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。

2、JAI中的Image I/O Tools，支持更多图片类型，例如JPEG-LS,JPEG2000,和TIFF。

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。

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JIM，JMagic等，但JDK本身能够读写图片，就用JDK的，开发和部署方便，不需要JAI这么高级庞大的东西，用Java Image I/O API即可。

**Java Image I/O API**

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。

下面这段代码可以展示，JDK内置支持的图片格式。  
import javax.imageio.\*;  
import java.util.Arrays;

public class HelloWorld {  
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));  
}  
}

主页上有一个文档，[Java Image I/O API Guide](http://java.sun.com/javase/6/docs/technotes/guides/imageio/spec/title.fm.html" \t "https://blog.csdn.net/fanhenghui/article/details/_blank)，很[通俗易懂](https://www.baidu.com/s?wd=%E9%80%9A%E4%BF%97%E6%98%93%E6%87%82&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/fanhenghui/article/details/_blank)，可以让你快速上手。以下  
内容主要来自这个文档的第3章。

**第3章 编写图像I/O程序  
3.1 读写图片**  
javax.imageio.ImageIO类提供了一组静态方法进行最简单的图像I/O操作。  
读取一个标准格式(GIF, PNG, or JPEG)的图片很简单：  
File f = new File(“c:\images\myimage.gif”);  
BufferedImage bi = ImageIO.read(f);

Java Image I/O API 会自动探测图片的格式并调用对应的插件进行解码，当安装了一个新  
插件，新的格式会被自动理解，程序代码不需要改变。

写图片同样简单：  
BufferedImage bi;  
File f = new File(“c:\images\myimage.png”);  
ImageIO.write(im, “png”, f);

**3.2 更进一步**  
上一节谈到的方法对于简单程序已经足够了。不过，Java Image I/O API 提供了为编写复  
杂程序的能力。为了利用API的高级特性，应用程序应当直接使用类ImageReader 和  
ImageWriter。

**3.3 ImageReader 类**  
与其用ImageIO类来进行所有的解码操作，不如用ImageIO类去得到一个ImageReader对象，  
再用这个对象去进行读操作：  
Iterator readers = ImageIO.getImageReadersByFormatName(“gif”);  
ImageReader reader = (ImageReader)readers.next();

ImageReader对象也可以基于文件内容、文件后缀或MIME类型获得。这个用于查找和初始  
化ImageReader对象的机制用到了javax.imageio.spi.ImageReaderSpi类，它可以在不用初  
始化插件的情况下获得插件的信息。”service provider interfaces” (SPIs)将会在下一  
章详细讨论。一旦获得了一个ImageReader对象，必须给它是指一个输入源。大部分  
ImageReader对象可以从ImageInputStream类输入源读取数据，ImageInputStream是Image  
I/O API定义的专用输入源。

获得一个ImageInputStream 是简单的。给定一个File或InputStream，一个  
ImageInputStream对象可以通过调用如下函数产生：  
Object source; // File or InputStream  
ImageInputStream iis = ImageIO.createImageInputStream(source);

一旦有了输入源，可以把它与一个ImageReader对象关联起来：  
reader.setInput(iis, true);

如果输入源文件包含多张图片，而程序不保证按顺序读取时，第二个参数应该设置为  
false。对于那些只允许存储一张图片的文件格式，永远传递true是合理的。

当ImageReader对象有了输入源后，我们就可以获取图片信息而不用把整张图片数据都读入  
内存。例如，调用reader.getImageWidth(0)可以让我们获得文件中第一张图片的宽度。一  
个好的插件[会试](https://www.baidu.com/s?wd=%E4%BC%9A%E8%AF%95&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/fanhenghui/article/details/_blank)图解码文件的必要部分，去获得图片的宽度，而不用读取任何一个像素。

为读取图片，可以调用reader.read(imageIndex), imageIndex是文件（当包含多张图片时）  
中图片的索引。这与上一节调用ImageIO.read()产生的结果相同。

**3.3.1 ImageReadParam**  
如果需要更多的控制，可以向read()方法传递一个ImageReadParam类型的参数。一个  
ImageReadParam对象可以让程序更好的利用内存。它不仅允许指定一个感兴趣的区域，还  
可以指定一个抽样因子，用于向下采样。

例如，为了只解码图片的左上角的1/4，程序可以先获取一个合适的ImageReadParam对象：  
ImageReadParam param = reader.getDefaultReadParam();

接下来，指定图片区域：  
import java.awt.Rectangle;  
int imageIndex = 0;  
int half\_width = reader.getImageWidth(imageIndex)/2;  
int half\_height = reader.getImageHeight(imageIndex)/2;  
Rectangle rect = new Rectangle(0, 0, half\_width, half\_height);  
param.setSourceRegion(rect);

最后，读取图片：  
BufferedImage bi = reader.read(imageIndex, param);

结果是一张新图片，宽和[高都](https://www.baidu.com/s?wd=%E9%AB%98%E9%83%BD&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/fanhenghui/article/details/_blank)只有原图片的一半。

另一个例子，为了读取每三个像素中的一个，产生一个原图片1/9大小的图片，可以用  
ImageReadParam指定抽样因子：  
param = reader.getDefaultImageParam();  
param.setSourceSubsampling(3, 3, 0, 0);  
BufferedImage bi3 = reader.read(0, param);

**3.3.2 IIOParamController**  
插件有时会提供一个IIOParamController类，这是可选的。略。

**3.3.3 读多图片文件**  
ImageReader 中所有与图片打交道的方法都有一个imageIndex 参数，这个参数用于读取多  
图片文件中的一张。

ImageReader.getNumImages()返回多图片文件中的图片个数。这个方法有一个boolean参数，  
allowSearch。有的图片格式，典型的GIF，没有提供任何获取文件中的图片个数方法，除  
非读取整个进行解析。这样代价很高，因此设置allowSearch为false可以让方法直接返回  
-1，而不是实际的图片个数。如果此参数是true，则该方法总会返回文件中实际的图片个  
数。

即使在不知道文件中图片个数的情况下，仍可以调用read(imageIndex); 如果索引值过大，  
该方法会抛出IndexOutOfBoundsException异常。因此，程序可以递增索引去获取图片，  
直到异常。

**3.3.4 读缩略图**  
有的图片格式允许一个（或多个）小的预览图，与主图片一起存储在文件中。这些  
“缩略图”对于快速识别图片很有用，不用解码整个图片。

程序可以调用如下代码，探测一张图片有多少张缩略图：  
reader.getNumThumbnails(imageIndex);

如果存在缩略图，可以调用如下代码获取：  
int thumbailIndex = 0;  
BufferedImage bi;  
bi = reader.readThumbnail(imageIndex, thumbnailIndex);

**3.4 ImageWriter 类**  
就像我们可以用ImageIO 的一个方法获取某种图片格式的ImageReader对象一样，我们也可  
以获取ImageWriter对象：  
Iterator writers = ImageIO.getImageWritersByFormatName(“png”);  
ImageWriter writer = (ImageWriter)writers.next();

一旦获取了一个ImageWriter对象，必须给它设置一个输出源ImageOutputStream。  
File f = new File(“c:\images\myimage.png”);  
ImageOutputStream **[iOS](http://lib.csdn.net/base/ios" \o "iOS知识库" \t "https://blog.csdn.net/fanhenghui/article/details/_blank)** = ImageIO.createImageOutputStream(f);  
writer.setOutput(ios);

最后，可以把图片写入到输出源：  
BufferedImage bi;  
writer.write(bi);

**3.4.1 写多图片文件**  
IIOImage类用于存储图片，缩略图或元信息的引用。下一节将讨论Metadata，目前，我们  
简单地给Metadata相关参数传递null。  
ImageWriter 类有一个方法write()，用于从IIOImage创建一个新文件，还有一个方法  
writeInsert()，用于向一个已存在文件添加一个IIOImage对象。通过调用这两者，可以创  
建一个多图片文件：  
BufferedImage first\_bi, second\_bi;  
IIOImage first\_IIOImage = new IIOImage(first\_bi, null, null);  
IIOImage second\_IIOImage = new IIOImage(second\_bi, null, null);  
writer.write(null, first\_IIOImage, null);  
if (writer.canInsertImage(1)) {  
writer.writeInsert(1, second\_IIOImage, null);  
} else {  
System.err.println(“Writer can’t append a second image!”);  
}

**3.5 处理 Metadata**  
所有与像素无关的信息，都属于在Metadata。javax.imageio.metadata 包含了用于访问  
Metadata的类和接口。

Image I/O API 将stream metadata 和image metadata区别对待。stream metadata与一个  
文件中存储了多张图片有关，image metadata只与单个图片有关。如果一个文件只包含一张  
图片，那么就只存在image metadata。

可以通过调用ImageReader.getStreamMetadata 和 getImageMetadata(int imageIndex)来  
获取metadata。这些方法会返回一个实现了IIOMetadata接口的对象，该对象会被向上转化  
为ImageReader类型，

**3.6 编码转换**  
略

**3.7 事件监听**  
略