

原 LIRe 源代码分析 3：基本接口（ImageSearcher）

2013年10月31日 20:48:59 阅读数：5673

=====

LIRe源代码分析系列文章列表：

[LIRe 源代码分析 1：整体结构](#)

[LIRe 源代码分析 2：基本接口（DocumentBuilder）](#)

[LIRe 源代码分析 3：基本接口（ImageSearcher）](#)

[LIRe 源代码分析 4：建立索引（DocumentBuilder）\[以颜色布局为例\]](#)

[LIRe 源代码分析 5：提取特征向量\[以颜色布局为例\]](#)

[LIRe 源代码分析 6：检索（ImageSearcher）\[以颜色布局为例\]](#)

[LIRe 源代码分析 7：算法类\[以颜色布局为例\]](#)

=====

上篇文章介绍了LIRe源代码里的DocumentBuilder的几个基本接口。本文继续研究一下源代码里的ImageSearcher的几个基本接口。

下面来看看与ImageSearcher相关的类的定义：

ImageSearcher：接口，定义了基本的方法。

AbstractImageSearcher：纯虚类，实现了ImageSearcher接口。

ImageSearcherFactory：用于创建ImageSearcher。

ImageSearcher相关的类的继承关系如下图所示。可见，各种算法类都继承了AbstractImageSearcher，而AbstractImageSearcher实现了ImageSearcher接口。

□

此外还有一个结构体：

ImageSearchHits：用于存储搜索的结果。

详细的源代码如下所示：

ImageSearcher

```
[java]
1.  /*
2.   * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
3.   * LIRe is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License as published by
5.   * the Free Software Foundation; either version 2 of the License, or
6.   * (at your option) any later version.
7.   *
8.   * LIRe is distributed in the hope that it will be useful,
9.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
10.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11.  * GNU General Public License for more details.
12.  *
13.  * You should have received a copy of the GNU General Public License
14.  * along with LIRe; if not, write to the Free Software
15.  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
16.  *
17.  * We kindly ask you to refer the following paper in any publication mentioning Lire:
18.  *
19.  * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval  欽0
20.  * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
21.  * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
22.  *
23.  * http://doi.acm.org/10.1145/1459359.1459577
24.  *
25.  * Copyright statement:
26.  * -----
27.  * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
```

```

28. *   http://www.semanticmetadata.net/lire
29. */
30.
31. package net.semanticmetadata.lire;
32.
33. import org.apache.lucene.document.Document;
34. import org.apache.lucene.index.IndexReader;
35.
36. import java.awt.image.BufferedImage;
37. import java.io.IOException;
38. import java.io.InputStream;
39. import java.util.Set;
40.
41. /**
42.  * <h2>Searching in an Index</h2>
43.  * Use the ImageSearcherFactory for creating an ImageSearcher, which will retrieve the images
44.  * for you from the index.
45.  * <p/>
46.  * <pre>
47.  * IndexReader reader = IndexReader.open(indexPath);
48.  * ImageSearcher searcher = ImageSearcherFactory.createDefaultSearcher();
49.  * FileInputStream imageStream = new FileInputStream("image.jpg");
50.  * BufferedImage bimg = ImageIO.read(imageStream);
51.  * // searching for an image:
52.  * ImageSearchHits hits = null;
53.  * hits = searcher.search(bimg, reader);
54.  * for (int i = 0; i < 5; i++) {
55.  * System.out.println(hits.score(i) + ": " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
56.  * }
57.  *
58.  * // searching for a document:
59.  * Document document = hits.doc(0);
60.  * hits = searcher.search(document, reader);
61.  * for (int i = 0; i < 5; i++) {
62.  * System.out.println(hits.score(i) + ": " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
63.  * }
64.  * </pre>
65.  * <p/>
66.  * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
67.  * <br>Date: 01.02.2006
68.  * <br>Time: 00:09:42
69.  *
70.  * @author Mathias Lux, mathias@juggle.at
71.  */
72. public interface ImageSearcher {
73.     /**
74.      * Searches for images similar to the given image.
75.      *
76.      * @param image the example image to search for.
77.      * @param reader the IndexReader which is used to dsearch through the images.
78.      * @return a sorted list of hits.
79.      * @throws java.io.IOException in case exceptions in the reader occurs
80.      */
81.     public ImageSearchHits search(BufferedImage image, IndexReader reader) throws IOException;
82.
83.     /**
84.      * Searches for images similar to the given image, defined by the Document from the index.
85.      *
86.      * @param doc the example image to search for.
87.      * @param reader the IndexReader which is used to dsearch through the images.
88.      * @return a sorted list of hits.
89.      * @throws java.io.IOException in case exceptions in the reader occurs
90.      */
91.     public ImageSearchHits search(Document doc, IndexReader reader) throws IOException;
92.
93.     /**
94.      * Searches for images similar to the given image.
95.      *
96.      * @param image the example image to search for.
97.      * @param reader the IndexReader which is used to dsearch through the images.
98.      * @return a sorted list of hits.
99.      * @throws IOException in case the image could not be read from stream.
100.     */
101.     public ImageSearchHits search(InputStream image, IndexReader reader) throws IOException;
102.
103.     /**
104.      * Identifies duplicates in the database.
105.      *
106.      * @param reader the IndexReader which is used to dsearch through the images.
107.      * @return a sorted list of hits.
108.      * @throws IOException in case the image could not be read from stream.
109.      */
110.     public ImageDuplicates findDuplicates(IndexReader reader) throws IOException;
111.
112.     /**
113.      * Modifies the given search by the provided positive and negative examples. This process follows the idea
114.      * of relevance feedback.
115.      *
116.      * @param originalSearch
117.      * @param positives
118.      * @param negatives
119.      * @return

```

```

119.         return
120.     */
121.     public ImageSearchHits relevanceFeedback(ImageSearchHits originalSearch,
122.                                             Set<Document> positives, Set<Document> negatives);
123. }

```

从接口的源代码可以看出，提供了5个方法，其中有3个名字都叫search()，只是参数不一样。一个是BufferedImage，一个是Document，而另一个是InputStream。

AbstractImageSearcher

```

[java]
1.  /*
2.   * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
3.   * LIRe is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License as published by
5.   * the Free Software Foundation; either version 2 of the License, or
6.   * (at your option) any later version.
7.   *
8.   * LIRe is distributed in the hope that it will be useful,
9.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
10.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11.  * GNU General Public License for more details.
12.  *
13.  * You should have received a copy of the GNU General Public License
14.  * along with LIRe; if not, write to the Free Software
15.  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
16.  *
17.  * We kindly ask you to refer the following paper in any publication mentioning Lire:
18.  *
19.  * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥?
20.  * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
21.  * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
22.  *
23.  * http://doi.acm.org/10.1145/1459359.1459577
24.  *
25.  * Copyright statement:
26.  * -----
27.  * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
28.  * http://www.semanticmetadata.net/lire
29.  */
30. package net.semanticmetadata.lire;
31.
32. import org.apache.lucene.document.Document;
33. import org.apache.lucene.index.IndexReader;
34.
35. import javax.imageio.ImageIO;
36. import java.awt.image.BufferedImage;
37. import java.io.IOException;
38. import java.io.InputStream;
39. import java.util.Set;
40.
41.
42. /**
43.  * Abstract ImageSearcher, which uses javax.imageio.ImageIO to create a BufferedImage
44.  * from an InputStream.
45.  * <p/>
46.  * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
47.  * <br>Date: 01.02.2006
48.  * <br>Time: 00:13:16
49.  *
50.  * @author Mathias Lux, mathias@juggle.at
51.  */
52. public abstract class AbstractImageSearcher implements ImageSearcher {
53.     /**
54.      * Searches for images similar to the given image. This simple implementation uses
55.      * {@link ImageSearcher#search(java.awt.image.BufferedImage, org.apache.lucene.index.IndexReader)},
56.      * the image is read using javax.imageio.ImageIO.
57.      *
58.      * @param image the example image to search for.
59.      * @param reader the IndexReader which is used to dsearch through the images.
60.      * @return a sorted list of hits.
61.      * @throws IOException in case the image could not be read from stream.
62.      */
63.     public ImageSearchHits search(InputStream image, IndexReader reader) throws IOException {
64.         BufferedImage bufferedImage = ImageIO.read(image);
65.         return search(bufferedImage, reader);
66.     }
67.
68.     public ImageSearchHits relevanceFeedback(ImageSearchHits originalSearch, Set<Document> positives, Set<Document> negatives) {
69.         throw new UnsupportedOperationException("Not implemented yet for this kind of searcher!");
70.     }
71. }

```

从代码中可以看出AbstractImageSearcher实现了ImageSearcher接口。其中的search(InputStream image, IndexReader reader)方法调用了search(BufferedImage image, IndexReader reader)方法。说白了，就是把2个函数的功能合并为一个函数。

ImageSearcherFactory

```
1.  /*
2.   * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
3.   * LIRe is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License as published by
5.   * the Free Software Foundation; either version 2 of the License, or
6.   * (at your option) any later version.
7.   *
8.   * LIRe is distributed in the hope that it will be useful,
9.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
10.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11.  * GNU General Public License for more details.
12.  *
13.  * You should have received a copy of the GNU General Public License
14.  * along with LIRe; if not, write to the Free Software
15.  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
16.  *
17.  * We kindly ask you to refer the following paper in any publication mentioning Lire:
18.  *
19.  * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥?
20.  * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
21.  * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
22.  *
23.  * http://doi.acm.org/10.1145/1459359.1459577
24.  *
25.  * Copyright statement:
26.  * ~~~~~
27.  * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
28.  * http://www.semanticmetadata.net/lire
29.  */
30.
31. package net.semanticmetadata.lire;
32.
33. import net.semanticmetadata.lire.imageanalysis.*;
34. import net.semanticmetadata.lire.impl.CorrelogramImageSearcher;
35. import net.semanticmetadata.lire.impl.GenericFastImageSearcher;
36. import net.semanticmetadata.lire.impl.SimpleImageSearcher;
37.
38. /**
39.  * <h2>Searching in an Index</h2>
40.  * Use the ImageSearcherFactory for creating an ImageSearcher, which will retrieve the images
41.  * for you from the index.
42.  * <p>
43.  * <pre>
44.  * IndexReader reader = IndexReader.open(indexPath);
45.  * ImageSearcher searcher = ImageSearcherFactory.createDefaultSearcher();
46.  * FileInputStream imageStream = new FileInputStream("image.jpg");
47.  * BufferedImage bimg = ImageIO.read(imageStream);
48.  * // searching for an image:
49.  * ImageSearchHits hits = null;
50.  * hits = searcher.search(bimg, reader);
51.  * for (int i = 0; i < 5; i++) {
52.  * System.out.println(hits.score(i) + " : " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
53.  * }
54.  *
55.  * // searching for a document:
56.  * Document document = hits.doc(0);
57.  * hits = searcher.search(document, reader);
58.  * for (int i = 0; i < 5; i++) {
59.  * System.out.println(hits.score(i) + " : " + hits.doc(i).getField(DocumentBuilder.FIELD_NAME_IDENTIFIER).stringValue());
60.  * }
61.  * </pre>
62.  * <p>
63.  * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
64.  * <br>Date: 03.02.2006
65.  * <br>Time: 00:30:07
66.  *
67.  * @author Mathias Lux, mathias@juggle.at
68.  */
69. public class ImageSearcherFactory {
70.     /**
71.      * Default number of maximum hits.
72.      */
73.     public static int NUM_MAX_HITS = 100;
74.
75.     /**
76.      * Creates a new simple image searcher with the desired number of maximum hits.
77.      *
78.      * @param maximumHits
79.      * @return the searcher instance
80.      * @deprecated Use ColorLayout, EdgeHistogram and ScalableColor features instead.
81.      */
82.     public static ImageSearcher createSimpleSearcher(int maximumHits) {
83.         return ImageSearcherFactory.createColorLayoutImageSearcher(maximumHits);
84.     }
85.
86.     /**
87.      * Returns a new default ImageSearcher with a predefined number of maximum
88.      * hits defined in the {@link ImageSearcherFactory#NUM_MAX_HITS} based on the {@link net.semanticmetadata.lire.imageanalysis.CEDD
```

```

89.     *
90.     * @return the searcher instance
91.     */
92.     public static ImageSearcher createDefaultSearcher() {
93.         return new GenericFastImageSearcher(NUM_MAX_HITS, CEDD.class, DocumentBuilder.FIELD_NAME_CEDD);
94.     }
95.
96.     /**
97.     * Returns a new ImageSearcher with the given number of maximum hits
98.     * which only takes the overall color into account. texture and color
99.     * distribution are ignored.
100.    *
101.    * @param maximumHits defining how many hits are returned in max (e.g. 100 would be ok)
102.    * @return the ImageSearcher
103.    * @see ImageSearcher
104.    * @deprecated Use ColorHistogram or ScalableColor instead
105.    */
106.    public static ImageSearcher createColorOnlySearcher(int maximumHits) {
107.        return ImageSearcherFactory.createScalableColorImageSearcher(maximumHits);
108.    }
109.
110.    /**
111.    * Returns a new ImageSearcher with the given number of maximum hits and
112.    * the specified weights on the different matching aspects. All weights
113.    * should be in [0,1] whereas a weight of 0 implies that the feature is
114.    * not taken into account for searching. Note that the effect is relative and
115.    * can only be fully applied if the {@link DocumentBuilderFactory#getExtensiveDocumentBuilder() extensive DocumentBuilder}
116.    * is used.
117.    *
118.    * @param maximumHits          defining how many hits are returned in max
119.    * @param colorHistogramWeight  a weight in [0,1] defining the importance of overall color in the images
120.    * @param colorDistributionWeight a weight in [0,1] defining the importance of color distribution (which color where) in the image
121.    *
122.    * @param textureWeight        defining the importance of texture (which edges where) in the images
123.    * @return the searcher instance or NULL if the weights are not appropriate, eg. all 0 or not in [0,1]
124.    * @see DocumentBuilderFactory
125.    * @deprecated Use ColorLayout, EdgeHistogram and ScalableColor features instead.
126.    */
127.    public static ImageSearcher createWeightedSearcher(int maximumHits,
128.                                                       float colorHistogramWeight,
129.                                                       float colorDistributionWeight,
130.                                                       float textureWeight) {
131.        if (isAppropriateWeight(colorHistogramWeight)
132.            && isAppropriateWeight(colorDistributionWeight)
133.            && isAppropriateWeight(textureWeight)
134.            && (colorHistogramWeight + colorDistributionWeight + textureWeight > 0f))
135.            return new SimpleImageSearcher(maximumHits, colorHistogramWeight, colorDistributionWeight, textureWeight);
136.        else
137.            return null;
138.    }
139.
140.    /**
141.    * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.AutoColorCorrelogram}
142.    * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
143.    *
144.    * @param maximumHits number of hits returned.
145.    * @return
146.    */
147.    public static ImageSearcher createAutoColorCorrelogramImageSearcher(int maximumHits) {
148.        return new GenericFastImageSearcher(maximumHits, AutoColorCorrelogram.class, DocumentBuilder.FIELD_NAME_AUTOCOLORCORRELOGRAM);
149.    }
150.
151.    /**
152.    * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.AutoColorCorrelogram}
153.    * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
154.    *
155.    * @param maximumHits number of hits returned.
156.    * @return
157.    * @deprecated Use #createAutoColorCorrelogramImageSearcher instead
158.    */
159.    public static ImageSearcher createFastCorrelogramImageSearcher(int maximumHits) {
160.        return new CorrelogramImageSearcher(maximumHits, AutoColorCorrelogram.Mode.SuperFast);
161.    }
162.
163.    /**
164.    * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.CEDD}
165.    * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
166.    *
167.    * @param maximumHits
168.    * @return
169.    */
170.    public static ImageSearcher createCEDDImageSearcher(int maximumHits) {
171.        return new CEDDImageSearcher(maximumHits);
172.    }
173.
174.    /**
175.    * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.FCTH}
176.    * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
177.    *

```

```

177.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.FCTH}
178.     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
179.     *
180.     * @param maximumHits
181.     * @return
182.     */
183.     public static ImageSearcher createFCTHImageSearcher(int maximumHits) {
184.         // return new GenericImageSearcher(maximumHits, FCTH.class, DocumentBuilder.FIELD_NAME_FCTH);
185.         return new GenericFastImageSearcher(maximumHits, FCTH.class, DocumentBuilder.FIELD_NAME_FCTH);
186.     }
187.
188.
189.     /**
190.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.JCD}
191.     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
192.     *
193.     * @param maximumHits
194.     * @return
195.     */
196.     public static ImageSearcher createJCDImageSearcher(int maximumHits) {
197.         return new GenericFastImageSearcher(maximumHits, JCD.class, DocumentBuilder.FIELD_NAME_JCD);
198.     }
199.
200.
201.     /**
202.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.JpegCoefficientHistogram}
203.     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
204.     *
205.     * @param maximumHits
206.     * @return
207.     */
208.     public static ImageSearcher createJpegCoefficientHistogramImageSearcher(int maximumHits) {
209.         return new GenericFastImageSearcher(maximumHits, JpegCoefficientHistogram.class, DocumentBuilder.FIELD_NAME_JPEGCOEFFS);
210.     }
211.
212.
213.     /**
214.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.SimpleColorHistogram}
215.     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
216.     *
217.     * @param maximumHits
218.     * @return
219.     */
220.     public static ImageSearcher createColorHistogramImageSearcher(int maximumHits) {
221.         // return new GenericImageSearcher(maximumHits, SimpleColorHistogram.class, DocumentBuilder.FIELD_NAME_COLORHISTOGRAM);
222.         return new GenericFastImageSearcher(maximumHits, SimpleColorHistogram.class, DocumentBuilder.FIELD_NAME_COLORHISTOGRAM);
223.     }
224.
225.     /**
226.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.Tamura}
227.     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
228.     *
229.     * @param maximumHits
230.     * @return
231.     */
232.     public static ImageSearcher createTamuraImageSearcher(int maximumHits) {
233.         return new GenericFastImageSearcher(maximumHits, Tamura.class, DocumentBuilder.FIELD_NAME_TAMURA);
234.     }
235.
236.     /**
237.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.Gabor}
238.     * image feature. Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
239.     *
240.     * @param maximumHits
241.     * @return
242.     */
243.     public static ImageSearcher createGaborImageSearcher(int maximumHits) {
244.         return new GenericFastImageSearcher(maximumHits, Gabor.class, DocumentBuilder.FIELD_NAME_GABOR);
245.     }
246.
247.     /**
248.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.ColorLayout}
249.     * image feature using the byte[] serialization. Be sure to use the same options for the ImageSearcher as
250.     * you used for the DocumentBuilder.
251.     *
252.     * @param maximumHits
253.     * @return
254.     */
255.     public static ImageSearcher createColorLayoutImageSearcher(int maximumHits) {
256.         return new GenericFastImageSearcher(maximumHits, ColorLayout.class, DocumentBuilder.FIELD_NAME_COLORLAYOUT);
257.     }
258.
259.     /**
260.     * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.ScalableColor}
261.     * image feature using the byte[] serialization. Be sure to use the same options for the ImageSearcher as
262.     * you used for the DocumentBuilder.
263.     *
264.     * @param maximumHits
265.     * @return
266.     */
267.     public static ImageSearcher createScalableColorImageSearcher(int maximumHits) {
268.         return new GenericFastImageSearcher(maximumHits, ScalableColor.class, DocumentBuilder.FIELD_NAME_SCALABLECOLOR);

```

```

269.     }
270.
271.     /**
272.      * Create and return an ImageSearcher for the {@link net.semanticmetadata.lire.imageanalysis.EdgeHistogram}
273.      * image feature using the byte[] serialization. Be sure to use the same options for the ImageSearcher as
274.      * you used for the DocumentBuilder.
275.      *
276.      * @param maximumHits
277.      * @return
278.      */
279.     public static ImageSearcher createEdgeHistogramImageSearcher(int maximumHits) {
280.         return new GenericFastImageSearcher(maximumHits, EdgeHistogram.class, DocumentBuilder.FIELD_NAME_EDGEHISTOGRAM);
281.     }
282.
283.
284.     /**
285.      * Checks if the weight is in [0,1]
286.      *
287.      * @param f the weight to check
288.      * @return true if the weight is in [0,1], false otherwise
289.      */
290.     private static boolean isAppropriateWeight(float f) {
291.         boolean result = false;
292.         if (f <= 1f && f >= 0) result = true;
293.         return result;
294.     }
295. }
296.

```

ImageSearcherFactory是用于创建ImageSearcher的。里面有各种create****ImageSearcher()。每个函数的作用在注释中都有详细的说明。

ImageSearchHits

```

1.  /*
2.   * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
3.   * LIRe is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License as published by
5.   * the Free Software Foundation; either version 2 of the License, or
6.   * (at your option) any later version.
7.   *
8.   * LIRe is distributed in the hope that it will be useful,
9.   * but WITHOUT ANY WARRANTY; without even the implied warranty of
10.  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11.  * GNU General Public License for more details.
12.  *
13.  * You should have received a copy of the GNU General Public License
14.  * along with LIRe; if not, write to the Free Software
15.  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
16.  *
17.  * We kindly ask you to refer the following paper in any publication mentioning Lire:
18.  *
19.  * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥?
20.  * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
21.  * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
22.  *
23.  * http://doi.acm.org/10.1145/1459359.1459577
24.  *
25.  * Copyright statement:
26.  * -----
27.  * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
28.  * http://www.semanticmetadata.net/lire
29.  */
30.
31. package net.semanticmetadata.lire;
32.
33. import org.apache.lucene.document.Document;
34.
35. /**
36.  * This class simulates the original Lucene Hits object.
37.  * Please note the only a certain number of results are returned.<br>
38.  * <p/>
39.  * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
40.  * <br>Date: 02.02.2006
41.  * <br>Time: 23:45:20
42.  *
43.  * @author Mathias Lux, mathias@juggle.at
44.  */
45. public interface ImageSearchHits {
46.     /**
47.      * Returns the size of the result list.
48.      *
49.      * @return the size of the result list.
50.      */
51.     public int length();
52.
53.     /**
54.      * Returns the score of the document at given position.
55.      * Please note that the score in this case is a distance,
56.      * which means a score of 0 denotes the best possible hit.
57.      * The result list starts with position 0 as everything
58.      * in computer science does.
59.      *
60.      * @param position defines the position
61.      * @return the score of the document at given position. The lower the better (its a distance measure).
62.      */
63.     public float score(int position);
64.
65.     /**
66.      * Returns the document at given position
67.      *
68.      * @param position defines the position.
69.      * @return the document at given position.
70.      */
71.     public Document doc(int position);
72. }

```

该类主要用于存储ImageSearcher类中search()方法返回的结果。

SimpleImageSearchHits是ImageSearcher的实现。该类的源代码如下所示：

```

1.  /*
2.   * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
3.   * LIRe is free software; you can redistribute it and/or modify
4.   * it under the terms of the GNU General Public License as published by
5.   * the Free Software Foundation; either version 2 of the License, or
6.   * (at your option) any later version.
7.   *

```



```

8.  * LIRE is distributed in the hope that it will be useful,
9.  * but WITHOUT ANY WARRANTY; without even the implied warranty of
10. * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
11. * GNU General Public License for more details.
12. *
13. * You should have received a copy of the GNU General Public License
14. * along with LIRE; if not, write to the Free Software
15. * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
16. *
17. * We kindly ask you to refer the following paper in any publication mentioning Lire:
18. *
19. * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 欵
20. * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
21. * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
22. *
23. * http://doi.acm.org/10.1145/1459359.1459577
24. *
25. * Copyright statement:
26. * -----
27. * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
28. * http://www.semanticmetadata.net/lire
29. */
30.
31. package net.semanticmetadata.lire.impl;
32.
33. import net.semanticmetadata.lire.ImageSearchHits;
34. import org.apache.lucene.document.Document;
35.
36. import java.util.ArrayList;
37. import java.util.Collection;
38. import java.util.Iterator;
39.
40. /**
41.  * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
42.  * <br>Date: 02.02.2006
43.  * <br>Time: 23:56:15
44.  *
45.  * @author Mathias Lux, mathias@juggle.at
46.  */
47. public class SimpleImageSearchHits implements ImageSearchHits {
48.     ArrayList<SimpleResult> results;
49.
50.     public SimpleImageSearchHits(Collection<SimpleResult> results, float maxDistance) {
51.         this.results = new ArrayList<SimpleResult>(results.size());
52.         this.results.addAll(results);
53.         // this step normalizes and inverts the distance ...
54.         // although its now a score or similarity like measure its further called distance
55.         for (Iterator<SimpleResult> iterator = this.results.iterator(); iterator.hasNext(); ) {
56.             SimpleResult result = iterator.next();
57.             result.setDistance(1f - result.getDistance() / maxDistance);
58.         }
59.     }
60.
61.     /**
62.      * Returns the size of the result list.
63.      *
64.      * @return the size of the result list.
65.      */
66.     public int length() {
67.         return results.size();
68.     }
69.
70.     /**
71.      * Returns the score of the document at given position.
72.      * Please note that the score in this case is a distance,
73.      * which means a score of 0 denotes the best possible hit.
74.      * The result list starts with position 0 as everything
75.      * in computer science does.
76.      *
77.      * @param position defines the position
78.      * @return the score of the document at given position. The lower the better (its a distance measure).
79.      */
80.     public float score(int position) {
81.         return results.get(position).getDistance();
82.     }
83.
84.     /**
85.      * Returns the document at given position
86.      *
87.      * @param position defines the position.
88.      * @return the document at given position.
89.      */
90.     public Document doc(int position) {
91.         return results.get(position).getDocument();
92.     }
93.
94.     private float sigmoid(float f) {
95.         double result = 0f;
96.         result = -1d + 2d / (1d + Math.exp(-2d * f / 0.6));
97.         return (float) (1d - result);
98.     }

```

可以看出检索的结果是存在名为results的ArrayList<SimpleResult> 类型的变量中的。

版权声明：本文为博主原创文章，未经博主允许不得转载。 <https://blog.csdn.net/leixiaohua1020/article/details/13770889>

文章标签：[lire](#) [源代码](#) [索引](#) [检索](#) [lucene](#)

个人分类：[MPEG7/图像检索](#) [LIRe](#)

所属专栏：[开源多媒体项目源代码分析](#)

此PDF由spygg生成,请尊重原作者版权!!!

我的邮箱:liushidc@163.com