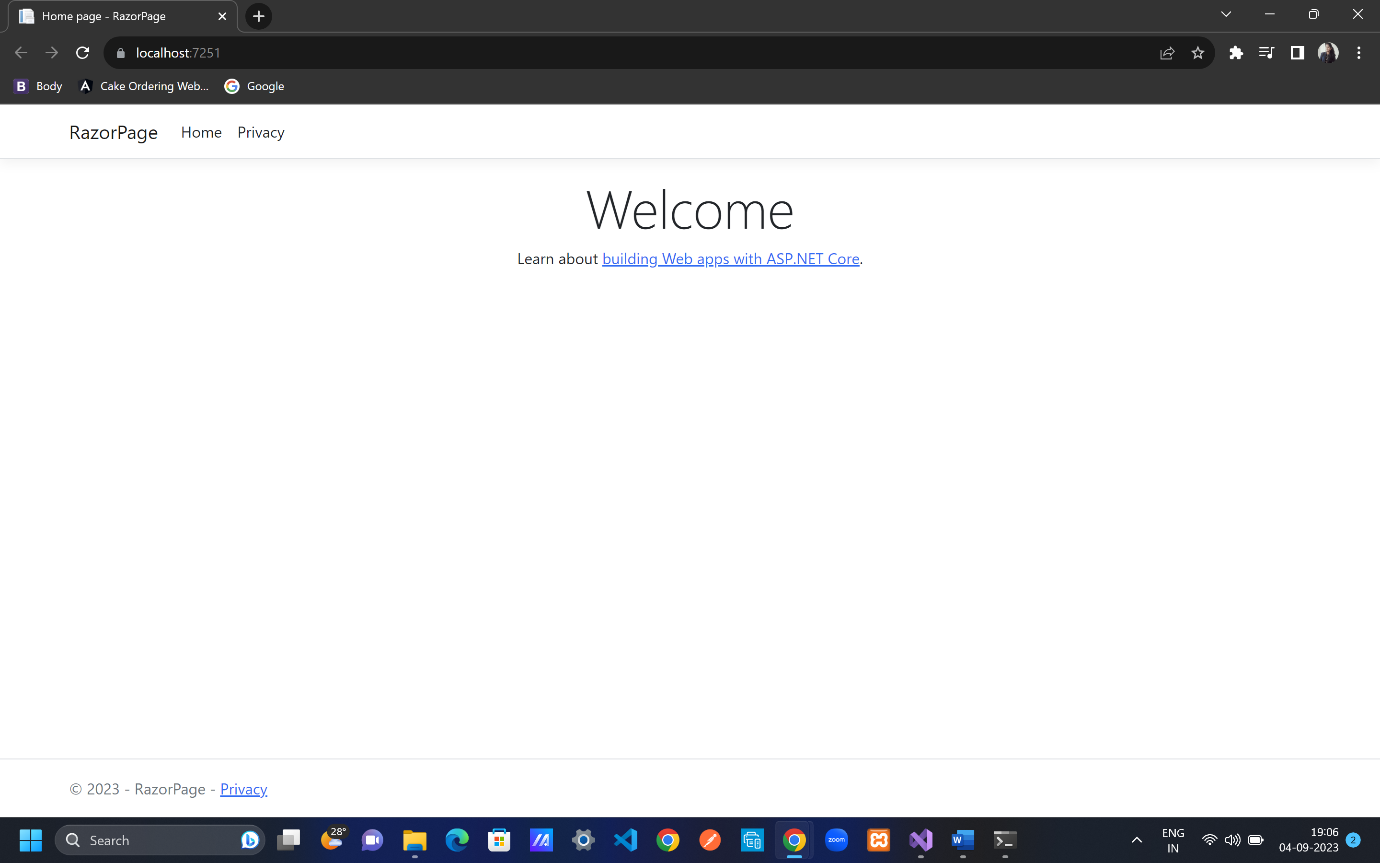
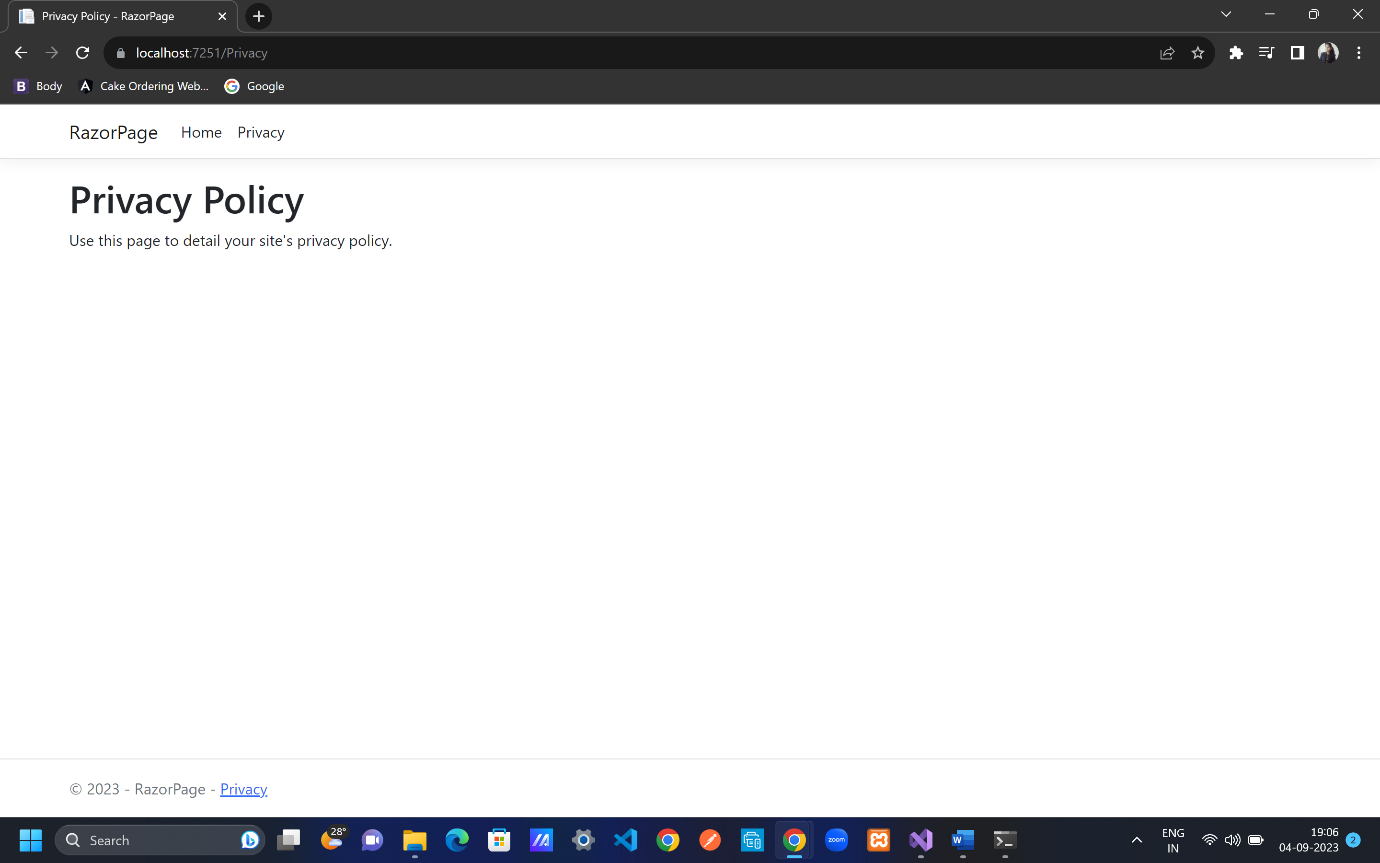
RAZOR PROJECT

CREATE RAZOR PROJECT

Open visual studio 2022; create a new project; search razor; select ASP.NET Core Web App; Next; Project name: RazorPage; Next; Create





When we use razor pages we do not have controller, model and view, we only have is pages.

What we have in razor pages projects?

Appsettings.json, Program.cs, no controller or action in razor page project that is why route is there - app.MapRazorPages();

Right click on application name; Add; New folder; rename as Models;

Right click on Models; Add; Class; rename class name as Category.cs

; Add

Category.cs

[Key]

public int Id { get; set; }

[Required]

[MaxLength(30)]

[DisplayName("Category Name")]

public string Name { get; set; }

[DisplayName("Display Order")]

[Range(1, 100, ErrorMessage ="Display Order must be between 1-100")]

public int DisplayOrder { get; set; }

Automatically comes after typing data members

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

Open ssms; to connect from visual studio go to view open sql serverobject explorer; click on add sql server; select local; see server name and copy it and pate it on ssms i.e (localdb)\MSSQLLocalDB; connect

CONNECTION STRING

appsettings.json

,

"ConnectionStrings": {

"DefaultConnection1": "Server=(localdb)\\MSSQLLocalDB; Database=Razor;Trusted\_Connection=True;TrustServerCertificate=True;"

}

Microsoft.EntityFrameworkCore.Design

NUGET PACKAGES FOR ENTITY FRAMEWORK CORE

To create database table, go file name and select manage nuget packages; search Microsoft.EntityFrameworkCore; install it; search Microsoft.EntityFrameworkCore.SqlServer; install it; make sure all the package shall be of same release since difference in release will create problem further; search Microsoft.EntityFrameworkCore.Tools; install it; right click on project file name and go to edit project file, we can see all installed packages with the released version.

SETUP ApplicationDbContext

right click on project file name and go to add; new folder name it: DATA;right click on data folder; add; class; name it ApplicationDbContext.cs

configuration

after typing DbContext hit ctrl + . to get the package name;ctor tab 2 times to get constructor;for base… hit tab

using Microsoft.EntityFrameworkCore;

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) {

}

}

Program.cs

Add services to the container

Add name of class from applicationDbContext.cs; give name => add connectionstring name from appsettings.json

builder.Services.AddDbContext<ApplicationDbContext>(options=>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection1")));

CREATE CATEGORY TABLE

ApplicationDbContext.cs

using RazorPage.Models;

This line will create table with category name; we can give name of table i.e Categories here

public DbSet<Category> Categories { get; set; }

SEED CATEGORY TABLE

ApplicationDbContext.cs

Adding data to database with asp.net facility

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Category>().HasData(

new Category { Id=1, Name="Action", DisplayOrder=1},

new Category { Id = 2, Name = "SciFi", DisplayOrder = 2 },

new Category { Id = 3, Name = "History", DisplayOrder = 3 }

);

}

Go to tools; nuget package manager; package manager console

> add-migration AddCategoryrazorPage

> update-database

Once done check in ssms; refresh from object explorer; go to database; go inside Razor; go inside Tables; right click on dbo.Categories and select Select top 1000 rows

Check all column name and column data is available.

Right click on Pages; Add; New Folder; rename file name as Categories; right click on Categories; Add; Razor page; razor empty Add; Index.cshtml; Add

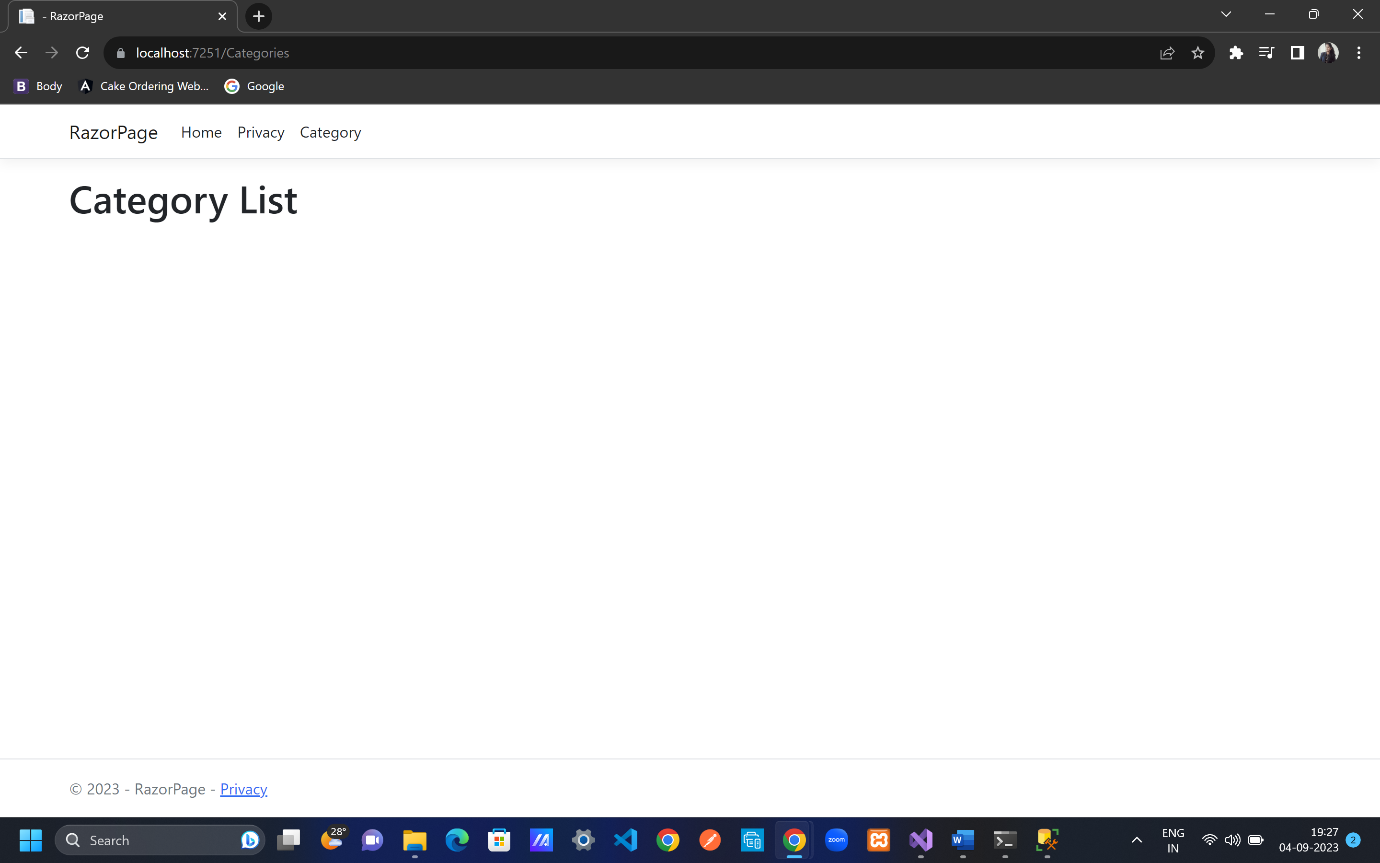
Always make sure the file name inside Models should not be same as of Pages files i.e Category != Categories

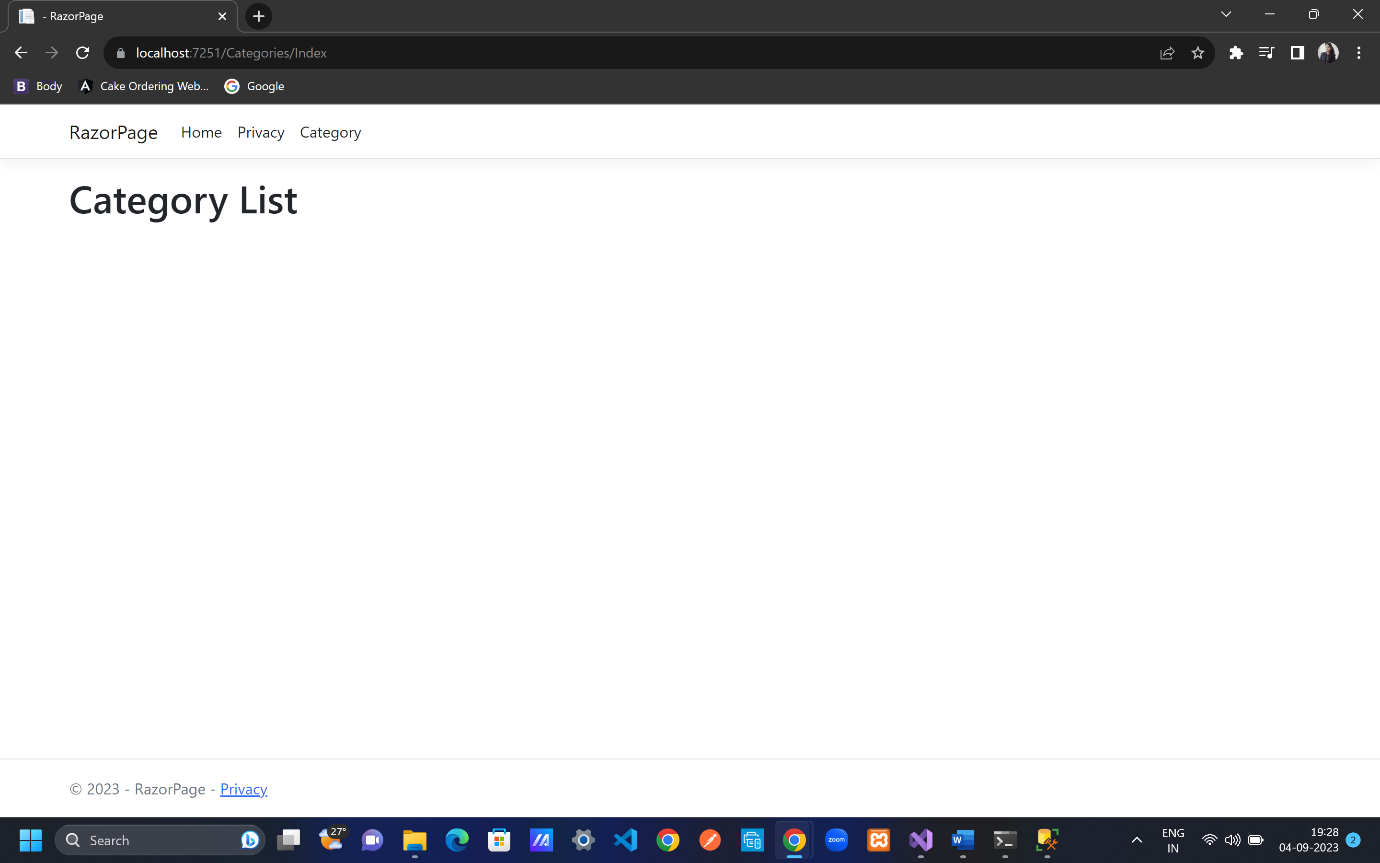
Categories folder; Index.cshtml

@page

@model RazorPage.Pages.Category.IndexModel

<h1>Category List</h1> //only this line is added





Razor pages does straight forward routing i.e simply by folder name cshtml files can be accessed, As above 2 images(url) represents the direct routing.

DISPLAY ALL CATEGORIES

Pages folder; Catgories folder

Index.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using RazorPage.Data;

using RazorPage.Models;

namespace RazorPage.Pages.Categories

{

public class IndexModel : PageModel

{

private readonly ApplicationDbContext \_db;

public List<Category> CategoryList { get; set; }

public IndexModel(ApplicationDbContext db)

{

\_db = db;

}

public void OnGet()

{

CategoryList = \_db.Categories.ToList();

}

}

}

Pages folder; Catgories folder

Index.cshtml

@page

@model RazorPage.Pages.Categories.IndexModel

<div class="container">

<div class="row pt-4 pb-3">

<div class="col-6">

<h2 class="text-primary">

Category List

</h2>

</div>

<div class="col-6 text-end">

<**a** **asp-page**="/categories/create" class="btn btn-primary">

<i class="bi bi-plus-circle"></i> Create New Category

</**a**>

</div>

</div>

<table class=" table table-bordered table-striped">

<thred>

<tr>

<th>

Category Name

</th>

<th>

Display Order

</th>

<th></th>

</tr>

</thred>

<tbody>

@foreach (var obj in Model.CategoryList.OrderBy(u => u.DisplayOrder))

{

<tr>

<td>

@obj.Name

</td>

<td>

@obj.DisplayOrder

</td>

<td>

<div class="w-75 btn-group" role="group">

<**a** **asp-page**="/categories/edit" **asp-route-id**="@obj.Id" class="btn btn-primary mx-2">

<i class="bi bi-pencil-square"></i> Edit

</**a**>

<**a** **asp-page**="/categories/delete" **asp-route-id**="@obj.Id" class="btn btn-danger mx-2">

<i class="bi bi-trash-fill"></i> Delete

</**a**>

</div>

</td>

</tr>

}

</tbody>

</table>

</div>

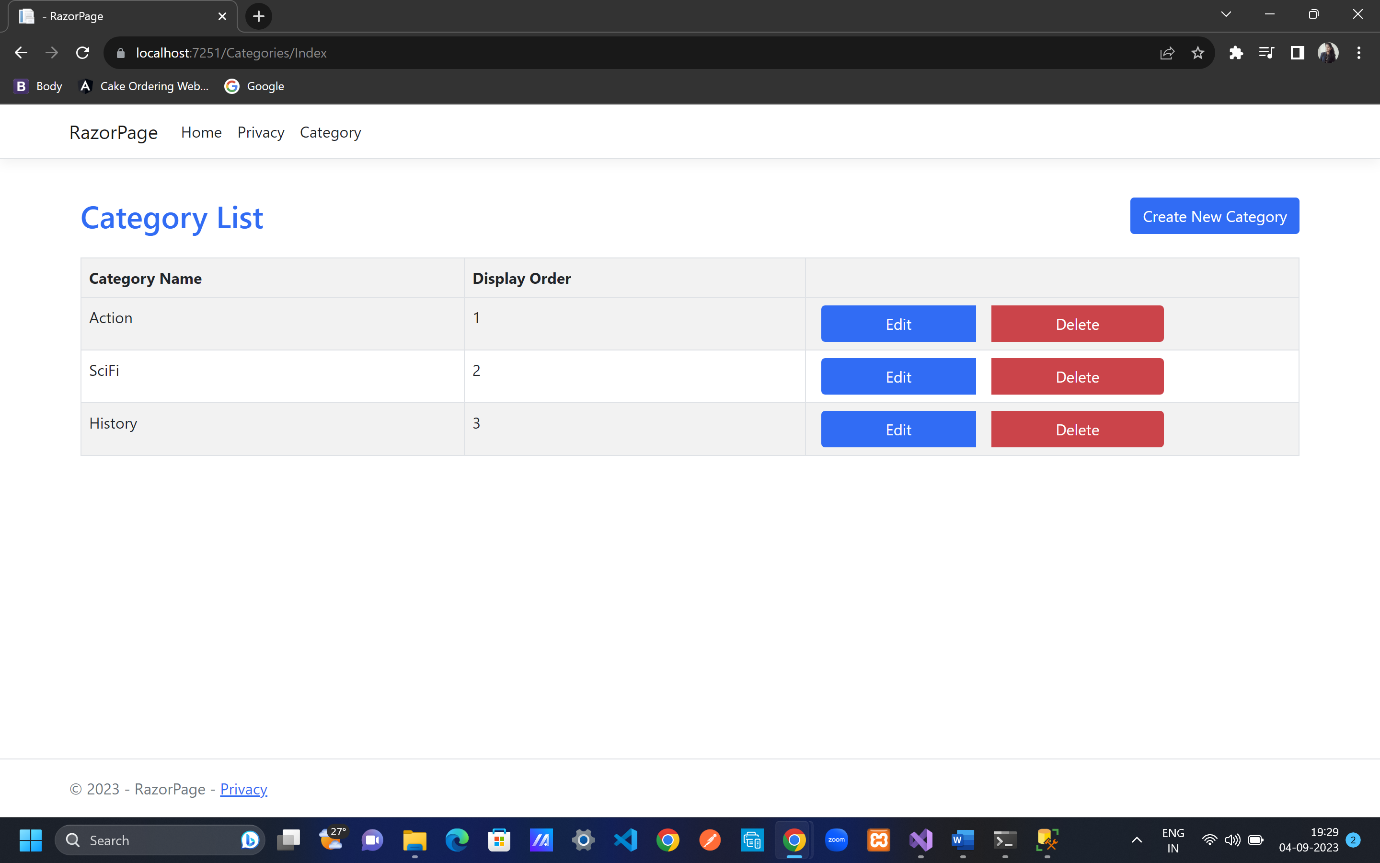
Pages folder; Shared folder

\_Layout.cshtml

<li class="nav-item">

<**a** class="nav-link text-dark" **asp-area**="" **asp-page**="/categories/index">Category</**a**>

</li>



CREATE CATEGORY UI

Pages folder; right click Categories folder; Add; empty Razor page; Add; rename it as Create.cshtml; Add

Create.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using RazorPage.Data;

using RazorPage.Models;

namespace RazorPage.Pages.Categories

{

public class CreateModel : PageModel

{

private readonly ApplicationDbContext \_db;

public Category Category { get; set; }

public CreateModel(ApplicationDbContext db)

{

\_db = db;

}

public void OnGet()

{

}

}

}

Create.cshtml

@page

@model RazorPage.Pages.Categories.CreateModel

﻿

<**form** method="post">

<div class="border p-3 mt-4">

<div class="row pb-2">

<h2 class="text-primary">Create Category</h2>

<hr/>

</div>

@\*<div asp-validation-summary="None"></div>\*@

<div class="mb-3 row p-1">

<**label** **asp-for**="Category.Name" class="p-0"></**label**>

<**input** **asp-for**="Category.Name" class="form-control" />

<**span** **asp-validation-for**="Category.Name" class="text-danger"></**span**>

</div>

<div class="mb-3 row p-1">

<**label** **asp-for**="Category.DisplayOrder" class="p-0"></**label**>

<**input** **asp-for**="Category.DisplayOrder" class="form-control" />

<**span** **asp-validation-for**="Category.DisplayOrder" class="text-danger"></**span**>

</div>

<div class="row">

<div class="col-6 col-md-3">

<button type="submit" class="btn btn-primary form-control" >Create</button>

</div>

<div class="col-6 col-md-3">

<**a** **asp-page**="/categories/index" class="btn btn-outline-secondary border form-control" > Back to List</**a**>

</div>

</div>

</div>

</**form**>

@section Scripts{

@{

<**partial** **name**="\_ValidationScriptsPartial"/>

}

}

CREATE CATEGORY POST HANDLER

If bindproperty is used

Pages folder; Categories folder

Create.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using RazorPage.Data;

using RazorPage.Models;

namespace RazorPage.Pages.Categories

{

public class CreateModel : PageModel

{

private readonly ApplicationDbContext \_db;

[BindProperty] //bind all labels data after onpost() is called basically to save data or update data

public Category Category { get; set; }

public CreateModel(ApplicationDbContext db)

{

\_db = db;

}

public void OnGet()

{

}

public IActionResult OnPost()

{

\_db.Categories.Add(Category);

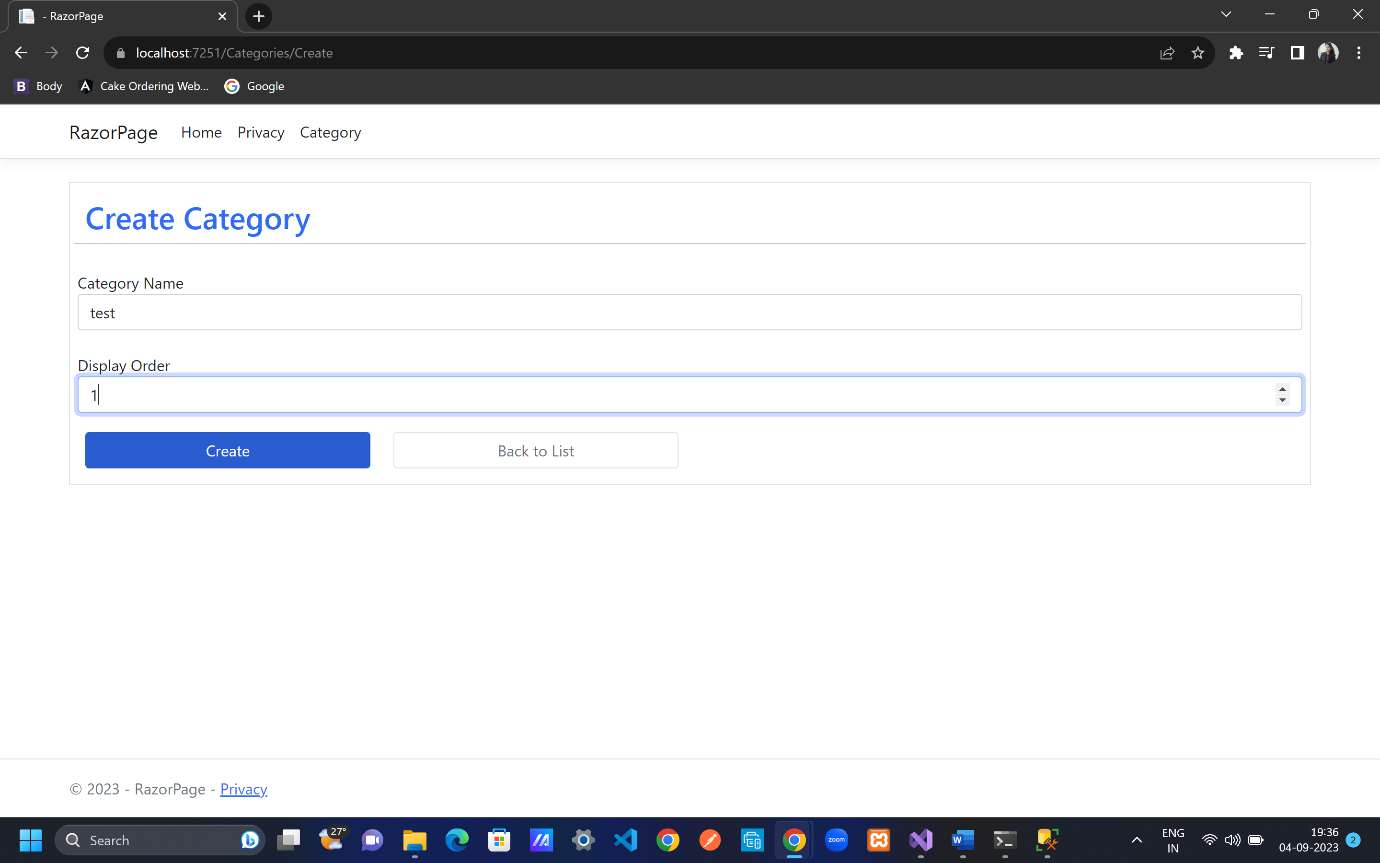
\_db.SaveChanges();

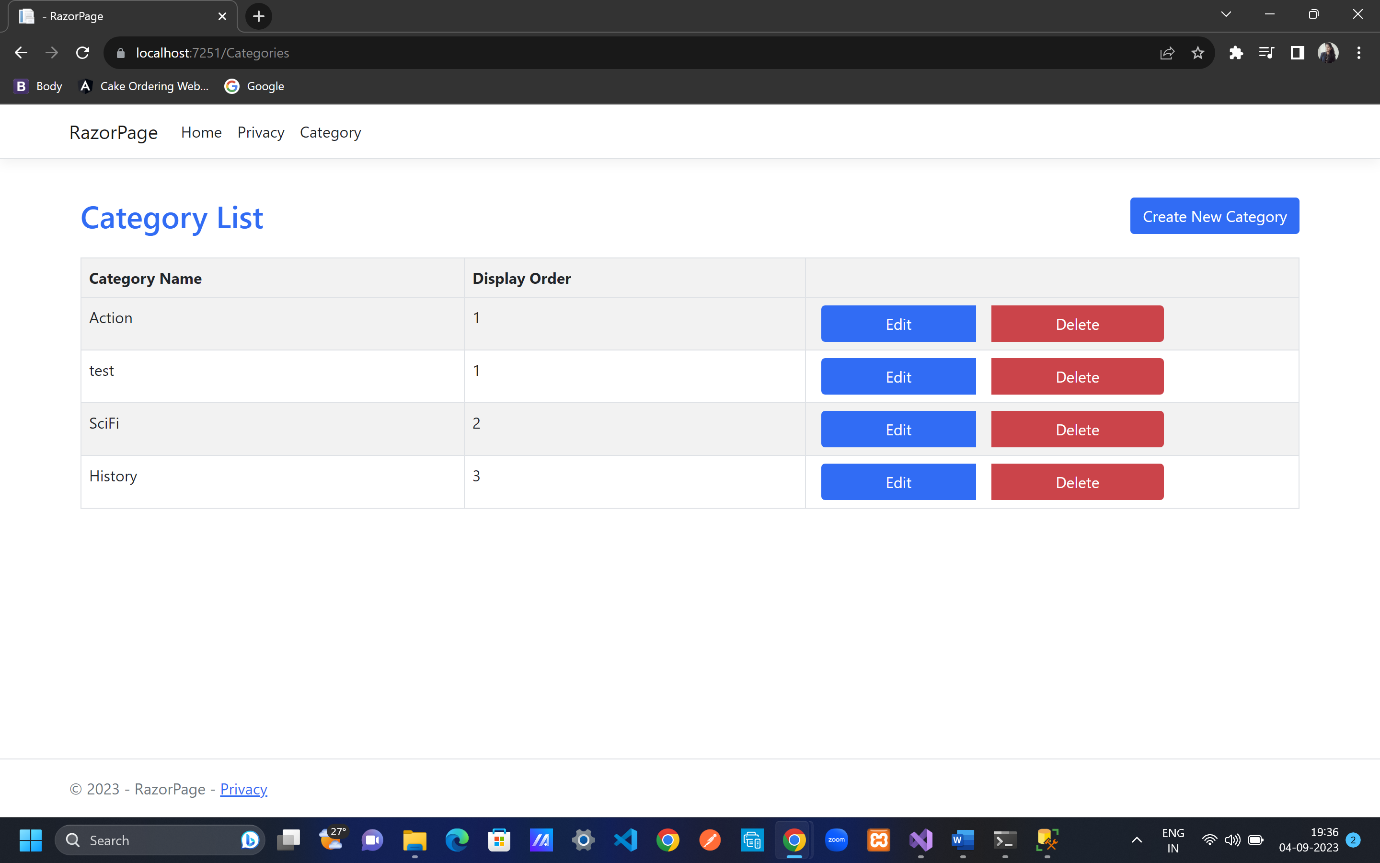
return RedirectToPage("Index");

}

}

}





If bindproperties is used

Pages folder; Categories folder

Create.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using Razor.Data;

using Razor.Models;

namespace Razor.Pages.Categories

{

[BindProperties] //bind all properties of page model

public class CreateModel : PageModel

{

private readonly ApplicationDbContext \_db;

//[BindProperty] //bind all labels data after onpost() is called basically to save data or update data

public Category Category { get; set; }

public CreateModel(ApplicationDbContext db)

{

\_db = db;

}

public void OnGet()

{

}

public IActionResult OnPost()

{

\_db.Categories.Add(Category);

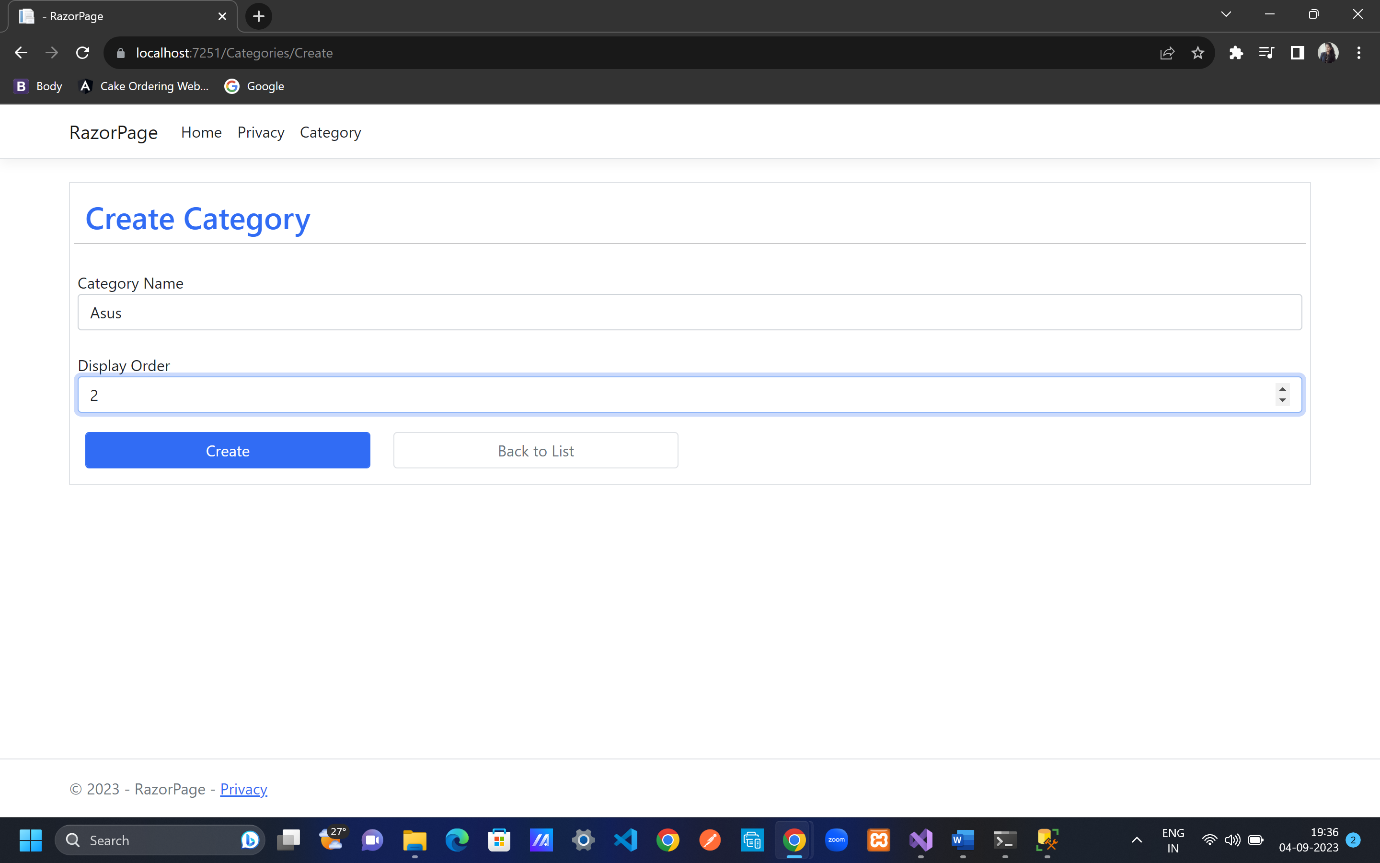
\_db.SaveChanges();

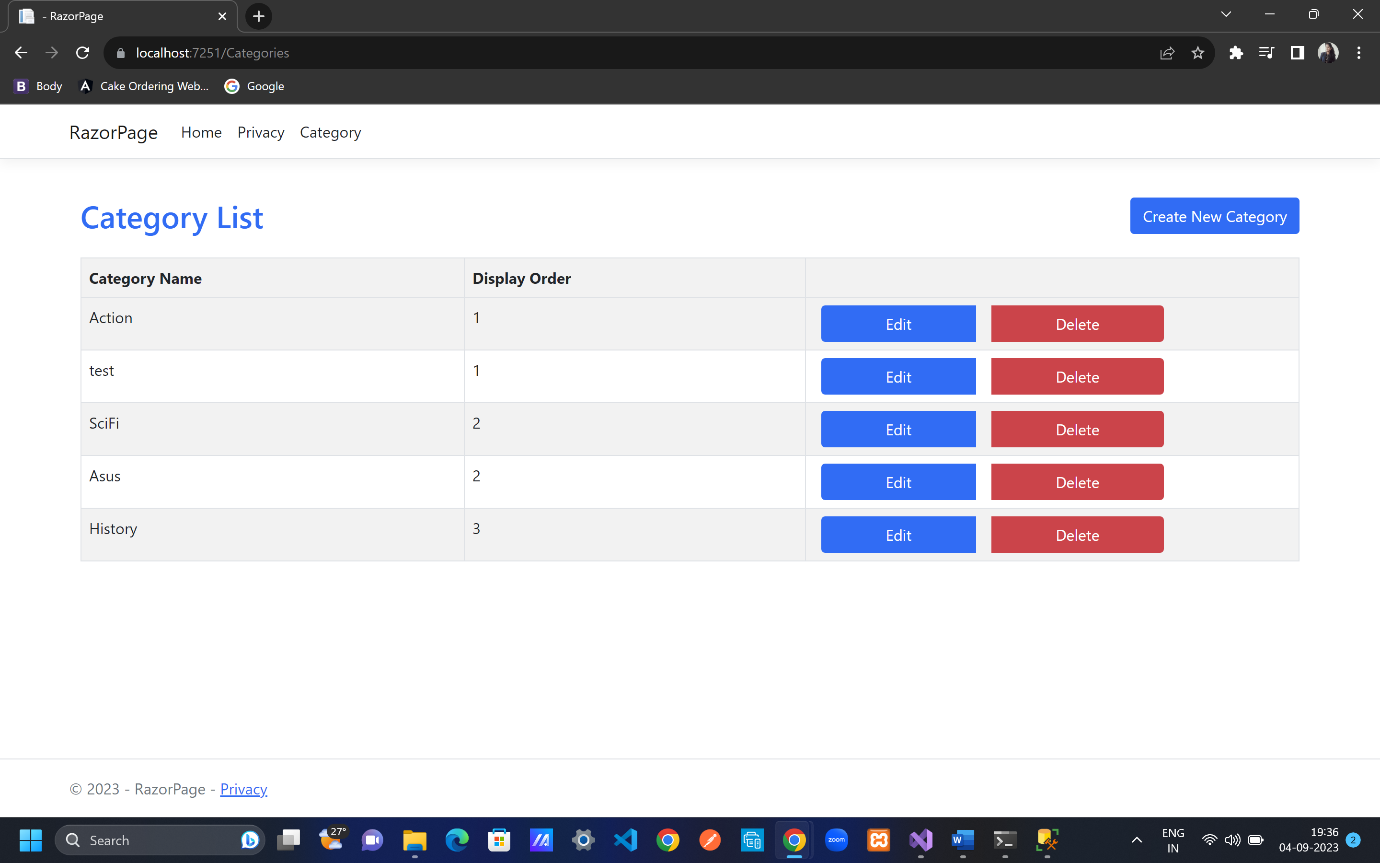
return RedirectToPage("Index");

}

}

}





But the resultant of both bindproperty and bindproperties is same. As shown above in all 4 images.

EDIT AND DELETE CATEGORY

Pages folder; right click Categories folder; Add; empty Razor page; Add; rename it as Edit.cshtml; Add

Pages folder; right click Categories folder; Add; empty Razor page; Add; rename it as Delete.cshtml; Add

Pages folder; Categories folder; Edit.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using RazorPage.Data;

using RazorPage.Models;

namespace RazorPage.Pages.Categories

{

[BindProperties]

public class EditModel : PageModel

{

private readonly ApplicationDbContext \_db;

//[BindProperty] //bind all labels data after onpost() is called basically to save data or update data

public Category Category { get; set; }

public EditModel(ApplicationDbContext db)

{

\_db = db;

}

public void OnGet(int? id)

{

if(id!=null && id!=0)

{

Category = \_db.Categories.Find(id);

}

}

public IActionResult OnPost()

{

if (ModelState.IsValid)

{

\_db.Categories.Update(Category); // what is to be add

\_db.SaveChanges(); //go to the database and create category

//TempData["success"] = "Category updated successfully";

return RedirectToPage("Index"); //view all categories in index() line 9;

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

//-> if we have different controller then add like

//this else keep only menthod/function name

}

return Page();

}

}

}

Pages folder; Categories folder; Edit.cshtml

@page

@model RazorPage.Pages.Categories.EditModel

<**form** method="post">

<**input** **asp-for**="Category.Id" hidden />

<div class="border p-3 mt-4">

<div class="row pb-2">

<h2 class="text-primary">Edit Category</h2>

<hr />

</div>

@\*<div asp-validation-summary="None"></div>\*@

<div class="mb-3 row p-1">

<**label** **asp-for**="Category.Name" class="p-0"></**label**>

<**input** **asp-for**="Category.Name" class="form-control" />

<**span** **asp-validation-for**="Category.Name" class="text-danger"></**span**>

</div>

<div class="mb-3 row p-1">

<**label** **asp-for**="Category.DisplayOrder" class="p-0"></**label**>

<**input** **asp-for**="Category.DisplayOrder" class="form-control" />

<**span** **asp-validation-for**="Category.DisplayOrder" class="text-danger"></**span**>

</div>

<div class="row">

<div class="col-6 col-md-3">

<button type="submit" class="btn btn-primary form-control">Update</button>

</div>

<div class="col-6 col-md-3">

<**a** **asp-page**="/categories/index" class="btn btn-outline-secondary border form-control"> Back to List</**a**>

</div>

</div>

</div>

</**form**>

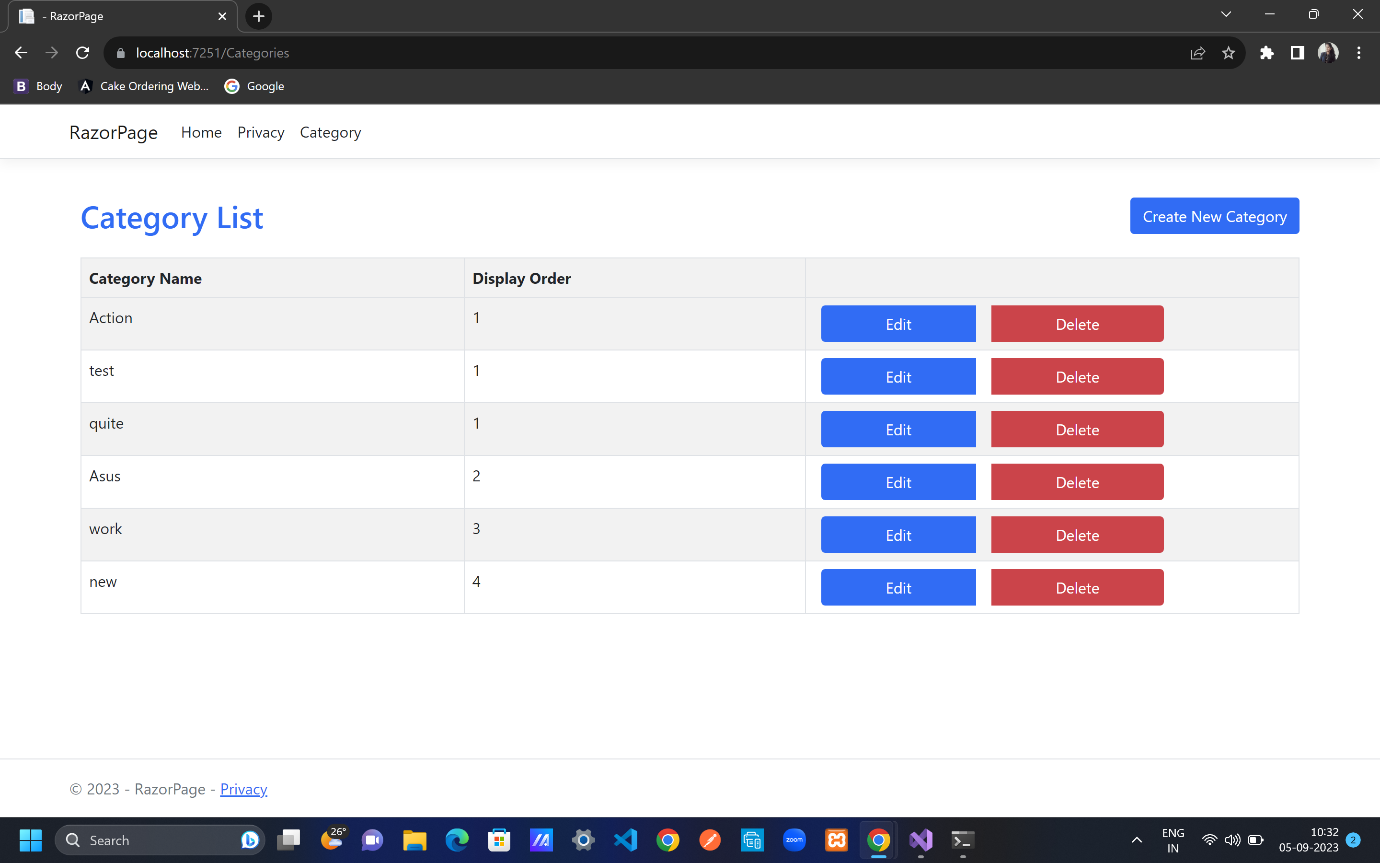
@section Scripts{

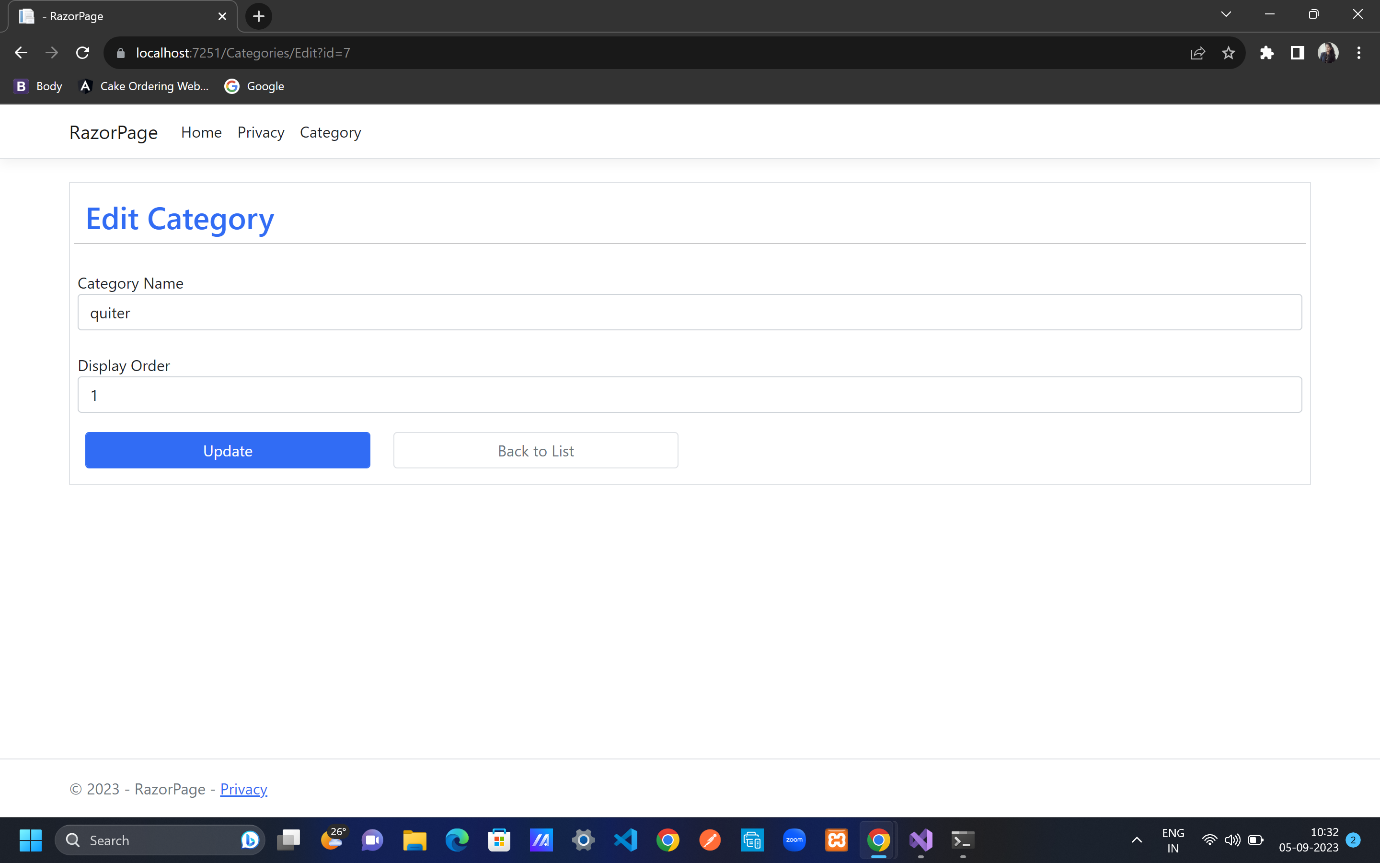
@{

<**partial** **name**="\_ValidationScriptsPartial" />

}

}





Pages folder; Categories folder; Delete.cshtml.cs

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.RazorPages;

using RazorPage.Data;

using RazorPage.Models;

namespace RazorPage.Pages.Categories

{

[BindProperties]

public class DeleteModel : PageModel

{

private readonly ApplicationDbContext \_db;

//[BindProperty] //bind all labels data after onpost() is called basically to save data or update data

public Category Category { get; set; }

public DeleteModel(ApplicationDbContext db)

{

\_db = db;

}

public void OnGet(int? id)

{

if (id != null && id != 0)

{

Category = \_db.Categories.Find(id);

}

}

public IActionResult OnPost()

{

Category? obj = \_db.Categories.Find(Category.Id); //we will get id from Category because of "BindProperties"

if (obj == null)

{

return NotFound();

}

\_db.Categories.Remove(obj);

\_db.SaveChanges(); //go to the database and create category

return RedirectToPage("Index"); //view all categories in index() line 9;

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

//-> if we have different controller then add like

//this else keep only menthod/function name

}

}

}

Pages folder; Categories folder; Delete.cshtml

@page

@model RazorPage.Pages.Categories.DeleteModel

<**form** method="post">

<**input** **asp-for**="Category.Id" hidden />

<div class="border p-3 mt-4">

<div class="row pb-2">

<h2 class="text-primary">Delete Category</h2>

<hr />

</div>

@\*<div asp-validation-summary="None"></div>\*@

<div class="mb-3 row p-1">

<**label** **asp-for**="Category.Name" class="p-0"></**label**>

<**input** **asp-for**="Category.Name" class="form-control" />

<**span** **asp-validation-for**="Category.Name" class="text-danger"></**span**>

</div>

<div class="mb-3 row p-1">

<**label** **asp-for**="Category.DisplayOrder" class="p-0"></**label**>

<**input** **asp-for**="Category.DisplayOrder" class="form-control" />

<**span** **asp-validation-for**="Category.DisplayOrder" class="text-danger"></**span**>

</div>

<div class="row">

<div class="col-6 col-md-3">

<button type="submit" class="btn btn-danger form-control">Delete</button>

</div>

<div class="col-6 col-md-3">

<**a** **asp-page**="/categories/index" class="btn btn-outline-secondary border form-control"> Back to List</**a**>

</div>

</div>

</div>

</**form**>

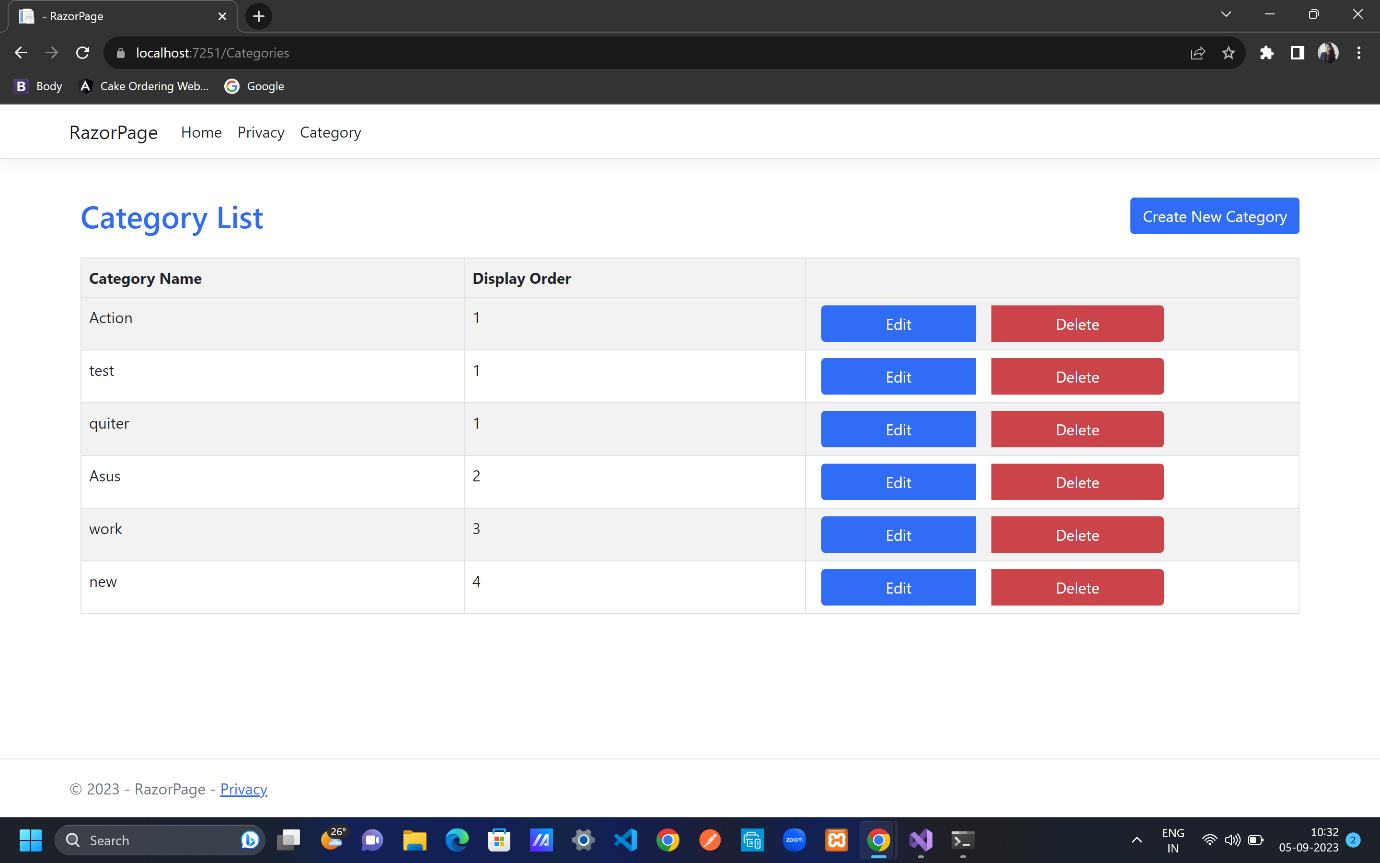
@section Scripts{

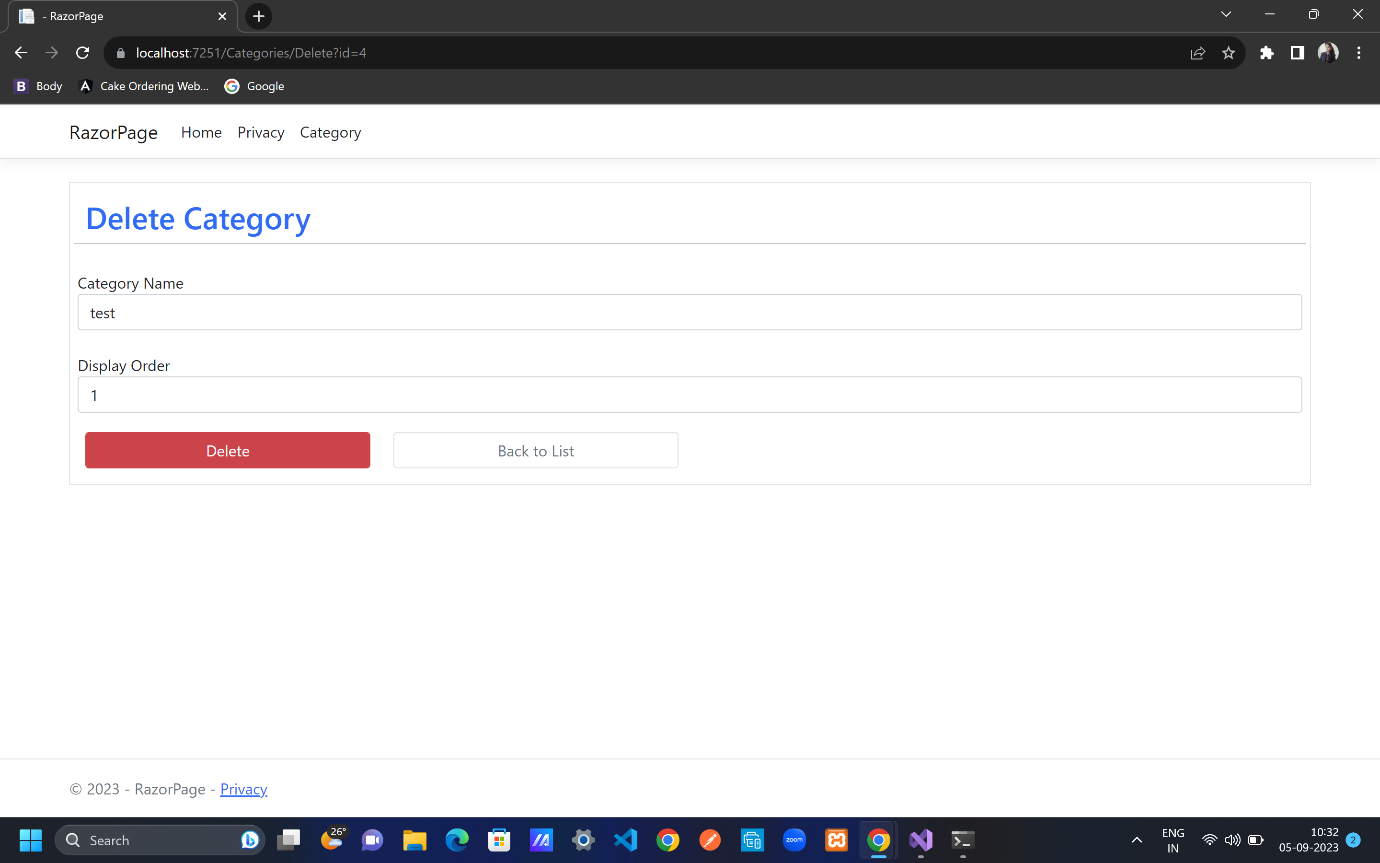
@{

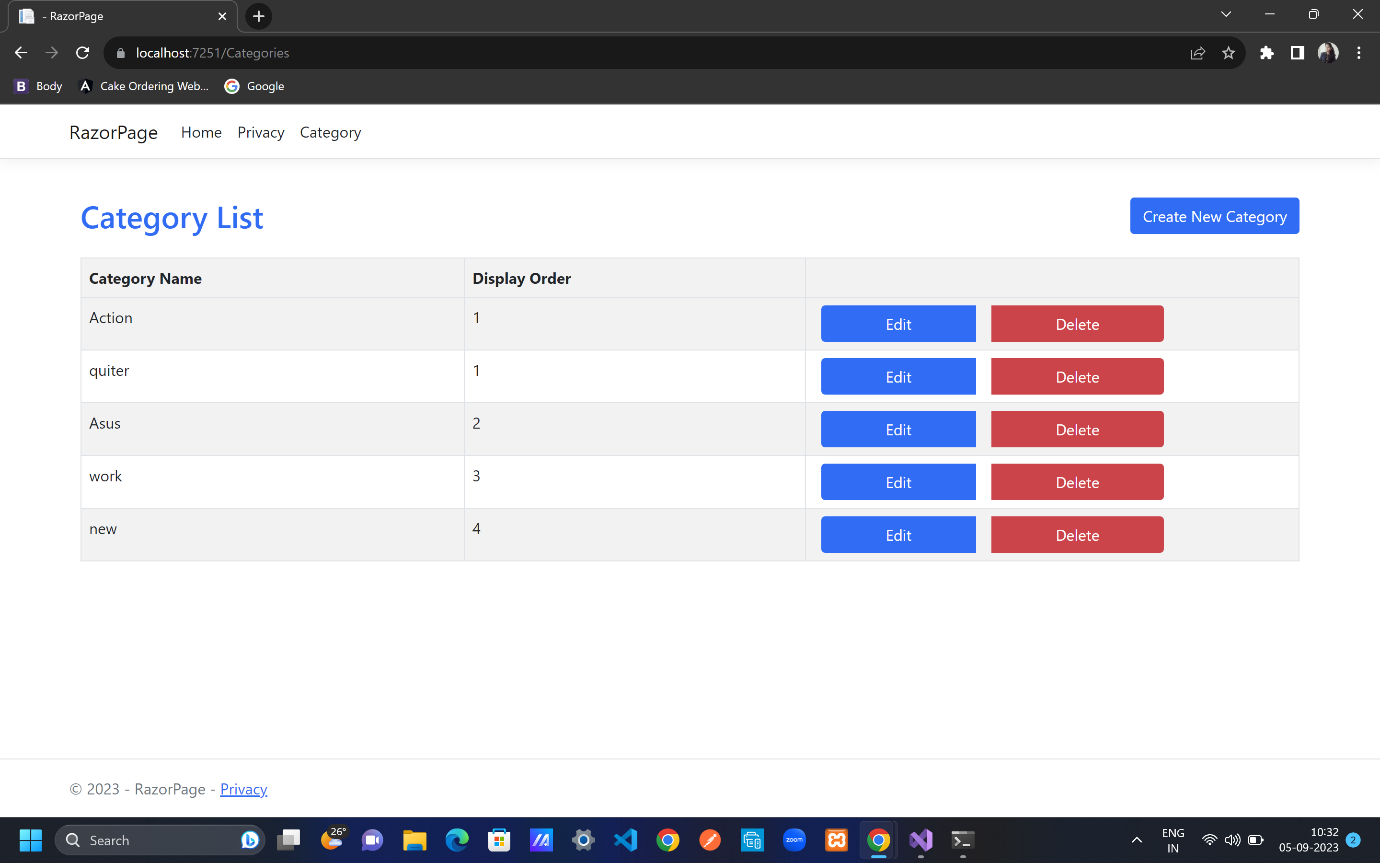
<**partial** **name**="\_ValidationScriptsPartial" />

}

}







Toastr NOTIFICATIONS AND PARTIAL VIEWS

To show notification pop-ups

Right click on Shared folder; Add; Razor Page; Add empty razorpage; \_Notification.cshtml; delete \_Notification.cshtml.cs by right click and click on delete

Shared folder; \_Notification.cshtml

﻿@if (TempData["success"] != null)

{

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

<script src="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>

<script type="text/javascript">

toastr.success('@TempData["success"]');

</script>

}

@if (TempData["error"] != null)

{

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

<script src="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/js/toastr.min.js"></script>

<script type="text/javascript">

toastr.error('@TempData["error"]');

</script>

}

Shared folder; \_Layoutcshtml

<head>

<meta charset="utf-8" />

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

<title>@ViewData["Title"] - RazorPage</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**="~/RazorPage.styles.css" **asp-append-version**="true" />

<link rel="stylesheet" href="//cdnjs.cloudflare.com/ajax/libs/toastr.js/latest/css/toastr.min.css" />

</head>

</header>

<div class="container">

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

<**partial** **name**="\_Notification"/>

@RenderBody()

</main>

</div>

Pages folder; Categories folder; Delete.cshtml.cs

public IActionResult OnPost()

{

Category? obj = \_db.Categories.Find(Category.Id); //we will get id from Category because of "BindProperties"

if (obj == null)

{

return NotFound();

}

\_db.Categories.Remove(obj);

\_db.SaveChanges(); //go to the database and create category

TempData["success"] = "Category deleted successfully"; //toastr notification

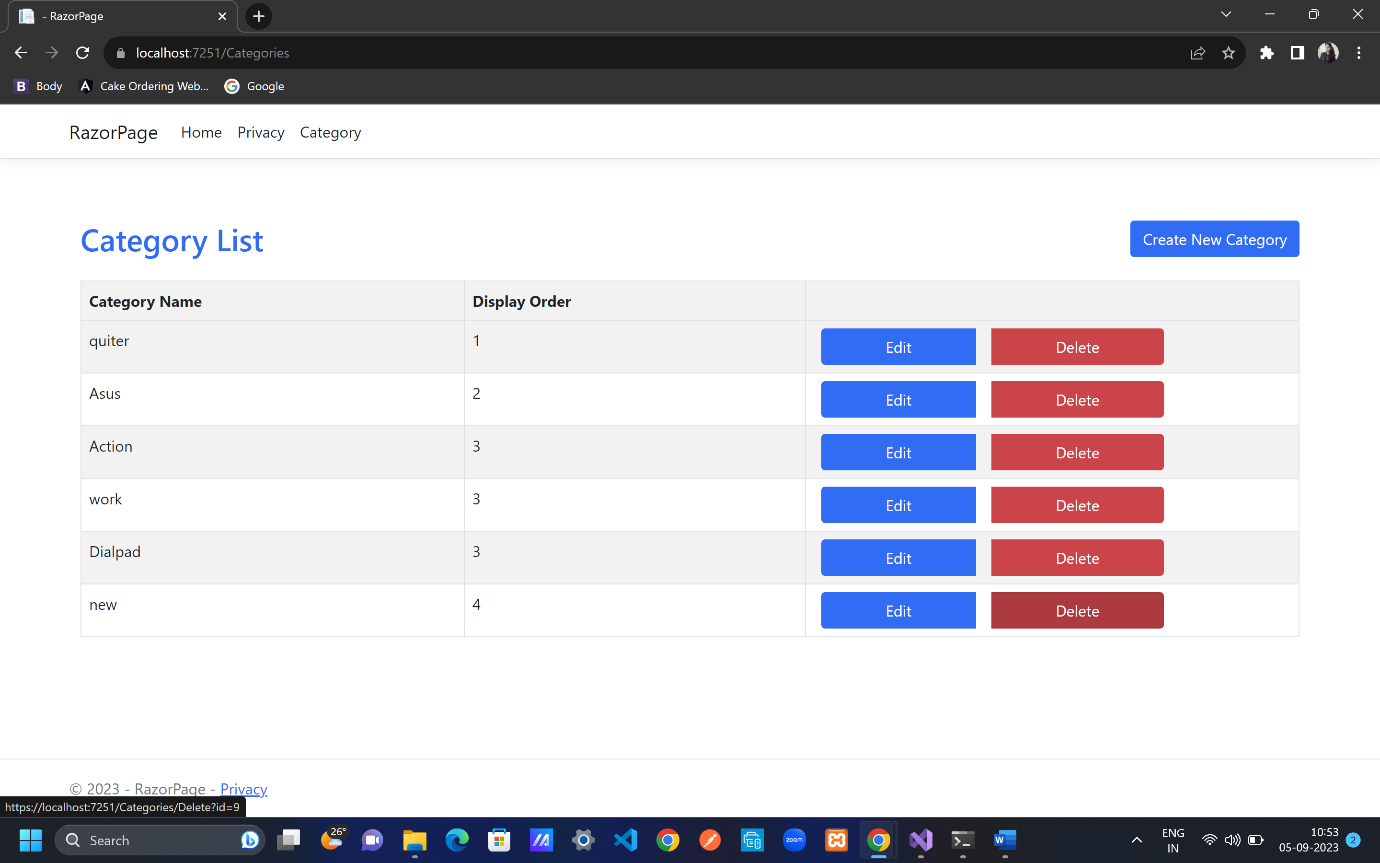
return RedirectToPage("Index"); //view all categories in index() line 9;

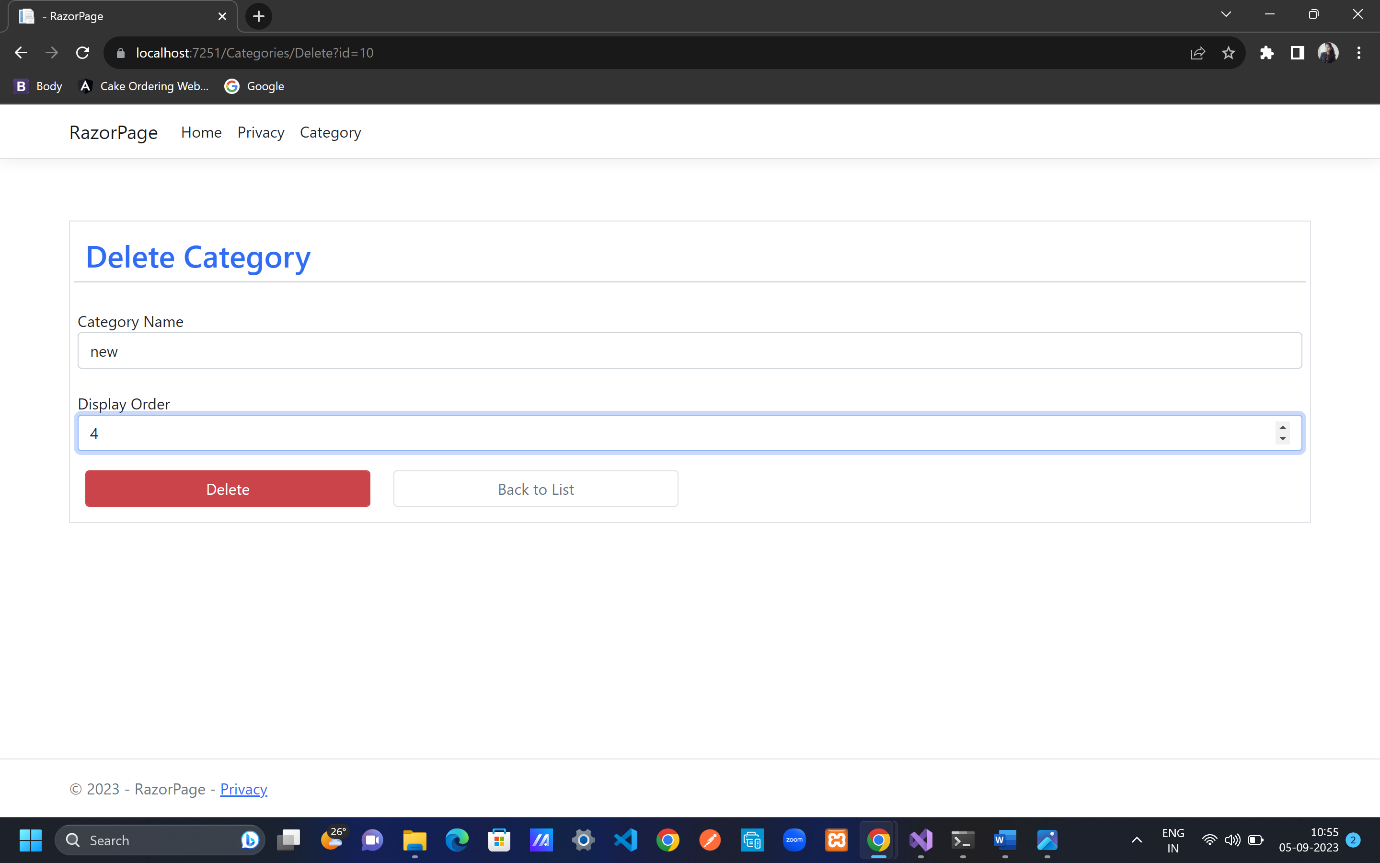
//return RedirectToAction("Index","Category");

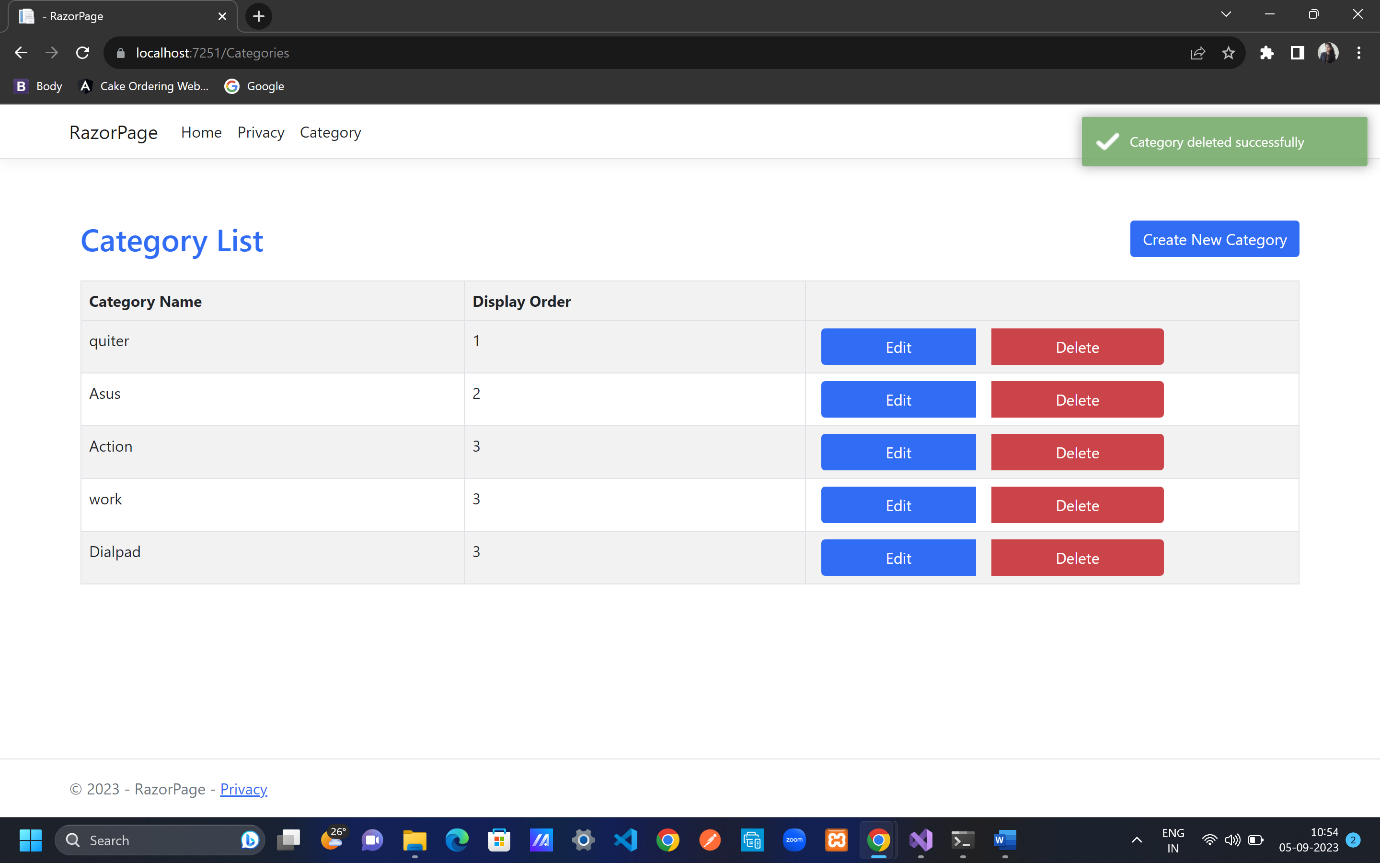
//-> if we have different controller then add like

//this else keep only menthod/function name

}







Pages folder; Categories folder; Edit.cshtml.cs

public IActionResult OnPost()

{

if (ModelState.IsValid)

{

\_db.Categories.Update(Category); // what is to be add

\_db.SaveChanges(); //go to the database and create category

TempData["success"] = "Category updated successfully"; //toastr notification

return RedirectToPage("Index"); //view all categories in index() line 9;

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

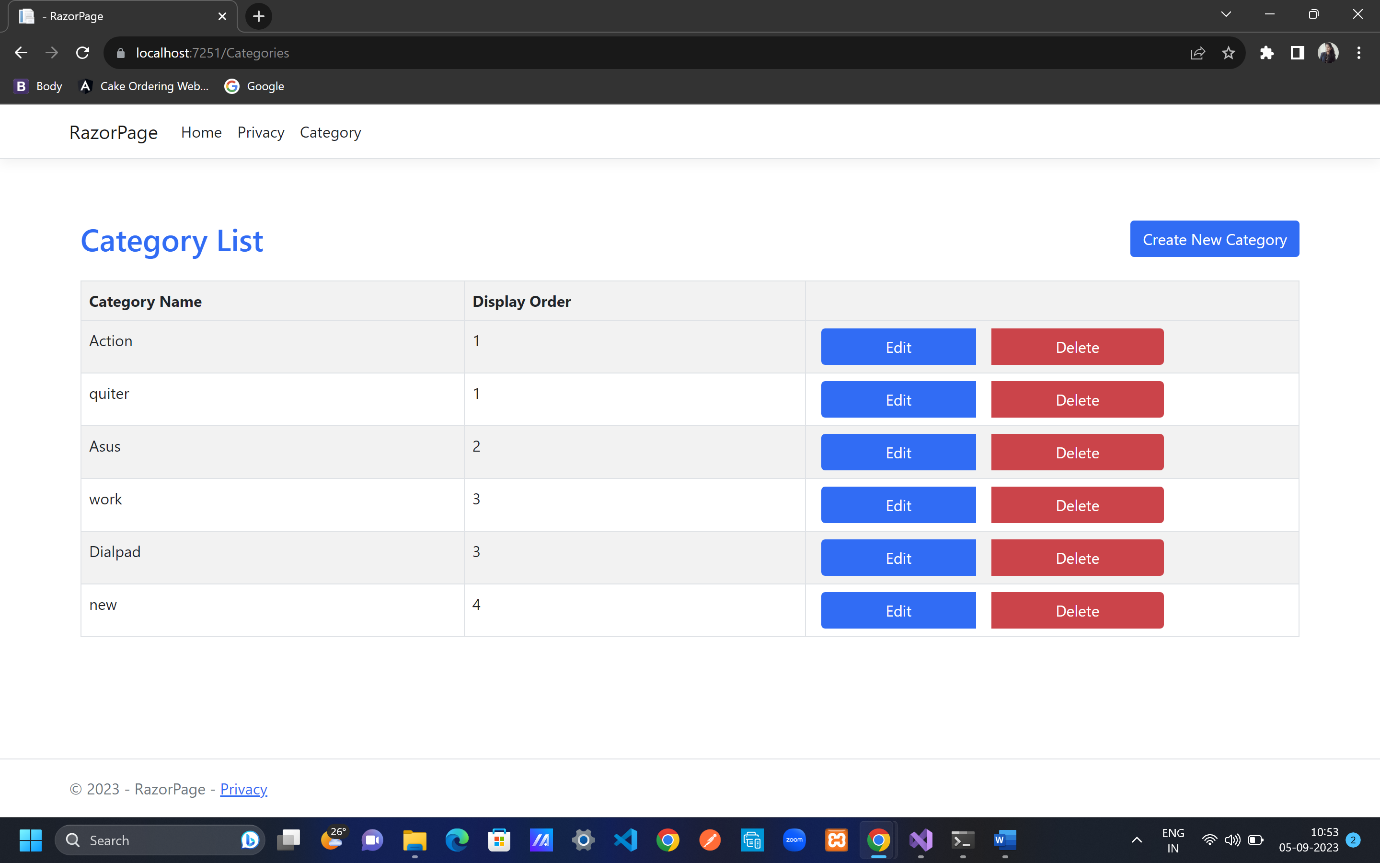
//-> if we have different controller then add like

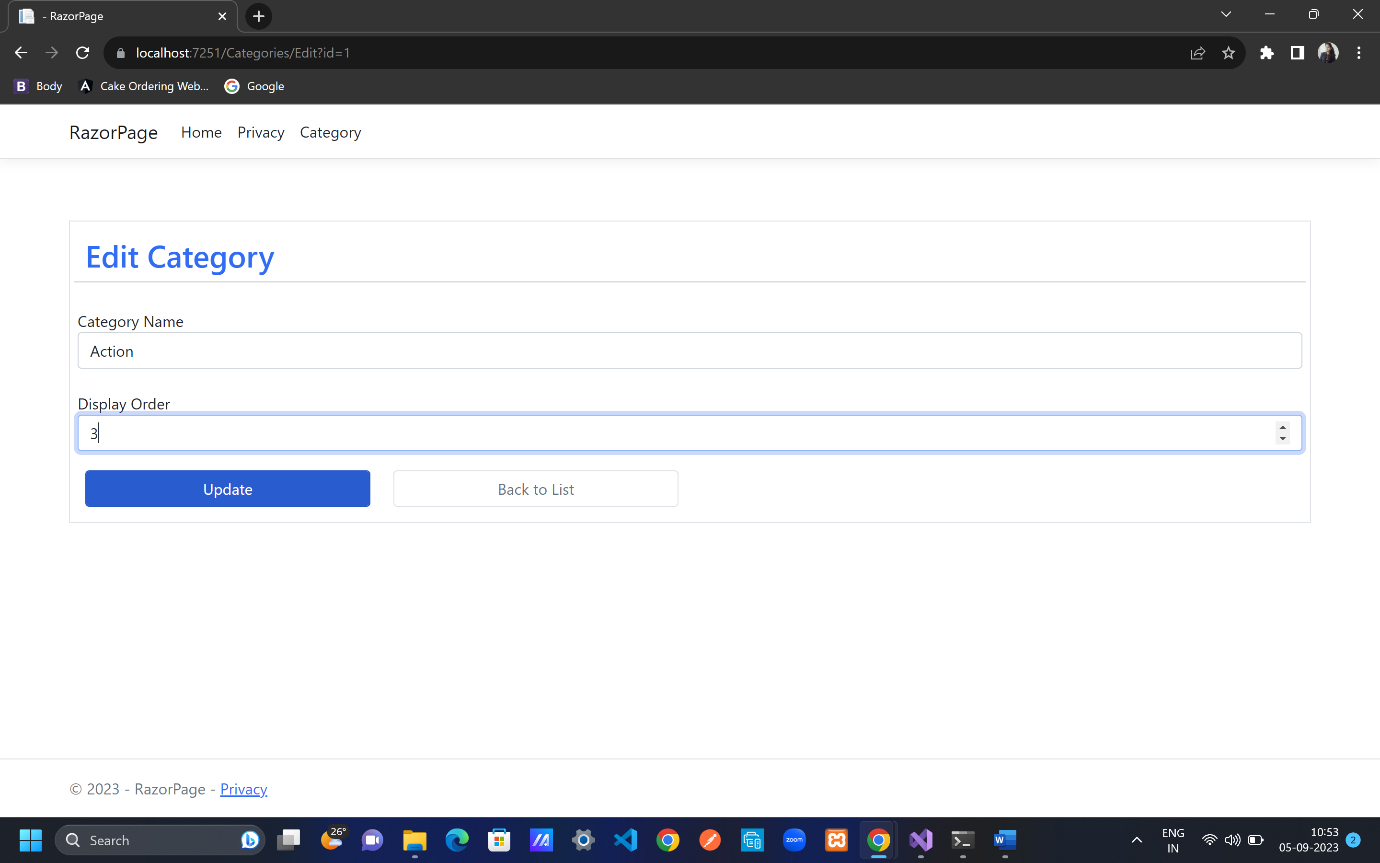
//this else keep only menthod/function name

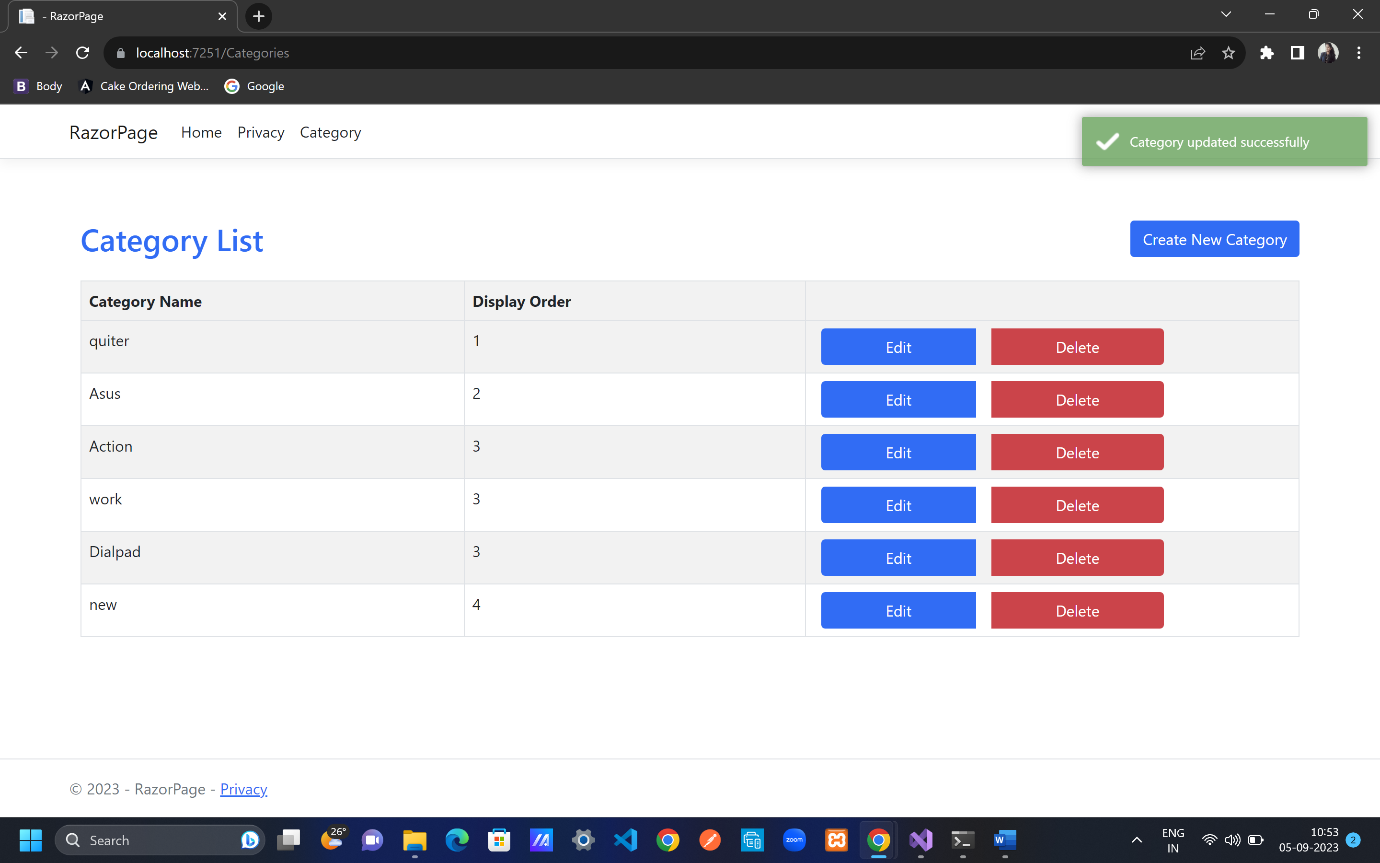
}

return Page();

}







Pages folder; Categories folder; Create.cshtml.cs

public IActionResult OnPost()

{

\_db.Categories.Add(Category);

\_db.SaveChanges();

TempData["success"] = "Category created successfully"; //toastr notification

return RedirectToPage("Index");

}

