Day 4

In day 4 session we will make an HTTP request and parse the page . after that Return all the links on the page as well as count the user given word on page”

Task Of Day

I. Implementation of Url Crawler
II. Logics
   A. Get Page from give Url without using any extra library
   B. Read Pages line by line
      Get all available links on page without using any extra library
   C. find user given word on page and store it to display on terminal .

Okay, so this second class (URLCrawler.java) was supposed to do three things:

1. Crawl the page (make an HTTP request and parse the page)
2. Search for a word
3. Return all the links on the page

Story 1 - Implementation of URLCrawler file

1- Create “URLCrawler.java” file into “processors” folder

2 - make Crawl method into the page for (make an HTTP request and parse the page)

```
package com.ncu.processors;
import java.util.List;
import java.net.*;
import java.io.InputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.File;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.util.LinkedList;
import java.util.Date;
import java.util.Properties;
import org.apache.log4j.Logger;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;
public class URLCrawler{
  // Storing Url in list .
  private List<String> links = new LinkedList<String>();
  String urlToSearch,searchWord,getUrl;
  Logger logger;
  JSONObject jsonObject;
  JSONArray jsonArryObject;
  Date beforeStart,afterStart;
  long diffInMilisecond;
  int maxCount=0;
  String recomendUrl;
  int maxGetWord;
  Properties message,constants;
  URLCrawler crawlerObject;
}
```

Description -

```
private List<String> links = new LinkedList<String>();
```

“We are using this line to store all available links on page”

3- create constructor to initialization of logger , configMessage and configConstants files

“Add below code into “URLCrawler” class

```
public URLCrawler(){
    // initialization of GetLogger to get logger object
    GetLogger loggerObject=new GetLogger();
    Logger logger=loggerObject.loggerValue("SearchText");
    this.logger=logger;

    // initialization of GetLogger to get logger object
    JSONArray jsonArryObject = new JSONArray();
    this.jsonArryObject=jsonArryObject;

    // initialization of configMessage to get messages
    GetConfiguration propertyObject=new GetConfiguration();
    Properties message=propertyObject.configMessages();
    this.message=message;

    // calling configConstants method to get constant values
    Properties constants=propertyObject.configConstants();
    this.constants=constants;
  }
```

4- create JSONObject class object to add information into json object and and this information we will store into json file in our further videos .

“Add below code into “crawl” method of URLCrawler class .

```
JSONObject jsonObject = new JSONObject();
   this.jsonObject=jsonObject;
   this.searchWord=searchWord;
```

```
   this.urlToSearch=urlToSearch;
   URLCrawler crawlerObject = new URLCrawler();
   this.crawlerObject=crawlerObject;
   InputStream streamObject = null;
   BufferedReader readerObject;
   String line;
   int linkCount=0;
   int lastIndex = 0;
   int wordCount = 0;
```

5- Creating object of URL class and pass current url which we want to reatrive

```
public void crawl(String urlToSearch,String searchWord)
  {
    JSONObject jsonObject = new JSONObject();
    this.jsonObject=jsonObject;
    this.searchWord=searchWord;
    this.urlToSearch=urlToSearch;
    URLCrawler crawlerObject = new URLCrawler();
    this.crawlerObject=crawlerObject;

   InputStream streamObject = null;
   BufferedReader readerObject;
   String line;
   int linkCount=0;
   int lastIndex = 0;
   int wordCount = 0;
   try
   {
      URL url = new URL(urlToSearch);
    }catch(){
   }
 }
```

6- in this try block Create “HttpURLConnection” with url and check status of response if response code will be 200 then only we will read page of url”

```
// Creating Connection with Url.
HttpURLConnection connection = (HttpURLConnection)url.openConnection();
           connection.setRequestMethod("GET");
           connection.connect();
           int code = connection.getResponseCode();
           logger.info("Response of Url is :- "+code);
           jsonObject.put("Url Status",code);
           this.getUrl=urlToSearch;
           if(code==200)
           {
           }
```

Description -

`HttpURLConnection connection = (HttpURLConnection)url.openConnection();`

"Line use to create connection with Url"

`jsonObject.put("Url Status",code);`

" Storing url status into json object"

7- Get the page of url and read it by using "BufferedReader"

"Add below code into if block of crawl method"

```
streamObject = url.openStream();
readerObject = new BufferedReader(new InputStreamReader(streamObject));
```

8- To read URL give page line by line . add below code into if() block of crawl method .

```
 while ((line = readerObject.readLine()) != null) {}
```

9- Search user given word on page and count occurrence .

"Add below code into upper while loop"

```
 // Counting Search Word Of User On Page.
 while((lastIndex = line.indexOf(searchWord, lastIndex)) != -1) {
      wordCount++;
      lastIndex += searchWord.length() - 1;
  }
```

10- find most recommended url "on basis of word occurrence on page "
“Add below code into top most while loop”

```
// Finding Most Recomanded Url
      if(wordCount>this.maxCount){
           this.maxCount=wordCount;
          this.recomendUrl=this.getUrl;
  }
```

11- find all links of current page by using ("href=") regex value .
“Add below code into top most while loop”

```
// Finding Links on page.
   if(line.contains("href="))
   {
   }
```

12- All code of this point you will add into upper (if) block
“Upper while loop will give whole line of page which contain url . so you should take only link part from line”.

12.1 - get link part from line

```
String htmlRegex = constants.getProperty("HTML_A_TAG_PATTERN");
Pattern patternTag = Pattern.compile(htmlRegex);
Matcher matcherTag = patternTag.matcher(line);
```

12.2 - remove “http:” from link part .

```
if(!href.contains("https:") && !href.contains("http:"))
     {
      Pattern que = Pattern.compile("\"([^\"]*)\"");
      Matcher qm = que.matcher(href);
        while (qm.find()) {
        this.links.add(qm.group(1));
        linkCount=linkCount+1;
  }
```

12.3 - now your if() block will look like

```
if(code==200)
{
streamObject = url.openStream();
readerObject = new BufferedReader(new InputStreamReader(streamObject));
 // Reading html Page line by line
  while ((line = readerObject.readLine()) != null) {
 // Counting Search Word Of User On Page.
  while((lastIndex = line.indexOf(searchWord, lastIndex)) != -1) {
      wordCount++;
      lastIndex += searchWord.length() - 1;
    }
 // Finding Most Recomanded Url
  if(wordCount>this.maxCount){
       this.maxCount=wordCount;
       this.recomendUrl=this.getUrl;
      }
 // Finding Links on page.
  if(line.contains("href="))
   {
String htmlRegex = constants.getProperty("HTML_A_TAG_PATTERN");
Pattern patternTag = Pattern.compile(htmlRegex);
Matcher matcherTag = patternTag.matcher(line);
    // Getting Link value from href
    while (matcherTag.find()){
        String href = matcherTag.group(1); // href
        String linkText = matcherTag.group(2); // link text
          if(!href.contains("https:") && !href.contains("http:"))
           {
             Pattern que = Pattern.compile("\"([^\"]*)\"");
             Matcher qm = que.matcher(href);
              while (qm.find()) {
                 this.links.add(qm.group(1));
                 linkCount=linkCount+1;
                 }
              }
           }
         }
    // Getting All Available Link On Of Page
   logger.info("Total Links Available On Page :- "+linkCount);
   jsonObject.put("Links_Availbale",linkCount);
```

```
    logger.info("Total "+searchWord+" Available On Page - "+wordCount);
        jsonObject.put("Word_Count",wordCount);
        jsonArryObject.add(jsonObject);
          wordCount=0;
          linkCount=0;
    logger.info("\n-------------------------------------------------\n");
        }else{
    logger.info(message.getProperty("urlProblem")+" "+getUrl);
logger.info("\n-------------------------------------------------\n");
}
```

Description -

"This lines we use to add information into json objects"

```
logger.info("Total Links Available On Page :- "+linkCount);
     jsonObject.put("Links_Availbale",linkCount);
     logger.info("Total "+searchWord+" Available On Page - "+wordCount);
     jsonObject.put("Word_Count",wordCount);
     jsonArryObject.add(jsonObject);
```

13- Store all relative link into list .

"Add bolow code into 'URLCrawler' class" .

```
// to get all links from user
        public List<String> getLinks()
        {
          return this.links;
        }
```

14 - call this method from

Story 2 - Source code of files .

Task 1 - Source code of URLCrawler.java file of processors

```
/**
*  URLCrawler class will generate HTTP request and collecting the links.
*  as well this class will it will find search word .
*
* @author  knight Learning Solutions
* @version 1.0
* @since    2019-1-5
*/

package com.ncu.processors;

import java.util.List;
import java.net.*;
import java.io.InputStream;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.io.IOException;
import java.io.File;
import java.util.regex.Pattern;
import java.util.regex.Matcher;
import java.util.LinkedList;
import java.util.Date;
import java.util.Properties;
import org.apache.log4j.Logger;
import org.json.simple.JSONArray;
import org.json.simple.JSONObject;

public class URLCrawler{

  // Storing Url in list .
```

```
private List<String> links = new LinkedList<String>();
String urlToSearch,searchWord,getUrl;
Logger logger;
JSONObject jsonObject;
JSONArray jsonArryObject;
Date beforeStart,afterStart;
long diffInMilisecond;
int maxCount=0;
String recomendUrl;
int maxGetWord;
Properties message,constants;
URLCrawler crawlerObject;

public URLCrawler(){
  // initialization of GetLogger to get logger object
  GetLogger loggerObject=new GetLogger();
  Logger logger=loggerObject.loggerValue("SearchText");
  this.logger=logger;

  // initialization of GetLogger to get logger object
  JSONArray jsonArryObject = new JSONArray();
  this.jsonArryObject=jsonArryObject;

  // initialization of configMessage to get messages
  GetConfiguration propertyObject=new GetConfiguration();
  Properties message=propertyObject.configMessages();
  this.message=message;

  // calling configConstants method to get constant values
  Properties constants=propertyObject.configConstants();
  this.constants=constants;
}

public void crawl(String urlToSearch,String searchWord)
{
    JSONObject jsonObject = new JSONObject();
    this.jsonObject=jsonObject;
    this.searchWord=searchWord;
    this.urlToSearch=urlToSearch;

  URLCrawler crawlerObject = new URLCrawler();
  this.crawlerObject=crawlerObject;
```

```
    InputStream streamObject = null;
    BufferedReader readerObject;

    String line;
    int linkCount=0;
    int lastIndex = 0;
    int wordCount = 0;

    try
    {
          URL url = new URL(urlToSearch);
          logger.info("Visiting Url :- "+url);
          jsonObject.put("Url",url);

 // Creating Connection with Url.
HttpURLConnection connection = (HttpURLConnection)url.openConnection();
          connection.setRequestMethod("GET");
          connection.connect();
          int code = connection.getResponseCode();
          logger.info("Response of Url is :- "+code);
          jsonObject.put("Url Status",code);
          this.getUrl=urlToSearch;
          if(code==200)
          {
          streamObject = url.openStream();
readerObject = new BufferedReader(new InputStreamReader(streamObject));
    // Reading html Page line by line
    while ((line = readerObject.readLine()) != null) {
        // Counting Search Word Of User On Page.
    while((lastIndex = line.indexOf(searchWord, lastIndex)) != -1) {
            wordCount++;
            lastIndex += searchWord.length() - 1;
         }
     // Finding Most Recomanded Url
         if(wordCount>this.maxCount){
           this.maxCount=wordCount;
           this.recomendUrl=this.getUrl;
                     }
       // Finding Links on page.
        if(line.contains("href="))
          {
```

```
String htmlRegex = constants.getProperty("HTML_A_TAG_PATTERN");
    Pattern patternTag = Pattern.compile(htmlRegex);
    Matcher matcherTag = patternTag.matcher(line);
        // Getting Link value from href
     while (matcherTag.find()){
          String href = matcherTag.group(1); // href
          String linkText = matcherTag.group(2); // link text
          if(!href.contains("https:") && !href.contains("http:"))
               {
             Pattern que = Pattern.compile("\"([^\"]*)\"");
             Matcher qm = que.matcher(href);
                while (qm.find()) {
                this.links.add(qm.group(1));
                linkCount=linkCount+1;
                 }
               }
             }
            }
         }
   // Getting All Available Link On Of Page
    logger.info("Total Links Available On Page :- "+linkCount);
    jsonObject.put("Links_Available",linkCount);
 logger.info("Total "+searchWord+" Available On Page - "+wordCount);
    jsonObject.put("Word_Count",wordCount);
    jsonArryObject.add(jsonObject);
    wordCount=0;
    linkCount=0;
  logger.info("\n-----------------------------------------------\n");
       }else{
  logger.info(message.getProperty("urlProblem")+" "+getUrl);
  logger.info("\n-----------------------------------------------\n");
      }
    }catch(Exception mue)
    {
          logger.info(message.getProperty("netWorkIssue"));
    }
  // At Last Calling Finally block to close Stream.
  finally {
    try {
        if (streamObject != null) streamObject.close();
    }catch(IOException ioe){
        System.out.println(this.getUrl);
```

```
      }
    }
  }

// get process information at class level.
public void processInformation(Date beforeStart,Date afterStart,long
diffInMilisecond)
      {
        this.beforeStart=beforeStart;
        this.afterStart=afterStart;
        this.diffInMilisecond=diffInMilisecond;
      }
  // to get all links from user
        public List<String> getLinks()
        {
          return this.links;
        }

      }
```

Story 4 - Compile and run your application of day 2

Task 1 - Compile code

1- compile all exceptions java files

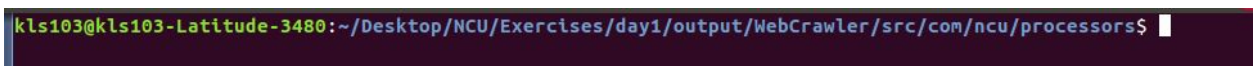
"Open terminal into exceptions folder of project"

2-compile all java file of exceptions folder .

```
javac  -d "/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes"
*.java
```

3- compile processors java files

"Open terminal into processors folder of project"



4-compile GetLogger.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.jar:
/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/json-simple-1.1.1.ja
r:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes"  GetLogger.java
```

5- compile GetConfiguration.java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/json-simple-1.1.1
.jar:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.ja
r:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes"
GetConfiguration.java
```

6- compile all java files of Validator folder

6.1- open terminal into validator folder .

```
kls103@kls103-Latitude-3480:~/Desktop/NCU/Exercises/day2/output/WebCrawler/src/com/ncu/validators$
```

6.2- compile validators file .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/json-simple-1.1.1
.jar:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.ja
r:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes"  *.java
```

7- compile all java file of processors folder .

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/json-simple-1.1.1
.jar:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.ja
r:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" *.java
```

8- compile Crawler.java file of main folder .

8.1 - open terminal into main folder

8.2 - compile Crawler.java file of main folder

```
javac -cp
".:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/json-simple-1.1.1
.jar:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.ja
r:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" -d
"/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes" Crawler.java
```

Task 2 - Run code

1 - open terminal in root of application “ WebCrawler”

2- Run your application

```
java -cp
".:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/json-simple-1.1.1
.jar:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.ja
r:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/classes"
com.ncu.main.Crawler
```

Story 5- Output of day 4

```
r/libs/json-simple-1.1.1.jar:/home/kls103/Desktop/NCU/Exercises/day4/output/WebCrawler/libs/log4j-1.2.17.jar:/home/kls103/Desktop/NCU/Exercises/
day4/output/WebCrawler/classes" com.ncu.main.Crawler
2019-01-19 13:05:42 INFO  class:31 - ====================================
2019-01-19 13:05:42 INFO  class:32 - ||        Welcome NCU Students       ||
2019-01-19 13:05:42 INFO  class:33 - ====================================


2019-01-19 13:05:42 INFO  class:56 - Please Enter Your Url :-
https://www.calcuttaairport.com


2019-01-19 13:06:04 INFO  class:78 - Enter Your String Which You Want to Search :-
Netaji


2019-01-19 13:06:10 INFO  class:102 - How Many Pages Do You Want To Search :-
10


2019-01-19 13:06:11 INFO  class:114 - Process Started . . . . . .
2019-01-19 13:06:12 INFO  class:88 - Visiting Url :- https://www.calcuttaairport.com
2019-01-19 13:06:19 INFO  class:96 - Response of Url is :- 200
2019-01-19 13:06:20 INFO  class:143 - Total Links Available On Page :- 35
2019-01-19 13:06:20 INFO  class:146 - Total Netaji Available On Page - 5
2019-01-19 13:06:20 INFO  class:151 -
---------------------------------------------

2019-01-19 13:06:20 INFO  class:88 - Visiting Url :- https://www.calcuttaairport.com/arrivals.php
2019-01-19 13:06:20 INFO  class:96 - Response of Url is :- 200
2019-01-19 13:06:21 INFO  class:143 - Total Links Available On Page :- 29
2019-01-19 13:06:21 INFO  class:146 - Total Netaji Available On Page - 0
2019-01-19 13:06:21 INFO  class:151 -
---------------------------------------------

2019-01-19 13:06:21 INFO  class:88 - Visiting Url :- https://www.calcuttaairport.com/departures.php
2019-01-19 13:06:21 INFO  class:96 - Response of Url is :- 200
2019-01-19 13:06:22 INFO  class:143 - Total Links Available On Page :- 29
2019-01-19 13:06:22 INFO  class:146 - Total Netaji Available On Page - 1
2019-01-19 13:06:22 INFO  class:151 -
---------------------------------------------
```