Práctica 1: Acceso a datos de la web

Sistemas de Información Distribuidos

Pedro Allué Tamargo (758267) Juan José Tambo Tambo (755742)

2 de marzo de 2021

Índice

1. Esfuerzos

- **Pedro**: Búsqueda de documentación, programación del parser, escritura en fichero *XML* y redacción de la memoria.
  - **Tiempo invertido:** 6:30 horas.
- **Juan José**: Búsqueda de documentación, puesta a punto de proyecto *Java*, programación del parser y programación del `SAXParser`.
  - **Tiempo invertido:** 6 horas.

2. Ejercicio 1

Para la realización de la práctica se ha partido de la plantilla del código de `parser.jj` incluidas junto con el enunciado. El entorno utilizado ha sido *intellij*, ya que se tiene más experiencia utilizando esta herramienta y proporciona un *plugin* para *javacc*.

El primer paso fue completar el método `escribeTablaEnFichero` que recibe una `Hashtable` con los nombres de las empresas y sus respectivas cotizaciones y una cadena que representa el fichero sobre el que escribir. El código relacionado con este método se puede encontrar en el Listing 1. El resultado de la escritura de una `Hashtable` sobre un fichero en formato *XML* se corresponde con:

```
<?xml version="1.0" ?>
<lista_cotizaciones>
    <cotizacion>
        <empresa>AENA</empresa>
        <cotizaciones>144.7</cotizaciones>
    </cotizacion>
    <cotizacion>
        <empresa>BANKINTER</empresa>
        <cotizaciones>5.48</cotizaciones>
    </cotizacion>
    [...]
</lista_cotizaciones>
```

Una vez realizada la escritura en el fichero se ha procedido a su lectura utilizando `SAXParser` y la plantilla del *handler* adjunta al enunciado. El código de la clase `CotizacionesHandler` se puede encontrar en el Listing 2. Se han utilizado las variables *empresa* y *cotizacion* que contendrán los valores del par empresa-cotización que se está leyendo, la variable *data* que almacena lo leído del fichero con la función `characters` y una tabla que almacenará todos los pares empresa-cotización. En la función `startElement()`, se comprueba qué apartado del fichero se está leyendo. Para ello, se comprueba si la variable *qname* coincide con el tag "empresa" o "cotizaciones", en cuyos casos se modifica el valor de las variables *leeEmpresa* y *leeCotizacion* a *true* respectivamente. Estas variables sirven para comprobar en la función `endElement()` si se está leyendo un campo que interesa (empresa o cotización) y se actualiza el valor en las variables que corresponda. En esta misma función, si se está leyendo el tag "cotizacion", indica que es el fin del par empresa-cotización, por lo que se añade a la tabla el par leído.

Finalmente con los pasos de lectura y escritura del fichero *XML* funcionando se procedió a desarrollar el código del *parser*. A partir de la plantilla del enunciado era necesario completar el contenido de los *tokens* `NOMBRE_EMPRESA` y `COTIZACION_EMPRESA`. Si se examina la estructura del *HTML* de la página destino se puede observar que cada empresa se encuentra embebida en una etiqueta `tr`. El nombre de la empresa se encuentra dentro de este `tr`, en una etiqueta `td`.

En la cotización de la empresa la cosa no es tan simple ya que las cotizaciones pueden *subir*, *bajar* o mantenerse iguales respecto a la última publicada. Esto se traduce en el uso de distintas clases del *CSS* para representar la información de forma visual. Por lo tanto el *token* `COTIZACION_EMPRESA` se puede dividir en tres *tokens* que pueden indicar la cotización de la misma. La definición de estos *tokens* sería la siguiente:

```
// La cotizacion ha subido
< COTIZACION_EMPRESA_POS : "<td class=\"price top\">" >
// La cotizacion ha bajado
< COTIZACION_EMPRESA_NEG : "<td class=\"price flop\">" >
// La cotizacion se mantiene igual
< COTIZACION_EMPRESA_EQ : "<td class=\"price equal\">" >
```

También se han tenido que añadir más *tokens* para representar los distintos elementos del *HTML* tales como la tabla de cotizaciones (`TABLE`), el cuerpo de la tabla (`TBODY`), cada fila de la misma (`TROW`) y el cierre de una celda de datos de una tabla (`TD_FIN`).
Al añadir estos *tokens* se ha tenido que modificar la función `saltar()` para ignorar el token `TD_FIN`.
La función `cotizaciones()` se encarga de añadir la pareja empresa-cotización a la tabla de pares. Para ello, el patrón utilizado se inicia con el *token TABLE*, ya que la información que se quiere obtener está en la tabla de la página. Cada par empresa-cotización está dentro de un *ROW* de la tabla, así que se debe repetir una o más veces el *token TROW* seguido de empresa y cotización, tal y como se muestra a continuacón:

```
<TABLE> saltar() <TBODY> ( <TROW> saltar() empresa = empresa() cotizacion =
    cotizacion()
        {
            tabla.put(empresa, cotizacion);
        }
     saltar()   )+ <TABLE_FIN>
```

Como se puede observar en el fragmento de código anterior, se han utilizado dos funciones diferentes para obtener la cotización y la empresa ya que son tipos de datos diferentes que se encuentran en diferentes campos del código *HTML*. Una vez se han obtenido los valores del par utilizando estas funciones, se añade esta información a la tabla que posteriormente devuelve la función `cotizaciones()`.

Las funciones `empresa()` y `cotizacion()` se han desarrollado siguiendo el ejemplo expuesto en la teoría de la asignatura. Cabe destacar que en el caso de empresa la variable *token* se asigna al *token CARACTERES* y en el caso de cotización se asigna a *NUMERO*. Además, en el caso de la cotización se debe reemplazar el carácter ',' por '.' para evitar errores a la hora de convertir el token a *Double*. El código del fichero *parser.jj* completo se puede encontrar en el Listing 3.

3. Urls de interés

`https://medium.com/@aroshamalithi/build-a-javacc-parser-grammar-file-with-comparison-operators-tutorial`

`https://cs.lmu.edu/~ray/notes/javacc/`

`https://www.javatpoint.com/javacc`

`https://www.ict.social/java/files/writing-xml-files-via-the-sax-approach-in-java`

`https://www.journaldev.com/1198/java-sax-parser-example`

Anexo 1: Códigos

Listing 1: Código del método para escribir la tabla de empresas-cotizaciones en un fichero

```
/// Funci n para escribir la tabla de cotizaciones en el FICHERO
void escribeTablaEnFichero(Hashtable <String, Double> tabla, String nombre) {
    PrintStream out = null;
    XMLOutputFactory xof = XMLOutputFactory.newInstance();
    XMLStreamWriter xsw = null;
    try {
        // Escribir la tabla en el fichero FICHERO
        out = new PrintStream(nombre);
        xsw = xof.createXMLStreamWriter(out);
        // Escribimos el fichero
        escrituraFichero(xsw, tabla);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/// Funci n interna para escribir la tabla de cotizaciones con un
   XMLStreamWriter
private void escrituraFichero(XMLStreamWriter xsw, Hashtable<String, Double>
    tabla) throws XMLStreamException {
    xsw.writeStartDocument();
    xsw.writeStartElement("lista_cotizaciones");
    // Para cada elemento de la lista creamos un elemento dentro del fichero
    for (Map.Entry<String, Double> el : tabla.entrySet()) {
        xsw.writeStartElement("cotizacion");

        xsw.writeStartElement("empresa");
        xsw.writeCharacters(el.getKey());
        xsw.writeEndElement();

        xsw.writeStartElement("cotizaciones");
        xsw.writeCharacters(el.getValue().toString());
        xsw.writeEndElement();

        xsw.writeEndElement();
    }
    xsw.writeEndElement();
    xsw.writeEndDocument();
}
```

Listing 2: Código de la clase de `CotizacionesHandler`

```
import java.util.*;

import org.xml.sax.*;
import org.xml.sax.helpers.*;

// Source: https://www.journaldev.com/1198/java-sax-parser-example
public class CotizacionesHandler extends DefaultHandler {
    private Hashtable<String, Double> tabla = new Hashtable<String, Double>();

    private String empresa;
    private boolean leeEmpresa;
    // Obtenemos los car cteres le dos por el SAXParser en el m todo characters
       ()
    private StringBuilder data = null;

    private Double cotizacion;
    private boolean leeCotizacion;

    @Override
    public void startDocument() throws SAXException {
//        System.out.println("Comienza lectura de fichero...");
        empresa = "";
        leeEmpresa = false;
        cotizacion = 0.0;
        leeCotizacion = false;
    }

    @Override
    public void endDocument() throws SAXException {
//        System.out.println("Finaliza lectura de fichero...");
    }

    @Override
    public void startElement(String uri, String localName, String qName, Attributes
        attributes) throws SAXException {
//        System.out.println("Comienza el elmento: <" + qName + ">");
        if (qName.equals("empresa")) {
            // Es una empresa --> Guardar su valor en la variable temporal
            leeEmpresa = true;
        } else if (qName.equals("cotizaciones")) {
            // Es una cotizacion --> Asociar a la empresa que sea
            leeCotizacion = true;
        }
        // Inicializamos el contenedor donde se almacena la informaci n leida
        data = new StringBuilder();
    }

    @Override
    public void endElement(String uri, String localName, String qName) throws
       SAXException {
        // Nada
//        System.out.println("Fin del elmento: <" + qName + ">");
        if (leeEmpresa) {
            empresa = data.toString();
            leeEmpresa = false;
```

```
        } else if (leeCotizacion) {
            cotizacion = Double.parseDouble(data.toString());
            leeCotizacion = false;
        }
        if (qName.equalsIgnoreCase("cotizacion")) {
            // a adimos el par empresa - cotizaci n a la tabla en caso de que sea
                un fin de elemento
            tabla.put(empresa, cotizacion);
        }
    }

    @Override
    public void characters(char ch[], int start, int length) throws SAXException {
        data.append(new String(ch, start, length));
    }

    public Hashtable<String, Double> getTabla() {
        return tabla;
    }
}
```

Listing 3: Código del fichero *parser.jj*

```
options
{
    STATIC = false;
}

PARSER_BEGIN(Parser)

import java.io.*;
import java.net.*;
import java.util.*;
import java.lang.Math;
import javax.xml.parsers.*;
import javax.xml.stream.XMLOutputFactory;
import javax.xml.stream.XMLStreamException;
import javax.xml.stream.XMLStreamWriter;

public class Parser {

    // URL donde obtener empresas y cotizaciones "actuales"
    private final static String URL = "http://www.infobolsa.es/acciones/ibex35";
    // Fichero donde obtener empresas y cotizaciones "obsoletas"
    private final static String FICHERO = "cotizacion.xml";

    /// Funci n para leer una tabla de cotizaciones del FICHCotizacionesHandlerERO
    Hashtable<String, Double> leeTablaDeFichero(String fichero) {
        try {
            // Lectura del FICHERO utilizando SAXParser
            SAXParser saxParser = SAXParserFactory.newInstance().newSAXParser();
            CotizacionesHandler handler = new CotizacionesHandler();
            saxParser.parse(new File(fichero), handler);
            return handler.getTabla();
        } catch (Exception e) {
            // En caso de excepci n (no existe el fichero, etc) --> Tabla vac a
            return new Hashtable<String, Double> ();
        }
    }

    /// Funci n para escribir la tabla de cotizaciones en el FICHERO
    void escribeTablaEnFichero(Hashtable<String, Double> tabla, String nombre) {
        PrintStream out = null;
        XMLOutputFactory xof = XMLOutputFactory.newInstance();
        XMLStreamWriter xsw = null;
        try {
            // Escribir la tabla en el fichero FICHERO
            out = new PrintStream(nombre);
            xsw = xof.createXMLStreamWriter(out);
            // Escribimos el fichero
            escrituraFichero(xsw, tabla);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    /// Funci n interna para escribir la tabla de cotizaciones con un
       XMLStreamWriter
```

```
    private void escrituraFichero(XMLStreamWriter xsw, Hashtable<String, Double>
        tabla) throws XMLStreamException {
        xsw.writeStartDocument();
        xsw.writeStartElement("lista_cotizaciones");
        // Para cada elemento de la lista creamos un elemento dentro del fichero
        for (Map.Entry<String, Double> el : tabla.entrySet()) {
            xsw.writeStartElement("cotizacion");

            xsw.writeStartElement("empresa");
            xsw.writeCharacters(el.getKey());
            xsw.writeEndElement();

            xsw.writeStartElement("cotizaciones");
            xsw.writeCharacters(el.getValue().toString());
            xsw.writeEndElement();

            xsw.writeEndElement();
        }
        xsw.writeEndElement();
        xsw.writeEndDocument();
    }

    public static void main(String args []) {
        try {
            // Entrada de datos de la web
            InputStream stream = new URL(URL).openStream();
            Parser parser = new Parser(stream);
            Hashtable<String, Double> tablaCotizaciones = parser.
                leeTablaCotizaciones();

            // Entrada de datos de teclado
            System.out.println("Introduzca el nombre de una empresa del IBEX 35: ")
                ;
            Scanner scanner = new Scanner(System.in);
            String empresa = scanner.nextLine();

            // Entrada de datos de fichero
            Hashtable<String, Double> tablaAnterior = parser.leeTablaDeFichero(
                FICHERO);


            // Obtenemos las cotizaciones de la empresa indicada de la ejecuci n
                actual y de la anterior
            Double cotizacionAnterior = tablaAnterior.get(empresa);
            Double cotizacionActual = tablaCotizaciones.get(empresa);

            if (cotizacionActual != null){
                // La empresa indicada est  en la lista obtenida del parser
                if (cotizacionAnterior!= null) {
                    // Existe una cotizacion anterior
                    Double diferencia = cotizacionActual - cotizacionAnterior;
                    if (diferencia < 0.0){
                        System.out.println("La cotizacion ha bajado " + Math.abs(
                            diferencia) + " puntos.");
                    } else if(diferencia > 0.0){
                        System.out.println("La cotizacion ha subido " + diferencia
```

```
                        + " puntos.");
                }else{
                    System.out.println("La cotizacion no ha variado con
                        respecto a la ejecucion anterior");
                }
            }
            System.out.println("La cotizacion actual de la empresa '" + empresa
                + "' es: " + cotizacionActual);
        }else{
            System.out.println("No se encuentra informacion de la empresa " +
                empresa);
        }

        // Salvamos los resultados en fichero
        parser.escribeTablaEnFichero(tablaCotizaciones, FICHERO);
    } catch (Exception e) {
        System.out.println("Exception " + e.getMessage());
    } catch (Error e) {
        System.out.println("Error " + e.getMessage());
    }
}

}

PARSER_END(Parser)

SKIP :
{
    " "
    | "\r"
    | "\t"
    | "\n"
}

TOKEN :
{
    < CABECERA : "<!DOCTYPE html>" >
    | < HTML : "<html>" >
    | < HTML_FIN : "</html>" >
    | < HEAD : "<head>" >
    | < HEAD_FIN : "</head>" >
    | < BODY : "<body class=\"ifb-menu-push\">" >
    | < BODY_FIN : "</body>" >
    | < NOMBRE_EMPRESA : "<td class=\"name\">" >
    | < COTIZACION_EMPRESA_POS : "<td class=\"price top\">" >
    | < COTIZACION_EMPRESA_NEG : "<td class=\"price flop\">" >
    | < COTIZACION_EMPRESA_EQ : "<td class=\"price equal\">" >
    | < A_FIN : "</a>" >
    | < ETIQUETA : "<" >
    | < ETIQUETA_FIN : ">" >
    | < BARRA : "/" >
    | < NUMERO: (["0"-"9"])* ( "," )? (["0"-"9"])+ >
    | < CARACTERES : ([ "A"-"Z", "a"-"z", "0"-"9", "\u00c1", "\u00c9", "\u00cd", "\
        u00d3", "\u00da", "\u00dc", "\u00d1",
    "\u00e1", "\u00e9", "\u00ed", "\u00f3", "\u00fa", "\u00fc", "\u00f1", "\u00a1",
        "!", "\u20ac", ".", "\u00bf", "?", ":", ";",
```

```
    ","," ''", "=", "\"", "_", "-", "+", "\u00ba", "*", "(", ")", "\\", "@", "%", "#
        ", "&", "[",
    "]", "|", "{", "}", "$" ])+ >
    | < TABLE : "<table class=\"fullTable\" width=\"100%\" cellspacing=\"0\"
        cellpadding=\"0\" border=\"0\">" >
    | < TBODY : "<tbody>" >
    | < TABLE_FIN : "</table>" >
    | < TD_FIN : "</td>" >
    | < TROW : "<tr class=\"normal\">" >
    //| < TROW_FIN : "</tr>" >
}

/// Funcion para leer la tabla de cotizaciones de la p gina web
Hashtable<String, Double> leeTablaCotizaciones() :
{
    Hashtable<String, Double> tabla = null;
}
{
    <CABECERA> <HTML> <HEAD> saltar() <HEAD_FIN> tabla = body() <HTML_FIN>
    {
        return tabla;
    }
}

/// Funci n para saltar la parte no importante hasta la tabla (skip lo que sea -->
    Comod n)
void saltar() :
{}
{
    ( <CARACTERES> | <NUMERO> | <ETIQUETA> | <ETIQUETA_FIN> | <BARRA> | <A_FIN> | <
        TD_FIN> )*
}


/// Funci n para obtener el cuerpo de la tabla
Hashtable<String, Double> body() :
{
    Hashtable<String, Double> tabla = null;
}
{
    <BODY> saltar() tabla = cotizaciones() saltar() <BODY_FIN>
    {
        return tabla;
    }
}

/// Funci n para obtener la cotizaci n de la empresa con nombre = NOMBRE_EMPRESA
Hashtable<String, Double> cotizaciones() :
{
    Hashtable<String, Double> tabla = new Hashtable<String, Double> ();
    String empresa = null;
    double cotizacion = 0;
}
{
     <TABLE> saltar() <TBODY> ( <TROW> saltar() empresa = empresa() cotizacion =
         cotizacion()
```

```
            {
                tabla.put(empresa, cotizacion);
            }
        saltar()   )+ <TABLE_FIN>
    {
        return tabla;
    }
}

/// Funci n para obtener el nombre de la empresa
String empresa() :
{
    StringBuilder nombreCompleto = new StringBuilder();
    Token nombre = null;
}
{
    <NOMBRE_EMPRESA> <ETIQUETA> ( <CARACTERES> | <BARRA> )* <ETIQUETA_FIN>
    (
        nombre = <CARACTERES>
        {nombreCompleto.append(nombre.image);}
    )+
    <A_FIN><TD_FIN>
    {
        return nombreCompleto.toString();
    }
}

/// Funci n para obtener la cotizacion de una empresa
Double cotizacion() :
{
    Token cotizacion;
}
{
    ( <COTIZACION_EMPRESA_POS> | <COTIZACION_EMPRESA_NEG> | <COTIZACION_EMPRESA_EQ>
        )
    (
        cotizacion = <NUMERO>
    )
    <TD_FIN>
    {
        return Double.parseDouble(cotizacion.image.replace(",", "."));
    }
}
```