**Picasso — Adapter Use (ListView, GridView, ...)**

 by [Norman Peitek](https://futurestud.io/tutorials/author/norman) on June 11 2015, tagged in [Development](https://futurestud.io/tutorials/tag/development), [Android](https://futurestud.io/tutorials/tag/android), [Picasso](https://futurestud.io/tutorials/tag/picasso)

The last two posts in this series have shown how to load a single image into an ImageView. This post will demonstrate a ListView implementation, where each row contains a single ImageView. This is similar to many image gallery apps.

**Picasso — Getting Started & Simple Loading Series Overview**

1. [Picasso — Getting Started & Simple Loading](https://futurestud.io/tutorials/picasso-getting-started-simple-loading)
2. [Picasso — Advanced Loading](https://futurestud.io/tutorials/picasso-advanced-loading)
3. **Picasso — Adapter Use (ListView, GridView, ...)**
4. [Picasso — How to Deal with Null/Empty Values (in ListViews)](https://futurestud.io/tutorials/picasso-how-to-deal-with-null-empty-values-in-listviews)
5. [Picasso — Placeholders, Errors and Fading](https://futurestud.io/tutorials/picasso-placeholders-errors-and-fading)
6. [Picasso — Image Resizing, Scaling and fit()](https://futurestud.io/tutorials/picasso-image-resizing-scaling-and-fit)
7. [Picasso — Ordering Requests by Image Priority](https://futurestud.io/tutorials/picasso-ordering-requests-by-image-priority)
8. [Picasso — Request Management by Grouping of Images via Tag()](https://futurestud.io/tutorials/request-management-by-grouping-of-images-via-tag)
9. [Picasso — Callbacks, RemoteViews and Notifications](https://futurestud.io/tutorials/picasso-callbacks-remoteviews-and-notifications)
10. [Picasso — Image Rotation and Transformation](https://futurestud.io/tutorials/picasso-image-rotation-and-transformation)
11. [Picasso — Influencing Image Caching](https://futurestud.io/tutorials/picasso-influencing-image-caching)
12. [Picasso — Cache Indicators, Logging & Stats](https://futurestud.io/tutorials/picasso-cache-indicators-logging-stats)
13. [Picasso — Customizing Picasso with Picasso.Builder](https://futurestud.io/tutorials/picasso-customizing-picasso-with-picasso-builder)
14. [Picasso — Custom Request Handlers](https://futurestud.io/tutorials/picasso-custom-request-handlers)
15. [Picasso — Series Round-Up](https://futurestud.io/tutorials/picasso-series-round-up)
16. [Picasso 2.5 — Integrate OkHttp3 as Network Stack](https://futurestud.io/tutorials/picasso-2-5-integrate-okhttp3-as-network-stack)
17. [Picasso — Additional Transformations with Transformation Library](https://futurestud.io/tutorials/picasso-additional-transformations-with-transformation-library)

**Sample Gallery Implementation: ListView**

First, we'll need some test images. We uploaded a selection of the best recipe images from our [eatfoody.com](http://eatfoody.com/) project to [imgur](http://imgur.com/a/uz4uZ):

public static String[] eatFoodyImages = {

"http://i.imgur.com/rFLNqWI.jpg",

"http://i.imgur.com/C9pBVt7.jpg",

"http://i.imgur.com/rT5vXE1.jpg",

"http://i.imgur.com/aIy5R2k.jpg",

"http://i.imgur.com/MoJs9pT.jpg",

"http://i.imgur.com/S963yEM.jpg",

"http://i.imgur.com/rLR2cyc.jpg",

"http://i.imgur.com/SEPdUIx.jpg",

"http://i.imgur.com/aC9OjaM.jpg",

"http://i.imgur.com/76Jfv9b.jpg",

"http://i.imgur.com/fUX7EIB.jpg",

"http://i.imgur.com/syELajx.jpg",

"http://i.imgur.com/COzBnru.jpg",

"http://i.imgur.com/Z3QjilA.jpg",

};

Second, we'll require an activity, which creates an adapter and sets it for a ListView:

public class UsageExampleAdapter extends ActionBarActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_usage\_example\_adapter);

listView.setAdapter(new ImageListAdapter(UsageExampleAdapter.this, eatFoodyImages));

}

}

Third, let's look at the layout files for the adapter. The layout file for a ListView item is very simple:

<?xml version="1.0" encoding="utf-8"?>

<ImageView xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="200dp"/>

This will result in a list of images, which each will have a height of 200dp and match the device's width. Obviously, this will not result in the prettiest image gallery, but that's not the focus of this exercise.

Before we can jump to the result, we'll need to implement an [adapter](http://developer.android.com/reference/android/widget/Adapter.html) for the ListView. We'll keep it simple and bind our eatfoody example images to the adapter. Each item will display one image.

public class ImageListAdapter extends ArrayAdapter {

private Context context;

private LayoutInflater inflater;

private String[] imageUrls;

public ImageListAdapter(Context context, String[] imageUrls) {

super(context, R.layout.listview\_item\_image, imageUrls);

this.context = context;

this.imageUrls = imageUrls;

inflater = LayoutInflater.from(context);

}

@Override

public View getView(int position, View convertView, ViewGroup parent) {

if (null == convertView) {

convertView = inflater.inflate(R.layout.listview\_item\_image, parent, false);

}

Picasso

.with(context)

.load(imageUrls[position])

.fit() // will explain later

.into((ImageView) convertView);

return convertView;

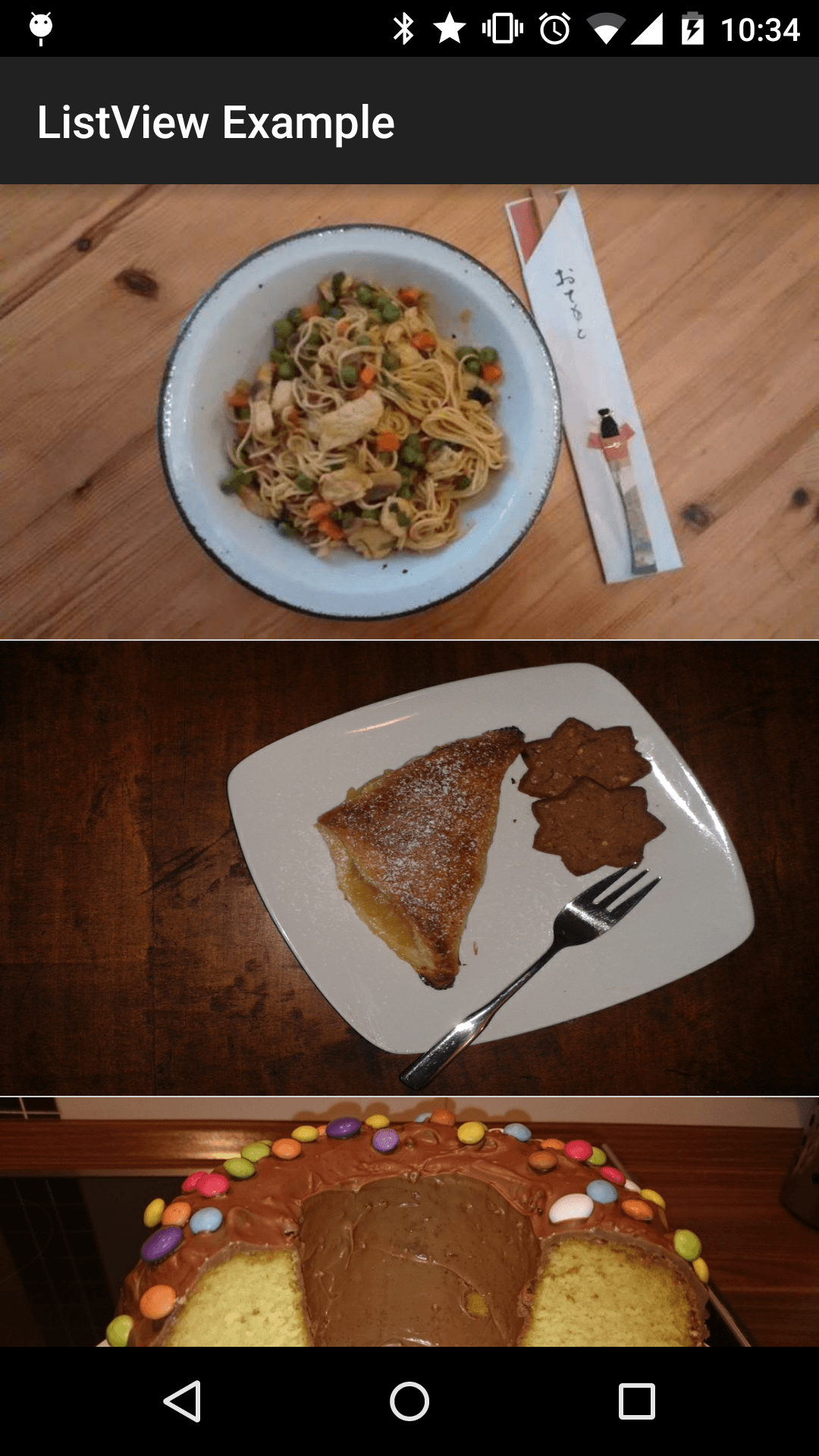
}

}

The interesting stuff happens in the getView() method of the ImageListAdapter. You'll see that the Picasso call is exactly the same as in the previously used 'regular' loading of images. The way to utilize Picasso stays the same, no matter what application you're trying to cover.

As an advanced Android developer you will know that we need to re-use layouts in ListViews to create a fast & smooth scrolling experience. One awesomeness of Picasso is that it automatically takes care of the request canceling, clearing of the ImageViews, and loading the correct image into the appropriate ImageView.

*Side note: there is an option to further optimize adapter use with fit() and tags(). We'll talk about those in an upcoming blog post.*

L

istView with eatfoody Images

**A Strength of Picasso: Caching**

When you scroll up and down a lot, you'll see that the images are displayed much faster than previously. As you can guess, these images come from cache and are not loaded from the network anymore. Picasso's cache implementation is well rounded and will make things for you a lot easier. The size of the implemented cache depends on the device's disk size.

When loading an image, Picasso uses three sources: memory, disk and network (ordered from fastest to slowest). Once again, there is nothing you'll have to do. Picasso hides all that complexity from you, while creating intelligently sized caches for you. Of course, aligned with Square philosophy, you can influence or replace the caching component. We'll take a closer look at the caching in a later blog post.

**GridView**

The implementation for a GridView with image elements is not any different from a ListView implementation. You actually can use the same adapter. Just switch out the activity layout to a GridView:

<?xml version="1.0" encoding="utf-8"?>

<GridView

android:id="@+id/usage\_example\_gridview"

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:numColumns="2"/>

This will result in the following design:

GridView with eatfoody Images

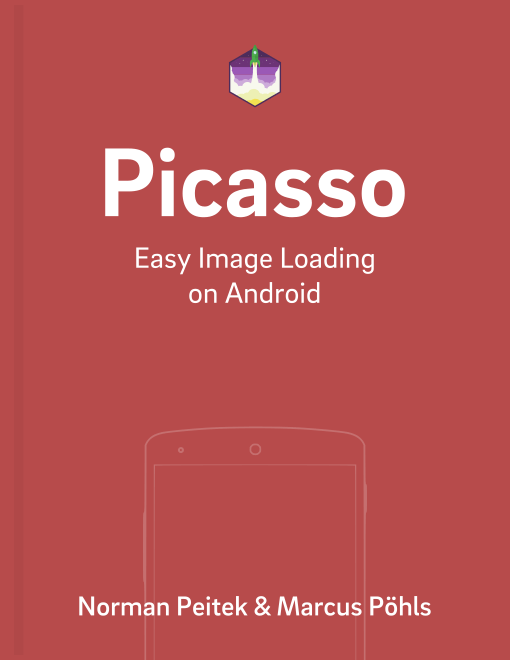
**Other Applications: ImageViews as Elements**

So far, we've only looked at examples where the entire adapter item is an ImageView. The approach still applies if one or more ImageViews are only a (small) part of the adapter item. Your getView() code will look a little different, but the loading of the Picasso item would be identical.

**Outlook**

At this point, you've learned how to load images with Picasso in 90% of the Android use cases. Before we cover the edge cases, we'll explain additional capabilities of Picasso (besides image loading and caching). Namely, next week will be all about placeholders.

**Not enough Picasso? Buy our book!**

[[](https://leanpub.com/picasso-image-loading-on-android)Picasso Book](https://leanpub.com/picasso-image-loading-on-android)

Learn how to create image-rich Android apps with Picasso. Boost your productivity and save yourself valuable time by learning all the required optimizations for an amazing user experience.

**Picasso: Easy Image Loading on Android** is available [for sale on Leanpub.com](https://leanpub.com/picasso-image-loading-on-android)

**Get Notified on New Future Studio Content**

Get your weekly push notification about  
new and trending Future Studio content.

Начало формы

Subscribe

Конец формы

[Norman Peitek](https://futurestud.io/tutorials/author/norman)

android & node.js dev

<http://twitter.com/peitek>