java实现爬虫爬网站图片 - javaxiaojian的专栏 - 博客频道

第一步，实现 LinkQueue，对url进行过滤和存储的操作

```
1. import java.util.ArrayList;
2. import java.util.Collections;
3. import java.util.HashSet;
4. import java.util.List;
5. import java.util.Set;
6. public class LinkQueue {
7.     // 已访问的 url 集合
8.     private static Set<String> visitedUrl = Collections.synchronizedSet(new HashSet<String>
   ());
9.     // 未访问的url
10.    private static List<String> unVisitedUrl = Collections.synchronizedList(new ArrayList<String
   >());
11.    // 未访问的URL出队列
12.    public static String unVisitedUrlDeQueue() {
13.        if (unVisitedUrl.size() > 0) {
14.            String url = unVisitedUrl.remove(0);
15.            visitedUrl.add(url);
16.            return url;
17.        }
18.        return null;
19.    }
20.    // 新的url添加进来的时候进行验证，保证只是添加一次
21.    public static void addUnvisitedUrl(String url) {
22.        if (url != null && !url.trim().equals("") && !visitedUrl.contains(url)
23.                && !unVisitedUrl.contains(url))
24.            unVisitedUrl.add(url);
25.    }
26.    // 判断未访问的URL队列中是否为空
27.    public static boolean unVisitedUrlsEmpty() {
28.        return unVisitedUrl.isEmpty();
```

```
    }
}
```

```
import java.util.HashSet;
import java.util.Set;
import org.htmlparser.Node;
import org.htmlparser.NodeFilter;
import org.htmlparser.Parser;
import org.htmlparser.filters.NodeClassFilter;
import org.htmlparser.filters.OrFilter;
import org.htmlparser.tags.LinkTag;
import org.htmlparser.util.NodeList;
import org.htmlparser.util.ParserException;
/**
 * 过滤http的url，获取可以符合规则的url
 * @author Administrator
 *
 */
public class ParserHttpUrl {
    // 获取一个网站上的链接,filter 用来过滤链接
    public static Set<String> extracLinks(String url, LinkFilter filter) {
        Set<String> links = new HashSet<String>();
        try {
            Parser parser = new Parser(url);
            // 过滤 <frame >标签的 filter，用来提取 frame 标签里的 src 属性所表示的链接
            NodeFilter frameFilter = new NodeFilter() {
                public boolean accept(Node node) {
                    if (node.getText().startsWith("frame src=")) {
                        return true;
                    } else {
                        return false;
                    }
                }
            };
            // OrFilter 来设置过滤 <a> 标签，和 <frame> 标签
            OrFilter linkFilter = new OrFilter(new NodeClassFilter(
                    LinkTag.class), frameFilter);
            // 得到所有经过过滤的标签
            NodeList list = parser.extractAllNodesThatMatch(linkFilter);
```

```
            for (int i = 0; i < list.size(); i++) {
                Node tag = list.elementAt(i);
                if (tag instanceof LinkTag)// <a> 标签
                {
                    LinkTag link = (LinkTag) tag;
                    String linkUrl = link.getLink();// url
                    if (filter.accept(linkUrl))
                        links.add(linkUrl);
                } else// <frame> 标签
                {
                    // 提取 frame 里 src 属性的链接如 <frame src="test.html"/>
                    String frame = tag.getText();
                    int start = frame.indexOf("src=");
                    frame = frame.substring(start);
                    int end = frame.indexOf(" ");
                    if (end == -1)
                        end = frame.indexOf(">");
                    String frameUrl = frame.substring(5, end - 1);
                    if (filter.accept(frameUrl))
                        links.add(frameUrl);
                }
            }
        } catch (ParserException e) {
            e.printStackTrace();
        }
        return links;
    }
}
```

```
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.List;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
/***
```

```
 * java抓取网络图片
 *
 * @author swinglife
 *
 */
public class DownLoadPic {
    // 编码
    private static final String ECODING = "UTF-8";
    // 获取img标签正则
    private static final String IMGURL_REG = "<img.*src=(.*?)[^>]*?>";
    // 获取src路径的正则
    private static final String IMGSRC_REG = "http:\"?(.*?)(\"|>|\\s+)";
    public static void downloadPic(String url) {
        // 获得html文本内容
        String HTML = null;
        try {
            HTML = DownLoadPic.getHTML(url);
        } catch (Exception e) {
            e.printStackTrace();
        }
        if (null != HTML && !"".equals(HTML)) {
            // 获取图片标签
            List<String> imgUrl = DownLoadPic.getImageUrl(HTML);
            // 获取图片src地址
            List<String> imgSrc = DownLoadPic.getImageSrc(imgUrl);
            // 下载图片
            DownLoadPic.download(imgSrc);
        }
    }
    /***
     * 获取HTML内容
     *
     * @param url
     * @return
     * @throws Exception
     */
    private static String getHTML(String url) throws Exception {
        URL uri = new URL(url);
```

```
49.         URLConnection connection = uri.openConnection();
50.         InputStream in = connection.getInputStream();
51.         byte[] buf = new byte[1024];
52.         int length = 0;
53.         StringBuffer sb = new StringBuffer();
54.         while ((length = in.read(buf, 0, buf.length)) > 0) {
55.             sb.append(new String(buf, ECODING));
56.         }
57.         in.close();
58.         return sb.toString();
59.     }
60.     /***
61.      * 获取ImageUrl地址
62.      *
63.      * @param HTML
64.      * @return
65.      */
66.     private static List<String> getImageUrl(String HTML) {
67.         Matcher matcher = Pattern.compile(IMGURL_REG).matcher(HTML);
68.         List<String> listImgUrl = new ArrayList<String>();
69.         while (matcher.find()) {
70.             listImgUrl.add(matcher.group());
71.         }
72.         return listImgUrl;
73.     }
74.     /***
75.      * 获取ImageSrc地址
76.      *
77.      * @param listImageUrl
78.      * @return
79.      */
80.     private static List<String> getImageSrc(List<String> listImageUrl) {
81.         List<String> listImgSrc = new ArrayList<String>();
82.         for (String image : listImageUrl) {
83.             Matcher matcher = Pattern.compile(IMGSRC_REG).matcher(image);
84.             while (matcher.find()) {
85.                 listImgSrc.add(matcher.group().substring(0,
86.                         matcher.group().length() - 1));
```

```
            }
        }
        return listImgSrc;
    }
    /***
     * 下载图片
     *
     * @param listImgSrc
     */
    private static void download(List<String> listImgSrc) {
        for (String url : listImgSrc) {
            try {
                String imageName = url.substring(url.lastIndexOf("/") + 1,
                        url.length());
                URL uri = new URL(url);
                InputStream in = uri.openStream();
                FileOutputStream fo = new FileOutputStream(new File(imageName));
                byte[] buf = new byte[1024];
                int length = 0;
                while ((length = in.read(buf, 0, buf.length)) != -1) {
                    fo.write(buf, 0, length);
                }
                in.close();
                fo.close();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }
}
```

```
public class Crawler {
    /**
     * 抓取过程
     *
     * @return
     * @param seeds
     */
    public void crawling(String url) { // 定义过滤器
```

```
        Filter filter = new Filter() {
            public boolean accept(String url) {
                //这里过滤规则随需要爬的网站的规则进行改变，推荐使用正则实现，本人是爬豆瓣网站
                if(url.indexOf("douban.com/group/topic") != -1 || url.indexOf("douban.com/group/haixiuzu/discussion?start") != -1 )
                    return true;
                else
                    return false;
            }
        };
        // 初始化 URL 队列
        LinkQueue.addUnvisitedUrl(url);
        // 循环条件，待抓取的链接不空
        while (!LinkQueue.unVisitedUrlsEmpty()) {
            // 队头URL出队列
            String visitUrl = (String) LinkQueue.unVisitedUrlDeQueue();
            if (visitUrl == null)
                continue;
            DownLoadPic.downloadPic(visitUrl);
            // 提取出下载网页中的 URL
            Set<String> links = ParserHttpUrl.extracLinks(visitUrl, filter);
            // 新的未访问的 URL 入队
            for (String link : links) {
                LinkQueue.addUnvisitedUrl(link);
            }
        }
    }
    // main 方法入口
    public static void main(String[] args) {
        Crawler crawler = new Crawler();
        crawler.crawling("http://www.douban.com/group/haixiuzu/discussion?start=0");
    }
}
```