr>}

<tr>

<td align="right" colspan="3">Subtotal</td>

<td>

@ViewBag.subtotal

</td>

</tr>

<tr>

<td align="right" colspan="3">Taxes</td>

<td>

@ViewBag.tax

</td>

</tr>

<tr colspan="4">

<td align="right" colspan="3">

Discount of $5.00 off your purchases for every $50.00 you spend!

</td>

<td>

@ViewBag.discount

</td>

</tr>

<tr>

<td align="right" colspan="3">Shipping</td>

<td>

@ViewBag.shipping

</td>

</tr>

<tr>

<td align="right" colspan="3">Total</td>

<td>

@ViewBag.total

</td>

</tr>

</table>

<div>

<p>Shipping Address same as Billing Address: Yes <input type="radio" onclick="javascript:yesnoCheck();" name="yesno" id="yesCheck" checked="checked"> No <input type="radio" onclick="javascript:yesnoCheck();" name="yesno" id="noCheck"></p>

</div>

<form asp-action="Invoice" asp-controller="Cart" onsubmit="return checkReqFields()" id="purchaseForm">

<table class="mList float-left" width="45%">

<tr colspan="2">

<td>

<h2>Billing Address</h2>

</td>

</tr>

<tr>

<td>

<label asp-for="BillName"></label>

</td>

<td>

@Html.TextBoxFor(m => m.BillName, new { id = "name", placeholder = "John Doe", @class = "form-control" })

</td>

</tr>

<tr>

<td>

<label asp-for="BillAdd1"></label>

</td>

<td>

@Html.TextBoxFor(m => m.BillAdd1, new { id = "add", placeholder = "123 Easy Street", @class = "form-control" })

</td>

</tr>

<tr>

<td>

<label asp-for="BillAdd2"></label>

</td>

<td>

@Html.TextBoxFor(m => m.BillAdd2, new { placeholder = "Apt 110", @class = "form-control" })

</td>

</tr>

<tr>

<td>

<label asp-for="BillState">State</label>

</td>

<td>

@Html.DropDownListFor(m => Model.BillState, Html.GetEnumSelectList<StateList>(), "Select State", new { id = "state", @class = "form-control" })

</td>

</tr>

<tr>

<td>

<label asp-for="BillZip"></label>

</td>

<td>

@Html.TextBoxFor(m => m.BillZip, new { id = "zip", placeholder = "00000", @class = "form-control" })

</td>

</tr>

</table>

<div id="ifYes" style="display:none">

<table class="mList float-right" width="45%">

<tr colspan="2">

<td>

<h2>Shipping Address</h2>

</td>

</tr>

<tr>

<td>

<label asp-for="ShipName"></label>

</td>

<td>

@Html.TextBoxFor(m => m.ShipName, new { placeholder = "John Doe", @class = "form-control" , @id = "ShipName"})

</td>

</tr>

<tr>

<td>

<label asp-for="ShipAdd1"></label>

</td>

<td>

@Html.TextBoxFor(m => m.ShipAdd1, new { placeholder = "123 Easy Street", @class = "form-control" })

</td>

</tr>

<tr>

<td>

<label asp-for="ShipAdd2"></label>

</td>

<td>

@Html.TextBoxFor(m => m.ShipAdd2, new { placeholder = "Apt 110", @class = "form-control" })

</td>

</tr>

<tr>

<td>

<fieldset>

<label asp-for="ShipState"></label>

</fieldset>

</td>

<td>

@Html.DropDownListFor(m => Model.ShipState, Html.GetEnumSelectList<StateList>(), "Select State", new { @class = "form-control" })

</td>

</tr>

<tr>

<td>

<label asp-for="ShipZip"></label>

</td>

<td>

@Html.TextBoxFor(m => m.ShipZip, new { placeholder = "00000", @class = "form-control" })

</td>

</tr>

</table>

</div>

<table class="mList" width="100%">

<tr colspan="5">

<td>

<h3>Credit Card</h3>

</td>

</tr>

<tr>

<td width="20%">

<label asp-for="Card"></label>

</td>

<td width="20%">

@Html.TextBoxFor(m => m.Card, new { id = "card", placeholder = "0000 0000 0000 0000", @class = "form-control", maxlength = "19", minlength = "14" })

</td>

<td width="20%" align="right">

<label asp-for="Cvv"></label>

</td>

<td width="20%">

@Html.TextBoxFor(m => m.Cvv, new { id = "cvv", placeholder = "000", @class = "form-control", minlength = "3", maxlength = "4" })

</td>

<td width="20%">

</td>

</tr>

<tr>

<td width="20%">

<label asp-for="Month"></label>

</td>

<td width="20%">

@Html.DropDownListFor(m => Model.Month, Html.GetEnumSelectList<MonthList>(), "Select Month", new { id = "month", @class = "form-control" })

</td>

<td width="20%" align="right">

<label asp-for="Year"></label>

</td>

<td width="20%">

@Html.DropDownListFor(m => Model.Year, Html.GetEnumSelectList<YearList>(), "Select Year", new { id = "year", @class = "form-control" })

</td>

<td width="20%">

</td>

</tr>

</table>

<input type="submit" value="Purchase" />

</form>

@section Scripts {

@{ await Html.RenderPartialAsync("\_ValidationScriptsPartial"); }

<script type="text/javascript">

$(document).ready(function () {

var checkformValidation = function () {

$("#purchaseForm").validate({

rules: {

name : "required",

add : "required",

state : "required",

zip : "required",

year : "required",

month : "required",

cvv : "required",

card : "required"

}

});

}

$('#submitApp').click(function (e) {

e.preventDefault();

checkformValidation();

if ($("#purchaseForm").valid()) {

alert("success");

return;

} else {

alert("wrong");

}

});

});

document.getElementById('card').addEventListener('input', function (e) {

e.target.value = e.target.value.replace(/[^\dA-Z]/g, '').replace(/(.{4})/g, '$1 ').trim();

});

</script>

}

<br>

### Invoice

@model MvcMovie.Models.PurchaseModel

@{

ViewData["Title"] = "Invoice";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Invoice - Daryl's Movie Emporium</h1>

<p>

Thank you, @ViewBag.BillName, for your business!<br />

Come back for all your movie purchasing needs!

</p>

<table cellpadding="2" cellspacing="2" border="1" class="table">

<tr>

<th>Title</th>

<th>Price</th>

<th>Quantity</th>

<th>Sub Total</th>

</tr>

@foreach (var item in ViewBag.cart)

{

<tr>

<td>@item.Movie.Title</td>

<td>@item.Movie.Price</td>

<td>@item.Quantity</td>

<td>@(item.Movie.Price \* item.Quantity)</td>

</tr>}

<tr>

<td align="right" colspan="3">Subtotal</td>

<td>

@ViewBag.subtotal

</td>

</tr>

<tr>

<td align="right" colspan="3">Taxes</td>

<td>

@ViewBag.tax

</td>

</tr>

<tr colspan="4">

<td align="right" colspan="3">

Discount of $5.00 off your purchases for every $50.00 you spend!

</td>

<td>

@ViewBag.discount

</td>

</tr>

<tr>

<td align="right" colspan="3">Shipping</td>

<td>

@ViewBag.shipping

</td>

</tr>

<tr>

<td align="right" colspan="3">Total</td>

<td>

@ViewBag.total

</td>

</tr>

</table>

<table width="100%">

<tr>

<td>

<h3>Billing Address</h3>

</td>

</tr>

<tr>

<td>

@ViewBag.BillName

</td>

</tr>

<tr>

<td>

@ViewBag.BillAdd1

</td>

</tr>

<tr>

<td>

@ViewBag.BillAdd2

</td>

</tr>

<tr>

<td>

@ViewBag.BillState

</td>

</tr>

<tr>

<td>

@ViewBag.BillZip

</td>

</tr>

<tr>

<td>

<h3>Shipping Address</h3>

</td>

</tr>

<tr>

<td>

@ViewBag.ShipName

</td>

</tr>

<tr>

<td>

@ViewBag.ShipAdd1

</td>

</tr>

<tr>

<td>

@ViewBag.ShipAdd2

</td>

</tr>

<tr>

<td>

@ViewBag.ShipState

</td>

</tr>

<tr>

<td>

@ViewBag.ShipZip

</td>

</tr>

<tr>

<td align="right">

<input type="button" onclick="window.print();return false;" value="Print" name="Print" class="btn-primary no-print" />

</td>

</tr>

</table>

## Shared

### EmptyCart (This is the only custom shared view)

@{

ViewData["Title"] = "EmptyCart";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Shopping Cart</h1>

<table class="table mList sales" cellpadding="2" cellspacing="2" border="1">

<tr>

<th>Option</th>

<th>Title</th>

<th>Thumbnail</th>

<th>Price</th>

<th>Quantity</th>

<th>Sub Total</th>

<th></th>

</tr>

<tr>

<td align="right" colspan="6">Subtotal</td>

<td>

0

</td>

</tr>

<tr>

<td align="right" colspan="6">Taxes</td>

<td>

0

</td>

</tr>

<tr>

<td align="right" colspan="6">Total</td>

<td>

0

</td>

</tr>

</table>

### Layout (included in MVC template, but this is modified)

@{

ViewData["Title"] = "EmptyCart";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Shopping Cart</h1>

<table class="table mList sales" cellpadding="2" cellspacing="2" border="1">

<tr>

<th>Option</th>

<th>Title</th>

<th>Thumbnail</th>

<th>Price</th>

<th>Quantity</th>

<th>Sub Total</th>

<th></th>

</tr>

<tr>

<td align="right" colspan="6">Subtotal</td>

<td>

0

</td>

</tr>

<tr>

<td align="right" colspan="6">Taxes</td>

<td>

0

</td>

</tr>

<tr>

<td align="right" colspan="6">Total</td>

<td>

0

</td>

</tr>

</table>

# CSS (Much of the CSS is from the template, but much is customized or added onto)

/\* Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Provide sufficient contrast against white background \*/

a {

color: #0366d6;

}

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Sticky footer styles

-------------------------------------------------- \*/

html {

font-size: 14px;

}

@media (min-width: 768px) {

html {

font-size: 16px;

}

}

.border-top {

border-top: 1px solid #e5e5e5;

}

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);

}

button.accept-policy {

font-size: 1rem;

line-height: inherit;

}

/\* Sticky footer styles

-------------------------------------------------- \*/

html {

position: relative;

min-height: 100%;

}

body {

/\* Margin bottom by footer height \*/

margin-bottom: 60px;

background-image: linear-gradient(rgb(128, 128, 128), lightblue);

}

.footer {

position: fixed;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 60px; /\* Vertically center the text there \*/

background-color: lightgray;

}

.stay-still {

position: fixed;

width: 100%;

margin-bottom: 1px;

z-index: 2;

}

.movable {

top: 65px;

margin-bottom: 55px;

position: relative;

z-index: 0;

}

img.resize {

max-width: 45%;

max-height: 45%;

}

img.thumbnail {

max-width: 100px;

max-height: 200px;

}

.mList {

font-weight: bold;

}

/\* The Modal (background) \*/

.modal {

display: none; /\* Hidden by default \*/

position: fixed; /\* Stay in place \*/

z-index: 1; /\* Sit on top \*/

left: 0;

top: 0;

width: 100%; /\* Full width \*/

height: 100%; /\* Full height \*/

overflow: auto; /\* Enable scroll if needed \*/

background-color: rgb(0,0,0); /\* Fallback color \*/

background-color: rgba(0,0,0,0.4); /\* Black w/ opacity \*/

background-size: 100% 100%;

}

/\* Modal Content/Box \*/

.modal-content {

background-color: #fefefe;

margin: 15% auto; /\* 15% from the top and centered \*/

padding: 20px;

border: 1px solid #888;

width: 80%; /\* Could be more or less, depending on screen size \*/

text-align: center;

font-size: 18px;

}

/\* The Close Button \*/

.close {

color: #aaa;

float: right;

font-size: 28px;

font-weight: bold;

text-align: right;

}

.close:hover, .close:focus {

color: black;

text-decoration: none;

cursor: pointer;

}

/\* Don't print the print button on the invoice\*/

@media print

{

.no-print, .no-print \* {

display: none !important;

}

}

.privacy {

background-image: linear-gradient(to bottom, rgb(164, 164, 164), rgb(245, 245, 245));

background-repeat: no-repeat;

background-position: center;

background-position-y: bottom;

margin-bottom: 0px;

margin-top: 0px;

}

.home {

background-image: url('../img/flames.gif');

background-color: rgb(0, 0, 0);

background-repeat: repeat-x;

background-position-x: center;

background-position-y: bottom;

margin-bottom: 55px;

color: orangered;

}

.sales {

background-color: rgb(0, 0, 0);

color: orangered;

}

.center {

text-align: center;

}

# JavaScript

// Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification

// for details on configuring this project to bundle and minify static web assets.

// Write your JavaScript code.

function displayModal(ID) {

var mod = ID + "\_modal";

var modal = document.getElementById(mod);

modal.style.display = "block";

modal.addEventListener("click", function () { modal.style.display = "none" });

}

function yesnoCheck() {

if (document.getElementById('yesCheck').checked) {

document.getElementById('ifYes').style.display = 'none';

}

else document.getElementById('ifYes').style.display = 'block';

}

# Helpers

## HelperClass

namespace MvcMovie.Helpers

{

/// <summary>

/// The HelperClass class contains helper functions.

/// </summary>

public class HelperClass

{

/// <summary>

/// The Combine function replaces spaces with underscores.

/// It is used for creating a pattern to div IDs for the JavaScript function that displays modals for the plot description of movies.

/// </summary>

/// <param name="a"></param>

/// <param name="b"></param>

/// <returns></returns>

public static string Combine(string a, string b)

{

return a.Replace(" ", "\_") + b;

}

}

}