Glide

Customizable Image Loading on Android

Norman Peitek & Marcus Pöhls

Glide: Customizable Image Loading on Android

Norman Peitek

This book is for sale at http://leanpub.com/glide-image-loading-on-android

This version was published on 2019-02-17

Also By Norman Peitek

Picasso: Easy Image Loading on Android

Retrofit: Love Working with APIs on Android

Gson: Enjoy JSON (De-)Serialization in Java

Gson Workbook

Contents

About the Book

In order to be competitive in today's Google Play environment, your app needs to be polished. Images need to be used consistently and loaded smoothly. If your Android app uses images, this book will save you a ton of time researching and avoid stressful evenings of bug fixes. If you value your time, this is something for you.

We cover every feature of Glide! This book is for beginners and advanced readers as well. We'll walk you through each topic with direct reference to code examples. Once you've worked through this book, you'll be the expert of image loading on Android with Glide.

This book is for beginners and advanced readers as well. We'll walk you through each topic with direct reference to code examples. Once you've worked through this book, you'll have an extensive knowledge of image loading on Android with Glide.

What Topics Are Covered in This Book?

The list below provides a comprehensive overview of covered topics within the book.

- Introduction to Glide
- Loading Images, Gifs & Local Videos
- Image Display & Placeholders
- **Image Resizing** & Thumbnails
- **Caching** & Request Priorities
- Callbacks With Glide Targets
- **Exception Handling** and Debugging
- **Image Rotation** and Glide Transformations
- Glide Animations
- Customizing Glide with **Glide Modules**
- App Release Preparations
- Upgrade Guide from Glide 3.x

Who Is This Book For?

This book is for Android developers who want to get an substantial overview and reference book of Glide. You'll benefit from the clearly recognizable code examples in regard to your daily work with Glide.

Rookie

If you're just starting out with Glide (or coming from any other image loading library like Picasso) this book will show you all important parts on how to work with images. The provided code snippets let you jumpstart and create an image-rich app within minutes.

Expert

You already worked with Glide before? You'll profit from our extensive code snippets and can improve your existing code base. Additionally, the book illustrates various optimizations for an even better user experience.

Code Samples

Every code snippet in the book has been tested. The example project is provided for free as an extra download. We recommend to download it from Leanpub's book extra section and check the full code base as you move through the book. We try to keep the code snippets in the book as short as possible. If more context is necessary, the example app's code provides it.

Chapter 1 — Loading Images, Gifs & Local Video Thumbnails

After a lot of success and feedback on our Picasso[1] series, we're following the requests to do an extensive series on another amazing image loading library: Glide[2]. Glide, like Picasso, can load and display images from many sources, while also taking care of caching and keeping a low memory impact when doing image manipulations. It has been used by official Google apps (like the app for Google I/O 2015) and is as popular as Picasso. In this book, we're going to explore the functionalities of Glide and why it's superior in certain aspects.

Why Use Glide?

Experienced Android developers can skip this section, but for the starters: you might ask yourself why you want to use Glide* instead of your own implementation.

Android is quite the diva when working with images, since it'll load images into the memory pixel by pixel[3]. A single photo of an average phone camera with the dimensions of 2592x1936 pixels (5 megapixels) will allocate around 19 MB of memory. If you add the complexity of network requests on spotty wireless connections, caching and image manipulations, you will save yourself a lot of time & headache, if you use a well-tested and developed library like Glide

In this book, we'll look at many of the features of Glide. Just take a peek at the table of contents and think if you really want to develop all of these features yourself.

* = *or any other image loading library, like Picasso, ION, etc.*

Adding Glide to Your Setup

Hopefully by now we've convinced you to use a library to handle your image loading requests. If you want to take a look at Glide, this is the guide for you!

First things first, add Glide to your dependencies. At the time of writing, the last stable version of Glide is `4.6.1`.

Gradle

As with most dependencies, pulling it in a Gradle project is a couple of lines in your `build.gradle`:

[1] https://futurestud.io/tutorials/picasso-series-round-up/
[2] https://github.com/bumptech/glide
[3] http://developer.android.com/training/displaying-bitmaps/index.html

```
// image loading library Glide itself
implementation 'com.github.bumptech.glide:glide:4.8.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.8.0'
```

Maven

While we moved all our projects to Gradle, Glide also supports Maven projects:

```
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>glide</artifactId>
  <version>4.8.0</version>
</dependency>
<dependency>
  <groupId>com.github.bumptech.glide</groupId>
  <artifactId>compiler</artifactId>
  <version>4.8.0</version>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>com.google.android</groupId>
  <artifactId>support-v4</artifactId>
  <version>r7</version>
</dependency>
```

Internet Permission

We use Glide to download and display images via HTTP requests against a server somewhere in the Internet. Executing those requests from an Android application requires the Internet permission to open network sockets. You need to define the permission within the `AndroidManifest.xml` file. If you didn't set the Internet permission yet, please add the following line within your manifest definition:

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

A common practice is to add your app permissions as the first elements in your manifest file. The following snippet is an excerpt from the sample project you can find within the extras of this book on Leanpub.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest package="io.futurestud.glidebook" xmlns:android="http://schemas.android.co\
m/apk/res/android">

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

First Peek: Loading Image from a URL

Like Picasso, the Glide library is using a fluent interface[4]. The Glide builder requires at least three parameters for a fully functional request:

- `with(Context context)` - Context[5] is necessary for many Android API calls. Glide is no difference here. Glide makes this very convenient by also extracting the context if you pass an `Activity` or `Fragment` object.
- `load(String imageUrl)` - here you specify which image should be loaded. Mostly it'll be a String representing a URL to an Internet image.
- `into(ImageView targetImageView)` - the target `ImageView` your image is supposed to get displayed in.

Theoretical explanations are always harder to grasp, so let's look at a hands-on example:

```
ImageView targetImageView = (ImageView) findViewById(R.id.imageView);
String internetUrl = "http://i.imgur.com/DvpvklR.png";

Glide
    .with(context)
    .load(internetUrl)
    .into(targetImageView);
```

That's it! If the image at the URL exists and your `ImageView` is visible, you'll see the image in a few seconds. In case the image does not exist, Glide will return to the error callbacks, which we'll look at later. You might already be convinced with this three-line example that Glide is useful to you, but this is just the tip of the feature iceberg.

Generated API for Glide 4.x

Glide is very customizable. This is true for the 3.x version, but the extension system got even more powerful for Glide 4.x. The fluent interface we've shown you above is fixed for Glide 3.x. Starting with the 4.0 release Glide creates a custom fluent interface depending on your customization. Glide's

[4] http://en.wikipedia.org/wiki/Fluent_interface
[5] http://developer.android.com/reference/android/content/Context.html

developers call this result *generated API*[6]. To differentiate between the standard Glide object and the generated API, we'll use different names.

To create the generated API, which we'll use in all upcoming chapters, you'll have to create a class extending `AppGlideModule` somewhere in your app project:

```
@GlideModule
public class FutureStudioAppGlideModule extends AppGlideModule {

}
```

Don't forget the `@GlideModule`! Then, by default, the generated API is accessible via `GlideApp`. For example:

```
ImageView targetImageView = (ImageView) findViewById(R.id.imageView);
String internetUrl = "http://i.imgur.com/DvpvklR.png";

GlideApp
    .with(context)
    .load(internetUrl)
    .into(targetImageView);
```

Whenever you see `GlideApp` in our code snippets you know we're using Glide 4.x, and the generated API. We'll go deeper into the generated API in an upcoming, new section.

In the next section, we'll start by looking at other options to load images, besides from an Internet URL. Specifically, we'll load an image from Android resources, local files and a Uri.

Advanced Loading

In the last section, we've looked at reasons for using Glide and a simple example request to load an image from an Internet source. But this is not the only possible image source for Glide. Glide can also load images from the Android resources, files and Uri[7]'s. In this part, we'll cover all three options.

Loading from Resources

First up is loading from Android resources. Instead of giving a `String` pointing to an Internet URL, you give a resource `int`.

[6] http://bumptech.github.io/glide/doc/generatedapi.html
[7] http://developer.android.com/reference/android/net/Uri.html

```
int resourceId = R.mipmap.ic_launcher;

GlideApp
    .with(context)
    .load(resourceId)
    .into(imageViewResource);
```

If you're confused by the *R.mipmap.*, it's Android's new way[8] of handling icons.

Of course, you can set a resource directly by using the methods of the `ImageView` class. However, this can be interesting if you're using more advanced topics like dynamic transformations.

Loading from File

Second up is loading from a file. This can be useful when you let the user select a photo to display an image (similar to a gallery). The parameter is just a `File` object. In an example, we look up the file from our external public directory. Note that this snipped most likely won't work for you (just if you've `Running.jpg` on your phone).

```
// this file probably does not exist on your device.
// However, you can use any file path, which points to an image file
File file = new File(
  Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES),
  "Running.jpg");

GlideApp
    .with(context)
    .load(file)
    .into(imageViewFile);
```

Loading from Uri

Lastly, you can also load images defined by an `Uri`. The request is no different from the previous options:

[8] http://android-developers.blogspot.de/2014/10/getting-your-apps-ready-for-nexus-6-and.html

```
// this could be any URI
// for demonstration purposes we're just
// creating an Uri pointing to a launcher icon
Uri uri = resourceIdToUri(context, R.mipmap.future_studio_launcher);

GlideApp
    .with(context)
    .load(uri)
    .into(imageViewUri);
```

The small helper function is a simple conversion from the resourceId to an `Uri`.

```
public static final String ANDROID_RESOURCE = "android.resource://";
public static final String FOREWARD_SLASH = "/";

private static Uri resourceIdToUri(Context context, int resourceId) {
    return Uri.parse(
        ANDROID_RESOURCE + context.getPackageName() + FOREWARD_SLASH + resourceId
    );
}
```

However, the `Uri` does not have to be generated from a resourceId. It can be any `Uri`.

Load Bitmap or Drawable

Starting with Glide `4.3.0`, you can pass a Bitmap or Drawable object to the `load()` method.

```
Bitmap bm = ...

Glide
    .with(context)
    .load(bm)
    .into(imageView);
```

The basic loading principles are done, now we can finally look at more interesting stuff. In the next section, we'll show you how to display Gifs and the thumbnail of local videos with Glide.

Displaying Gifs & Video Thumbnails

This section will show you a unique feature of Glide: displaying Gifs and the thumbnail of local videos. A lot of image loading libraries ship with the functionality to load and display images. Gif

support is something special and very helpful if you need it in your app. Glide makes the experience with Gifs especially amazing because it's so simple. If you want to display a Gif, you can just use the same regular call you've been using in the past with images:

```
String gifUrl = "http://i.kinja-img.com/gawker-media/image/upload/s--B7tUiM5l--/gf2r\
69yorbdesguga10i.gif";

GlideApp
    .with(context)
    .load(gifUrl)
    .into(imageViewGif);
```

That's it! This will display the Gif in the `ImageView` and automatically start playing it. Another great thing about Glide is that you still can use all your standard calls to manipulate the Gif:

```
GlideApp
    .with(context)
    .load(gifUrl)
    .placeholder(R.drawable.cupcake)
    .error(R.drawable.full_cake)
    .into(imageViewGif);
```

Gif Check

A potential issue with the code above is, that if the source you provide is not a Gif, and maybe just a regular image, there is no way of registering that issue. Glide accepts Gifs or images as the `.load()` parameter. If you, the developer, expect that the URL is a Gif, which is not the case, Glide can't automatically detect that. Thus, they introduced an additional method to force Glide into expecting a Gif `.asGif()`:

```
GlideApp
    .with(context)
    .asGif()
    .load(gifUrl)
    .error(R.drawable.full_cake)
    .into(imageViewGifAsGif);
```

If the gifUrl is a gif, nothing changes. However, unlike before, if the gifUrl is not a Gif, Glide will understand the load as failed. This has the advantage, that the `.error()` callback gets called and the error placeholder gets shown, even if the gifUrl is a perfectly good image (but not a Gif).

Cache Settings

In most cases, the Gif will load significantly faster when you're using `.diskCacheStrategy(DiskCacheStrategy.DATA`

```
GlideApp
    .with(context)
    .load(gifUrl)
    .asGif()
    .error(R.drawable.full_cake)
    .diskCacheStrategy(DiskCacheStrategy.DATA)
    .into(imageViewGif);
```

More Information on Cache Settings Later in the Book

We'll come back to the `diskCacheStrategy` and the reasoning in the fourth chapter on caches. You'll learn why it makes Gifs (and in some cases even photos) load so much faster.

Display Gif as Bitmap

If your app displays an unknown list of Internet URLs, it might encounter regular images or Gifs. In some cases, you might be interested in not displaying the entire Gif. If you only want to display the first frame of the Gif, you can call `.asBitmap()` to guarantee the display as a regular image, even if the URL is pointing to a Gif.

```
GlideApp
    .with(context)
    .asBitmap()
    .load(gifUrl)
    .into(imageViewGifAsBitmap);;
```

This should give you all the knowledge to display Gifs with Glide. It's easy, try it!

Display of Local Video Thumbnails

Another step up from Gifs are videos. Glide is also able to display the first frame of videos, as long as they're stored on the phone. Let's assume you get the file path by letting the user select a video:

```
String filePath = "/storage/emulated/0/Pictures/example_video.mp4";

GlideApp
    .with(context)
    .asBitmap()
    .load(Uri.fromFile(new File(filePath)))
    .into(imageViewGifAsBitmap);
```

One more time: it's important to note that this only works for **local** videos and only displays the thumbnail. Videos, which are not stored on the device (for example Internet URLs), will not work! If you want to play videos or display them from an Internet URL, look into VideoView[9].

After reading this section, you should be able to work with Gifs and local videos as well as you already can with images. Glide makes working with Gifs very smooth and convenient.

Chapter Summary

In this chapter, you learned the basics of Glide. You should have learned:

- [x] What Glide is
- [x] Why to use Glide
- [x] How to integrate Glide into your project
- [x] How to load images from various resources
- [x] How to load Gifs
- [x] How to load the thumbnail of a local video

In the next chapter we'll cover adapter use and Glide's caching in `ListViews` and `GridViews`. We also will show you what placeholders Glide offers. Lastly, will demonstrate Glide's fade animations.

Additional Chapter Resources

- Glide on Github[10]
- Official Documentation[11]
- Glide's Javadocs[12]
- Android SDK & Android Studio IDE[13]

[9] http://developer.android.com/reference/android/widget/VideoView.html
[10] https://github.com/bumptech/glide
[11] http://bumptech.github.io/glide/
[12] http://bumptech.github.io/glide/javadocs/latest/com/bumptech/glide/package-summary.html
[13] https://developer.android.com/sdk/index.html

Outro

Our goal is to truly help you getting started and improve your knowledge around Glide. We hope you learned many new things throughout this book. We want you to save time while learning the basics and in-depth details about Glide. We think the existing Glide documentation lacks various information and this book should help you to gain extensive in-depth knowledge without loosing time searching StackOverflow for correct answers.

We'll update the content of this book to later Glide versions as new releases become available. However, it will take some time to update the code for breaking changes introduced to Glide. Of course, we'll let you about any book updates.

As a reader of this book, we'll always reward your loyalty by publishing exclusive content and any future update will —of course— be free for you!

For us it's really important to exceed our reader's expectations. In all our products and guides we aim for a high quality. If you feel like a section or chapter in this book wasn't clear or extensive enough, please let us know at info@futurestud.io[14]. We always love hearing back from you, so if you have anything to say, don't hesitate to shoot us an email. We welcome any feedback, critic, suggestions for new topics or whatever is currently on your Glide mind!

Don't forget, we're publishing new tutorials every Wednesday and Thursday, mainly about Android and Node.js within the Future Studio University. Feel free to visit our homepage[15] and the University[16] :)

Finally, we're also releasing a YouTube video in 4k resolution every Monday and Friday. We're working on a 10+ part series on Glide. Feel free to check out our Future Studio YouTube channel[17] in order to miss not any upcoming Glide videos :)

Thanks a lot for reading this book! We truly appreciate your interest and hope you learned a lot from reading this book! <3

— Norman & Marcus

[14] mailto:info@futurestud.io
[15] https://futurestud.io
[16] https://futurestud.io/tutorials
[17] https://www.youtube.com/c/FutureStudio