Fresco

1ViewonAttachedToWindowonDetachedFromWindowonStartTemporaryDetachonFinishTemporaryDetach

Fresco

1FrescoFacebookFacebook

2Fresco Android2.3(API level 9) ,androidandroid 4.0(API level 14);

35.0FrescoAPPOOMOOMandroidAndroid16M

4setImageURI()

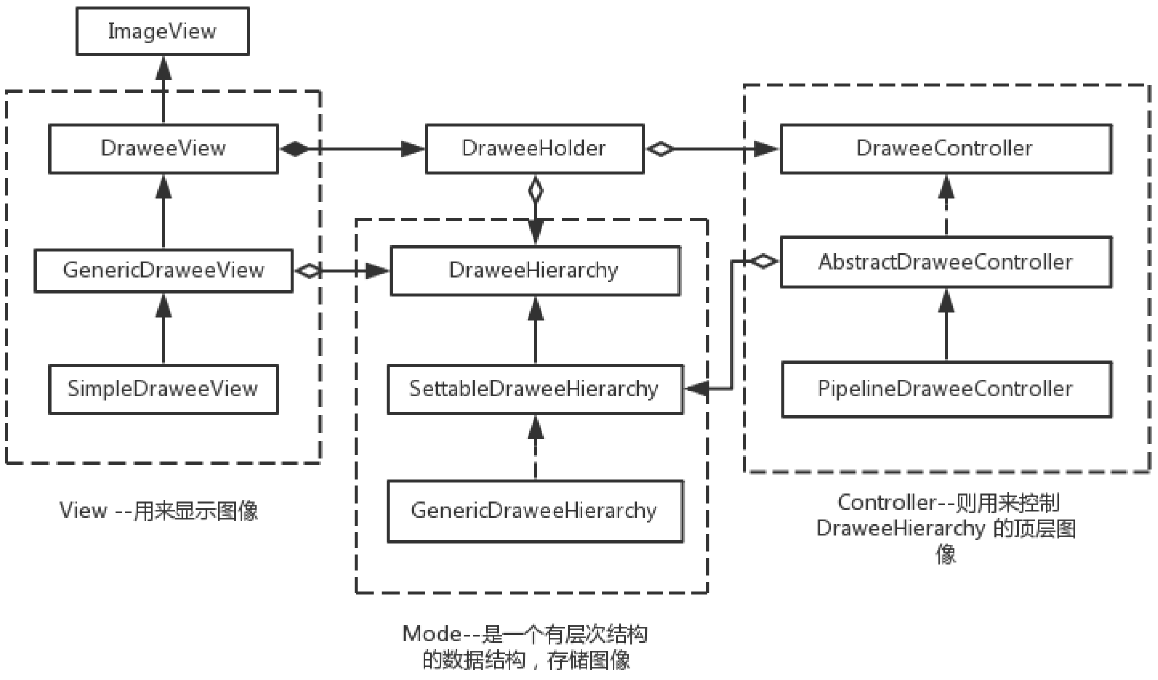
5Bitmap

6okHttpvolleyvolley

7jpg/jpegpngjpeggifwebP

Fresco

1FrescoMVC



DraweeViewDraweeHolderDraweeHolderDraweeHierarchyDraweeControllerDraweeHierarchyDraweeControllerDraweeHierarchyDraweeControllerDraweeHolder DraweeHolderDraweeView Event Controller Controller Event   Hierarchy DraweeView getTopLevelDrawable

2Fresco

FrescosetImageURI()freso

DraweeViewView

```
@Override
protected void onAttachedToWindow() {
  super.onAttachedToWindow();
  mDraweeHolder.onAttach();
}

@Override
protected void onDetachedFromWindow() {
  super.onDetachedFromWindow();
  mDraweeHolder.onDetach();
}

@Override
public void onStartTemporaryDetach() {
  super.onStartTemporaryDetach();
  mDraweeHolder.onDetach();
}

@Override
public void onFinishTemporaryDetach() {
  super.onFinishTemporaryDetach();
  mDraweeHolder.onAttach();
}
```

`onAttachedToWindow` view view windowaddviewthis view

`onDetachedFromWindow` view viewwindow removeviewthis view;

`onStartTemporaryDetach viewviewViewGrouplistviewitem`

`onFinishTemporaryDetach viewview`onStartTemporaryDetachViewGrouplistviewitem

DraweeViewviewmDraweeHolder.onAttach()mDraweeHolder.onDetach()mDraweeHolder.onAttach()ControllermDraweeHolder.onAttach()

3fresco

Frescoandroid5.0Java heapashmem buffernative heap"ashmem"BitmapBitmapGC "ashmem"JavaLinux GC5.05.0 FrescoBitmap

"ashmem"Bitmap "Purgeable Bitmap"BitmapBitmapFactory.OptionsinPurgeable

```
BitmapFactory.Options = new BitmapFactory.Options();
options.inPurgeable = true;
Bitmap bitmap = BitmapFactory.decodeByteArray(jpeg, 0, jpeg.length, options);
```

inPurgeable = trueBitmap"ashmem""ashmem"

FrescoBitmap5.0"ashmem"GC"ashmeme" Fresco SharedReferenceaddReference()deleteReference() Bitmap.recycle()Fresco CloseableReferenceSharedReferenceCloseableReference CloneableCloseable.clone()addReference() .close()deleteReference()Fresco CloseableReferenceCloseableReference

1CloseableReference.clone()

2.close()finally

```
void gee() {
     CloseableReference<Value> ref = foo();
     try {
          haa(ref);
         } finally {
              ref.close();
         }
}
```

Fresco

1DraweeView

frescofrescoSimpleDraweeViewWubaDraweeView

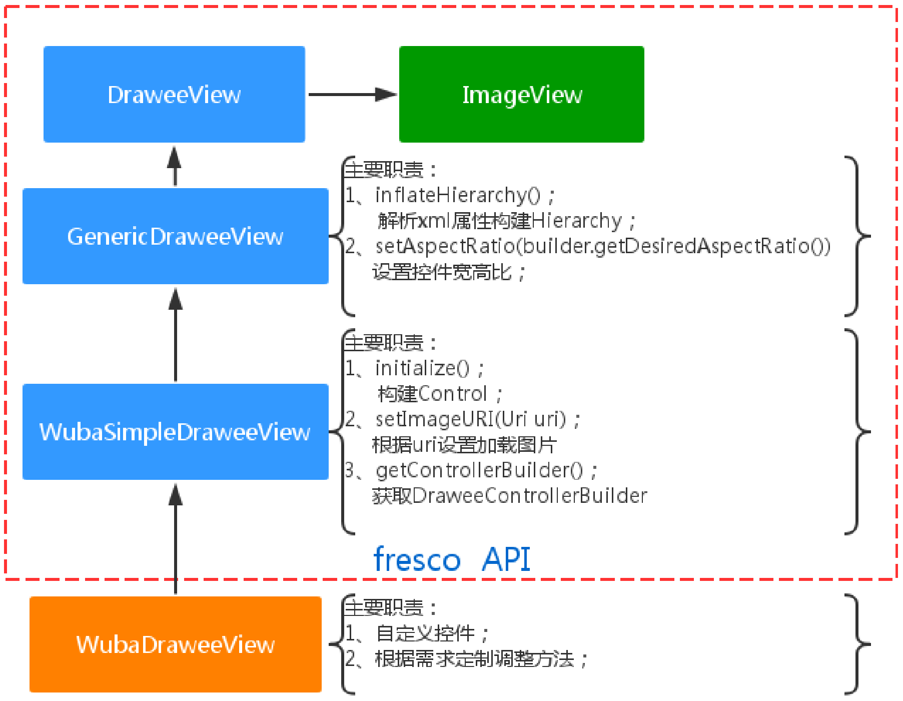
1SimpleDraweeView

2WubaDraweeView

3WubaSimpleDraweeViewPipelineDraweeControllerBuilderFrescoWubaCorePipelineDraweeControllerBuilderContextPipelineDraweeControllerBuilderPipelineDraweeControllerBuilderContextID

4xmlview

5WubaDraweeView



2CathKey

cdncdnUripath+query

```
public class CdnAwareCacheKeyFactory extends DefaultCacheKeyFactory {
    @Override
    public CacheKey getBitmapCacheKey(ImageRequest request) {
        CacheKey key = new CdnAwareBitmapMemoryCacheKey(
                getCacheKeySourceUri(request.getSourceUri()).toString(),
                getMatchCacheKey(request),
                request.getResizeOptions(),
                request.getAutoRotateEnabled(),
                request.getImageDecodeOptions(),
                null,
                null);
        LOGGER.d(DefaultConfigCentre.DEFAULT_TAG, "getBitmapCacheKey-" +
key.toString());
        return key;
    }
    ......
    @Override
    public CacheKey getEncodedCacheKey(ImageRequest request) {
        //BitmapCacheKeySimpleKeyurl
        CacheKey key = new SimpleCacheKey(getMatchCacheKey(request));
        LOGGER.d(DefaultConfigCentre.DEFAULT_TAG, "getEncodedCacheKey-" +
key.toString());
        return key;
    }
    @Override
    public Uri getCacheKeySourceUri(Uri sourceUri) {
        return sourceUri;
    }
    public String getMatchCacheKey(ImageRequest request){
        if (request==null){
            return "";
        }
        Uri uri = request.getSourceUri();
        if (uri !=null){
            String path = uri.getPath();
            String query = uri.getQuery()==null?"":uri.getQuery();
            return path+query;
        }
        return "";
    }
}
```

getMatchCacheKey(ImageRequest request)uriPathQueryCachKeykey

3

LruCach

1fresco

?

2image pipeline `setMainDiskCacheConfig` `setSmallImageDiskCacheConfig`

image request, ImageType :

```
ImageRequest request = ImageRequest.newBuilderWithSourceUri(uri)
.setImageType(ImageType.SMALL)
```

`setSmallImageDiskCacheConfigImage pipeline ImageType`

2

-

```
DiskCacheConfig diskCacheConfig = DiskCacheConfig.newBuilder()
        .setBaseDirectoryName()//
        .setBaseDirectoryPath()//
        .setMaxCacheSize()//
        .setMaxCacheSizeOnLowDiskSpace()//
        .setMaxCacheSizeOnVeryLowDiskSpace()//
        .build();
DiskCacheConfig smallDiskCacheConfig = DiskCacheConfig.newBuilder()
        .setBaseDirectoryName()//
        .setBaseDirectoryPath()//
        .setMaxCacheSize()//
        .setMaxCacheSizeOnLowDiskSpace()//
        .setMaxCacheSizeOnVeryLowDiskSpace()//
        .build();

ImagePipelineConfig pipelineConfig = ImagePipelineConfig.newBuilder(mContext)
        .setSmallImageDiskCacheConfig(smallDiskCacheConfig)//
        .setMainDiskCacheConfig(diskCacheConfig)//
        .setCacheKeyFactory(new CdnAwareCacheKeyFactory())//uripathcdn
        .build();
```

- `WubaDraweeView`

```
public void setSmallDiskImageURI(Uri uri){
    ImageRequest request = ImageRequestBuilder.newBuilderWithSource(uri)
            .setImageType(ImageRequest.ImageType.SMALL)
            .build();
    DraweeController controller = FrescoWubaCore.newDraweeControllerBuilder()
            .setImageRequest(request)
            .setOldController(getController())
            .build();
    setController(controller);
}
```

-
- ImageRequest.ImageTypefrescoDefaultDiskCacheConfigdisk

```
/**
 * An enum describing type of the image.
 */
public enum ImageType {
  /* Indicates that this image should go in the small disk cache, if one is being used */
  SMALL,

  /* Default */
  DEFAULT,
}
```

```
public void produceResults(
    final Consumer<EncodedImage> consumer,
    final ProducerContext producerContext) {
  ImageRequest imageRequest = producerContext.getImageRequest();
  if (!imageRequest.isDiskCacheEnabled()) {
    maybeStartInputProducer(consumer, consumer, producerContext);
    return;
  }

  producerContext.getListener().onProducerStart(producerContext.getId(), PRODUCER_NAME);

  final CacheKey cacheKey =
      mCacheKeyFactory.getEncodedCacheKey(imageRequest, producerContext.getCallerContext());
  boolean isSmallRequest = (imageRequest.getImageType() == ImageRequest.ImageType.SMALL);
  final BufferedDiskCache preferredCache = isSmallRequest ?
      mSmallImageBufferedDiskCache : mDefaultBufferedDiskCache;//disk
  ......
}
```

4socpu

FrescoAPP_ABI := armeabi-v7a armeabi arm64-v8a x86 x86_64sosocpusoarmeabisoarm64-v8a cpuapparm64-v8aso"nativeLibraryDirectories=[/data/app/com.xxx.xxx()/lib/arm64, /vendor/lib64, /system/lib64]]] couldn't find "libxxxx.so"

1build.gradlesoarmeabisoarmeabiapkso

```
splits {
    abi {
        enable true
        reset()
        universalApk false
        include 'armeabi'
    }
}
```

2ABIAPKABIAPK1

3apksoarmeabisoapk1.3M

5

1muddleproguard-fresco.pro

```
# Keep our interfaces so they can be used by other ProGuard rules.
# See http://sourceforge.net/p/proguard/bugs/466/
-keep,allowobfuscation @interface com.facebook.common.internal.DoNotStrip

# Do not strip any method/class that is annotated with @DoNotStrip
-keep @com.facebook.common.internal.DoNotStrip class *
-keepclassmembers class * {
@com.facebook.common.internal.DoNotStrip *;
}

# Keep native methods
-keepclassmembers class * {
native <methods>;
}

-dontwarn okio.**
-dontwarn javax.annotation.**
-dontwarn com.android.volley.toolbox.**
-dontwarn com.facebook.**
# Works around a bug in the animated GIF module which will be fixed in 0.12.0

-keep class com.facebook.imagepipeline.animated.factory.AnimatedFactoryImpl {
public AnimatedFactoryImpl(com.facebook.imagepipeline.bitmaps.PlatformBitmapFactory,
com.facebook.imagepipeline.core.ExecutorSupplier);
 }
-keep class com.facebook.imagepipeline.animated.**{*;}
```

2build.gradle

```
buildTypes {
    release {
        minifyEnabled project.runProguard as boolean
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-fresco.pro'
        signingConfig signingConfigs.release
    }
    debug {
        minifyEnabled true
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-fresco.pro'
    }
}
```

6

bitmap

```java
public void loadBitmap(final WubaDraweeView view, String path,Context context){
            GenericDraweeHierarchy hierarchy = view.getHierarchy();
            hierarchy.setFailureImage(context.getResources().getDrawable(R.drawable.discover_list_item_
e_bg_modea));
            hierarchy.setPlaceholderImage(R.drawable.discover_list_item_image_bg_modea);
            view.setHierarchy(hierarchy);
            ImageRequest imageRequest =
                    ImageRequestBuilder.newBuilderWithSource(UriUtil.parseUri(path)).build();
            BaseControllerListener baseControllerListener = new BaseControllerListener<ImageInfo>() {
                @Override
                public void onFinalImageSet(String id, ImageInfo imageInfo, Animatable animatable) {
                    super.onFinalImageSet(id, imageInfo, animatable);
                    int width = 0;
                    int height = 0;
                    float scale = 0 ;
                    try{
                        if (imageInfo!=null) {
                            width = imageInfo.getWidth();
                            height = imageInfo.getHeight();
                        }
                        scale = width / (float) height;
                    }catch (Exception e){
                        LOGGER.e("DiscoverAdapter","loadBitmap:onFinalImageSet",e);
                    }
                    if (width > height && scale > 4 / 3.0f) {//4:3,
                        GenericDraweeHierarchy hierarchy =  view.getHierarchy();
                        hierarchy.setActualImageScaleType(ScalingUtils.ScaleType.CENTER_CROP);
                    } else {
                        GenericDraweeHierarchy hierarchy =  view.getHierarchy();
                        hierarchy.setActualImageScaleType(ScalingUtils.ScaleType.FIT_CENTER);
                    }
                }

                @Override
                public void onFailure(String id, Throwable throwable) {
                    super.onFailure(id, throwable);

                }
            };
            DraweeController draweeController = FrescoWubaCore.newDraweeControllerBuilder()
                    .setOldController(view.getController())
                    .setRetainImageOnFailure(true)
                    .setImageRequest(imageRequest)
                    .setControllerListener(baseControllerListener)
                    .build();
            view.setController(draweeController);
        }

    }
```

7

fresco

1uri

```
/**
 *
 * @param loadUri
 * @return
 */
private boolean isDownloaded(Uri loadUri) {
    if (loadUri == null) {
        return false;
    }
    ImageRequest imageRequest =
ImageRequestBuilder.newBuilderWithSource(loadUri)
            .build();
    CacheKey cacheKey = new CdnAwareCacheKeyFactory()
            .getEncodedCacheKey(imageRequest);
    return ImagePipelineFactory.getInstance()
            .getMainFileCache().hasKey(cacheKey);
}
```

2

```
/**
 * uri
 * @param loadUri
 * @return
 */
private String getPath(Uri loadUri){
    ImageRequest imageRequest =
ImageRequestBuilder.newBuilderWithSource(loadUri)
            .build();
    CacheKey cacheKey = new CdnAwareCacheKeyFactory()
            .getEncodedCacheKey(imageRequest);
    BinaryResource resource = ImagePipelineFactory.getInstance()
            .getMainFileCache().getResource(cacheKey);
    File file=((FileBinaryResource)resource).getFile();
    return file.getPath();
}
```

cachkeyhasKey(cacheKey)

```
@Override
public boolean hasKey(final CacheKey key) {
  synchronized (mLock) {
    if (hasKeySync(key)) {
      return true;
    }
    try {
      String resourceId = null;
      boolean retval = false;
      if (mIndex.containsKey(key)) {
        resourceId = mIndex.get(key);
```

```
          retval = mStorage.contains(resourceId, key);
        } else {
          List<String> resourceIds = getResourceIds(key);
          for (int i = 0; i < resourceIds.size(); i++) {
            resourceId = resourceIds.get(i);
            retval = mStorage.contains(resourceId, key);
            if (retval) {
              break;
            }
          }
        }
        if (retval) {
          mIndex.put(key, resourceId);
        } else {
          mIndex.remove(key);
        }
        return retval;
      } catch (IOException e) {
        return false;
      }
    }
  }
  @VisibleForTesting
  static List<String> getResourceIds(final CacheKey key) {
    try {
      final List<String> ids;
      if (key instanceof MultiCacheKey) {
        List<CacheKey> keys = ((MultiCacheKey) key).getCacheKeys();
        ids = new ArrayList<>(keys.size());
        for (int i = 0; i < keys.size(); i++) {

  ids.add(SecureHashUtil.makeSHA1HashBase64(keys.get(i).toString().getBytes(
  "UTF-8")));
        }
      } else {
        ids = new ArrayList<>(1);

  ids.add(SecureHashUtil.makeSHA1HashBase64(key.toString().getBytes("UTF-8")
  ));
      }
      return ids;
    } catch (UnsupportedEncodingException e) {
      // This should never happen. All VMs support UTF-8
      throw new RuntimeException(e);
    }
  }

  public static String makeSHA1HashBase64(byte[] bytes) {
    try {
      MessageDigest md = MessageDigest.getInstance("SHA-1");
      md.update(bytes, 0, bytes.length);
      byte[] sha1hash = md.digest();
      return Base64.encodeToString(sha1hash, Base64.URL_SAFE |
  Base64.NO_PADDING | Base64.NO_WRAP);
    } catch (NoSuchAlgorithmException e) {
      throw new RuntimeException(e);
    }
  }
```

CachKeySecureHashUtil.makeSHA1HashBase64(keys.get(i).toString().getBytes("UTF-8"))UTF8CachKeytoStringencodeSHA-1Base64 encodeCachKeytoString

8Gif

1fresco0.10.0gifwebPgif0.10.0gif

Breaking change!

If you are using animated images (animated GIFs or animated WebPs) you need to add the following dependencies to your Gradle file:

animated GIF support:
`compile 'com.facebook.fresco:animated-gif:0.10.0'`

animated WebP support:
`compile 'com.facebook.fresco:animated-webp:0.10.0'`

if you’re using Gingerbread you also need to add:
`compile 'com.facebook.fresco:animated-base-support:0.10.0'`

2

agif

```
WubaDraweeView animatedGifView = (WubaDraweeView)
findViewById(R.id.animated_gif);
DraweeController gifController =
FrescoWubaCore.newDraweeControllerBuilder(context)
    .setAutoPlayAnimations(true)
    .setUri(UriUtil.parseUriFromResId(gifId))
    .build();
animatedGifView.setController(gifController);
```

bgif

```
WubaDraweeView animatedGifView = (WubaDraweeView)
findViewById(R.id.animated_gif);
DraweeController gifController =
FrescoWubaCore.newDraweeControllerBuilder(context)
    .setUri(UriUtil.parseUriFromResId(gifId))
    .setControllerListener(new BaseControllerListener<ImageInfo>() {
      @Override
      public void onFinalImageSet(
          String id,
          ImageInfo imageInfo,
          Animatable anim) {
        //
      }
    })
    .build();
animatedGifView.setController(gifController);
```

3)
0.10.0gifbug
# Works around a bug in the animated GIF module which will be fixed in 0.12.0

-keep class com.facebook.imagepipeline.animated.factory.AnimatedFactoryImpl {

public AnimatedFactoryImpl(com.facebook.imagepipeline.bitmaps.PlatformBitmapFactory,com.facebook.imagepipeline.core.ExecutorSupplier);

0.12.0

9OOMDownsamplingsetResizeOptions

a

image pipeline

```
.setDownsampleEnabled(true)
```

pipeline `setResizeOptions`

setResizeOptionsJPEG

resize JPEG PNG WebP()

b

```
/**
 * uri
 *
 * @param resizeWidth  px
 * @param resizeHeight  px
 * @param uri uri
 *
 */
public void setResizeOptionsImageURI(@Nullable Uri uri,int resizeWidth,int
resizeHeight){

LOGGER.d(DefaultConfigCentre.DEFAULT_TAG,"WubaDraweeView:setResizeOptionsI
mageURI == resizeWidth="+resizeWidth+",resizeHeight="+resizeHeight);
    ImageRequest imageRequest = null;
    if (uri == null){

LOGGER.d(DefaultConfigCentre.DEFAULT_TAG,"WubaDraweeView:setResizeOptionsI
mageURI == uri is null");
    }else {
        ResizeOptions options = null;
        if (resizeHeight>0&&resizeWidth>0) {
            options = new ResizeOptions(resizeWidth, resizeHeight);
        }else{

LOGGER.d(DefaultConfigCentre.DEFAULT_TAG,"WubaDraweeView:setResizeOptionsI
mageURI == resizeHeight < 0 or resizeWidth < 0");
        }
        imageRequest = ImageRequestBuilder.newBuilderWithSource(uri)
                .setResizeOptions(options)
                .build();
    }
    setControllerWithParams(imageRequest,null);
}
/**
 * uri
 *
 * @param resizeOptionsType
DefaultConfigCentre.ResizeOptionsType.SMALL_TYPE/MIDDLE_TYPE/BIG_TYPE
 * @param uri uri
 *
 */
public void setResizeOptionsTypeImageURI(Uri uri,int resizeOptionsType){
    setResizeOptionsImageURI(uri,
WubaResizeOptionsUtil.getNewResizeOptionsByType(resizeOptionsType));
}
```

c

```
/**
 * ResizeOptions
 * @param resizeOptionsType
 */
public static ResizeOptions getNewResizeOptionsByType(int
resizeOptionsType){
    int width = 0;
    int height = 0;
    switch (resizeOptionsType){
        case DefaultConfigCentre.ResizeOptionsType.SMALL_TYPE:
            width = 200;
            height = 150;
            break;
        case DefaultConfigCentre.ResizeOptionsType.MIDDLE_TYPE:
            width = 360;
            height = 300;
            break;
        case DefaultConfigCentre.ResizeOptionsType.BIG_TYPE:
            width = 720;
            height = 600;
            break;
        default:
            checkResizeOptionsType(resizeOptionsType);
            break;

    }
    return new ResizeOptions(width,height);
}
```

d

DefaultConfigCentre.ResizeOptionsType.BIG_TYPEOOM

```
@Override
public void setImageURI(Uri uri, @Nullable Object callerContext) {

setResizeOptionsTypeImageURI(uri,DefaultConfigCentre.ResizeOptionsType.BIG
_TYPE);
}
```

10

http://www.fresco-cn.org