# 下载和设置

## Android SDK 要求

使用最低要求 - 使用 Glide 要求 SDK 版本为 API 14 (Ice Cream Sandwich) 及以上。

编译最低要求 - 编译 Glide 要在 SDK 版本为 API 26 (Oreo) 及以上。

### jar

你可以直接在 GitHub 下载[最新的jar包](https://github-production-release-asset-2e65be.s3.amazonaws.com/11267509/922de154-f001-11e4-9c02-b2e6913a1377?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIWNJYAX4CSVEH53A/20170928/us-east-1/s3/aws4_request&X-Amz-Date=20170928T111042Z&X-Amz-Expires=300&X-Amz-Signature=8ee48a992fc893e7ffc4568ba4bfe97d38684254731676563ebb3da22c6e4ed5&X-Amz-SignedHeaders=host&actor_id=0&response-content-disposition=attachment;%20filename=glide-3.6.0.jar&response-content-type=application/octet-stream)

### Gradle

如果使用 Gradle，可从 Maven Central 或 JCenter 中添加对 Glide 的依赖。同样，你还需要添加 Android 支持库的依赖。

repositories {

mavenCentral()

maven { url 'https://maven.google.com' }

}

dependencies {

compile 'com.github.bumptech.glide:glide:4.1.1'

annotationProcessor 'com.github.bumptech.glide:compiler:4.1.1'

}

### Kotlin

如果你在 Kotlin 编写的类里使用 Glide 注解，你需要引入一个 kapt 依赖，以代替常规的 annotationProcessor 依赖：

dependencies {

kapt 'com.github.bumptech.glide:compiler:4.1.1'

}

# 开始使用

## 基本用法

加载图片

Glide.with(fragment)

.load(myUrl)

.into(imageView);

取消加载图片

Glide.with(fragment).clear(imageView);

## 在RecyclerView 中使用

@Override

public void onBindViewHolder(ViewHolder holder, int position) {

String url = urls.get(position);

Glide.with(fragment)

.load(url)

.into(holder.imageView);

}

## 加载占位图

## 加载过程中的占用(Placeholder)

RequestOptions requestOptions = new RequestOptions();

requestOptions.placeholder(R.drawable.placeholder);

requestOptions.transform(new GlideCircleformation());

Glide.with(this).load(url).apply(requestOptions).into(view);

## 加载失败后显示的图片(Error)

RequestOptions requestOptions = new RequestOptions();

requestOptions.error(R.drawable.error);

requestOptions.transform(new GlideCircleformation());

Glide.with(this).load(url).apply(requestOptions).into(view);

## 图片的转换

Glide中的大部分设置项都可以通过 RequestOptions 类和 apply() 方法来应用到程序中。   
使用 request options 可以实现（包括但不限于）：

* 占位图(Placeholders)
* 转换(Transformations)
* 缓存策略(Caching Strategies)
* 组件特有的设置项，例如编码质量，或Bitmap的解码配置等。

### 加载圆形图片

Glide.with(this)

.load(url)

.apply(RequestOptions.circleCropTransform())

.into(ivTest);

### 加载图片带淡入淡出的动画效果

Glide.with(this)

.load(url)

.transition(withCrossFade())

.into(ivTest);

等等很多的转换效果,具体自己可以一个一个试试

## 缓存

Glide的默认缓存策略是AUTOMATIC,

### 在磁盘缓存

GlideApp.with(fragment)

.load(url)

.into(imageView);

RequestOptions requestOptions = new RequestOptions();

requestOptions.placeholder(R.drawable.a1);

requestOptions.diskCacheStrategy(DiskCacheStrategy.ALL)

Glide.with(this).load(url).apply(requestOptions).into(view);

### 仅从缓存加载图片

GlideApp.with(fragment)

.load(url)

.onlyRetrieveFromCache(true)

.into(imageView);

### 跳过内存缓存

GlideApp.with(fragment)

.load(url)

.skipMemoryCache(true)

.into(view);

### 跳过磁盘缓存

GlideApp.with(fragment)

.load(url)

.diskCacheStrategy(DiskCacheStrategy.NONE)

.into(view);

### 跳过所有的缓存

GlideApp.with(fragment)

.load(url)

.diskCacheStrategy(DiskCacheStrategy.NONE)

.skipMemoryCache(true)

.into(view);

### 清理磁盘的缓存

Glide.get(applicationContext).clearDiskCache();

# 高级用法

## 加载一个图片为高斯模糊效果

### 使用方法

Glide.with(getActivity()).load("http://img1.imgtn.bdimg.com/it/u=594559231,2167829292&fm=27&gp=0.jpg").apply(RequestOptions.bitmapTransform(new GlideBlurformation(getActivity()))).into(ivTest);

### 用到的其他工具类

package com.qiezzi.clinic.chengqi.common.utils;

import android.content.Context;

import android.graphics.Bitmap;

import android.support.annotation.NonNull;

import com.bumptech.glide.load.engine.bitmap\_recycle.BitmapPool;

import com.bumptech.glide.load.resource.bitmap.BitmapTransformation;

import java.security.MessageDigest;

/\*\*

\* Created by yukuoyuan on 2017/9/29.

\*/

public class GlideBlurformation extends BitmapTransformation {

private Context context;

public GlideBlurformation(Context context) {

this.context = context;

}

@Override

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

return BlurBitmapUtil.instance().blurBitmap(context, toTransform, 20,outWidth,outHeight);

}

@Override

public void updateDiskCacheKey(MessageDigest messageDigest) {

}

}

package com.qiezzi.clinic.chengqi.common.utils;

import android.annotation.TargetApi;

import android.content.Context;

import android.graphics.Bitmap;

import android.os.Build;

import android.renderscript.Allocation;

import android.renderscript.Element;

import android.renderscript.RenderScript;

import android.renderscript.ScriptIntrinsicBlur;

/\*\*

\* Created by yukuoyuan on 2017/9/29.

\*/

public class BlurBitmapUtil {

private static BlurBitmapUtil sInstance;

private BlurBitmapUtil() {

}

public static BlurBitmapUtil instance() {

if (sInstance == null) {

synchronized (BlurBitmapUtil.class) {

if (sInstance == null) {

sInstance = new BlurBitmapUtil();

}

}

}

return sInstance;

}

/\*\*

\* @param context 上下文对象

\* @param image 需要模糊的图片

\* @param outWidth 输入出的宽度

\* @param outHeight 输出的高度

\* @return 模糊处理后的Bitmap

\*/

@TargetApi(Build.VERSION\_CODES.JELLY\_BEAN\_MR1)

public Bitmap blurBitmap(Context context, Bitmap image, float blurRadius, int outWidth, int outHeight) {

// 将缩小后的图片做为预渲染的图片

Bitmap inputBitmap = Bitmap.createScaledBitmap(image, outWidth, outHeight, false);

// 创建一张渲染后的输出图片

Bitmap outputBitmap = Bitmap.createBitmap(inputBitmap);

// 创建RenderScript内核对象

RenderScript rs = RenderScript.create(context);

// 创建一个模糊效果的RenderScript的工具对象

ScriptIntrinsicBlur blurScript = ScriptIntrinsicBlur.create(rs, Element.U8\_4(rs));

// 由于RenderScript并没有使用VM来分配内存,所以需要使用Allocation类来创建和分配内存空间

// 创建Allocation对象的时候其实内存是空的,需要使用copyTo()将数据填充进去

Allocation tmpIn = Allocation.createFromBitmap(rs, inputBitmap);

Allocation tmpOut = Allocation.createFromBitmap(rs, outputBitmap);

// 设置渲染的模糊程度, 25f是最大模糊度

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.JELLY\_BEAN\_MR1) {

blurScript.setRadius(blurRadius);

}

// 设置blurScript对象的输入内存

blurScript.setInput(tmpIn);

// 将输出数据保存到输出内存中

blurScript.forEach(tmpOut);

// 将数据填充到Allocation中

tmpOut.copyTo(outputBitmap);

return outputBitmap;

}

}

具体其他效果就自己写吧,其实原理很简单,就是通过继承BitmapTransformation接口,然后在里边把bitmap处理为自己想要的效果.没有什么过于复杂的过程.

参考资料 : [Glide官方文档](https://muyangmin.github.io/glide-docs-cn/)