Scenario: Building a Blogging Platform

You are tasked with creating a blogging platform using ASP.NET Core MVC. The platform should allow users to create, edit, and delete blog posts. Each blog post should have a title, content, publication date, and an associated author. Users should be able to view a list of blog posts, click on a post to view its details, and perform CRUD (Create, Read, Update, Delete) operations on blog posts.

1. \*\*Model\*\*: Design the data model for this blogging platform. Define the classes and relationships needed to represent blog posts, authors, and any other relevant entities. Also, consider how you would handle data storage.

2. \*\*View\*\*: Create the necessary views and templates for displaying the list of blog posts, viewing individual blog posts, and providing forms for creating and editing posts. Ensure that the user interface is intuitive and user-friendly.

3. \*\*Controller\*\*: Implement the controllers to handle user requests and interactions. Define actions for displaying the list of posts, creating new posts, editing existing posts, and deleting posts. Make sure to handle validation and error cases gracefully.

4. \*\*Routing\*\*: Configure routing in ASP.NET Core MVC to ensure that URLs are user-friendly and correspond to the expected actions. For example, `/posts` should display the list of posts, and `/posts/create` should lead to the post creation form.

6. \*\*Testing\*\*: Describe how you would test your blogging platform to ensure that it functions correctly. Consider unit tests for controllers and integration tests for the entire application.

Feel free to provide answers or discuss any specific aspect of this case-let question related to ASP.NET Core MVC that you'd like to explore further.

Model

Author

using System.ComponentModel.DataAnnotations;

namespace BlogApplication.Models

{

public class Author

{

[Key]

public int Id { get; set; }

[Required(ErrorMessage ="AuthorName is Mandatory")]

public string AuthorsName { get; set; }

[RegularExpression(@"^([0-9a-zA-Z]([-\.\w]\*[0-9a-zA-Z])\*@([0-9a-zA-Z][-\w]\*[0-9a-zA-Z]\.)+[a-zA-Z]{2,9})$", ErrorMessage = "Invalid email address")]

public string Email { get; set; }

[Required(ErrorMessage = "Mandatory Field,Can't be blank!")]

[RegularExpression(@"^(\+91[\-\s]?)?[0]?(91)?[789]\d{9}$", ErrorMessage = "Invalid phone number")]

public string Phone { get; set; }

public string Description { get; set; }

public string? Pic {get; set; }

}

}

BlogPost

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace BlogApplication.Models

{

public class BlogPost

{

[Key]

public int Id { get; set; }

[Required(ErrorMessage ="Title Name Is Mandatory")]

public string Title { get; set; }

public string BPost { get; set; }

public DateTime Post\_Publish\_DateTime { get; set; }

[Required(ErrorMessage = "Mandatory Field! Can't be blank.")]

public string AuthorsName { get; set; }

//[ForeignKey("AuthorsName")]

//public Author? Author { get; set; }

//public int AuthorId { get; set; }

//[ForeignKey("AuthorId")]

//public Author? Author { get; set; }

//public int TagId { get; set; }

//[ForeignKey("TagId")]

//public CategoryTag? CategoryTag { get; set; }

public ICollection<CategoryTag>? Tags { get; set; }

}

}

Category

using System.ComponentModel.DataAnnotations;

namespace BlogApplication.Models

{

public class CategoryTag

{

[Key]

public int Id { get; set; }

[Required(ErrorMessage ="Category Tag is mandatory")]

public string TagTitle { get; set; }

}

}

Login

using System.ComponentModel.DataAnnotations;

namespace BlogApplication.Models

{

public class Login

{

[Required(ErrorMessage = "User Id is manditory")]

public string Username { get; set; }

[Required(ErrorMessage = "Password is manditory")]

public string Password { get; set; }

}

}

Repository

Author Repository

using BlogApplication.BlogContext;

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Utilities;

using Microsoft.EntityFrameworkCore;

namespace BlogApplication.repositories

{

public class AuthorRepository : IRepository<int, Author>

{

private readonly BContexts \_context;

public AuthorRepository(BContexts contexts)

{

\_context=contexts;

}

static List<Author> autcnt = new List<Author>();

#region Add

public Author Add(Author entity)

{

//entity.Id = GenerateIndex();

\_context.authors.Add(entity);

\_context.SaveChanges();

return entity;

}

#endregion

#region Delete

public Author Delete(int key)

{

var author = GetById(key);

if (author != null)

{

\_context.authors.Remove(author);

\_context.SaveChanges();

return author;

}

throw new NoSuchEntityException("author");

}

#endregion

#region GetAll

public ICollection<Author> GetAll()

{

var author = \_context.authors;

if(author.Count()==0)

{

throw new NoSuchEntityException("author");

}

return author.ToList();

}

#region

// private int GenerateIndex()

//{

// int id = autcnt.Count();

// return ++id;

//}

#endregion

#endregion

#region GetById

public Author GetById(int key)

{

var author = \_context.authors.SingleOrDefault(a=>a.Id == key);

if(author != null)

{

return author;

}

throw new NoSuchEntityException("author");

}

#endregion

#region Update

public Author Update(Author entity)

{

var author = GetById(entity.Id);

if (author == null)

throw new NoSuchEntityException("Author");

\_context.authors.Update(author);

\_context.SaveChanges();

return entity;

}

#endregion

}

}

BlogPostRepository

using BlogApplication.BlogContext;

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Utilities;

namespace BlogApplication.repositories

{

public class BlogPostRepository : IRepository<int , BlogPost>

{

private readonly BContexts \_context;

public BlogPostRepository(BContexts contexts)

{

\_context=contexts;

}

static List<BlogPost> blpcnt = new List<BlogPost>();

public object GetAllBlogPost => throw new NotImplementedException();

#region Add

public BlogPost Add(BlogPost entity)

{

//entity.Id = GenerateIndex();

\_context.blogPosts.Add(entity);

\_context.SaveChanges();

return entity;

}

#endregion

//#region

//private int GenerateIndex()

//{

// int id = blpcnt.Count();

// return ++id;

//}

//#endregion

#region Delete

public BlogPost Delete(int key)

{

var blp = GetById(key);

if(blp != null)

{

\_context.blogPosts.Remove(blp);

\_context.SaveChanges();

return blp;

}

throw new NoSuchEntityException("Blog");

}

#endregion

#region GetAll

public ICollection<BlogPost> GetAll()

{

var blp = \_context.blogPosts;

if (blp.Count() == 0)

{

throw new NoSuchEntityException("Blog");

}

return blp.ToList();

}

#endregion

#region GetById

public BlogPost GetById(int key)

{

var blp = \_context.blogPosts.SingleOrDefault(b=>b.Id == key);

if(blp != null)

{

return blp;

}

throw new NoSuchEntityException("Blog");

}

#endregion

#region Update

public BlogPost Update(BlogPost entity)

{

var blog = GetById(entity.Id);

if (blog == null) {

throw new NoSuchEntityException("Blog");

}

\_context.blogPosts.Update(blog);

\_context.SaveChanges();

return entity;

}

#endregion

}

}

Category Repository

using BlogApplication.BlogContext;

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Utilities;

using Microsoft.EntityFrameworkCore;

namespace BlogApplication.repositories

{

public class CategoryTagRepository : IRepository<int, CategoryTag>

{

private readonly BContexts \_context;

public CategoryTagRepository(BContexts contexts)

{

\_context=contexts;

}

#region

static List<CategoryTag> catcnt = new List<CategoryTag>();

public CategoryTag Add(CategoryTag entity)

{

//entity.Id=GenerateIndex();

\_context.categoryTags.Add(entity);

\_context.SaveChanges();

return entity;

}

#endregion

//#region

//private int GenerateIndex()

//{

// int id = catcnt.Count();

// return ++id;

//}

//#endregion

#region Delete

public CategoryTag Delete(int key)

{

var tag = GetById(key);

if(tag != null)

{

\_context.categoryTags.Remove(tag);

\_context.SaveChanges();

return tag;

}

throw new NoSuchEntityException("CategoryTag");

}

#endregion

#region GetAll

public ICollection<CategoryTag> GetAll()

{

var tag = \_context.categoryTags;

if (tag.Count() == 0)

throw new NoSuchEntityException("Category");

return tag.ToList();

}

#endregion

#region GetById

public CategoryTag GetById(int key)

{

var tag = \_context.categoryTags.SingleOrDefault(c=>c.Id == key);

if(tag != null)

{

return tag;

}

throw new NoSuchEntityException("Category");

}

public CategoryTag Update(CategoryTag entity)

{

var tag = GetById(entity.Id);

if(tag == null)

{

throw new NoSuchEntityException("Category");

}

\_context.categoryTags.Update(tag);

\_context.SaveChanges();

return entity;

}

#endregion

}

}

InterFaces

Irepository

namespace BlogApplication.Interfaces

{

public interface IRepository<K,T> where T : class

{

public T Add(T entity);

public T Update(T entity);

public T Delete(K key);

public T GetById(K key);

public ICollection<T> GetAll();

}

}

ILoginService

using BlogApplication.Models;

namespace BlogApplication.Interfaces

{

public interface ILoginService

{

public Author LoginCheck(Login login);

}

}

IcategoryService

using BlogApplication.Models;

namespace BlogApplication.Interfaces

{

public interface ICategoryTagService

{

public CategoryTag UpdateTitle(CategoryTag tag);

public IList<CategoryTag> GetAllCategory();

}

}

IBlogPostService

using BlogApplication.Models;

namespace BlogApplication.Interfaces

{

public interface IBlogPostService

{

public BlogPost UpdateTitle(BlogPost blogPost);

public BlogPost UpdateBPost(BlogPost blogPost);

//public BlogPost UpdateAuthorId(BlogPost blogPost);

//public BlogPost UpdateTagId(BlogPost blogPost);

public IList<BlogPost> GetAllBlogPost();

}

}

IAuthorService

using BlogApplication.Models;

namespace BlogApplication.Interfaces

{

public interface IAuthorService

{

public Author UpdateName(Author author);

public Author UpdateEmail(Author author);

public Author UpdateDesc(Author author);

public Author UpdatePic(Author author);

public IList<Author> GetAllAuthor();

}

}

BContexts

using BlogApplication.Models;

using Microsoft.EntityFrameworkCore;

using System.Numerics;

namespace BlogApplication.BlogContext

{

public class BContexts :DbContext

{

public BContexts(DbContextOptions options): base(options)

{

}

#region CollectionToTable

public DbSet<Author> authors { get; set; }// author

public DbSet<BlogPost> blogPosts { get; set; } //blogpost

public DbSet<CategoryTag> categoryTags { get; set; } // categoryTag

#endregion

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Author>().HasData(

new Author

{

Id = 1,

AuthorsName = "ABC",

Phone = "8856904770",

Description = "full time Blooger",

Email="abc@gmail.com",

Pic = "/images/pic1.png"

},

new Author

{

Id = 2,

AuthorsName = "DEF",

Description = "Bloger on Social Media",

Email="def@gmail.com",

Phone = "8856904771",

Pic = "/images/pic2.png"

});

modelBuilder.Entity<CategoryTag>().HasData(

new CategoryTag

{

Id = 1,

TagTitle = "Social"

},

new CategoryTag

{

Id = 2,

TagTitle="animal"

},

new CategoryTag

{

Id = 3,

TagTitle="country"

}

);

modelBuilder.Entity<BlogPost>().HasData(

new BlogPost

{

Id = 1,

AuthorsName = "ABC",

Title = "Blog",

BPost = "blog, in full Web log or Weblog, online journal where an individual, group, or corporation presents a record of activities, thoughts, or beliefs. Some blogs operate mainly as news filters, collecting various online sources and adding short comments and Internet links. Other blogs concentrate on presenting original material. In addition, many blogs provide a forum to allow visitors to leave comments and interact with the publisher. “To blog” is the act of composing material for a blog. Materials are largely written, but pictures, audio, and videos are important elements of many blogs. The “blogosphere” is the online universe of blogs.",

//AuthorId = 1,

//TagId = 1

},

new BlogPost

{

Id = 2,

Title = "Tiger",

AuthorsName = "DEF",

BPost = "The tiger has a muscular body with strong forelimbs, a large head and a tail that is about half the length of its body. Its pelage colouration varies between shades of orange with a white underside and distinctive vertical black stripes; the patterns of which are unique in each individual",

// AuthorId = 2,

// TagId = 2

//}

}

);

}

}

}

Program.cs

using Microsoft.EntityFrameworkCore;

using BlogApplication.Interfaces;

using System.Numerics;

using BlogApplication.BlogContext;

using BlogApplication.repositories;

using BlogApplication.Models;

using BlogApplication.Services;

namespace HospitalClinicApp

{

public class Program

{

public static void Main(string[] args)

{

var builder = WebApplication.CreateBuilder(args);

// Add services to the container.

builder.Services.AddControllersWithViews();

#region AddingConetexts

builder.Services.AddDbContext<BContexts>(opts =>

{

opts.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection"));

});

#endregion

builder.Services.AddSession(opts =>

{

opts.IdleTimeout = TimeSpan.FromMinutes(5);

});

builder.Services.AddMemoryCache();

#region injection

builder.Services.AddScoped<IRepository<int , Author >,AuthorRepository>();

builder.Services.AddScoped<IRepository<int, BlogPost>, BlogPostRepository>();

builder.Services.AddScoped<IRepository<int, CategoryTag>, CategoryTagRepository>();

builder.Services.AddScoped<IAuthorService , AuthorService>();

builder.Services.AddScoped<IBlogPostService, BlogPostService>();

builder.Services.AddScoped<ICategoryTagService, CategoryTagService>();

builder.Services.AddScoped<LoginService>();

#endregion

var app = builder.Build();

AppContext.SetSwitch("Npgsql.EnableLegacyTimestampBehavior", true);

// Configure the HTTP request pipeline.

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/Home/Error");

}

app.UseStaticFiles();

app.UseRouting();

app.UseSession();

app.UseResponseCaching();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

pattern: "{controller=Login}/{action=UserLogin}/{id?}");

app.Run();

}

}

}

Application settings

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"ConnectionStrings": {

"DefaultConnection": "Host=localhost;Port=5432;Database=dbblog;UserId=postgres;Password=root;"

},

"AllowedHosts": "\*"

}

Services

Author service

using BlogApplication.Interfaces;

using BlogApplication.Models;

namespace BlogApplication.Services

{

public class AuthorService : IAuthorService

{

private readonly IRepository<int, Author> \_repository;

public AuthorService( IRepository<int , Author> repository)

{

\_repository = repository;

}

#region GetAll Author

public IList<Author> GetAllAuthor()

{

return \_repository.GetAll().ToList();

}

#endregion

#region update description

public Author UpdateDesc(Author author)

{

var myauthor = \_repository.GetById(author.Id);

myauthor.Description = author.Description;

\_repository.Update(myauthor);

return myauthor;

}

#endregion

#region Update Email

public Author UpdateEmail(Author author)

{

var myauthor = \_repository.GetById(author.Id);

myauthor.Email = author.Email;

\_repository.Update(myauthor);

return myauthor;

}

#endregion

#region Update Name

public Author UpdateName(Author author)

{

var myauthor = \_repository.GetById(author.Id);

myauthor.AuthorsName = author.AuthorsName;

\_repository.Update(myauthor);

return myauthor;

}

#endregion

#region UpdatePic

public Author UpdatePic(Author author)

{

var myauthor = \_repository.GetById(author.Id);

myauthor.Pic = author.Pic;

\_repository.Update(myauthor);

return myauthor;

}

#endregion

}

}

BlogPostService

using BlogApplication.Interfaces;

using BlogApplication.Models;

namespace BlogApplication.Services

{

public class BlogPostService : IBlogPostService

{

private readonly IRepository<int, BlogPost> \_repository;

public BlogPostService(IRepository<int ,BlogPost> repository)

{

\_repository = repository;

}

#region Get all Blog

public IList<BlogPost> GetAllBlogPost()

{

return \_repository.GetAll().ToList();

}

#endregion

//#region Update Author id for that blog

//public BlogPost UpdateAuthorId(BlogPost blogPost)

//{

// var blp = \_repository.GetById(blogPost.BlogPostId);

// //blp.AuthorId = blogPost.AuthorId;

// \_repository.Update(blp);

// return blp;

//}

//#endregion

public BlogPost UpdateBPost(BlogPost blogPost)

{

var blp = \_repository.GetById(blogPost.Id);

blp.BPost = blogPost.BPost;

blp.Title = blogPost.Title;

\_repository.Update(blp);

return blp;

}

//public BlogPost UpdateTagId(BlogPost blogPost)

//{

// var blp = \_repository.GetById(blogPost.BlogPostId);

// //blp.TagId = blogPost.TagId;

// \_repository.Update(blp);

// return blp;

//}

public BlogPost UpdateTitle(BlogPost blogPost)

{

var blp = \_repository.GetById(blogPost.Id);

blp.Title = blogPost.Title;

\_repository.Update(blp);

return blp;

}

}

}

Category service

using BlogApplication.Interfaces;

using BlogApplication.Models;

namespace BlogApplication.Services

{

public class CategoryTagService : ICategoryTagService

{

private readonly IRepository<int, CategoryTag> \_repository;

public CategoryTagService(IRepository <int , CategoryTag> repository)

{

\_repository = repository;

}

public CategoryTag UpdateTitle(CategoryTag tag)

{

var t = \_repository.GetById(tag.Id);

t.TagTitle = tag.TagTitle;

\_repository.Update(t);

return t;

}

public IList<CategoryTag> GetAllCategory()

{

return \_repository.GetAll().ToList();

}

}

}

Login Service

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Utilities;

namespace BlogApplication.Services

{

public class LoginService : ILoginService

{

private readonly IRepository<int, Author> \_repository;

public LoginService(IRepository<int, Author> repository)

{

\_repository = repository;

}

public Author LoginCheck(Login login)

{

var author = \_repository.GetAll();

var myAuthor = author.SingleOrDefault(x => x.AuthorsName == login.Username && x.Phone == login.Password);

if (myAuthor == null)

{

throw new InvalidCredentialsException();

}

return myAuthor;

}

}

}

Exceptions

namespace BlogApplication.Utilities

{

public class InvalidCredentialsException : Exception

{

public override string Message => "Invalid username or password";

}

}

namespace BlogApplication.Utilities

{

public class NoSuchEntityException : Exception

{

string message;

public NoSuchEntityException(string name)

{

message = $"No Such {name} found...!!";

}

public override string Message => base.Message;

}

}

namespace BlogApplication.Utilities

{

public class UnableToAddException : Exception

{

public override string Message => "Unable to add";

}

}

namespace BlogApplication.Utilities

{

public class UnableToUpdateException : Exception

{

string message;

public UnableToUpdateException(string name)

{

message = $"Unnable to update {name} ...!!";

}

public override string Message => base.Message;

}

}

Controller

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Utilities;

using Microsoft.AspNetCore.Mvc;

using System.Diagnostics;

using System.Numerics;

namespace BlogApplication.Controllers

{

public class AuthorController : Controller

{

private readonly IAuthorService \_authorService;

private readonly ILogger<AuthorController> \_logger;

private readonly IRepository<int, Author> \_repository;

public AuthorController(IAuthorService authorService, ILogger<AuthorController> logger, IRepository<int, Author> repository)

{

\_authorService = authorService;

\_logger = logger;

\_repository = repository;

}

static List<Author> MyAuthor = new List<Author>();

#region Index

public IActionResult Index()

{

var authors = new List<Author>();

try

{

if (ViewBag.User = HttpContext.Session.GetString("Username").ToString() == null)

{

return RedirectToAction("UserLogin", "Login");

}

//ViewBag.Username = HttpContext.Session.GetString("Username");

//authors = \_authorService.GetAllAuthor().ToList();

}

catch (Exception e)

{

ViewBag.ErrorMessage = e.Message;

\_logger.LogError("No Blog found..!!");

}

return View(authors = \_authorService.GetAllAuthor().ToList());

}

#endregion

#region Details

[HttpGet]

public IActionResult Details(int id)

{

var a = \_repository.GetById(id);

return View(a);

}

#endregion

#region Create Author author

[HttpGet]

public IActionResult SignUpAuthor()

{

return View();

}

[HttpPost]

public IActionResult SignUpAuthor(Author author)

{

try

{

//author.Pic = "/images/" + author.Pic;

var aut =\_repository.Add(author);

\_logger.LogInformation("Created Author record");

return RedirectToAction("UserLogin" , "Login");

}

catch (Exception e)

{

ViewBag.ErrorMessage = e.Message;

//Logging for programmer

\_logger.LogError("Unable to add doctor " + DateTime.Now.ToString());

}

return View(author);

}

#endregion

#region Update drscription

[HttpGet]

public IActionResult UpdateDescription(int id)

{

var aut = \_authorService.GetAllAuthor().SingleOrDefault(x => x.Id == id);

var author = new Author { Id = id, Description = aut.Description };

return View(author);

}

[HttpPost]

public IActionResult UpdateDescription(int id, Author author)

{

try

{

var result = \_authorService.UpdateDesc(author);

return RedirectToAction("Index");

}

catch (Exception e)

{

throw new NoSuchEntityException("Author");

}

}

#endregion

#region Update

[HttpGet]

public IActionResult Update(int id)

{

var aut = \_repository.GetById(id);

return View(aut);

//var aut = \_authorService.GetAllAuthor().SingleOrDefault(x => x.AuthorId == id);

//var author = new Author { AuthorId = id, Description = aut.Description };

//return View(author);

}

[HttpPost]

public IActionResult Update(int id, Author author)

{

ViewBag.Message = "";

try

{

\_repository.Update(author);

return RedirectToAction("Index");

//var a = \_repository.GetById(author.AuthorId);

//var result = \_repository.Update(author);

//return RedirectToAction("Index");

}

catch (Exception e)

{

ViewBag.Message = "Unable to update author";

Debug.WriteLine(e.Message);

//throw new NoSuchEntityException("Author"

}

return View(author);

}

#endregion

#region Update Name

[HttpGet]

public IActionResult UpdateName(int id)

{

var aut = \_authorService.GetAllAuthor().SingleOrDefault(x => x.Id == id);

var author = new Author { Id = id, AuthorsName = aut.AuthorsName };

return View(author);

}

[HttpPost]

public IActionResult UpdateName(int id, Author author)

{

try

{

var result = \_authorService.UpdateName(author);

return RedirectToAction("Index");

}

catch (Exception e)

{

throw new NoSuchEntityException("Author");

}

}

#endregion

#region Update Email

[HttpGet]

public IActionResult UpdateEmail(int id)

{

var aut = \_authorService.GetAllAuthor().SingleOrDefault(x => x.Id == id);

var author = new Author { Id = id, Email = aut.Email };

return View(author);

}

[HttpPost]

public IActionResult UpdateEmail(int id, Author author)

{

try

{

var result = \_authorService.UpdateEmail(author);

return RedirectToAction("Index");

}

catch (Exception e)

{

throw new NoSuchEntityException("Author");

}

}

#endregion

#region Delete

[HttpGet]

public IActionResult Delete(int id)

{

var aut = \_repository.GetById(id);

return View(aut);

//Author aut = MyAuthor.SingleOrDefault(x=>x.AuthorId== id);

//return View(aut);

}

[HttpPost]

public IActionResult Delete(int id, Author author)

{

//MyAuthor.Remove(author);

//return RedirectToAction("Index");

ViewBag.Message = "";

try

{

\_repository.Delete(id);

return RedirectToAction("Index");

}

catch (Exception e)

{

Debug.WriteLine(e.Message);

ViewBag.Message = "Unable to delete author";

}

return View(author);

}

#endregion

}

}

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Services;

using BlogApplication.Utilities;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore.Storage;

using System.Reflection.Metadata;

using static System.Reflection.Metadata.BlobBuilder;

namespace BlogApplication.Controllers

{

public class BlogPostController : Controller

{

#region injection

private readonly IBlogPostService \_blogPostService;

private readonly ILogger<BlogPost> \_logger;

private readonly IRepository<int, BlogPost> \_repository;

private readonly ICategoryTagService \_Tag;

private readonly IAuthorService \_author;

public BlogPostController(IBlogPostService blogPostService,

ICategoryTagService categoryTagService,

ILogger<BlogPost> logger,

IRepository<int , BlogPost> repository,

IAuthorService authorService)

{

\_blogPostService = blogPostService;

\_logger = logger;

\_repository = repository;

\_Tag=categoryTagService;

\_author= authorService;

}

#endregion

#region Index

static List<BlogPost> Myblp = new List<BlogPost>();

public IActionResult Index()

{

var blogs = new List<BlogPost>();

try

{

blogs = \_blogPostService.GetAllBlogPost().ToList();

var authBlogs = blogs.Where(blog => blog.AuthorsName.Contains

(ViewBag.User = HttpContext.Session.GetString("Username").ToString()));

if (authBlogs != null)

return View(authBlogs);

//blogPosts = \_repository.GetAll().ToList();

//return View(blogPosts);

}

catch (Exception e)

{

\_logger.LogInformation("There are no blogPost");

}

return View(blogs);

}

#endregion

#region details

[HttpGet]

public IActionResult Details(int id)

{

BlogPost b = \_repository.GetById(id);

return View(b);

}

#endregion

#region Create blogPost

[HttpGet]

public IActionResult CreateBlogPost(int id)

{

//ViewBag.BlogPosts = GetCategoryTag();

//ViewBag.authors = GetAuthor();

//BlogPost blogPost= new BlogPost();

// blogPost.AuthorId=id;

// blogPost.Post\_Publish\_DateTime = DateTime.Now;

return View();

}

[HttpPost]

public IActionResult CreateBlogPost(int id, BlogPost blogPost)

{

//ViewBag.BlogPosts = GetCategoryTag();

//ViewBag.authors = GetAuthor();

if (ModelState.IsValid)

{

try

{

var myblp = \_repository.Add(blogPost);

return RedirectToAction("index");

}

catch (Exception e)

{

\_logger.LogInformation("Unable to add BlogPost");

throw new UnableToAddException();

}

}

return View();

}

#endregion

#region Get category

private List<SelectListItem> GetCategoryTag()

{

List<SelectListItem> tag = new List<SelectListItem>();

foreach(var item in \_Tag.GetAllCategory())

{

var li = new SelectListItem()

{

Text = item.Id.ToString(),

Value = item.Id.ToString()

};

tag.Add(li);

}return tag;

}

#endregion

#region Get Author

private List<SelectListItem> GetAuthor()

{

List<SelectListItem> aut = new List<SelectListItem>();

foreach (var item in \_author.GetAllAuthor())

{

var li = new SelectListItem()

{

Text = item.Id.ToString(),

Value = item.Id.ToString()

};

aut.Add(li);

}

return aut;

}

#endregion

#region Update Bpost

[HttpGet]

public IActionResult UpdateBlogPost(int id)

{

var aut = \_blogPostService.GetAllBlogPost().SingleOrDefault(x => x.Id == id);

var blp = new BlogPost { Id = id };

return View(blp);

}

[HttpPost]

public IActionResult UpdateBlogPost(int id, BlogPost blogPost)

{

try

{

var result = \_blogPostService.UpdateBPost(blogPost);

return RedirectToAction("Index");

} // method oupput update hot nahi

catch (Exception)

{

\_logger.LogError("Unable to Update Bpost" + DateTime.Now.ToString());

throw new UnableToUpdateException("Bpost");

}

}

#endregion

#region Update TagId

//[HttpGet]

//public IActionResult UpdateT(int id)

//{

// var aut = \_blogPostService.GetAllBlogPost().SingleOrDefault(x => x.Id == id);

// var blp = new BlogPost { Id = id };

// return View(blp);

//}

//[HttpPost]

//public IActionResult UpdateT(int id, BlogPost blogPost)

//{

// try

// {

// var result = \_blogPostService.UpdateTitle(blogPost);

// return RedirectToAction("Index");

// }

// catch (Exception)

// {

// \_logger.LogError("Unable to Update TagId" + DateTime.Now.ToString());

// throw new UnableToUpdateException("TagId");

// }

//}

#endregion

#region Delete

[HttpGet]

public IActionResult Delete(int id)

{

var blp = \_repository.GetById(id);

return View(blp);

}

[HttpPost]

public IActionResult Delete(int id, BlogPost blp)

{

try

{

\_repository.Delete(id);

return RedirectToAction("Index");

}

catch (Exception e)

{

ViewBag.ErrorMessage = e.Message;

\_logger.LogError("Unable to delete Blog");

}

return View();

}

#endregion

#region Update

//[HttpGet]

//public IActionResult Update(int id)

//{

// var aut = \_blogPostService.GetAllBlogPost().SingleOrDefault(x => x.Id == id);

// var author = new Author { Id = id };

// return View(author);

//}

//[HttpPost]

//public IActionResult Update(int id, BlogPost blp)

//{

// try

// {

// var result = \_repository.Update(blp);

// return RedirectToAction("Index" , "BlogPost");

// }

// catch (Exception e)

// {

// throw new NoSuchEntityException("Author");

// }

//}

#endregion

#region Delete Blog

//[HttpGet]

//public IActionResult Delete(int id)

//{

// var blog = \_repository.GetByID(id);

// return View(blog);

//}

//[HttpPost]

//public IActionResult Delete(Blog blog)

//{

// ViewBag.ErrorMessage = "";

// try

// {

// \_blogService.Delete(blog);

// return RedirectToAction("Index");

// }

// catch (Exception e)

// {

// ViewBag.ErrorMessage = e.Message;

// \_logger.LogError("Unable to delete Blog");

// }

// return View();

//}

#endregion

#region Update TagId

//[HttpGet]

//public IActionResult UpdateTagId(int id)

//{

// var aut = \_blogPostService.GetAllBlogPost().SingleOrDefault(x => x.BlogPostId == id);

// var blp = new BlogPost { BlogPostId = id };

// return View(blp);

//}

//[HttpPost]

//public IActionResult UpdateTagId(int id, BlogPost blogPost)

//{

// try

// {

// var result = \_blogPostService.UpdateTagId(blogPost);

// return RedirectToAction("Index");

// }

// catch (Exception)

// {

// \_logger.LogError("Unable to Update TagId" + DateTime.Now.ToString());

// throw new UnableToUpdateException("TagId");

// }

//}

#endregion

}

}

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Utilities;

using Microsoft.AspNetCore.Mvc;

namespace BlogApplication.Controllers

{

public class CategoryTagController : Controller

{

#region injection

private readonly ICategoryTagService \_service;

private readonly ILogger<CategoryTagController> \_logger;

private readonly IRepository<int, CategoryTag> \_repository;

public CategoryTagController(ICategoryTagService categoryTagService ,IRepository<int , CategoryTag> repository,ILogger <CategoryTagController> logger)

{

\_service = categoryTagService;

\_logger = logger;

\_repository = repository;

}

#endregion

#region index

public ActionResult Index()

{

var tag = new List<CategoryTag>();

try

{

tag = \_service.GetAllCategory().ToList();

}

catch (Exception)

{

\_logger.LogError("No Tag Available");

}

return View(tag);

}

#endregion

#region Create tag

[HttpGet]

public ActionResult Createtag()

{

return View();

}

[HttpPost]

public ActionResult Createtag(CategoryTag tag)

{

try

{

var mytag = \_repository.Add(tag);

\_logger.LogInformation("created tag record");

return RedirectToAction("Index");

}

catch (Exception e)

{

\_logger.LogError("Unable to add tag " + DateTime.Now.ToString());

}

return View();

}

#endregion

#region Update

[HttpGet]

public IActionResult UpdateT(int id)

{

var t =\_service.GetAllCategory().SingleOrDefault(x=>x.Id == id);

var mytag = new CategoryTag { Id = id };

return View(mytag);

}

[HttpPost]

public IActionResult UpdateT(int id , CategoryTag tag)

{

try

{

var result = \_service.UpdateTitle(tag);

return RedirectToAction("Index");

}

catch (Exception)

{

\_logger.LogError("Unable to update");

throw new UnableToUpdateException("TagTitle");

}

}

#endregion

}

}

using BlogApplication.Models;

using Microsoft.AspNetCore.Mvc;

using System.Diagnostics;

namespace BlogApplication.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

try

{

ViewBag.User = HttpContext.Session.GetString("Username").ToString();

}

catch (Exception)

{

return RedirectToAction("UserLogin", "Login");

}

return View();

}

public IActionResult Privacy()

{

return View();

}

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

public IActionResult Error()

{

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

}

}

}

using BlogApplication.Interfaces;

using BlogApplication.Models;

using BlogApplication.Services;

using BlogApplication.Utilities;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using System.Security.Authentication;

namespace BlogApplication.Controllers

{

public class LoginController : Controller

{

private readonly LoginService \_loginservice;

public LoginController(LoginService service)

{

\_loginservice=service;

}

[HttpGet]

//Route("Login")]

public IActionResult UserLogin()

{

return View();

}

[HttpPost]

//[Route("Login")]

public IActionResult UserLogin(Login login)

{

if (ModelState.IsValid)

{

try

{

var author = \_loginservice.LoginCheck(login);

if (author != null)

{

HttpContext.Session.SetString("Username", login.Username);

return RedirectToAction("Index", "Home");

}

ViewBag.ErrorMessage = "Invalid username or password";

return View(login);

}

catch (Exception)

{

ViewBag.ErrorMessage = "Invalid username or password";

return View(login);

}

}

return View(login);

//var result =\_loginservice.LoginCheck(login);

// if(result!= null)

// {

// HttpContext.Session.SetString("Username", login.Username);

// return RedirectToAction("Index", "Home");

// }

// ViewBag.ErrorMessage = "Invalid username or password";

// return View(login);

}

}

}

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

namespace BlogApplication.Controllers

{

public class ProfileController : Controller

{

[HttpGet]

public IActionResult UserProfile()

{

try

{

ViewBag.User = HttpContext.Session.GetString("Username").ToString();

}

catch (Exception)

{

return RedirectToAction("UserLogin", "User");

}

return View();

}

[HttpGet]

public IActionResult UserLogout(string un)

{

HttpContext.Session.Clear();

return RedirectToAction( "UserLogin" , "Login");

}

}

}

Virews

Authors

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

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

<div>

<h4>Author</h4>

<hr />

<dl class="row">

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

</dl>

<form asp-action="Delete">

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

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

</form>

</div>

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "Details";

}

<h1>Details</h1>

<div>

<h4>Author</h4>

<hr />

<dl class="row">

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

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

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

</dt>

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

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

</dd>

</dl>

</div>

<div>

@Html.ActionLink("Uodate", "Update", new { id=Model.Id/\* id = Model.PrimaryKey \*/ }) |

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

</div>

@model IEnumerable<BlogApplication.Models.Author>

@{

ViewData["Title"] = "Index";

}

<style>

#docDiv {

float: left;

position: relative;

margin: 5px;

}

</style>

<h1>Index</h1>

@foreach (var item in Model)

{

<div class="card" id="AutDiv" style="width: 18rem;">

<img style="height:200px;width:200px;" class="card-img-top" src="@item.Pic" alt="Card image cap">

<div class="card-body">

<h5 class="card-title">@item.AuthorsName</h5>

<p class="card-text">The Aut has @item.Description and is a @item.Email</p>

@\* <a asp-controller="Author" asp-action="Details" asp-route-id="@item.AuthorId" class="btn btn-primary">View Details</a> \*@

<a asp-controller="BlogPost" asp-action="CreateBlogPost" asp-route-id="@item.Id" class="btn btn-primary">New Blog</a>

</div>

</div>

<td>

@Html.ActionLink("Update", "Update", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

}

<div class="alert alert-danger">

@ViewBag.ErrorMessage

</div>

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "SignUpAuthor";

}

<h1>SignUpAuthor</h1>

<h4>Author</h4>

<hr />

<div class="row">

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

<form asp-action="SignUpAuthor">

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

@\* <div class="form-group">

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

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

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

</div> \*@

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

<span asp-validation-for="Phone" 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">

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

<input type="file" asp-for="Pic" class="form-control" />

<span asp-validation-for="Pic" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "Update";

}

<h1>Update</h1>

<h4>Author</h4>

<hr />

<div class="row">

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

<form asp-action="Update">

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

@\* <div class="form-group">

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

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

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

</div> \*@

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

<span asp-validation-for="Phone" 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">

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

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

<span asp-validation-for="Pic" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "UpdateDescription";

}

<h1>UpdateDescription</h1>

<h4>Author</h4>

<hr />

<div class="row">

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

<form asp-action="UpdateDescription">

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

@\* <div class="form-group">

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

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

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

</div> \*@

@\* <div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

<span asp-validation-for="Phone" 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">

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

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

<span asp-validation-for="Pic" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "UpdateEmail";

}

<h1>UpdateEmail</h1>

<h4>Author</h4>

<hr />

<div class="row">

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

<form asp-action="UpdateEmail">

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

@\* <div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

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

</div> \*@

<div class="form-group">

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

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

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

</div>

@\* <div class="form-group">

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

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

<span asp-validation-for="Phone" 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">

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

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

<span asp-validation-for="Pic" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model BlogApplication.Models.Author

@{

ViewData["Title"] = "UpdateName";

}

<h1>UpdateName</h1>

<h4>Author</h4>

<hr />

<div class="row">

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

<form asp-action="UpdateName">

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

@\* <div class="form-group">

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

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

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

</div> \*@

<div class="form-group">

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

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

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

</div>

@\* <div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

<span asp-validation-for="Phone" 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">

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

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

<span asp-validation-for="Pic" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

BlogPosts

@model BlogApplication.Models.BlogPost

@{

ViewData["Title"] = "CreateBlogPost";

}

<h1>CreateBlogPost</h1>

<h4>BlogPost</h4>

<hr />

<div class="row">

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

<form asp-action="CreateBlogPost">

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

@\* <div class="form-group">

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

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

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

</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="BPost" class="control-label"></label>

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

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

</div>

<div class="form-group">

<label asp-for="Post\_Publish\_DateTime" class="control-label"></label>

<input asp-for="Post\_Publish\_DateTime" class="form-control" />

<span asp-validation-for="Post\_Publish\_DateTime" class="text-danger"></span>

</div>

<div class="form-group">

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

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

<span asp-validation-for="AuthorsName" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model BlogApplication.Models.BlogPost

@{

ViewData["Title"] = "Delete";

}

<h1>Delete</h1>

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

<div>

<h4>BlogPost</h4>

<hr />

<dl class="row">

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

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

</dt>

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

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

</dd>

<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.BPost)

</dt>

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

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

</dd>

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

@Html.DisplayNameFor(model => model.Post\_Publish\_DateTime)

</dt>

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

@Html.DisplayFor(model => model.Post\_Publish\_DateTime)

</dd>

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

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

</dt>

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

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

</dd>

</dl>

<form asp-action="Delete">

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

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

</form>

</div>

@model BlogApplication.Models.BlogPost

@{

ViewData["Title"] = "Details";

}

<h1>Details</h1>

<div>

<h4>BlogPost</h4>

<hr />

<dl class="row">

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

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

</dt>

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

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

</dd>

<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.BPost)

</dt>

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

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

</dd>

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

@Html.DisplayNameFor(model => model.Post\_Publish\_DateTime)

</dt>

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

@Html.DisplayFor(model => model.Post\_Publish\_DateTime)

</dd>

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

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

</dt>

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

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

</dd>

</dl>

</div>

<div>

@Html.ActionLink("Update", "Update", new { id=Model.Id/\* id = Model.PrimaryKey \*/ }) |

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

</div>

@model IEnumerable<BlogApplication.Models.BlogPost>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

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

</p>

<table class="table">

<thead>

<tr>

<th>

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

</th>

<th>

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

</th>

<th>

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

</th>

<th>

@Html.DisplayNameFor(model => model.Post\_Publish\_DateTime)

</th>

<th>

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

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.DisplayFor(modelItem => item.Post\_Publish\_DateTime)

</td>

<td>

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

</td>

<td>

@Html.ActionLink("Update", "UpdateBlogPost", new { id=item.Id }) |

@Html.ActionLink("Details", "Details", new { id=item.Id }) |

@Html.ActionLink("Delete", "Delete", new { id=item.Id })

</td>

</tr>

}

</tbody>

</table>

@model BlogApplication.Models.BlogPost

@{

ViewData["Title"] = "UpdateBlogPost";

}

<h1>UpdateBlogPost</h1>

<h4>BlogPost</h4>

<hr />

<div class="row">

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

<form asp-action="UpdateBlogPost">

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

@\* <div class="form-group">

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

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

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

</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="BPost" class="control-label"></label>

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

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

</div>

<div class="form-group">

<label asp-for="Post\_Publish\_DateTime" class="control-label"></label>

<input asp-for="Post\_Publish\_DateTime" class="form-control" />

<span asp-validation-for="Post\_Publish\_DateTime" class="text-danger"></span>

</div>

<div class="form-group">

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

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

<span asp-validation-for="AuthorsName" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model BlogApplication.Models.CategoryTag

@{

ViewData["Title"] = "Createtag";

}

<h1>Createtag</h1>

<h4>CategoryTag</h4>

<hr />

<div class="row">

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

<form asp-action="Createtag">

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

<span asp-validation-for="TagTitle" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@model IEnumerable<BlogApplication.Models.CategoryTag>

@{

ViewData["Title"] = "Index";

}

<h1>Index</h1>

<p>

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

</p>

<table class="table">

<thead>

<tr>

<th>

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

</th>

<th>

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

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

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

</td>

<td>

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

</td>

<td>

@Html.ActionLink("Update", "Update", new { id=item.Id /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Details", "Details", new {id=item.Id /\* id=item.PrimaryKey \*/ }) |

@Html.ActionLink("Delete", "Delete", new {id=item.Id /\* id=item.PrimaryKey \*/ })

</td>

</tr>

}

</tbody>

</table>

@model BlogApplication.Models.CategoryTag

@{

ViewData["Title"] = "UpdateT";

}

<h1>UpdateT</h1>

<h4>CategoryTag</h4>

<hr />

<div class="row">

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

<form asp-action="UpdateT">

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

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

<span asp-validation-for="TagTitle" 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="Index">Back to List</a>

</div>

@section Scripts {

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

}

@{

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

}

<div class="text-center">

<h1 class="display-4">Welcome @ViewBag.User</h1>

</div>

Login

@model BlogApplication.Models.Login

@{

ViewData["Title"] = "UserLogin";

}

<h4>Login</h4>

<hr />

<div class="row">

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

<form asp-action="UserLogin">

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

<div class="form-group">

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

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

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

</div>

<div class="form-group">

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

<input type="password" asp-for="Password" class="form-control" />

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

</div>

<br />

<div class="form-group">

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

<a class="btn btn-primary" asp-area="" asp-controller="Author" asp-action="SignUpAuthor">SignUp</a>

<br/>

</div>

</form>

</div>

</div>

@if (ViewBag.ErrorMessage != null)

{

<div class="alert alert-danger">

@ViewBag.ErrorMessage

</div>

}

@section Scripts {

@{

await Html.RenderPartialAsync("\_ValidationScriptsPartial");

}

}

Profile

@model BlogApplication.Models.Login

@{

ViewData["Title"] = "UserProfile";

}

<h1>UserProfile</h1>

<div>

<h4>User</h4>

<hr />

<dl class="row">

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

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

</dt>

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

<span>@ViewBag.User</span>

</dd>

<a asp-action="UserLogout" asp-controller="Profile" asp-route-un="@ViewBag.User" class="btn btn-primary">Logout</a>

</dl>

</div>

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - BlogApplication</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />

<link rel="stylesheet" href="~/BlogApplication.styles.css" asp-append-version="true" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container-fluid">

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">BlogApplication</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Profile" asp-action="UserProfile">Profile</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Author" asp-action="Index">Authors</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="BlogPost" asp-action="Index">Blogs</a>

</li>

@\* <li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Author" asp-action="Index">Authors</a>

</li> \*@

@\* <li class="nav-item">

<a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</li> \*@

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2023 - BlogApplication - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@await RenderSectionAsync("Scripts", required: false)

</body>

</html>