BAHRIA UNIVERSITY (KARACHI CAMPUS)

ASSIGNMENT # 1 - SPRING 2021

Cloud Computing (SEN-325)

[CLO 3]

Class: **BSE-8 (B)** Max Marks: **5**
Course Instructor: **Engr. Muhammad Faisal**
Student Name: **Qaiser Abbas** Registration No. **57245**

[The marks of this assignment may increase or decrease]

Read Carefully:

- The deadline for this assignment is *before* or *on* **Monday, 13th May, 2021.**

WARNINGS:

- This is an individual assignment; you must implement it by yourself. Any form of plagiarism will result in receiving zero in the assignment.
- Late submission will not be accepted. Any assignment submitted after the cutoff time will receive zero.

In this assignment you are supposed to develop a web API that will contain the following

1) Model:

Class ➔ **Order Attributes:**

a) public int OrdId { get; set; }
b) public string CusName { get; set; }
c) public int Laptop { get; set; }
d) public int LCD { get; set; }
e) public int Phone { get; set; }

2) Repository:

The Repository contain an interface **IOrderRepository** and a class **OrderRepository**
Interface ➔ **IOrderRepository Methods:**

a) IEnumerable<Order> GetAll();
b) Order Get(int id);
c) Order Add(Order item);
d) void Remove(int id);
e) bool Update(Order item);

Methods Description:
This code creates an IOrderRepository interface and declares the following methods:

a) GetAll(): An implementation of this method should return an IEnumerable<Order> object that contains details of all the order.

b) Get(int id): An implementation of this method should return an Order object of the specified Id passed as parameters to the method.

c) Add(Order item): An implementation of this method should add a new Order object to the OrderRepository object. Once added, this method should return the new Order object.
d) Remove(int id): An implementation of this method should remove an Order object specified by the Id passed as parameter from the OrderRepository object.
e) Update(Order item): An implementation of this method should update the OrderRepository object with the Order object passed as parameter

<u>OrderRepository</u>

<u>class OrderRepository : IOrderRepository</u>

In this code, the OrderRepository class implements the IOrderRepository interface. For each of the methods declared in the IOrderRepository interface, the OrderRepository class provides implementation to retrieve, add, and delete order that the Order model represents.

3) <u>Controller</u>

Class OrderController

a) The Get() method accesses the Order repository to return all orders as an IEnumerable<Order> object.
b) The Get(int id) method accesses the order repository to return an order with the specified Id as an Order object. Similarly, the GetOrderByCusName(string CName) method returns all orders of the specified customer as an IEnumerable<Order> object.
c) The Post(Order order) method adds the Order object passed as parameter to the order repository. The Put(int id, Order order) method updates an album in the album repository based on the specified id.
d) The Delete(int id) method deletes an order from the order repository based on the specified id.

Defining Routes:

Once you have created the Web API controller, you need to register it with the ASP.NET routing Framework. When the Web API application receives a request, the routing Framework tries to match the Uniform Resource Identifier (URI) against one of the route templates defined in the WebApiConfig.cs file.

The assignment should be submitted as zip file containing the entire project solution. The deadline is:

13th May, 2021

API is like an endpoint which asks for certain inputs and based upon those inputs it will perform job for you without you having to install anything new or you having to worry about dependencies etc.

A web API is same as any other api in terms that you dont have to worry about dependancies, internal working. All you need is a way to be able to access that api over web (i.e. Via internet) and give the required inputs. Wait for the api to finish the job and get the output back in form of JSON or XML. Since those are more purely data-focused things like map data, lists of tweets matching a search term, additional user profile data, that sort of thing.

Repository:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Assignment1.Models;

namespace Assignment1.RepositoryPattern
{
    interface IOrderRepository
    {
        Task<IEnumerable<Order>> GetAll();
        Task<Order> Get(int id);
        Task Add(Order item);
        Task Remove(int id);
        Task Update(Order item);

    }
}

using Microsoft.SqlServer.Server;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Threading.Tasks;
using Assignment1.Models;
using System.Data.Entity;

namespace Assignment1.RepositoryPattern
{
    public class OrderRepository : IOrderRepository
    {

        private readonly SQLDatabaseContext db = new SQLDatabaseContext();
        public async Task Add(Order item)
        {
            item.OrderId = int.Parse(Guid.NewGuid().ToString());
            db.Orders.Add(item);
            try
            {
                await db.SaveChangesAsync();
            }
            catch
            {
                throw;
            }

        }

        public async Task<Order> Get(int id)
        {
            try
            {
                Order order = await db.Orders.FindAsync(id);
                if (order == null)
                {
                    return null;
                }
                return order;
            }
            catch
            {
                throw;
            }

        }

        public async Task<IEnumerable<Order>> GetAll()
        {
            try
            {
                var order = await db.Orders.ToListAsync();
                return order.AsQueryable();
            }
            catch
            {
                throw;
            }

        }

        public async Task Remove(int id)
        {
            try
            {
                Order order = await db.Orders.FindAsync(id);
                db.Orders.Remove(order);
                await db.SaveChangesAsync();
            }
            catch
            {
                throw;
            }
        }

        public async Task Update(Order item)
        {
            try
            {
                db.Entry(item).State = EntityState.Modified;
                await db.SaveChangesAsync();
            }
            catch
            {
                throw;
            }

        }
        private bool OrderExists(int id)
        {
            return db.Orders.Count(e => e.OrderId == id) > 0;
        }

    }
}
```

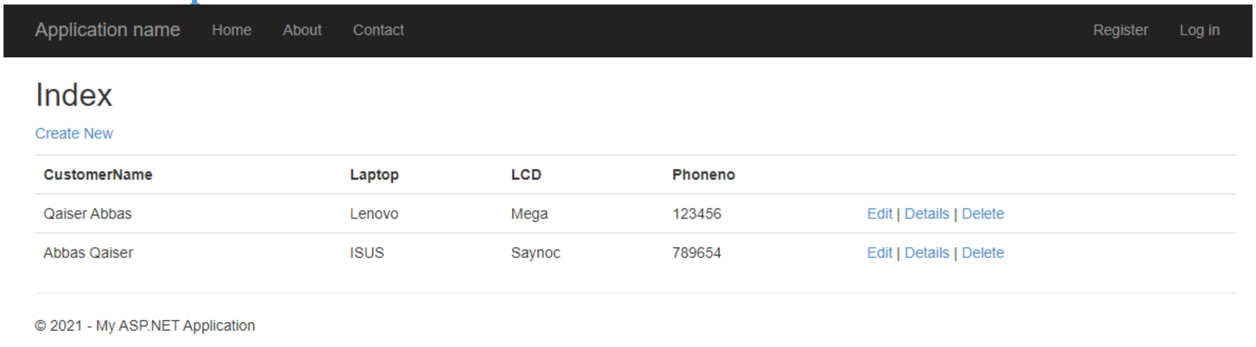
Model:

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


namespace Assignment1.Models
{
    [Table("Order")]
    public class Order
    {
        [Key]
        public int OrderId { get; set; }
        public string CustomerName { get; set; }
        public string Laptop { get; set; }
        public string LCD { get; set; }
        public int Phoneno { get; set; }
    }
}
```

Controller:

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

namespace Assignment1.Controllers
{
    public class OrdersController : Controller
    {
        private SQLDatabaseContext db = new SQLDatabaseContext();

        // GET: Orders
        public ActionResult Index()
        {
            return View(db.Orders.ToList());
        }

        // GET: Orders/Details/5
        public ActionResult Details(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Order order = db.Orders.Find(id);
            if (order == null)
            {
                return HttpNotFound();
            }
            return View(order);
        }

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

        // POST: Orders/Create
        // To protect from overposting attacks, please enable the specific properties you 
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Create([Bind(Include = 
"OrderId,CustomerName,Laptop,LCD,Phoneno")] Order order)
        {
            if (ModelState.IsValid)
            {
                db.Orders.Add(order);
                db.SaveChanges();
                return RedirectToAction("Index");
            }

            return View(order);
        }

        // GET: Orders/Edit/5
        public ActionResult Edit(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Order order = db.Orders.Find(id);
            if (order == null)
            {
                return HttpNotFound();
            }
            return View(order);
        }

        // POST: Orders/Edit/5
        // To protect from overposting attacks, please enable the specific properties you
want to bind to, for
        // more details see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult Edit([Bind(Include =
"OrderId,CustomerName,Laptop,LCD,Phoneno")] Order order)
        {
            if (ModelState.IsValid)
            {
                db.Entry(order).State = EntityState.Modified;
                db.SaveChanges();
                return RedirectToAction("Index");
            }
            return View(order);
        }

        // GET: Orders/Delete/5
        public ActionResult Delete(int? id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Order order = db.Orders.Find(id);
            if (order == null)
            {
                return HttpNotFound();
            }
            return View(order);
        }

        // POST: Orders/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public ActionResult DeleteConfirmed(int id)
        {
            Order order = db.Orders.Find(id);
            db.Orders.Remove(order);
            db.SaveChanges();
            return RedirectToAction("Index");
        }

        protected override void Dispose(bool disposing)
        {
            if (disposing)
            {
                db.Dispose();
            }
            base.Dispose(disposing);
        }
    }
}
```

Output:

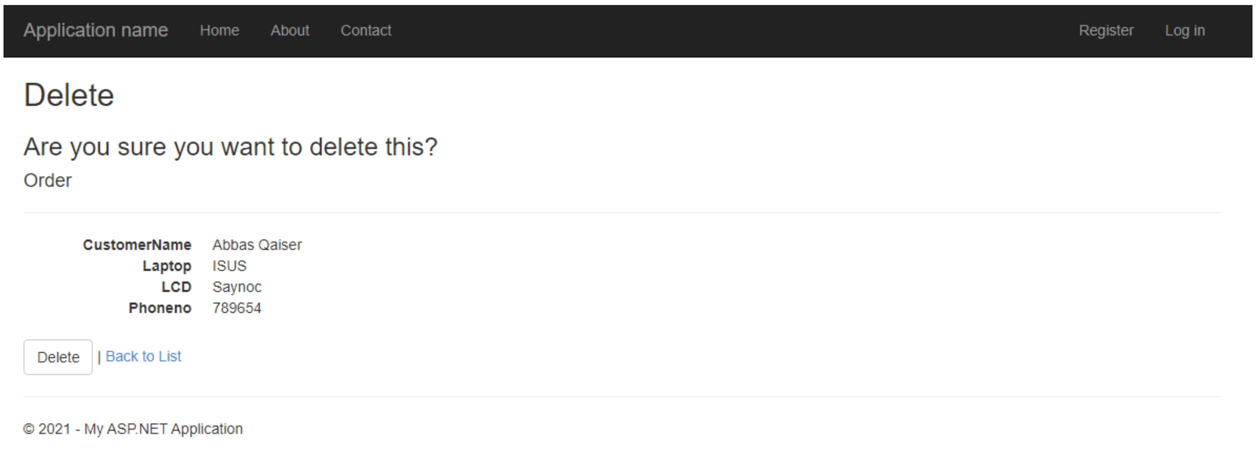


Index

Create New

CustomerName	Laptop	LCD	Phoneno	
Qaiser Abbas	Lenovo	Mega	123456	Edit | Details | Delete
Abbas Qaiser	ISUS	Saynoc	789654	Edit | Details | Delete

© 2021 - My ASP.NET Application



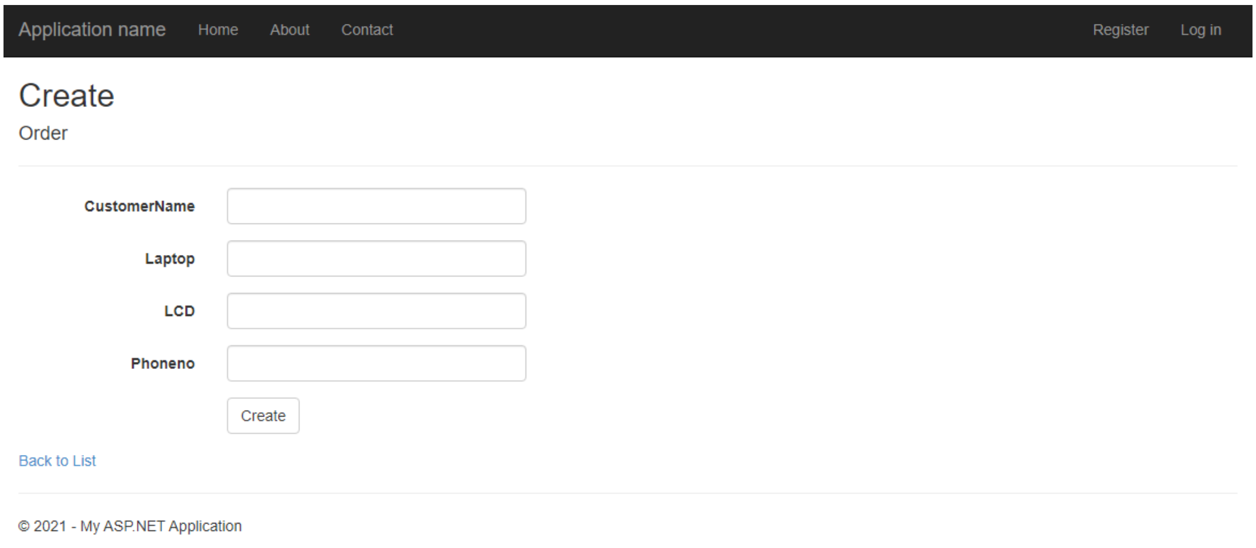
Delete

Are you sure you want to delete this?

Order

CustomerName	Abbas Qaiser
Laptop	ISUS
LCD	Saynoc
Phoneno	789654

© 2021 - My ASP.NET Application

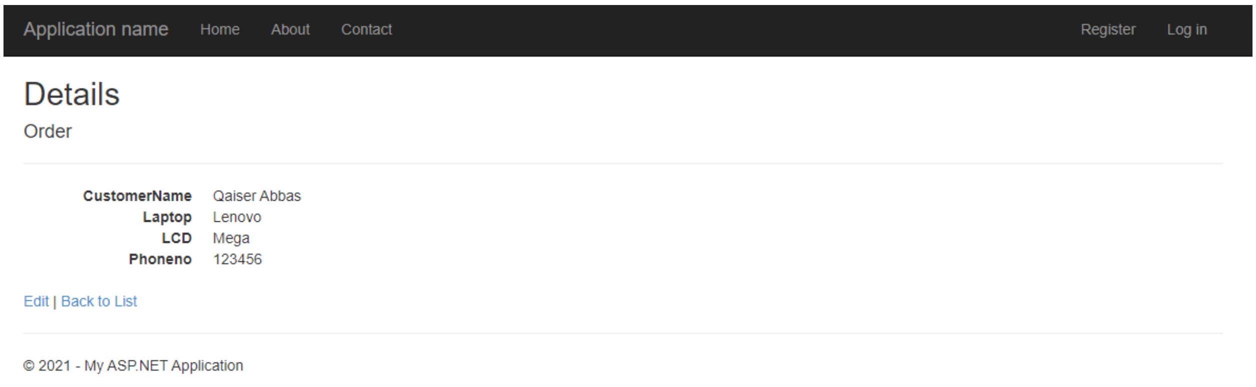


Create

Order

CustomerName	
Laptop	
LCD	
Phoneno	

© 2021 - My ASP.NET Application



Application name Home About Contact Register Log in

Details

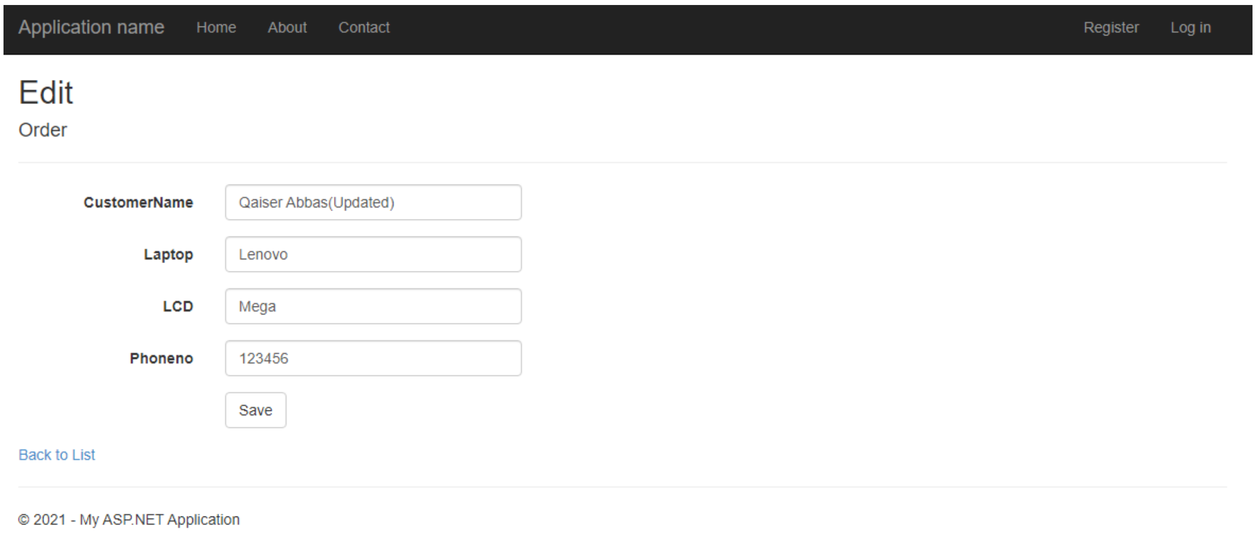
Order

CustomerName	Qaiser Abbas
Laptop	Lenovo
LCD	Mega
Phoneno	123456

Edit | Back to List

© 2021 - My ASP.NET Application

Update:



Application name Home About Contact Register Log in

Edit

Order

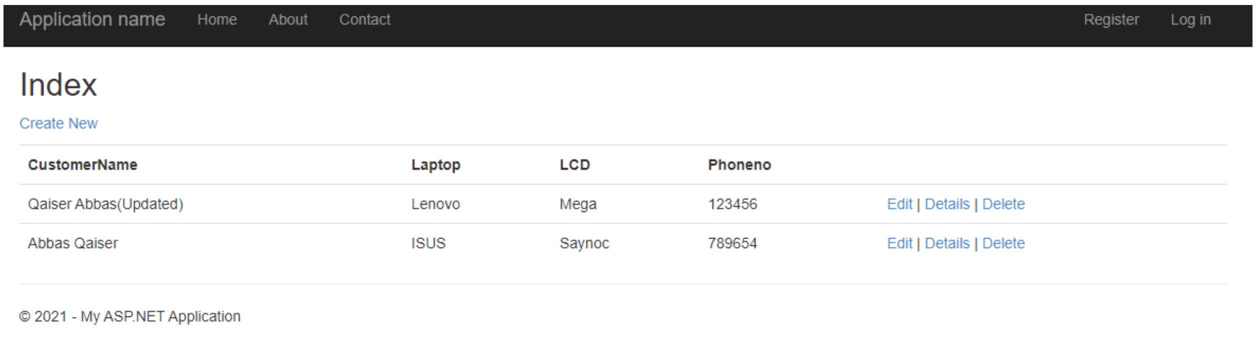
CustomerName	Qaiser Abbas(Updated)
Laptop	Lenovo
LCD	Mega
Phoneno	123456

Save

Back to List

© 2021 - My ASP.NET Application

Updated Data:



Application name Home About Contact Register Log in

Index

Create New

CustomerName	Laptop	LCD	Phoneno	
Qaiser Abbas(Updated)	Lenovo	Mega	123456	Edit | Details | Delete
Abbas Qaiser	ISUS	Saynoc	789654	Edit | Details | Delete

© 2021 - My ASP.NET Application

Note: The below code file is of more than 50 Mbs. LMS does not support to upload more than 15mb file. So I am providing github repository link that contain all the code.

https://github.com/iQaiserAbbas/Web-Api-csharp.git/