ADO EMPLOYEE CRUD APP

1. **Create model for Employee**

   Employee.cs

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

namespace ADO_CrudApp.Models  // create model to fetch data ....
{
        public class Employee
        {
                public int Id { get; set; }
    public string Name { get; set; }
    public string Email { get; set; }
    public int Age { get; set; }

  }
}
```

2. **Create Employee DAL**

   EmployeeDAL.cs

```
using System.Collections.Generic;
using System.Configuration;
using ADO_CrudApp.Models;
using System.Data;
using System.Data.SqlClient;

namespace ADO_MVC.Models
{
  public class EmployeeDAL
  {
    string conString =
ConfigurationManager.ConnectionStrings["EmployeeDbConnection"].ToString();
```

```
        // Get All Employees
        public List<Employee> GetAllEmployees()
        {
            List<Employee> employees = new List<Employee>();
            using (SqlConnection con = new SqlConnection(conString))
            {
                SqlCommand cmd = new SqlCommand("SELECT * FROM Employees", con);
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();
                while (rdr.Read())
                {
                    employees.Add(new Employee
                    {
                        Id = (int)rdr["Id"],
                        Name = rdr["Name"].ToString(),
                        Email = rdr["Email"].ToString(),
                        Age = (int)rdr["Age"]
                    });
                }
            }
            return employees;
        }

        // Insert Employee
        public void AddEmployee(Employee emp)
        {
            using (SqlConnection con = new SqlConnection(conString))
            {
                string query = "INSERT INTO Employees (Name, Email, Age) VALUES
(@Name, @Email, @Age)";
                SqlCommand cmd = new SqlCommand(query, con);

                cmd.Parameters.AddWithValue("@Name", emp.Name);
                cmd.Parameters.AddWithValue("@Email", emp.Email);
                cmd.Parameters.AddWithValue("@Age", emp.Age);
                con.Open();
                cmd.ExecuteNonQuery();
            }
        }

        // Update Employee
        public void UpdateEmployee(Employee emp)
        {
            using (SqlConnection con = new SqlConnection(conString))
            {
                string query = "UPDATE Employees SET Name=@Name, Email=@Email,
Age=@Age WHERE Id=@Id";
                SqlCommand cmd = new SqlCommand(query, con);
                cmd.Parameters.AddWithValue("@Id", emp.Id);
                cmd.Parameters.AddWithValue("@Name", emp.Name);
                cmd.Parameters.AddWithValue("@Email", emp.Email);
                cmd.Parameters.AddWithValue("@Age", emp.Age);
                con.Open();
                cmd.ExecuteNonQuery();
            }
        }

        // Delete Employee
        public void DeleteEmployee(int id)
        {
            using (SqlConnection con = new SqlConnection(conString))
            {
                string query = "DELETE FROM Employees WHERE Id=@Id";
                SqlCommand cmd = new SqlCommand(query, con);
                cmd.Parameters.AddWithValue("@Id", id);
                con.Open();
                cmd.ExecuteNonQuery();
            }
        }

        // Get Employee By ID
        public Employee GetEmployeeById(int id)
        {
            Employee emp = new Employee();
            using (SqlConnection con = new SqlConnection(conString))
            {
                string query = "SELECT * FROM Employees WHERE Id=@Id";
                SqlCommand cmd = new SqlCommand(query, con);
                cmd.Parameters.AddWithValue("@Id", id);
                con.Open();
                SqlDataReader rdr = cmd.ExecuteReader();
                if (rdr.Read())
                {
                    emp.Id = (int)rdr["Id"];
                    emp.Name = rdr["Name"].ToString();
                    emp.Email = rdr["Email"].ToString();
                    emp.Age = (int)rdr["Age"];
                }
            }
            return emp;
        }
    }
}
```

3. Configure database connection in web.config

```
<connectionStrings>
          <add name="EmployeeDbConnection"
                    connectionString="data
source=SHILPA\SQLEXPRESS;Database=EmployeeDB;Integrated Security=True;"
                 />
</connectionStrings>
```

4. Create an Employee Controller

```
using ADO_CrudApp.Models;
using ADO_MVC.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Mvc;

namespace ADO_CrudApp.Controllers
{
    public class EmployeeController : Controller
    {

        EmployeeDAL empDAL = new EmployeeDAL();

        // GET: Employee
        public ActionResult Index()
        {
            var employees = empDAL.GetAllEmployees();
            return View(employees);

        }
```

```
// GET: Employee/Create
public ActionResult Create()
{
   return View();
}

// POST: Employee/Create
[HttpPost]

public ActionResult Create(Employee emp)
{
   if (ModelState.IsValid)
   {
      empDAL.AddEmployee(emp);
      return RedirectToAction("Index");
   }
   return View(emp);
}


// GET: Employee/Edit/5
public ActionResult Edit(int id)
{
   var emp = empDAL.GetEmployeeById(id);
   return View();
}

// POST: Employee/Edit/5
[HttpPost]

public ActionResult Edit(Employee emp)
{
   if (ModelState.IsValid)
   {
      empDAL.UpdateEmployee(emp);
      return RedirectToAction("Index");
   }
   return View(emp);
}


// GET: Employee/Delete/5
public ActionResult Delete(int id)
{
```

```
            var emp = empDAL.GetEmployeeById(id);
            return View();
        }

        // POST: Employee/Delete/5
        [HttpPost, ActionName("Delete")]
        public ActionResult DeleteConfirmed(int id)
        {
            empDAL.DeleteEmployee(id);
            return RedirectToAction("Index");
        }

    }
}
```

5. **Create view for crud operations**

Index.cshtml

```
@model IEnumerable<ADO_CrudApp.Models.Employee>

@{
    ViewBag.Title = "Index";
}

<h2>Index</h2>

<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table class="table">
    <tr>
        <th>
            @Html.DisplayNameFor(model => model.Name)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Email)
        </th>
        <th>
            @Html.DisplayNameFor(model => model.Age)
        </th>
        <th></th>
    </tr>
```

```
@foreach (var item in Model) {
    <tr>
        <td>
            @Html.DisplayFor(modelItem => item.Name)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Email)
        </td>
        <td>
            @Html.DisplayFor(modelItem => item.Age)
        </td>
        <td>
            @Html.ActionLink("Edit", "Edit", new { id=item.Id }) |

            @Html.ActionLink("Delete", "Delete", new { id=item.Id })
        </td>
    </tr>
}

</table>
```

Create.cshtml

```
@model ADO_CrudApp.Models.Employee

@{
    ViewBag.Title = "Create";
}

<h2>Create</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Employee</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class =
"control-label col-md-2" })
```

```
          <div class="col-md-10">
             @Html.EditorFor(model => model.Name, new { htmlAttributes = new {
@class = "form-control" } })
             @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
          </div>
       </div>

       <div class="form-group">
          @Html.LabelFor(model => model.Email, htmlAttributes: new { @class =
"control-label col-md-2" })
          <div class="col-md-10">
             @Html.EditorFor(model => model.Email, new { htmlAttributes = new {
@class = "form-control" } })
             @Html.ValidationMessageFor(model => model.Email, "", new { @class =
"text-danger" })
          </div>
       </div>

       <div class="form-group">
          @Html.LabelFor(model => model.Age, htmlAttributes: new { @class =
"control-label col-md-2" })
          <div class="col-md-10">
             @Html.EditorFor(model => model.Age, new { htmlAttributes = new { @class
= "form-control" } })
             @Html.ValidationMessageFor(model => model.Age, "", new { @class = "text-
danger" })
          </div>
       </div>

       <div class="form-group">
          <div class="col-md-offset-2 col-md-10">
             <input type="submit" value="Create" class="btn btn-primary" />
          </div>
       </div>
   </div>
}

<div>
   @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
   @Scripts.Render("~/bundles/jqueryval")
```

```
}
```

Edit.cshtml

```
@model ADO_CrudApp.Models.Employee

@{
    ViewBag.Title = "Edit";
}

<h2>Edit</h2>


@using (Html.BeginForm())
{
    @Html.AntiForgeryToken()

    <div class="form-horizontal">
        <h4>Employee</h4>
        <hr />
        @Html.ValidationSummary(true, "", new { @class = "text-danger" })
        @Html.HiddenFor(model => model.Id)

        <div class="form-group">
            @Html.LabelFor(model => model.Name, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Name, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Name, "", new { @class =
"text-danger" })
            </div>
        </div>

        <div class="form-group">
            @Html.LabelFor(model => model.Email, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Email, new { htmlAttributes = new {
@class = "form-control" } })
                @Html.ValidationMessageFor(model => model.Email, "", new { @class =
"text-danger" })
            </div>
        </div>
        <div class="form-group">
            @Html.LabelFor(model => model.Age, htmlAttributes: new { @class =
"control-label col-md-2" })
            <div class="col-md-10">
                @Html.EditorFor(model => model.Age, new { htmlAttributes = new { @class
= "form-control" } })
                @Html.ValidationMessageFor(model => model.Age, "", new { @class = "text-
danger" })
            </div>
        </div>

        <div class="form-group">
            <div class="col-md-offset-2 col-md-10">
                <input type="submit" value="Save" class="btn btn-success" />
            </div>
        </div>
    </div>
}

<div>
    @Html.ActionLink("Back to List", "Index")
</div>

@section Scripts {
    @Scripts.Render("~/bundles/jqueryval")
}
```

Delete.cshtml

```
@model ADO_CrudApp.Models.Employee


@{

    ViewBag.Title = "Delete";

}


<h2>Delete</h2>
```

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

<div>

    <h4>Employee</h4>

    <hr />

    <dl class="dl-horizontal">

        <dt>

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

        </dt>


        <dd>

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

        </dd>


        <dt>

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

        </dt>


        <dd>

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

        </dd>


        <dt>

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

        </dt>


        <dd>

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

        </dd>
    </dl>


    @using (Html.BeginForm())
    {
        @Html.AntiForgeryToken()


        <div class="form-actions no-color">
            <input type="submit" value="Delete" class="btn btn-danger" /> |
            @Html.ActionLink("Back to List", "Index")
        </div>
    }
</div>
```

6. **Run Application**

   http://localhost:xxxx/Employee/Index