**LogParser**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Vezbe8novembar

{

internal class LogParser

{

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

public static string cutter(StringBuilder line, string delim)

{

string oneLine = line.ToString();

int x = oneLine.IndexOf(delim);

int length = delim.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());

return result;

}

// 621

public static LogEntry parseLine(string line)

{

StringBuilder logLine = new StringBuilder(line);

Console.WriteLine(line);

string ip = cutter(logLine, " ");

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

cutter(logLine, " "); //brisemo deo podataka koji nam ne treba

cutter(logLine, "[");

string dateString = cutter(logLine, " ");

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

DateTime date=parseDate(dateString);

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

cutter(logLine, " \"");

string request = cutter(logLine, "\"");

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

cutter(logLine, " ");

string statusStr= cutter(logLine, " ");

int status=int.Parse(statusStr);

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

string returnedBytesStr=cutter(logLine, " ");

int returnedBytesInt=int.Parse(returnedBytesStr);

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

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

}

private static DateTime parseDate(string dateString)

{

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

StringBuilder date = new StringBuilder(dateString);

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

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

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

}

}

}

**TextAnalyzer**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Vezbe8novembar

{

internal 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.parseLine(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.IpAddress))

{

unique.Add(lg.IpAddress);

}

}

}

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;

}

}

}

**Glavni program**

using System.IO;

namespace Vezbe8novembar

{

internal class Program

{

static void Main(string[] args)

{

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

/\*

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

LogParser.parseLine(line);

\*/

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.uniqueIpAddress();

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);

}

}

}

}