Java サムネイルイメージの作成

[Java]

```
import java.awt.geom.AffineTransform;
import java.awt.image.AffineTransformOp;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import javax.imageio.ImageIO;

/**
 * サムネイル Jpeg を生成する
 * @author Yagi Hiroto
 */
public class ThumnailImageCreater {
  /**
   * InputSteram に関連付けられたイメージ
   *
   * @param in
   * @param out
   * @param formatName
   * @param rate
   * @throws IOException
   * @throws ImageServiceException
   */
  public void writeThumbNailImage(InputStream in,
          OutputStream out,
          String formatName,
          double rate) throws Exception {

    BufferedImage image = ImageIO.read(in);
    writeThumbNailImage(out, image, formatName, rate);
  }

  /**
   * @param in
   * @param out
   * @param rate
   * @throws IOException
   * @throws ImageServiceException
   */
  public void writeThumbNailImage(InputStream in,
          OutputStream out,
          double rate) throws Exception {

    writeThumbNailImage(in, out, "jpeg", rate);
  }

  /**
   * @param in
   * @param out
   * @param width
   * @param height
   * @throws IOException
   * @throws ImageServiceException
   */
  public void writeThumbNailImage(InputStream in,
          OutputStream out,
          int width,
          int height) throws Exception {

    writeThumbNailImage(in, out, "jpeg", width, height);
  }

  /**
   * @param in
   * @param out
   * @param formatName
   * @param width
   * @param height
   * @throws IOException
   * @throws ImageServiceException
   */
```

```
public void writeThumbNailImage(InputStream in,
        OutputStream out,
        String formatName,
        int width,
        int height) throws Exception {

  BufferedImage image = ImageIO.read(in);
  double rate = calcRatio(width, height, image);
  writeThumbNailImage(out, image, formatName, rate);
}

/**
 * 指定されたサイズに収まる縮小率を算出する
 *
 * @param width
 * @param height
 * @param image
 * @return
 */
private double calcRatio(int width, int height, BufferedImage image) {
  double wRate = (double) width  / (double) image.getWidth();
  double hRate = (double) height / (double) image.getHeight();
  return (wRate < hRate) ? wRate : hRate;
}

/**
 * @param out
 * @param image
 * @param formatName
 * @param rate
 * @throws IOException
 * @throws ImageServiceException
 */
public void writeThumbNailImage( OutputStream out,
               BufferedImage image,
               String formatName,
               double rate) throws Exception {

  BufferedImage shrinkImage = new BufferedImage(
        (int) (image.getWidth() * rate),
        (int) (image.getHeight() * rate),
        image.getType());

  AffineTransformOp atOp = new AffineTransformOp(
        AffineTransform.getScaleInstance(rate, rate), null);

  atOp.filter(image, shrinkImage);
  boolean ret = ImageIO.write(shrinkImage, formatName, out);
  if (!ret) {
    throw new Exception("writer not found.");
  }
}

/**
 * @param args
 */
public static void main(String[] args) {

  if (args.length <= 3) {
    printUsage();
  }

  FileInputStream in = null;
  FileOutputStream out = null;
  try {
    String srcFilename = args[0].trim();
    String dstFilename = args[1].trim();

    if (srcFilename.equals(dstFilename)) {
      System.out.println("ERROR : 変換前のファイル名と、変換後のファイル名が同じです ");
    }

    int width = 0;
    int height = 0;
    double ratio = 0.0;

    for (int i = 0; i < args.length; i++) {
      if ("-w".equalsIgnoreCase(args[i].trim())) {
        width = Integer.parseInt(args[i + 1]);
      } else if ("-h".equalsIgnoreCase(args[i].trim())) {
        height = Integer.parseInt(args[i + 1]);
      } else if ("-r".equalsIgnoreCase(args[i].trim())) {
```

```
          ratio = Double.parseDouble(args[i + 1]);
        }
      }
      in  = new FileInputStream(new File(srcFilename));
      out = new FileOutputStream(new File(dstFilename));

      ThumnailImageCreater me = new ThumnailImageCreater();

      if (width > 0 || height > 0) {
        me.writeThumbNailImage(in, out, width, height);
      } else if (ratio > 0) {
        me.writeThumbNailImage(in, out, ratio);
      } else {
        System.out.println("ERROR : サイズが指定されていません ");
      }
      System.out.println(" 完了しました ");

    } catch (Exception e) {
      e.printStackTrace();
      printUsage();
    } finally {
      try {
        if (in != null)
          in.close();
        if (out != null)
          out.close();
      } catch (Exception e) {
      }
    }
  }

  /**
   *
   */
  private static void printUsage() {
    String usage = "usage ThumNailImage " + "originalfilename "
        + "shrinkfilename " + "[-w width]" + "[-h height]"
        + "[-r ratio]";
    System.out.println(usage);
    System.exit(0);
  }
}
```