案例：<https://gitee.com/zx19890628/spring-boot-example/tree/master/lab_069_util_image>

# 常见的图像处理工具

Java中进行图像I/O（即读图片和写图片，不涉及到复杂图像处理）有三个方法：

1. Java Image I/O API，支持常见图片，从Java 2 version 1.4.0开始就内置了。

主页：http://java.sun.com/javase/6/docs/technotes/guides/imageio/index.html

2. JAI 中的 Image I/O Tools，支持更多图片类型，例如JPEG-LS, JPEG2000, 和 TIFF。是Oracle提供的一个Java扩展。

JAI 是一个关于图像处理的框架，很庞大，其中仅仅jai-imageio是关于图像I/O的，其他的可以不看。

从2016年就不维护了，不建议使用

3. JAI的com.sun.media.jai.codec 也有一定的图像解码能力

当然，还有众多的java开源工具包可以读写图像，例如JIMI, JMagic等，但JDK目前本身能

够读写图片，就用JDK的，开发和部署方便，不需要额外下载jar包。

如果你仅仅想读取常见格式的图片，不需要用JAI这么高级这么庞大的东西，用Java Image I/O API即可。

4.Thumbnailator

<https://www.cnblogs.com/tangjiang-code/p/7911527.html>

5.ImageMagick

<https://www.cnblogs.com/chanshuyi/p/6813540.html>

6.SimpleImage

SimpleImage是阿里巴巴的一个Java图片处理的类库，可以实现图片缩略、水印等处理。

地址：https://github.com/alibaba/simpleimage

# 使用JDK处理图像

Java Image I/O API 主要在 javax.imageio 下面。JDK已经内置了常见图片格式的插件，

但它提供了插件体系结构，第三方也可以开发插件支持其他图片格式。

案例中使用的是JDK自带的ImageIO

## 获得ImageIO运行读写图片的类型

public static void main(String args[]) {

String readFormats[] = ImageIO.getReaderFormatNames();

String writeFormats[] = ImageIO.getWriterFormatNames();

System.out.println("Readers:" + Arrays.asList(readFormats));

System.out.println("Writers:" + Arrays.asList(writeFormats));

// Readers:[JPG, jpg, bmp, BMP, gif, GIF, WBMP, png, PNG, wbmp, jpeg, JPEG]

// Writers:[JPG, jpg, bmp, BMP, gif, GIF, WBMP, png, PNG, wbmp, jpeg, JPEG]

}

## 读取图片的属性

public static void main(String[] args) throws IOException {

File img = new File("src/main/resources/img/demo.jpg");

BufferedImage image = ImageIO.read(img);

System.out.println("高=" + image.getHeight());

System.out.println("宽=" + image.getWidth());

System.out.println("类型=" + image.getType());

}

## 读取图片后生产一个新的图片

public static void main(String[] args) throws IOException {

File img = new File("src/main/resources/img/demo.jpg");

BufferedImage image = ImageIO.read(img);

ImageIO.write(image, "jpg", new File("D://demo.png"));

}

## 切割图片

public static void main(String[] args) throws IOException {

// 获得一个图片读取器

Iterator<ImageReader> iterator = ImageIO.getImageReadersByFormatName("jpg");

ImageReader reader = (ImageReader)iterator.next();

// 将图片装换为图片的输入流

File file = new File("src/main/resources/img/demo.jpg");

ImageInputStream iis = ImageIO.createImageInputStream(file);

reader.setInput(iis);

// 读取图片的参数（ImageReadParam支持读取一个文件有多个图片）

ImageReadParam param = reader.getDefaultReadParam();

System.out.println("文件中图片的数量=" + reader.getNumImages(true));

int imageIndex = 0;//读取文件中第一个图片

//切割图片的起始位置

int width\_begin = 0;

int height\_begin = 0;

int width\_end = reader.getWidth(imageIndex)/2;

int height\_end = reader.getHeight(imageIndex)/2;

Rectangle rect = new Rectangle(width\_begin, height\_begin, width\_end, height\_end);

param.setSourceRegion(rect);

// 获得切割后图片的对象

BufferedImage bi = reader.read(imageIndex, param);

ImageIO.write(bi, "jpg", new File("D://11.png"));

iis.close();

}

## 生成缩略图

public static void main(String[] args) throws IOException {

// 获得一个图片读取器

Iterator<ImageReader> iterator = ImageIO.getImageReadersByFormatName("jpg");

ImageReader reader = (ImageReader)iterator.next();

// 将图片装换为图片的输入流

File file = new File("src/main/resources/img/demo.jpg");

ImageInputStream iis = ImageIO.createImageInputStream(file);

reader.setInput(iis);

// 读取图片的参数（ImageReadParam支持读取一个文件有多个图片）

ImageReadParam param = reader.getDefaultReadParam();

param.setSourceSubsampling(3, 3, 0, 0);//为了读取每3个像素中的一个，产生一个原图片1/9大小的图片

BufferedImage bi1 = reader.read(0, param);

ImageIO.write(bi1, "jpg", new File("D://22.png"));

iis.close();

}

## 将多个图片写到一个图片里

不好用（很多格式不支持）

public static void main(String[] args) throws IOException {

// 获得一个图片读取器

Iterator<ImageReader> iterator = ImageIO.getImageReadersByFormatName("jpg");

ImageReader reader = (ImageReader)iterator.next();

// 将图片装换为图片的输入流

File file = new File("src/main/resources/img/demo.jpg");

ImageInputStream iis = ImageIO.createImageInputStream(file);

reader.setInput(iis);

// 读取图片的参数（ImageReadParam支持读取一个文件有多个图片）

ImageReadParam param = reader.getDefaultReadParam();

param.setSourceSubsampling(3, 3, 0, 0);//为了读取每3个像素中的一个，产生一个原图片1/9大小的图片

BufferedImage bi1 = reader.read(0, param);//缩略图

BufferedImage bi0 = ImageIO.read(file);//原图片

// 获得一个图片写入器

Iterator<ImageWriter> writers = ImageIO.getImageWritersByFormatName("gif");

ImageWriter writer = writers.next();

// 创建一个用于生成写入后文件的图片

// 一旦获取了一个ImageWriter对象，必须给它设置一个输出源ImageOutputStream

File file2 = new File("D://多图片合并.gif");

ImageOutputStream ios = ImageIO.createImageOutputStream(file2);

writer.setOutput(ios);

IIOImage iioImage0 = new IIOImage(bi0, null, null);

IIOImage iioImage1 = new IIOImage(bi1, null, null);

// writer.writeInsert(1, iioImage1, null);

writer.prepareWriteSequence(null);

writer.writeToSequence(iioImage0, null);

writer.writeToSequence(iioImage1, null);

}

## 画一个箭头

public static void main(String[] args) throws IOException {

BufferedImage sourceImage = new BufferedImage(200, 200, BufferedImage.TYPE\_4BYTE\_ABGR);

// 绘制箭头

Graphics2D g = (Graphics2D) sourceImage.getGraphics();

// 箭头各个点的位置

int[] xPoints = { 9, 11, 11, 18, 10, 2, 9 };

int[] yPoints = { 2, 2, 14, 10, 18, 10, 14 };

g.setColor(Color.BLACK);

int len = xPoints.length;

g.drawPolygon(xPoints, yPoints, len);

g.fillPolygon(xPoints, yPoints, len);

ImageIO.write(sourceImage, "PNG", new File("D://7.png"));

}

## 图片内容旋转

public static void main(String[] args) throws IOException {

// 读取原图

BufferedImage sourceImage = ImageIO.read(new File("src/main/resources/img/demo3.png"));

Graphics2D g = (Graphics2D) sourceImage.getGraphics();

// 获取原图宽高

int w = sourceImage.getWidth();

int h = sourceImage.getHeight();

//存放旋转后的图像

BufferedImage rotatedImage = new BufferedImage(w, h, sourceImage.getType());

// 使用矩阵旋转

AffineTransform at = new AffineTransform();

// 将角度转为弧度，顺时针旋转30度

double rad = Math.toRadians(30);

at.rotate(rad, w / 2, h / 2);// 旋转图象

at.translate(0, 0);// 原点坐标

AffineTransformOp op = new AffineTransformOp(at, AffineTransformOp.TYPE\_BICUBIC);

op.filter(sourceImage, rotatedImage);

// 最终出图

BufferedImage finalImg = new BufferedImage(w, h, sourceImage.getType());

g = finalImg.createGraphics();

g.fillRect(0, 0, w, h);

g.drawImage(rotatedImage, 0, 0, w, h, null);

ImageIO.write(rotatedImage, "PNG", new File("D://8.png"));

}

## 图片写字

比较实用

public static void main(String[] args) throws IOException {

// 读取原图

BufferedImage template = ImageIO.read(new File("src/main/resources/img/timg.jpg"));

Graphics2D g = (Graphics2D) template.getGraphics();

g.setFont(new Font("", Font.PLAIN, 30));//设置笔刷白色（这个影响图中所有的颜色）

g.setColor(Color.BLACK);//设置字的颜色

g.drawString("你好，XXX", 100, 170);//每行字高度30，行距20，即行高50

g.drawString("这是一封信", 160, 220);

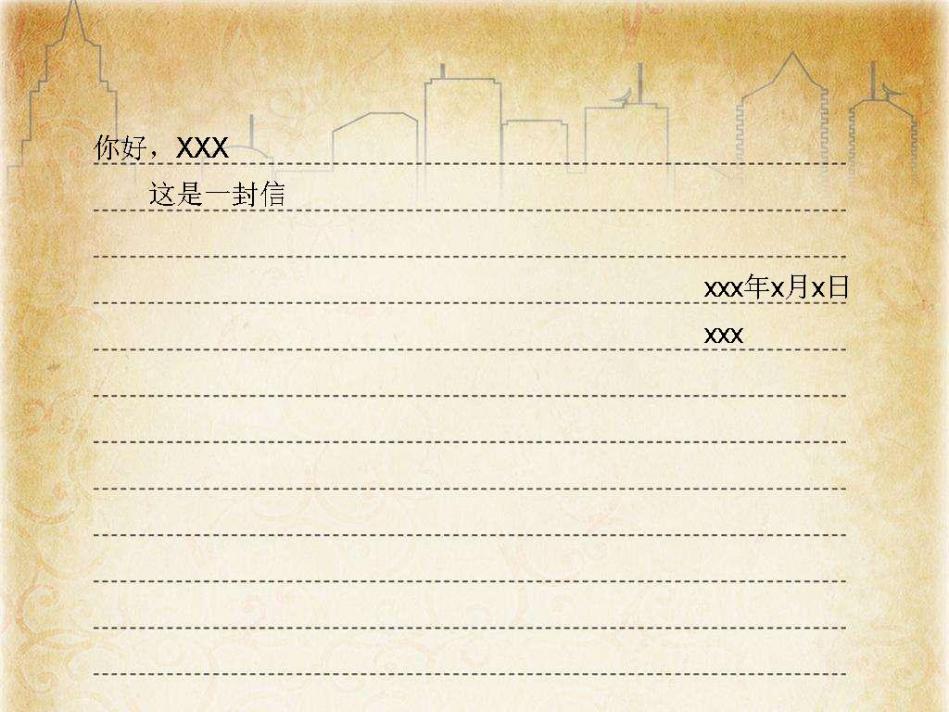
g.drawString("xxx年x月x日", 760, 320);

g.drawString("xxx", 760, 370);

ImageIO.write(template, "jpg", new File("D://9.png"));

}

效果如下



## 图片与文件结合

public static void main(String[] args) throws IOException {

// 创建一张图,使用TYPE\_4BYTE\_ABGR 默认背景色是透明的

BufferedImage template = new BufferedImage(1000, 1000, BufferedImage.TYPE\_4BYTE\_ABGR);

Graphics2D g = (Graphics2D) template.getGraphics();

g.setColor(Color.GRAY);//设置笔刷白色，注释后背景色是透明的

g.fillRect(0,0,1000,1000);//填充整个屏幕，相当于背景色，注释后背景色是透明的

g.setFont(new Font("", Font.PLAIN, 30));//设置字体

g.setColor(Color.BLACK);//设置笔刷黑色

g.drawLine(100, 100, 900, 100);//画一条线

g.drawString("你好，XXX", 100, 90);//每行字高度30，行距20，即行高50

g.drawString("这是一封信", 160, 140);

g.drawString("xxx年x月x日", 760, 240);

g.drawString("xxx", 760, 290);

// 向图片中添加添加一张图片

BufferedImage tt = ImageIO.read(new File("src/main/resources/img/tt.jpg"));

ImageIcon icon = new ImageIcon(tt);

// 300 和 500 是新图片的位置

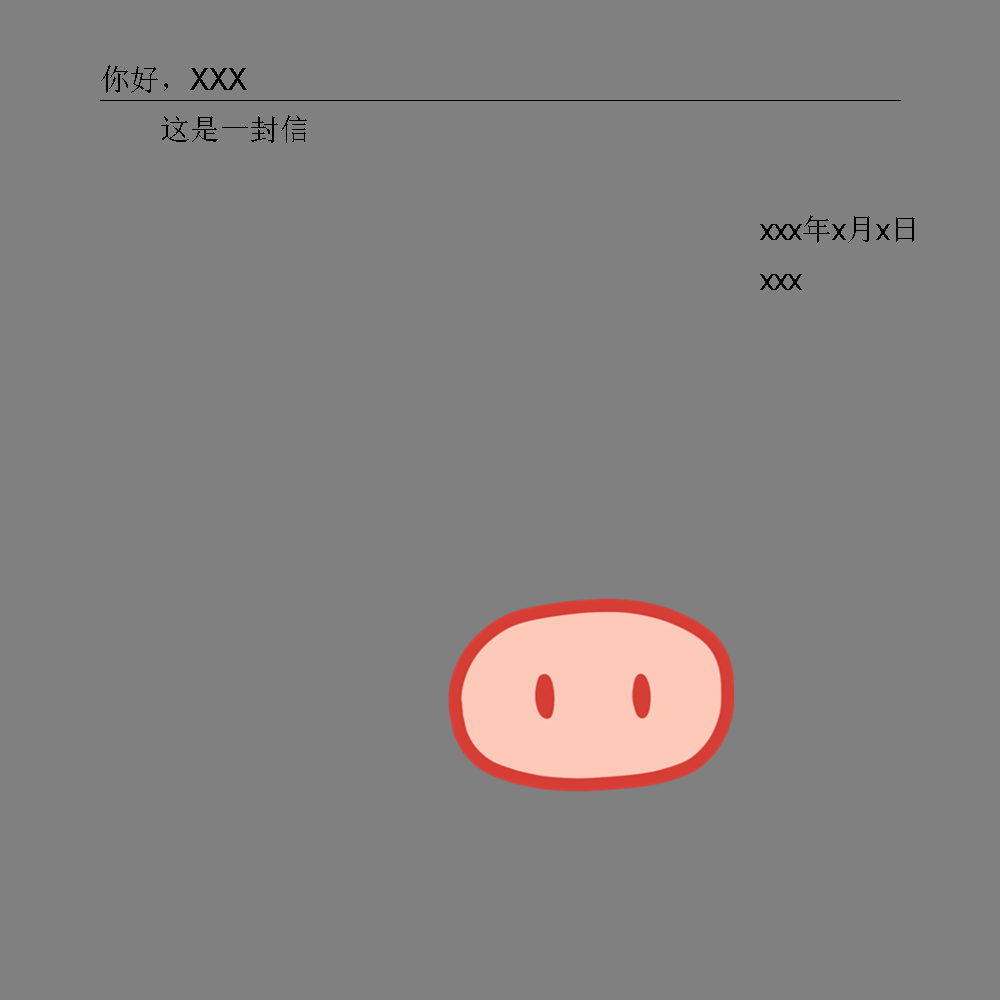
g.drawImage(icon.getImage(), 300, 500, icon.getImageObserver());

g.dispose();

ImageIO.write(template, "png", new File("D://9.1.png"));

}

效果如下



## HTML图片

将图片直接导入HTML中

img src支持的格式，这种方式多用于将一些小的图片直接嵌入到页面中。

可以用于发送邮件中，将邮件已HTML的格式发送，图片直接嵌入到页面中。

data:,文本数据

data:text/plain,文本数据

data:text/html,HTML代码

data:text/html;base64,base64编码的HTML代码

data:text/css,CSS代码

data:text/css;base64,base64编码的CSS代码

data:text/javascript,Javascript代码

data:text/javascript;base64,base64编码的Javascript代码

data:image/gif;base64,base64编码的gif图片数据

data:image/png;base64,base64编码的png图片数据

data:image/jpeg;base64,base64编码的jpeg图片数据

data:image/x-icon;base64,base64编码的icon图片数据

public static void main(String[] args) throws IOException {

//用于保存图片

ByteArrayOutputStream stream = new ByteArrayOutputStream();

//读取图片

BufferedImage read = ImageIO.read(new File("src/main/resources/img/demo.jpg"));

//将图片写到输出流中

ImageIO.write(read, "png", stream);

//获得图片的字节

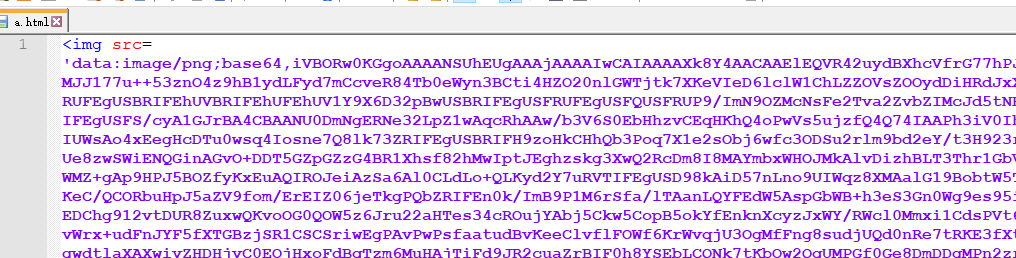
byte[] encode = Base64.getEncoder().encode(stream.toByteArray());

//将内容复制到HTML格式的文本中，即可在浏览器中显示出图片

System.out.println("<img src='data:image/png;base64," + new String(encode) + "'/>");

}

生成一个很大的文本。浏览器打开就是图片



# Thumbnailator

Thumbnailator 是一个优秀的图片处理的Google开源Java类库。处理效果远比Java API的好。从API提供现有的图像文件和图像对象的类中简化了处理过程，两三行代码就能够从现有图片生成处理后的图片，且允许微调图片的生成方式，同时保持了需要写入的最低限度的代码量。还支持对一个目录的所有图片进行批量处理操作。

支持的处理操作：图片缩放，区域裁剪，水印，旋转，保持比例。

public static void main(String[] args) throws IOException {

//1. 指定大小进行缩放 size(宽度, 高度)

/\*

\* 若图片横比200小，高比300小，不变

\* 若图片横比200小，高比300大，高缩小到300，图片比例不变

\* 若图片横比200大，高比300小，横缩小到200，图片比例不变

\* 若图片横比200大，高比300大，图片按比例缩小，横为200或高为300

\*/

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(200, 300)

.toFile("d:/a380\_200x300.jpg");

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(2560, 2048)

.toFile("d:/a380\_2560x2048.jpg");

//2.按照比例进行缩放 scale(比例)

Thumbnails.of("src/main/resources/img/demo.jpg")

.scale(0.25f)

.toFile("d:/a380\_25%.jpg");

Thumbnails.of("src/main/resources/img/demo.jpg")

.scale(1.10f)

.toFile("d:/a380\_110%.jpg");

//3、不按照比例，指定大小进行缩放

//keepAspectRatio(false)默认是按照比例缩放的

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(200,200)

.keepAspectRatio(false)

.toFile("d:/a380\_200x200.jpg");

//4、旋转

//rotate(角度),正数：顺时针负数：逆时针

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.rotate(90)

.toFile("d:/a380\_rotate+90.jpg");

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.rotate(-90)

.toFile("d:/a380\_rotate-90.jpg");

//5、水印

//watermark(位置，水印图，透明度)

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.watermark(Positions.BOTTOM\_RIGHT,ImageIO.read(new File("src/main/resources/img/demo.jpg")), 0.5f)

.outputQuality(0.8f)

.toFile("d:/a380\_watermark\_bottom\_right.jpg");

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.watermark(Positions.CENTER,ImageIO.read(new File("src/main/resources/img/demo.jpg")),0.5f)

.outputQuality(0.8f)

.toFile("d:/a380\_watermark\_center.jpg");

//6、裁剪

//sourceRegion()

//图片中心400\*400的区域

Thumbnails.of("src/main/resources/img/demo.jpg")

.sourceRegion(Positions.CENTER,400,400)

.size(200,200)

.keepAspectRatio(false)

.toFile("d:/a380\_region\_center.jpg");

//图片右下400\*400的区域

Thumbnails.of("src/main/resources/img/demo.jpg")

.sourceRegion(Positions.BOTTOM\_RIGHT,400,400)

.size(200,200)

.keepAspectRatio(false)

.toFile("d:/a380\_region\_bootom\_right.jpg");

//指定坐标

Thumbnails.of("src/main/resources/img/demo.jpg")

.sourceRegion(600,500,400,400)

.size(200,200)

.keepAspectRatio(false)

.toFile("d:/a380\_region\_coord.jpg");

//7、转化图像格式

//outputFormat(图像格式)

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.outputFormat("png")

.toFile("d:/a380\_1280x1024.png");

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.outputFormat("gif")

.toFile("d:/a380\_1280x1024.gif");

//8、输出到OutputStream

//toOutputStream(流对象)

OutputStream os=new FileOutputStream("d:/a380\_1280x1024\_OutputStream.png");

Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.toOutputStream(os);

//9、输出到BufferedImage

//asBufferedImage()返回BufferedImage

BufferedImage thumbnail=Thumbnails.of("src/main/resources/img/demo.jpg")

.size(1280,1024)

.asBufferedImage();

ImageIO.write(thumbnail,"jpg",new File("d:/a380\_1280x1024\_BufferedImage.jpg"));

}

# ImageMagick

ImageMagick是一个免费的创建、编辑、合成图片的软件，可以实现图片切割、颜色替换、图片缩略图、图片水印等各种效果。ImageMagick是免费开源软件，支持大多数操作系统，包括Linux、Windows操作系统。ImageMagick的大多数功能的使用都来源于命令行工具。

## 安装

<http://www.imagemagick.org/script/download.php>

需要安装客户端

## 基本命令

ImageMagick图片处理命令的格式如下：

imagick deal\_type origin\_pic\_path deal\_param dest\_pic\_path

deal\_type：图片处理类型。ImageMagick提供了几十种图片处理类型，包括：decorate（边框）、convert（转换）、montage（拼接）、animation（动画）等，你可以在Features and Capabilities查看更多图片处理类型。

origin\_pic\_path：源图片路径。

deal\_param：处理参数。不同的图片处理类型有不同的处理参数。

dest\_pic\_path：目标图片路径。

运行下面的命令将图片顺时针旋转90度：

magick doraemon.jpg -rotate 90 doraemon\_rotate.jpg

IM4Java的作用是将Java代码转成对ImageMagick的命令行调用，因此IM4Java API与ImageMagick的命令有一一对应的关系。

ImageMagick的图片处理类型（deal\_type）对应xxxCmd，如：convert处理类型对应ConvertCmd，montage处理类型对应MontageCmd。

ImageMagick的图片处理参数（deal\_param）对应IMOperation方法，如：-rotate 90参数对应operation.rotate(90d)，-resize 900x600参数对应operation.resize(900,600)。

## 调用API

public static void main(String[] args) throws IOException, InterruptedException, IM4JavaException {

IMOperation operation = new IMOperation();

operation.addImage("src/main/resources/img/demo.jpg");

operation.rotate(90d);

operation.addImage("src/main/resources/img/demo2.jpg");

ConvertCmd cmd = new ConvertCmd();

cmd.setSearchPath("d:/aaaa.jpg"); //Windows需要设置，Linux不需要

cmd.run(operation);

}