import java.awt.Image;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import javax.imageio.ImageIO;

import com.sun.image.codec.jpeg.JPEGCodec;

import com.sun.image.codec.jpeg.JPEGEncodeParam;

import com.sun.image.codec.jpeg.JPEGImageEncoder;

public class ImageCompressUtil {

/\*\*

\* 直接指定压缩后的宽高：

\* (先保存原文件，再压缩、上传)

\* 壹拍项目中用于二维码压缩

\* @param oldFile 要进行压缩的文件全路径

\* @param width 压缩后的宽度

\* @param height 压缩后的高度

\* @param quality 压缩质量

\* @param smallIcon 文件名的小小后缀(注意，非文件后缀名称),入压缩文件名是yasuo.jpg,则压缩后文件名是yasuo(+smallIcon).jpg

\* @return 返回压缩后的文件的全路径

\*/

public static String zipImageFile(String oldFile, int width, int height,

float quality, String smallIcon) {

if (oldFile == null) {

return null;

}

String newImage = null;

try {

/\*\*对服务器上的临时文件进行处理 \*/

Image srcFile = ImageIO.read(new File(oldFile));

/\*\* 宽,高设定 \*/

BufferedImage tag = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

tag.getGraphics().drawImage(srcFile, 0, 0, width, height, null);

String filePrex = oldFile.substring(0, oldFile.indexOf('.'));

/\*\* 压缩后的文件名 \*/

newImage = filePrex + smallIcon + oldFile.substring(filePrex.length());

/\*\* 压缩之后临时存放位置 \*/

FileOutputStream out = new FileOutputStream(newImage);

JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(out);

JPEGEncodeParam jep = JPEGCodec.getDefaultJPEGEncodeParam(tag);

/\*\* 压缩质量 \*/

jep.setQuality(quality, true);

encoder.encode(tag, jep);

out.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return newImage;

}

/\*\*

\* 保存文件到服务器临时路径(用于文件上传)

\* @param fileName

\* @param is

\* @return 文件全路径

\*/

public static String writeFile(String fileName, InputStream is) {

if (fileName == null || fileName.trim().length() == 0) {

return null;

}

try {

/\*\* 首先保存到临时文件 \*/

FileOutputStream fos = new FileOutputStream(fileName);

byte[] readBytes = new byte[512];// 缓冲大小

int readed = 0;

while ((readed = is.read(readBytes)) > 0) {

fos.write(readBytes, 0, readed);

}

fos.close();

is.close();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

return fileName;

}

/\*\*

\* 等比例压缩算法：

\* 算法思想：根据压缩基数和压缩比来压缩原图，生产一张图片效果最接近原图的缩略图

\* @param srcURL 原图地址

\* @param deskURL 缩略图地址

\* @param comBase 压缩基数

\* @param scale 压缩限制(宽/高)比例 一般用1：

\* 当scale>=1,缩略图height=comBase,width按原图宽高比例;若scale<1,缩略图width=comBase,height按原图宽高比例

\* @throws Exception

\* @author shenbin

\* @createTime 2014-12-16

\* @lastModifyTime 2014-12-16

\*/

public static void saveMinPhoto(String srcURL, String deskURL, double comBase,

double scale) throws Exception {

File srcFile = new java.io.File(srcURL);

Image src = ImageIO.read(srcFile);

int srcHeight = src.getHeight(null);

int srcWidth = src.getWidth(null);

int deskHeight = 0;// 缩略图高

int deskWidth = 0;// 缩略图宽

double srcScale = (double) srcHeight / srcWidth;

/\*\*缩略图宽高算法\*/

if ((double) srcHeight > comBase || (double) srcWidth > comBase) {

if (srcScale >= scale || 1 / srcScale > scale) {

if (srcScale >= scale) {

deskHeight = (int) comBase;

deskWidth = srcWidth \* deskHeight / srcHeight;

} else {

deskWidth = (int) comBase;

deskHeight = srcHeight \* deskWidth / srcWidth;

}

} else {

if ((double) srcHeight > comBase) {

deskHeight = (int) comBase;

deskWidth = srcWidth \* deskHeight / srcHeight;

} else {

deskWidth = (int) comBase;

deskHeight = srcHeight \* deskWidth / srcWidth;

}

}

} else {

deskHeight = srcHeight;

deskWidth = srcWidth;

}

BufferedImage tag = new BufferedImage(deskWidth, deskHeight, BufferedImage.TYPE\_3BYTE\_BGR);

tag.getGraphics().drawImage(src, 0, 0, deskWidth, deskHeight, null); //绘制缩小后的图

FileOutputStream deskImage = new FileOutputStream(deskURL); //输出到文件流

JPEGImageEncoder encoder = JPEGCodec.createJPEGEncoder(deskImage);

encoder.encode(tag); //近JPEG编码

deskImage.close();

}

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

ImageCompressUtil.zipImageFile("f:/食尸鬼 - 藿香.jpg", 1280, 1280, 1f, "x2");

ImageCompressUtil.saveMinPhoto("f:/食尸鬼 - 藿香.jpg", "f:/11.jpg", 139, 0.9d);

}

}