Java 核心技术(进阶)

第四章 高级文件处理

第四节 图形图像简介及解析

华东师范大学 陈良育

大纲

- 图形图像基本概念
- Java图形图像关键类
- 典型应用
  - 图像文件基本读写
  - 验证码生成
  - 统计图生成

图形图像基础概念

- 图形：Graph
  - 矢量图，根据几何特性来画的，比如点、直线、弧线等
- 图像：Image
  - 由像素点组成
  - 格式：jpg, png, bmp, svg, wmf, gif, tiff 等
  - 颜色：RGB(Red, Green, Blue)

Java图形图像关键类

- 图形： Graph
  - java.awt 包
  - Java 2D库: Graphics2D, Line2D, Rectangle2D, Ellipse2D,Arc2D
  - Color, Stroke
- 图像： Image
  - javax.imageio包
  - ImageIO, BufferedImage, ImageReader, ImageWriter

Java图像关键类描述

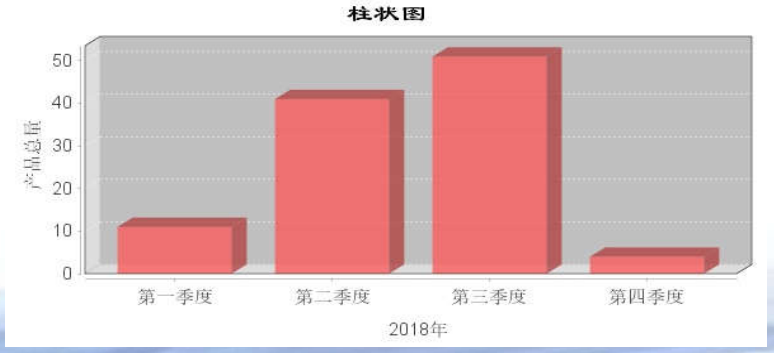
- Java原生支持jpg, png, bmp, wbmp, gif
- javax.imageio.ImageIO
  - 自动封装多种ImageReader和ImageWriter，读写图像文件
  - read 读取图片 write 写图片
- java.awt.image.BufferedImage,图像在内存中的表示类
  - getHeight 获取高度
  - getWidth 获取宽度
- 图像文件读写/截取/合并

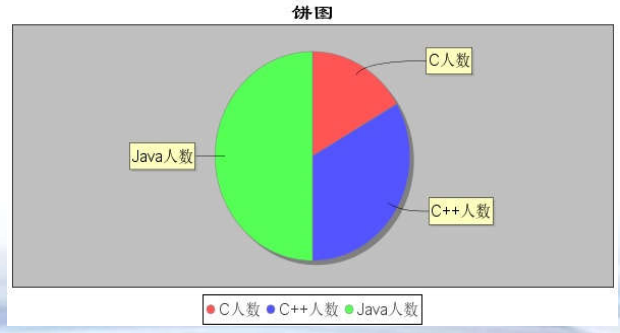
验证码生成

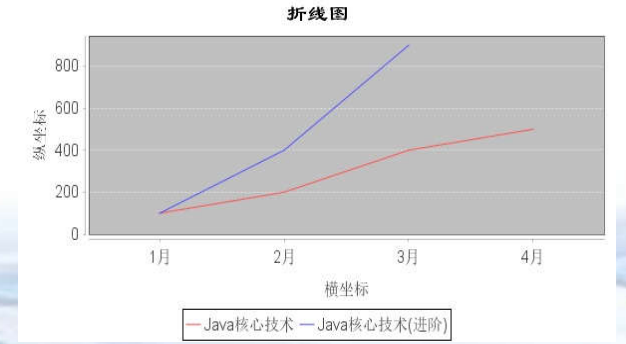
- 验证码，一个图片文件
  - 外框
  - 底色
  - 干扰线
    - 随机产生一些直线
  - 字母
    - 字母选择，不要0，o，1，I，L
    - 字母颜色(RGB)
    - 字母位置

统计图生成

- 统计图
  - 柱状图/饼图/折线图
  - Java原生的Graphics 2D可以画，比较繁琐
  - 基于jFreeChart(www.jfree.org/jfreechart)可以快速实现统计图生成
    - 设定数据集
    - 调用ChartFactory生成图形







Java图形图像编程

- 总结
  - Java的AWT包提供了一些基础的图形工具Graphics 2D
  - Javax的imageio包提供了基础的图像读写和剪辑
  - 借助第三方库jFreeChart完成统计类图
  - API很多，需要多查询、多练习

代码(1) ImageTest.java

```java
package basic;

import java.awt.Rectangle;

public class ImageTest {

    public static void main(String[] args) throws Exception {
        readAndWrite();
        readComparison();
        cropImage("c:/temp/ecnu.jpg", "c:/temp/shida.jpg", 750, 250, 700, 300, "jpg", "jpg");
        combineImagesHorizontally("c:/temp/ecnu.jpg","c:/temp/ecnu.jpg","jpg", "c:/temp/ecnu2.jpg");
        combineImagesVertically("c:/temp/ecnu.jpg","c:/temp/ecnu.jpg","jpg", "c:/temp/ecnu3.jpg");
    }

    public static void readAndWrite() throws Exception {
        BufferedImage image = ImageIO.read(new File("c:/temp/ecnu.jpg"));
        System.out.println("Height: " + image.getHeight()); // 高度像素
        System.out.println("Width: " + image.getWidth()); // 宽度像素
        ImageIO.write(image, "png", new File("c:/temp/ecnu.png"));
    }
```

代码(2) ImageTest.java


```
public static void readComparison() throws Exception {
    System.out.println("============加载速度测试==============");

    // ImageIO需要测试图片的类型，加载合适的ImageReader来读取图片，耗时更长
    long startTime = System.nanoTime();
    BufferedImage image = ImageIO.read(new File("c:/temp/ecnu.jpg"));
    System.out.println("Height: " + image.getHeight()); // 高度像素
    System.out.println("Width: " + image.getWidth()); // 宽度像素
    long endTime = System.nanoTime();
    System.out.println((endTime - startTime) / 1000000.0 + "毫秒");

    // 指定用jpg Reader来加载，速度会加快
    startTime = System.nanoTime();
    Iterator<ImageReader> readers = ImageIO.getImageReadersByFormatName("jpg");
    ImageReader reader = (ImageReader) readers.next();
    System.out.println(reader.getClass().getName());
    ImageInputStream iis = ImageIO.createImageInputStream(new File("c:/temp/ecnu.jpg"));
    reader.setInput(iis, true);
    System.out.println("Height:" + reader.getHeight(0));
    System.out.println("Width:" + reader.getWidth(0));
    endTime = System.nanoTime();
    System.out.println((endTime - startTime) / 1000000.0 + "毫秒");
}
```

代码(3) ImageTest.java

```
public static void cropImage(String fromPath, String toPath, int x, int y, int width, int height, String readImageFormat,
        String writeImageFormat) throws Exception {
    FileInputStream fis = null;
    ImageInputStream iis = null;
    try {
        // 读取原始图片文件
        fis = new FileInputStream(fromPath);
        Iterator<ImageReader> it = ImageIO.getImageReadersByFormatName(readImageFormat);
        ImageReader reader = it.next();
        iis = ImageIO.createImageInputStream(fis);
        reader.setInput(iis, true);

        // 定义一个矩形 并放入切割参数中
        ImageReadParam param = reader.getDefaultReadParam();
        Rectangle rect = new Rectangle(x, y, width, height);
        param.setSourceRegion(rect);

        //从源文件读取一个矩形大小的图像
        BufferedImage bi = reader.read(0, param);

        //写入到目标文件
        ImageIO.write(bi, writeImageFormat, new File(toPath));
    } finally {
        fis.close();
        iis.close();
    }
}
```

代码(4) ImageTest.java


```
public static void combineImagesHorizontally(String firstPath, String secondPath,String imageFormat, String toPath){
    try {
        //读取第一张图片
        File  first  =  new  File(firstPath);
        BufferedImage  imageOne = ImageIO.read(first);
        int  width1  =  imageOne.getWidth();//图片宽度
        int  height1  =  imageOne.getHeight();//图片高度
        //从第一张图片中读取RGB
        int[]  firstRGB  =  new  int[width1*height1];
        firstRGB  =  imageOne.getRGB(0,0,width1,height1,firstRGB,0,width1);

        //对第二张图片做同样的处理
        File  second  =  new  File(secondPath);
        BufferedImage  imageTwo  =  ImageIO.read(second);
        int width2 = imageTwo.getWidth();
        int height2 = imageTwo.getHeight();
        int[]   secondRGB  =  new  int[width2*height2];
        secondRGB  =  imageTwo.getRGB(0,0,width2,height2,secondRGB,0,width2);
```

代码(5) ImageTest.java


```
        //生成新图片
        int height3 = (height1>height2)?height1:height2; //挑选高度大的，作为目标文件的高度
        int width3  = width1 + width2;                   //宽度，两张图片相加
        BufferedImage  imageNew  =  new  BufferedImage(width3,height3,BufferedImage.TYPE_INT_RGB);

        //设置左半部分的RGB 从(0,0) 开始
        imageNew.setRGB(0,0,width1,height1,firstRGB,0,width1);
        //设置右半部分的RGB 从(width1, 0) 开始
        imageNew.setRGB(width1,0,width2,height2,secondRGB,0,width2);

        //保存图片
        ImageIO.write(imageNew,  imageFormat,  new  File(toPath));
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

代码(6) ImageTest.java

```
public static void combineImagesVertically(String firstPath, String secondPath,String imageFormat, String toPath){
    try {
        //读取第一张图片
        File  first  =  new  File(firstPath);
        BufferedImage  imageOne = ImageIO.read(first);
        int  width1  =  imageOne.getWidth();//图片宽度
        int  height1  =  imageOne.getHeight();//图片高度
        //从图片中读取RGB
        int[]  firstRGB  =  new  int[width1*height1];
        firstRGB  =  imageOne.getRGB(0,0,width1,height1,firstRGB,0,width1);

        //对第二张图片做相同的处理
        File  second  =  new  File(secondPath);
        BufferedImage  imageTwo  =  ImageIO.read(second);
        int width2 = imageTwo.getWidth();
        int height2 = imageTwo.getHeight();
        int[]   secondRGB  =  new  int[width2*height2];
        secondRGB  =  imageTwo.getRGB(0,0,width2,height2,secondRGB,0,width2);
```

代码(7) ImageTest.java


```
            //生成新图片
            int width3 = (width1>width2)?width1:width2; //挑选宽度大的，作为目标文件的宽度
            int height3 = height1+height2;              //高度，两张图片相加
            BufferedImage  imageNew  =  new  BufferedImage(width3,height3,BufferedImage.TYPE_INT_RGB);
            //设置上半部分的RGB 从(0,0) 开始
            imageNew.setRGB(0,0,width1,height1,firstRGB,0,width1);
            //设置下半部分的RGB 从(0, height1) 开始
            imageNew.setRGB(0,height1,width2,height2,secondRGB,0,width2);

            //保存图片
            ImageIO.write(imageNew, imageFormat, new File(toPath));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

代码(8) ValidateCodeTest.java


```
package code;

import java.awt.Color;



public class ValidateCodeTest {

    //没有1 I L 0 o
    static char[] codeSequence = { 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'J', 'K', 'M', 'N', 'P', 'Q', 'R', 'S', 'T',
            'U', 'V', 'W', 'X', 'Y', 'Z', '2', '3', '4', '5', '6', '7', '8', '9' };
    static int charNum = codeSequence.length;

    public static void main(String[] a) throws IOException
    {
        generateCode("c:/temp/code.jpg");
    }
```

代码(9) ValidateCodeTest.java


```
public static void generateCode(String filePath) throws IOException {
    // 首先定义验证码图片框
    int width = 80; // 验证码图片的宽度
    int height = 32; // 验证码图片的高度
    BufferedImage buffImg = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);


    //定义图片上的图形和干扰线
    Graphics2D gd = buffImg.createGraphics();
    gd.setColor(Color.LIGHT_GRAY);    // 将图像填充为浅灰色
    gd.fillRect(0, 0, width, height);
    gd.setColor(Color.BLACK);          // 画边框。
    gd.drawRect(0, 0, width - 1, height - 1);
    // 随机产生16条灰色干扰线，使图像中的认证码不易识别
    gd.setColor(Color.gray);
    // 创建一个随机数生成器类，用于随机产生干扰线
    Random random = new Random();
    for (int i = 0; i < 16; i++) {
        int x = random.nextInt(width);
        int y = random.nextInt(height);
        int xl = random.nextInt(12);
        int yl = random.nextInt(12);
        gd.drawLine(x, y, x + xl, y + yl);
    }
```

代码(10) ValidateCodeTest.java


```
//计算字的位置坐标
int codeCount = 4; // 字符个数
int fontHeight; // 字体高度
int codeX; // 第一个字符的x坐标，因为后面的字符坐标依次递增，所以它们的x轴值是codeX的倍数
int codeY; // 验证字符的y坐标，因为并排所以值一样
// width-4 除去左右多余的位置，使验证码更加集中显示，减得越多越集中。
// codeCount+1 //等比分配显示的宽度，包括左右两边的空格
codeX = (width - 4) / (codeCount + 1); //第一个字母的起始位置
fontHeight = height - 10;  // height - 10 高度中间区域显示验证码
codeY = height - 7;
```

代码(11) ValidateCodeTest.java


```
        // 创建字体，字体的大小应该根据图片的高度来定。
        Font font = new Font("Fixedsys", Font.PLAIN, fontHeight);
        gd.setFont(font);

        // 随机产生codeCount数字的验证码。
        for (int i = 0; i < codeCount; i++) {
            // 每次随机拿一个字母，赋予随机的颜色
            String strRand = String.valueOf(codeSequence[random.nextInt(charNum)]);
            int red = random.nextInt(255);
            int green = random.nextInt(255);
            int blue = random.nextInt(255);
            gd.setColor(new Color(red,green,blue));
            //把字放到图片上!!!
            gd.drawString(strRand, (i + 1) * codeX, codeY);
        }

        ImageIO.write(buffImg, "jpg", new File(filePath));
    }
}
```

代码(12) JFreeChartTest.java


```
package charts;

import java.awt.Font;

public class JFreeChartTest {

    public static void main(String[] args) {
        writeBar("c:/temp/bar.jpg"); // 柱状图
        writePie("c:/temp/pie.jpg"); // 饼图
        writeLine("c:/temp/line.jpg");// 折线图
    }

    public static StandardChartTheme getChineseTheme()
    {
        StandardChartTheme chineseTheme = new StandardChartTheme("CN");
        chineseTheme.setExtraLargeFont(new Font("隶书", Font.BOLD, 20));
        chineseTheme.setRegularFont(new Font("宋书", Font.PLAIN, 15));
        chineseTheme.setLargeFont(new Font("宋书", Font.PLAIN, 15));
        return chineseTheme;
    }
```

代码(13) JFreeChartTest.java


```
public static void writeBar(String fileName) {
    DefaultCategoryDataset dataset = new DefaultCategoryDataset();
    dataset.addValue(11, "", "第一季度");
    dataset.addValue(41, "", "第二季度");
    dataset.addValue(51, "", "第三季度");
    dataset.addValue(4, "", "第四季度");

    // PlotOrientation.HORIZONTAL横向 PlotOrientation.VERTICAL 竖向
    // 引入中文主题样式
    ChartFactory.setChartTheme(getChineseTheme());
    JFreeChart chart = ChartFactory.createBarChart3D("柱状图", "2018年", "产品总量", dataset, PlotOrientation.VERTICAL,
            false, false, false);

    try {
        ChartUtilities.saveChartAsJPEG(new File(fileName), chart, 600, 300);
    } catch (IOException e) {
        e.printStackTrace();
    }

}
```

代码(14) JFreeChartTest.java


```
public static void writePie(String fileName) {
    DefaultPieDataset pds = new DefaultPieDataset();
    pds.setValue("C人数", 100);
    pds.setValue("C++人数", 200);
    pds.setValue("Java人数", 300);
    try {
        ChartFactory.setChartTheme(getChineseTheme());
        JFreeChart chart = ChartFactory.createPieChart("饼图", pds);

        ChartUtilities.saveChartAsJPEG(new File(fileName), chart, 600, 300);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

代码(15) JFreeChartTest.java


```
public static void writeLine(String fileName) {
    DefaultCategoryDataset lines = new DefaultCategoryDataset();
    //第一条线
    lines.addValue(100, "Java核心技术", "1月");
    lines.addValue(200, "Java核心技术", "2月");
    lines.addValue(400, "Java核心技术", "3月");
    lines.addValue(500, "Java核心技术", "4月");

    //第二条线
    lines.addValue(100, "Java核心技术(进阶)", "1月");
    lines.addValue(400, "Java核心技术(进阶)", "2月");
    lines.addValue(900, "Java核心技术(进阶)", "3月");
    try {
        ChartFactory.setChartTheme(getChineseTheme());
        JFreeChart chart = ChartFactory.createLineChart("折线图", "时间", "人数", lines);
        ChartUtilities.saveChartAsJPEG(new File(fileName), chart, 600, 300);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}
```

谢 谢！