**Model (Region.cs)**

Create the Region model class in the Models folder:

csharp

Copy code

namespace YourNamespace.Models

{

public class Region

{

public string Code { get; set; }

public string Name { get; set; }

public string ShortDescription { get; set; }

public string Description { get; set; }

public string Status { get; set; }

}

}

**Service (RegionService.cs)**

Create a service to manage the regions in-memory in the Services folder:

csharp

Copy code

using System.Collections.Generic;

using YourNamespace.Models;

namespace YourNamespace.Services

{

public class RegionService

{

private readonly List<Region> \_regions = new List<Region>

{

new Region { Code = "AM", Name = "Americas", ShortDescription = "Americas", Description = "Americas", Status = "Active" },

new Region { Code = "APAC", Name = "APAC", ShortDescription = "Asia Pacific", Description = "Asia Pacific", Status = "Active" },

new Region { Code = "CH", Name = "Switzerland", ShortDescription = "CH", Description = "Switzerland and Luxembourg", Status = "Active" },

new Region { Code = "EMEA", Name = "EMEA", ShortDescription = "EMEA", Description = "Excluding UK & Switzerland", Status = "Active" },

new Region { Code = "UK", Name = "UK", ShortDescription = "United Kingdom", Description = "United Kingdom", Status = "Active" }

};

public List<Region> GetAllRegions() => \_regions;

public void AddRegion(Region region)

{

\_regions.Add(region);

}

public void UpdateRegion(Region region)

{

var existingRegion = \_regions.Find(r => r.Code == region.Code);

if (existingRegion != null)

{

existingRegion.Name = region.Name;

existingRegion.ShortDescription = region.ShortDescription;

existingRegion.Description = region.Description;

existingRegion.Status = region.Status;

}

}

public void DeleteRegion(string code)

{

var region = \_regions.Find(r => r.Code == code);

if (region != null)

{

\_regions.Remove(region);

}

}

}

}

**Controller (RegionController.cs)**

Create a controller in the Controllers folder to manage the views and actions:

csharp

Copy code

using Microsoft.AspNetCore.Mvc;

using YourNamespace.Models;

using YourNamespace.Services;

namespace YourNamespace.Controllers

{

public class RegionController : Controller

{

private readonly RegionService \_regionService;

public RegionController(RegionService regionService)

{

\_regionService = regionService;

}

public IActionResult Index()

{

var regions = \_regionService.GetAllRegions();

return View(regions);

}

[HttpPost]

public IActionResult Save(Region region)

{

if (string.IsNullOrEmpty(region.Code))

{

\_regionService.AddRegion(region);

}

else

{

\_regionService.UpdateRegion(region);

}

return RedirectToAction("Index");

}

[HttpPost]

public IActionResult Delete(string code)

{

\_regionService.DeleteRegion(code);

return RedirectToAction("Index");

}

}

}

**View (Index.razor)**

Now create the view using MudBlazor in the Views/Region folder:

razor

Copy code

@page "/regions"

@inject RegionService RegionService

<MudText Typo="Typo.h3">Tax Risk Reporting Tool</MudText>

<MudTable Items="regions" Striped="true">

<HeaderContent>

<MudTh>Region Code</MudTh>

<MudTh>Region Name</MudTh>

<MudTh>Region Short Description</MudTh>

<MudTh>Region Description</MudTh>

<MudTh>Status</MudTh>

<MudTh>Actions</MudTh>

</HeaderContent>

<RowTemplate>

<MudTd DataLabel="Region Code">@context.Code</MudTd>

<MudTd DataLabel="Region Name">@context.Name</MudTd>

<MudTd DataLabel="Region Short Description">@context.ShortDescription</MudTd>

<MudTd DataLabel="Region Description">@context.Description</MudTd>

<MudTd DataLabel="Status">@context.Status</MudTd>

<MudTd DataLabel="Actions">

<MudButton Variant="Variant.Filled" Color="Color.Primary" OnClick="@(() => EditRegion(context))">Edit</MudButton>

<MudButton Variant="Variant.Filled" Color="Color.Error" OnClick="@(() => DeleteRegion(context.Code))">Delete</MudButton>

</MudTd>

</RowTemplate>

</MudTable>

<MudButton Variant="Variant.Filled" Color="Color.Success" OnClick="AddNewRegion">New</MudButton>

<MudDialog @bind-Open="showDialog">

<DialogContent>

<MudTextField @bind-Value="selectedRegion.Code" Label="Region Code" />

<MudTextField @bind-Value="selectedRegion.Name" Label="Region Name" />

<MudTextField @bind-Value="selectedRegion.ShortDescription" Label="Region Short Description" />

<MudTextField @bind-Value="selectedRegion.Description" Label="Region Description" />

<MudTextField @bind-Value="selectedRegion.Status" Label="Status" />

</DialogContent>

<DialogActions>

<MudButton Variant="Variant.Filled" Color="Color.Primary" OnClick="SaveRegion">Save</MudButton>

<MudButton Variant="Variant.Filled" Color="Color.Secondary" OnClick="CancelEdit">Cancel</MudButton>

</DialogActions>

</MudDialog>

@code {

private List<Region> regions;

private Region selectedRegion = new Region();

private bool showDialog = false;

protected override void OnInitialized()

{

regions = RegionService.GetAllRegions();

}

private void AddNewRegion()

{

selectedRegion = new Region();

showDialog = true;

}

private void EditRegion(Region region)

{

selectedRegion = region;

showDialog = true;

}

private void SaveRegion()

{

if (string.IsNullOrEmpty(selectedRegion.Code))

{

RegionService.AddRegion(selectedRegion);

}

else

{

RegionService.UpdateRegion(selectedRegion);

}

regions = RegionService.GetAllRegions();

showDialog = false;

}

private void DeleteRegion(string code)

{

RegionService.DeleteRegion(code);

regions = RegionService.GetAllRegions();

}

private void CancelEdit()

{

showDialog = false;

}

}

**Explanation:**

1. **MudTable**: Displays the list of regions in a table format.
2. **MudDialog**: Provides a modal dialog for adding or editing regions.
3. **RegionService**: Manages the regions in memory, allowing you to add, edit, and delete rows.
4. **RegionController**: Handles the actions and routes in an MVC structure.
5. **In-Memory Operations**: The service stores regions in a list, which allows you to simulate CRUD operations without a database.