**Федеральное агентство связи**

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования   
«Московский технический университет связи и информатики»

Кафедра «МКИиТ»

Лабораторная работа №7

по дисциплине «Кроссплатформенные технологии программирования»

Выполнил: студент

группы БСТ1801

Скоморохов Виктор

Вариант 21

Москва 2020

**1. Задание на разработку программы**

В данной лабораторной работе нужно реализовать веб-сканер.

**2. Ход выполнения работы**

2.1)Crawler.java

import java.io.BufferedReader;

import java.io.IOException;

import java.net.MalformedURLException;

import java.net.UnknownHostException;

import java.sql.SQLException;

import java.util.HashMap;

import java.util.HashSet;

import java.util.LinkedList;

import java.util.concurrent.LinkedBlockingDeque;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

public class Crawler

{

static final String *CONST1* = "(href=\").\*?\"";

static final String *CONST2* = "(<a).\*?>";

private static final Pattern *TAG\_A\_PATTEN\_COMPILE* = Pattern.*compile*(*CONST2*);

private static final Pattern *CONST1\_COMPILE* = Pattern.*compile*(*CONST1*);

private HashSet<String> perem2 = new HashSet<String>();

private HashSet<String> allDomain = new HashSet<String>();

private LinkedBlockingDeque <URLDepthPair> unCheckedList = new LinkedBlockingDeque <URLDepthPair>();

private LinkedList <URLDepthPair> perem1 = new LinkedList <URLDepthPair>();

private LinkedList <URLDepthPair> noneCheckedList = new LinkedList <URLDepthPair>();

private int maxDepth;

public Crawler(String url, int maxDepth) throws UnknownHostException, IOException, ClassNotFoundException, SQLException

{

URLDepthPair urlpair = new URLDepthPair(url,0);

unCheckedList.add(urlpair);

perem2.add(urlpair.url);

allDomain.add(urlpair.domain);

this.maxDepth = maxDepth;

}

public void startCrawl() throws IOException, InterruptedException

{

while(!unCheckedList.isEmpty())

{

URLDepthPair urlpair = unCheckedList.pollFirst();

if (urlpair.depth<=maxDepth)

{

if(readURL(urlpair)) perem1.addLast(urlpair);

}

else noneCheckedList.add(urlpair);

}

}

private boolean readURL(URLDepthPair urlpair) throws IOException

{

System.*out*.println(urlpair.url);

SocketConnection conn = new SocketConnection(urlpair);

if (!conn.connect()) return false;

conn.sendGET();

if (conn.code.equals("200"))

{

String line;

BufferedReader br = conn.br;

while (!(line = br.readLine()).equals("0"))

{

ParseNewURL(line,urlpair);

}

conn.close();

return true;

}

else return false;

}

private boolean ParseNewURL(String line, URLDepthPair url)

{

try

{

Matcher m = *TAG\_A\_PATTEN\_COMPILE*.matcher(line);

m.find();

String tagA = line.substring(m.start(),m.end());

Matcher m2 = *CONST1\_COMPILE*.matcher(tagA);

m2.find();

String href = tagA.substring(m2.start(),m2.end());

URLDepthPair newURL = new URLDepthPair(href,url.depth+1,url.domain);

if (!perem2.contains(newURL.url))

{

perem2.add(newURL.url);

if (allDomain.contains(newURL.domain)) unCheckedList.addLast(newURL);

else

{

unCheckedList.addFirst(newURL);

allDomain.add(newURL.domain);

}

return true;

}

return false;

}

catch (IllegalStateException e)

{

return false;

}

catch (MalformedURLException e)

{

return false;

}

}

public LinkedList <URLDepthPair> getÑheckedSites()

{

return perem1;

}

public LinkedList <URLDepthPair> getUnÑheckedSites()

{

return noneCheckedList;

}

}

2.2)SocketConnection.java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.HashMap;

import java.util.Map;

public class SocketConnection

{

public String code;

private final String PROTOCOL = "HTTP/1.1";

private final int READ\_TIMEOUT = 20000;

private final int CONNECTION\_PORT = 80;

public PrintWriter pw;

public BufferedReader br;

public String domain;

public String path;

private Socket sock;

public HashMap<String, String> request;

public SocketConnection(URLDepthPair url)

{

this.domain = url.domain;

this.path = url.path;

}

public SocketConnection(String domain, String path)

{

this.domain = domain;

this.path = path;

}

public SocketConnection(String domain)

{

this.domain = domain;

this.path = "/";

}

public boolean connect()

{

try

{

this.sock = new Socket(domain,CONNECTION\_PORT);

sock.setSoTimeout(READ\_TIMEOUT);

OutputStream os = sock.getOutputStream();

this.pw = new PrintWriter(os,true);

InputStream is = sock.getInputStream();

InputStreamReader in = new InputStreamReader(is);

this.br = new BufferedReader(in);

return true;

}

catch (UnknownHostException e)

{

return false;

}

catch (IOException e)

{

return false;

}

}

public void sendGET () throws IOException

{

pw.println("GET "+path+" " + PROTOCOL);

pw.println("Host: "+ domain);

pw.println("");

getCode();

}

private void getCode() throws IOException

{

String line = br.readLine();

code = line.substring(PROTOCOL.length()+1,PROTOCOL.length()+4);

while (!(line = br.readLine()).equals(""));

}

public void close() throws IOException

{

this.sock.close();

}

}

2.3)URLDepthPair.java

import java.net.MalformedURLException;

import java.util.regex.\*;

public class URLDepthPair {

public static final int *START\_HREF* = 6;

public static final int *HREF\_TOEND* = 1;

public static final String *HTML\_EXTENSION* = ".html";

public static final String *URL\_PREFIX\_HTTP* = "http://";

public static final String *PREFIX\_PATTERN* = "(http://)";

public static final String *CONST1* = "([\\da-zа-я\\.-]+)\\.([a-zа-я\\.]{2,6})";

public static final String *CONST2* = "([/\\wа-я\\.-]\*)\*\\/?";

public static final String *CONST3* = "\\.[\\wа-я\\-]+$";

public static final String *URL\_PATTERN* = *PREFIX\_PATTERN*+*CONST1*+*CONST2*;

public String url;

public String prefix;

public String domain;

public String path;

public int depth;

public URLDepthPair(String url, int depth) throws MalformedURLException {

if (testURL(url)) this.url= url;

else this.url = searchPattern(url,*URL\_PATTERN*);

Pattern p = Pattern.*compile*(*CONST1*);

Matcher m = p.matcher(this.url);

m.find();

this.domain = this.url.substring(m.start(),m.end());

this.path = this.url.substring(m.end());

if (this.path.isEmpty()) this.path = "/";

this.depth = depth;

checkExtension();

}

public URLDepthPair(String href, int depth, String domain) throws MalformedURLException {

String hrefContent = href.substring(*START\_HREF*,href.length()-*HREF\_TOEND*);

try {

this.url = searchPattern(hrefContent,*URL\_PATTERN*);

Pattern p = Pattern.*compile*(*CONST1*);

Matcher m = p.matcher(this.url);

m.find();

this.domain = this.url.substring(m.start(),m.end());

this.path = this.url.substring(m.end());

if (this.path.isEmpty()) this.path = "/";

this.depth = depth;

checkExtension();

}

catch (MalformedURLException e) {

if (!Pattern.*matches*(*CONST1*,domain)) throw new MalformedURLException("Wrong URL");

hrefContent = searchPattern(hrefContent,*CONST2*);

if (hrefContent.isEmpty()) throw new MalformedURLException("Wrong URL");

if (hrefContent.charAt(0) != '/') hrefContent = "/"+hrefContent;

this.url = *URL\_PREFIX\_HTTP*+domain+hrefContent;

this.domain = domain;

this.path = hrefContent;

this.depth = depth;

checkExtension();

}

}

public boolean testURL(String url) {

return Pattern.*matches*(*URL\_PATTERN*, url);

}

public String searchPattern(String url,String pattern) throws MalformedURLException {

Pattern p = Pattern.*compile*(pattern);

Matcher m = p.matcher(url);

try {

m.find();

return url.substring(m.start(),m.end());

}

catch (IllegalStateException e) {

throw new MalformedURLException("Wrong URL");

}

}

private void checkExtension() throws MalformedURLException {

String extension = null;

try {

extension = searchPattern(path,*CONST3*);

}

catch (MalformedURLException e) {

extension = *HTML\_EXTENSION*;

}

finally {

if (!extension.equals(*HTML\_EXTENSION*)) throw new MalformedURLException("Wrong extension");

}

}

@Override

public int hashCode() {

return url.hashCode();

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (getClass() != obj.getClass())

return false;

URLDepthPair other = (URLDepthPair) obj;

if (url.equals(other.url)) return true;

else return false;

}

}

2.4)Scan.java

import java.io.\*;

import java.net.UnknownHostException;

import java.sql.SQLException;

public class Scan {

public static void main(String[] args) throws UnknownHostException, IOException, ClassNotFoundException, SQLException, InterruptedException

{

String test = "http://www.gov.ru/";

Crawler crawl = new Crawler(test, 1);

crawl.startCrawl();

}

}

**3. Результат работы программы**

Скриншот сайта для сканирования



Скриншот работы программы

