Participation 4 – Movies

1. **Index Page**

A screenshot of a computer

Description automatically generated

1. **Edit Page**

**A screenshot of a computer

Description automatically generated**

1. **Delete Page**

A screenshot of a computer

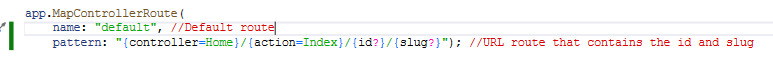
Description automatically generated

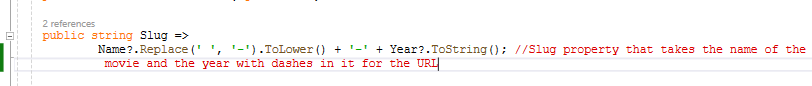
1. **Program.cs code**

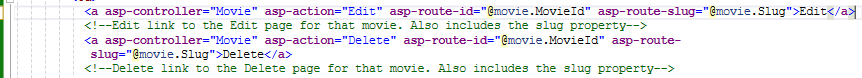
**A screenshot of a computer program

Description automatically generated**

1. **Slugs URL**







1. **HomeController**

A screenshot of a computer program

Description automatically generated

1. **MovieController**

using \_4\_1Movies.Models;

using Microsoft.AspNetCore.Mvc;

namespace \_4\_1Movies.Controllers

{

/// <summary>

/// Movie Controller based on the MVC controller class

/// </summary>

public class MovieController : Controller

{

private MovieContext context { get; set; } //Creates a local instance of the MovieContext class

public MovieController(MovieContext ctx) //Constructs the class and passes in the context class

{

context = ctx; //Sets the local context value to the passed in context

}

/// <summary>

/// Action method to add a new movie to the database

/// </summary>

/// <returns>Edit view but for adding a new movie</returns>

public IActionResult Add()

{

ViewBag.Action = "Add"; //Sets the title to Add

ViewBag.Genres = context.Genres.OrderBy(g => g.Name).ToList(); //Populates the Genre Drop Down

return View("Edit", new Movie()); //Returns the edit with the new movie (id = 0) so that the view knows it's for adding

}

/// <summary>

/// Action method to Edit Getting an existing movie and sending it to the user

/// </summary>

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

/// <returns>Edit view</returns>

[HttpGet]

public IActionResult Edit(int id)

{

ViewBag.Action = "Edit"; //Sets the title to Edit

ViewBag.Genres = context.Genres.OrderBy(g => g.Name).ToList(); //Populates the Genre Drop Down

var movie = context.Movies.Find(id); //Finds the movie information based on the passed id

return View(movie); //Returns the Edit view for the movie information that the id found

}

/// <summary>

/// Action method for Edit Posting the new information to the server

/// </summary>

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

/// <returns></returns>

[HttpPost]

public IActionResult Edit(Movie movie)

{

if (ModelState.IsValid) //If the user inputted valid information

{

if (movie.MovieId == 0) //If the movie is a new movie

{

context.Movies.Add(movie); //Add the movie to the list of the context list

}

else //If the movie is an existing move

{

context.Movies.Update(movie); //Update the movie with the new information

}

context.SaveChanges(); //Saves the movie list back to the database

return RedirectToAction("Index", "Home"); //Returns the view back to the Index Home

}

else //If the user inputted incorrect information

{

ViewBag.Action = (movie.MovieId == 0) ? "Add" : "Edit"; //Sets the title to the correct one

ViewBag.Genres = context.Genres.OrderBy(g => g.Name).ToList(); //Populates the genre list again

return View(movie); //Returns the same Edit view with the passed in movie with errors

}

}

/// <summary>

/// Action method for Delete getting the information and sending it to the user

/// </summary>

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

/// <returns></returns>

[HttpGet]

public IActionResult Delete(int id)

{

ViewBag.Action = "Delete"; //Sets the title to Delete

var movie = context.Movies.Find(id); //Finds the movie information based on the passed ID

return View(movie); //Returns the delete view with the found movie information

}

/// <summary>

/// Action method for Delete posting the information back to the server with the user's answer

/// </summary>

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

/// <returns></returns>

[HttpPost]

public IActionResult Delete(Movie movie)

{

context.Movies.Remove(movie); //Removes the movie from the local list

context.SaveChanges(); //Saves the new list back to the database

return RedirectToAction("Index", "Home"); //Sends the user to the index home view

}

}

}

1. **MovieContext**

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata.Internal;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace \_4\_1Movies.Models

{

public class MovieContext : DbContext

{

/// <summary>

/// MovieContext Constructor with the Connection String and Connection options

/// </summary>

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

public MovieContext(DbContextOptions<MovieContext> options) : base(options)

{}

public DbSet<Movie> Movies { get; set; } = null!; //Database Table called Movies initilized

public DbSet<Genre> Genres { get; set; } = null!; //Database Table called Genres initilized

/// <summary>

/// Creates the Model using the Model Builder parameter

/// </summary>

/// <param name="modelBuilder"></param>

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Genre>().HasData( //Builds the Genre table with the following seed data

new Genre { GenreId = "A", Name = "Action" },

new Genre { GenreId = "C", Name = "Comedy" },

new Genre { GenreId = "D", Name = "Drama" },

new Genre { GenreId = "H", Name = "Horror" },

new Genre { GenreId = "M", Name = "Musical" },

new Genre { GenreId = "R", Name = "RomCom" },

new Genre { GenreId = "S", Name = "SciFi" }

);

modelBuilder.Entity<Movie>().HasData( // Builds the Movie table with the following seed data

new Movie

{

MovieId = 1,

Name = "Casablanca",

Year = 1942,

GenreId = "D",

Rating = 5

},

new Movie

{

MovieId = 2,

Name = "Wonder Woman",

Year = 2017,

GenreId = "A",

Rating = 3

},

new Movie

{

MovieId = 3,

Name = "Moonstruck",

Year = 1988,

GenreId = "R",

Rating = 4

},

new Movie

{

MovieId = 4,

Name = "Peter's Peppers",

Year = 2023,

GenreId = "C",

Rating = 5

}

);

}

}

}