链滴

分享一下 java 处理图像的工具类

作者：s1121937257
原文链接：https://ld246.com/article/1479715216943
来源网站：链滴

```
package com.mss.utils;

import java.awt.AlphaComposite;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.color.ColorSpace;
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.awt.image.ColorConvertOp;
import java.awt.image.CropImageFilter;
import java.awt.image.FilteredImageSource;
import java.awt.image.ImageFilter;
import java.io.File;
import java.io.IOException;

import javax.imageio.ImageIO;

/**

● 图片处理类

●

*/
@SuppressWarnings("unused")
public class ImgUtils {


private String IMAGE_TYPE_GIF = "gif";// 图形交换格式
private String IMAGE_TYPE_JPG = "jpg";// 联合照片专家组
private String IMAGE_TYPE_JPEG = "jpeg";// 联合照片专家组
private String IMAGE_TYPE_BMP = "bmp";// 英文Bitmap（位图）的简写，它是Windows操作系
中的标准图像文件格式
private String IMAGE_TYPE_PNG = "png";// 可移植网络图形
private String IMAGE_TYPE_PSD = "psd";// Photoshop的专用格式Photoshop

/**
 * 按比例缩放图片
 *
 *
 * @date 2016年11月21日
```

```
 * @param srcImageFile 文件路径
 * @param result 文件存放路径
 * @param scale 放大或缩小倍数
 * @param flag true：放大,false:缩小
 * void
 */
public void scale(String srcImageFile, String result,
        int scale, boolean flag) {
    try {
        BufferedImage src = ImageIO.read(new File(srcImageFile)); // 读入文件
        int width = src.getWidth(); // 得到源图宽
        int height = src.getHeight(); // 得到源图长
        if (flag) {// 放大
            width = width * scale;
            height = height * scale;
        } else {// 缩小
            width = width / scale;
            height = height / scale;
        }
        Image image = src.getScaledInstance(width, height,Image.SCALE_DEFAULT);
        BufferedImage tag = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB);
        Graphics g = tag.getGraphics();
        g.drawImage(image, 0, 0, null); // 绘制缩小后的图
        g.dispose();
        ImageIO.write(tag, "JPEG", new File(result));// 输出到文件流
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/**
 *
 * 根据宽高缩放
 *
 * @date 2016年11月21日
 * @param srcImageFile  文件路径
 * @param result 存放文件路径
 * @param height 缩放高度
 * @param width 缩放宽度
 * @param bb 是否缩放
 * void
 */
@SuppressWarnings("static-access")
public  void scale2(String srcImageFile, String result, int height, int width, boolean bb) {
    try {
        double ratio = 0.0; // 缩放比例
        File f = new File(srcImageFile);
        BufferedImage bi = ImageIO.read(f);
        Image itemp = bi.getScaledInstance(width, height, bi.SCALE_SMOOTH);
        // 计算比例
        if ((bi.getHeight() > height) || (bi.getWidth() > width)) {
            if (bi.getHeight() > bi.getWidth()) {
                ratio = (new Integer(height)).doubleValue() / bi.getHeight();
            } else {
```

```
            ratio = (new Integer(width)).doubleValue() / bi.getWidth();
        }
        AffineTransformOp op = new AffineTransformOp(AffineTransform
                .getScaleInstance(ratio, ratio), null);
        itemp = op.filter(bi, null);
    }
    if (bb) {//补白
        BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_R
B);
        Graphics2D g = image.createGraphics();
        g.setColor(Color.white);
        g.fillRect(0, 0, width, height);
        if (width == itemp.getWidth(null))
            g.drawImage(itemp, 0, (height - itemp.getHeight(null)) / 2,itemp.getWidth(null), ite
p.getHeight(null),
                    Color.white, null);
        else
            g.drawImage(itemp, (width - itemp.getWidth(null)) / 2, 0,itemp.getWidth(null), itemp
getHeight(null),
                    Color.white, null);
        g.dispose();
        itemp = image;
    }
    ImageIO.write((BufferedImage) itemp, "JPEG", new File(result));
  } catch (IOException e) {
    e.printStackTrace();
  }
}

/**
 *
 * 按指定起点坐标和宽高 进行切割
 * 注意 y+height！= 图片本身的高度，否则会以补白填充。
 * 坐标0，0 是以坐上角开始
 * @date 2016年11月21日
 * @param srcImageFile
 * @param result
 * @param x
 * @param y
 * @param width
 * @param height
 * void
 */
public void cut(String srcImageFile, String result,
    int x, int y, int width, int height) {
  try {
    // 读取源图像
    BufferedImage bi = ImageIO.read(new File(srcImageFile));
    int srcWidth = bi.getHeight(); // 源图宽度
    int srcHeight = bi.getWidth(); // 源图高度
    if (srcWidth > 0 && srcHeight > 0) {
        Image image = bi.getScaledInstance(srcWidth, srcHeight,Image.SCALE_DEFAULT);
        // 四个参数分别为图像起点坐标和宽高
        // 即: CropImageFilter(int x,int y,int width,int height)
```

```
            ImageFilter cropFilter = new CropImageFilter(x, y, width, height);
            Image img = Toolkit.getDefaultToolkit().createImage(new FilteredImageSource(image.
etSource(),cropFilter));
            BufferedImage tag = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB)

            Graphics g = tag.getGraphics();
            g.drawImage(img, 0, 0, width, height, null); // 绘制切割后的图
            g.dispose();
            // 输出为文件
            ImageIO.write(tag, "JPEG", new File(result));
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 *
 * 指定切片的行数和列数
 *
 * @date 2016年11月21日
 * @param srcImageFile
 * @param descDir
 * @param rows
 * @param cols
 * void
 */
public  void cut2(String srcImageFile, String descDir,
        int rows, int cols) {
    try {
        if(rows<=0||rows>20) rows = 2; // 切片行数
        if(cols<=0||cols>20) cols = 2; // 切片列数
        // 读取源图像
        BufferedImage bi = ImageIO.read(new File(srcImageFile));
        int srcWidth = bi.getHeight(); // 源图宽度
        int srcHeight = bi.getWidth(); // 源图高度
        if (srcWidth > 0 && srcHeight > 0) {
            Image img;
            ImageFilter cropFilter;
            Image image = bi.getScaledInstance(srcWidth, srcHeight, Image.SCALE_DEFAULT);
            int destWidth = srcWidth; // 每张切片的宽度
            int destHeight = srcHeight; // 每张切片的高度
            // 计算切片的宽度和高度
            if (srcWidth % cols == 0) {
                destWidth = srcWidth / cols;
            } else {
                destWidth = (int) Math.floor(srcWidth / cols) + 1;
            }
            if (srcHeight % rows == 0) {
                destHeight = srcHeight / rows;
            } else {
                destHeight = (int) Math.floor(srcWidth / rows) + 1;
            }
            // 循环建立切片
```

```
            // 改进的想法:是否可用多线程加快切割速度
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                    // 四个参数分别为图像起点坐标和宽高
                    // 即: CropImageFilter(int x,int y,int width,int height)
                    cropFilter = new CropImageFilter(j * destWidth, i * destHeight,destWidth, destHe
ght);
                    img = Toolkit.getDefaultToolkit().createImage(new FilteredImageSource(image.ge
Source(),cropFilter));
                    BufferedImage tag = new BufferedImage(destWidth,destHeight, BufferedImage.T
PE_INT_RGB);
                    Graphics g = tag.getGraphics();
                    g.drawImage(img, 0, 0, null); // 绘制缩小后的图
                    g.dispose();
                    // 输出为文件
                    ImageIO.write(tag, "JPEG", new File(descDir
                            + "_r" + i + "_c" + j + ".jpg"));
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 *
 * 指定切片的宽度和高度
 *
 * @date 2016年11月21日
 * @param srcImageFile
 * @param descDir
 * @param destWidth
 * @param destHeight
 * void
 */
public void cut3(String srcImageFile, String descDir,
        int destWidth, int destHeight) {
    try {
        if(destWidth<=0) destWidth = 200; // 切片宽度
        if(destHeight<=0) destHeight = 150; // 切片高度
        // 读取源图像
        BufferedImage bi = ImageIO.read(new File(srcImageFile));
        int srcWidth = bi.getHeight(); // 源图宽度
        int srcHeight = bi.getWidth(); // 源图高度
        if (srcWidth > destWidth && srcHeight > destHeight) {
            Image img;
            ImageFilter cropFilter;
            Image image = bi.getScaledInstance(srcWidth, srcHeight, Image.SCALE_DEFAULT);
            int cols = 0; // 切片横向数量
            int rows = 0; // 切片纵向数量
            // 计算切片的横向和纵向数量
            if (srcWidth % destWidth == 0) {
                cols = srcWidth / destWidth;
```

```
            } else {
                cols = (int) Math.floor(srcWidth / destWidth) + 1;
            }
            if (srcHeight % destHeight == 0) {
                rows = srcHeight / destHeight;
            } else {
                rows = (int) Math.floor(srcHeight / destHeight) + 1;
            }
            // 循环建立切片
            // 改进的想法:是否可用多线程加快切割速度
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                    // 四个参数分别为图像起点坐标和宽高
                    // 即: CropImageFilter(int x,int y,int width,int height)
                    cropFilter = new CropImageFilter(j * destWidth, i * destHeight,destWidth, destHe
ght);
                    img = Toolkit.getDefaultToolkit().createImage(new FilteredImageSource(image.ge
Source(),cropFilter));
                    BufferedImage tag = new BufferedImage(destWidth,destHeight, BufferedImage.T
PE_INT_RGB);
                    Graphics g = tag.getGraphics();
                    g.drawImage(img, 0, 0, null); // 绘制缩小后的图
                    g.dispose();
                    // 输出为文件
                    ImageIO.write(tag, "JPEG", new File(descDir
                            + "_r" + i + "_c" + j + ".jpg"));
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
/**
 *
 * 图像类型转换
 *
 * @date 2016年11月21日
 * @param srcImageFile 文件路径
 * @param formatName  转换格式
 * @param destImageFile  文件存储路径
 * void
 */
public void convert(String srcImageFile, String formatName, String destImageFile) {
    try {
        File f = new File(srcImageFile);
        f.canRead();
        f.canWrite();
        BufferedImage src = ImageIO.read(f);
        ImageIO.write(src, formatName, new File(destImageFile));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
/**
 *
 * 彩色图片转黑白
 *
 * @date 2016年11月21日
 * @param srcImageFile
 * @param destImageFile
 * void
 */
public void gray(String srcImageFile, String destImageFile) {
        try {
            BufferedImage src = ImageIO.read(new File(srcImageFile));
            ColorSpace cs = ColorSpace.getInstance(ColorSpace.CS_GRAY);
            ColorConvertOp op = new ColorConvertOp(cs, null);
            src = op.filter(src, null);
            ImageIO.write(src, "JPEG", new File(destImageFile));
    } catch (IOException e) {
        e.printStackTrace();
    }
}

/**
 *
 * 给图片添加文字水印
 *
 * @date 2016年11月21日
 * @param pressText 水印文字
 * @param srcImageFile 文件路径
 * @param destImageFile 文件保存路径
 * @param fontName 字体名称 例如：宋体
 * @param fontStyle 字体风格
 * @param color 字体颜色
 * @param fontSize 字体大小
 * @param x 文字的起始坐标
 * @param y
 * @param alpha  文字透明度
 * void
 */
public void pressText(String pressText,
        String srcImageFile, String destImageFile, String fontName,
        int fontStyle, Color color, int fontSize,int x,
        int y, float alpha) {
    try {
        File img = new File(srcImageFile);
        Image src = ImageIO.read(img);
        int width = src.getWidth(null);
        int height = src.getHeight(null);
        BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB)

        Graphics2D g = image.createGraphics();
        g.drawImage(src, 0, 0, width, height, null);
        g.setColor(color);
        g.setFont(new Font(fontName, fontStyle, fontSize));
```

```
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_ATOP,alpha));
        // 在指定坐标绘制水印文字
        g.drawString(pressText, (width - (getLength(pressText) * fontSize))/ 2 + x, (height - fontSi
e) / 2 + y);
        g.dispose();
        ImageIO.write((BufferedImage) image, "JPEG", new File(destImageFile));// 输出到文件流
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
 *
 *
 *
 * @date 2016年11月21日
 * @param pressText        水印文字
 * @param srcImageFile        文件路径
 * @param destImageFile     文件保存路径
 * @param fontName            字体名称 例如：宋体
 * @param fontStyle        字体风格
 * @param color            字体颜色
 * @param fontSize            字体大小
 * @param x                文字的起始坐标
 * @param y
 * @param alpha            文字透明度
 * void
 */
public void pressText2(String pressText, String srcImageFile,String destImageFile,
        String fontName, int fontStyle, Color color, int fontSize, int x,
        int y, float alpha) {
    try {
        File img = new File(srcImageFile);
        Image src = ImageIO.read(img);
        int width = src.getWidth(null);
        int height = src.getHeight(null);
        BufferedImage image = new BufferedImage(width, height,BufferedImage.TYPE_INT_RGB)

        Graphics2D g = image.createGraphics();
        g.drawImage(src, 0, 0, width, height, null);
        g.setColor(color);
        g.setFont(new Font(fontName, fontStyle, fontSize));
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_ATOP,alpha));
        // 在指定坐标绘制水印文字
        g.drawString(pressText, (width - (getLength(pressText) * fontSize))/ 2 + x, (height - fontSi
e) / 2 + y);
        g.dispose();
        ImageIO.write((BufferedImage) image, "JPEG", new File(destImageFile));
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**
```

```
 *
 * 给图片添加水印图片
 *
 * @date 2016年11月21日
 * @param pressImg            水印文件路径
 * @param srcImageFile        原文件路径
 * @param destImageFile       存储文件路径
 * @param x
 * @param y
 * @param alpha               透明度
 * void
 */
public void pressImage(String pressImg, String srcImageFile,String destImageFile,
        int x, int y, float alpha) {
    try {
        File img = new File(srcImageFile);
        Image src = ImageIO.read(img);
        int wideth = src.getWidth(null);
        int height = src.getHeight(null);
        BufferedImage image = new BufferedImage(wideth, height,
                BufferedImage.TYPE_INT_RGB);
        Graphics2D g = image.createGraphics();
        g.drawImage(src, 0, 0, wideth, height, null);
        // 水印文件
        Image src_biao = ImageIO.read(new File(pressImg));
        int wideth_biao = src_biao.getWidth(null);
        int height_biao = src_biao.getHeight(null);
        g.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_ATOP,
                alpha));
        g.drawImage(src_biao, (wideth - wideth_biao) / 2,
                (height - height_biao) / 2, wideth_biao, height_biao, null);
        // 水印文件结束
        g.dispose();
        ImageIO.write((BufferedImage) image,  "JPEG", new File(destImageFile));
    } catch (Exception e) {
        e.printStackTrace();
    }
}


public static int getLength(String text) {
    int length = 0;
    for (int i = 0; i < text.length(); i++) {
        if (new String(text.charAt(i) + "").getBytes().length > 1) {
            length += 2;
        } else {
            length += 1;
        }
    }
    return length / 2;
}

}
```