=========Person Controller ========================

```
using Assignment1.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace Assignment1.Controllers
{
   public class PersonsController : Controller
   {
      // GET: Persons
      public ActionResult Index()
      {
         return View();
      }

      // GET: Persons/Details/5
      public ActionResult Details()
      {

         string name = (string)Session["value1"];
         Person obj = Person.GetDetails(name);
         return View(obj);
      }
```

```
// GET: Persons/Create
public ActionResult Create()
{
   List < City > cities = City.GetAllCities();
   List<SelectListItem> cityList = new List<SelectListItem>();
   foreach (var item in cities)
   {
      cityList.Add( new SelectListItem { Text = item.CityName, Value = item.CityId.ToString() });
   }
   ViewBag.Cities = cityList;
   return View();
}

// POST: Persons/Create
[HttpPost]
public ActionResult Create(Person obj)
{
   try
   {
      Person.InsertPerson(obj);
      return RedirectToAction("Login");
   }
   catch
   {
      return View();
   }
}
```

```
// GET: Persons/Edit/5
public ActionResult Edit(string id)
{
    Person obj = Person.GetDetails(id);
    List<City> cities = City.GetAllCities();
    List<SelectListItem> cityList = new List<SelectListItem>();
    foreach (var item in cities)
    {
        cityList.Add(new SelectListItem { Text = item.CityName, Value = item.CityId.ToString() });
    }
    ViewBag.Cities = cityList;
    return View(obj);
}

// POST: Persons/Edit/5
[HttpPost]
public ActionResult Edit(string id, Person obj)
{
    try
    {
        Person.UpdatePerson(obj);

        return RedirectToAction("Details");
    }
    catch
    {
```

```
        return View();
    }
}

// GET: Persons/Delete/5
public ActionResult Delete(int id)
{
    return View();
}

// POST: Persons/Delete/5
[HttpPost]
public ActionResult Delete(int id, FormCollection collection)
{
    try
    {
        // TODO: Add delete logic here

        return RedirectToAction("Index");
    }
    catch
    {
        return View();
    }
}

//================Login============================
```

```
// GET: Persons/Login
public ActionResult Login()
{
   HttpCookie objCookie = Request.Cookies["PersonLogin"];
   if (objCookie==null)
   {
      return View();
   }
   else
   {
      string name = objCookie.Values["key1"];
      Session["value1"] = name;
      return RedirectToAction("Details");
   }

}

// POST: Persons/Login
[HttpPost]
public ActionResult Login(Person obj)
{
   try
   {


      bool valid = Person.ValidPerson(obj);
```

```
        if (valid)
        {
            Session["value1"] = obj.LoginName;
            if (obj.isActive)
            {
                HttpCookie objCookie = new HttpCookie("PersonLogin");
                objCookie.Expires = DateTime.Now.AddDays(1);
                objCookie.Values["key1"] = obj.LoginName;
                Response.Cookies.Add(objCookie);
            }



            return RedirectToAction("Details");
        }
        else
        {
            return RedirectToAction("Create");
        }

    }
    catch
    {
        return View();
    }
}
//======================logout====================
```

```
        public ActionResult Logout()
        {
            Session.Abandon();
            HttpCookie objCookie = new HttpCookie("PersonLogin");
            objCookie.Expires = DateTime.Now.AddDays(-1);
            Response.Cookies.Add(objCookie);

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

---

========Person.cs==========

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace Assignment1.Models
{
    public class Person
    {
        [Required(ErrorMessage = "Please Enter a Unique Login Name")]
        /* [Range(1,20,ErrorMessage ="Login Name Should be less than 20 words")]*/
        /* [RegularExpression(@"^[a-zA-Z''-'\s]{1,20}$",
             ErrorMessage = "Characters are not allowed.")]*/
```

```
public string LoginName { get; set; }

[Required(ErrorMessage = "Please Enter Password")]
[DataType(DataType.Password)]
public string Password { get; set; }

[Required(ErrorMessage = "Please Enter Confirm Password")]
[Compare("Password", ErrorMessage = "Please enter same confirm password as password")]
[DataType(DataType.Password)]
public string ConfirmPassword { get; set; }

[Required(ErrorMessage = "Please Enter Full Name")]
[DataType(DataType.Text)]
public string FullName { get; set; }

[Required(ErrorMessage = "Please Enter Email ID")]
[DataType(DataType.EmailAddress, ErrorMessage = "Enter Valid Email ID")]
public string EmailId { get; set; }
public int CityId { get; set; }

[Required(ErrorMessage = "Please enter Phone Number")]
/* [Range(1,10,ErrorMessage ="Please Enter Valid Mobile Number")]*/
[DataType(DataType.PhoneNumber, ErrorMessage = "Please Enter Valid Mobile ")]
public long Phone { get; set; }


public static void InsertPerson(Person obj)
```

```
    {
        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=JKJuly2022;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFa
ilover=False";
        try
        {
            cn.Open();
            SqlCommand cmdInsert = new SqlCommand();
            cmdInsert.Connection = cn;
            cmdInsert.CommandType = System.Data.CommandType.StoredProcedure;
            cmdInsert.CommandText = "InsertPerson";
            cmdInsert.Parameters.AddWithValue("@LoginName", obj.LoginName);
            cmdInsert.Parameters.AddWithValue("@Password", obj.Password);
            cmdInsert.Parameters.AddWithValue("@FullName", obj.FullName);
            cmdInsert.Parameters.AddWithValue("@EmailId", obj.EmailId);
            cmdInsert.Parameters.AddWithValue("@CityId", obj.CityId);
            cmdInsert.Parameters.AddWithValue("@Phone", obj.Phone);

            cmdInsert.ExecuteNonQuery();
        }
        catch (Exception ex)
        {

        }
        finally
        {
            cn.Close();
```

```
        }
    }

    /*  public static void UpdatePerson(Person obj)
     {
        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=JKJuly2022;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFa
ilover=False";
        try
        {
            cn.Open();
            SqlCommand cmdUpdate = new SqlCommand();
            cmdUpdate.Connection = cn;
            cmdUpdate.CommandType = System.Data.CommandType.StoredProcedure;
            cmdUpdate.CommandText = "UpdatePerson";
            cmdUpdate.Parameters.AddWithValue("@LoginName", obj.LoginName);
            cmdUpdate.Parameters.AddWithValue("@Password", obj.Password);
            cmdUpdate.Parameters.AddWithValue("@FullName", obj.FullName);
            cmdUpdate.Parameters.AddWithValue("@EmailId", obj.EmailId);
            cmdUpdate.Parameters.AddWithValue("@CityId", obj.CityId);
            cmdUpdate.Parameters.AddWithValue("@Phone", obj.Phone);

            cmdUpdate.ExecuteNonQuery();
        }
       catch (Exception ex)
        {
```

```
        }
        finally
        {
            cn.Close();
        }
    }*/
    //=====================================
    public static void UpdatePerson(Person obj)
    {
        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=JKJuly2022;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFa
ilover=False";
        // try
        //  {
            cn.Open();
            SqlCommand cmdUpdate = new SqlCommand();
            cmdUpdate.Connection = cn;
            cmdUpdate.CommandType = System.Data.CommandType.StoredProcedure;
            //cmdUpdate.CommandText = "update from Persons set Password=@Password,
FullName=@FullName, EmailId=@EmailId, CityId=@CityId, Phone=@Phone where
LoginName=@LoginName";
            cmdUpdate.CommandText = "UpdatePerson";
            cmdUpdate.Parameters.AddWithValue("@LoginName", obj.LoginName);
            cmdUpdate.Parameters.AddWithValue("@Password", obj.Password);
            cmdUpdate.Parameters.AddWithValue("@FullName", obj.FullName);
            cmdUpdate.Parameters.AddWithValue("@EmailId", obj.EmailId);
            cmdUpdate.Parameters.AddWithValue("@CityId", obj.CityId);
```

```
            cmdUpdate.Parameters.AddWithValue("@Phone", obj.Phone);


            cmdUpdate.ExecuteNonQuery();
        //  }
         //  catch (Exception ex)
         //  {


        //  }
         //  finally
        //  {
            cn.Close();
         //  }
        }




        //=======================================




        public static bool ValidPerson(Person obj)
        {
            Person per = new Person();
            SqlConnection cn = new SqlConnection();
            cn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=JKJuly2022;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFa
ilover=False";
            try
```

```
        {
            cn.Open();
            SqlCommand cmdValid = new SqlCommand();
            cmdValid.Connection = cn;
            cmdValid.CommandType = System.Data.CommandType.Text;
            cmdValid.CommandText = "select count(*) from Persons where LoginName= @LoginName
and Password=@Password";
            cmdValid.Parameters.AddWithValue("@LoginName", obj.LoginName);
            cmdValid.Parameters.AddWithValue("@Password", obj.Password);

            int val = (int)cmdValid.ExecuteScalar();

            if (val == 1)
            {
                return true;
            }


        }
        catch
        {

        }
        finally
        {
            cn.Close();
        }
        return false;
```

```
    }

    public bool isActive { get; set; }

    //================================================

    public static Person GetDetails(string name)
    {
        Person per = new Person();
        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=JKJuly2022;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFa
ilover=False";
        try
        {
            cn.Open();
            SqlCommand cmdValid = new SqlCommand();
            cmdValid.Connection = cn;
            cmdValid.CommandType = System.Data.CommandType.Text;
            cmdValid.CommandText = "select * from Persons where LoginName= @LoginName";
            cmdValid.Parameters.AddWithValue("@LoginName", name);

            SqlDataReader dr = cmdValid.ExecuteReader();
            while (dr.Read())
            {
                per.FullName = (string)dr["FullName"];
                per.EmailId = (string)dr["EmailId"];
                per.LoginName = (string)dr["LoginName"];
```

```
                per.Password = (string)dr["Password"];
                per.CityId = (int)dr["CityId"];
                per.Phone = (long)dr["Phone"];
            }
            dr.Close();

        }
        catch
        {

        }
        finally
        {
            cn.Close();
        }
        return per;
    }
  }
}
```

---

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Linq;
using System.Web;

namespace Assignment1.Models
{
```

```
public class City
{
    public int CityId { get; set; }
    public string CityName { get; set; }

    public static List<City> GetAllCities()
    {
        List<City> cities = new List<City>();
        SqlConnection cn = new SqlConnection();
        cn.ConnectionString = @"Data Source=(localdb)\MSSQLLocalDB;Initial
Catalog=JKJuly2022;Integrated Security=True;Connect
Timeout=30;Encrypt=False;TrustServerCertificate=False;ApplicationIntent=ReadWrite;MultiSubnetFa
ilover=False";
        try
        {
            cn.Open();
            SqlCommand cmdSelect = new SqlCommand();
            cmdSelect.Connection = cn;
            cmdSelect.CommandType = System.Data.CommandType.Text;
            cmdSelect.CommandText = "select * from Cities";

            SqlDataReader dr = cmdSelect.ExecuteReader();
            while (dr.Read())
            {
                cities.Add(new City { CityId = (int)dr["CityId"], CityName = (string)dr["CityName"]});
            }
            dr.Close();

        }
```

```
            catch
            {

            }
            finally
            {
                cn.Close();
            }
            return cities;
        }
    }
}
```

---

```
using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.Linq;
using System.Web;

namespace DataAnnotations.Models
{
    public class Employee
    {
        [Key]
        public int EmpNo { get; set; }

        [DataType(DataType.Text)]
        [Required(ErrorMessage = "Please enter name")]
```

```
[StringLength(10, ErrorMessage = "The {0} value cannot exceed {1} characters. ")]
public string Name { get; set; }

[Range(1000, 500000, ErrorMessage = "Please enter values between 1000-500000")]
[MaxLength(6), MinLength(4)]
[Display(Name = "Basic Salary")]
[DataType(DataType.Currency)]
public decimal Basic { get; set; }
public int DeptNo { get; set; }

[ScaffoldColumn(false)]
public string Dummy { get; set; }

[EmailAddress]
public string EmailId { get; set; }

[Required(ErrorMessage = "Please enter password")]
[DataType(DataType.Password)]
public string Password { get; set; }

[Required(ErrorMessage = "Please enter confirm password")]
[Compare("Password", ErrorMessage = "Password and confirm password should be the same")]
[DataType(DataType.Password)]
public string ConfirmPassword { get; set; }

// Allow up to 40 uppercase and lowercase
// characters. Use custom error.
```

```
        [RegularExpression(@"^[a-zA-Z''-'\s]{1,40}$",
            ErrorMessage = "Characters are not allowed.")]
        public string FirstName { get; set; }

        // Allow up to 40 uppercase and lowercase
        // characters. Use standard error.
        [RegularExpression(@"^[a-zA-Z''-'\s]{1,40}$")]
        public string LastName { get; set; }
    }
}
```

//https://docs.microsoft.com/en-us/dotnet/api/system.componentmodel.dataannotations?view=netframework-4.8

---

(localdb)\MsSqlLocalDb

```
@{ string s = Convert.ToString(item.ProductId);}
        @Html.ActionLink(s,"Edit",new { id = item.ProductId})
```

---

```
@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>
```

RenderPartial

```
@{
    Html.RenderPartial("PartialView1");
}
```

with html.partial....

@Html.Partial("PartialView1")

<h2>after partial</h2>

---

```
List<Department> deps = Department.getAllDepartments();
        List<SelectListItem> objDepts1 = new List<SelectListItem>();

        foreach (var item in deps)
        {
           objDepts1.Add(new SelectListItem { Text = item.DeptName, Value = item.DeptNo.ToString()
});
        }
        ViewBag.Departments = objDepts1;
```

==========

```
@Html.DropDownListFor(model => model.DeptNo,
(IEnumerable<SelectListItem>)ViewBag.Departments)
```

---