原 LIRe 源代码分析 2：基本接口（DocumentBuilder）

2013年10月31日 19:24:27　阅读数：6219

==========

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]

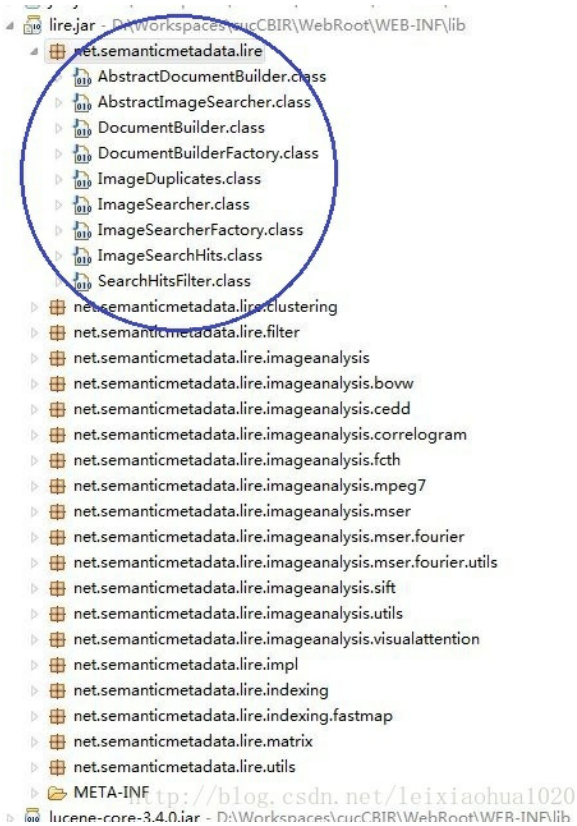
LIRe 源代码分析 6：检索（ImageSearcher）[以颜色布局为例]

LIRe 源代码分析 7：算法类[以颜色布局为例]

==========

本文分析LIRe的基本接口。LIRe的基本接口完成的工作不外乎两项：生成索引和检索。生成索引就是根据图片提取特征向量，然后存储特征向量到索引的过程。检索就是根据输入图片的特征向量到索引中查找相似图片的过程。

LIRe的基本接口位于net.semanticmetadata.lire的包中，如下图所示：



将这些接口分为2类：

DocumentBuilder：用于生成索引

ImageSearcher：用于检索

下面来看看与DocumentBuilder相关的类的定义：

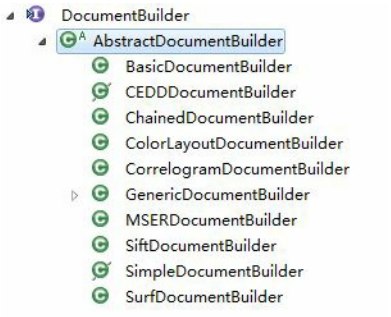
（LIRe在代码注释方面做得很好，每个函数的作用都写得很清楚）

DocumentBuilder：接口，定义了基本的方法。

AbstractDocumentBuilder：纯虚类，实现了DocumentBuilder接口。

DocumentBuilderFactory：用于创建DocumentBuilder。

DocumentBuilder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AbstractDocumentBuilder，而AbstractDocumentBuilder实现了DocumentBuilder。



详细的源代码如下所示：

DocumentBuilder

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe 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.
 *
 * LIRe 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 LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * ~~~~~~~~~~~~~~~~~~~
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire;

import org.apache.lucene.document.Document;

import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;

/**
 * <h2>Creating an Index</h2>
 * <p/>
 * Use DocumentBuilderFactory to create a DocumentBuilder, which
 * will create Lucene Documents from images. Add this documents to
 * an index like this:
 * <p/>
 * <pre>
 * System.out.println(">> Indexing " + images.size() + " files.");
 * DocumentBuilder builder = DocumentBuilderFactory.getExtensiveDocumentBuilder();
 * IndexWriter iw = new IndexWriter(indexPath, new SimpleAnalyzer(LuceneUtils.LUCENE_VERSION), true);
 * int count = 0;
 * long time = System.currentTimeMillis();
 * for (String identifier : images) {
 * Document doc = builder.createDocument(new FileInputStream(identifier), identifier);
 * iw.addDocument(doc);
 * count ++;
 * if (count % 25 == 0) System.out.println(count + " files indexed.");
 * }
 * long timeTaken = (System.currentTimeMillis() - time);
 * float sec = ((float) timeTaken) / 1000f;
 *
 * System.out.println(sec + " seconds taken, " + (timeTaken / count) + " ms per image.");
```

```
 * iw.optimize();
 * iw.close();
 * </pre>
 * <p/>
 * <p/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 31.01.2006
 * <br>Time: 23:02:00
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public interface DocumentBuilder {
    public static final int MAX_IMAGE_SIDE_LENGTH = 800;

    public static final String FIELD_NAME_SCALABLECOLOR = "descriptorScalableColor";
    public static final String FIELD_NAME_COLORLAYOUT = "descriptorColorLayout";
    public static final String FIELD_NAME_EDGEHISTOGRAM = "descriptorEdgeHistogram";
    public static final String FIELD_NAME_AUTOCOLORCORRELOGRAM = "featureAutoColorCorrelogram";
    public static final String FIELD_NAME_COLORHISTOGRAM = "featureColorHistogram";
    public static final String FIELD_NAME_CEDD = "featureCEDD";
    public static final String FIELD_NAME_FCTH = "featureFCTH";
    public static final String FIELD_NAME_JCD = "featureJCD";
    public static final String FIELD_NAME_TAMURA = "featureTAMURA";
    public static final String FIELD_NAME_GABOR = "featureGabor";
    public static final String FIELD_NAME_SIFT = "featureSift";
    public static final String FIELD_NAME_SIFT_LOCAL_FEATURE_HISTOGRAM = "featureSiftHistogram";
    public static final String FIELD_NAME_SIFT_LOCAL_FEATURE_HISTOGRAM_VISUAL_WORDS = "featureSiftHistogramVWords";
    public static final String FIELD_NAME_IDENTIFIER = "descriptorImageIdentifier";
    public static final String FIELD_NAME_CEDD_FAST = "featureCEDDfast";
    public static final String FIELD_NAME_COLORLAYOUT_FAST = "featureColorLayoutfast";
    public static final String FIELD_NAME_SURF = "featureSurf";
    public static final String FIELD_NAME_SURF_LOCAL_FEATURE_HISTOGRAM = "featureSURFHistogram";
    public static final String FIELD_NAME_SURF_LOCAL_FEATURE_HISTOGRAM_VISUAL_WORDS = "featureSurfHistogramVWords";
    public static final String FIELD_NAME_MSER_LOCAL_FEATURE_HISTOGRAM = "featureMSERHistogram";
    public static final String FIELD_NAME_MSER_LOCAL_FEATURE_HISTOGRAM_VISUAL_WORDS = "featureMSERHistogramVWords";
    public static final String FIELD_NAME_MSER = "featureMSER";
    public static final String FIELD_NAME_BASIC_FEATURES = "featureBasic";
    public static final String FIELD_NAME_JPEGCOEFFS = "featureJpegCoeffs";


    /**
     * Creates a new Lucene document from a BufferedImage. The identifier can be used like an id
     * (e.g. the file name or the url of the image)
     *
     * @param image      the image to index. Cannot be NULL.
     * @param identifier an id for the image, for instance the filename or an URL. Can be NULL.
     * @return a Lucene Document containing the indexed image.
     */
    public Document createDocument(BufferedImage image, String identifier);

    /**
     * Creates a new Lucene document from an InputStream. The identifier can be used like an id
     * (e.g. the file name or the url of the image)
     *
     * @param image      the image to index. Cannot be NULL.
     * @param identifier an id for the image, for instance the filename or an URL. Can be NULL.
     * @return a Lucene Document containing the indexed image.
     * @throws IOException in case the image cannot be retrieved from the InputStream
     */
    public Document createDocument(InputStream image, String identifier) throws IOException;

}
```

从接口的源代码可以看出，提供了两个方法，名字都叫createDocument()，只是参数不一样，一个是从BufferedImage，另一个是从InputStream。

此外，定义了很多的字符串。

AbstractDocumentBuilder

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe 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.
 *
 * LIRe 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 LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * --------------------
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire;

import org.apache.lucene.document.Document;

import javax.imageio.ImageIO;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.io.InputStream;

/**
 * Abstract DocumentBuilder, which uses javax.imageio.ImageIO to create a BufferedImage
 * from an InputStream.
 * <p/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 31.01.2006
 * <br>Time: 23:07:39
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public abstract class AbstractDocumentBuilder implements DocumentBuilder {
    /**
     * Creates a new Lucene document from an InputStream. The identifier can be used like an id
     * (e.g. the file name or the url of the image). This is a simple implementation using
     * javax.imageio.ImageIO
     *
     * @param image      the image to index. Please note that
     * @param identifier an id for the image, for instance the filename or an URL.
     * @return a Lucene Document containing the indexed image.
     * @see javax.imageio.ImageIO
     */
    public Document createDocument(InputStream image, String identifier) throws IOException {
        assert (image != null);
        BufferedImage bufferedImage = ImageIO.read(image);
        return createDocument(bufferedImage, identifier);
    }
}
```

从抽象类的定义可以看出，只有一个createDocument(InputStream image, String identifier)，里面调用了createDocument(BufferedImage image, String identifier)。

其实说白了，就是把接口的那两个函数合成了一个函数。

DocumentBuilderFactory

```java
/*
 * This file is part of the LIRe project: http://www.semanticmetadata.net/lire
 * LIRe 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.
 *
 * LIRe 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 LIRe; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 * We kindly ask you to refer the following paper in any publication mentioning Lire:
 *
 * Lux Mathias, Savvas A. Chatzichristofis. Lire: Lucene Image Retrieval 鈥
 * An Extensible Java CBIR Library. In proceedings of the 16th ACM International
 * Conference on Multimedia, pp. 1085-1088, Vancouver, Canada, 2008
 *
 * http://doi.acm.org/10.1145/1459359.1459577
 *
 * Copyright statement:
 * ~~~~~~~~~~~~~~~~~~~
 * (c) 2002-2011 by Mathias Lux (mathias@juggle.at)
 *     http://www.semanticmetadata.net/lire
 */

package net.semanticmetadata.lire;

import net.semanticmetadata.lire.imageanalysis.*;
import net.semanticmetadata.lire.impl.ChainedDocumentBuilder;
import net.semanticmetadata.lire.impl.CorrelogramDocumentBuilder;
import net.semanticmetadata.lire.impl.GenericDocumentBuilder;
import net.semanticmetadata.lire.impl.GenericFastDocumentBuilder;

/**
 * Use DocumentBuilderFactory to create a DocumentBuilder, which
 * will create Lucene Documents from images.  <br/>
 * This file is part of the Caliph and Emir project: http://www.SemanticMetadata.net
 * <br>Date: 31.01.2006
 * <br>Time: 23:00:32
 *
 * @author Mathias Lux, mathias@juggle.at
 */
public class DocumentBuilderFactory {
    /**
     * Creates a simple version of a DocumentBuilder. In this case the
     * {@link net.semanticmetadata.lire.imageanalysis.CEDD} is used as a feature
     *
     * @return a simple and efficient DocumentBuilder.
     * @see net.semanticmetadata.lire.imageanalysis.CEDD
     */
    public static DocumentBuilder getDefaultDocumentBuilder() {
        return new GenericFastDocumentBuilder(CEDD.class, DocumentBuilder.FIELD_NAME_CEDD);
    }

    /**
     * Creates a simple version of a DocumentBuilder using the MPEG/-7 visual features features
     * all available descriptors are used.
     *
     * @return a fully featured DocumentBuilder.
     * @see net.semanticmetadata.lire.imageanalysis.ColorLayout
     * @see net.semanticmetadata.lire.imageanalysis.EdgeHistogram
     * @see net.semanticmetadata.lire.imageanalysis.ScalableColor
     * @deprecated Use ChainedDocumentBuilder instead
     */
    public static DocumentBuilder getExtensiveDocumentBuilder() {
        ChainedDocumentBuilder cb = new ChainedDocumentBuilder();
        cb.addBuilder(DocumentBuilderFactory.getColorLayoutBuilder());
        cb.addBuilder(DocumentBuilderFactory.getEdgeHistogramBuilder());
        cb.addBuilder(DocumentBuilderFactory.getScalableColorBuilder());
        return cb;
    }

    /**
     * Creates a fast (byte[] based) version of the MPEG-7 ColorLayout document builder.
     *
     * @return the document builder.
     */
    public static DocumentBuilder getColorLayoutBuilder() {
        return new GenericFastDocumentBuilder(ColorLayout.class, DocumentBuilder.FIELD_NAME_COLORLAYOUT);
    }

    /**
     * Creates a fast (byte[] based) version of the MPEG-7 EdgeHistogram document builder.
     *
     * @return the document builder.
     */
    public static DocumentBuilder getEdgeHistogramBuilder() {
        return new GenericFastDocumentBuilder(EdgeHistogram.class, DocumentBuilder.FIELD_NAME_EDGEHISTOGRAM);
    }

    /**
     * Creates a fast (byte[] based) version of the MPEG-7 ColorLayout document builder.
     *
     * @return the document builder.
     */
    public static DocumentBuilder getScalableColorBuilder() {
        return new GenericFastDocumentBuilder(ScalableColor.class, DocumentBuilder.FIELD_NAME_SCALABLECOLOR);
```

```
    }

    /**
     * Creates a simple version of a DocumentBuilder using ScalableColor.
     *
     * @return a fully featured DocumentBuilder.
     * @see net.semanticmetadata.lire.imageanalysis.ScalableColor
     * @deprecated Use ColorHistogram and the respective factory methods to get it instead
     */
    public static DocumentBuilder getColorOnlyDocumentBuilder() {
        return DocumentBuilderFactory.getScalableColorBuilder();
    }

    /**
     * Creates a simple version of a DocumentBuilder using the ColorLayout feature. Don't use this method any more but
     * use the respective feature bound method instead.
     *
     * @return a simple and fast DocumentBuilder.
     * @see net.semanticmetadata.lire.imageanalysis.ColorLayout
     * @deprecated use MPEG-7 feature ColorLayout or CEDD, which are both really fast.
     */
    public static DocumentBuilder getFastDocumentBuilder() {
        return DocumentBuilderFactory.getColorLayoutBuilder();
    }

    /**
     * Creates a DocumentBuilder for the AutoColorCorrelation feature. Note that the extraction of this feature
     * is especially slow! So use it only on small images! Images that do not fit in a 200x200 pixel box are
     * resized by the document builder to ensure shorter processing time. See
     * {@link net.semanticmetadata.lire.imageanalysis.AutoColorCorrelogram} for more information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created AutoCorrelation feature DocumentBuilder.
     */
    public static DocumentBuilder getAutoColorCorrelogramDocumentBuilder() {
        return new GenericDocumentBuilder(AutoColorCorrelogram.class, DocumentBuilder.FIELD_NAME_AUTOCOLORCORRELOGRAM, GenericDocumen
Builder.Mode.Fast);
    }

    /**
     * Creates a DocumentBuilder for the AutoColorCorrelation feature. Note that the extraction of this feature
     * is especially slow, but this is a more fast, but less accurate settings version!
     * Images that do not fit in a defined bounding box they are
     * resized by the document builder to ensure shorter processing time. See
     * {@link net.semanticmetadata.lire.imageanalysis.AutoColorCorrelogram} for more information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created AutoCorrelation feature DocumentBuilder.
     * @deprecated Use #getAutoColorCorrelogramDocumentBuilder instead.
     */
    public static DocumentBuilder getFastAutoColorCorrelationDocumentBuilder() {
        return new CorrelogramDocumentBuilder(AutoColorCorrelogram.Mode.SuperFast);
    }

    /**
     * Creates a DocumentBuilder for the CEDD feature. See
     * {@link net.semanticmetadata.lire.imageanalysis.CEDD} for more information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created CEDD feature DocumentBuilder.
     */
    public static DocumentBuilder getCEDDDocumentBuilder() {
//        return new CEDDDocumentBuilder();
        return new GenericFastDocumentBuilder(CEDD.class, DocumentBuilder.FIELD_NAME_CEDD);
    }


    /**
     * Creates a DocumentBuilder for the FCTH feature. See
     * {@link net.semanticmetadata.lire.imageanalysis.FCTH} for more information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created FCTH feature DocumentBuilder.
     */
    public static DocumentBuilder getFCTHDocumentBuilder() {
        return new GenericDocumentBuilder(FCTH.class, DocumentBuilder.FIELD_NAME_FCTH, GenericDocumentBuilder.Mode.Fast);
    }

    /**
     * Creates a DocumentBuilder for the JCD feature. See
     * {@link net.semanticmetadata.lire.imageanalysis.JCD} for more information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created DocumentBuilder
     */
    public static DocumentBuilder getJCDDocumentBuilder() {
        return new GenericFastDocumentBuilder(JCD.class, DocumentBuilder.FIELD_NAME_JCD);
    }

    /**
     * Creates a DocumentBuilder for the JpegCoefficientHistogram feature. See
```

```
     * {@link net.semanticmetadata.lire.imageanalysis.JpegCoefficientHistogram} for more
     * information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created DocumentBuilder
     */
    public static DocumentBuilder getJpegCoefficientHistogramDocumentBuilder() {
        return new GenericDocumentBuilder(JpegCoefficientHistogram.class, DocumentBuilder.FIELD_NAME_JPEGCOEFFS, GenericDocumentBuild
r.Mode.Fast);
    }

    /**
     * Creates a DocumentBuilder for simple RGB color histograms. See
     * {@link net.semanticmetadata.lire.imageanalysis.SimpleColorHistogram} for more
     * information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created feature DocumentBuilder.
     */
    public static DocumentBuilder getColorHistogramDocumentBuilder() {
        return new GenericDocumentBuilder(SimpleColorHistogram.class, DocumentBuilder.FIELD_NAME_COLORHISTOGRAM, GenericDocumentBuild
r.Mode.Fast);
    }

    /**
     * Creates a DocumentBuilder for three Tamura features. See
     * {@link net.semanticmetadata.lire.imageanalysis.Tamura} for more
     * information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created Tamura feature DocumentBuilder.
     */
    public static DocumentBuilder getTamuraDocumentBuilder() {
        return new GenericFastDocumentBuilder(Tamura.class, DocumentBuilder.FIELD_NAME_TAMURA);
    }

    /**
     * Creates a DocumentBuilder for the Gabor feature. See
     * {@link net.semanticmetadata.lire.imageanalysis.Gabor} for more
     * information on the image feature.
     * Be sure to use the same options for the ImageSearcher as you used for the DocumentBuilder.
     *
     * @return the created Tamura feature DocumentBuilder.
     */
    public static DocumentBuilder getGaborDocumentBuilder() {
        return new GenericFastDocumentBuilder(Gabor.class, DocumentBuilder.FIELD_NAME_GABOR);
    }


    /**
     * Creates and returns a DocumentBuilder, which contains all available features. For
     * AutoColorCorrelogram the getAutoColorCorrelogramDocumentBuilder() is used. Therefore
     * it is compatible with the respective Searcher.
     *
     * @return a combination of all available features.
     */
    public static DocumentBuilder getFullDocumentBuilder() {
        ChainedDocumentBuilder cdb = new ChainedDocumentBuilder();
        cdb.addBuilder(DocumentBuilderFactory.getExtensiveDocumentBuilder());
        cdb.addBuilder(DocumentBuilderFactory.getAutoColorCorrelogramDocumentBuilder());
        cdb.addBuilder(DocumentBuilderFactory.getCEDDDocumentBuilder());
        cdb.addBuilder(DocumentBuilderFactory.getFCTHDocumentBuilder());
        cdb.addBuilder(DocumentBuilderFactory.getColorHistogramDocumentBuilder());
        cdb.addBuilder(DocumentBuilderFactory.getTamuraDocumentBuilder());
        cdb.addBuilder(DocumentBuilderFactory.getGaborDocumentBuilder());
        return cdb;
    }
}
```

DocumentBuilderFactory是用于创建DocumentBuilder的。里面有各种get****DocumentBuilder()。其中以下2种是几个DocumentBuilder的合集：

getExtensiveDocumentBuilder()：使用MPEG-7中的ColorLayout，EdgeHistogram，ScalableColor

getFullDocumentBuilder()：使用所有的DocumentBuilder

文章标签： lire 源代码 索引 检索 lucene

个人分类： LIRe MPEG7/图像检索

所属专栏： 开源多媒体项目源代码分析