Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**13**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **1** | **Employee API with CRUD** |
| **2** | **Create a payroll api and perform crud operations on it** |
|  |  |

**Task No. 1: Employee API with CRUD.**

**Solution:**

**Employees Controller:**

namespace MVCwithWebAPI.Controllers

{

public class EmployeesController : Controller

{

readonly string apiBaseAddress = ConfigurationManager.AppSettings["apiBaseAddress"];

public async Task<ActionResult> Index()

{

IEnumerable<Employee> employees = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync("employees/get");

if (result.IsSuccessStatusCode)

{

employees = await result.Content.ReadAsAsync<IList<Employee>>();

}

else

{

employees = Enumerable.Empty<Employee>()

;

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

return View(employees);

}

public async Task<ActionResult> Details(int id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"employees/details/{ id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Employee>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "Name,Address,Gender,Company,Designation")] Employee employee)

{

if (ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.PostAsJsonAsync("employees/Create", employee);

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

}

return View(employee);

}

public async Task<ActionResult> Edit(int id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress)

;

var result = await client.GetAsync($"employees/details/{ id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Employee>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "Id,Name,Address,Gender,Company,Designation")] Employee employee

)

{

if (ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.PutAsJsonAsync("employees/edit", employee);

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

return RedirectToAction("Index");

}

return View(employee);

}

public async Task<ActionResult> Delete(int id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Employee employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"employees/details/{ id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Employee>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.DeleteAsync($"employees/delete/{ id}");

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

return View();

}

}

}

**Employee Repository:**

namespace MVCwithWebAPI.Models

{

public class EmployeeRepository : IEmployeeRepository

{

private readonly SqlDbContext db = new SqlDbContext(

);

public async Task Add(Employee employee)

{

employee.Id = Convert.ToInt32(Guid.NewGuid());

db.Employees.Add(employee);

try

{

await db.SaveChangesAsync();

}

catch

{

throw;

}

}

public async Task<Employee> GetEmployee(int id)

{

try

{

Employee employee = await db.Employees.FindAsync(id);

if (employee == null)

{

return null;

}

return employee;

}

catch

{

throw;

}

}

public async Task<IEnumerable<Employee>> GetEmployees()

{

try

{

var employees = await db.Employees.ToListAsync();

return employees.AsQueryable();

}

catch

{

throw;

}

}

public async Task Update(Employee employee)

{

try

{

db.Entry(employee).State = EntityState.Modified;

await db.SaveChangesAsync();

}

catch

{

throw;

}

}

public async Task Delete(int id)

{

try

{

Employee employee = await db.Employees.FindAsync(id);

db.Employees.Remove(employee);

await db.SaveChangesAsync();

}

catch

{

throw;

}

}

private bool EmployeeExists(int id)

{

return db.Employees.Count(e => e.Id == id) > 0;

}

}

}

**Connection String:**

<connectionStrings>

<add name="SqlConn" connectionString="Data Source=DIYAN-ALI-SHAIK\SQLEXPRESS;Initial Catalog=TestOel;Integrated Security=True" providerName="System.Data.SqlClient" />

</connectionStrings>

<appSettings>

<add key="webpages:Version" value="3.0.0.0" />

<add key="webpages:Enabled" value="false" />

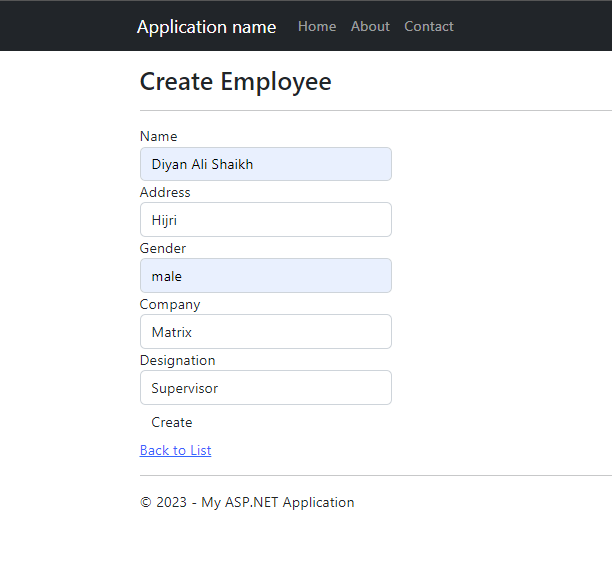
<add key="ClientValidationEnabled" value="true" />

<add key="UnobtrusiveJavaScriptEnabled" value="true" />

<add key="apiBaseAddress" value="https://localhost:44399/api/" />

</appSettings>

**Output:**

**A screenshot of a computer

Description automatically generated with medium confidence**

Toyota

Shah Faisl

Shah Faisl

Muaz shahzad

Toyota

Muaz shahzad

**A screenshot of a computer

Description automatically generated with low confidence**

Toyota

Shah Faisl

Muaz shahzad

**Task No. 2: Create a payroll api and perform crud operations on it.**

**Solution:**

**Payroll Controller:**

namespace MVCwithWebAPI.Controllers

{

public class PayrollsController : Controller

{

readonly string apiBaseAddress = ConfigurationManager.AppSettings["apiBaseAddress"];

public async Task<ActionResult> Index()

{

IEnumerable<Payroll> employees = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync("payroll/get");

if (result.IsSuccessStatusCode)

{

employees = await result.Content.ReadAsAsync<IList<Payroll>>();

}

else

{

employees = Enumerable.Empty<Payroll>()

;

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

return View(employees);

}

public async Task<ActionResult> Details(int id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Payroll employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"payroll/details/{id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Payroll>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

public ActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Create([Bind(Include = "Name,Salary")] Payroll employee)

{

if (ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.PostAsJsonAsync("payroll/Create", employee);

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

}

return View(employee);

}

public async Task<ActionResult> Edit(int id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Payroll employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress)

;

var result = await client.GetAsync($"payroll/details/{id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Payroll>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<ActionResult> Edit([Bind(Include = "Id,Name,Salary")] Payroll employee

)

{

if (ModelState.IsValid)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.PutAsJsonAsync("payroll/edit", employee);

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

return RedirectToAction("Index");

}

return View(employee);

}

public async Task<ActionResult> Delete(int id)

{

if (id == null)

{

return new HttpStatusCodeResult(HttpStatusCode.BadRequest);

}

Payroll employee = null;

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var result = await client.GetAsync($"payroll/details/{id}");

if (result.IsSuccessStatusCode)

{

employee = await result.Content.ReadAsAsync<Payroll>();

}

else

{

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

}

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<ActionResult> DeleteConfirmed(int id)

{

using (var client = new HttpClient())

{

client.BaseAddress = new Uri(apiBaseAddress);

var response = await client.DeleteAsync($"payroll/delete/{id}");

if (response.IsSuccessStatusCode)

{

return RedirectToAction("Index");

}

else

ModelState.AddModelError(string.Empty, "Server error try after some time.");

}

return View();

}

}

}

**Payroll Repository:**

namespace MVCwithWebAPI.Models

{

public class PayrollRepository : IPayrollRepository

{

private readonly SqlDbContext db = new SqlDbContext(

);

public async Task Add(Payroll employee)

{

employee.Id = Convert.ToInt32(Guid.NewGuid());

db.Payroll.Add(employee);

try

{

await db.SaveChangesAsync();

}

catch

{

throw;

}

}

public async Task<Payroll> GetEmployee(int id)

{

try

{

Payroll employee = await db.Payroll.FindAsync(id);

if (employee == null)

{

return null;

}

return employee;

}

catch

{

throw;

}

}

public async Task<IEnumerable<Payroll>> GetEmployees()

{

try

{

var employees = await db.Payroll.ToListAsync();

return employees.AsQueryable();

}

catch

{

throw;

}

}

public async Task Update(Payroll employee)

{

try

{

db.Entry(employee).State = EntityState.Modified;

await db.SaveChangesAsync();

}

catch

{

throw;

}

}

public async Task Delete(int id)

{

try

{

Payroll employee = await db.Payroll.FindAsync(id);

db.Payroll.Remove(employee);

await db.SaveChangesAsync();

}

catch

{

throw;

}

}

private bool EmployeeExists(int id)

{

return db.Payroll.Count(e => e.Id == id) > 0;

}

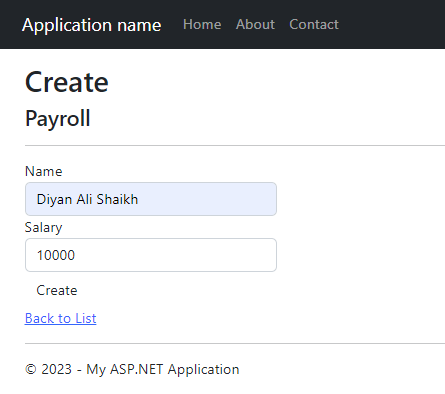
}

}

**Output:**

**A screenshot of a computer

Description automatically generated with medium confidence**

****

20000

Muaz Shahzad