**LOGENTRY klasa**

using System;

using System.Collections.Generic;

using System.Text;

namespace LogEntry

{

//4.189.158.117 - - [21/Sep/2015:07:59:14 -0400] "GET /favicon.ico HTTP/1.0" 404 621

class LogEntry

{

private string ipAddress;

private DateTime accessTime;

private string request;

private int statusCode;

private int bytesReturned;

public LogEntry(string ipAddress, DateTime accessTime, string request, int statusCode, int bytesReturned)

{

this.ipAddress = ipAddress;

this.accessTime = accessTime;

this.request = request;

this.statusCode = statusCode;

this.bytesReturned = bytesReturned;

}

//prop +TAB+TAB

public int MyProperty { get; set; }

public string IpAddres {

get

{

return ipAddress;

}

set

{

ipAddress = value;

}

}

public DateTime AccessTime

{

get

{

return accessTime;

}

set

{

accessTime = value;

}

}

public string Request {

get

{

return request;

}

set

{

request = value;

}

}

public int StatusCode {

get

{

return statusCode;

}

set

{

statusCode = value;

}

}

public int BytesReturned

{

get

{

return bytesReturned;

}

set

{

bytesReturned = value;

}

}

//4.189.158.117 - - [21/Sep/2015:07:59:14 -0400] "GET /favicon.ico HTTP/1.0" 404 621

public override string ToString()

{

return ipAddress + " " + accessTime + " " + request + " " + statusCode + " " + bytesReturned;

}

}

}

using System.Text; //za String Builder



Text

Description automatically generated

Sve ovo postavljamo u posebnu klasu.

**LOGPARSER klasa**

using System;

using System.Collections.Generic;

using System.Text;

namespace LogEntry

{

class LogParser

{

public static string Cutter(StringBuilder line, string delimiter)

{

string oneLine = line.ToString();

int x = oneLine.IndexOf(delimiter);

int length = delimiter.Length;

if (x == -1)

{

x = oneLine.Length;

length = 0;

}

string result = oneLine.Substring(0, x);

line.Remove(0, x + length);

//Console.WriteLine("Ostatak: " + line.ToString());//za testiranje i kontrolu da li izdvaja kako treba

return result;

}

public static LogEntry LogEntryParse(string line)

{

//"4.189.158.117 - - [21/Sep/2015:07:59:14 -0400] "GET /favicon.ico HTTP/1.0" 404 621"

StringBuilder logline = new StringBuilder(line);

string ip = Cutter(logline, " ");

Console.WriteLine("IP adresa: " + ip);

Cutter(logline, " ");

Cutter(logline, "[");

string dateString=Cutter(logline, " ");

Console.WriteLine("datum: " + dateString);

DateTime date = parseDate(dateString);

Console.WriteLine("datum (kao tip podataka DateTime): "+date.ToString());

Cutter(logline, " \""); //--> ovde je bila greska na casu, treba space pa znak navoda da se trazi

string request=Cutter(logline, "\"");

Console.WriteLine("request: " + request);

Cutter(logline, " ");

string statusStr=Cutter(logline, " ");

int statusInt=int.Parse(statusStr);

Console.WriteLine("status: " + statusInt);

string returnedBytes=Cutter(logline, " ");

int returnedBytesInt = int.Parse(returnedBytes);

Console.WriteLine("bytes: " + returnedBytesInt);

return new LogEntry(ip, date,request,statusInt, returnedBytesInt);

}

public static DateTime parseDate(string dateStr)

{

StringBuilder sb= new StringBuilder(dateStr);

//21/Sep/2015:07:59:14 -0400

int x=dateStr.IndexOf(":");

sb.Replace(":", " ", x, 1);

return DateTime.Parse(sb.ToString());

}

}

}

**TEXTANALYZER klasa**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

namespace LogEntry

{

class TextAnalyzer

{

private List<LogEntry> records;

public TextAnalyzer()

{

records = new List<LogEntry>();

}

public void readFile(string path)

{

List<string> lines=File.ReadAllLines(path).ToList();

foreach (string line in lines)

{

LogEntry le=LogParser.LogEntryParse(line);

records.Add(le);

}

}

public void printAll()

{

foreach(LogEntry entry in records)

{

Console.WriteLine(entry.ToString());

}

}

public int countUniqueIPs()

{

List<string> unique=new List<string>();

foreach(LogEntry entry in records)

{

if(!unique.Contains(entry.IpAddress))

unique.Add(entry.IpAddress);

}

return unique.Count();

}

public List<string> uniqueIpAddress()

{

List<string> unique=new List<string>();

foreach(LogEntry le in records)

{

if(!unique.Contains(le.IpAddress))

unique.Add(le.IpAddress);

}

return unique;

}

public Dictionary<string, int> countVisitsPerIp() {

Dictionary<string, int> counts=new Dictionary<string, int>();

foreach(LogEntry le in records)

{

string ip=le.IpAddress;

if ( !counts.ContainsKey(ip) )

{

counts.Add(ip, 1);

}

else

{

counts[ip] = ++counts[ip];

}

}

return counts;

}

//jos neke metode koje omogucavaju analizu podataka iz datoteke

//lista IP adresa za odredjeni datum koji se zadaje

public List<string> uniqueIPVisitsOnDay(string someday)

{

List<string> unique = new List<string>();

foreach (LogEntry lg in records)

{

if (lg.AccessTime.ToString().Contains(someday))

{

if (!unique.Contains(lg.IpAddres))

{

unique.Add(lg.IpAddres);

}

}

}

return unique;

}

//max broj poseta koju je ostvarila jedna IP adresa, ne zanima nas koja je to bila IP adresa, nego koji je max broj poseta

public int mostNumberVisitsByIP(Dictionary<string, int> counts)

{

int most = 0;

foreach (int v in counts.Values)

{

if (most <= v)

{

most = v;

}

}

return most;

}

//max broj poseta koju je ostvarila jedna IP adresa, ispisujemo i koja je to bila IP adresa

public string mostNumberVisitsByIP2(Dictionary<string, int> counts)

{

int most = 0;

string ip = "";

foreach (KeyValuePair<string, int> le in counts)

{

if (most <= le.Value)

{

most = le.Value;

ip = le.Key;

}

}

return "najvise poseta: "+most + ", IP: " + ip;

}

//pravi se lista IP adresa koje imaju zadati broj poseta (preko ulaznog parametra max se zadaje koliko to poseta da bude),

//npr. da se napravi lista IP adresa koje imaju 3 posete

public List<string> iPsMostVisits(Dictionary<string, int> counts,

int max)

{

List<string> mostVisit = new List<string>();

foreach (string s in counts.Keys)

{

if (counts[s] == max)

{

mostVisit.Add(s);

}

}

return mostVisit;

}

}

}

**PROGRAM.CS**

using System;

using System.Collections.Generic;

using System.Text;

namespace LogEntry

{

class Program

{

static void Main(string[] args)

{

/\* StringBuilder sb = new StringBuilder("4.189.158.117 - - [21/Sep/2015:07:59:14 -0400] \"GET / favicon.ico HTTP / 1.0\" 404 621");

string str = "4.189.158.117 - - [21/Sep/2015:07:59:14 -0400] \"GET / favicon.ico HTTP / 1.0\" 404 621";

LogParser.parseLine(str);

\*/

string path = @"c:\tmp\logfajl.txt";

TextAnalyzer textAnalyzer = new TextAnalyzer();

textAnalyzer.readFile(path);

textAnalyzer.printAll();

Console.WriteLine("\n\nBroj jedinstvenih IP adresa: " + textAnalyzer.countUniqueIPs());

Console.WriteLine("\nLista jedinstvenih IP adresa: ");

List<string> list1 = textAnalyzer.uniqueIpAddresses();

foreach (string s in list1)

{

Console.WriteLine(s);

}

Console.WriteLine("\n\nBroj pojavljivanja svake IP adrese: ");

Dictionary<string, int> count = textAnalyzer.countVisitsPerIP();

foreach (KeyValuePair<string, int> kvp in count)

{

Console.WriteLine("Key (IP): {0}, Value: {1}", kvp.Key, kvp.Value);

}

Console.WriteLine("\nLista IP adresa za odredjeni datum: ");

List<string> list2 = textAnalyzer.uniqueIPVisitsOnDay("30-Sep-15");

foreach (string s in list2)

{

Console.WriteLine(s);

}

Console.WriteLine("\n\nNajvise poseta: " + textAnalyzer.mostNumberVisitsByIP(count));

Console.WriteLine("\n\nNajvise poseta i koja je to IP: ");

Console.WriteLine(textAnalyzer.mostNumberVisitsByIP2(count));

//zadajemo koliko poseta treba da ima IP adresa da bi je ispisali, npr. 3

Console.WriteLine("\nLista IP adresa koje imaju 3 posete: ");

List<string> list3 = textAnalyzer.iPsMostVisits(count, 3);

foreach (string s in list3)

{

Console.WriteLine(s);

}

}

}

}