**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 6**

по дисциплине «Введение в облачные вычисления»

на тему: «*Windows Azure Blob*»

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2020

**Цель работы:** освоить разработку *Web*-приложений, использующих в качестве хранилища бинарных файлов *Azure Blob* контейнеры.

**Задание**

Разработать *Web*-приложение, реализующее сервис для хранения фотографий, хранилищем для фотографий должны выступать *Blob* контейнеры, обеспечить хранение нескольких изображений в одном *Blob* контейнере.

**Ход работы**

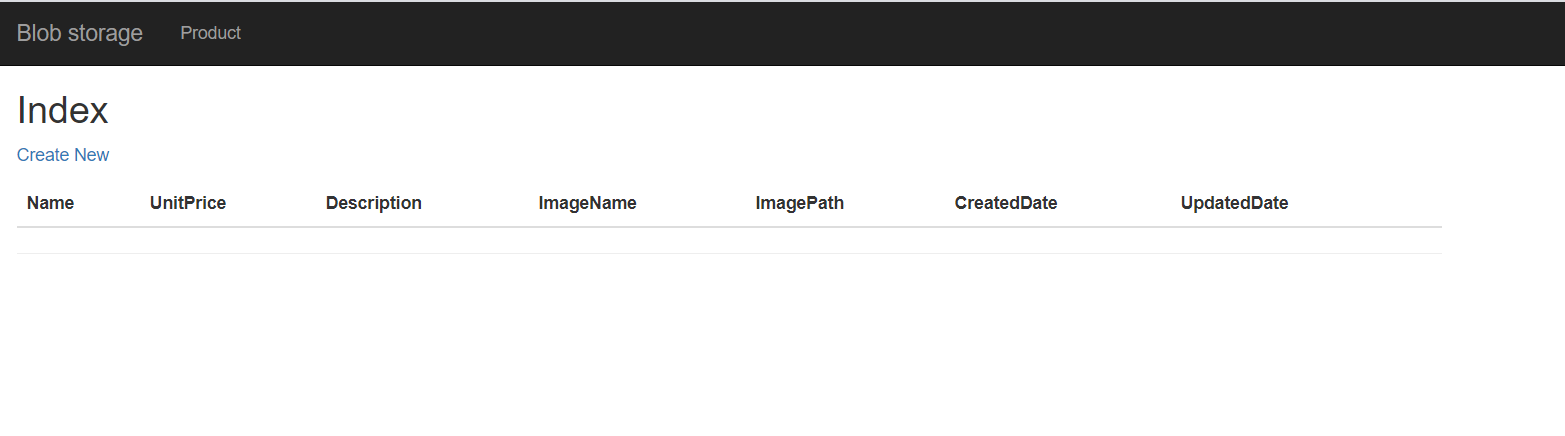


Рисунок 1 – Начальная страница *Web*-приложения

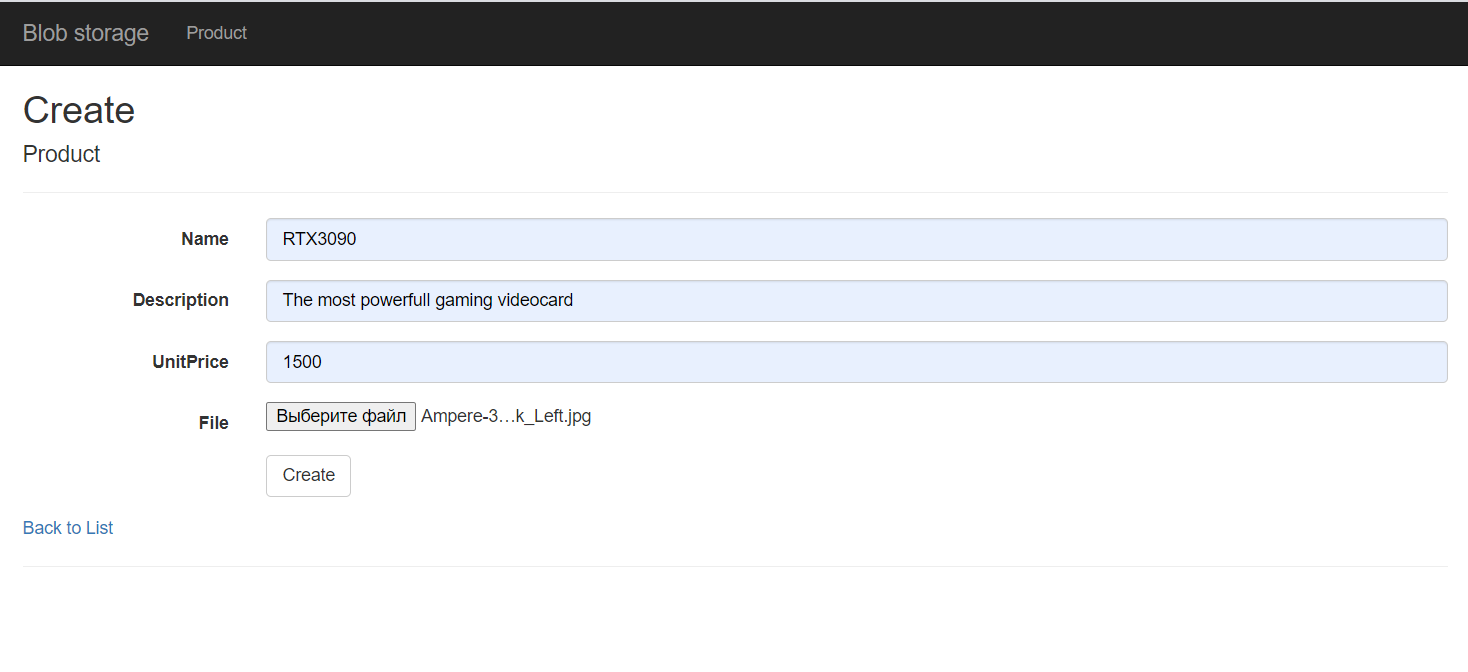


Рисунок 2 – Страница создания записи

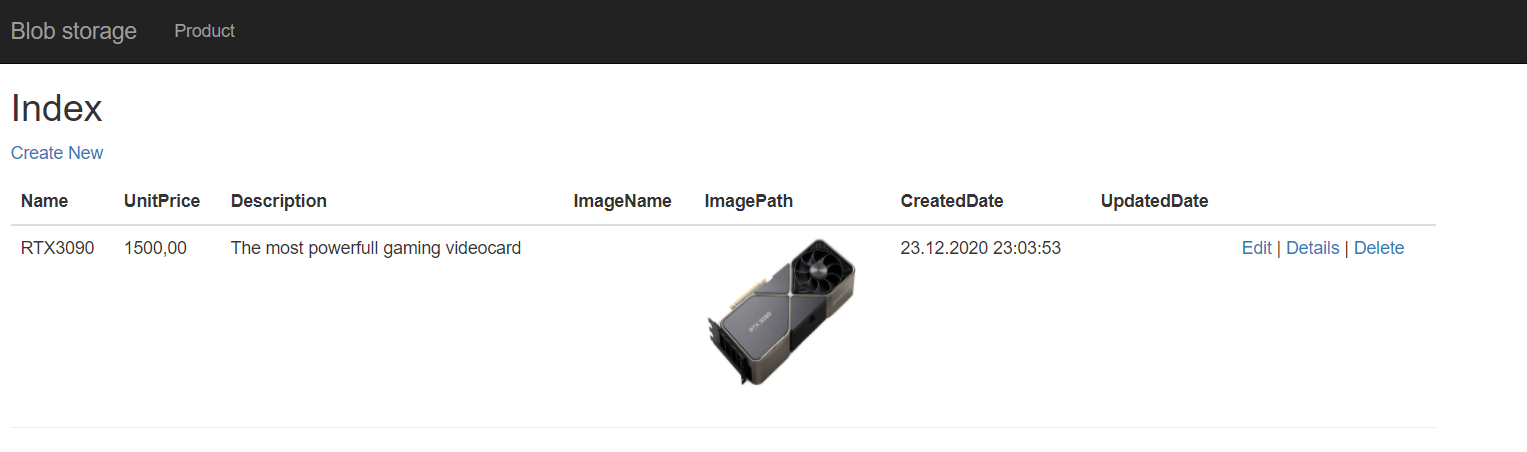


Рисунок 3 – Результат создания записи

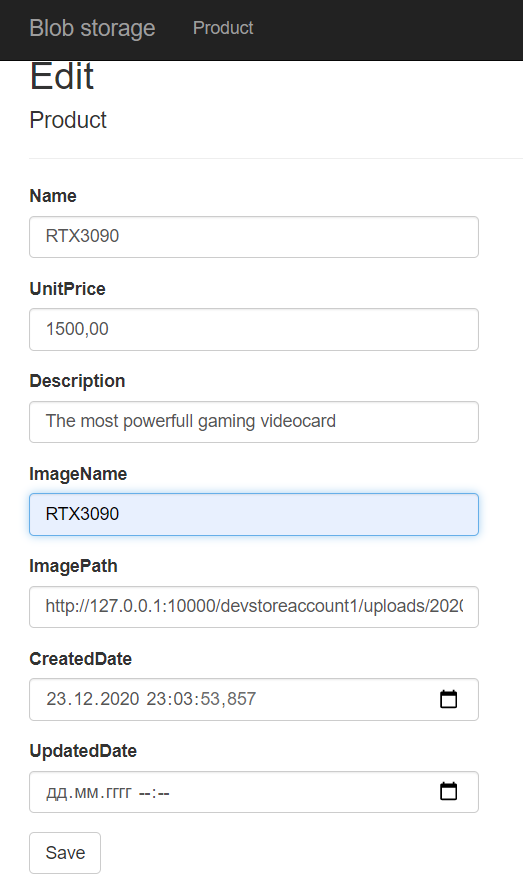


Рисунок 4 – Страница редактирования записи

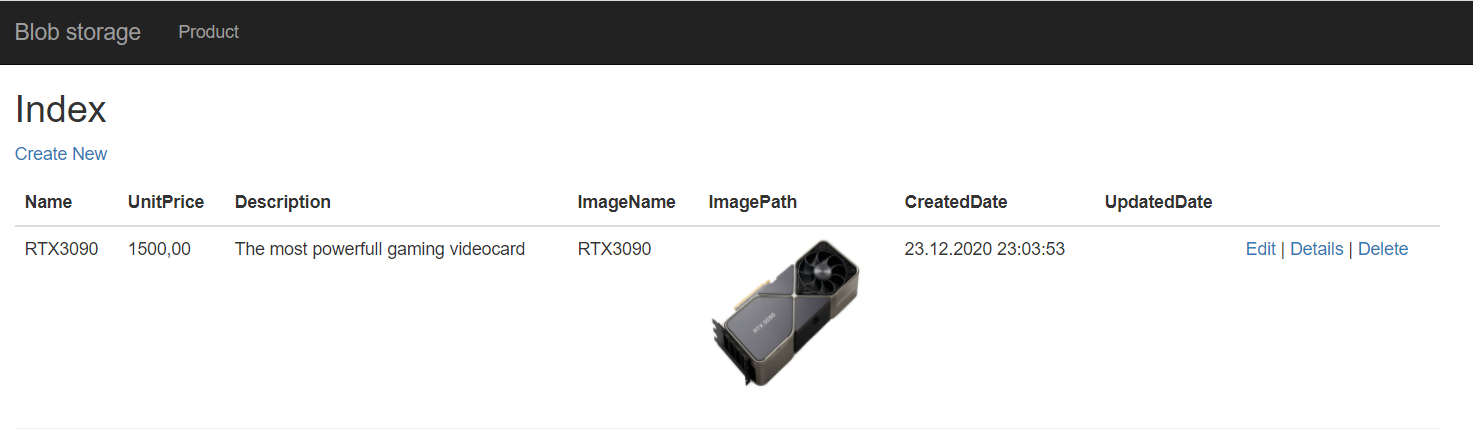


Рисунок 5 – Результат редактирования записи

**Вывод:** в результате лабораторной работы была освоена разработка *Web*-приложения, использующего в качестве хранилища бинарных файлов *Azure Blob* контейнеры.

**ПРИЛОЖЕНИЕ А**

**Листинг созданных классов**

using Microsoft.Extensions.Configuration;

using System;

using System.Collections.Generic;

using System.IO;

using System.Text;

namespace ServiceLayer.AppConfig

{

public static class AppConfiguration

{

private static IConfiguration currentConfig;

public static void SetConfig(IConfiguration configuration)

{

currentConfig = configuration;

}

public static string GetConfiguration(string configKey)

{

try

{

string connectionString = currentConfig.GetConnectionString(configKey);

return connectionString;

}

catch (Exception ex)

{

throw (ex);

}

return "";

}

}

}

using Microsoft.WindowsAzure.Storage;

using Microsoft.WindowsAzure.Storage.Blob;

using ServiceLayer.AppConfig;

using System;

using System.Collections.Generic;

using System.IO;

using System.Threading.Tasks;

namespace ServiceLayer

{

public class BlobStorageService

{

string accessKey = string.Empty;

public BlobStorageService()

{

this.accessKey = AppConfiguration.GetConfiguration("AccessKey");

}

public string UploadFileToBlob(string strFileName, byte[] fileData, string fileMimeType)

{

try

{

var \_task = Task.Run(() => this.UploadFileToBlobAsync(strFileName, fileData, fileMimeType));

\_task.Wait();

string fileUrl = \_task.Result;

return fileUrl;

}

catch (Exception ex)

{

throw (ex);

}

}

public async void DeleteBlobData(string fileUrl)

{

Uri uriObj = new Uri(fileUrl);

string BlobName = Path.GetFileName(uriObj.LocalPath);

CloudStorageAccount cloudStorageAccount = CloudStorageAccount.Parse(accessKey);

CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();

string strContainerName = "uploads";

CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(strContainerName);

string pathPrefix = DateTime.Now.ToUniversalTime().ToString("yyyy-MM-dd") + "/";

CloudBlobDirectory blobDirectory = cloudBlobContainer.GetDirectoryReference(pathPrefix);

// get block blob refarence

CloudBlockBlob blockBlob = blobDirectory.GetBlockBlobReference(BlobName);

// delete blob from container

await blockBlob.DeleteAsync();

}

private string GenerateFileName(string fileName)

{

string strFileName = string.Empty;

string[] strName = fileName.Split('.');

strFileName = DateTime.Now.ToUniversalTime().ToString("yyyy-MM-dd") + "/" + DateTime.Now.ToUniversalTime().ToString("yyyyMMdd\\THHmmssfff") + "." + strName[strName.Length - 1];

return strFileName;

}

private async Task<string> UploadFileToBlobAsync(string strFileName, byte[] fileData, string fileMimeType)

{

try

{

CloudStorageAccount cloudStorageAccount = CloudStorageAccount.Parse(accessKey);

CloudBlobClient cloudBlobClient = cloudStorageAccount.CreateCloudBlobClient();

string strContainerName = "uploads";

CloudBlobContainer cloudBlobContainer = cloudBlobClient.GetContainerReference(strContainerName);

string fileName = this.GenerateFileName(strFileName);

if (await cloudBlobContainer.CreateIfNotExistsAsync())

{

await cloudBlobContainer.SetPermissionsAsync(new BlobContainerPermissions { PublicAccess = BlobContainerPublicAccessType.Blob });

}

if (fileName != null && fileData != null)

{

CloudBlockBlob cloudBlockBlob = cloudBlobContainer.GetBlockBlobReference(fileName);

cloudBlockBlob.Properties.ContentType = fileMimeType;

await cloudBlockBlob.UploadFromByteArrayAsync(fileData, 0, fileData.Length);

return cloudBlockBlob.Uri.AbsoluteUri;

}

return "";

}

catch (Exception ex)

{

throw (ex);

}

}

}

}

using System;

using System.Collections.Generic;

using System.Text;

namespace ServiceLayer

{

public class NumberToWord

{

public string ConvertToWords(double inputValue, bool isIndianFormat = true)

{

String numb = Convert.ToString(inputValue);

String val = "", wholeNo = numb, points = "", andStr = "", pointStr = "";

String endStr = "Only";

try

{

int decimalPlace = numb.IndexOf(".");

if (decimalPlace > 0)

{

wholeNo = numb.Substring(0, decimalPlace);

points = numb.Substring(decimalPlace + 1);

if (Convert.ToInt32(points) > 0)

{

andStr = "and ";

endStr = (isIndianFormat ? "Paisa " : "Cents ") + endStr;

if (isIndianFormat)

pointStr = ConvertDecimalsIndia(points);

else

pointStr = ConvertDecimals(points);

}

}

if (isIndianFormat)

val = String.Format("{0} {1}{2} {3}", ConvertWholeNumberIndianFormat(wholeNo).Trim(), andStr, pointStr, endStr);

else

val = String.Format("{0} {1}{2} {3}", ConvertWholeNumberGlobalFormat(wholeNo).Trim(), andStr, pointStr, endStr);

}

catch { }

return val;

}

private static String ConvertWholeNumberGlobalFormat(String Number)

{

string word = "";

try

{

bool beginsZero = false;

bool isDone = false;

double dblAmt = (Convert.ToDouble(Number));

//if ((dblAmt > 0) && number.StartsWith("0"))

if (dblAmt > 0)

{//test for zero or digit zero in a nuemric

beginsZero = Number.StartsWith("0");

int numDigits = Number.Length;

int pos = 0;

String place = "";//digit grouping name:hundres,thousand,etc...

switch (numDigits)

{

case 1://ones' range

word = ones(Number);

isDone = true;

break;

case 2://tens' range

word = tens(Number);

isDone = true;

break;

case 3://hundreds' range

pos = (numDigits % 3) + 1;

place = " Hundred ";

break;

case 4://thousands' range

case 5:

case 6:

pos = (numDigits % 4) + 1;

place = " Thousand ";

break;

case 7://millions' range

case 8:

case 9:

pos = (numDigits % 7) + 1;

place = " Million ";

break;

case 10://Billions's range

case 11:

case 12:

pos = (numDigits % 10) + 1;

place = " Billion ";

break;

//add extra case options for anything above Billion...

default:

isDone = true;

break;

}

if (!isDone)

{//if transalation is not done, continue...(Recursion comes in now!!)

if (Number.Substring(0, pos) != "0" && Number.Substring(pos) != "0")

{

try

{

word = ConvertWholeNumberGlobalFormat(Number.Substring(0, pos)) + place + ConvertWholeNumberGlobalFormat(Number.Substring(pos));

}

catch { }

}

else

{

word = ConvertWholeNumberGlobalFormat(Number.Substring(0, pos)) + ConvertWholeNumberGlobalFormat(Number.Substring(pos));

}

//check for trailing zeros

//if (beginsZero) word = " and " + word.Trim();

}

//ignore digit grouping names

if (word.Trim().Equals(place.Trim())) word = "";

}

}

catch { }

return word.Trim();

}

private static String ConvertWholeNumberIndianFormat(String Number)

{

string word = "";

try

{

bool beginsZero = false;

bool isDone = false;

double dblAmt = (Convert.ToDouble(Number));

if (dblAmt > 0)

{

beginsZero = Number.StartsWith("0");

int numDigits = Number.Length;

int pos = 0;

String place = "";

switch (numDigits)

{

case 1://ones' range

word = ones(Number);

isDone = true;

break;

case 2://tens' range

word = tens(Number);

isDone = true;

break;

case 3://hundreds' range

pos = (numDigits % 3) + 1;

place = " Hundred ";

break;

case 4://thousands' range

case 5:

pos = (numDigits % 4) + 1;

place = " Thousand ";

break;

case 6://Lakh' range

case 7:

pos = (numDigits % 6) + 1;

place = " Lakh ";

break;

case 8://Crore's range

case 9:

case 10:

case 11:

case 12:

case 13:

case 14:

case 15:

pos = (numDigits % 8) + 1;

place = " Crore ";

break;

//add extra case options for anything above Billion...

default:

isDone = true;

break;

}

if (!isDone)

{//if transalation is not done, continue...(Recursion comes in now!!)

if (Number.Substring(0, pos) != "0" && Number.Substring(pos) != "0")

{

try

{

word = ConvertWholeNumberIndianFormat(Number.Substring(0, pos)) + place + ConvertWholeNumberIndianFormat(Number.Substring(pos));

}

catch { }

}

else

{

word = ConvertWholeNumberIndianFormat(Number.Substring(0, pos)) + ConvertWholeNumberIndianFormat(Number.Substring(pos));

}

//check for trailing zeros

//if (beginsZero) word = " and " + word.Trim();

}

//ignore digit grouping names

if (word.Trim().Equals(place.Trim())) word = "";

}

}

catch { }

return word.Trim();

}

private static String tens(String Number)

{

int \_Number = Convert.ToInt32(Number);

String name = null;

switch (\_Number)

{

case 10:

name = "Ten";

break;

case 11:

name = "Eleven";

break;

case 12:

name = "Twelve";

break;

case 13:

name = "Thirteen";

break;

case 14:

name = "Fourteen";

break;

case 15:

name = "Fifteen";

break;

case 16:

name = "Sixteen";

break;

case 17:

name = "Seventeen";

break;

case 18:

name = "Eighteen";

break;

case 19:

name = "Nineteen";

break;

case 20:

name = "Twenty";

break;

case 30:

name = "Thirty";

break;

case 40:

name = "Fourty";

break;

case 50:

name = "Fifty";

break;

case 60:

name = "Sixty";

break;

case 70:

name = "Seventy";

break;

case 80:

name = "Eighty";

break;

case 90:

name = "Ninety";

break;

default:

if (\_Number > 0)

{

name = tens(Number.Substring(0, 1) + "0") + " " + ones(Number.Substring(1));

}

break;

}

return name;

}

private static String ones(String Number)

{

int \_Number = Convert.ToInt32(Number);

String name = "";

switch (\_Number)

{

case 1:

name = "One";

break;

case 2:

name = "Two";

break;

case 3:

name = "Three";

break;

case 4:

name = "Four";

break;

case 5:

name = "Five";

break;

case 6:

name = "Six";

break;

case 7:

name = "Seven";

break;

case 8:

name = "Eight";

break;

case 9:

name = "Nine";

break;

}

return name;

}

private static String ConvertDecimals(String number)

{

String cd = "", digit = "", engOne = "";

for (int i = 0; i < number.Length; i++)

{

digit = number[i].ToString();

if (digit.Equals("0"))

{

engOne = "Zero";

}

else

{

engOne = ones(digit);

}

cd += " " + engOne;

}

return cd;

}

private static String ConvertDecimalsIndia(String number)

{

String engOne = "";

int numDigits = number.Length;

switch (numDigits)

{

case 1://ones' range

engOne = ones(number);

break;

case 2://tens' range

engOne = tens(number);

break;

default:

engOne = "Zero";

break;

}

return engOne;

}

}

}

using DataContext.Abstractions;

using DataContext.DataContext;

using DataContext.Implementation;

using Microsoft.EntityFrameworkCore;

namespace DataContext

{

public class UnitOfWork : IUnitOfWork

{

private DbContext db;

public UnitOfWork()

{

db = new EFDBContext();

}

private IProductRepository \_ProductRepo;

public IProductRepository ProductRepo

{

get

{

if (\_ProductRepo == null)

\_ProductRepo = new ProductRepository(db);

return \_ProductRepo;

}

}

public int SaveChanges()

{

return db.SaveChanges();

}

}

}

using DataContext;

using DataContext.Abstractions;

namespace DataContext

{

public interface IUnitOfWork

{

IProductRepository ProductRepo { get; }

int SaveChanges();

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Text;

using Microsoft.AspNetCore.Http;

namespace DataContext.Models

{

[Table("Product", Schema = "Core")]

public class Product

{

public Product()

{

CreatedDate = DateTime.Now;

}

[Key]

public int ProductId { get; set; }

[Required(ErrorMessage = "Please Enter Name")]

[Column(TypeName = "varchar(50)")]

public string Name { get; set; }

public decimal UnitPrice { get; set; }

[Required(ErrorMessage = "Please Enter Description")]

[Column(TypeName = "varchar(500)")]

public string Description { get; set; }

[Column(TypeName = "varchar(50)")]

public string ImageName { get; set; }

[Column(TypeName = "varchar(250)")]

public string ImagePath { get; set; }

public DateTime CreatedDate { get; set; }

public DateTime? UpdatedDate { get; set; }

[NotMapped]

public IFormFile File { get; set; }

}

}

using System;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

namespace DataContext.Migrations

{

public partial class t1 : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.EnsureSchema(

name: "Core");

migrationBuilder.CreateTable(

name: "Product",

schema: "Core",

columns: table => new

{

ProductId = table.Column<int>(nullable: false)

.Annotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn),

Name = table.Column<string>(type: "varchar(50)", nullable: false),

UnitPrice = table.Column<decimal>(nullable: false),

Description = table.Column<string>(type: "varchar(500)", nullable: false),

ImageName = table.Column<string>(type: "varchar(50)", nullable: true),

ImagePath = table.Column<string>(type: "varchar(250)", nullable: true),

CreatedDate = table.Column<DateTime>(nullable: false),

UpdatedDate = table.Column<DateTime>(nullable: true)

},

constraints: table =>

{

table.PrimaryKey("PK\_Product", x => x.ProductId);

});

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Product",

schema: "Core");

}

}

}

// <auto-generated />

using System;

using DataContext.DataContext;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace DataContext.Migrations

{

[DbContext(typeof(EFDBContext))]

partial class EFDBContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "2.1.4-rtm-31024")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("DataContext.Models.Product", b =>

{

b.Property<int>("ProductId")

.ValueGeneratedOnAdd()

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("CreatedDate");

b.Property<string>("Description")

.IsRequired()

.HasColumnType("varchar(500)");

b.Property<string>("ImageName")

.HasColumnType("varchar(50)");

b.Property<string>("ImagePath")

.HasColumnType("varchar(250)");

b.Property<string>("Name")

.IsRequired()

.HasColumnType("varchar(50)");

b.Property<decimal>("UnitPrice");

b.Property<DateTime?>("UpdatedDate");

b.HasKey("ProductId");

b.ToTable("Product","Core");

});

#pragma warning restore 612, 618

}

}

}

using DataContext.Abstractions;

using DataContext.DataContext;

using DataContext.Models;

using Microsoft.EntityFrameworkCore;

namespace DataContext.Implementation

{

public class ProductRepository : Repository<Product>, IProductRepository

{

private EFDBContext context

{

get

{

return db as EFDBContext;

}

}

public ProductRepository(DbContext db)

{

this.db = db;

}

}

}

using DataContext.Abstractions;

using Microsoft.EntityFrameworkCore;

using System.Collections.Generic;

using System.Linq;

namespace DataContext.Implementation

{

public class Repository<TEntity> : IRepository<TEntity> where TEntity : class

{

protected DbContext db { get; set; }

public void Add(TEntity model)

{

db.Set<TEntity>().Add(model);

}

public void Delete(TEntity model)

{

db.Set<TEntity>().Remove(model);

}

public void DeleteById(object Id)

{

TEntity entity = db.Set<TEntity>().Find(Id);

this.Delete(entity);

}

public IEnumerable<TEntity> GetAll()

{

return db.Set<TEntity>().ToList();

}

public TEntity GetById(object Id)

{

return db.Set<TEntity>().Find(Id);

}

public void Modify(TEntity model)

{

db.Entry<TEntity>(model).State = EntityState.Modified;

}

}

}

using DataContext.Models;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Text;

namespace DataContext.DataContext

{

public class EFDBContext :DbContext

{

public EFDBContext()

{

Database.EnsureCreated();

}

public EFDBContext(DbContextOptions<EFDBContext> options) : base(options)

{

Database.EnsureCreated();

}

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

protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)

{

if (!optionsBuilder.IsConfigured)

{

optionsBuilder.UseSqlServer(@"Server=(localdb)\mssqllocaldb;Database=AzureDB;Trusted\_Connection=True;MultipleActiveResultSets=true");

}

base.OnConfiguring(optionsBuilder);

}

}

}

using DataContext.Models;

namespace DataContext.Abstractions

{

public interface IProductRepository : IRepository<Product>

{

}

}

using System.Collections.Generic;

namespace DataContext.Abstractions

{

public interface IRepository<TEntity> where TEntity : class

{

void Add(TEntity model);

IEnumerable<TEntity> GetAll();

TEntity GetById(object Id);

void Modify(TEntity model);

void Delete(TEntity model);

void DeleteById(object Id);

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using DataContext;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using ServiceLayer.AppConfig;

namespace BlobStorage\_File\_Upload

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.Configure<CookiePolicyOptions>(options =>

{

// This lambda determines whether user consent for non-essential cookies is needed for a given request.

options.CheckConsentNeeded = context => true;

options.MinimumSameSitePolicy = SameSiteMode.None;

});

services.AddTransient<IUnitOfWork, UnitOfWork>();

services.AddMvc(option => option.EnableEndpointRouting = false).SetCompatibilityVersion(CompatibilityVersion.Version\_3\_0);

services.AddSingleton(\_ => Configuration);

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseCookiePolicy();

AppConfiguration.SetConfig(Configuration);

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

template: "{controller=Products}/{action=Index}/{id?}");

});

}

}

}

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Logging;

namespace BlobStorage\_File\_Upload

{

public class Program

{

public static void Main(string[] args)

{

CreateWebHostBuilder(args).Build().Run();

}

public static IWebHostBuilder CreateWebHostBuilder(string[] args) =>

WebHost.CreateDefaultBuilder(args)

.UseStartup<Startup>();

}

}

{

"Logging": {

"LogLevel": {

"Default": "Warning"

}

},

"AllowedHosts": "\*",

"BlobConnections": "",

"ConnectionStrings": {

"AccessKey": "AccountName=devstoreaccount1;AccountKey=Eby8vdM02xNOcqFlqUwJPLlmEtlCDXJ1OUzFT50uSRZ6IFsuFq2UVErCz4I6tq/K1SZFPTOtr/KBHBeksoGMGw==;DefaultEndpointsProtocol=http;BlobEndpoint=http://127.0.0.1:10000/devstoreaccount1;QueueEndpoint=http://127.0.0.1:10001/devstoreaccount1;TableEndpoint=http://127.0.0.1:10002/devstoreaccount1"

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using DataContext.DataContext;

using DataContext.Models;

using System.IO;

using Microsoft.AspNetCore.Hosting;

using DataContext;

using System.Net.Http;

using ServiceLayer;

using Microsoft.Extensions.Configuration;

namespace BlobStorage\_File\_Upload.Controllers

{

public class ProductsController : Controller

{

private UnitOfWork \_context;

IHostingEnvironment env;

private NumberToWord objNumberToWord;

public ProductsController(IHostingEnvironment \_env)

{

this.\_context = new UnitOfWork();

env = \_env;

this.objNumberToWord = new NumberToWord();

}

// GET: Products

public IActionResult Index()

{

return View(\_context.ProductRepo.GetAll());

}

// GET: Products/Details/5

public IActionResult Details(int? id)

{

if (id == null)

{

return NotFound();

}

var product = \_context.ProductRepo.GetById(id);

if (product == null)

{

return NotFound();

}

return View(product);

}

// GET: Products/Create

public IActionResult Create()

{

return View();

}

[HttpPost]

[ValidateAntiForgeryToken]

public IActionResult Create(Product product)

{

if (ModelState.IsValid)

{

#region Read File Content

var uploads = Path.Combine(env.WebRootPath, "uploads");

bool exists = Directory.Exists(uploads);

if (!exists)

Directory.CreateDirectory(uploads);

string fileName = Path.GetFileName(product.File.FileName);

byte[] fileData;

using (var target = new MemoryStream())

{

product.File.CopyTo(target);

fileData = target.ToArray();

}

//var fileStream = new FileStream(Path.Combine(uploads, product.File.FileName), FileMode.Create);

string mimeType = product.File.ContentType;

//= new byte[product.File.Length];

BlobStorageService objBlobService = new BlobStorageService();

product.ImagePath = objBlobService.UploadFileToBlob(product.File.FileName, fileData, mimeType);

#endregion

\_context.ProductRepo.Add(product);

\_context.SaveChanges();

return RedirectToAction(nameof(Index));

}

return View(product);

}

// GET: Products/Edit/5

public IActionResult Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var product = \_context.ProductRepo.GetById(id);

if (product == null)

{

return NotFound();

}

return View(product);

}

// POST: Products/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 IActionResult Edit(int id, [Bind("ProductId,Name,UnitPrice,Description,ImageName,ImagePath,CreatedDate,UpdatedDate")] Product product)

{

if (id != product.ProductId)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.ProductRepo.Modify(product);

\_context.SaveChanges();

}

catch (DbUpdateConcurrencyException)

{

if (!ProductExists(product.ProductId))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

return View(product);

}

// GET: Products/Delete/5

public IActionResult Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var product = \_context.ProductRepo.GetById(id);

if (product == null)

{

return NotFound();

}

return View(product);

}

// POST: Products/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public IActionResult DeleteConfirmed(int id)

{

var product = \_context.ProductRepo.GetById(id);

BlobStorageService objBlob = new BlobStorageService();

objBlob.DeleteBlobData(product.ImagePath);

\_context.ProductRepo.Delete(product);

\_context.SaveChanges();

return RedirectToAction(nameof(Index));

}

private bool ProductExists(int id)

{

return \_context.ProductRepo.GetAll().Any(e => e.ProductId == id);

}

}

}