链滴

JAVA 通过 BufferedImage 进行图片的绘制，缩放，裁剪，水印等操作

作者：hjljy
原文链接：https://ld246.com/article/1583055802905
来源网站：链滴

最近开发当中，通过JAVA对图片进行了很多的操作，之前很少接触这方面的知识，特此记录下来

读取图片

```
//读取图片  通过JAVA自带的ImageIO里面的read方法
BufferedImage bufImage = ImageIO.read(File input);
BufferedImage bufImage = ImageIO.read(URL input);
BufferedImage bufImage = ImageIO.read(InputStream input);
BufferedImage bufImage = ImageIO.read(ImageInputStream input)
```

保存图片

```
/**
 *  image:RenderedImage 接口的实现类, BufferedImage 实现了 RenderedImage 接口
 *  formatName: 保存的图片格式的名称
 *  output: 结果输出位置
 */
ImageIO.write(RenderedImage image, String formatName, File output);
ImageIO.write(RenderedImage image, String formatName, OutputStream output);
```

绘制图片

```
/**
 * 创建一个指定宽高的图片内存对象
 * width : 宽度
 * height : 高度
 * imageType : 图片类型   参数为BufferedImage 自身定义的常量
 *    TYPE_3BYTE_BGR ： 代表8位RGB分量图像
       TYPE_INT_ARGB ： 代表8位RGBA颜色组件包装成整数像素的图像
    ......
```

原文链接：JAVA 通过 BufferedImage 进行图片的绘制，缩放，裁剪，水印等操作

```
 */
BufferedImage image = new BufferedImage(width, height, imageType);
//获取图片的画布
Graphics2D graphics = image.createGraphics();

//然后使用 Graphics 类在图片上绘制线段、矩形、图片、文本，设置背景颜色等等操作

// 设置画布颜色
void setColor(Color c)
// 设置字体颜色
void setFont(Font font)
// 设置线的宽度
setStroke(Stroke s)
// 设置背景颜色
void setBackground(Color c)
// 擦除某一区域（擦除后显示背景色）
void clearRect(int x, int y, int width, int height)
// 回收 Graphics 释放资源   操作完毕后一定要释放资源
void dispose()

// 绘制一条线段（如果两点为同一点，则绘制点）
void  drawLine(int x1,  int y1,  int x2,  int y2)
// 绘制一个矩形（空心）
void  drawRect(int x,  int y,  int width,  int height)
// 绘制一个椭圆（空心）
void  drawOval(int x,  int y,  int width,  int height)
// 绘制一个圆弧（弧线）
void  drawArc(int x,  int y,  int width,  int height,  int startAngle,  int arcAngle)
// 绘制一张图片 ImageObserver:接收有关 Image 信息通知的异步更新接口,没用到直接传空
boolean  drawImage(Image image,  int x,  int y, ImageObserver observer)
// 绘制一段文本
void  drawString(String str,  int x,  int y)
.............................
```

缩放图片

```
/**
 *@param   width: 缩放后的宽度
 *@param   height: 缩放后的高度
 *@param   hints:  图像重采样算法的类型
 *
 *  hints 参数取值为以下之一（Image 类中的常量）：
 *     SCALE_AREA_AVERAGING: 使用 Area Averaging 图像缩放算法;
 *     SCALE_DEFAULT: 使用默认的图像缩放算法;
 *     SCALE_SMOOTH: 选择图像平滑度比缩放速度具有更高优先级的图像缩放算法。
 *     ......
 */
Image getScaledInstance(int width, int height, int hints);
```

缩放的使用实例代码：

```
  /**
     * JAVA 图像等比缩放
```

```
     * @param srcImageFile  缩放的图片
     * @param destImageFile  缩放后的图片
     * @param scale  缩放比例
     * @return
     */
    public static boolean scale(File srcImageFile, File destImageFile,float scale){
        try {
            //使用ImageIO的read方法读取图片
            BufferedImage read = ImageIO.read(srcImageFile);
            //获取缩放后的宽高
            int width = (int) (read.getWidth()*scale);
            int height = (int) (read.getHeight()*scale);
            //调用缩放方法获取缩放后的图片
            Image img = read.getScaledInstance(width , height, Image.SCALE_DEFAULT);
            //创建一个新的缓存图片
            BufferedImage image = new BufferedImage(width, height, BufferedImage.TYPE_INT_R
B);
            //获取画笔
            Graphics2D graphics = image.createGraphics();
            //将Image对象画在画布上,最后一个参数,ImageObserver:接收有关 Image 信息通知的异步
新接口,没用到直接传空
            graphics.drawImage(img, 0, 0,null);
            //一定要释放资源
            graphics.dispose();
            //获取到文件的后缀名
            String fileName = srcImageFile.getName();
            String formatName = fileName.substring(fileName.lastIndexOf(".") + 1);
            //使用ImageIO的write方法进行输出
            ImageIO.write(image,formatName,destImageFile);
        } catch (IOException e) {
            e.printStackTrace();
            return false;
        }
        return true;
    }
```

裁剪图片

```
/**
*
* @param x     裁剪时x的坐标（左上角）
* @param y     裁剪时y的坐标（左上角）
* @param width  裁剪后的图片宽度
* @param height 裁剪后的图片高度
*
* 裁剪后返回的是一个新的图片缓存对象
*/
 BufferedImage  getSubimage(int x,int y, int width  ,int height )
```

裁剪图片实例代码:

```
    /**
     * JAVA裁剪图片
     * @param srcImageFile  需要裁剪的图片
```

```
     * @param x      裁剪时x的坐标（左上角）
     * @param y      裁剪时y的坐标（左上角）
     * @param width  裁剪后的图片宽度
     * @param height 裁剪后的图片高度
     * @param destImageFile  裁剪后的图片
     * @return
     */
    public static boolean cut(File srcImageFile, int x,int y,int width,int height,File destImageFile){
        try {
            //使用ImageIO的read方法读取图片
            BufferedImage read = ImageIO.read(srcImageFile);
            //调用裁剪方法
            BufferedImage image = read.getSubimage(x, y, width, height);
            //获取到文件的后缀名
            String fileName = srcImageFile.getName();
            String formatName = fileName.substring(fileName.lastIndexOf(".") + 1);
            //使用ImageIO的write方法进行输出
            ImageIO.write(image,formatName,destImageFile);
        } catch (IOException e) {
            e.printStackTrace();
            return  false;
        }
        return true;
    }
```

添加水印

水印的添加主要是通过下面的方法进行的添加。

```
// 绘制一张图片 ImageObserver:接收有关 Image 信息通知的异步更新接口,没用到直接传空
boolean  drawImage(Image image,  int x,  int y, ImageObserver observer)
//设置水印透明度
void setComposite(Composite comp)
// 绘制一段文本
void  drawString(String str,  int x,  int y)
```

文本水印

```
    /**
     *  JAVA添加文字水印
     * @param srcImageFile  目标图片
     * @param destImageFile 结果图片
     * @param text          文字内容
     * @param x             水印x坐标
     * @param y             水印y坐标
     * @return
     */
    public static boolean watermarkText(File srcImageFile, File destImageFile, String text,int x,in
y) {
        try {
            //使用ImageIO的read方法读取图片
            BufferedImage read = ImageIO.read(srcImageFile);
```

```
            Graphics2D graphics = read.createGraphics();
        // 设置“抗锯齿”的属性
        graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_
NTIALIAS_ON);
            // 设置字体类型和大小
            graphics.setFont(new Font("微软雅黑", Font.PLAIN, 20));
            // 设置颜色
            graphics.setColor(new Color(255,255,255));
            // 添加文字
            graphics.drawString(text,x,y);
            graphics.dispose();
            //获取到文件的后缀名
            String fileName = srcImageFile.getName();
            String formatName = fileName.substring(fileName.lastIndexOf(".") + 1);
            //使用ImageIO的write方法进行输出
            ImageIO.write(read,formatName,destImageFile);
        } catch (IOException e) {
            e.printStackTrace();
        }
        return true;
    }
```

图片水印

```
    /**
     *  JAVA添加文字水印
     * @param srcImageFile  目标图片
     * @param destImageFile 结果图片
     * @param waterImage    水印图片
     * @param x             水印x坐标
     * @param y             水印y坐标
     * @return
     */
    public static boolean watermarkImage(File srcImageFile,File destImageFile, File waterImage,
nt x,int y) {
        try {
            //使用ImageIO的read方法读取图片
            BufferedImage read = ImageIO.read(srcImageFile);
            BufferedImage image = ImageIO.read(waterImage);
            //获取画布
            Graphics2D graphics = read.createGraphics();
            //设置透明度为0.5
            graphics.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_ATOP,0.5f));
            //添加水印
            graphics.drawImage(image,x,y,null);
            //关闭透明度
            //graphics.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER));
            graphics.dispose();
            //获取到文件的后缀名
            String fileName = srcImageFile.getName();
            String formatName = fileName.substring(fileName.lastIndexOf(".") + 1);
            //使用ImageIO的write方法进行输出
            ImageIO.write(read,formatName,destImageFile);
        } catch (IOException e) {
```

```
            e.printStackTrace();
        }
        return true;
    }
```

如何擦除水印?

第一种方法：

// 擦除某一区域（擦除后显示背景色）

void clearRect(int x, int y, int width, int height)

第二种方法：色素替代法

找到水印的颜色编码，然后用背景色颜色编码替代。

代码实现：略　　（这种清除水印的需求还是交给PS这种专业软件去做吧）

贝塞尔曲线

通常绘制线段直接使用一下的方法就可以了drawLine方法就可以了。但是在实现曲线的时候就很难看所以需要用到贝塞尔曲线。

可以通过Path类来实现贝塞尔曲线的效果

```
BufferedImage image = new BufferedImage(500, 500, BufferedImage.TYPE_INT_RGB);
Graphics2D graphics = image.createGraphics();
// 设置“抗锯齿”的属性
graphics.setRenderingHint(RenderingHints.KEY_ANTIALIASING, RenderingHints.VALUE_ANTIA
IAS_ON);
//获取路径操作
Path2D path = new GeneralPath();
//通过移动到指定的浮点精度指定的指定的坐标，向路径添加一个点。
path.moveTo(192, 18);
//添加一个弯曲段，由两个新定义的点，通过绘制一条二次曲线相交的当前坐标和指定的坐标 (x2,y2)
路径，使用指定的点 (x1,y1)作为二次参考
path.quadTo(120, 12, 253, 67);
graphics.draw(path);
graphics.dispose();
```

总结

JAVA操作图片的话算是一个非常冷门的功能了，最多就是图片的缩放，裁剪，水印这三种情况。如需要进行一些额外的操作的话，就非常需要认真的查阅下JDK的API，并且JAVA操作图片调试起来也常的不方便。总的来说就是复杂的图片操作需求还是交给更专业的软件（PS）来进行操作吧!!!

JDK1.8在线API