PRÁCTICA 1:

Proyecto Software:

Web-Scraping

ASIGNATURA: Modelos de computación

FECHA: 28 de mayo de 2021

ALUMNO: Álvaro Sanz Cortés

TRABAJO: Memoria

ÍNDICE

1. Análisis y descripción del proyecto

1.1. Introducción a la práctica

En esta práctica veremos el uso de las herramientas JFlex y JSoup para obtener información de páginas web. Como por ejemplo obtener los enlaces web, verificar si el código html escrito es correcto o obtener el número de enlaces web a imágenes hay en una página web.

1.2. Parte 1: JFlex

1.2.1. Resumen del uso que ha realizado de JFLEX

En la parte 1 he utilizado JFlex para que muestre los enlaces obtenidos de las URL de imágenes y/o enlaces, obtenidas del archivo prueba1.html, localizado en la carpeta test. En la clase JFlexScraper.java he creado un autómata para que recorriera todas las etiquetas del fichero prueba1.html., además de métodos en los que obtenían las listas de: URL de enlaces web y URL de enlaces de imágenes, por último, he comprobado si el código html del fichero es correcto. Tras esto he obtenido los enlaces para así imprimirlos por pantalla. Las URL utilizadas son:

URL página web:

https://es.wikipedia.org/wiki/Clasificaci%C3%B3n_de_autom%C3%B3viles

URL imagen:

https://d500.epimg.net/cincodias/imagenes/2019/04/03/motor/1554282540_943256_1554282615_noticia_normal.jpg

1.2.2. Autómata que ilustre el funcionamiento del analizador programado

En cuanto el funcionamiento del autómata en la clase JFlexScraper.java para leer las etiquetas utilizadas, he optado por el siguiente autómata:

El autómata está compuesto por 7 estados:

- Estado inicial y final: Comienza y finaliza el autómata. Pasa al estado 1

- Estado 1: Paso a este estado si he leído “<”, un token de tipo OPEN. Pasando así al estado 2 y 6.

- Estado 2: Se llega a este estado tras haber leído un token de tipo palabra como “a” o “img”. Pasando así al estado final tras haber leído “>” Como “<HTML>. Por otra parte, si se hay leído “/” se pasa al estado 5.

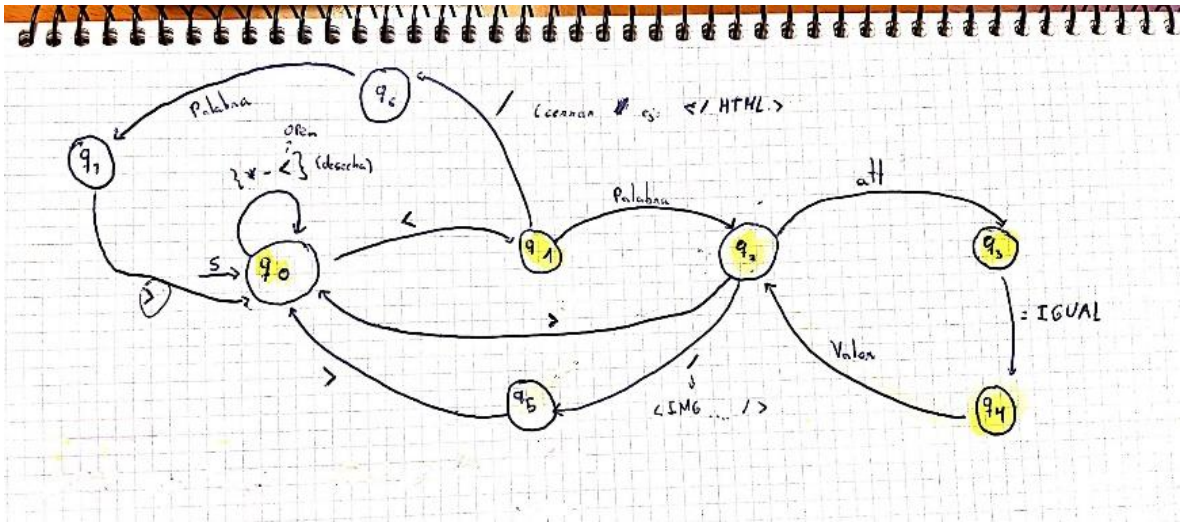
- Estado 3: Se llega si se ha leído un atributo, un token de tipo palabra como href de <A (enlaces) o src de <IMG (enlaces de imágenes). Pasando así al estado 4.

- Estado 4: Se llega si se ha leído un igual para igualar el atributo del estado 3 con el valor. Pasando así al estado 2 tras haber leído el valor, en este caso, los enlaces a las páginas web.

- Estado 5: Se llega desde el estado 2 si se ha leído “/”. Pasando así al estado final tras haber leído un token de tipo CLOSE. Como: <IMG src = “” />

- Estado 6: Se llega desde el estado 1 tras haber leído “/”. Pasando así al estado 7 tras haber leído un token palabra, en este caso, HTML. Como </HTML

- Estado 7: Tras haber leído el estado 6, se añade un token de tipo CLOSE. Como </HTML>.



1.2.3. Decisiones de diseño tomadas en la práctica y como se han implementado

Una de las decisiones de la práctica ha sido como implementar el autómata para obtener el enlace de la web y de la imagen; esto ha sido implementado mediante un switch para así hacer de cada caso del switch un estado. Por otra parte, para imprimir los enlaces se ha creado un fichero con todos los hiperenlaces y posteriormente se ha comprobado si está bien balanceado.

1.3. Parte 2: JSoup

1.3.1. Resumen del uso que ha realizado de JSoup

En la parte 2 he utilizado JSoup para que muestre loa cantidad de enlaces obtenidos de las URL de imágenes y enlaces, obtenidas de una página web, localizado en la carpeta test. En la clase JsoupScraper.java están implementados los métodos para obtener el número de enlaces obtenidos.

1.3.2. Decisiones de diseño tomadas en la práctica y como se han implementado

En esta segunda parte de la práctica, he optado por escribir el código bajo comentarios debido a errores de compilación. En cuanto a decisiones de diseño he creído conveniente crear una ArrayList de enlaces de imágenes y otra distinta hiperenlaces en relación a atributos de token palabra como “a”,”br”,”div”,”li”… Tras esto he utilizado un for para recorre los ArrayList y mostrar el número total de la etiqueta utilizada.

2. Listado de todo el código fuente

WebCrawler.java

```
package es.ceu.gisi.modcomp.webcrawler;
import es.ceu.gisi.modcomp.webcrawler.jflex.JFlexScraper;
import java.io.File;
import java.io.IOException;
import es.ceu.gisi.modcomp.webcrawler.jsoup.JsoupScraper;

/**
 * Esta aplicación contiene el programa principal que ejecuta ambas partes del
 * proyecto de programación.
 *
 * @author Alvaro Sanz
 */
public class WebCrawler {

   private final static String PATH_PRUEBAS = new java.io.File("").getAbsolutePath()
         + "/test/es/ceu/gisi/modcomp/webcrawler/jflex/test/";
```

```
    private static File ficheroPrueba1 = new File(PATH_PRUEBAS + "fichero1.html");

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

        // Deberá inicializar JFlexScraper con el fichero HTML a analizar
        // Y creará un fichero con todos los hiperenlaces que encuentre.
        // También deberá indicar, mediante un mensaje en pantalla que
        // el fichero HTML que se ha pasado está bien balanceado.

        JFlexScraper j = new JFlexScraper(ficheroPrueba1);
        j.obtenerHiperenlaces();
        j.obtenerHiperenlacesImagenes();
        j.esDocumentoHTMLBienBalanceado();
       System.out.println(j.esDocumentoHTMLBienBalanceado());
       System.out.println("\nEnlacesA: " + j.obtenerHiperenlaces());
       System.out.println("\nEnlacesIMG: " + j.obtenerHiperenlacesImagenes());

        // Deberá inicializar JsoupScraper con la DIRECCIÓN HTTP de una página
        // web a analizar. Creará un fichero con todos los hiperenlaces que
        // encuentre en la página web. También obtendrá estadísticas de uso
        // de las etiquetas HTML más comunes: a, br, div, li, ul, p, span, table, td, tr
    }
}
```

JFlexScraper

```
package es.ceu.gisi.modcomp.webcrawler.jflex;

import es.ceu.gisi.modcomp.webcrawler.jflex.lexico.Tipo;
import es.ceu.gisi.modcomp.webcrawler.jflex.lexico.Token;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
```

```
import java.io.FileReader;
import java.io.Reader;
import java.util.ArrayList;
import java.io.IOException;
import java.util.Stack;

/**
 * Esta clase encapsula toda la lógica de interacción con el parser HTML.
 *
 * @author Alvaro Sanz
 */
public class JFlexScraper {

   HTMLParser analizador;

   ArrayList<String> enlacesA = new ArrayList<>();
   Stack<String> etiquetasAbiertas = new Stack();
   ArrayList<String> enlacesIMG = new ArrayList<>();

   public JFlexScraper(File fichero) throws FileNotFoundException, IOException {
      Reader reader1 = new BufferedReader(new FileReader(fichero));
      analizador = new HTMLParser(reader1);

      int estado = 0;
      Token tk = analizador.nextToken();
      boolean etiquetaA = false;
      boolean etiquetaIMG = false;
      boolean valorHREF = false;
      boolean valorSRC = false;
      boolean bienBalanceado = false;


      while ( tk != null){
```

```
switch(estado){
   case 0:
      //Estando en el estado 0
      if(tk.getTipo()== Tipo.OPEN) { // <
         estado = 1;
      }
      break;
   case 1:
      //Estamos en el estado 1
      //En este estado estamos si hemos leido un OPEN <
      if(tk.getTipo()== Tipo.PALABRA){
         estado = 2;
         etiquetasAbiertas.push(tk.getValor().toLowerCase());

         if(tk.getValor().equalsIgnoreCase("a")){
            etiquetaA = true;
         }else if(tk.getValor().equalsIgnoreCase("img")){
            etiquetaIMG = true;
         }
      }else if(tk.getTipo() == Tipo.SLASH){
         estado = 6;
      }
      break;
   case 2:
      //Estando en el estado 2
      //Ahora debemos leer att=valor o fin de etiqueta
      if(tk.getTipo() == Tipo.PALABRA){
         estado = 3;
         if(etiquetaA){
            if(tk.getValor().equalsIgnoreCase("href")){
               valorHREF = true;
            }
         }else if(etiquetaIMG){
```

```
            if (tk.getValor().equalsIgnoreCase("src")){
               valorSRC = true;
            }
         }
      }if (tk.getTipo() == Tipo.CLOSE){
         bienBalanceado = true;
      }else if(tk.getTipo() == Tipo.SLASH){
         estado = 5;
      }
      break;
   case 3:
      //Estando en el estado 3
      //Ahora debemos leer "="
      if(tk.getTipo()==Tipo.IGUAL){
         estado = 4;
      }
      break;
   case 4:
      //Estando en el estado 4
      //Esto es un valor de un atributo
      if(tk.getTipo() == Tipo.VALOR){
         if(valorHREF){
            enlacesA.add(tk.getValor());
         }
         if(valorSRC){
            enlacesIMG.add(tk.getValor());
         }
         estado = 2;
      }
      break;
   case 5:
      //Estando en el estado 5
      //Se cierran las palabras con atributos
```

```
            if(tk.getTipo() == Tipo.CLOSE){
               bienBalanceado = true;
            }
            break;
         case 6:
            //Estando en el estado 6
            //Se añaden las palabras tras un SLASH del estado 1
            if(tk.getTipo()== Tipo.PALABRA){
               if(tk.getValor().equalsIgnoreCase(etiquetasAbiertas.peek())){
                  etiquetasAbiertas.pop();
                  estado = 7;
               }else {
                  bienBalanceado = false;
               }
            }
            break;
         case 7:
            //Estando en el estado 7
            //Se cierran las palabras
            if(tk.getTipo() == Tipo.CLOSE){
               bienBalanceado = true;
            }
      }
      tk = analizador.nextToken();
   }
}



public ArrayList<String> obtenerHiperenlaces(){
   for(int x=0;x< enlacesA.size();x++) {
      System.out.println(enlacesA.get(x));
   }
   return enlacesA;
```

```
    }

    public ArrayList<String> obtenerHiperenlacesImagenes() {
        return enlacesIMG;
    }

    public boolean esDocumentoHTMLBienBalanceado() {
        return true;
    }
}
```

JsoupScraper.java

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package es.ceu.gisi.modcomp.webcrawler.jsoup;



import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

/**
 * Esta clase encapsula toda la lógica de interacción con el analizador Jsoup.
 *
 * @author Alvaro Sanz
```

```
 */
public class JsoupScraper {

  private Document doc;

  /**
   * Este constructor crea un documento a partir de la URL de la página web a
   * analizar.
   *
   * @param url Una URL que apunte a un documento HTML (p.e.
   *          http://www.servidor.com/index.html)
   */
  public JsoupScraper(URL url) throws IOException {
    // La variable deberá inicializarse de alguna manera utilizando una URL...
    // De momento, se inicializa a null para que compile...
    try{
      URL url1 = new URL("http://www.servidor.com/index.html");
      doc = Jsoup.connect(url1.toString()).get();
    }catch(IOException e){
      System.out.println(e);
    }
  }

  /**
   * Este constructor crea un documento a partir del contenido HTML del String
   * que se pasa como parámetro.
   *
   * @param html Un documento HTML contenido en un String.
   */
  public JsoupScraper(String html) throws IOException {
    doc = Jsoup.parse(html);
  }
```

```
/**
 * Realiza estadísticas sobre el número de etiquetas de un cierto tipo.
 *
 * @param etiqueta La etiqueta sobre la que se quieren estadísticas
 *
 * @return El número de etiquetas de ese tipo que hay en el documento HTML
 */
public int estadisticasEtiqueta(String etiqueta) {

   /*HTMLParser analizador = new HTMLParser(fichero1.html);
   if(analizador = etiqueta){
      ArrayList<> etiqueta = new ArrayList<>();
      for(int i = 0, i < etiqueta.size(), i++ ){
            i += 1;
         System.out.println(i);
   }else if{
      System.out.print("false");
   }*/
   return 0;
}

/**
 * Obtiene todos los hiperenlaces que se encuentran en el documento creado.
 *
 * @return Una lista con todas las URLs de los hiperenlaces
 */
public List<String> obtenerHiperenlaces() {
   /*if(etiqueta = "href"){
      ArrayList<> hiperenlaces = new ArrayList<>();
      hiperenlaces.add(href.next());
   }*/
  return new ArrayList<String>();
}
```

```
    /**
     * Obtiene todos los hiperenlaces de las imágenes incrustadas.
     *
     * @return Una lista con todas las URLs de los hiperenlaces
     */
    public List<String> obtenerHiperenlacesImagenes() {
        /*if(etiqueta = "src"){
            ArrayList<> hiperenlacesImagenes = new ArrayList<>();
            hiperenlacesImagenes.add(src.next());
        }*/
        return new ArrayList<String>();
    }


    /**
     * Obtiene la imagen a la que hace referencia LA PRIMERA etiqueta IMG que
     * encontramos.
     *
     * @return El nombre (o ruta) de la primera imagen insertada en el documento
     *         HTML.
     */
    public String obtenerContenidoImg() {
        Element elemento = doc.select("IMG").first();
        String imagen = elemento.attr("src");
        return imagen;
    }
}
```

3. Conclusiones (que incluirán una valoración del tiempo dedicado a la práctica)

Durante la realización de la práctica he tenido dificultades a la hora de programar el fichero WebCrawler.java del cual se ejecutan las 2 partes de la práctica, además de la realización de la parte 2 del JsoupCrawler. En cuanto al JSoup y al JFlex, me parecen herramientas muy útiles cuando se quiere analizar un código html debido a que facilita ver tanto los errores (bien balanceado) como para mostrar la información deseada de ciertos atributos. Esto es muy útil debido a que podemos obtener una guía de como está escrito el código de la página web sin gastar el tiempo de leer todo el código, que está disponible gratis en cualquier página.

En cuanto al tiempo dedicado a la práctica ha sido un total de unas 10h aproximadamente contando tanto el código fuente como los commits, como la realización de este pdf.