**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 7**

по дисциплине «Введение в облачные вычисления»

на тему: «*Windows Azure Queue*»

Выполнил: студент гр. ИТП-31

Коркуц С. И.

Принял: преподаватель

Гуменников Е.Д.

Гомель 2020

**Цель работы:** освоить разработку *Web*-приложений, использующих для связи компонентов *Windows Azure Queue*.

**Задание**

Разработать *Web*-приложение, реализующее добавление строк в очередь *Windows Azure Queue* и обеспечивающее функционал чтения записанных строк. В случае попытки извлечения сообщения из пустой очереди приложение должно отображать соответствующее предупреждение.

**Ход работы**

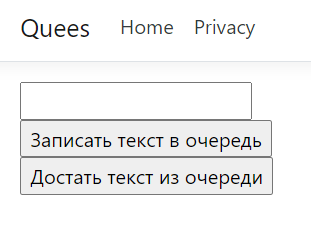


Рисунок 1 – Начальная страница *Web*-приложения

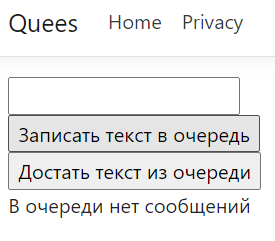


Рисунок 2 – Вывод, что в очереди нет никаких сообщений

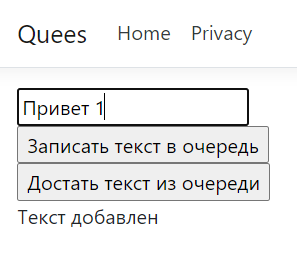


Рисунок 3 – Результат добавления сообщения в очередь

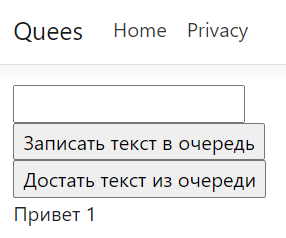


Рисунок 4 – Извлечение из очереди первого сообщения

**Вывод:** в результате лабораторной работы была освоена разработка *Web*-приложения, использующего для связи компонентов *Windows Azure Queue*.

**ПРИЛОЖЕНИЕ А**

**Листинг созданных классов**

using Microsoft.Azure.Storage;

using Microsoft.Azure.Storage.Queue;

using System;

namespace Queue\_Storage\_In\_Azure

{

public class QueueOperations

{

public const string connstring = "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;";

public const string queueName = "stringqueue";

public void AddMessage(string message)

{

CloudStorageAccount storageAccount = CloudStorageAccount.Parse(connstring);

CloudQueueClient cloudQueueClient = storageAccount.CreateCloudQueueClient();

CloudQueue cloudQueue = cloudQueueClient.GetQueueReference(queueName);

CloudQueueMessage queueMessage = new CloudQueueMessage(message);

cloudQueue.AddMessage(queueMessage);

}

public CloudQueueMessage RetrieveMessage()

{

CloudStorageAccount storageAccount = CloudStorageAccount.Parse(connstring);

CloudQueueClient cloudQueueClient = storageAccount.CreateCloudQueueClient();

CloudQueue cloudQueue = cloudQueueClient.GetQueueReference(queueName);

CloudQueueMessage queueMessage = cloudQueue.GetMessage();

Console.WriteLine(queueMessage.AsString);

cloudQueue.DeleteMessage(queueMessage);

return queueMessage;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace Quees

{

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.AddControllersWithViews();

}

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

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Logging;

namespace Quees

{

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

{

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using Quees.Models;

namespace Quees.Controllers

{

public class HomeController : Controller

{

public HomeController()

{

}

public ActionResult Index(string text)

{

ViewData["result"] = text;

return View();

}

[HttpPost]

public ActionResult UploadMessage(string text)

{

Queue\_Storage\_In\_Azure.QueueOperations operations = new Queue\_Storage\_In\_Azure.QueueOperations();

operations.AddMessage(text);

return RedirectToAction("Index", new { text = "Текст добавлен" });

}

[HttpPost]

public ActionResult DownloadMessage()

{

try

{

Queue\_Storage\_In\_Azure.QueueOperations operations = new Queue\_Storage\_In\_Azure.QueueOperations();

return RedirectToAction("Index", new { text = operations.RetrieveMessage().AsString });

}

catch(NullReferenceException)

{

return RedirectToAction("Index", new { text = "В очереди нет сообщений" });

}

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}