Parsing XML using JSR 172

The intention of this code snippet is to give an idea about how to parse an xml file using JSR 172.

- This example receives an URL of an XML file
- Opens an HttpConnection to the file to get the InputStream
- Uses the Input Stream as a means to create a SAXParser

When the SAXParser is successfully created, the application will use the callbacks to track the whole document.

```
import java.io.IOException;
import java.io.InputStream;
import javax.microedition.io.Connector;
import javax.microedition.io.HttpConnection;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.Attributes;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

/**
 *Class in Charge of http connectivity and xml parsing
 */
public class XMLParser extends DefaultHandler implements Runnable {

    private String URL;
    private HttpConnection http;
    StringBuffer log = new StringBuffer();

    /**
     * Starts connecting to the given URL to fetch the XML file.
     * The connection is started in a separate thread
     * @param URL
     */
    public void connect(String URL) {
        this.URL = URL;
        Thread thread = new Thread(this);
        thread.start();
    }

    /**
     * Opens the connection and fetches the xml file
     */
    public void run() {
        loadXML(getInputStream());
    }

    private InputStream getInputStream() {
        InputStream input = null;
        try {
            http = (HttpConnection) Connector.open(URL);

            // Getting the response code will open the connection,
            int rc = http.getResponseCode();
            if (rc == HttpConnection.HTTP_OK) {
                String type = http.getType();
                if (type.startsWith("text/xml")) {

                    input = http.openInputStream();
                }
            }
        } catch (IOException ex) {
        }
        return input;
    }

    /**
     * Starts parsing the received XML file using a SAXParser
     */
    public void loadXML(InputStream input) {

        SAXParser parser;
        try {
            SAXParserFactory factory =
                    SAXParserFactory.newInstance();
            parser =
                    factory.newSAXParser();

            InputSource fis = new InputSource(input);
            parser.parse(fis, this);

        } catch (Exception e) {
            e.printStackTrace();
        }

    }

    /**
     * Callback invoked when a start tag is loaded
     * @param uri
     * @param localName
     * @param qName
     * @param attributes
     * @throws org.xml.sax.SAXException
     */
    public void startElement(String uri, String localName, String qName, Attributes attributes) throws SAXException {
    }

    /**
     * Callback invoked when an end tag is found
     * @param uri
     * @param localName
     * @param qName   Nombre de la etiqueta
     * @throws org.xml.sax.SAXException
     */
    public void endElement(String uri, String localName, String qName)
```


(C) Copyright Nokia 2013. All rights reserved.

```
        throws SAXException {
    }

    /**
     * Callback invoked when the tag contains a text.
     * Text is saved according to the tag being analyzed
     * as the callback does not pass any reference to the current tag
     * a global variable currentTag is used to store the value.
     * Data is saved in the current visit
     * @param ch
     * @param start
     * @param length
     * @throws org.xml.sax.SAXException
     */
    public void characters(char[] ch, int start, int length) throws SAXException {
        String chars = new String(ch, start, length).trim();
        if (chars.length() > 0) {
        }
    }
}
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.