Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**6**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| **1** | Implement an ASP.NET Core Web API employing RESTful principles, incorporating endpoints for GET (to fetch products), POST (to add products), PUT (to update products), and DELETE (to remove products) methods, streamlining the administration of a product inventory system. |
| 2 | Construct an ASP.NET Core Web API adhering to REST conventions, designed to manage student records through CRUD operations (GET, POST, PUT, and DELETE), enabling the seamless addition, retrieval, modification, and deletion of student information. |

Submitted On:

4 April 2024

\_\_\_\_\_\_\_\_\_\_\_\_

(Date: DD/MM/YY)

**Task 1**

Implement an ASP.NET Core Web API employing RESTful principles, incorporating endpoints for GET (to fetch products), POST (to add products), PUT (to update products), and DELETE (to remove products) methods, streamlining the administration of a product inventory system.

**Products.cs**

namespace LAB06Task1.Models

{

public class Product

{

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string Price { get; set; }

}

}

**ProductsDbContext.cs**

using Microsoft.EntityFrameworkCore;

namespace LAB06Task1.Models

{

public class ProductsDbContext : DbContext

{

public ProductsDbContext(DbContextOptions options):base(options) {

}

public DbSet<Product> productss { get; set; }

}

}

**ProductssController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using LAB05Task1.Models;

namespace LAB05Task1.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class productssController : ControllerBase

{

private readonly ProductsDbContext \_context;

public productssController(ProductsDbContext context)

{

\_context = context;

}

// GET: api/productss

[HttpGet]

public async Task<ActionResult<IEnumerable<Product>>> Getproductss()

{

if (\_context.productss == null)

{

return NotFound();

}

return await \_context.productss.ToListAsync();

}

// GET: api/productss/5

[HttpGet("{id}")]

public async Task<ActionResult<Product>> GetProduct(int id)

{

if (\_context.productss == null)

{

return NotFound();

}

var product = await \_context.productss.FindAsync(id);

if (product == null)

{

return NotFound();

}

return product;

}

// PUT: api/productss/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutProduct(int id, Product product)

{

if (id != product.Id)

{

return BadRequest();

}

\_context.Entry(product).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!ProductExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/productss

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Product>> PostProduct(Product product)

{

if (\_context.productss == null)

{

return Problem("Entity set 'productssDbContext.productss' is null.");

}

\_context.productss.Add(product);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetProduct", new { id = product.Id }, product);

}

// DELETE: api/productss/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteProduct(int id)

{

if (\_context.productss == null)

{

return NotFound();

}

var product = await \_context.productss.FindAsync(id);

if (product == null)

{

return NotFound();

}

\_context.productss.Remove(product);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool ProductExists(int id)

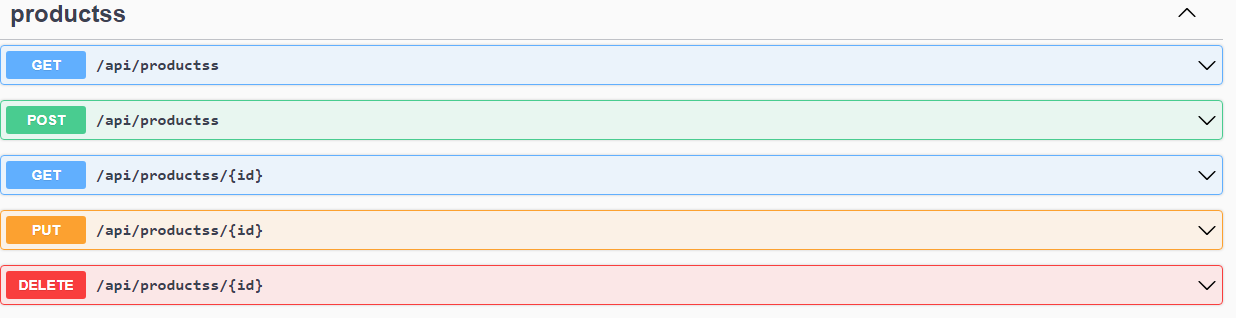
{

return (\_context.productss?.Any(e => e.Id == id)).GetValueOrDefault();

}

}

}

****

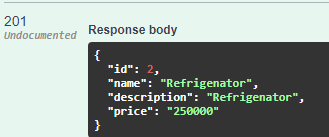
**A screen shot of a computer code

Description automatically generatedGet Request Response**

**A screenshot of a computer program

Description automatically generatedPost Request Response**

**Get Request api/productss/2**

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**Task 2**

Construct an ASP.NET Core Web API adhering to REST conventions, designed to manage student records through CRUD operations (GET, POST, PUT, and DELETE), enabling the seamless addition, retrieval, modification, and deletion of student information.

**Student.cs**

using System;

using System.Collections.Generic;

namespace CCLAB06Task2.Models

{

public partial class Student

{

public int Id { get; set; }

public string StudentName { get; set; } = null!;

public string StudentGender { get; set; } = null!;

public int Age { get; set; }

public int Standard { get; set; }

public string FatherName { get; set; } = null!;

}

}

**StudentContext.cs**

using System;

using System.Collections.Generic;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Metadata;

namespace CCLAB06Task2.Models

{

public partial class EmployeeContext : DbContext

{

public EmployeeContext()

{

}

public EmployeeContext(DbContextOptions<EmployeeContext> options)

: base(options)

{

}

public virtual DbSet<Student> Students { get; set; } = null!;

public virtual DbSet<Teacher> Teachers { get; set; } = null!;

public virtual DbSet<UserTable> UserTables { get; set; } = null!;

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

#warning To protect potentially sensitive information in your connection string, you should move it out of source code. You can avoid scaffolding the connection string by using the Name= syntax to read it from configuration - see https://go.microsoft.com/fwlink/?linkid=2131148. For more guidance on storing connection strings, see http://go.microsoft.com/fwlink/?LinkId=723263.

}

}

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

modelBuilder.Entity<Student>(entity =>

{

entity.ToTable("Student");

entity.Property(e => e.Id).HasColumnName("ID");

entity.Property(e => e.FatherName)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.StudentGender)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.StudentName)

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<Teacher>(entity =>

{

entity.ToTable("Teacher");

entity.Property(e => e.Id).HasColumnName("ID");

entity.Property(e => e.Name)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.Qualification)

.HasMaxLength(50)

.IsUnicode(false);

});

modelBuilder.Entity<UserTable>(entity =>

{

entity.ToTable("UserTable");

entity.Property(e => e.Email)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.Gender)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.Name)

.HasMaxLength(50)

.IsUnicode(false);

entity.Property(e => e.Password)

.HasMaxLength(10)

.IsFixedLength();

});

OnModelCreatingPartial(modelBuilder);

}

partial void OnModelCreatingPartial(ModelBuilder modelBuilder);

}

}

**StudentsController.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using CCLAB05Task2.Models;

namespace CCLAB05Task2.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class StudentsController : ControllerBase

{

private readonly EmployeeContext \_context;

public StudentsController(EmployeeContext context)

{

\_context = context;

}

// GET: api/Students

[HttpGet]

public async Task<ActionResult<IEnumerable<Student>>> GetStudents()

{

if (\_context.Students == null)

{

return NotFound();

}

return await \_context.Students.ToListAsync();

}

// GET: api/Students/5

[HttpGet("{id}")]

public async Task<ActionResult<Student>> GetStudent(int id)

{

if (\_context.Students == null)

{

return NotFound();

}

var student = await \_context.Students.FindAsync(id);

if (student == null)

{

return NotFound();

}

return student;

}

// PUT: api/Students/5

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPut("{id}")]

public async Task<IActionResult> PutStudent(int id, Student student)

{

if (id != student.Id)

{

return BadRequest();

}

\_context.Entry(student).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!StudentExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

// POST: api/Students

// To protect from overposting attacks, see https://go.microsoft.com/fwlink/?linkid=2123754

[HttpPost]

public async Task<ActionResult<Student>> PostStudent(Student student)

{

if (\_context.Students == null)

{

return Problem("Entity set 'EmployeeContext.Students' is null.");

}

\_context.Students.Add(student);

await \_context.SaveChangesAsync();

return CreatedAtAction("GetStudent", new { id = student.Id }, student);

}

// DELETE: api/Students/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteStudent(int id)

{

if (\_context.Students == null)

{

return NotFound();

}

var student = await \_context.Students.FindAsync(id);

if (student == null)

{

return NotFound();

}

\_context.Students.Remove(student);

await \_context.SaveChangesAsync();

return NoContent();

}

private bool StudentExists(int id)

{

return (\_context.Students?.Any(e => e.Id == id)).GetValueOrDefault();

}

}

}

**A group of colorful rectangular objects

Description automatically generated with medium confidence**

**A computer screen shot of a computer code

Description automatically generatedGet Request Response**

**Post Request Response**

**A screenshot of a computer code

Description automatically generated**

**A screen shot of a computer code

Description automatically generated**

**Get Request api/Students/2**

**A screen shot of a computer code

Description automatically generated**

**A screenshot of a computer

Description automatically generated**