Day 3

In day 3 session we will store bunch of urls into list . and track of pages that we've already visited . Put a limit on the number of pages to search so this doesn't run for eternity.

Task Of Day

I. Implementation of TextCrawler
II. Logics -
   A. - Use set to store unique urls .
   B. - Use List to store all visited url .
   C. - System should search limited url given by user
   D. - System should display process time of your processes.

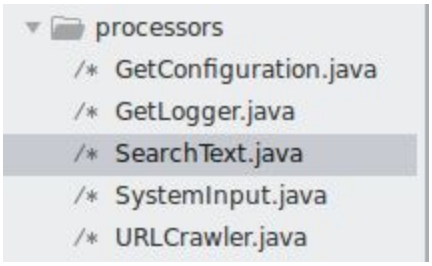
Story 1 - Implementation of TextCrawler

“ This TextCrawler class will do mainly two tasks”

- Keep track of pages that we've already visited
- Put a limit on the number of pages to search so this doesn't run for eternity.

Let's sketch out the first draft of our SearchText.java file .

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



2- Create “SearchText” class into “SearchText.java” file .

```
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import java.util.Date;
import java.util.Properties;
import org.apache.log4j.Logger;
import java.net.*;
public class SearchText
{
// Using Set to contains unique entries.
private Set<String> pagesVisited = new HashSet<String>();
// Using List- This is just storing a bunch of URLs we have to visit
next.
private List<String> pagesToVisit = new LinkedList<String>();
}
```

Description -

Why is pagesVisited a Set? Remember that a set, by definition, contains unique entries. In other words, no duplicates. All the pages we visit will be unique (or at least their URL will be unique). We can enforce this idea by choosing the right data structure, in this case a set.

Why is pagesToVisit a List? This is just storing a bunch of URLs we have to visit next. When the crawler visits a page it collects all the URLs on that page and we just append them to this list. Recall that Lists have special methods that Sets ordinarily do not, such as adding an entry to the end of a list or adding an entry to the beginning of a list. Every time our crawler visits a webpage, we want to collect all the URLs on that page and add them to the end of our big list of pages to visit .

Is this necessary? No. But it makes our crawler a little more consistent, in that it'll always crawl sites in a breadth-first approach (as opposed to a depth-first approach).

Remember how we don't want to visit the same page twice? Assuming we have values in these two data structures, can you think of a way to determine the next site to visit?

Okay, here's my method for the SearchText.java class:

Add below code into " SearchText" class

```
// getting url one by one from list
  private String nextUrl()
  {
    String nextUrl;
    do{
      nextUrl = pagesToVisit.remove(0);
    } while(pagesVisited.contains(nextUrl));
    pagesVisited.add(nextUrl);
    return nextUrl;
  }
```

A little Explanation -

We get the first entry from pagesToVisit, make sure that URL isn't in our set of URLs we visited, and then return it. If for some reason we've already visited the URL (meaning it's in our set pagesVisited) we keep looping through the list of pagesToVisit and returning the next URL.

3- Assuming we have this other class that's going to do the work listed above, can we write one public method for this SearchText.java class

This method will take three value in argument

A. url, first url give by user
B. searchWord , word which user want to search .
C. numberOfPage , limit of number of pages to search

```
public URLCrawler search(String url, String searchWord, int numberOfPage)
  {
  }
```

4- Create URL class object and pass user give url , to get

1. url Protocol (//http)
2. Authority (//example.com:8080) .

"it can generate exception so we will write this into try and catch block"

```
    try{
      // creating URL class object to get content of url
      URL urlObj = new URL(url);
      String  protocolValue = urlObj.getProtocol(); //http
      this.protocolValue=protocolValue;
      String  domainValue = urlObj.getAuthority(); //example.com:80
      this.domainValue=domainValue;
    }catch(Exception e){
      logger.info(message.getProperty("notAccessibleUrl"));
      // Calling inputUrl() Method To Take Input From User.
      systemInputObject.inputUrl();
    }
```

5- add custom message into "exceptions.properties" for "notAccessibleUrl"

```
notAccessibleUrl="Oops.. This Url Is Not Accessible Please Try With
Different Url... "
```

6- Create root url using Protocol and Authority , which we have retrieve

```
// consider first utl as a root url.
    this.roolUrl=this.protocolValue+"://"+this.domainValue+"/";
```

7- create object of date class to calculate data and time before start process

```
// getting Time before start Process .
    Date beforeProcess = new Date();
    long startTime = System.currentTimeMillis( );
```

8- create URLCrawler class object which we will implement in future session

```
// creating Object of URLCrawler class.
    URLCrawler utlCrawlerObj = new URLCrawler();
```

9- Implements Limit on page to search (search only user given number of page to search value) -

```
try{
      while(pagesVisited.size() < maxPageToSearch){
      // check first time url is empty.
        String currentUrl;
        if(pagesToVisit.isEmpty()){
          currentUrl = url;
          pagesVisited.add(url);
        }
        else{
          currentUrl = this.nextUrl();
          currentUrl=this.roolUrl+currentUrl;
        }
// Calling crawl method to get available url and search word of given
Url
        utlCrawlerObj.crawl(currentUrl,searchWord);
// get all url page and calling getLinks method to store value in
list
        pagesToVisit.addAll(utlCrawlerObj.getLinks());
        if(pagesToVisit.isEmpty()){
           logger.info(message.getProperty("noMorePage"));
         break;
        }
      }
    }catch (Exception e) {
      if(pagesVisited.size() < maxPageToSearch)
      {
        logger.info(message.getProperty("noMorePage"));
```

```
      }
    }
```

Upper code have line -

```
// Calling crawl method to get available url and search word of given
Url
        utlCrawlerObj.crawl(currentUrl,searchWord);
```

Description of line -

Okay, so we can determine the next URL to visit, but then what? We still have to do all the work of HTTP requests, parsing the document, and collecting words and links. But let's leave that for another class and wrap this one up. This is an idea of separating out functionality. Let's assume that we'll write another class (we'll call it URLCrawler.java) to do that work we have created object .

10 - To display process time create “processTime” method into “SearchText.java” file

```
// Calculating Process Time .
public long processTime(Date beforeProcess,Date afterProcess,long
startTime,long endTime)
  {
logger.info("__________ Time Duration _________________\n");
    logger.info("Time Before Process :- "+beforeProcess);
    // Getting time after Process
    logger.info("Time After Process :- "+afterProcess);
    // checking time difference of start and end time of process.
    long timedifference = endTime - startTime;
    logger.info("Duration In MilliSecond -: " + timedifference+"ms");
    logger.info("_____________________________________________");
    return timedifference;
  }
```

Story 2 - Source code of files .

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

```
/**
* 1- The SearchText class use to store all unique url entries. In other
words, no duplicates.
*    All the pages we visit will be unique (or at least their URL will be
unique).
*    We can enforce this idea by choosing the right data structure, in this
case a set. and
* 2- This is just storing a bunch of URLs we have to visit next. When the
crawler visits a page it collects
*    all the URLs on that page and we just append them to this list
*
* @author  knight Learning Solutions
* @version 1.0
* @since   2019-1-5
*/
package com.ncu.processors;
import java.util.HashSet;
import java.util.LinkedList;
import java.util.List;
import java.util.Set;
import java.util.Date;
import java.util.Properties;
import org.apache.log4j.Logger;
import java.net.*;
public class SearchText
{
  // Using Set to contains unique entries.
  private Set<String> pagesVisited = new HashSet<String>();
  // Using List- This is just storing a bunch of URLs we have to visit
next.
  private List<String> pagesToVisit = new LinkedList<String>();
  String roolUrl;
  Logger logger;
  String protocolValue,domainValue;
  int maxPageToSearch;
  Properties message;
```

```
SystemInput systemInputObject;
SearchText searchTextObject;
//Creating Constructor to Initialization of Logger and config
public SearchText()
{
  // initialization of GetLogger to get logger object
  GetLogger loggerObject=new GetLogger();
  Logger logger=loggerObject.loggerValue("SearchText");
  this.logger=logger;
  // initialization of configMessage to get messages
  GetConfiguration propertyObject=new GetConfiguration();
  Properties message=propertyObject.configMessages();
  this.message=message;
}
// This method will set all urls into list and visited url into set.
public URLCrawler search(String url, String searchWord, int numberOfPage)
{
  int maxPageToSearch = numberOfPage;
  this.maxPageToSearch=maxPageToSearch;
  // Creating Object Of SystemInput class ..
  SystemInput systemInputObj=new SystemInput();
  this.systemInputObject=systemInputObj;
  // Creating Object of Own Class ...
  SearchText searchTextObject = new SearchText();
  this.searchTextObject=searchTextObject;
  try{
    // creating URL class object to get content of url
    URL urlObj = new URL(url);
    String  protocolValue = urlObj.getProtocol(); //http
    this.protocolValue=protocolValue;
    String  domainValue = urlObj.getAuthority(); //example.com:80
    this.domainValue=domainValue;
  }catch(Exception e){
    logger.info(message.getProperty("notAccessibleUrl"));
    // Calling inputUrl() Method To Take Input From User.
    systemInputObject.inputUrl();
  }
  // consider first url as a root url.
  this.roolUrl=this.protocolValue+"://"+this.domainValue+"/";
  // getting Time before start Process .
  Date beforeProcess = new Date();
  long startTime = System.currentTimeMillis( );
```

```
    // creating Object of URLCrawler class.
    URLCrawler utlCrawlerObj = new URLCrawler();
     try{
      while(pagesVisited.size() < maxPageToSearch){
      // check first time url is empty.
        String currentUrl;
        if(pagesToVisit.isEmpty()){
          currentUrl = url;
          pagesVisited.add(url);
        }
        else{
          currentUrl = this.nextUrl();
          currentUrl=this.roolUrl+currentUrl;
        }
        // Calling crawl method to get available url and search word of
given Url
        utlCrawlerObj.crawl(currentUrl,searchWord);
        // get all url page and calling getLinks method to store value in
list
        pagesToVisit.addAll(utlCrawlerObj.getLinks());
        if(pagesToVisit.isEmpty()){
           logger.info(message.getProperty("noMorePage"));
         break;
        }
      }
    }catch (Exception e) {
      if(pagesVisited.size() < maxPageToSearch)
      {
        logger.info(message.getProperty("noMorePage"));
      }
    }
    System.out.println("\n");
    logger.info("**Done** Visited " + pagesVisited.size() + " web
page(s)");
    // End Time of Process .....
    Date afterProcess = new Date();
    long endTime = System.currentTimeMillis();
    long timedifference =
searchTextObject.processTime(beforeProcess,afterProcess,startTime,endTime);
utlCrawlerObj.processInformation(beforeProcess,afterProcess,timedifference)
;
```

```
    return utlCrawlerObj;
  }
// Calculating Process Time .
public long processTime(Date beforeProcess,Date afterProcess,long
startTime,long endTime)
  {
  logger.info("_____________ Time Duration _____________________\n");
    logger.info("Time Before Process :- "+beforeProcess);
    // Getting time after Process
    logger.info("Time After Process :- "+afterProcess);
    // checking time difference of start and end time of process.
    long timedifference = endTime - startTime;
    logger.info("Duration In MilliSecond -: " + timedifference+"ms");
    logger.info("_____________________________________________________");
    return timedifference;
  }
  // getting url one by one from list
  private String nextUrl()
  {
    String nextUrl;
    do{
      nextUrl = pagesToVisit.remove(0);
    } while(pagesVisited.contains(nextUrl));
    pagesVisited.add(nextUrl);
    return nextUrl;
  }
}
```

Task 2 - Source code of file “exceptions.properties” file of constants

```
emptyUrlMessage="Oops.. Empty Url Is Not Acceptable ...!"
invalidUrlMessage="Oops.. Given Url Is Not Valid Plz Try With Other Url
...!"
emptySearchMessage="Oops.. Empty Search Is Not Acceptable ...!"
searchLengthMessage="You have Given Long File Name .This System Accept Only
Less Than 25 Characters To File Name ..!"
maxPageNumberMessage="Oops.. This System Accept Only Less Than 20 Pages
Numbers To Search ..!"
```

```
validIntegerInput="Oops.. Mismatch Input Not Please Give Value In Integer
Format..!"
emptyFileNameMessage="Oops.. Empty File Name Is Not Acceptable ...!"
extensionFormatMessage="Oops.. Extension Is Messing .You Should Also Give
.json Extension ...!"
invalidFileExtension="Oops.. This is not json file ! This System Accept
Only json file ...!"
specialcharacterMessage="Oops.. You have given special characters into file
name . This System does not take Special characters ...!"
fileExistMessage="Oops.. This File Is Already Exist Into Directory , Please
Try With Different Name...!"
notWriteInFileMessage="..................... Console Information Is Not
Written Into File ....................";
exitMessage="Do You Want To Exist From System... Please Write (exit) or Any
Other Key To Continue Process ... !";
notAccessibleUrl="Oops.. This Url Is Not Accessible Please Try With
Different Url...... "
noMorePage="Opps.. No More Pages Is Available To Search ....."
netWorkIssue="Oops There are some netWork Issues or Not Accessible Url ";
urlProblem="Status Is Not Get 200 - There Are Some Problem In Url Try Other
Url";
```

Story 4 - Compile and run your application of day 3

Note - “we still have remaining implementation of "URLCrawler.java file " . so we will compile current day code into day 4 session”