第一种方式

RequestOption soptions = new

RequestOptions().error(R.drawable.img\_load\_failure).bitmapTransform(new RoundedCorners(30));//图片圆角为30

Glide.with(this).load(URL) //图片地址

.apply(options)

.into(ImagView);

1.2，第二种实现：

RequestOptions requestOptions = new RequestOptions();

requestOptions.placeholder(R.drawable.ic\_launcher\_background);

requestOptions.circleCropTransform();

requestOptions.transforms( new RoundedCorners(30));

Glide.with(this).load(URL) //图片地址

.apply(options)

.into(ImagView);

1.3，第三种实现：

RequestOptions options = new RequestOptions().centerCrop() .transform(new RoundTransform(this,30));

Glide.with(this).load(URL) //图片地址

.apply(options)

.into(ImagView);

public class RoundTransform extends BitmapTransformation {

private static float radius = 0f;

public RoundTransform(Context context) {

this(context, 4);

}

public RoundTransform(Context context, int dp) {

super(context);

this.radius = Resources.getSystem().getDisplayMetrics().density \* dp;

}

@Override

protected Bitmap transform(BitmapPool pool, Bitmap toTransform, int outWidth, int outHeight) {

Bitmap bitmap = TransformationUtils.centerCrop(pool, toTransform, outWidth, outHeight);

return roundCrop(pool, bitmap);

}

private static Bitmap roundCrop(BitmapPool pool, Bitmap source) {

if (source == null) return null;

Bitmap result = pool.get(source.getWidth(), source.getHeight(), Bitmap.Config.ARGB\_8888);

if (result == null) {

result = Bitmap.createBitmap(source.getWidth(), source.getHeight(), Bitmap.Config.ARGB\_8888);

}

Canvas canvas = new Canvas(result);

Paint paint = new Paint();

paint.setShader(new BitmapShader(source, BitmapShader.TileMode.CLAMP, BitmapShader.TileMode.CLAMP));

paint.setAntiAlias(true);

RectF rectF = new RectF(0f, 0f, source.getWidth(), source.getHeight());

canvas.drawRoundRect(rectF, radius, radius, paint);

return result;

}

public String getId() {

return getClass().getName() + Math.round(radius);

}

@Override

public void updateDiskCacheKey(MessageDigest messageDigest) {

}

}

方法二：

自定义ImageView：

<ImageView

android:id="@+id/iv"

android:layout\_width="300dp"

android:layout\_height="300dp"

android:layout\_centerHorizontal="true"

/>

ImageView iv = findViewById(R.id.iv);

Bitmap bitmap =BitmapFactory.decodeResource(getResources(), R.drawable.fengjing);

Bitmap outBitmap =getRoundBitmapByShader(bitmap, 500,300,20, 3);

iv.setImageBitmap(outBitmap);

public class RoundRectImageView extends ImageView{

private Paint paint;

public RoundRectImageView(Context context) {

this(context,null);

}

public RoundRectImageView(Context context, AttributeSet attrs) {

this(context, attrs,0);

}

public RoundRectImageView(Context context, AttributeSet attrs, int defStyle) {

super(context, attrs, defStyle);

paint = new Paint();

}

/\*\*

\* 绘制圆角矩形图片

\* @author caizhiming

\*/

@Override

protected void onDraw(Canvas canvas) {

Drawable drawable = getDrawable();

if (null != drawable) {

Bitmap bitmap = getBitmapFromDrawable(drawable);

// Bitmap bitmap = ((BitmapDrawable) drawable).getBitmap();

Bitmap b = getRoundBitmapByShader(bitmap,getWidth(),getHeight(), 50,0);

final Rect rectSrc = new Rect(0, 0, b.getWidth(), b.getHeight());

final Rect rectDest = new Rect(0,0,getWidth(),getHeight());

paint.reset();

canvas.drawBitmap(b, rectSrc, rectDest, paint);

} else {

super.onDraw(canvas);

}

}

/\*\*

\* 把资源图片转换成Bitmap

\* @param drawable

\* 资源图片

\* @return 位图

\*/

public static Bitmap getBitmapFromDrawable(Drawable drawable) {

int width = drawable.getIntrinsicWidth();

int height = drawable.getIntrinsicHeight();

Bitmap bitmap = Bitmap.createBitmap(width, height, drawable

.getOpacity() != PixelFormat.OPAQUE ? Bitmap.Config.ARGB\_8888

: Bitmap.Config.RGB\_565);

Canvas canvas = new Canvas(bitmap);

//drawable.setBounds(-4, -4, width + 4, height + 4);

drawable.draw(canvas);

return bitmap;

}

public static Bitmap getRoundBitmapByShader(Bitmap bitmap, int outWidth, int outHeight, int radius, int boarder) {

if (bitmap == null) {

return null;

}

int width = bitmap.getWidth();

int height = bitmap.getHeight();

float widthScale = outWidth \* 1f / width;

float heightScale = outHeight \* 1f / height;

Matrix matrix = new Matrix();

matrix.setScale(widthScale, heightScale);

//创建输出的bitmap

Bitmap desBitmap = Bitmap.createBitmap(outWidth, outHeight, Bitmap.Config.ARGB\_8888);

//创建canvas并传入desBitmap，这样绘制的内容都会在desBitmap上

Canvas canvas = new Canvas(desBitmap);

Paint paint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

//创建着色器

BitmapShader bitmapShader = new BitmapShader(bitmap, Shader.TileMode.CLAMP, Shader.TileMode.CLAMP);

//给着色器配置matrix

bitmapShader.setLocalMatrix(matrix);

paint.setShader(bitmapShader);

//创建矩形区域并且预留出border

RectF rect = new RectF(boarder, boarder, outWidth - boarder, outHeight - boarder);

//把传入的bitmap绘制到圆角矩形区域内

canvas.drawRoundRect(rect, radius, radius, paint);

if (boarder > 0) {

//绘制boarder

Paint boarderPaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

boarderPaint.setColor(Color.GREEN);

boarderPaint.setStyle(Paint.Style.STROKE);

boarderPaint.setStrokeWidth(boarder);

canvas.drawRoundRect(rect, radius, radius, boarderPaint);

}

return desBitmap;

}

}

方法三：

对图片进行处理，此方法还可以加边框

/\*\*

\* 通过BitmapShader实现圆形边框

\* @param bitmap

\* @param outWidth 输出的图片宽度

\* @param outHeight 输出的图片高度

\* @param radius 圆角大小

\* @param boarder 边框宽度

\*/

public static Bitmap getRoundBitmapByShader(Bitmap bitmap, int outWidth, int outHeight, int radius, int boarder) {

if (bitmap == null) {

return null;

}

int height = bitmap.getHeight();

int width = bitmap.getWidth();

float widthScale = outWidth \* 1f / width;

float heightScale = outHeight \* 1f / height;

Matrix matrix = new Matrix();

matrix.setScale(widthScale, heightScale);

//创建输出的bitmap

Bitmap desBitmap = Bitmap.createBitmap(outWidth, outHeight, Bitmap.Config.ARGB\_8888);

//创建canvas并传入desBitmap，这样绘制的内容都会在desBitmap上

Canvas canvas = new Canvas(desBitmap);

Paint paint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

//创建着色器

BitmapShader bitmapShader = new BitmapShader(bitmap, Shader.TileMode.CLAMP, Shader.TileMode.CLAMP);

//给着色器配置matrix

bitmapShader.setLocalMatrix(matrix);

paint.setShader(bitmapShader);

//创建矩形区域并且预留出border

RectF rect = new RectF(boarder, boarder, outWidth - boarder, outHeight - boarder);

//把传入的bitmap绘制到圆角矩形区域内

canvas.drawRoundRect(rect, radius, radius, paint);

if (boarder > 0) {

//绘制boarder

Paint boarderPaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

boarderPaint.setColor(Color.GREEN);

boarderPaint.setStyle(Paint.Style.STROKE);

boarderPaint.setStrokeWidth(boarder);

canvas.drawRoundRect(rect, radius, radius, boarderPaint);

}

return desBitmap;

}

实现圆形和边框：

/\*\*

\* 通过BitmapShader实现圆形边框

\* @param bitmap

\* @param outWidth 输出的图片宽度

\* @param outHeight 输出的图片高度

\* @param boarder 边框大小

\*/

public static Bitmap getCircleBitmapByShader(Bitmap bitmap, int outWidth, int outHeight, int boarder) {

int radius;

int width = bitmap.getWidth();

int height = bitmap.getHeight();

float widthScale = outWidth \* 1f / width;

float heightScale = outHeight \* 1f / height;

Bitmap desBitmap = Bitmap.createBitmap(outWidth, outHeight, Bitmap.Config.ARGB\_8888);

if (outHeight > outWidth) {

radius = outWidth / 2;

} else {

radius = outHeight / 2;

}

//创建canvas

Canvas canvas = new Canvas(desBitmap);

Paint paint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

BitmapShader bitmapShader = new BitmapShader(bitmap, Shader.TileMode.CLAMP, Shader.TileMode.CLAMP);

Matrix matrix = new Matrix();

matrix.setScale(widthScale, heightScale);

bitmapShader.setLocalMatrix(matrix);

paint.setShader(bitmapShader);

canvas.drawCircle(outWidth / 2, outHeight / 2, radius - boarder, paint);

if (boarder > 0) {

//绘制boarder

Paint boarderPaint = new Paint(Paint.ANTI\_ALIAS\_FLAG);

boarderPaint.setColor(Color.GREEN);

boarderPaint.setStyle(Paint.Style.STROKE);

boarderPaint.setStrokeWidth(boarder);

canvas.drawCircle(outWidth / 2, outHeight / 2, radius - boarder, boarderPaint);

}

return desBitmap;