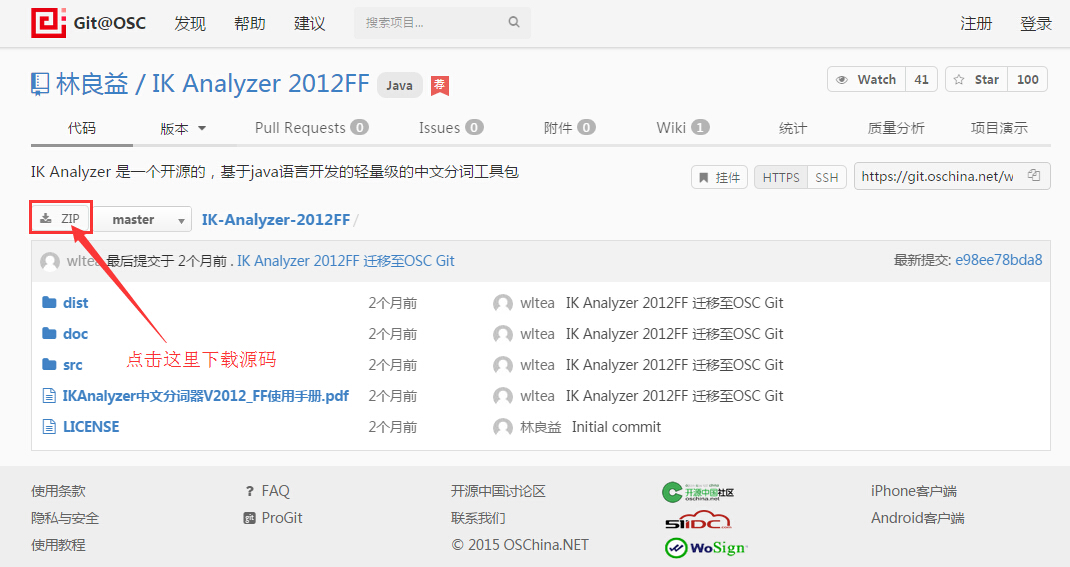
IKAnalyzer分词器编译及使用

之前的示例中，使用的是默认的StandardAnalyzer分词器，不能有效的进行中文分词，下面演示下如何在Lucene5.0中使用IKAnalyzer分词器。

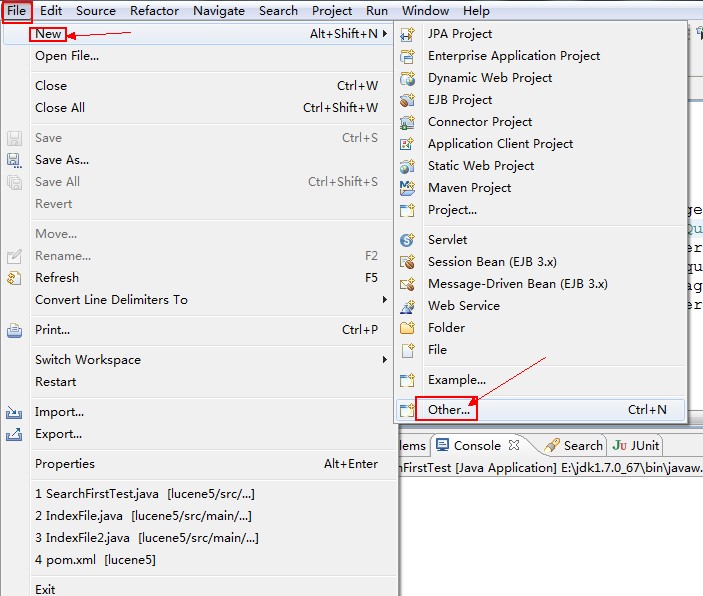
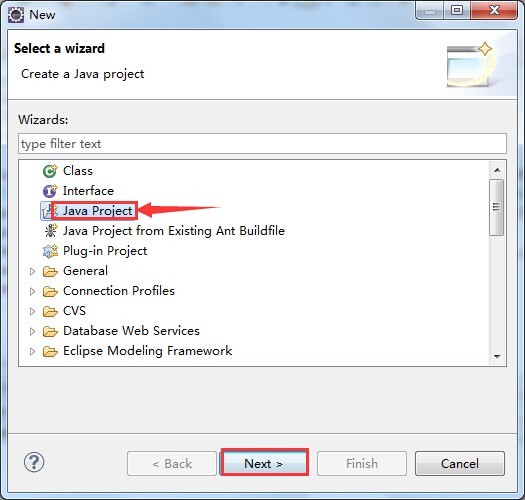
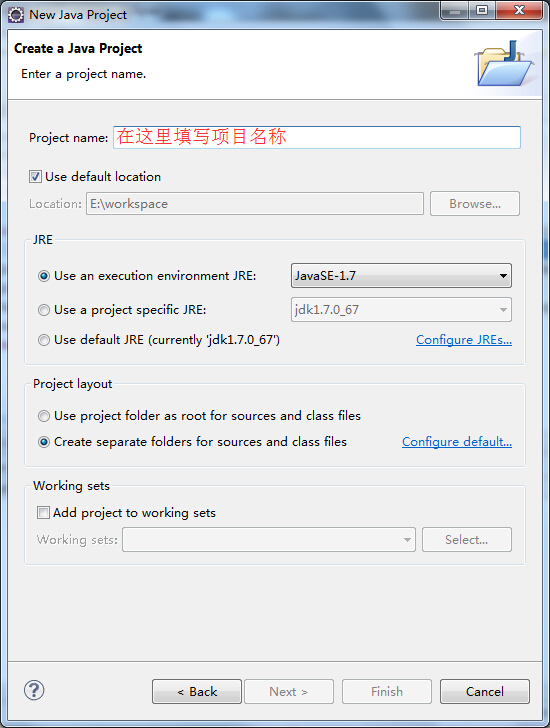
首先下载IKAnalyzer分词器源码，IKAnalyzer分词器源码托管在OSChina的git上。下载地址：

**http://git.oschina.net/wltea/IK-Analyzer-2012FF**

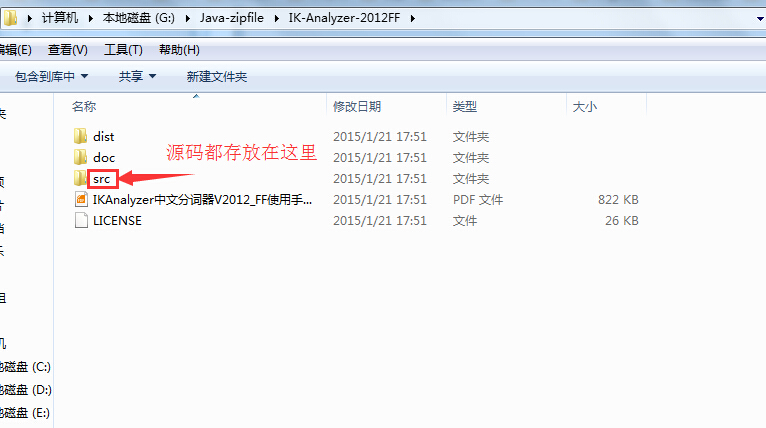
请如图下载IK的源代码：



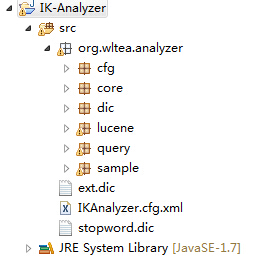
然后打开Eclipse新建一个Java Project:

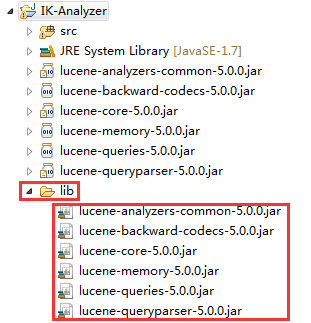
 然后解压下载下来的IKAnalyzer源码压缩包，你将得到如图这样的目录结构：



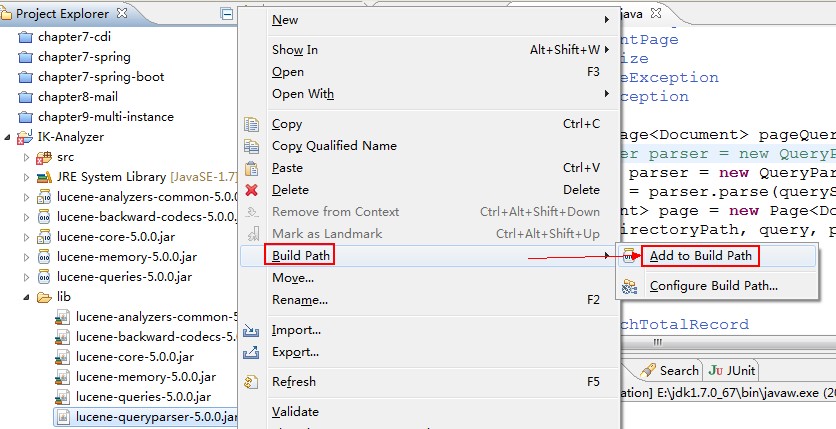
打开src目录，ctrl+A，然后ctrl + C,粘帖到新建的project的src源码包下，如图：



然后新建一个lib目录存放IK依赖的Lucene Jar包，如图：



 然后全选lib下的jar包鼠标右键Build Path -->Add to Build Path,如图：



导好Jar包后，由于Lucene5.0 API上有些变化，我们需要对IK源码做些修改，具体修改如下：

第一处需要修改的就是IKTokenizer类，在其构造函数里把//super(in);这句注释掉即可，下面这是我修改过后的源码：

Java代码  [收藏代码](javascript:void())

1. /\*\*
2. \* IK 中文分词  版本 5.0.1
3. \* IK Analyzer release 5.0.1
4. \*
5. \* Licensed to the Apache Software Foundation (ASF) under one or more
6. \* contributor license agreements.  See the NOTICE file distributed with
7. \* this work for additional information regarding copyright ownership.
8. \* The ASF licenses this file to You under the Apache License, Version 2.0
9. \* (the "License"); you may not use this file except in compliance with
10. \* the License.  You may obtain a copy of the License at
11. \*
12. \*     http://www.apache.org/licenses/LICENSE-2.0
13. \*
14. \* Unless required by applicable law or agreed to in writing, software
15. \* distributed under the License is distributed on an "AS IS" BASIS,
16. \* WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
17. \* See the License for the specific language governing permissions and
18. \* limitations under the License.
19. \*
20. \* 源代码由林良益(linliangyi2005@gmail.com)提供
21. \* 版权声明 2012，乌龙茶工作室
22. \* provided by Linliangyi and copyright 2012 by Oolong studio
23. \*
24. \*
25. \*/
26. package org.wltea.analyzer.lucene;
28. import java.io.IOException;
29. import java.io.Reader;
31. import org.apache.lucene.analysis.Tokenizer;
32. import org.apache.lucene.analysis.tokenattributes.CharTermAttribute;
33. import org.apache.lucene.analysis.tokenattributes.OffsetAttribute;
34. import org.apache.lucene.analysis.tokenattributes.TypeAttribute;
35. import org.apache.lucene.util.Version;
36. import org.wltea.analyzer.core.IKSegmenter;
37. import org.wltea.analyzer.core.Lexeme;
39. /\*\*
40. \* IK分词器 Lucene Tokenizer适配器类
41. \* 兼容Lucene 4.0版本
42. \*/
43. public final class IKTokenizer extends Tokenizer {
45. //IK分词器实现
46. private IKSegmenter \_IKImplement;
48. //词元文本属性
49. private final CharTermAttribute termAtt;
50. //词元位移属性
51. private final OffsetAttribute offsetAtt;
52. //词元分类属性（该属性分类参考org.wltea.analyzer.core.Lexeme中的分类常量）
53. private final TypeAttribute typeAtt;
54. //记录最后一个词元的结束位置
55. private int endPosition;
57. private Version version = Version.LATEST;
58. /\*\*
59. \* Lucene 4.0 Tokenizer适配器类构造函数
60. \* @param in
61. \* @param useSmart
62. \*/
63. public IKTokenizer(Reader in , boolean useSmart){
64. //super(in);
65. offsetAtt = addAttribute(OffsetAttribute.class);
66. termAtt = addAttribute(CharTermAttribute.class);
67. typeAtt = addAttribute(TypeAttribute.class);
68. \_IKImplement = new IKSegmenter(input , useSmart);
69. }
71. /\* (non-Javadoc)
72. \* @see org.apache.lucene.analysis.TokenStream#incrementToken()
73. \*/
74. @Override
75. public boolean incrementToken() throws IOException {
76. //清除所有的词元属性
77. clearAttributes();
78. Lexeme nextLexeme = \_IKImplement.next();
79. if(nextLexeme != null){
80. //将Lexeme转成Attributes
81. //设置词元文本
82. termAtt.append(nextLexeme.getLexemeText());
83. //设置词元长度
84. termAtt.setLength(nextLexeme.getLength());
85. //设置词元位移
86. offsetAtt.setOffset(nextLexeme.getBeginPosition(),
87. nextLexeme.getEndPosition());
88. //记录分词的最后位置
89. endPosition = nextLexeme.getEndPosition();
90. //记录词元分类
91. typeAtt.setType(nextLexeme.getLexemeTypeString());
92. //返会true告知还有下个词元
93. return true;
94. }
95. //返会false告知词元输出完毕
96. return false;
97. }
99. /\*
100. \* (non-Javadoc)
101. \* @see org.apache.lucene.analysis.Tokenizer#reset(java.io.Reader)
102. \*/
103. @Override
104. public void reset() throws IOException {
105. super.reset();
106. \_IKImplement.reset(input);
107. }
109. @Override
110. public final void end() {
111. // set final offset
112. int finalOffset = correctOffset(this.endPosition);
113. offsetAtt.setOffset(finalOffset, finalOffset);
114. }
115. }

还有一个比较要的类就是IKAnalyzer，其中的createComponents方法是继承Luecene的Analyzer接口的，由于Lucene5.0里把createComponents方法的第二个参数去掉了，所以需要对该方法做如下修改：

Java代码  [收藏代码](javascript:void())

1. /\*\*
2. \* 重载Analyzer接口，构造分词组件
3. \*/
4. @Override
5. protected TokenStreamComponents createComponents(String text) {
6. Reader reader = new BufferedReader(new StringReader(text));
7. Tokenizer \_IKTokenizer = new IKTokenizer(reader , this.useSmart());
8. return new TokenStreamComponents(\_IKTokenizer);
9. }

当然还有其他地方需要修改，但其他的地方我觉得很容易就知道怎么改，我就不过多说明了。这里我把我修改好了的IKAnalyzer分词器源码以及打包好的jar包下载地址分享给你们，如果你们不想自己再弄一遍，就访问这个地址去下载吧：

IKAnalyzer5.0下载地址：

**http://pan.baidu.com/s/1o6wr5zk**

IKAnalyzer Jar包弄好了，我们还需要把Jar包安装到我们本地仓库中，因为我们的demo是Maven Project,至于如何把Jar包安装到本地仓库中，请参看我这篇博文[《Maven如何安装Jar包到本地仓库》](http://iamyida.iteye.com/blog/2193493)。jar安装到本地仓库后，我们就可以在pom.xml中配置IK依赖了，从而导入IKAnalyzer jar包。

Xml代码  [收藏代码](javascript:void())

1. <dependency>
2. <groupId>org.wltea.analyzer</groupId>
3. <artifactId>IKAnalyzer</artifactId>
4. <version>5.0</version>
5. </dependency>

接下来我们创建分词器的时候，直接new IKAnalyzer()即可。注意创建索引时候使用的分词器对象类型必须要和查询时使用的分词器对象类型保持一致。

至于IKAnalyzer如何添加自定义词典和自定义停用词词典，这个很简单，你只需要从下载下来的源码包里找到默认的两个dic词典文件，复制一份重命名一下，然后复制到项目的src目录下添加词典内容，然后把IKAnalyzer.cfg.xml配置文件也copy到项目的src目录下，打开IKAnalyzer.cfg.xml配置文件，在里面配置上你新添加的词典文件，多个词典文件是用分号分割的，如**ext.dic;aaaaa.dic;bbbbb.dic;**