SOLR 技术文档

1. 了解 lucene 原理,全文搜索概念,参考(http://wishlife.javaeye.com/category/30179) .建立自己的索引库.
2. 了解 solr 参考(http://www.ibm.com/developerworks/cn/java/j-solr1/, http://www.ibm.com/developerworks/cn/java/j-solr2/).并下载实例程序.
3. 搭建 SOLR 服务器
   3.1 官方下载 apache-solr-1.3.0.zip 和 tomcat5.5
   3.2 将 apache-solr-1.3.0\example\webapps\solr.war 部署有 tomcat 下
   3.3 设置 solr 环境变量 apache-tomcat-5.5.26\conf\Catalina\localhost\下新建 solr.xml 文件内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
 <Context docBase="" debug="0" crossContext="true" >
    <Environment name="solr/home" type="java.lang.String" value="e:/tomcat/solr"
        override="true" />
</Context>
```

   设置 e:/tomcat/solr 为 solr 主目录 ,建立文件夹.
   3.4 将 apache-solr-1.3.0\example\solr 下所有文件复制到 e:/tomcat/solr 下
   3.5 启动 tomcat -> 浏览 http://localhost:8080/solr/admin/ 能访问 Solr Admin 页面说明 Solr 服务器设置成功.
4. 为 Solr 创建索引库
   4.1 在 e:/tomcat/solr 目录下新建名为 data 的文件夹,再在 data 下新建 index 名为 文件夹
   4.2 将 lucene 创建好的索引放入 e:/tomcat/solr/data/index 下
5. Solr 索引设置
   5.1 在 e:/tomcat/solr /conf 下 solrconfig.xml, schema.xml <2>中的技术文档有详细说明
   5.2 中文支持,如果你的索引要支持中文搜索的话,在此推荐庖丁分词,参考 (http://www.javaeye.com/topic/110148) schema.xml 设置如下:

中文词组分词

```
<fieldtype name="text" class="solr.TextField" positionIncrementGap="100">
 创建索引时
<analyzer type="index">
                    <tokenizer
class="com.chuangs.fulltextsearch.analyzer.ChineseTokenizerFactory"
mode="most-words"/>
                </analyzer>
```

搜索时

```
<analyzer type="query">
                     <tokenizer
class="com.chuangs.fulltextsearch.analyzer.ChineseTokenizerFactory"
mode="most-words"/>
                </analyzer>
        </fieldtype>
```

中文单词分词

```
        <fieldtype name="word" class="solr.TextField" positionIncrementGap="100">
```

```
                <analyzer type="index">
                    <tokenizer
class="org.apache.solr.analysis.StandardTokenizerFactory"/>
                </analyzer>
        </fieldtype>
```

相关搜索 field 设置

词组

```
<field name="XXXX " type="text" indexed="true" stored="true" multiValued="true"
omitNorms="true"/>
```

单词

```
<field name=" XXXX Word" type="word" indexed="true" stored="true" multiValued="true"
omitNorms="true"/>
```

注: com.fulltextsearch.analyzer.ChineseTokenizerFactory 是用 solr 封装的庖丁分词,如下:

```
import java.io.Reader;
import java.util.Map;

import net.paoding.analysis.analyzer.PaodingTokenizer;
import net.paoding.analysis.analyzer.TokenCollector;
import net.paoding.analysis.analyzer.impl.MaxWordLengthTokenCollector;
import net.paoding.analysis.analyzer.impl.MostWordsTokenCollector;
import net.paoding.analysis.knife.PaodingMaker;

import org.apache.lucene.analysis.TokenStream;
import org.apache.solr.analysis.BaseTokenizerFactory;

public class ChineseTokenizerFactory extends BaseTokenizerFactory {
    /**
     * 最多切分 默认模式
     */
    public static final String MOST_WORDS_MODE = "most-words";

    /**
     * 按最大切分
     */
    public static final String MAX_WORD_LENGTH_MODE = "max-word-length";

    private String mode = null;

    public void setMode(String mode) {
        if (mode == null || MOST_WORDS_MODE.equalsIgnoreCase(mode)
                || "default".equalsIgnoreCase(mode)) {
            this.mode = MOST_WORDS_MODE;
        } else if (MAX_WORD_LENGTH_MODE.equalsIgnoreCase(mode)) {
            this.mode = MAX_WORD_LENGTH_MODE;
```

```
            } else {
                throw new IllegalArgumentException("不合法的分析器 Mode 参数设置:" +
    mode);
            }
        }

        @Override
        public void init(Map args) {
            super.init(args);
            setMode(args.get("mode").toString());
        }

        public TokenStream create(Reader input) {
            return new PaodingTokenizer(input, PaodingMaker.make(),
                    createTokenCollector());
        }

        private TokenCollector createTokenCollector() {
            if (MOST_WORDS_MODE.equals(mode))
                return new MostWordsTokenCollector();
            if (MAX_WORD_LENGTH_MODE.equals(mode))
                return new MaxWordLengthTokenCollector();
            throw new Error("never happened");
        }
    }
```

6 .Solr 搜索

6.1 相关 url 介绍

http://localhost:8080/solr/ select/ 查询索引路径

http://localhost:8080/solr/update/ 查询更新路径

6.2 查询参数介绍

```
fl=*,score&q.op=AND&start=0&rows=16&hl=true&hl.fl=merHeading&h
l.snippets=3&hl.simple.pre=<font
color=red>&hl.simple.post=</font>&facet=true&facet.field=merCategory&
q=+(merHeading%3A%E4%BD%A0%E5%A5%BD+AND+merHeadingWithWord%3A%E6%BD%9
8 )  +merActualendTime:[1239264030468 TO
1240473630468]&sort=merActualendTime asc
```

`fl` 表示索引显示那些 field(*表示所有 field, `score` 是 `solr` 的一个匹配热度)

`q.op` 表示 `q` 中 查询语句的 各条件的逻辑操作 `AND(`与`)` `OR(`或`)`

`start` 开始返回条数

`rows` 返回多少条

`hl` 是否高亮

`hl.fl` 高亮 field

`hl.snippets` 不太清楚`(`反正是设置高亮 `3` 就可以了`)`

`hl.simple.pre` 高亮前面的格式

hl.simple.post 高亮后面的格式

facet 是否启动统计

facet.field 统计 field

q 查询语句(类似 SQL) 相关详细的操作还需 lucene 的 query 语法

sort 排序

6.3 中文搜索

中文搜索时必须要将中文 URLEncoder.encode 用 UTF-8, tomcat 还需设置 URIEncoding="UTF-8" ,用 GET 方式发送请求.

如:merHeading%3A%E4%BD%A0%E5%A5%BD+AND+merHeadingWithWord%3A%E6%BD%98

我是要搜索 “你好 潘” 因为庖丁分词没有单词分词,当搜索中有单词是还需要使用另外的 field ,所以在程序中要分解“你好 潘”成” 你好””潘”,然后”你好”用 merHeading 搜索, ”潘”用 merHeadingWithWord 搜索再取他们的并集(AND 操作)

7. Solr 相关命令

7.1 新建和更新

```
<add>
  <doc><field name="id">1</field><field name="merHeading">
你好潘修艳</field>
<field name="merHeadingWithWord">你好潘修艳</field>
  </doc>
  <doc>
<field name="id">2</field><field name="merHeading">你好潘修艳 1</field>
<field name="merHeadingWithWord">你好潘修艳 1</field>
  </doc>
</add>
```

新建和更新的文档格式一样,注 field 有”&”字符用”&”替换.

发送到 http://localhost:8080/solr/update/ 用 POST 方式

7.2 删除

```
<delete><id>1</id><id>2</id>
```

发送到 http://localhost:8080/solr/update/ 用 POST 方式

7.3 提交

以上操作如果需要生效的需要提交命令<commit/>

发送到 http://localhost:8080/solr/update/ 用 POST 方式

7.4 优化索引库

```
<optimize/>
```

发送到 http://localhost:8080/solr/update/ 用 POST 方式

8. 总结

以上是个人项目开发时的技术总结,如有错误请指出,与大家一齐分享成果是一件很开心的事.

作者:潘修艳

2009/04/09