Fórum:GoBot.java (dokumentace)

Class **GoogleSearch** je objekt komunikující s Google API (http://code.google.com/apis/ajaxsearch/web.html). Jeho metoda **getThePage()** získává od http serveru Google API stránku objektů JSON, metoda **getTheSearch()** opakovaně vrací odkazy nalezené Googlem. Metodě **TheSearch()** je nutné nastavit proměnné **refererURL** a **apiKey** jako parametry komunikace s Google API (pro více informací kontaktujte autora botu).

Zdrojový kód

```
//Go
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */

package gobot;
import java.io.*;
import java.net.*;
import java.util.*;
//import java.util.Iterator;
import org.json.*; //http://www.json.org/java/
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import org.xml.sax.Attributes;
import org.xml.sax.helpers.DefaultHandler;
import org.xml.sax.helpers.XMLReaderFactory;


/**
 *
 * @author Pavel Dusek/GoBot
 */
public class Main {

    /**
     * @param args the command line arguments/ no command line arguments
     */
    public static void main(String[] args) {
    String link = new String();

    try {
            MyWikiSkriptaListAllPagesApiSax wikiListAllPagesAPI = new MyWikiSkriptaListAllPagesApiSax();
            //wikiListAllPagesAPI.listAllPages();
            MyWikiSkriptaGetContentApiSax wikiGetContentAPI = new MyWikiSkriptaGetContentApiSax("<Nejaky wikiclanek>");
            wikiGetContentAPI.getContent();
            TheSearch mySearch = new TheSearch("WikiSkripta");
            link = mySearch.getTheSearch();
    } catch (Exception excp) {
            excp.printStackTrace();
            return;
    }
    }

}

class TheSearch{
    String googleAPI = "http://ajax.googleapis.com/ajax/services/search/web?v=1.0&";
    String apiKey = "<API KEY>";
    String refererURL = "<referrer URL>";
    JSONObject jsonSearch;
    String search;
    String queryCriterion;
    String pageLink;
    String start;
    List<String> startList = new ArrayList();
    List<String> linkList = new ArrayList();


    TheSearch(String criterion){
        queryCriterion = criterion;
}


    String getThePage(String pageLink){
        //URLEncoder --- doplnit encoding
    String thePage = new String();

    try {
            URL u;
            u = new URL(pageLink);
            URLConnection connection;
            String s;
            connection = u.openConnection(); //throws IOException
            connection.addRequestProperty("Referer", this.refererURL);
            BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream()));

            while ( (s = reader.readLine()) != null) {
                thePage += s;
            }
            reader.close();

    } catch (MalformedURLException mue) {
            System.out.println("MalformedURLException…");
            mue.printStackTrace();
            System.exit(1);
        } catch (IOException ioe) {
            System.out.println("IOException…");
```

```java
            ioe.printStackTrace();
            System.exit(1);
    }
    return thePage;
    }

    String getTheSearch(){
        //pouziva org.json.*
        try {
            //inicializace parseru, zjištění možností hledání na daný dotaz od Google API
            String pageLinkParameters = "q=" + URLEncoder.encode(this.queryCriterion, "UTF-8") + "&key=" + this.apiKey;
            pageLink = this.googleAPI + pageLinkParameters;
            jsonSearch = new JSONObject(this.getThePage(pageLink));
            search = "<root>" + XML.toString(jsonSearch) + "</root>";

            MyGoogleSearchSAX GoogleSearchParser = new MyGoogleSearchSAX();
            GoogleSearchParser.xmlDocument = search;
            GoogleSearchParser.parse();
            GoogleSearchParser.list.clear();//parsovalo se kvuli <start> tagu na google, vyhledane odkazy nejsou v tuto chvili zadouci
            startList.addAll(GoogleSearchParser.startList);
            for (String start: startList)
            {
                pageLinkParameters = "q=" + URLEncoder.encode(this.queryCriterion, "UTF-8") + "&start=" + start + "&key=" + this.apiKey;
                pageLink = this.googleAPI + pageLinkParameters;
                jsonSearch = new JSONObject(this.getThePage(pageLink));
                search = "<root>" + XML.toString(jsonSearch) + "</root>";
                GoogleSearchParser.xmlDocument = search;
                GoogleSearchParser.parse();
            }
            linkList.addAll(GoogleSearchParser.list);
    } catch (UnsupportedEncodingException uee) {
            uee.printStackTrace();
            System.exit(1);
        } catch (JSONException jsone) {
            jsone.printStackTrace();
            System.exit(1);
    } catch (NoSuchElementException nsee) {
            nsee.printStackTrace();
            return "";
        } catch (Exception excp) {
            excp.printStackTrace();
        }
        return "";
    }
}

class MyGoogleSearchSAX extends DefaultHandler{
    public MyGoogleSearchSAX(){
        super();
    }

    String lastOpenedNode = new String();
    String value = new String();
    String googleStart = new String();
    String xmlDocument;
    List<String> startList = new ArrayList();
    List<String> list = new ArrayList();


    void parse()
        throws Exception
    {
        XMLReader xr = XMLReaderFactory.createXMLReader();
        xr.setContentHandler(this);
        xr.setErrorHandler(this);
        StringReader r = new StringReader(xmlDocument);
        xr.parse(new InputSource(r));
    }


   public void startElement(String uri, String name, String qname, Attributes att)
    {
       lastOpenedNode = name;
    }
   public void endElement(String uri, String name, String qname)
   {
       if (name.equals("url") && name.equals(lastOpenedNode))
           this.list.add(value);
       if (name.equals("start") && name.equals(lastOpenedNode))
           this.startList.add(value);
       value = new String();
   }
   public void characters(char[] ch, int start, int length)
   {
        value = value + new String(ch,start,length).trim();
   }
}

class MyWikiSkriptaListAllPagesApiSax extends DefaultHandler {
    List<String> allpages = new ArrayList();
    String xmlDocument = new String();
    String lastOpenedNode = new String();
    String value = new String();
    String page = new String();
    String cont = new String();
    boolean queryContinue = false;

    MyWikiSkriptaListAllPagesApiSax(){
        super();
    }
    public void listAllPages(){
        try{
            URL wikiAPI = new URL("https://www.wikiskripta.eu/api.php");
            URLConnection wc = wikiAPI.openConnection();
            wc.setDoOutput(true);
            OutputStreamWriter writer = new OutputStreamWriter(wc.getOutputStream());
            writer.write("action=query&list=allpages&format=xml");
            writer.close();
            BufferedReader reader = new BufferedReader(new InputStreamReader(wc.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null)
            {
                xmlDocument += line + '\n';
            }
            reader.close();
            this.parse(xmlDocument);

            while (!cont.equals("")){
                xmlDocument = new String();
                wc = wikiAPI.openConnection();
                wc.setDoOutput(true);
                writer = new OutputStreamWriter(wc.getOutputStream());
                writer.write("action=query&list=allpages&apfrom=" + cont + "&format=xml");
                writer.close();
                reader = new BufferedReader(new InputStreamReader(wc.getInputStream()));
                while ((line = reader.readLine()) != null){
                    xmlDocument += line + '\n';
                }
                reader.close();
                try
                {
                    this.parse(xmlDocument);
                } catch (org.xml.sax.SAXParseException spe)
                {
                    spe.printStackTrace();
                }
            }

        } catch (Exception excp) {
            excp.printStackTrace();
        }
    }
    public void parse(String xml)
           throws Exception
    {
        XMLReader xr = XMLReaderFactory.createXMLReader();
        xr.setContentHandler(this);
        xr.setErrorHandler(this);
        StringReader r = new StringReader(xml);
        xr.parse(new InputSource(r));
    }


   public void startElement(String uri, String name, String qname, Attributes att)
    {
       lastOpenedNode = name;
       if (name.equals("p"))
       {
           allpages.add(att.getValue("", "title"));
           queryContinue = false;
           cont = "";
       }
       if (name.equals("query-continue"))
           queryContinue = true;
       if (name.equals("allpages") && queryContinue)
           cont = att.getValue("", "apfrom");
    }
}

class MyWikiSkriptaGetContentApiSax extends DefaultHandler {
    String title;
    String content;
    String openedNode;
    String value;
    String xmlDocument = new String();
    List<String> sentences = new ArrayList();

    MyWikiSkriptaGetContentApiSax(String title)
    {
        super();
        this.title = title;
    }

    void getContent()
    {
        try
        {
            URL wikiAPI = new URL("https://www.wikiskripta.eu/api.php");
            URLConnection wc = wikiAPI.openConnection();
            wc.setDoOutput(true);
            OutputStreamWriter writer = new OutputStreamWriter(wc.getOutputStream());
            writer.write("action=query&prop=revisions&rvprop=content&titles=" + title + "&format=xml");
            writer.close();
            BufferedReader reader = new BufferedReader(new InputStreamReader(wc.getInputStream()));
            String line;
            while ((line = reader.readLine()) != null)
            {
                xmlDocument += line + '\n';
            }
            reader.close();
            this.parse(xmlDocument);
            System.out.println(content);
        } catch (Exception excp)
        {
            excp.printStackTrace();
        }
    }

    public void parse(String xml)
           throws Exception
    {
        XMLReader xr = XMLReaderFactory.createXMLReader();
        xr.setContentHandler(this);
        xr.setErrorHandler(this);
        StringReader r = new StringReader(xml);
        xr.parse(new InputSource(r));
    }
   public void startElement(String uri, String name, String qname, Attributes att)
    {
       openedNode = name;
    }
   public void endElement(String uri, String name, String qname)
   {
       if (name.equals("rev") && name.equals(openedNode))
       {
           openedNode = new String();
           content = value;
       }
       value = new String();
   }
   public void characters(char[] ch, int start, int length)
   {
        value = value + new String(ch,start,length).trim();
   }

}
```

Článek neobsahuje vše, co by měl.

Můžete se přidat k jeho autorům (https://www.wikiskripta.eu/index.php?title=F%C3%B3rum:GoBot.java_(dokumentace)&action=history) a jej.

O vhodných změnách se lze poradit v diskusi.