三目鸟

爬虫入门 手写一个Java爬虫

本文内容 涞源于 罗刚 老师的 书籍 << 自己动手写网络爬虫一书 >> ;

本文将介绍 1: 网络爬虫的是做什么的? 2: 手动写一个简单的网络爬虫;

1: 网络爬虫是做什么的? 他的主要工作就是 跟据指定的url地址 去发送请求,获得响应, 然后解析响应 , 一方面从响应中查找出想要查找的数据,另一方面从响应中解析出新的URL路径,

然后继续访问,继续解析;继续查找需要的数据和继续解析出新的URL路径 .

这就是网络爬虫主要干的工作. 下面是流程图:

昵称：三目鸟
园龄：1年11个月
粉丝：19
关注：58
+加关注

< 2018年9月 >

日	一	二	三	四	五	六
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

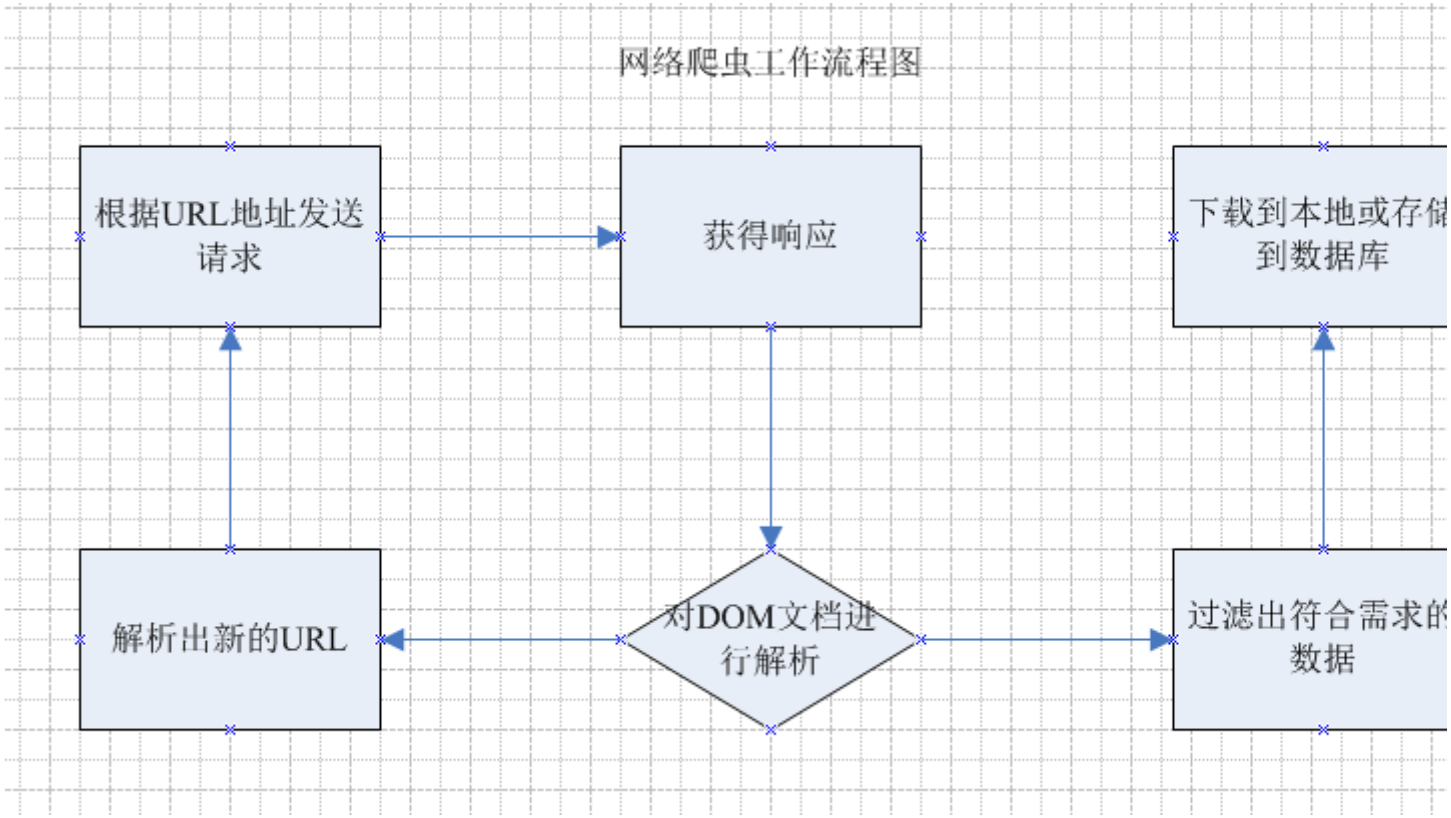
搜索

找找看

谷歌搜索

常用链接

我的随笔



通过上面的流程图 能大概了解到 网络爬虫 干了哪些活 ,根据这些 也就能设计出一个简单的网络爬虫出来.

一个简单的爬虫 必需的功能:

1: 发送请求和获取响应的功能 ;

2: 解析响应的功能 ;

3: 对 过滤出的数据 进行存储 的功能 ;

4: 对解析出来的URL路径 处理的功能 ;

下面是包结构:

我的评论
我的参与
最新评论
我的标签

我的标签

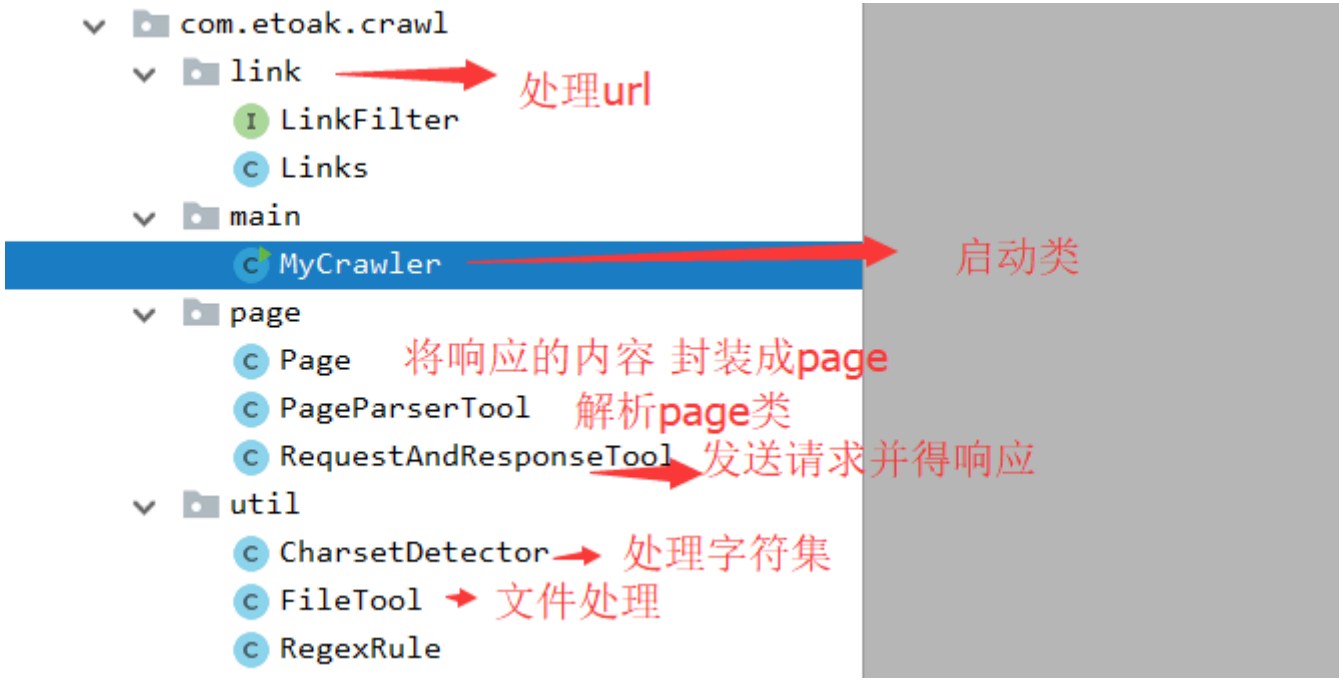
java 接口 对象(1)
win10 ubuntu 双系统 系统引导(1)
爬虫入门(1)

随笔分类

Java Web 入门(13)
java 基础知识(7)
Java 中级知识(1)
python web(1)
大数据入门(1)
爬虫相关
数据库(1)

随笔档案

2018年7月 (2)
2018年6月 (1)
2018年5月 (3)
2018年4月 (1)
2018年3月 (1)
2018年1月 (1)
2017年12月 (6)
2017年11月 (1)
2017年10月 (2)
2017年7月 (1)
2017年6月 (8)
2016年12月 (1)
2016年11月 (1)



下面就上代码：

RequestAndResponseTool 类： 主要方法： 发送请求 返回响应 并把 响应 封装成 page 类 ；

```
package com.etoak.crawl.page;

import org.apache.commons.httpclient.DefaultHttpMethodRetryHandler;
import org.apache.commons.httpclient.HttpClient;
import org.apache.commons.httpclient.HttpException;
import org.apache.commons.httpclient.HttpStatus;
import org.apache.commons.httpclient.methods.GetMethod;
import org.apache.commons.httpclient.params.HttpMethodParams;

import java.io.IOException;

public class RequestAndResponseTool {
```

文章分类

编程思想

相册

前端常用知识点图册(1)
双系统安装(2)

最新评论

1. Re:爬虫入门 手写一个Java爬虫
@Yinjiawei我把qq 私信 发给你哈 ....
--三目鸟

2. Re:爬虫入门 手写一个Java爬虫
大佬，可以给个联系方式么 和您请教请教=。=
--Yinjiawei

3. Re:爬虫入门 手写一个Java爬虫

大佬您好，我想爬这个歌词的信息（p标签下的），我应该怎么写cssSelector呀
--Yinjiawei

4. Re:爬虫入门 手写一个Java爬虫
感谢回答，我刚看了一下。更改了一下文件的路径。可能是一开始我的这个包放在了桌面的一个中文文件夹下。我看你写的储存路径就是包内。可能是识别问题？将整个文件剪切到d盘根目录下，运行正常。
--Tjiao

5. Re:爬虫入门 手写一个Java爬虫
@Tjiao您好，我觉得是 这个方法 写的比较丑陋;FileTool.getFileNameByUrl(String url, String contentType)你可以更改一下这个方法...
--三目鸟

```
    public static Page  sendRequstAndGetResponse(String url) {
        Page page = null;
        // 1.生成 HttpClinet 对象并设置参数
        HttpClient httpClient = new HttpClient();
        // 设置 HTTP 连接超时 5s
        httpClient.getHttpConnectionManager().getParams().setConnectionTimeout(5000);
        // 2.生成 GetMethod 对象并设置参数
        GetMethod getMethod = new GetMethod(url);
        // 设置 get 请求超时 5s
        getMethod.getParams().setParameter(HttpMethodParams.SO_TIMEOUT, 5000);
        // 设置请求重试处理
        getMethod.getParams().setParameter(HttpMethodParams.RETRY_HANDLER, new
DefaultHttpMethodRetryHandler());
        // 3.执行 HTTP GET 请求
        try {
            int statusCode = httpClient.executeMethod(getMethod);
        // 判断访问的状态码
            if (statusCode != HttpStatus.SC_OK) {
                System.err.println("Method failed: " + getMethod.getStatusLine());
            }
        // 4.处理 HTTP 响应内容
            byte[] responseBody = getMethod.getResponseBody();// 读取为字节 数组
            String contentType = getMethod.getResponseHeader("Content-Type").getValue(); // 得到当
前返回类型
            page = new Page(responseBody,url,contentType); //封装成为页面
        } catch (HttpException e) {
        // 发生致命的异常，可能是协议不对或者返回的内容有问题
            System.out.println("Please check your provided http address!");
            e.printStackTrace();
        } catch (IOException e) {
        // 发生网络异常
            e.printStackTrace();
        } finally {
        // 释放连接
            getMethod.releaseConnection();
        }
        return page;
```


阅读排行榜

1. 用 Eclipse 创建一个简单的web项目(37459)
2. 爬虫入门 手写一个Java爬虫(36080)
3. 怎样在Win10下安装ubuntu双系统(17266)
4. java实现 比较两个文本相似度-- java中文版 simHash 实现 ,(3491)
5. java web 入门级 开发 常用页面调试方法(2739)

评论排行榜

1. 爬虫入门 手写一个Java爬虫(7)
2. java实现 比较两个文本相似度-- java中文版 simHash 实现 ,(4)
3. java 多线程 Callable -- 分段处理一个大的list 然后再合并结果(1)

推荐排行榜

1. 爬虫入门 手写一个Java爬虫(7)
2. 用 Eclipse 创建一个简单的web项目(2)

```
    }
}
```

page 类： 主要作用： 保存响应的相关内容 对外提供访问方法；

```
package com.etoak.crawl.page;


import com.etoak.crawl.util.CharsetDetector;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;

import java.io.UnsupportedEncodingException;

/*
* page
*   1: 保存获取到的响应的相关内容;
* */
public class Page {

    private byte[] content ;
    private String html ;  //网页源码字符串
    private Document doc  ;//网页Dom文档
    private String charset ;//字符编码
    private String url ;//url路径
    private String contentType ;// 内容类型


    public Page(byte[] content , String url , String contentType){
        this.content = content ;
        this.url = url ;
        this.contentType = contentType ;
    }

    public String getCharset() {
        return charset;
```

```
}
public String getUrl(){return url ;}
public String getContentType(){ return contentType ;}
public byte[] getContent(){ return content ;}

/**
 * 返回网页的源码字符串
 *
 * @return 网页的源码字符串
 */
public String getHtml() {
    if (html != null) {
        return html;
    }
    if (content == null) {
        return null;
    }
    if(charset==null){
        charset = CharsetDetector.guessEncoding(content); // 根据内容来猜测 字符编码
    }
    try {
        this.html = new String(content, charset);
        return html;
    } catch (UnsupportedEncodingException ex) {
        ex.printStackTrace();
        return null;
    }
}

/*
*  得到文档
* */
public Document getDoc(){
    if (doc != null) {
        return doc;
    }
    try {
        this.doc = Jsoup.parse(getHtml(), url);
        return doc;
```

```
        } catch (Exception ex) {
            ex.printStackTrace();
            return null;
        }
    }


}
```

PageParserTool： 类 主要作用 提供了 根据选择器来选取元素 属性 等方法 ；

```
package com.etoak.crawl.page;

import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Set;

public class PageParserTool {


    /* 通过选择器来选取页面的 */
    public static Elements select(Page page , String cssSelector) {
        return page.getDoc().select(cssSelector);
    }

    /*
     *  通过css选择器来得到指定元素;
     *
     *  */
    public static Element select(Page page , String cssSelector, int index) {
        Elements eles = select(page , cssSelector);
        int realIndex = index;
        if (index < 0) {
```

```
            realIndex = eles.size() + index;
        }
        return eles.get(realIndex);
    }


    /**
     * 获取满足选择器的元素中的链接 选择器cssSelector必须定位到具体的超链接
     * 例如我们想抽取id为content的div中的所有超链接，这里
     * 就要将cssSelector定义为div[id=content] a
     *  放入set 中 防止重复；
     * @param cssSelector
     * @return
     */
    public static  Set<String> getLinks(Page page ,String cssSelector) {
        Set<String> links  = new HashSet<String>() ;
        Elements es = select(page , cssSelector);
        Iterator iterator  = es.iterator();
        while(iterator.hasNext()) {
            Element element = (Element) iterator.next();
            if ( element.hasAttr("href") ) {
                links.add(element.attr("abs:href"));
            }else if( element.hasAttr("src") ){
                links.add(element.attr("abs:src"));
            }
        }
        return links;
    }



    /**
     * 获取网页中满足指定css选择器的所有元素的指定属性的集合
     * 例如通过getAttrs("img[src]","abs:src")可获取网页中所有图片的链接
     * @param cssSelector
     * @param attrName
     * @return
     */
    public static ArrayList<String> getAttrs(Page page , String cssSelector, String attrName) {
```

```
        ArrayList<String> result = new ArrayList<String>();
        Elements eles = select(page ,cssSelector);
        for (Element ele : eles) {
            if (ele.hasAttr(attrName)) {
                result.add(ele.attr(attrName));
            }
        }
        return result;
    }
}
```

Link 包 ；

Links 类: 两个属性： 一个是存放 已经访问的url集合的set ；一个是存放待访问url集合的 queue ；

```
1  package com.etoak.crawl.link;
2
3  import java.util.HashSet;
4  import java.util.LinkedList;
5  import java.util.Set;
6
7  /*
8  *  Link主要功能;
9  *  1: 存储已经访问过的URL路径 和 待访问的URL 路径;
10 *
11 *
12 * */
13 public class Links {
14
15     //已访问的 url 集合  已经访问过的 主要考虑 不能再重复了 使用set来保证不重复;
16     private static Set visitedUrlSet = new HashSet();
17
18     //待访问的 url 集合  待访问的主要考虑 1:规定访问顺序;2:保证不提供重复的带访问地址;
19     private static LinkedList unVisitedUrlQueue = new LinkedList();
20
```

```
21      //获得已经访问的 URL 数目
22      public static int getVisitedUrlNum() {
23          return visitedUrlSet.size();
24      }
25
26      //添加到访问过的 URL
27      public static void addVisitedUrlSet(String url) {
28          visitedUrlSet.add(url);
29      }
30
31      //移除访问过的 URL
32      public static void removeVisitedUrlSet(String url) {
33          visitedUrlSet.remove(url);
34      }
35
36
37
38      //获得 待访问的 url 集合
39      public static LinkedList getUnVisitedUrlQueue() {
40          return unVisitedUrlQueue;
41      }
42
43      // 添加到待访问的集合中  保证每个 URL 只被访问一次
44      public static void addUnvisitedUrlQueue(String url) {
45          if (url != null && !url.trim().equals("")  && !visitedUrlSet.contains(url)  && !unVisitedUrlQue
46              unVisitedUrlQueue.add(url);
47          }
48      }
49
50      //删除 待访问的url
51      public static Object removeHeadOfUnVisitedUrlQueue() {
52          return unVisitedUrlQueue.removeFirst();
53      }
54
```

```
55      //判断未访问的 URL 队列中是否为空
56      public static boolean unVisitedUrlQueueIsEmpty() {
57          return unVisitedUrlQueue.isEmpty();
58      }
59
60  }
```

LinkFilter 接口： 可以起过滤作用；

```
package com.etoak.crawl.link;

public interface LinkFilter {
    public boolean accept(String url);
}
```

util 工具类

CharsetDetector 类： 获取字符编码

```
/*
 * Copyright (C) 2014 hu
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
```

```
package com.etoak.crawl.util;

import org.mozilla.universalchardet.UniversalDetector;

import java.io.UnsupportedEncodingException;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * 字符集自动检测
 *
 * @author hu
 */
public class CharsetDetector {

    //从Nutch借鉴的网页编码检测代码
    private static final int CHUNK_SIZE = 2000;

    private static Pattern metaPattern = Pattern.compile(
            "<meta\\s+([^>]*http-equiv=(\"|')?content-type(\"|')?[^>]*)>",
            Pattern.CASE_INSENSITIVE);
    private static Pattern charsetPattern = Pattern.compile(
            "charset=\\s*([a-z][_\\-0-9a-z]*)", Pattern.CASE_INSENSITIVE);
    private static Pattern charsetPatternHTML5 = Pattern.compile(
            "<meta\\s+charset\\s*=\\s*[\"']?([a-z][_\\-0-9a-z]*)[^>]*>",
            Pattern.CASE_INSENSITIVE);

    //从Nutch借鉴的网页编码检测代码
    private static String guessEncodingByNutch(byte[] content) {
        int length = Math.min(content.length, CHUNK_SIZE);

        String str = "";
        try {
            str = new String(content, "ascii");
        } catch (UnsupportedEncodingException e) {
            return null;
        }

        Matcher metaMatcher = metaPattern.matcher(str);
```

```
        String encoding = null;
        if (metaMatcher.find()) {
            Matcher charsetMatcher = charsetPattern.matcher(metaMatcher.group(1));
            if (charsetMatcher.find()) {
                encoding = new String(charsetMatcher.group(1));
            }
        }
        if (encoding == null) {
            metaMatcher = charsetPatternHTML5.matcher(str);
            if (metaMatcher.find()) {
                encoding = new String(metaMatcher.group(1));
            }
        }
        if (encoding == null) {
            if (length >= 3 && content[0] == (byte) 0xEF
                    && content[1] == (byte) 0xBB && content[2] == (byte) 0xBF) {
                encoding = "UTF-8";
            } else if (length >= 2) {
                if (content[0] == (byte) 0xFF && content[1] == (byte) 0xFE) {
                    encoding = "UTF-16LE";
                } else if (content[0] == (byte) 0xFE
                        && content[1] == (byte) 0xFF) {
                    encoding = "UTF-16BE";
                }
            }
        }

        return encoding;
    }

    /**
     * 根据字节数组，猜测可能的字符集，如果检测失败，返回utf-8
     *
     * @param bytes 待检测的字节数组
     * @return 可能的字符集，如果检测失败，返回utf-8
     */
    public static String guessEncodingByMozilla(byte[] bytes) {
        String DEFAULT_ENCODING = "UTF-8";
        UniversalDetector detector = new UniversalDetector(null);
```

```
        detector.handleData(bytes, 0, bytes.length);
        detector.dataEnd();
        String encoding = detector.getDetectedCharset();
        detector.reset();
        if (encoding == null) {
            encoding = DEFAULT_ENCODING;
        }
        return encoding;
    }

    /**
     * 根据字节数组，猜测可能的字符集，如果检测失败，返回utf-8
     * @param content 待检测的字节数组
     * @return 可能的字符集，如果检测失败，返回utf-8
     */
    public static String guessEncoding(byte[] content) {
        String encoding;
        try {
            encoding = guessEncodingByNutch(content);
        } catch (Exception ex) {
            return guessEncodingByMozilla(content);
        }

        if (encoding == null) {
            encoding = guessEncodingByMozilla(content);
            return encoding;
        } else {
            return encoding;
        }
    }
}
```

FileTool 文件下载类：

```
package com.etoak.crawl.util;
```

```
import com.etoak.crawl.page.Page;

import java.io.DataOutputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;

/*  本类主要是 下载那些已经访问过的文件*/
public class FileTool {

    private static String dirPath;


    /**
     * getMethod.getResponseHeader("Content-Type").getValue()
     * 根据 URL 和网页类型生成需要保存的网页的文件名，去除 URL 中的非文件名字符
     */
    private static String getFileNameByUrl(String url, String contentType) {
        //去除 http://
        url = url.substring(7);
        //text/html 类型
        if (contentType.indexOf("html") != -1) {
            url = url.replaceAll("[\\?/:*|<>\"]", "_") + ".html";
            return url;
        }
        //如 application/pdf 类型
        else {
            return url.replaceAll("[\\?/:*|<>\"]", "_") + "." +
                    contentType.substring(contentType.lastIndexOf("/") + 1);
        }
    }

    /*
    *  生成目录
    * */
    private static void mkdir() {
        if (dirPath == null) {
            dirPath = Class.class.getClass().getResource("/").getPath() + "temp\\";
```

```
        }
        File fileDir = new File(dirPath);
        if (!fileDir.exists()) {
            fileDir.mkdir();
        }
    }

    /**
     * 保存网页字节数组到本地文件，filePath 为要保存的文件的相对地址
     */

    public static void saveToLocal(Page page) {
        mkdir();
        String fileName =  getFileNameByUrl(page.getUrl(), page.getContentType()) ;
        String filePath = dirPath + fileName ;
        byte[] data = page.getContent();
        try {
            //Files.lines(Paths.get("D:\\jd.txt"),
StandardCharsets.UTF_8).forEach(System.out::println);
            DataOutputStream out = new DataOutputStream(new FileOutputStream(new File(filePath)));
            for (int i = 0; i < data.length; i++) {
                out.write(data[i]);
            }
            out.flush();
            out.close();
            System.out.println("文件："+ fileName + "已经被存储在"+ filePath  );
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

RegexRule 正则表达式类：

```
/*
 * Copyright (C) 2014 hu
```

```
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version 2
 * of the License, or (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA  02111-1307, USA.
 */
package com.etoak.crawl.util;


import java.util.ArrayList;
import java.util.regex.Pattern;

/**
 *
 * @author hu
 */
public class RegexRule {

    public RegexRule(){

    }
    public RegexRule(String rule){
        addRule(rule);
    }

    public RegexRule(ArrayList<String> rules){
        for (String rule : rules) {
            addRule(rule);
        }
    }
```

```
public boolean isEmpty(){
    return positive.isEmpty();
}

private ArrayList<String> positive = new ArrayList<String>();
private ArrayList<String> negative = new ArrayList<String>();



/**
 * 添加一个正则规则 正则规则有两种，正正则和反正则
 * URL符合正则规则需要满足下面条件： 1.至少能匹配一条正正则 2.不能和任何反正则匹配
 * 正正则示例：+a.*c是一条正正则，正则的内容为a.*c，起始加号表示正正则
 * 反正则示例：-a.*c时一条反正则，正则的内容为a.*c，起始减号表示反正则
 * 如果一个规则的起始字符不为加号且不为减号，则该正则为正正则，正则的内容为自身
 * 例如a.*c是一条正正则，正则的内容为a.*c
 * @param rule 正则规则
 * @return 自身
 */
public RegexRule addRule(String rule) {
    if (rule.length() == 0) {
        return this;
    }
    char pn = rule.charAt(0);
    String realrule = rule.substring(1);
    if (pn == '+') {
        addPositive(realrule);
    } else if (pn == '-') {
        addNegative(realrule);
    } else {
        addPositive(rule);
    }
    return this;
}



/**
```

```
     * 添加一个正正则规则
     * @param positiveregex
     * @return 自身
     */
    public RegexRule addPositive(String positiveregex) {
        positive.add(positiveregex);
        return this;
    }


    /**
     * 添加一个反正则规则
     * @param negativeregex
     * @return 自身
     */
    public RegexRule addNegative(String negativeregex) {
        negative.add(negativeregex);
        return this;
    }


    /**
     * 判断输入字符串是否符合正则规则
     * @param str 输入的字符串
     * @return 输入字符串是否符合正则规则
     */
    public boolean satisfy(String str) {

        int state = 0;
        for (String nregex : negative) {
            if (Pattern.matches(nregex, str)) {
                return false;
            }
        }

        int count = 0;
        for (String pregex : positive) {
            if (Pattern.matches(pregex, str)) {
                count++;
```

```
            }
        }
        if (count == 0) {
            return false;
        } else {
            return true;
        }

    }
}
```

主类：

MyCrawler ：

```
1  package com.etoak.crawl.main;
2
3  import com.etoak.crawl.link.LinkFilter;
4  import com.etoak.crawl.link.Links;
5  import com.etoak.crawl.page.Page;
6  import com.etoak.crawl.page.PageParserTool;
7  import com.etoak.crawl.page.RequestAndResponseTool;
8  import com.etoak.crawl.util.FileTool;
9  import org.jsoup.select.Elements;
10
11 import java.util.Set;
12
13 public class MyCrawler {
14
15     /**
16      * 使用种子初始化 URL 队列
17      *
18      * @param seeds 种子 URL
19      * @return
20      */
```

```
21      private void initCrawlerWithSeeds(String[] seeds) {
22          for (int i = 0; i < seeds.length; i++){
23              Links.addUnvisitedUrlQueue(seeds[i]);
24          }
25      }
26
27
28      /**
29       * 抓取过程
30       *
31       * @param seeds
32       * @return
33       */
34      public void crawling(String[] seeds) {
35
36          //初始化 URL 队列
37          initCrawlerWithSeeds(seeds);
38
39          //定义过滤器，提取以 http://www.baidu.com 开头的链接
40          LinkFilter filter = new LinkFilter() {
41              public boolean accept(String url) {
42                  if (url.startsWith("http://www.baidu.com"))
43                      return true;
44                  else
45                      return false;
46              }
47          };
48
49          //循环条件：待抓取的链接不空且抓取的网页不多于 1000
50          while (!Links.unVisitedUrlQueueIsEmpty()  && Links.getVisitedUrlNum() <= 1000) {
51
52              //先从待访问的序列中取出第一个；
53              String visitUrl = (String) Links.removeHeadOfUnVisitedUrlQueue();
54              if (visitUrl == null){
```

```
55                  continue;
56              }
57
58              //根据URL得到page;
59              Page page = RequestAndResponseTool.sendRequstAndGetResponse(visitUrl);
60
61              //对page进行处理：  访问DOM的某个标签
62              Elements es = PageParserTool.select(page,"a");
63              if(!es.isEmpty()){
64                  System.out.println("下面将打印所有a标签：  ");
65                  System.out.println(es);
66              }
67
68              //将保存文件
69              FileTool.saveToLocal(page);
70
71              //将已经访问过的链接放入已访问的链接中；
72              Links.addVisitedUrlSet(visitUrl);
73
74              //得到超链接
75              Set<String> links = PageParserTool.getLinks(page,"img");
76              for (String link : links) {
77                  Links.addUnvisitedUrlQueue(link);
78                  System.out.println("新增爬取路径: " + link);
79              }
80          }
81      }
82
83
84      //main 方法入口
85      public static void main(String[] args) {
86          MyCrawler crawler = new MyCrawler();
87          crawler.crawling(new String[]{"http://www.baidu.com"});
88      }
```

```
89  }
```

运行结果：

源码下载链接： https://pan.baidu.com/s/1ge7Nkzx 下载密码： mz5b

文章主要参考： 1： 自己动手写网络爬虫；
2： https://github.com/CrawlScript/WebCollector
WebCollector是一个无须配置、便于二次开发的JAVA爬虫框架（内核），它提供精简的的API，只需少量代码即可实现一个功能强大的爬虫。WebCollector-Hadoop是WebCollector的Hadoop版本，支持分布式爬取。

自己动手写网络爬虫 编辑

《自己动手写网络爬虫》是2010年清华大学出版社出版的图书，作者是罗刚。

书　名	自己动手写网络爬虫	出版社	清华大学出版社
作　者	罗刚	出版时间	2010-10-1
ISBN	9787302236474	装　帧	平装
页　数	346	开　本	16
定　价	43.00元	版　次	1
		字　数	535000

分类: Java Web 入门

标签: 爬虫入门

好文要顶 关注我 收藏该文

三目鸟
关注 - 58
粉丝 - 19
+加关注

7 0

« 上一篇：java web 入门级 开发 常用页面调试方法
» 下一篇：spring 多线程 写入数据库 和 写入 xml文件

posted @ 2017-11-18 17:38 三目鸟 阅读(36085) 评论(7) 编辑 收藏

评论

#1楼 2018-08-06 09:53 | Tjiao

您好 我这个运行后报了找不到文件路径我想问一下 这个路径是自己写的么？要在哪修改啊a
java.io.FileNotFoundException:
C:\Users\%e4%b9%9d\Desktop\%e6%9f%9c%e5%ad%90\crawl\target\classes\temp\www.baidu.com.ht
ml (系统找不到指定的路径。)
at java.io.FileOutputStream.open0(Native Method)
at java.io.FileOutputStream.open(FileOutputStream.java:270)
at java.io.FileOutputStream.<init>(FileOutputStream.java:213)
at java.io.FileOutputStream.<init>(FileOutputStream.java:162)
at com.etoak.crawl.util.FileTool.saveToLocal(FileTool.java:61)
at com.etoak.crawl.main.MyCrawler.crawling(MyCrawler.java:69)
at com.etoak.crawl.main.MyCrawler.main(MyCrawler.java:87)
新增爬取路径: http://www.baidu.com/img/gs.gif
新增爬取路径: http://www.baidu.com/img/bd_logo1.png
java.io.FileNotFoundException:
C:\Users\%e4%b9%9d\Desktop\%e6%9f%9c%e5%ad%90\crawl\target\classes\temp\www.baidu.com_i
mg_gs.gif.gif (系统找不到指定的路径。)
at java.io.FileOutputStream.open0(Native Method)
at java.io.FileOutputStream.open(FileOutputStream.java:270)
at java.io.FileOutputStream.<init>(FileOutputStream.java:213)
at java.io.FileOutputStream.<init>(FileOutputStream.java:162)
at com.etoak.crawl.util.FileTool.saveToLocal(FileTool.java:61)
at com.etoak.crawl.main.MyCrawler.crawling(MyCrawler.java:69)
at com.etoak.crawl.main.MyCrawler.main(MyCrawler.java:87)

支持(0) 反对(0)

#2楼[楼主] 2018-08-07 09:38 | 三目鸟

@ Tjiao
您好 从报错信息看 这个路径确实不是正确的路径.,

支持(0) 反对(0)

#3楼[楼主] 2018-08-07 10:06 | 三目鸟

@ Tjiao
您好，我觉得是 这个方法 写的比较丑陋;

FileTool.getFileNameByUrl(String url, String contentType)
你可以更改一下这个方法

支持(0) 反对(0)

#4楼 2018-08-07 10:42 | Tjiao

感谢回答，我刚看了一下。更改了一下文件的路径。可能是一开始我的这个包放在了桌面的一个中文文件夹下。我看你写的储存路径就是包内。可能是识别问题？将整个文件剪切到d盘根目录下，运行正常。

支持(0) 反对(0)

#5楼 2018-08-09 15:52 | Yinjiawei

大佬您好，我想爬这个歌词的信息（p标签下的），我应该怎么写cssSelector呀

支持(0) 反对(0)

#6楼 2018-08-09 15:56 | Yinjiawei

大佬，可以给个联系方式么 和您请教请教=。=

支持(0) 反对(0)

#7楼[楼主] 2018-08-19 10:25 | 三目鸟

@ Yinjiawei
我把qq 私信 发给你哈 .

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

【推荐】超50万VC++源码：大型组态工控、电力仿真CAD与GIS源码库！
【免费】要想入门学习Linux系统技术，你应该先选择一本适合自己的书籍
【推荐】企业SaaS应用开发实战，快速构建企业运营/运维系统
【推荐】ActiveReports 报表控件，全面满足 .NET开发需求

最新IT新闻:
· 京东物流杀入C端快递 北上广三城试点揽件
· 享骑电单车故障频发投诉无门 享受骑行还是危险骑行
· 马云展望"后阿里时代"：筹划投身慈善 希望"回去教书"
· 特斯拉Model 3排美国八月销量榜第五 月销两万辆
· 打车服务公司Lyft推出首批电动滑板车 由小米生产
» 更多新闻...

最新知识库文章:
· 如何招到一个靠谱的程序员
· 一个故事看懂"区块链"
· 被踢出去的用户
· 成为一个有目标的学习者
· 历史转折中的"杭派工程师"

» 更多知识库文章...

Copyright ©2018 三目鸟