**Федеральное агентство связи**

Ордена Трудового Красного Знамени

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

Кафедра «МКИиТ»

Лабораторная работа №8

по дисциплине «Кроссплатформенные технологии программирования»

Выполнил: студент

группы БСТ1801

Скоморохов Виктор

Вариант 21

Москва 2020

**1. Задание на разработку программы**

В данной лабораторной работе я модифицировал веб-сканер из 7 лабораторной работы

**2. Код программы**

2.1)Crawler.java

import java.io.\*;

import java.net.\*;

import java.util.regex.\*;

import java.sql.SQLException;

public class Crawler {

public URLPool urlPool;

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 int depth;

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

if (args.length != 2) {

System.*out*.println("Bad format\nYou should start app like this:\njava Crawler <URL> <MAX\_DEPTH>");

return;

}

String url = args[0];

Integer depth = Integer.*parseInt*(args[1]);

Crawler crawl = new Crawler(url, depth);

crawl.startCrawl();

crawl.met8(10);

}

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

URLDepthPair urlpair = new URLDepthPair(url,0);

urlPool = new URLPool();

urlPool.addURL(urlpair);

this.depth = depth;

}

public class CrawlerTask implements Runnable {

public void run() {

while(true) {

try {

startCrawl();

} catch (IOException e) {

e.printStackTrace();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public void met8(int n) throws ClassNotFoundException {

for(int i = 0; i < n; i++) {

CrawlerTask run = new CrawlerTask();

Thread myThread = new Thread(run,"Thread" + i);

myThread.start();

}

while(true) {

try {

Thread.*sleep*(1000);

} catch (InterruptedException e) {

e.printStackTrace();

}

if (urlPool.COUNT == n) {

System.*exit*(0);

}

}

}

private void startCrawl() throws IOException, InterruptedException {

URLDepthPair urlpair = urlPool.getURL();

if (urlpair.depth <= depth) {

if(readURL(urlpair)) urlPool.addCheckedURL(urlpair);

}

else urlPool.addNoneCheckedURL(urlpair);

}

@SuppressWarnings("finally")

private boolean readURL(URLDepthPair urlpair) throws IOException, SocketTimeoutException {

SocketConnection conn = new SocketConnection(urlpair);

try {

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

conn.sendGET();

if (conn.checkConnection()) {

String line;

BufferedReader buffReader = conn.buffReader;

while ((line = buffReader.readLine()) != null) {

ParseNewURL(line,urlpair);

}

System.*out*.printf("New website found! %s\n", urlpair.url);

return true;

}

else return false;

}

catch (SocketTimeoutException exception) {

return false;

}

finally {

conn.close();

}

}

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);

urlPool.addURL(newURL);

return true;

}

catch (Exception e) {

return false;

}

}

}

2.2)SocketConnection.java

import java.io.\*;

import java.net.\*;

import java.util.HashMap;

public class SocketConnection {

public String code;

public String domain;

public String path;

private Socket sock;

private final String PROTOCOL = "HTTP/1.1";

private final int READ\_TIMEOUT = 20000;

private final int CONNECTION\_PORT = 80;

public PrintWriter pWriter;

public BufferedReader buffReader;

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.pWriter = new PrintWriter(os,true);

InputStream is = sock.getInputStream();

InputStreamReader in = new InputStreamReader(is);

this.buffReader = new BufferedReader(in);

return true;

}

catch (UnknownHostException e) {

return false;

}

catch (IOException e) {

return false;

}

}

public void sendGET () throws IOException {

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

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

pWriter.println("");

getCode();

}

private void getCode() throws IOException {

String line = buffReader.readLine();

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

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

}

public boolean checkConnection() throws IOException {

try {

String line = buffReader.readLine();

if (this.code.length() == 0) code = line.substring(PROTOCOL.length()+1,PROTOCOL.length()+4);

while ((line = buffReader.readLine()).length() > 0);

if (Integer.*parseInt*(code) == 200) {

return true;

}

return false;

}

catch (Exception e) {

return false;

}

}

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)URLPool.java

import java.util.HashSet;

import java.util.LinkedList;

public class URLPool {

public int COUNT = 0;

public HashSet<String> allURL = new HashSet<String>();

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

public LinkedList <URLDepthPair> unVisitedList = new LinkedList <URLDepthPair>();

public LinkedList <URLDepthPair> visitedList = new LinkedList <URLDepthPair>();

LinkedList <URLDepthPair> noneVisitedList = new LinkedList <URLDepthPair>();

public void addNoneCheckedURL (URLDepthPair urlpair) {

noneVisitedList.addLast(urlpair);

}

public void addCheckedURL(URLDepthPair urlpair) {

visitedList.addLast(urlpair);

}

public synchronized boolean addURL(URLDepthPair urlpair){

if (allURL.contains(urlpair.url)) return false;

else {

allURL.add(urlpair.url);

if (allDomain.contains(urlpair.domain)) unVisitedList.addLast(urlpair);

else {

unVisitedList.addFirst(urlpair);

allDomain.add(urlpair.domain);

}

this.notify();

return true;

}

}

public synchronized URLDepthPair getURL() throws InterruptedException {

if (unVisitedList.isEmpty()) {

COUNT++;

this.wait();

COUNT--;

return unVisitedList.pollFirst();

}

return unVisitedList.pollFirst();

}

}

**3. Результат работы программы**

Скриншот сайта



Скриншот работы программы

