转载链接：http://www.xdemo.org/jsoup-html-parse/

Jsoup应该说是最简单快速的Html解析程序了，完善的API以及与JS类似的操作方式，为Java的Html解析带来极大的方便，结合多线程适合做一些网络数据的抓取，本文从一下几个方面介绍一下，篇幅有限，化繁为简。

下载Jsouphttp://jsoup.org/download

查看官方提供的手册：http://jsoup.org/cookbook/

教程：https://my.oschina.net/flashsword/blog?catalog=390084

官方文档中文版：http://www.open-open.com/jsoup/parsing-a-document.htm

1. 获取一个Document，这是Jsoup最核心的一个对象

   有三种途径来加载Document：字符串，URL地址，文件

```
/**
 *
 */
package org.xdemo.example.jsoupdemo.input;

import java.io.File;
import java.io.IOException;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.safety.Whitelist;

/**
 * @作者 Goofy
 * @邮件 252878950@qq.com
 * @日期 2014-4-2上午10:54:53
 * @描述
 */
public class ParseDocument {

    /**
     * 将String转换成Document
     * @return org.jsoup.nodes.Document
     */
    public static Document parseHtmlFromString(){
        String html = "<html><head><title>标题</title></head>"
                + "<body><p>段落</p></body></html>";
        Document doc = Jsoup.parse(html);
        return doc;
    }

    /**
     * 注意：这是一个不安全的方法
     * 将String转换成Html片段,注意防止跨站脚本攻击
     * @return Element
     */
    public static Element parseHtmlFragmentFromStringNotSafe(){
        String html = "<div><p>Lorem ipsum.</p>";
        Document doc = Jsoup.parseBodyFragment(html);
        Element body = doc.body();
        return body;
    }
```

```
44
45          /**
46           * 这是一个安全的方法
47           * 将String转换成Html片段,注意防止跨站脚本攻击
48           * @return Element
49           */
50          public static Element parseHtmlFragmentFromStringSafe(){
51              String html = "<div><p>Lorem ipsum.</p>";
52              //白名单列表定义了哪些元素和属性可以通过清洁器，其他的元素和属性一律移除
53              Whitelist wl=new Whitelist();
54              //比较松散的过滤，包括
55              //"a", "b", "blockquote", "br", "caption", "cite", "code", "col",
56              //"colgroup", "dd", "div", "dl", "dt", "em", "h1", "h2", "h3", "h4", "h5", "h6"
57              //"i", "img", "li", "ol", "p", "pre", "q", "small", "strike", "strong",
58              //"sub", "sup", "table", "tbody", "td", "tfoot", "th", "thead", "tr", "u",
59              //"ul"
60              Whitelist.relaxed();
61              //没有任何标签，只有文本
62              Whitelist.none();
63              //常规的过滤器
64              //"a", "b", "blockquote", "br", "cite", "code", "dd", "dl", "dt", "em",
65              //"i", "li", "ol", "p", "pre", "q", "small", "strike", "strong", "sub",
66              //"sup", "u", "ul"
67              Whitelist.basic();
68              //常规的过滤器，多了一个img标签
69              Whitelist.basicWithImages();
70              //文本类型的标签
71              //"b", "em", "i", "strong", "u"
72              Whitelist.simpleText();
73              //另外还可以自定义过滤规则,例如
74              wl.addTags("a");
75              //执行过滤
76              Jsoup.clean(html, wl);
77              Document doc = Jsoup.parseBodyFragment(html);
78              Element body = doc.body();
79              return body;
80          }
81
82          /**
83           * 从URL加载
84           * @return Document
85           */
86          public static Document parseDocumentFromUrl(){
87              Document doc = null;
88              try {
89                  doc = Jsoup.connect("http://www.google.com/").get();
90                  //获取标题
91                  String title = doc.title();
92                  System.out.println(title);//输出：Google
93                  //data(key,value)是该URL要求的参数
94                  //userAgent制定用户使用的代理类型
95                  //cookie带上cookie，如cookie("JSESSIONID","FDE234242342342423432432")
96                  //连接超时时间
97                  //post或者get方法
98                  doc = Jsoup.connect("http://www.xxxxx.com/")
99                            .data("query", "Java")
100                           .userAgent("Mozilla")
101                           .cookie("auth", "token")
102                           .timeout(3000)
103                           .post();
104
105             } catch (IOException e) {
106                 e.printStackTrace();
107             }
108             return doc;
109         }
110         /**
111          * 从文件加载
112          * @return Document
113          */
114         public static Document parseDocumentFromFile(){
115             File input = new File("/tmp/input.html");
116             Document doc=null;
117             try {
118                 //从文件加载Document文档
```

```
            doc = Jsoup.parse(input, "UTF-8");
            System.out.println(doc.title());
        } catch (IOException e) {
            e.printStackTrace();
        }
        return doc;
    }


}
```

2.选择器

```
package org.xdemo.example.jsoupdemo.extracter;

import java.util.regex.Pattern;

import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;

/**
 * @作者 Goofy
 * @邮件 252878950@qq.com
 * @日期 2014-4-2上午10:41:19
 * @描述 选择器 操作示例
 */
public class Selector {

public static void main(String[] args) {
Document doc;
try {
//获取文档
doc=Jsoup.connect("http://xxx.com/").get();

/*****获取单一元素******/
//与JS类似的根据ID选择的选择器<div id="content"></div>
Element content = doc.getElementById("content");

/*****一下方法的返回值都是Elements集合******/

//获取所有的a标签<a href="#"></a>
content.getElementsByTag("a");
//类选择器<div></div>
doc.getElementsByClass("divClass");
//获取Document的所有元素
doc.getAllElements();
//根据属性获取元素<a href="#"></a>
doc.getElementsByAttribute("href");
//根据属性前缀获取元素 <li data-name="Peter Liu" data-city="ShangHai" data-lang="CSharp"
doc.getElementsByAttributeStarting("data-");
//根据key-value选择如<a href="http://xdemo.org"></a>
doc.getElementsByAttributeValue("href","http://xdemo.org");
//和上面的正好相反
doc.getElementsByAttributeValueNot("href","http://xdemo.org");
//根据key-value,其中value可能是key对应属性的一个子字符串，选择如<a href="http://xdemo.org">
doc.getElementsByAttributeValueContaining("href", "xdemo");
//根据key-value,其中key对应值的结尾是value，选择如<a href="http://xdemo.org"></a>
doc.getElementsByAttributeValueEnding("href", "org");
//和上面的正好相反
doc.getElementsByAttributeValueStarting("href","http://xdemo");
//正则匹配，value需要满足正则表达式，<a href="http://xdemo.org"></a>,如href的值含有汉字
doc.getElementsByAttributeValueMatching("href",Pattern.compile("[\u4e00-\u9fa5]"));
//同上
doc.getElementsByAttributeValueMatching("href", "[\u4e00-\u9fa5]");
//根据元素所在的z-index获取元素
doc.getElementsByIndexEquals(0);
```

```
//获取z-index大于x的元素
doc.getElementsByIndexGreaterThan(0);
//和上面的正好相反
doc.getElementsByIndexLessThan(10);

//遍历标签
for (Element link : content.getElementsByTag("a")) {
 String linkHref = link.attr("href");
 String linkText = link.text();
}

/**************一些其他常用的方法**************/
//获取网页标题
doc.title();
//获取页面的所有文本
doc.text();

//为元素添加一个css class
content.addClass("newClass");
//根据属性获取值
content.attr("id");
//获取所有子元素
content.children();
//获取元素内的所有文本
content.text();
//获取同级元素
content.siblingElements();


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

}

}
```

3.最后说一点，就是安全问题，解析html的时候要防止跨站脚本攻击cross-site scripting (XSS)，作者也考虑到了这一点，所以真正使用时候需要注意。

[转载请注明来源:http://www.xdemo.org/jsoup-html-parse/]