Darshan Institute of Engineering & Technology

Certificate

This is to certify that
Mr./Miss_____Khunt Karan K._____

Enrolment No. 21010101108_____, B.Tech. CSE Semester 5th has satisfactorily completed the course in the Subject *ASP.NET Core-I (2101CS511)* in this Institute.

Submission Date: 03/10/2023

Staff in Charge **Program Coordinator**

_____ _____

Sr.	Particulars	Date	Sign
1	Write a program to calculate Celsius to Fahrenheit and vice-versa using function		
2	Write a program to find out Simple Interest using function. (I = PRN/100)		
3	Write a program to create a Simple Calculator for two numbers (Addition, Multiplication, Subtraction, Division)		
4	Write a program to create a class named Candidate with ID, Name, Age, Weight and Height as data members and also create a member functions like GetCandidateDetails() and DisplayCandidateDetails()		
5	Write a program to Define a class Distance have data member's dist1, dist2, dist3. Initialize the two data members using constructor and store their addition in third data member using function and display addition		
6	Write a program for implementing single inheritance which creates one class Account_Details for getting account information and another class Interest for calculating and displaying total interest from the data inserted from account details		
7	Write a program to create an abstract class Sum having abstract methods SumOfTwo(int a, int b) and SumOfThree(int a, int b,int c). Create another class Calculate which extends the abstract class and implements the abstract methods.		
8	Write a program to create interface named Shape. In this interface, we have three methods Circle(), Triangle() and Square() which calculates area of Circle, Triangle and Square respectively. Implement Shape interface		
9	Write a program using method overloading by changing datatype of arguments to perform addition of two integer numbers and two float numbers		
10	Create a List for StudentName and perform following operations		
11	Client Side Validation		
12	Server Side Validation		
13	Routing		
14	Areas		
15	Student Registration Database		
16	Stored Procedures		
17	CRUD Operation		

Table of Contents

Program: - 1 | Write a program to calculate Celsius to Fahrenheit and vice-versa using function.

```
using System;
class Program
{
    static double convertToCelsius(double ferTemp)
    {
        double cTemp = Math.Round((ferTemp - 32) * 5 / 9, 2);
        return cTemp;
    }

    static double convertToFahrenheit(double celTemp)
    {
        double fTemp = Math.Round((9 * celTemp) / 5 + 32, 2);
        return fTemp;
    }

    static void Main(string[] args)
    {
        double celTemp = 0, ferTemp = 0;

        Console.WriteLine("Enter 1 for Fahrenheit to Celcius: ");
        Console.WriteLine("Enter 2 for Celcius to Fahrenheit: ");

        Console.Write("Enter choice: ");
        int choice = Convert.ToInt32(Console.ReadLine());

        switch (choice)
        {
            case 1:
                Console.Write("Enter the value of temperature in Fahrenheite(°F): ");
                ferTemp = Convert.ToInt32(Console.ReadLine());
                celTemp = convertToCelsius(ferTemp);
                Console.WriteLine("Celsius temperature is(°C) : " + celTemp);
                break;

            case 2:
                Console.Write("Enter the value of temperature in Celsius(°C): ");
                celTemp = Convert.ToInt32(Console.ReadLine());
                ferTemp = convertToFahrenheit(celTemp);
                Console.WriteLine("Fahrenheite temperature is(°F) : " + ferTemp);
                break;

            default:
                Console.Write("Invalid Choice ...");
                break;
        }
    }
}
```

Output:

```
Enter 1 for Fahrenheit to Celcius:
Enter 2 for Celcius to Fahrenheit:
Enter choice: 1
Enter the value of temperature in Fahrenheite(°F): 56
Celsius temperature is(°C) : 13.33
```

```
Enter 1 for Fahrenheit to Celcius:
Enter 2 for Celcius to Fahrenheit:
Enter choice: 2
Enter the value of temperature in Celsius(°C): 78
Fahrenheite temperature is(°F) : 172.4
```

Program: - 2 | Write a program to find out Simple Interest using function. (I = PRN/100)

```
using System;
namespace Interest
{
    class Program
    {
        static void Main(string[] args)
        {
            int year;
            double princamt, rate, interest, total_amt;

            Console.Write("Enter The Loan Amount : ");
            princamt = Convert.ToDouble(Console.ReadLine());

            Console.Write("Enter The Number of Years : ");
            year = Convert.ToInt32(Console.ReadLine());

            Console.Write("Enter the Rate Of Interest : ");
            rate = Convert.ToDouble(Console.ReadLine());

            interest = princamt * year * rate / 100;

            total_amt = princamt + interest;

            Console.WriteLine("Total Amount : {0}", total_amt);
        }
    }
}
```

Output:

```
Enter The Loan Amount : 500000
Enter The Number of Years : 3
Enter the Rate Of Interest : 11
Total Amount : 665000
```

Program: - 3 | Write a program to create a Simple Calculator for two numbers (Addition, Multiplication, Subtraction, Division)

```
using System;
class Program
{
    static void Main(string[] args)
    {
        int Num1, Num2, result;
        char option;

        Console.Write("Enter the First Number : ");
        Num1 = Convert.ToInt32(Console.ReadLine());

        Console.Write("Enter the Second Number : ");
        Num2 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Main Menu");
        Console.WriteLine("1. Addition");
        Console.WriteLine("2. Subtraction");
        Console.WriteLine("3. Multiplication");
        Console.WriteLine("4. Division");

        Console.Write("Enter the Operation you want to perform : ");
        option = Convert.ToChar(Console.ReadLine());

        switch (option)
        {
            case '1':
                result = Num1 + Num2;
                Console.WriteLine("The result of Addition is: {0}", result);
                break;

            case '2':
                result = Num1 - Num2;
                Console.WriteLine("The result of Subtraction is: {0}", result);
                break;

            case '3':
                result = Num1 * Num2;
                Console.WriteLine("The result of Multiplication is:{0}", result);
                break;

            case '4':
                result = Num1 / Num2;
                Console.WriteLine("The result of Division is : {0}", result);
                break;

            default:
                Console.WriteLine("Invalid Option");
                break;
        }
    }
}
```

Output:

```
Enter the First Number : 10
Enter the Second Number : 20
Main Menu
1. Addition
2. Subtraction
3. Multiplication
4. Division
Enter the Operation you want to perform : 1
The result of Addition is : 30
```

Program: - 4 | Write a program to create a class named Candidate with ID, Name, Age, Weight and Height as data members and also create a member functions like GetCandidateDetails() and DisplayCandidateDetails().

```
using System;
class Candidate
{
    int Id, Age;
    String Name;
    double Weight, Height;

    public void GetCandidateDetails()
    {
        Console.Write("Enter Id : ");
        this.Id = int.Parse(Console.ReadLine());

        Console.Write("Enter Name : ");
        this.Name = Console.ReadLine();

        Console.Write("Enter Age : ");
        this.Age = int.Parse(Console.ReadLine());

        Console.Write("Enter Weight : ");
        this.Weight = double.Parse(Console.ReadLine());

        Console.Write("Enter Height : ");
        this.Height = double.Parse(Console.ReadLine());
    }

    public void DiplayCandidateDetails()
    {
        Console.WriteLine("ID : " + this.Id);
        Console.WriteLine("Name : " + this.Name);
        Console.WriteLine("Age : " + this.Age);
        Console.WriteLine("Weight : " + this.Weight);
        Console.WriteLine("Height : " + this.Height);
    }
}
public class Program
{
    public static void Main()
    {

        Candidate c = new Candidate();
        c.GetCandidateDetails();

        Console.WriteLine("\n----------Candidate Details-----------\n");
        c.DiplayCandidateDetails();

        Console.ReadLine();
    }
}
```

Output:

```
Enter Id : 10
Enter Name : Raj
Enter Age : 20
Enter Weight : 73
Enter Height : 176

----------Candidate Details-----------

ID : 10
Name : Raj
Age : 20
Weight : 73
Height : 176
```

Program: - 5 | Write a program to Define a class Distance have data members dist1, dist2, dist3. Initialize the two data members using constructor and store their addition in third data member using function and display addition.

```
using System;
public class Program
{
    public static void Main()
    {
        Console.WriteLine("Please enter distance 1 : ");
        double dist1 = Convert.ToDouble(Console.ReadLine());

        Console.WriteLine("Please enter distance 2 : ");
        double dist2 = Convert.ToDouble(Console.ReadLine());

        Distance d = new Distance(dist1, dist2);
        d.Sum();
        d.Display();
    }
}
public class Distance
{
    double dist1, dist2, dist3;

    public Distance(double dist1, double dist2)
    {
        this.dist1 = dist1;
        this.dist2 = dist2;
    }

    public void Sum()
    {
        this.dist3 = this.dist1 + this.dist2;
    }

    public void Display()
    {
        Console.WriteLine("Answer : {0}", this.dist3);
    }
}
```

Output:

```
Please enter distance 1 :
50
Please enter distance 2 :
100
Answer : 150
```

Program: - 6 | Write a program for implementing single inheritance which creates one class Account_Details for getting account information and another class Interest for calculating and displaying total interest from the data inserted from account details.

```
using System;
class Account_Details
{
    public int AccountNo;
    public string UserName;
    public double Principle, RateOfInerest, TimePeriod;

    public void GetAccountDetails()
    {
        Console.WriteLine("Please enter Account No : ");
        AccountNo = int.Parse(Console.ReadLine());

        Console.WriteLine("Please enter Username : ");
        UserName = Console.ReadLine();

        Console.WriteLine("Please enter Principle : ");
        Principle = double.Parse(Console.ReadLine());

        Console.WriteLine("Please enter Rate : ");
        RateOfInerest = double.Parse(Console.ReadLine());

        Console.WriteLine("Please enter Time : ");
        TimePeriod = double.Parse(Console.ReadLine());
    }
}

class Interest : Account_Details
{
    public Interest()
    {
        GetAccountDetails();
    }

    public void DisplayInterest()
    {
        Console.WriteLine("Simple Interest: {0}", ((Principle * RateOfInerest *
        TimePeriod) / 100));
    }
}

class Program
{
    public static void Main(string[] args)
    {
        Interest interest = new Interest();
        interest.DisplayInterest();
    }
}
```

Output:

```
Please enter Account No :
0790215
Please enter Username :
Raj
Please enter Principle :
10000
Please enter Rate :
7.85
Please enter Time :
2
Simple Interest: 1570
```

Program: - 7 | Write a program to create an abstract class Sum having abstract methods SumOfTwo(int a, int b) and SumOfThree(int a, int b,int c). Create another class Calculate which extends the abstract class and implements the abstract methods.

```
using System;

public abstract class Sum
{
    public abstract int sumOfTwo(int n1, int n2);
    public abstract int sumOfThree(int n1, int n2, int n3);
}

public class Program : Sum
{
    public override int sumOfTwo(int a, int b)
    {
        return a + b;
    }

    public override int sumOfThree(int a, int b, int c)
    {
        return a + b + c;
    }

    public static void Main(String[] args)
    {
        Sum s = new Program();

        Console.WriteLine("M1:" + s.sumOfTwo(10, 20));

        Console.WriteLine("M2:" + s.sumOfThree(10, 20, 30));
    }
}
```

Output:

```
M1:30
M2:60
```

Program: - 8 | Write a program to create interface named Shape. In this interface, we have three methods Circle(), Triangle() and Square() which calculates area of Circle, Triangle and Square respectively. Implement Shape interface.

```
using System;
namespace MyApplication
{
    interface Shape
    {
        double Circle(double r);
        double Triangle(double b, double h);
        double Square(double l);
    }
    class Program : Shape
    {
        public double Circle(double r)
        {
            return Math.Round(Math.PI * r * r);
        }

        public double Triangle(double b, double h)
        {
            return Math.Round((b * h) / 2);
        }

        public double Square(double l)
        {
            return (l * l);
        }

        static void Main(string[] args)
        {
            Program myObj = new Program();
            Console.WriteLine("Area of Circle: {0}", myObj.Circle(10));
            Console.WriteLine("Area of Triangle: {0}", myObj.Triangle(20, 10));
            Console.WriteLine("Area of Square: {0}", myObj.Square(15));
        }
    }
}
```

Output:

```
Area of Circle: 314
Area of Triangle: 100
Area of Square: 225
```

Program: - 9 | Write a program using method overloading by changing datatype of arguments to perform addition of two integer numbers and two float numbers.

```
using System;

class Program
{
    public static void Main(String[] args)
    {
        Console.WriteLine("Enter length of Square : ");
        double l = Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Enter Length of Rectangle : ");
        double l2 = Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Enter Breadth of Recangle : ");
        double b2 = Convert.ToDouble(Console.ReadLine());
        Console.WriteLine("Area of Square = " + Area(l));
        Console.WriteLine("Area of Rectangle = " + Area(l2, b2));
        Console.ReadLine();
    }

    public static double Area(double l)
    {
        return l * l;
    }

    public static double Area(double l, double b)
    {
        return l * b;
    }
}
```

Output:

```
Enter length of Square :
9
Enter Length of Rectangle :
21
Enter Breadth of Recangle :
15
Area of Square = 81
Area of Rectangle = 315
```

Program: - 10 | Create a List for StudentName and perform following operations:

a. Add() - To Add new student in list

b. Remove() - To Remove Student with specified index

c. RemoveRange() - To Remove student with specified range.

d. Clear() - To clear all the student from the list

```
using System;
using System.Collections.Generic;
class Program
{
    public static void Main()
    {
        List<string> studentNames = new List<string>();

        // 1. Add()
        studentNames.Add("John");
        studentNames.Add("Alice");
        studentNames.Add("Bob");

        studentNames.Add("Emily");
        Console.WriteLine("Student Names:");
        PrintList(studentNames);

        // 2. Remove()
        int indexToRemove = 2;
        studentNames.RemoveAt(indexToRemove);
        Console.WriteLine("\nStudent Names after removing student at index {0} "
                          , indexToRemove);
        PrintList(studentNames);

        // 3. RemoveRange()
        int rangeStartIndex = 0;
        int rangeCount = 2;
        studentNames.RemoveRange(rangeStartIndex, rangeCount);
        Console.WriteLine($"\nStudent Names after removing range from index
            { rangeStartIndex} to { rangeStartIndex + rangeCount - 1}:");
        PrintList(studentNames);

        // 4. Clear()
        studentNames.Clear();
        Console.WriteLine("\nStudent Names after clearing the list:");

        PrintList(studentNames);
    }

    static void PrintList(List<string> list)
    {
        if (list.Count > 0)
            foreach (object item in list)
            {
                Console.WriteLine(item);
            }
        else
            Console.WriteLine("No elements found...");
    }}
```

Output:

```
Student Names:
John
Alice
Bob
Emily

Student Names after removing student at index 2
John
Alice
Emily

Student Names after removing range from index 0 to 1:
Emily

Student Names after clearing the list:
No elements found...
```

Program: - 11 | Client Side Valdiation.

A) Create a Model

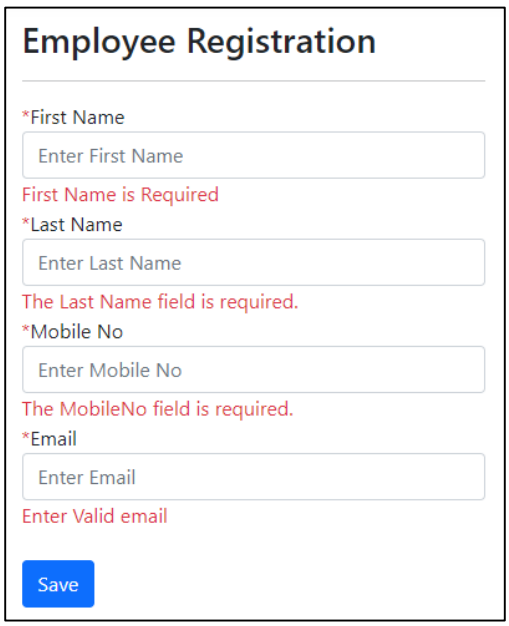
```
public class EmployeeModel
{
    [Required(ErrorMessage = "First Name is Required")]
    [Display(Name = "First name")]
    public string FirstName { get; set; }

    [Required]
    [Display(Name = "Last Name")]
    public string LastName { get; set; }

    [Required]
    [StringLength(10)]
    public string MobileNo { get; set; }

    [Required(ErrorMessage = "Enter Valid email")]
    [EmailAddress]
    public string Email { get; set; }
}
```

B) Create a View

```
<div class="form-group col-md-4">
    <h3>Employee Registrtion </h3> <hr />
</div>

<div class="form-row">

    <form role="form" method="post" asp-controller="Employee" asp-action="Save">

        <div class="form-group">
            <div class="form-group col-md-4">
                <label for="inputCity"><span class="text-danger">*</span>
                    First Name</label>
                <input type="text" class="form-control" placeholder="Enter First Name"
                          asp-for="FirstName">
                <span asp-validation-for="FirstName" class="text-danger"></span>
            </div>
        </div>

        <div class="form-group">
            <div class="form-group col-md-4">
                <label for="inputCity"><span class="text-danger">*</span>
                    Last Name</label>
                <input type="text" class="form-control" placeholder="Enter Last Code"
                    asp-for="LastName">
                <span asp-validation-for="LastName" class="text-danger"></span>
            </div>
        </div>

        <div class="form-group">
            <div class="form-group col-md-4">
                <label for="inputCity"><span class="text-danger">*</span>
```

```
                        Mobile No</label>
                <input type="text" class="form-control" placeholder="Enter State Code"
                    asp-for="MobileNo">
                <span asp-validation-for="MobileNo" class="text-danger"></span>
            </div>
        </div>

        <div class="form-group">
            <div class="form-group col-md-4">
                <label for="inputCity"><span class="text-danger">*</span>
                        Email</label>
                <input type="text" class="form-control" placeholder="Enter State Code"
                    asp-for="Email">
                <span asp-validation-for="Email" class="text-danger"></span>
            </div>
        </div>

        <br />
        <button type="submit" class="btn btn-primary">Save</button>
    </form>
</div>
```

Output:



Program: - 12 | Server Side Validation

A)Create a Model

```
namespace WebApplication1.Models
{
    4 references
    public class StudentModel
    {
        5 references
        public string? Name { get; set; }
        5 references
        public string? Address { get; set; }
        7 references
        public int? Age { get; set; }
    }
}
```

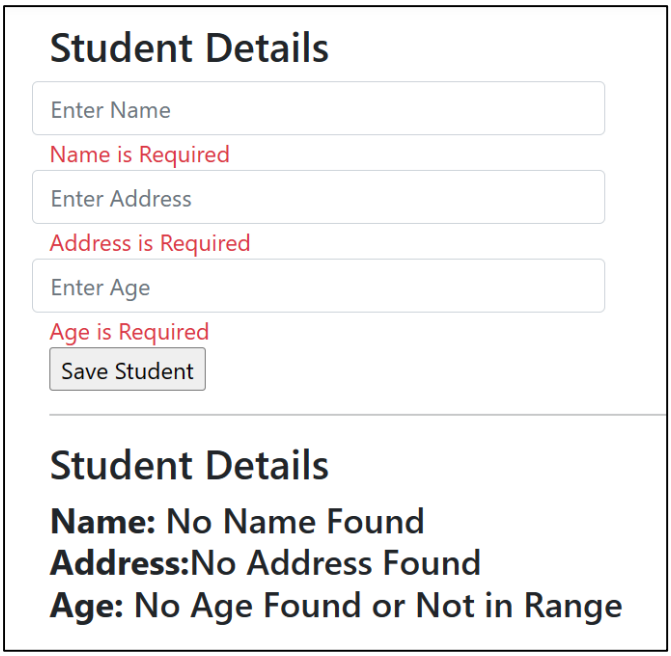
B)Create a View

```
@{ ViewBag.Title = "Home Page - Student Details"; }
<h3>Student Details</h3>
@using (Html.BeginForm("Save", "Home", FormMethod.Post))
{
    <div class="col-md-4">
        <div class="row">
            @Html.TextBoxFor(m => m.Name,new{@class="form-control",@placeholder="Enter Name"})
            @Html.ValidationMessageFor(m => m.Name,null, new{@class="text-danger"})
        </div>
        <div class="row">
            @Html.TextBoxFor(m => m.Address ,new{@class="form-control",@placeholder="Enter Address"})
            @Html.ValidationMessageFor(m => m.Address,null, new{@class="text-danger"})
        </div>
        <div class="row">
            @Html.TextBoxFor(m => m.Age , new{@class="form-control",@placeholder="Enter Age"})
            @Html.ValidationMessageFor(m => m.Age,null, new{@class="text-danger"})
        </div>
    </div>
    <input type="submit" value="Save Student" />
}
<hr/>
<h3>Student Details</h3>
<h4>
    <b>Name:</b> @ViewBag.name<br />
    <b>Address:</b>@ViewBag.address<br />
    <b>Age: </b>@ViewBag.age<br />
</h4>
```

C)Create a Controller

```
public IActionResult Save(StudentModel modelStudent)
{
    if (string.IsNullOrEmpty(modelStudent.Name))
        ModelState.AddModelError("Name", "Name is Required");
    if (string.IsNullOrEmpty(modelStudent.Address))
        ModelState.AddModelError("Address", "Address is Required");

    if (modelStudent.Age == null)
        ModelState.AddModelError("Age", "Age is Required");
    if (modelStudent.Age == 0 || modelStudent.Age > 120)
        ModelState.AddModelError("Age", "Please Enter Valid Age between 1-120");

    if (ModelState.IsValid)
    {
        ViewBag.name = modelStudent.Name;
        ViewBag.address = modelStudent.Address;
        ViewBag.age = modelStudent.Age;
        return View("StudentAddEdit");
    }
    else
    {
        ViewBag.name = "No Name Found";
        ViewBag.address = "No Address Found";
        ViewBag.age = "No Age Found or Not in Range";
        return View("StudentAddEdit");
    }
}
```

Output:



Student Details

Enter Name

Name is Required

Enter Address

Address is Required

121

Please Enter Valid Age between 1-120

Save Student

Student Details

Name: No Name Found
Address: No Address Found
Age: No Age Found or Not in Range

Student Details

Darshan Patel

Darshan University, Rajkot-Morbi Highway , 363650

21

Save Student

Student Details

Name: Darshan Patel
Address: Darshan University, Rajkot-Morbi Highway , 363650
Age: 21

Program: - 13 | Routing

- There are two types of routing for action methods:
  - Conventional Routing OR Convention-Based Routing
  - Attribute Routing OR Attribute-Based Routing

Conventional Routing

- In Conventional Based Routing, the route is determined based on the conventions defined in the Routing Middleware which will map the incoming HTTP Requests (i.e. URLs) to controller action methods.
- For example, if we issue a request to the **/Home/Index** URL, then it is the Index action method of the Home Controller class that is going to handle the request as shown in the below image.

http://localhost:52190/Home/Index

```
public class HomeController : Controller
{
    public ViewResult Index()
    {
        return View();
    }
}
```

- How the **/Home/Index** URL is mapped to the Index action method and how **/Home/Details/2** URL is mapped to the Details action method of the Home Controller class.

Program.cs

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
builder.Services.AddControllersWithViews();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(
    name: "default",
    pattern: "{controller=Home}/{action=Index}/{id?}");

app.Run();
```

This is Required for Routing

This will define the route pattern along with the Default Route

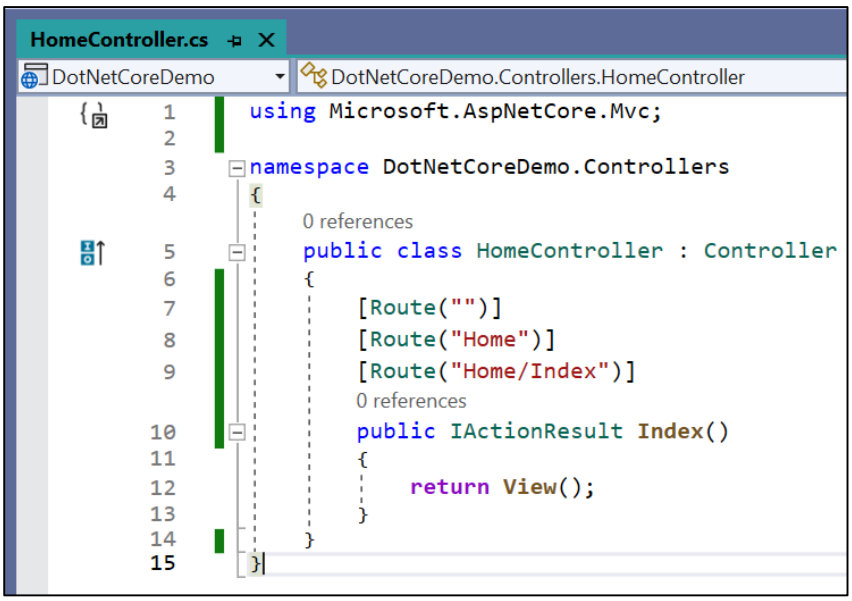
- Student Controller

```
public class StudentController : Controller
{
    public string Index()
    {
        return "Index() Action Method of StudentController";
    }

    public string Details(int? id)
    {
        return $"Details({id}) Action Method of StudentController";
    }
}
```

- Now, the URL **"/Student/Index"** is mapped to the **Index()** action method of the **StudentController** class, and the URL **"/Student/Details"** or **"/Student/Details/5"** both are mapped to the **Details(int? id)** action method of the **StudentController**.

❖ Attribute Routing

- With the help of ASP.NET Core Attribute Routing, we can use the Route attribute to define routes for our application. We can use the Route attribute either at the Controller level or at the Controller Action Methods level.



- With the above three Route attribute, now we can access the Index() action method of the HomeController using the following 3 URLs.
    - http://localhost:5280/
    - http://localhost:5280/Home
    - http://localhost:5280/Home/Index

❖ Attribute Routing (With Parameters)

- With conventional based routing, we can specify the route parameters as part of the route template.
- We can also do the same with attribute routing. That means we can also define Route Attribute with parameters. This is done by a process called Model binding.

HomeController.cs

DotNetCoreDemo | DotNetCoreDemo.Controllers.HomeController

```
using Microsoft.AspNetCore.Mvc;

namespace DotNetCoreDemo.Controllers
{
    0 references
    public class HomeController : Controller
    {
        [Route("Home/Details/{id}")]
        0 references
        public string Details(int id)
        {
            return "Details() Action Method of " +
                   "HomeController, ID Value = " + id;
        }
    }
}
```

Output

localhost:5280/Home/Details/2

http://localhost:5280/Home/Details/2

```
Details() Action Method of HomeController, ID Value = 2
```

❖ Attribute Routing (With Optional Parameters)

- Like conventional based routing, we can also make a parameter as optional in Attribute Routing.
- To make the Route parameter optional, simply add a question mark "?" at the end of the parameter.
- Now, we check the output with given images.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace DotNetCoreDemo.Controllers
{
    0 references
    public class HomeController : Controller
    {
        [Route("Home/Details/{id?}")]
        0 references
        public string Details(int id)
        {
            return "Details() Action Method of " +
                   "HomeController, ID Value = " + id;
        }
    }
}
```

localhost:5280/Home/Details/2

http://localhost:5280/Home/Details/2

```
Details() Action Method of HomeController, ID Value = 2
```

localhost:5280/Home/Details

http://localhost:5280/Home/Details

```
Details() Action Method of HomeController, ID Value = 0
```

❖ Attribute Routing (At Controller Level)

- In the ASP.NET Core MVC application, it is also possible to apply the Route() attribute on the Controller class as well as on individual action methods.
- If you want to make the attribute routing less repetitive, then you need to use the route attributes on the controller level as well as on the individual action methods level.
- The Route template applied on the controller level is prepended to the route template applied to the action method level.

HomeController.cs

```
using Microsoft.AspNetCore.Mvc;

namespace DotNetCoreDemo.Controllers
{
    [Route("Home")]
    0 references
    public class HomeController : Controller
    {
        [Route("")]
        [Route("Index")]
        0 references
        public string Index()
        {
            return "Index() Action Method of HomeController";
        }

        [Route("Details/{id?}")]
        0 references
        public string Details(int id)
        {
            return "Details() Action Method of " +
                   "HomeController, ID Value = " + id;
        }
    }
}
```

❖ Ignore the Route Template

- In order to ignore the Route Template placed at the Controller level, you need to use / or ~/ at the action method level.
- If the action method route template starts with / or ~/, then the controller route template is not going to be combined with the action method route template.

HomeController.cs

DotNetCoreDem | DotNetCoreDemo.Controllers.HomeController

```
using Microsoft.AspNetCore.Mvc;

namespace DotNetCoreDemo.Controllers
{
    [Route("Home")]
    0 references
    public class HomeController : Controller
    {
        [Route("~/Index")]
        0 references
        public string Index()
        {
            return "Index() Action Method of HomeController";
        }

        [Route("/List")]
        0 references
        public string MyList()
        {
            return "MyList() Action Method of HomeController";
        }
    }
}
```

Output

localhost:5280/index

http://localhost:5280/index

Index() Action Method of HomeController

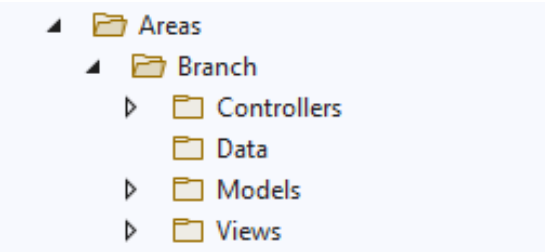
localhost:5280/list

http://localhost:5280/list

MyList() Action Method of HomeController

Program: - 14 | Areas

- Areas provide a way to partition an ASP.NET Core Web app into smaller functional groups, each with its own set of Razor Pages, controllers, views, and models.
- The ASP.NET Core runtime uses naming conventions to create the relationship between these components.
- Areas are an ASP.NET feature used to organize related functionality into a group as a separate:
  - Namespace for routing.
  - Folder structure for views and Razor Pages.
- Using areas creates a hierarchy for the purpose of routing by adding another route parameter, area, to controller and action or a Razor Page page.
- A ASP.NET Core web app using areas, controllers, and views contains the following:
  - An Area folder structure.
  - Controllers with the **[Area]** attribute to associate the controller with the area.
  - The area route added to Program.cs file.
- An area folder structure:



- The area route added to Program.cs file.

```
app.MapControllerRoute(
    name: "areas",
    pattern: "{area:exists}/{controller=Home}/{action=Index}/{id?}“
);
```

- Controllers with the **[Area]** attribute to associate the controller with the area.

```
[Area("Branch")]
public class MST_BranchController : Controller
{

}
```

Program: - 15 | Student Registration Database

Database Name: Student_Master

LOC_Country				
Key	**ColumnName**	**DataType**	**AN/NN**	**Remarks**
Primary Key	**CountryID**	**Int**	**Not Null**	**Auto Increment**
	CountryName	Varchar(100)	Not Null	
	CountryCode	Varchar(50)	Not Null	
	Created	Datetime	Not Null	Default getdate()
	Modified	Datetime	Not Null	Default getdate()

LOC_State				
Key	**ColumnName**	**DataType**	**AN/NN**	**Remarks**
Primary Key	**StateID**	**Int**	**Not Null**	**Auto Increment**
Foreign Key	**CountryID**	**Int**	**Not Null**	
	StateName	Varchar(100)	Not Null	
	StateCode	Varchar(50)	Not Null	
	Created	Datetime	Not Null	Default getdate()
	Modified	Datetime	Not Null	Default getdate()

LOC_City				
Key	**ColumnName**	**DataType**	**AN/NN**	**Remarks**
Primary Key	**CityID**	**Int**	**Not Null**	**Auto Increment**
Foreign Key	**StateID**	**Int**	**Not Null**	
Foreign Key	**CountryID**	**Int**	**Not Null**	
	CityName	Varchar(100)	Not Null	
	CityCode	Varchar(100)	Not Null	
	Created	Datetime	Not Null	Default getdate()
	Modified	Datetime	Not Null	Default getdate()

Table: MST_Branch				
Key	**ColumnName**	**DataType**	**AN/NN**	**Remarks**
Primary Key	**BranchID**	**Int**	**Not Null**	**Auto Increment**
	BranchName	Varchar(100)	Not Null	
	BranchCode	Varchar(100)	Not Null	
	Created	Datetime	Not Null	Default getdate()
	Modified	Datetime	Not Null	Default getdate()

Table: MST_Student				
Key	**ColumnName**	**DataType**	**AN/NN**	**Remarks**
Primary Key	**StudentID**	**Int**	**Not Null**	**Auto Increment**
Foreign Key	**BranchID**	**Int**	**Not Null**	**FK Branch**
Foreign Key	**CityID**	**Int**	**Not Null**	**FK City**
	StudentName	Varchar(100)	Not Null	
	MobileNoStudent	Varchar(100)	Not Null	
	MobileNoFather	Varchar(100)	Allow Null	
	Email	Varchar(100)	Not Null	
	Address	Varchar(500)	Allow Null	
	BirthDate	Datetime	Allow Null	
	Age	Int	Allow Null	
	Gender	Varchar(50)	Not Null	
	IsActive	Bit	Not Null	
	Password	Varchar(100)	Allow Null	
	Created	Datetime	Not Null	Default getdate()
	Modified	Datetime	Not Null	Default getdate()

Program: - 16 | Stored Procedures [Reference]

1. SelectAll Procedure [For List Page]

```
CREATE PROCEDURE [dbo].[PR_City_SelectAll]
AS

SELECT [dbo].[LOC_City].[CityID]
      ,[dbo].[LOC_City].[CityName]
      ,[dbo].[LOC_City].[StateID]
      ,[dbo].[LOC_State].[StateName]
      ,[dbo].[LOC_City].[Pincode]
      ,[dbo].[LOC_City].[StdCode]
      ,[dbo].[LOC_City].[CreationDate]
      ,[dbo].[LOC_Country].[CountryName]

FROM [dbo].[LOC_City]

INNER JOIN [dbo].[LOC_State]
ON [dbo].[LOC_State].[StateID] = [dbo].[LOC_City].[StateID]

INNER JOIN [dbo].[LOC_Country]
ON [dbo].[LOC_Country].[CountryID] = [dbo].[LOC_State].[CountryID]

ORDER BY  [dbo].[LOC_Country].[CountryName]
         ,[dbo].[LOC_State].[StateName]
         ,[dbo].[LOC_City].[CityName]
```

---

2. SelectByPK Procedure [Edit time record fetch & fill controls]

```
CREATE PROCEDURE [dbo].[PR_City_SelectByPK]
@CityID int
AS

SELECT
       [dbo].[LOC_City].[CityID]
      ,[dbo].[LOC_City].[CityName]
      ,[dbo].[LOC_City].[StateID]
      ,[dbo].[LOC_City].[CountryID]
      ,[dbo].[LOC_City].[Pincode]
      ,[dbo].[LOC_City].[StdCode]
      ,[dbo].[LOC_City].[CreationDate]
      ,[dbo].[LOC_Country].[CountryName]
FROM [dbo].[LOC_City]

INNER JOIN [dbo].[LOC_State]
ON [dbo].[LOC_State].[StateID] = [dbo].[LOC_City].[StateID]

INNER JOIN [dbo].[LOC_Country]
ON [dbo].[LOC_Country].[CountryID] = [dbo].[LOC_City].[CountryID]
```

```
WHERE [dbo].[LOC_City].[CityID] = @CityID

ORDER BY
            [dbo].[LOC_Country].[CountryName]
            ,[dbo].[LOC_State].[StateName]
            ,[dbo].[LOC_City].[CityName]
```

---

3. Insert Procedure [To add any new record]

```
CREATE PROCEDURE [dbo].[PR_City_Insert]

     @CityName          varchar(100),
     @StateID           int,
     @CountryID         int,
     @Pincode           varchar(6),
     @StdCode           varchar(5),
     @CreationDate      datetime
AS

INSERT INTO [dbo].[LOC_City]
 (
      [CityName]
     ,[StateID]
     ,[CountryID]
     ,[Pincode]
     ,[StdCode]
     ,[CreationDate]
)
VALUES
(
     @CityName,
     @StateID,
     @CountryID,
     @Pincode,
     @StdCode,
     @CreationDate
)
```

---

4. UpdateByPK Procedure [To update/modify existing record]

```
CREATE PROCEDURE [dbo].[PR_City_UpdateByPK]

     @CityID            int,
     @CityName          varchar(100),
     @StateID           int,
     @Pincode           varchar(50),
     @StdCode           varchar(5),
     @CountryID         int,
     @CreationDate      datetime

AS
```

```
UPDATE [dbo].[LOC_City]

   SET [CityName] = @CityName,
       [StateID] = @StateID,
       [Pincode] = @Pincode,
       [StdCode] = @StdCode,
       [CountryID] = @CountryID,
       [CreationDate] = @CreationDate

WHERE [dbo].[LOC_City].[CityID] = @CityID
```

---

5. DeleteByPK Procedure [To Delete record]

```
CREATE PROCEDURE [dbo].[PR_City_DeleteByPK]
     @CityID      int
AS
DELETE
FROM [dbo].[LOC_City]
WHERE [dbo].[LOC_City].[CityID] = @CityID
```

Program: - 17 | CRUD Operation [Reference]

_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - ListInsertUpdate</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="~/ListInsertUpdate.styles.css" asp-append-version="true" />
</head>
<body>
<!-- Header -->
    <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-controller="Home" asp-action="Index">
                            <img class="img img-responsive" width="120px"
                            src="~/Images/Demo.png"></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="Home" asp-action="Privacy">Privacy</a>
                        </li>
                    </ul>
                </div>
            </div>
        </nav>
    </header>
<!-- Body -->
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
<!-- Footer -->
    <footer class="border-top footer text-muted">
        <div class="container">
            &copy; 2023 - ListInsertUpdate -
       <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>
    <script src="~/js/main.js"></script>
    @await RenderSectionAsync("Scripts", required: false)
</body></html>
```

StateList.cshtml

```
@model DataTable

<div class="container">
   <a class="btn btn-success" asp-controller="Home" asp-action="Add">Add New State</a>
<br />
<div class="row container">
    <table class="table">
        <thead>
            <tr>
                <th>Country Name</th>
                <th>State Name</th>
                <th>State Code</th>
                <th>Action</th>
                  </tr>
       </thead>
      <tbody>
          @*Iterate through every Row of Data Table*@

          @{
              if (Model.Rows.Count > 0)
                  foreach (DataRow dr in Model.Rows)
                  {
                      <tr>

                          <td>@dr["CountryName"]</td>
                          <td>@dr["StateName"]</td>
                          <td>@dr["StateCode"]</td>
                          <td>
                            <form method="post" asp-controller="Home" asp-action="Delete">
                              <input type="hidden" name="StateID" value="@dr["StateID"]">
                              <button type="submit" class="btn btn-danger btn-xs"
                                onclick="return fun1();">Delete</button>
                              <a class="btn btn-info btn-xs" asp-controller="Home"
                                asp-action="Add" asp-route-StateID="@dr["StateID"]">
                                      Edit</a>
                            </form>
                          </td>
                      </tr>
                  }
              else
              {
                  <tr>
                    <td colspan="5" class="text-center">
                     <p class="col-form-label">No Record Found</p>
                    </td>
                  </tr>
              }
          }
      </tbody>
  </table>
 </div>
</div>
@section Scripts
    {
<script>
    function fun1() {
       if (confirm("Are you sure you want to delete?")) {
           return true;
       }
```

```
        else {
            return false;
        }
    }
</script>
```

StateAddEdit.cshtml

```
<h4 class="font-green-sharp">@TempData["Message"]</h4>
<h3>State Add/Edit </h3>
<hr />
<div class="form-row">
    <form role="form" method="post" asp-controller="Home" asp-action="Save">

        @Html.HiddenFor(x=>x.StateID)

        <div class="form-row">
            <div class="form-group col-md-6">
        <label for="inputState"><span class="text-danger">*</span>Country</label>

  <select id="inputState" class="form-control" asp-for="CountryID"
  asp-items="@(new SelectList(ViewBag.CountryList,"CountryID","CountryName"))">

       <option disabled selected>Select Country</option>
  </select>
         <span asp-validation-for="CountryID" class="text-danger"></span>

         </div>
        </div>
        <div class="form-group">
            <div class="form-group col-md-6">

      <label for="inputCity"><span class="text-danger">*</span>State Name</label>
      <input type="text" class="form-control" placeholder="Enter State Name"
              asp-for="StateName">
      <span asp-validation-for="StateName" class="text-danger"></span>

     </div>
  </div>
  <div class="form-group">
     <div class="form-group col-md-6">

      <label for="inputCity"><span class="text-danger">*</span>State Code</label>
      <input type="text" class="form-control" placeholder="Enter State Code"
              asp-for="StateCode">
      <span asp-validation-for="StateCode" class="text-danger"></span>

      </div>
  </div>
  <br />
  <button type="submit" class="btn btn-primary">Save</button>
  <a class="btn btn-danger" asp-controller="Home" asp-action="Index">Cancel</a>
</form>
</div>

@section Scripts{
<partial name="_ValidationScriptsPartial.cshtml" />
  }
```

StateController.cs

```
public class StateController : Controller
    {
        private IConfiguration Configuration;
        public StateController(IConfiguration _configuration)
        {
            Configuration = _configuration;
        }

//StateList Action

        public IActionResult Index()
        {

            FillCountryDDL();
            DataTable dt = new DataTable();

        string str =this.Configuration.GetConnectionString("ABConnectionString");
            SqlConnection conn1 = new SqlConnection(str);
            conn1.Open();
            SqlCommand objCmd = conn1.CreateCommand();
            objCmd.CommandType = CommandType.StoredProcedure;
            objCmd.CommandText = "PR_State_SelectAll";

            SqlDataReader objSDR = objCmd.ExecuteReader();

            //Check if Data Retrived from DB Has data or not
            if (objSDR.HasRows)
            {
                    dt.Load(objSDR);
            }

            return View(dt);
        }

//Add | Edit Action

        public IActionResult Add(int? StateID)
        {

            FillCountryDDL();


            if (StateID != null)
            {

                SqlConnection objConn = new
SqlConnection(this.Configuration.GetConnectionString("ABConnectionString"));

                objConn.Open();

                SqlCommand objCmd = objConn.CreateCommand();
                objCmd.CommandType = CommandType.StoredProcedure;

                objCmd.CommandText = "PR_State_SelectByPK";
                objCmd.Parameters.AddWithValue("@StateID", StateID);

                SqlDataReader objSDR = objCmd.ExecuteReader();

                LOC_StateModel state = new LOC_StateModel();
```

```
                if (objSDR.HasRows)
                {
                    while (objSDR.Read())
                    {
                        state.StateName = objSDR["StateName"].ToString();
                        state.CountryID = Convert.ToInt32(objSDR["CountryID"]);
                        state.StateCode = objSDR["StateCode"].ToString();
                    }
                }
                return View("StateAddEdit", state);
            }



            return View("StateAddEdit");
        }

//Save Record Action

        [HttpPost]
        public IActionResult Save(LOC_StateModel model_LOC_State)
        {
            if (ModelState.IsValid)
            {

                //Get Connection String and Create connection to SQL
                SqlConnection objConn = new
SqlConnection(this.Configuration.GetConnectionString("ABConnectionString"));
                objConn.Open();
                SqlCommand objCmd = objConn.CreateCommand();

                //CommandType defines type of command to be executed
                objCmd.CommandType = CommandType.StoredProcedure;

                //check if new record is inserted or existing record is going to
                //be updated and call appropriate SP
                if (model_LOC_State.StateID == null)
                {
                  objCmd.CommandText = "PR_State_Insert";
                  objCmd.Parameters.AddWithValue("@Created", DateTimeOffset.Now);
                }
                else
                {
                  objCmd.CommandText = "PR_State_UpdateByPK";
             objCmd.Parameters.AddWithValue("@StateID", model_LOC_State.StateID);
                }

          //pass parameters to SP
          objCmd.Parameters.AddWithValue("@StateName", model_LOC_State.StateName);
          objCmd.Parameters.AddWithValue("@StateCode", model_LOC_State.StateCode);
          objCmd.Parameters.AddWithValue("@CountryID", model_LOC_State.CountryID);
          objCmd.Parameters.AddWithValue("@Modified", DateTimeOffset.Now);


                //check for command executed successfully
                if (Convert.ToBoolean(objCmd.ExecuteNonQuery()))
                {
                    if (model_LOC_State.StateID == null)
                        TempData["Message"] = "Record Inserted Successfully";
                    else
                    {
                        TempData["Message"] = "Record Updated Successfully";
```

```
                        return RedirectToAction("Index");
                    }
                }
            }

            return RedirectToAction("Add");
        }

//Delete Action

        public IActionResult Delete(int StateID)
        {
            SqlConnection objConn = new
SqlConnection(this.Configuration.GetConnectionString("ABConnectionString"));
            objConn.Open();

            SqlCommand objCmd = objConn.CreateCommand();

            objCmd.CommandType = CommandType.StoredProcedure;
            objCmd.CommandText = "PR_State_DeleteByPK";
            objCmd.Parameters.Add("@StateID", SqlDbType.Int).Value = StateID;

            if (Convert.ToBoolean(objCmd.ExecuteNonQuery()))
            {
                TempData["Message"] = " ";
            }
            objConn.Close();

            return RedirectToAction("Index");
        }

//Fill DropDown for Country

        public void FillCountryDDL()
        {

            string str = this.Configuration.GetConnectionString("ABConnectionString");

            List<LOC_CountryDropDownModel> loc_Country = new
                          List<LOC_CountryDropDownModel>();
            SqlConnection objConn = new SqlConnection(str);
            objConn.Open();
            SqlCommand objCmd = objConn.CreateCommand();
            objCmd.CommandType = CommandType.StoredProcedure;
            objCmd.CommandText = "PR_Country_SelectDropDownList";
            SqlDataReader objSDR = objCmd.ExecuteReader();


            if (objSDR.HasRows)
            {
                while (objSDR.Read())
                {
                    LOC_CountryDropDownModel country = new
                                 LOC_CountryDropDownModel()
                    {
                        CountryID = Convert.ToInt32(objSDR["CountryID"]),
                        CountryName = objSDR["CountryName"].ToString()
                    };
                    loc_Country.Add(country);
                }
                objSDR.Close();
            }
```

```
            objConn.Close();
            ViewBag.CountryList = loc_Country;

        }
    }
```

Model Classes

- **StateModel.cs**

```
public class LOC_StateModel
    {
        public int? StateID { get; set; }

        [Required(ErrorMessage = "State Name is Required")]
        [DisplayName("State Name")]
        public string StateName { get; set; }

        [Required(ErrorMessage = "Please Select Country")]
        [DisplayName("Country Name")]
        public int CountryID { get; set; }

        [Required(ErrorMessage = "State Code is Required")]
        [DisplayName("State Code")]
        [StringLength(maximumLength:100, MinimumLength =1)]
        public string StateCode { get; set; }

        public DateTime Created { get; set; }

        public DateTime Modified { get; set; }


    }
```

- **CountryModel.cs**

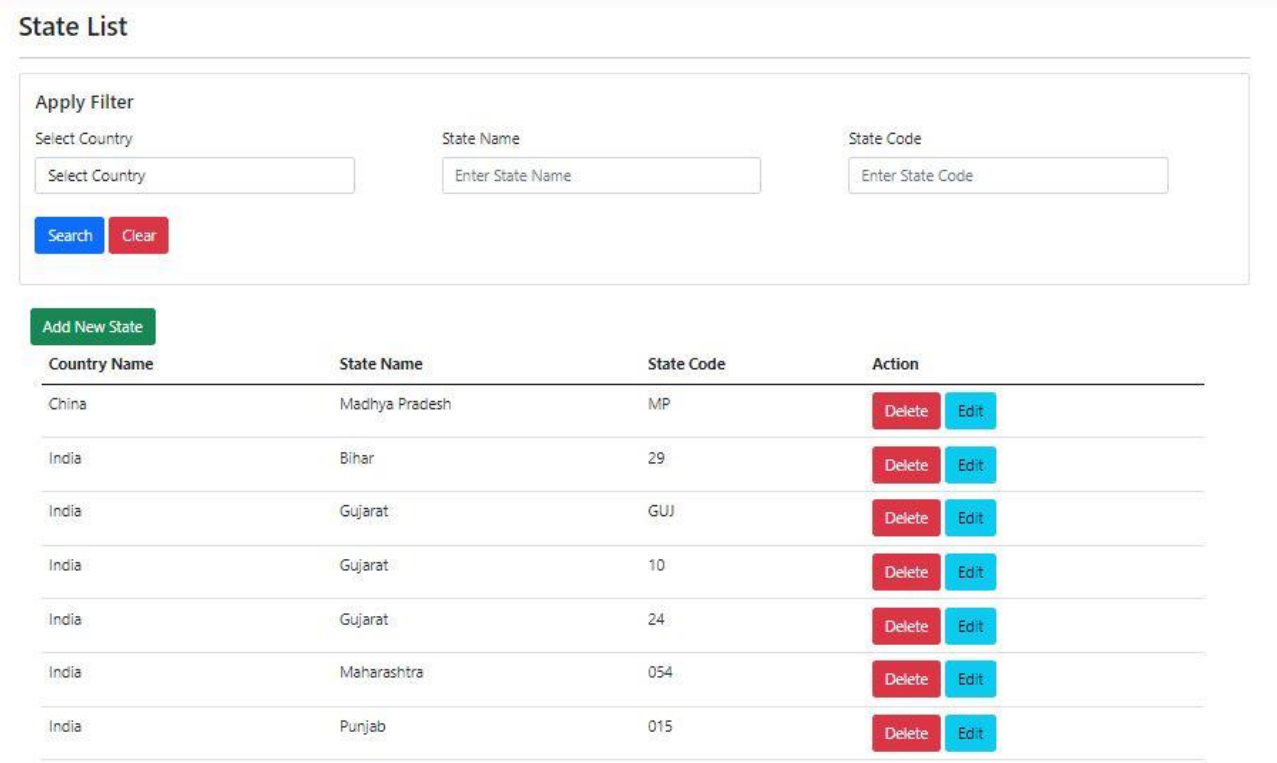
```
public class LOC_CountryModel
    {
        public int? CountryID { get; set; }

        [Required]
        [DisplayName("Country Name")]
        public string CountryName { get; set; }

        [Required]
        [DisplayName("Country Code")]
        public string CountryCode { get; set; }

        public DateTime Created { get; set; }

        public DateTime Modified { get; set; }
    }
```

```
public class LOC_CountryDropDownModel
{
    public int CountryID { get; set; }
    public string CountryName { get; set; }
}
```

appsettings.json (For Connection string to Database)

```
"ConnectionStrings": {
    "ABConnectionString": "Data Source=DESKTOP-SJ5GNSA;Initial Catalog=AddressBook;
Trusted_Connection=true"
  }
```

List Page Output:



Add Edit Page Output: