МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САМАРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИМЕНИ АКАДЕМИКА С.П. КОРОЛЕВА»

ФАКУЛЬТЕТ ИНФОРМАТИКИ

КАФЕДРА ПРОГРАММНЫХ СИСТЕМ

Отчет к лабораторной работе № 1

«Исследование web-сайта»

по курсу «Сети ЭВМ и телекоммуникации»

Выполнил:

Студент группы 6304-090301D

Алеев Ибрагим Ильясович

Проверил:

Пигусов Андрей Сергеевич

Цель работы: освоение принципов работы протокола прикладного уровня HTTP, приобретение навыков формирования и обработки за просов информации по протоколу HTTP.

**Задание**

Создать сетевое клиентское приложение, реализующее обращения к www-серверу по протоколу HTTP и производящее обработку полученных данных. Приложение должно выполнять функции, изложенные в задании, и обладать дружественным к пользователю интерфейсом.

Составить и вывести список всех рисунков, используемых страницами www-сервера. Вывод разделить на две части: рисунки, расположенные на сервере, и рисунки, расположенные на других серверах

**Листинг программы:**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Net;

using System.Net.Http;

using System.Net.Sockets;

using System.Security.Policy;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml;

using System.Xml.Linq;

namespace Network\_1

{

public partial class Form1 : Form

{

Uri baseUri;

List<Uri> localPages = new List<Uri>();

List<Uri> deltaPages = new List<Uri>();

List<Uri> outsidePages = new List<Uri>();

List<Uri> outsideImg = new List<Uri>();

List<Uri> imageLinks = new List<Uri>();

int max\_init = 500;

int max;

int errors = 0;

public Form1()

{

InitializeComponent();

}

private string Get(Uri uri)

{

HttpWebRequest request = (HttpWebRequest)WebRequest.Create(uri);

request.AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate;

try

{

using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())

using (Stream stream = response.GetResponseStream())

using (StreamReader reader = new StreamReader(stream))

{

return reader.ReadToEnd();

}

} catch (WebException ee)

{

return null;

}

}

private void recursion()

{

imageLinks.Clear();

localPages.Clear();

outsidePages.Clear();

int localPage\_i = 0;

max = max\_init;

errors = 0;

while (max > 0 && ((localPage\_i == 0 && localPages.Count == 0 || localPage\_i < localPages.Count) && deltaPages.Count > 0) )

{

if (deltaPages.Count == 0)

{

deltaPages.Add(localPages[localPage\_i]);

localPage\_i++;

}

for (int i = 0; i < deltaPages.Count && max > 0; i++)

{

var item = deltaPages[i];

max--;

update\_progress((int)Math.Floor(100 - max \* 100f / max\_init));

var doc\_raw = Get(item);

if (doc\_raw == null)

{

errors++;

//MessageBox.Show(item.ToString());

continue;

}

var doc\_prep = doc\_raw.Replace("\n", "").Replace("\r", "").Replace("\t"," ");

var doc = Regex.Replace(doc\_prep, @"\s\s+", " ").Replace("><", ">\n<").Replace("> <", ">\n<").Split('\n');

foreach (var div in doc)

{

Match m = Regex.Match(div, @"<img[\s:;/.a-zA-Z0-9?""=-]\*src=[""']([:/.a-zA-Zа-яА-Я0-9?=-]+)[""'][\s:;/.a-zA-Z0-9?""=-]\*/?>");

if (m.Success)

{

var extracted\_img = m.Groups[1].Value;

if (extracted\_img[0] == '/')

extracted\_img = baseUri + extracted\_img.Substring(1);

if (imageLinks.Contains(new Uri(extracted\_img)))

continue;

if (extracted\_img.Contains(baseUri.AbsoluteUri))

imageLinks.Add(new Uri(extracted\_img));

else

if (!outsideImg.Contains(new Uri(extracted\_img)))

outsideImg.Add(new Uri(extracted\_img));

continue;

}

m = Regex.Match(div, @"<a[\s:;/.a-zA-Z0-9?""=-]\*href=[""']([:/.a-zA-Z0-9?=-]+)[""'][\s:;/.a-zA-Z0-9?""=-]\*/?>");

if (!m.Success)

continue;

var extracted = m.Groups[1].Value;

if (extracted.Contains(baseUri.Host) || extracted[0] == '/' || extracted[0] == '?' || !extracted.Contains("/"))

{

if (extracted[0] == '?' || !extracted.Contains("/"))

extracted = baseUri + extracted;

if (extracted[0] == '/')

extracted = baseUri + extracted.Substring(1);

if (localPages.Contains(new Uri(extracted)))

continue;

localPages.Add(new Uri(extracted));

deltaPages.Add(new Uri(extracted));

continue;

}

if (!outsidePages.Contains(new Uri(m.Groups[1].Value)))

outsidePages.Add(new Uri(m.Groups[1].Value));

}

deltaPages.Remove(item);

}

}

if (max != 0)

update\_progress(100);

}

private void button1\_Click(object sender, EventArgs e)

{

if (!(sender as Button).Enabled)

return;

int.TryParse(maskedTextBox1.Text, out max\_init);

max = max\_init;

try

{

baseUri = new Uri(this.textBox1.Text);

}

catch (UriFormatException ee)

{

//MessageBox.Show(ee.Message, "BadURI");

throw;

}

(sender as Button).Enabled = false;

this.listBox1.Items.Clear();

this.listBox2.Items.Clear();

this.listBox3.Items.Clear();

deltaPages.Clear();

deltaPages.Add(baseUri);

var t = new Task(() => recursion());

t.Start();

}

private void update\_progress(int percentage)

{

this.Invoke((MethodInvoker)delegate {

this.Text = "delta: " + deltaPages.Count;

this.progressBar1.Value = percentage;

if (percentage >= 100)

complete();

});

}

private void complete()

{

this.Invoke((MethodInvoker)delegate {

this.listBox1.Items.AddRange(localPages.ToArray());

this.listBox2.Items.AddRange(imageLinks.ToArray());

this.listBox3.Items.AddRange(outsideImg.ToArray());

label1.Text = String.Format("Изображений: {0} Внешних: {2} Локальных: {1} Err: {3}", imageLinks.Count, localPages.Count, outsideImg.Count, errors);

this.button1.Enabled = true;

this.Invalidate();

});

}

}

}

**Результат выполнения программы:**

