二维码、条形码

# 依赖

|  |
| --- |
| *<!-- zxing -->* <**dependency**>  <**groupId**>com.google.zxing</**groupId**>  <**artifactId**>core</**artifactId**>  <**version**>${zxing.version}</**version**> </**dependency**> <**dependency**>  <**groupId**>com.google.zxing</**groupId**>  <**artifactId**>javase</**artifactId**>  <**version**>${zxing.version}</**version**> </**dependency**>  <**zxing.version**>3.1.0</**zxing.version**> |

# 二维码

## 新建两个工具类

|  |
| --- |
| **package** com.bkgtsoft.srm.web.helper;  **import** com.google.zxing.\*; **import** com.google.zxing.common.BitMatrix; **import** com.google.zxing.common.HybridBinarizer; **import** com.google.zxing.qrcode.decoder.ErrorCorrectionLevel;  **import** javax.imageio.ImageIO; **import** java.awt.\*; **import** java.awt.geom.RoundRectangle2D; **import** java.awt.image.BufferedImage; **import** java.io.File; **import** java.io.OutputStream; **import** java.util.Hashtable; */\*\*  \** ***@author*** *iszengziqiang@163.com  \** ***@date*** *2020/11/18 14:16  \** ***@Version:*** *1.0  \** ***@desc*** *//二维码  \*/* **public class** QRCodeHelper {   **private static final** String ***CHARSET*** = **"utf-8"**;  **private static final** String ***FORMAT\_NAME*** = **"JPG"**;  *// 二维码尺寸* **private static final int *QRCODE\_SIZE*** = 300;  *// LOGO宽度* **private static final int *WIDTH*** = 60;  *// LOGO高度* **private static final int *HEIGHT*** = 60;   **private static** BufferedImage createImage(String content, String imgPath, **boolean** needCompress) **throws** Exception {  Hashtable hints = **new** Hashtable();  hints.put(EncodeHintType.***ERROR\_CORRECTION***, ErrorCorrectionLevel.***H***);  hints.put(EncodeHintType.***CHARACTER\_SET***, ***CHARSET***);  hints.put(EncodeHintType.***MARGIN***, 1);  BitMatrix bitMatrix = **new** MultiFormatWriter().encode(content, BarcodeFormat.***QR\_CODE***, ***QRCODE\_SIZE***, ***QRCODE\_SIZE***,  hints);  **int** width = bitMatrix.getWidth();  **int** height = bitMatrix.getHeight();  BufferedImage image = **new** BufferedImage(width, height, BufferedImage.***TYPE\_INT\_RGB***);  **for** (**int** x = 0; x < width; x++) {  **for** (**int** y = 0; y < height; y++) {  image.setRGB(x, y, bitMatrix.get(x, y) ? 0xFF000000 : 0xFFFFFFFF);  }  }  **if** (imgPath == **null** || **""**.equals(imgPath)) {  **return** image;  }  *// 插入图片* QRCodeHelper.*insertImage*(image, imgPath, needCompress);  **return** image;  }   **private static void** insertImage(BufferedImage source, String imgPath, **boolean** needCompress) **throws** Exception {  File file = **new** File(imgPath);  **if** (!file.exists()) {  System.***err***.println(**""** + imgPath + **" 该文件不存在！"**);  **return**;  }  Image src = ImageIO.*read*(**new** File(imgPath));  **int** width = src.getWidth(**null**);  **int** height = src.getHeight(**null**);  **if** (needCompress) { *// 压缩LOGO* **if** (width > ***WIDTH***) {  width = ***WIDTH***;  }  **if** (height > ***HEIGHT***) {  height = ***HEIGHT***;  }  Image image = src.getScaledInstance(width, height, Image.***SCALE\_SMOOTH***);  BufferedImage tag = **new** BufferedImage(width, height, BufferedImage.***TYPE\_INT\_RGB***);  Graphics g = tag.getGraphics();  g.drawImage(image, 0, 0, **null**); *// 绘制缩小后的图* g.dispose();  src = image;  }  *// 插入LOGO* Graphics2D graph = source.createGraphics();  **int** x = (***QRCODE\_SIZE*** - width) / 2;  **int** y = (***QRCODE\_SIZE*** - height) / 2;  graph.drawImage(src, x, y, width, height, **null**);  Shape shape = **new** RoundRectangle2D.Float(x, y, width, width, 6, 6);  graph.setStroke(**new** BasicStroke(3f));  graph.draw(shape);  graph.dispose();  }   **public static boolean** encode(String content, String imgPath, String destPath, **boolean** needCompress) **throws** Exception {  BufferedImage image = QRCodeHelper.*createImage*(content, imgPath, needCompress);  *mkdirs*(destPath);  *// String file = new Random().nextInt(99999999)+".jpg";  // ImageIO.write(image, FORMAT\_NAME, new File(destPath+"/"+file));* **boolean** write = ImageIO.*write*(image, ***FORMAT\_NAME***, **new** File(destPath));  **return** write;  }   **public static** BufferedImage encode(String content, String imgPath, **boolean** needCompress) **throws** Exception {  BufferedImage image = QRCodeHelper.*createImage*(content, imgPath, needCompress);  **return** image;  }   **public static void** mkdirs(String destPath) {  File file = **new** File(destPath);  *// 当文件夹不存在时，mkdirs会自动创建多层目录，区别于mkdir．(mkdir如果父目录不存在则会抛出异常)* **if** (!file.exists() && !file.isDirectory()) {  file.mkdirs();  }  }   **public static void** encode(String content, String imgPath, String destPath) **throws** Exception {  QRCodeHelper.*encode*(content, imgPath, destPath, **false**);  }  *// 被注释的方法  /\*  \* public static void encode(String content, String destPath, boolean  \* needCompress) throws Exception { QRCodeUtil.encode(content, null, destPath,  \* needCompress); }  \*/* **public static void** encode(String content, String destPath) **throws** Exception {  QRCodeHelper.*encode*(content, **null**, destPath, **false**);  }   **public static void** encode(String content, String imgPath, OutputStream output, **boolean** needCompress)  **throws** Exception {  BufferedImage image = QRCodeHelper.*createImage*(content, imgPath, needCompress);  ImageIO.*write*(image, ***FORMAT\_NAME***, output);  }   **public static void** encode(String content, OutputStream output) **throws** Exception {  QRCodeHelper.*encode*(content, **null**, output, **false**);  }   **public static** String decode(File file) **throws** Exception {  BufferedImage image;  image = ImageIO.*read*(file);  **if** (image == **null**) {  **return null**;  }  BufferedImageLuminanceSource source = **new** BufferedImageLuminanceSource(image);  BinaryBitmap bitmap = **new** BinaryBitmap(**new** HybridBinarizer(source));  Result result;  Hashtable hints = **new** Hashtable();  hints.put(DecodeHintType.***CHARACTER\_SET***, ***CHARSET***);  result = **new** MultiFormatReader().decode(bitmap, hints);  String resultStr = result.getText();  **return** resultStr;  }   **public static** String decode(String path) **throws** Exception {  **return** QRCodeHelper.*decode*(**new** File(path));   }  } |
| **package** com.bkgtsoft.srm.web.helper;  **import** java.awt.Graphics2D; **import** java.awt.geom.AffineTransform; **import** java.awt.image.BufferedImage; **import** com.google.zxing.LuminanceSource; */\*\*  \** ***@author*** *iszengziqiang@163.com  \** ***@date*** *2020/11/18 14:18  \** ***@Version:*** *1.0  \** ***@desc*** *//****todo*** *\*/* **public class** BufferedImageLuminanceSource **extends** LuminanceSource {  **private final** BufferedImage **image**;  **private final int left**;  **private final int top**;   **public** BufferedImageLuminanceSource(BufferedImage image) {  **this**(image, 0, 0, image.getWidth(), image.getHeight());  }   **public** BufferedImageLuminanceSource(BufferedImage image, **int** left, **int** top, **int** width, **int** height) {  **super**(width, height);   **int** sourceWidth = image.getWidth();  **int** sourceHeight = image.getHeight();  **if** (left + width > sourceWidth || top + height > sourceHeight) {  **throw new** IllegalArgumentException(**"Crop rectangle does not fit within image data."**);  }   **for** (**int** y = top; y < top + height; y++) {  **for** (**int** x = left; x < left + width; x++) {  **if** ((image.getRGB(x, y) & 0xFF000000) == 0) {  image.setRGB(x, y, 0xFFFFFFFF); *// = white* }  }  }   **this**.**image** = **new** BufferedImage(sourceWidth, sourceHeight, BufferedImage.***TYPE\_BYTE\_GRAY***);  **this**.**image**.getGraphics().drawImage(image, 0, 0, **null**);  **this**.**left** = left;  **this**.**top** = top;  }   @Override  **public byte**[] getRow(**int** y, **byte**[] row) {  **if** (y < 0 || y >= getHeight()) {  **throw new** IllegalArgumentException(**"Requested row is outside the image: "** + y);  }  **int** width = getWidth();  **if** (row == **null** || row.**length** < width) {  row = **new byte**[width];  }  **image**.getRaster().getDataElements(**left**, **top** + y, width, 1, row);  **return** row;  }   @Override  **public byte**[] getMatrix() {  **int** width = getWidth();  **int** height = getHeight();  **int** area = width \* height;  **byte**[] matrix = **new byte**[area];  **image**.getRaster().getDataElements(**left**, **top**, width, height, matrix);  **return** matrix;  }   @Override  **public boolean** isCropSupported() {  **return true**;  }   @Override  **public** LuminanceSource crop(**int** left, **int** top, **int** width, **int** height) {  **return new** BufferedImageLuminanceSource(**image**, **this**.**left** + left, **this**.**top** + top, width, height);  }   @Override  **public boolean** isRotateSupported() {  **return true**;  }   @Override  **public** LuminanceSource rotateCounterClockwise() {  **int** sourceWidth = **image**.getWidth();  **int** sourceHeight = **image**.getHeight();  AffineTransform transform = **new** AffineTransform(0.0, -1.0, 1.0, 0.0, 0.0, sourceWidth);  BufferedImage rotatedImage = **new** BufferedImage(sourceHeight, sourceWidth, BufferedImage.***TYPE\_BYTE\_GRAY***);  Graphics2D g = rotatedImage.createGraphics();  g.drawImage(**image**, transform, **null**);  g.dispose();  **int** width = getWidth();  **return new** BufferedImageLuminanceSource(rotatedImage, **top**, sourceWidth - (**left** + width), getHeight(), width);  }  *//https://blog.csdn.net/jam\_fanatic/article/details/82818857* } |

## 调用方法

|  |
| --- |
| **参数讲解：**  **1，二维码中的文本(扫描出来的文字)**  **2，二维码中间的logo图片，一张本地图片的绝对地址（不要logo就传空）**  **3，二维码生成的图片地址，绝对路径（例如Linux上：/usr/local/zxingcode.png）**  **4，传true即可**  **boolean** success = QRCodeHelper.*encode*(currentCodeByEnum, **""**, **imagePath**, **true**);  二维码生成成功后，将二维码的绝对路径图片上传到服务器，返回它的网络地址  FileDTO fileDTO = updateFile(**imagePath**); |

# 条形码

新建工具类，支持条形码待文本和条形码下面不带文本

|  |
| --- |
| **package** com.bkgtsoft.srm.web.helper; **import** com.google.zxing.BarcodeFormat; **import** com.google.zxing.EncodeHintType; **import** com.google.zxing.WriterException;  **import** com.google.zxing.client.j2se.MatrixToImageWriter; **import** com.google.zxing.common.BitMatrix; **import** com.google.zxing.oned.Code128Writer; **import** org.apache.commons.lang3.StringUtils;   **import** javax.imageio.ImageIO; **import** java.awt.\*; **import** java.awt.image.BufferedImage; **import** java.io.File; **import** java.io.IOException; **import** java.util.HashMap; **import** java.util.Map; */\*\*  \** ***@author*** *iszengziqiang@163.com  \** ***@date*** *2020/11/18 14:47  \** ***@Version:*** *1.0  \** ***@desc*** *//条形码  \*/* **public class** BarCodeHelper {   */\*\* 条形码宽度 \*/* **private static final int *WIDTH*** = 300;   */\*\* 条形码高度 \*/* **private static final int *HEIGHT*** = 50;   */\*\* 加文字 条形码 \*/* **private static final int *WORDHEIGHT*** = 75;  */\*\*  \* 设置 条形码参数  \*/* **private static** Map<EncodeHintType, Object> *hints* = **new** HashMap<EncodeHintType, Object>() {  **private static final long *serialVersionUID*** = 1L;  {  *// 设置编码方式* put(EncodeHintType.***CHARACTER\_SET***, **"utf-8"**);  }  };  */\*\*  \* 生成 图片缓冲  \** ***@author*** *fxbin  \** ***@param vaNumber*** *VA 码  \** ***@return*** *返回BufferedImage  \*/* **public static** BufferedImage getBarCode(String vaNumber){  **try** {  Code128Writer writer = **new** Code128Writer();  *// 编码内容, 编码类型, 宽度, 高度, 设置参数* BitMatrix bitMatrix = writer.encode(vaNumber, BarcodeFormat.***CODE\_128***, ***WIDTH***, ***HEIGHT***, *hints*);  **return** MatrixToImageWriter.*toBufferedImage*(bitMatrix);  } **catch** (WriterException e) {  e.printStackTrace();  }  **return null**;  }  */\*\*  \* 把带logo的二维码下面加上文字  \** ***@author*** *fxbin  \** ***@param image*** *条形码图片  \** ***@param words*** *文字  \** ***@return*** *返回BufferedImage  \*/* **public static** BufferedImage insertWords(BufferedImage image, String words){  *// 新的图片，把带logo的二维码下面加上文字* **if** (StringUtils.*isNotEmpty*(words)) {   BufferedImage outImage = **new** BufferedImage(***WIDTH***, ***WORDHEIGHT***, BufferedImage.***TYPE\_INT\_RGB***);   Graphics2D g2d = outImage.createGraphics();   *// 抗锯齿  setGraphics2D*(g2d);  *// 设置白色  setColorWhite*(g2d);   *// 画条形码到新的面板* g2d.drawImage(image, 0, 0, image.getWidth(), image.getHeight(), **null**);  *// 画文字到新的面板* Color color=**new** Color(0, 0, 0);  g2d.setColor(color);  *// 字体、字型、字号* g2d.setFont(**new** Font(**"微软雅黑"**, Font.***PLAIN***, 18));  *//文字长度* **int** strWidth = g2d.getFontMetrics().stringWidth(words);  *//总长度减去文字长度的一半 （居中显示）* **int** wordStartX=(***WIDTH*** - strWidth) / 2;  *//height + (outImage.getHeight() - height) / 2 + 12* **int** wordStartY=***HEIGHT***+20;   *// 画文字* g2d.drawString(words, wordStartX, wordStartY);  g2d.dispose();  outImage.flush();  **return** outImage;  }  **return null**;  }   */\*\*  \* 设置 Graphics2D 属性 （抗锯齿）  \** ***@param g2d*** *Graphics2D提供对几何形状、坐标转换、颜色管理和文本布局更为复杂的控制  \*/* **private static void** setGraphics2D(Graphics2D g2d){  g2d.setRenderingHint(RenderingHints.***KEY\_ANTIALIASING***, RenderingHints.***VALUE\_ANTIALIAS\_ON***);  g2d.setRenderingHint(RenderingHints.***KEY\_STROKE\_CONTROL***, RenderingHints.***VALUE\_STROKE\_DEFAULT***);  Stroke s = **new** BasicStroke(1, BasicStroke.***CAP\_ROUND***, BasicStroke.***JOIN\_MITER***);  g2d.setStroke(s);  }   */\*\*  \* 设置背景为白色  \** ***@param g2d*** *Graphics2D提供对几何形状、坐标转换、颜色管理和文本布局更为复杂的控制  \*/* **private static void** setColorWhite(Graphics2D g2d){  g2d.setColor(Color.***WHITE***);  *//填充整个屏幕* g2d.fillRect(0,0,600,600);  *//设置笔刷* g2d.setColor(Color.***BLACK***);  }  **public static void** main(String[] args) **throws** IOException { *// BufferedImage image = insertWords(getBarCode("123456789"), "123456789"); // A80/90R8A(8A侧通孔)* BufferedImage image = *insertWords*(*getBarCode*(**"A80/90R8A8A"**), **"A80/90R8A(8A侧通孔)"**);   **boolean** jpg = ImageIO.*write*(image, **"jpg"**, **new** File(**"/Users/zengziqiang/Desktop/barcode.png"**));  System.***out***.println(jpg);  }  } |



成功后的本地图片

## 调用方法

|  |
| --- |
| 不要文本就把第二个参数传空  BufferedImage image = *insertWords*(*getBarCode*(**"A80/90R8A8A"**), **"A80/90R8A(8A侧通孔)"**);  **boolean** jpg = ImageIO.*write*(image, **"jpg"**, **new** File(**"/Users/zengziqiang/Desktop/barcode.png"**));  jpg=true表示成功。  最后将本地图片文件上传到服务器即可 |