河北师范大学软件学院
Software College of Hebei Normal University

Glide实战

智能设备教研室

- Glide简介

- 下载和设置
- 基本用法
- Generated API特性

Glide简介

- Glide是一个快速高效的Android图片加载库，注重于平滑的滚动。
- Glide提供了易用的API，统一了显示本地图片和网络图片的接口。
- Glide v4最低支持Android Ice Cream Sandwich (API level 14)。

Glide简介

- Glide 支持获取，解码和展示视频快照，图片，和GIF动画。
- 默认情况下，Glide使用的是一个定制化的基于HttpUrlConnection的方式，但同时也提供了与Google Volley和Square OkHttp快速集成的工具库。

Glide性能

- Glide 充分考虑了Android图片加载性能的两个关键方面：
  - 图片解码速度
  - 解码图片带来的资源压力

- Glide简介
  - 下载和设置
  - 基本用法
- Generated API特性

下载和设置

- 源码地址：https://github.com/bumptech/glide
- Android SDK 要求
  - Min Sdk Version - API 14 (Ice Cream Sandwich) 或更高。
  - Compile Sdk Version - API 26 (Oreo) 或更高。
  - Support Library Version - 支持库版本为 27。

下载和设置

```
repositories {
    mavenCentral()
    maven{
        url 'https://maven.google.com'
    }
}
dependencies {
 implementation 'com.android.support:appcompat-v7:27.0.2'
 //glide
 implementation 'com.github.bumptech.glide:glide:4.9.0'
 annotationProcessor 'com.github.bumptech.glide:compiler:4.5.0'
}
```

下载和设置

- 如果需要加载网络图片

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name=
                  "android.permission.ACCESS_NETWORK_STATE"/>
```

- 要从本地文件夹或 DCIM 或图库中加载图片

```
<uses-permission android:name=
                "android.permission.READ_EXTERNAL_STORAGE" />
```

- 如果要将 Glide 的缓存存储到SD 卡上

```
<uses-permission android:name=
                "android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Glide简介
- 下载和设置
- 基本用法

- Generated API特性

基本用法

```
String myUrl = "http://XXXX.jpg";
imageView = (ImageView) findViewById(R.id.image);
Glide.with(this)
     .load(myUrl)
     .into(imageView);
```

基本用法

- with()：参数支持Activity、Fragment、Context、FragmentActivity 四种类型。
- load()：支持加载远程图片，本地图片文件，图片资源，多媒体数据库的uri等。
- into()：指定图片在哪个ImageView中显示 。

指定图片格式

- 只允许加载静态图片

```
Glide.with(this)
        .asBitmap()
        .load(myUrl)
        .into(imageView);
```

- 只允许加载动态图片

```
Glide.with(this)
        .asGif()
        .load(myUrl)
        .into(imageView);
```

请求选项

- 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 RequestOptions可以实现（包括但不限于）：
  - 占位符(Placeholders)
  - 转换(Transformations)
  - 缓存策略(Caching Strategies)

占位符(Placeholders)

- Glide允许用户指定三种不同类型的占位符，分别在三种不同场景使用：
  - placeholder（占位符）：当请求正在执行时展示
  - error（错误符）：在请求永久性失败时展示
  - fallback（后备回调符）：在请求的url为 null 时展示

占位符(Placeholders)

```
//使用请求选项设置占位符
RequestOptions requestOptions = new RequestOptions()
        .placeholder(R.drawable.photo)
        .error(R.drawable.error)
        .fallback(R.drawable.fall_back);
//使用apply()方法应用占位符
Glide.with(this)
             .load(myUrl)
             .apply(requestOptions)
             .into(imageView);
```

转换(Transformations)

- 在Glide中，Transformations 可以获取资源图片并修改它，然后返回被修改后的资源。通常转换操作是用来完成剪裁或对位图应用过滤器。
- Glide 提供了很多内置的转换，包括：
  - CenterCrop
  - FitCenter
  - CircleCrop

转换(Transformations)

- 通过 RequestOptions 类可以应用转换：

```
//使用请求选择设置转换
RequestOptions requestOptions = new RequestOptions()
        .centerCrop();
//使用apply()方法应用转换
Glide.with(this)
        .load(myUrl)
        .apply(requestOptions)
        .into(imageView);
```

转换(Transformations)

- 在Glide中，当你为一个 ImageView 开始加载时，Glide 可能会自动应用 FitCenter 或 CenterCrop ，这取决于ImageView的 scaleType 。
  - 如果 scaleType 是 CENTER_CROP , Glide 将会自动应用 CenterCrop 变换。
  - 如果 scaleType 为 FIT_CENTER 或 CENTER_INSIDE ，Glide会自动使用 FitCenter 变换。

缓存策略(Caching Strategies)

- 默认情况下，Glide 会在开始一个新的图片请求之前检查以下多级的缓存：
  - 活动资源 (Active Resources)：现在是否有另一个 View 正在展示这张图片？
  - 内存缓存 (Memory cache)： 该图片是否最近被加载过并仍存在于内存中？
  - 资源类型（ Resource ）：该图片是否之前曾被解码、转换并写入过磁盘缓存？
  - 数据来源 (Data)：构建这个图片的资源是否之前曾被写入过文件缓存？

缓存策略(Caching Strategies)

- 磁盘缓存策略（DiskCacheStrategy）
  - DiskCacheStrategy.AUTOMATIC（默认的）
  - DiskCacheStrategy.ALL：缓存所有size的图片和源文件
  - DiskCacheStrategy.DATA
  - DiskCacheStrategy.RESOURCE：只缓存原图
  - DiskCacheStrategy.NONE：不缓存

缓存策略(Caching Strategies)

- 指定缓存策略

```
//使用请求选项指定缓存策略
RequestOptions requestOptions = new RequestOptions()
        .diskCacheStrategy(DiskCacheStrategy.ALL);
//使用apply()方法应用缓存策略
Glide.with(this)
        .load(myUrl)
        .apply(requestOptions)
        .into(imageView);
```

- Glide简介
  - 下载和设置
  - 基本用法
- Generated API特性

Glide V4 中Generated API特性

- Glide v4 使用注解处理器 (Annotation Processor) 来生成出一个 API，在 Application 模块中可使用该流式 API 一次性调用到RequestOptions中所有的选项。

Glide V4 中Generated API特性

- Generated API 模式的设计出于以下两个目的：
  - 可以为 Generated API 扩展自定义请求选项
  - 可以将常用的选项组打包成一个选项在Generated API 中使用

Glide V4 中Generated API特性

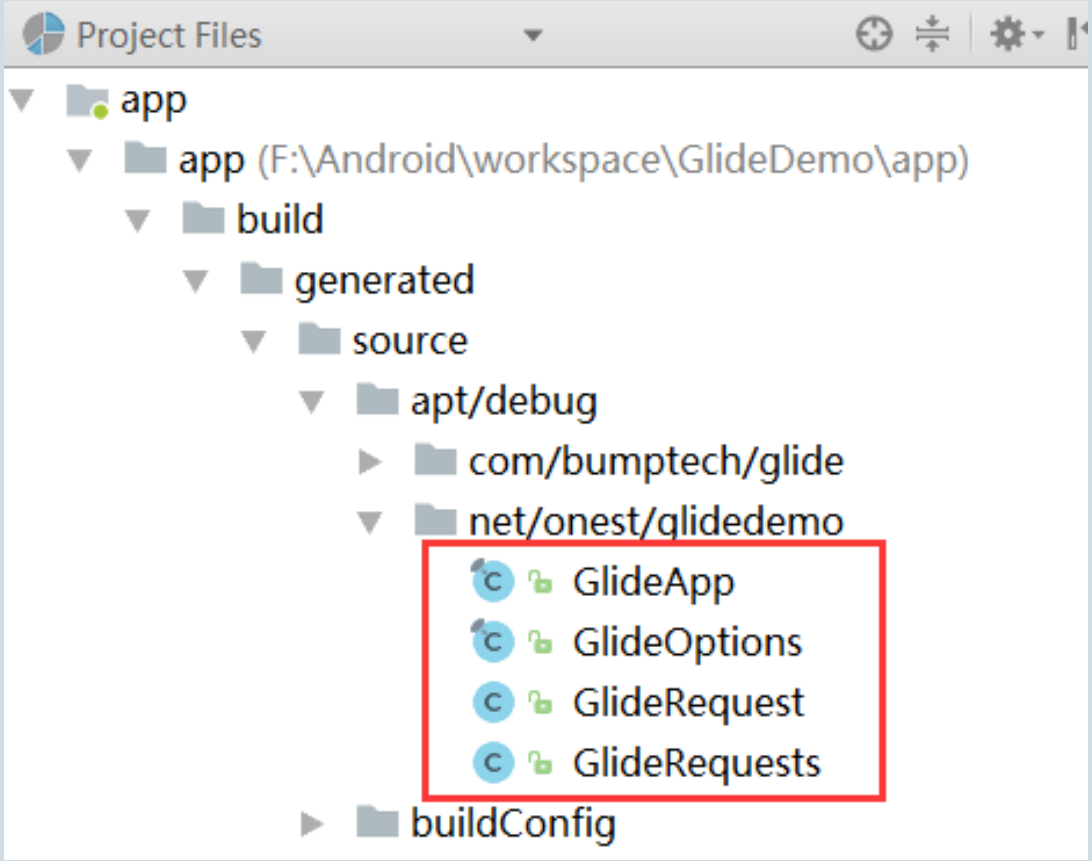
- 要在 Application 模块中使用 Generated API，你需要执行以下两步：
  - 添加 Glide 注解处理器的依赖
  - 在 Application 中定义一个 AppGlideModule 的子类，并且添加@GlideModule注解

Glide V4 中Generated API特性

```
repositories {
    mavenCentral()
}
dependencies {
annotationProcessor 'com.github.bumptech.glide:compiler:4.9.0'
}


@GlideModule
public class MyAppGlideModule extends AppGlideModule{
}
```

Glide V4 中Generated API特性



Glide V4 中Generated API特性

- Generated API 默认名为 GlideApp ，与AppGlideModule的子类包名相同。将 Glide.with() 替换为 GlideApp.with()，即可使用该 API 去完成加载图片工作。

```
GlideApp.with(this)
        .load(myUrl)
        .into(imageView);
```

Glide V4 中Generated API特性

- 使用 Generated API，所有的请求选项方法都会被内联并可以直接使用：

```
GlideApp.with(this)
        .load(myUrl)
        .placeholder(R.drawable.photo)
        .error(R.drawable.error)
        .fallback(R.drawable.fall_back)
        .into(imageView);
```

Glide V4 中Generated API特性

```
//转换
GlideApp.with(this)
        .load(myUrl)
        .centerCrop()
        .fitCenter()
        .circleCrop()
        .into(imageView);
```

Thank You!