LinkTracker

Data pulling from websites

Sk. Ekramul Islam (Pavel)

**Introduction:**

LinkTracker Application is used for tracking URL links from websites source. Now a day it is an important for business Organization, research center and various data center. LinkTracker can track easily all the URL links from HTML code.

**Description:**

LinkTracker is a Java platform-based application. The application takes only a web URL link, where assigned many information and other important web address links. If we want to get all the information manually, should open new tab on the browser, copy the data and save them. It is so tough for large scale website. But we can get them easily, just provide a URL link and get all the important data.

**Methodology:**

LinkTracker will take an input from users. Then will connect to the for downloading the HTML code from website. The code will be saved. Then parses HTML code for sorting URL link, which is expected. After that the shorting data will be processed by various searching stage. For this kind of process, it will be integrated with API and run automatically. Then the searching option read the website for pulling data and store them in a file.

**Working Approach:**

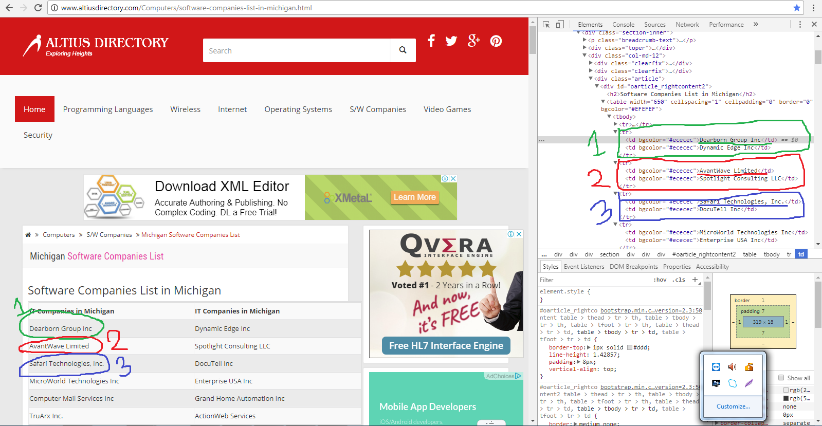
For tracking data, very first the HTML source code is needed of the website. So, readfromWeb function is developed for downloading the HTML code. Where the expecting data is assigned. The data is assigned in html code by various Tags. Like as, <H2>, <title>, <a>, <div class =” “> etc. When want to take the data, have got but it is only for some of websites. The problem is HTML code pattern. The pattern is different from each other. As a result, the code will be customized for each website.

Figure: 1 (http://www.altiusdirectory.com)

On the above screen shot the target data is assigned into a table, <tr>, <td> tags.

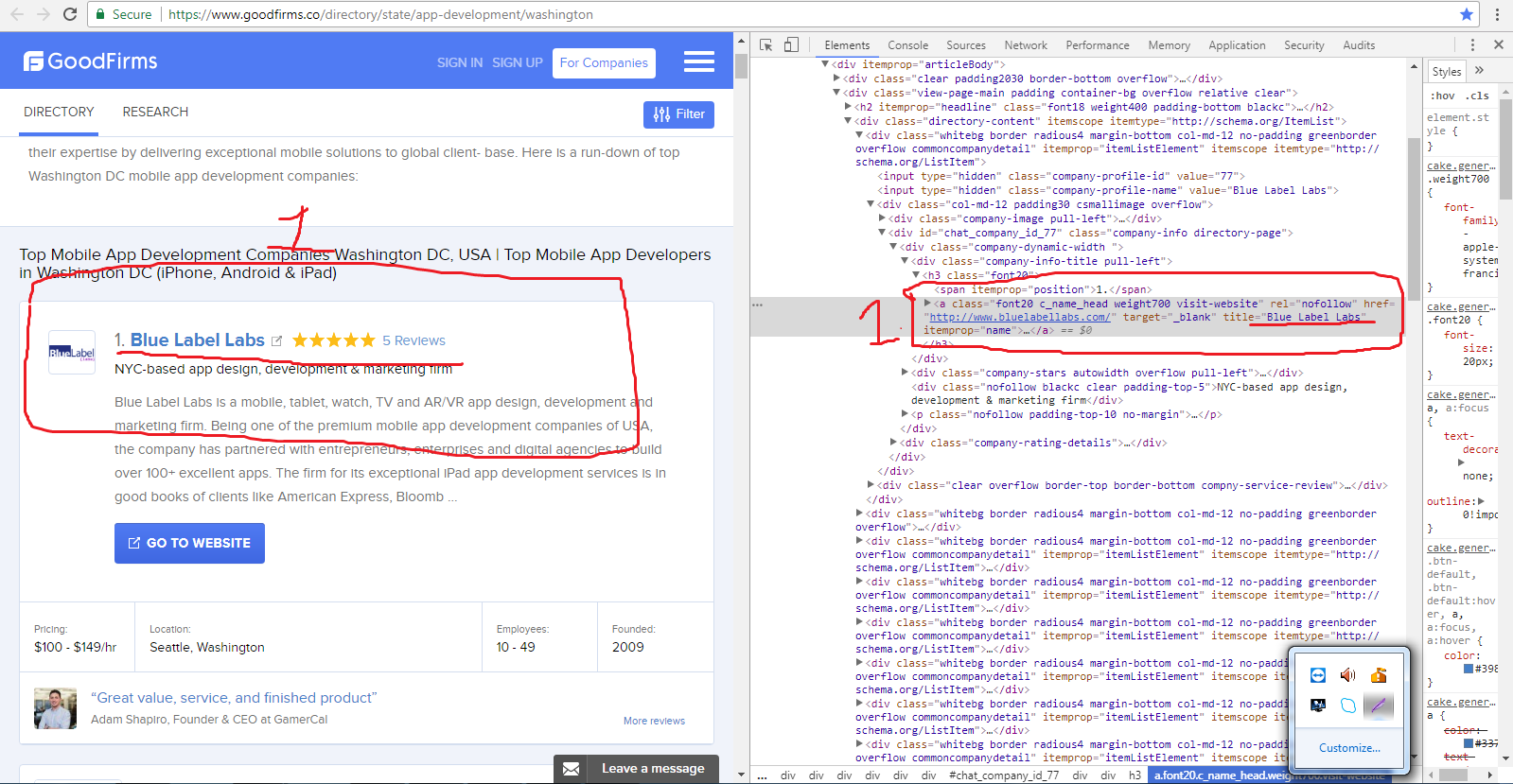


Figure: 2 (www.gust.com)

And for the website, target data is assigned into anchor tag.

In this case the LinkTracker cannot detect the target data and program output and expecting output not similar.

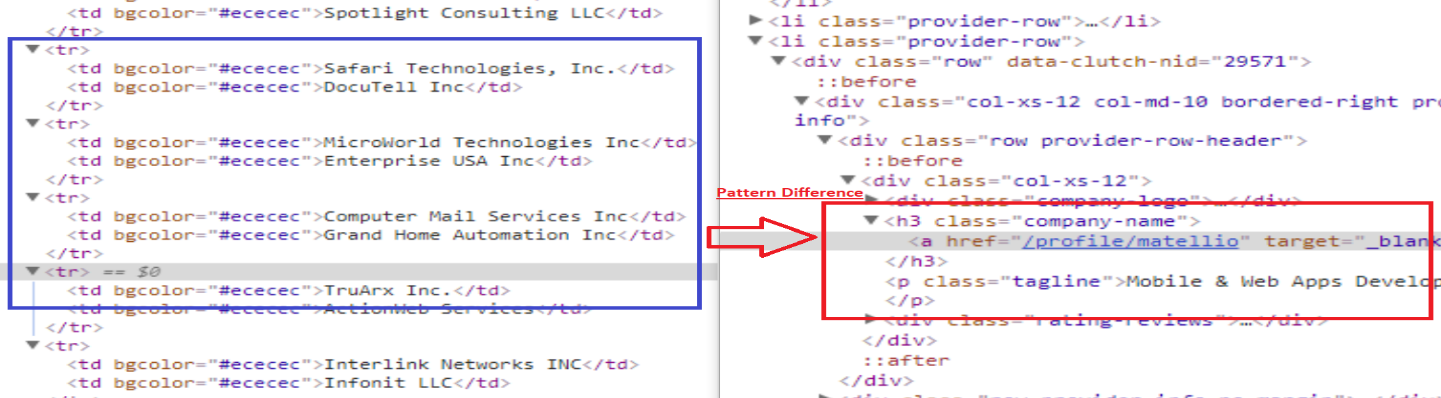
On the other hand, there are also many directory websites which has the company name with website link. The ratio is probably 5:1. So, if we take the [ http:\\(\*)] link, we will cover maximum target. After getting source code, can parse links from source code by regular expression and sort the links.

**Future Work:**

The storage links are not always viable. We should browse the website links and shorted them about the target company. For this purpose, have to need search engine and atomically done the process it will be linked up the search engine’s API and then it may will be process automatically.

**Limitation:**

There is some limitation for all these good works. LinkTracker also has some limitations. First limitation is, when the program is run the internet should must be active. Some difficulties are faced when the websites are enable SSL or other security module.

There are also another kind of problem faced, the websites source code pattern is not same. In this case LinkTracker don’t work properly. And Last part of my task is integrated with LinkedIn search option and it will be processed automatically, but unfortunately LinkedIn don’t provide free API for all the users. They provide that the paid account holders and have some rules and regulation.

**Conclusion:**

The Program is very helpful for Business Organization or research center. It is time consuming and efficient. It may also reduce researcher hassle and mistakes.

**References:**

1. Book of Crawling the web with Java by James Holmes.
2. <https://en.wikipedia.org/wiki/Web_crawler>
3. Video Tutorial from YouTube, Udemy.
4. <https://www.geeksforgeeks.org/google-search-works/>

**Code:**

**NamefromWeb .java–**

package linktracker;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

//Read Name from web source.....................

public class NamefromWeb {

public static void main(String[] args) throws Exception {

try {

// Connection with the browser...

Document doc = Jsoup.connect("https://www.goodfirms.co/directory/country/top-software-development-companies/us").get();

Elements tmp = doc.select("div.col-md-12 padding30 csmallimage overflow");

int i = 0;

// For class type div col-md-12 padding30 csmallimage overflow

for (Element companylist : tmp) {

i++;

System.err.println(i + " " + companylist.getElementsByTag("a.title").first().text());

}

} catch (Exception e) {

}

}

}

**LinkTracker.java -**

package linktracker;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.Reader;

import java.net.URI;

import java.net.URISyntaxException;

import java.net.URL;

import java.net.URLConnection;

import java.util.Scanner;

import javax.swing.text.BadLocationException;

import javax.swing.text.EditorKit;

import javax.swing.text.html.HTMLDocument;

import javax.swing.text.html.HTMLEditorKit;

import org.jsoup.Jsoup;

import org.jsoup.nodes.Document;

import org.jsoup.nodes.Element;

import org.jsoup.select.Elements;

public class LinkTracker {

public static void main(String[] args) throws URISyntaxException,IOException, BadLocationException

{

Scanner scan = new Scanner(System.in);

System.err.println("Ente the Urk link below...");

String s = scan.nextLine();

HTMLDocument doc = new HTMLDocument() {

public HTMLEditorKit.ParserCallback getReader(int pos) {

return new HTMLEditorKit.ParserCallback() {

public void handleText(char[] data, int pos) {

System.out.println(data);

}

};

}

};

// Define the url link for crawling website.....

URL url = new URI(s).toURL();

URLConnection conn = url.openConnection();

Reader rd = new InputStreamReader(conn.getInputStream());

//save the links in link.txt file

OutputStreamWriter writer = new OutputStreamWriter(new FileOutputStream("link.txt"), "UTF-8");

EditorKit kit = new HTMLEditorKit();

kit.read(rd, doc, 0);

try {

Document docs = Jsoup.connect(s).get();

Elements links = docs.select("a[href]");

Elements elements = docs.select("\*");

System.out.println("Total Links :" + links.size());

for (Element element : elements) {

System.out.println(element.ownText());

}

//links are assinged into link and check them with write line by line

for (Element link : links) {

String hrefUrl = link.attr("href");

if (!"#".equals(hrefUrl) && !(hrefUrl).isEmpty()) {

System.out.println(hrefUrl);

writer.write(link.text() + " => \n" + hrefUrl + "\n");

}

}

} catch (Exception e) {

} finally {

writer.close();

}

}

}