# Week 5

@wesreisz

# Agenda

- Review Test
- Android Manifest
- Simple Data Sharing
- ListView / GridView

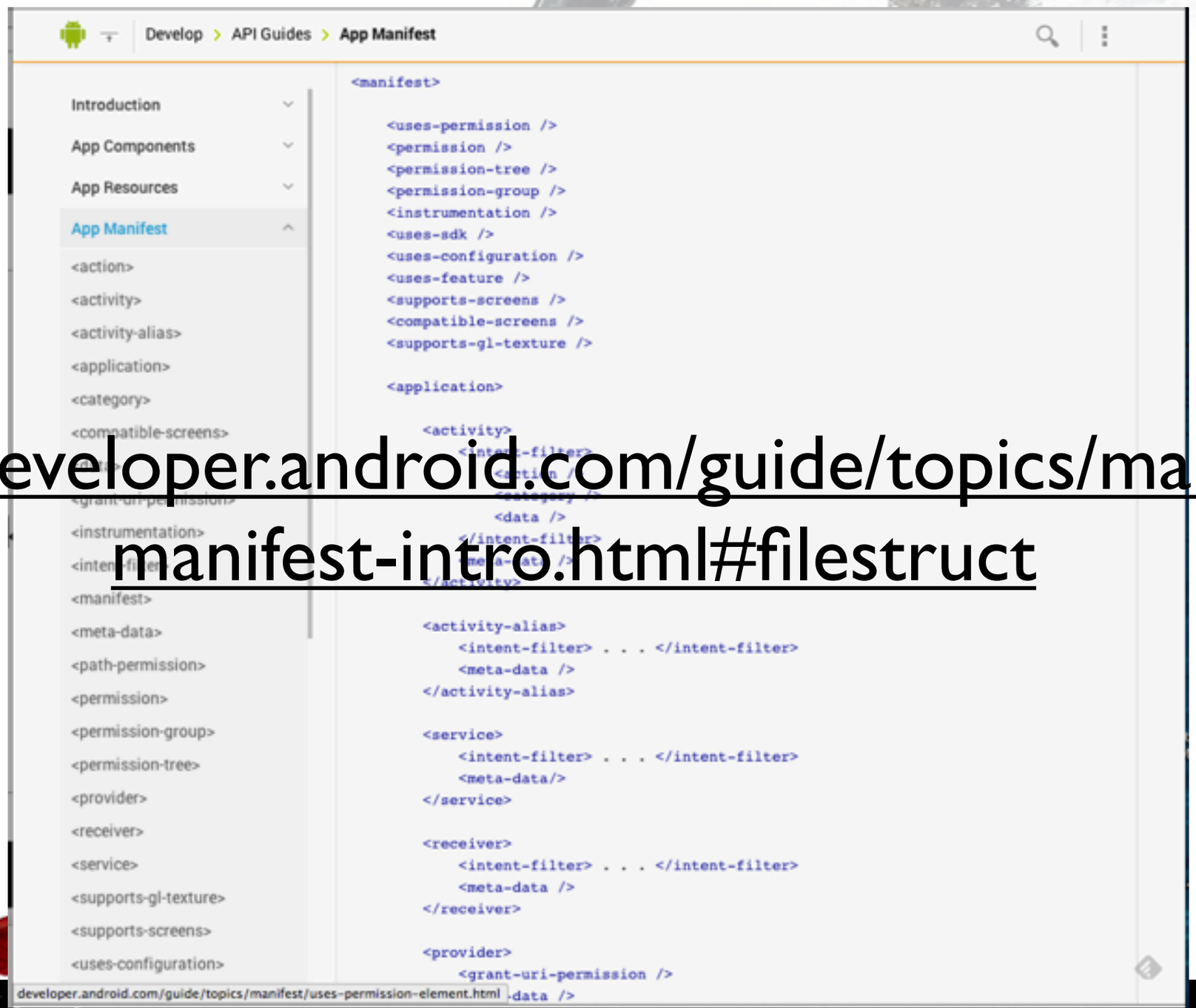
# Android Manifest File

# App Manifest

Every application must have an AndroidManifest.xml file (with precisely that name) in its root directory. The manifest file presents essential information about your app to the Android system, information the system must have before it can run any of the app's code.

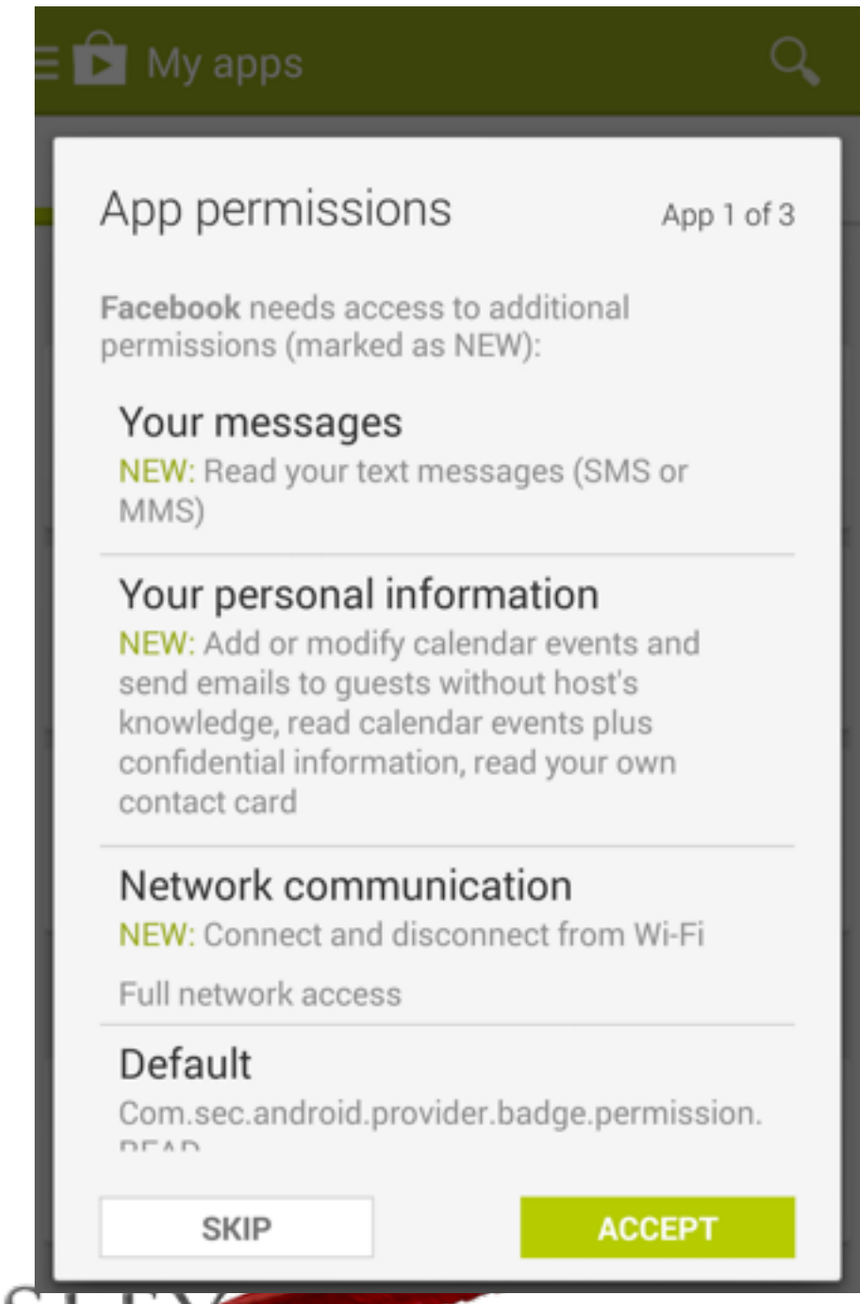
- It names the Java package for the application. The package name serves as a unique identifier for the application.
- It describes the components of the application — the activities, services, broadcast receivers, and content providers that the application is composed of. It names the classes that implement each of the components and publishes their capabilities (for example, which `Intent` messages they can handle). These declarations let the Android system know what the components are and under what conditions they can be launched.
- It determines which processes will host application components.
- It declares which permissions the application must have in order to access protected parts of the API and interact with other applications.
- It also declares the permissions that others are required to have in order to interact with the application's components.
- It lists the `Instrumentation` classes that provide profiling and other information as the application is running. These declarations are present in the manifest only while the application is being developed and tested; they're removed before the application is published.
- It declares the minimum level of the Android API that the application requires.
- It lists the libraries that the application must be linked against.

# App Manifest




http://developer.android.com/guide/topics/manifest/manifest-intro.html#filestruct

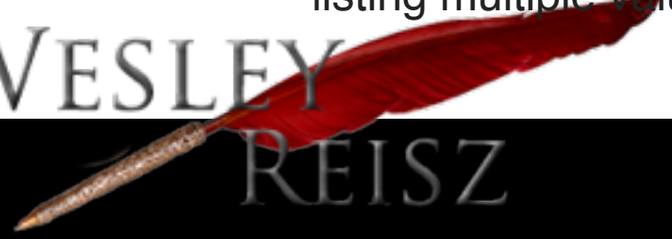
# App Manifest




- http://developer.android.com/reference/android/Manifest.permission.html

# Conventions

- Only the <manifest> and <application> elements are required, they each must be present and can occur only once.
- Most of the others can occur many times or not at all — although at least some of them must be present for the manifest to accomplish anything meaningful.
- If an element contains anything at all, it contains other elements. All values are set through attributes, not as character data within an element.
- Elements at the same level are generally not ordered. For example, <activity>, <provider>, and<service> elements can be intermixed in any sequence. (An <activity-alias> element is the exception to this rule: It must follow the <activity> it is an alias for.)
- Except for some attributes of the root <manifest> element, all attribute names begin with an `android:`prefix — for example, `android:alwaysRetainTaskState`. Because the prefix is universal, the documentation generally omits it when referring to attributes by name.
- As a shorthand for full classpaths, if the first character of the string is a period, the string is appended to the application's package name
- If more than one value can be specified, the element is almost always repeated, rather than listing multiple values within a single element.



# App Manifest

- Review Android Manifests
  - https://github.com/wesreisz/cis490-75/blob/master/week4/RockPaperScissors/app/src/main/AndroidManifest.xml
  - http://theblaine.com/2011/07/sample-androidmanifest-xml-file/

# Simple Data Sharing

# Intents

An intent contains certain header data, e.g., the desired action, the type, etc. Optionally an intent can also contain additional data based on an instance of the `Bundle` class which can be retrieved from the intent via the `getExtras()`method.

You can also add data directly to the `Bundle` via the overloaded `putExtra()` methods of the `Intent` objects. Extras are key/value pairs. The key is always of type `String`. As value you can use the primitive data types (`int`, `float`, ...) plus objects of type `String`, `Bundle`, `Parceable` and `Serializable`.

The receiving component can access this information via the `getAction()` and `getData()` methods on the `Intent`object. This `Intent` object can be retrieved via the `getIntent()` method.

The component which receives the intent can use the `getIntent().getExtras()` method call to get the extra data. That is demonstrated in the following code snippet.

# Intents

```
@Override
public void onClick(View view) {
    Intent intent = new Intent(this,WinLoseActivity.class);
    intent.putExtra(RockPaperScissorsUtil.INPUT_TYPE,view.getId());
    startActivity(intent);
}
```

# Intents

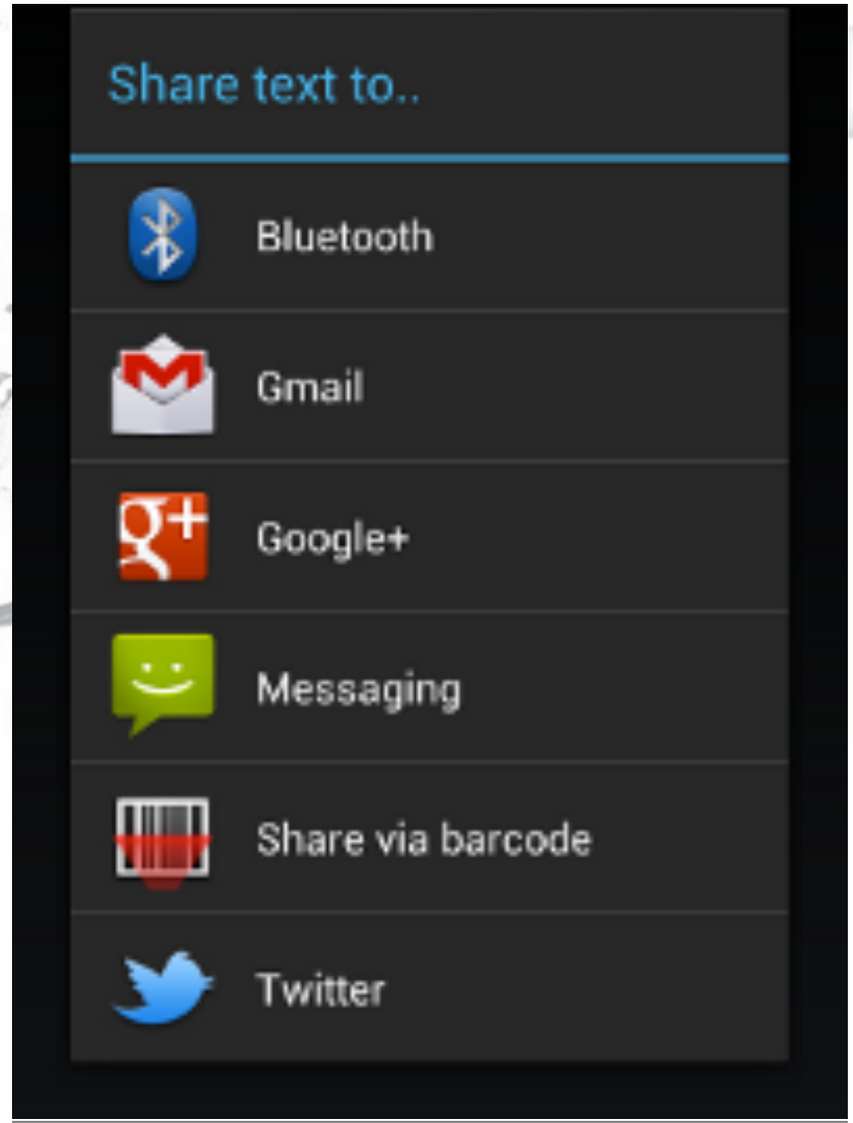
## Intent Resolution

There are two primary forms of intents you will use.

- **Explicit Intents** have specified a component (via `setComponent(ComponentName)` or `setClass(Context, Class)`), which provides the exact class to be run. Often these will not include any other information, simply being a way for an application to launch various internal activities it has as the user interacts with the application.
- **Implicit Intents** have not specified a component; instead, they must include enough information for the system to determine which of the available components is best to run for that intent.

# Demo

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT,
    "This is my text to send.");
sendIntent.setType("text/plain");
startActivity(sendIntent);
```




http://developer.android.com/training/sharing/send.html#send-text-content

# Implicit Intent Examples

http://developer.android.com/training/basics/intents/sending.html

# Start an Activity with the Intent

Once you have created your `Intent` and set the extra info, call `startActivity()` to send it to the system. If the system identifies more than one activity that can handle the intent, it displays a dialog for the user to select which app to use, as shown in figure 1. If there is only one activity that handles the intent, the system immediately starts it.


**Figure 1.** Example of the selection dialog that appears when more than one app can handle an intent.

```
startActivity(intent);
```

Here's a complete example that shows how to create an intent to view a map, verify that an app exists to handle the intent, then start it:

```
// Build the intent
Uri location = Uri.parse("geo:0,0?q=1600+Amphitheatre+Parkway,+Mountain+View,+California"
Intent mapIntent = new Intent(Intent.ACTION_VIEW, location);

// Verify it resolves
PackageManager packageManager = getPackageManager();
List<ResolveInfo> activities = packageManager.queryIntentActivities(mapIntent, 0);
boolean isIntentSafe = activities.size() > 0;

// Start an activity if it's safe
if (isIntentSafe) {
    startActivity(mapIntent);
}
```

http://developer.android.com/training/basics/intents/sending.html#StartActivity

# Intent For Result

## Start the Activity

There's nothing special about the `Intent` object you use when starting an activity for a result, but you do need to pass an additional integer argument to the `startActivityForResult()` method.

The integer argument is a "request code" that identifies your request. When you receive the result `Intent`, the callback provides the same request code so that your app can properly identify the result and determine how to handle it.

For example, here's how to start an activity that allows the user to pick a contact:

```
static final int PICK_CONTACT_REQUEST = 1;  // The request code
...
private void pickContact() {
    Intent pickContactIntent = new Intent(Intent.ACTION_PICK, Uri.parse("content://contac
    pickContactIntent.setType(Phone.CONTENT_TYPE); // Show user only contacts w/ phone nu
    startActivityForResult(pickContactIntent, PICK_CONTACT_REQUEST);
}
```

http://developer.android.com/training/basics/intents/result.html#StartActivity

# Intent For Result

## Receive the Result

When the user is done with the subsequent activity and returns, the system calls your activity's `onActivityResult()` method. This method includes three arguments:

- The request code you passed to `startActivityForResult()`.
- A result code specified by the second activity. This is either `RESULT_OK` if the operation was successful or `RESULT_CANCELED` if the user backed out or the operation failed for some reason.
- An `Intent` that carries the result data.

For example, here's how you can handle the result for the "pick a contact" intent:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // Check which request we're responding to
    if (requestCode == PICK_CONTACT_REQUEST) {
        // Make sure the request was successful
        if (resultCode == RESULT_OK) {
            // The user picked a contact.
            // The Intent's data Uri identifies which contact was selected.

            // Do something with the contact here (bigger example below)
        }
    }
}
```

In this example, the result `Intent` returned by Android's Contacts or People app provides a content `Uri` that identifies the contact the user selected.

In order to successfully handle the result, you must understand what the format of the result `Intent` will be. Doing so is easy when the activity returning a result is one of your own activities. Apps included with the Android platform offer their own APIs that you can count on for specific result data. For instance, the People app (Contacts app on some older versions) always returns a result with the content URI that identifies the selected contact, and the Camera app returns a `Bitmap` in the `"data"` extra (see the class about Capturing Photos).



http://developer.android.com/training/basics/intents/result.html#ReceiveResult

# Demo: Taking a Picture

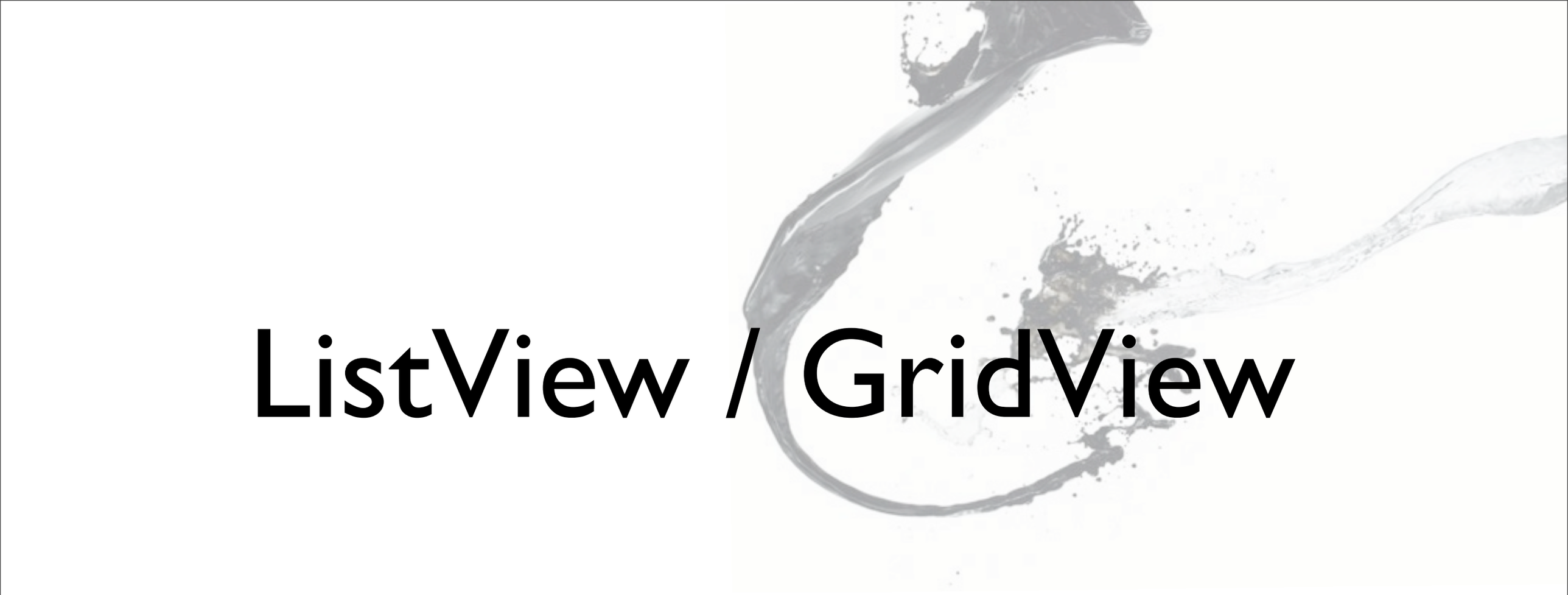
1. reiszwt@li168-101:/tmp (less)

```
commit bc738f0c8c18f0508b3c6c05242ce1659b006beb
Author: Wesley Reisz <wes@wesleyreisz.com>
Date:   Mon Oct 13 12:34:54 2014 -0400

    added landscape

commit 74af6204d36b6658226fa3b91963f2a8439aa917
Author: Wesley Reisz <wes@wesleyreisz.com>
Date:   Mon Oct 13 12:31:36 2014 -0400

    displayed bitmap from camera result

commit 1011a2091a77293be23006392f52725d3a7bb18d
Author: Wesley Reisz <wes@wesleyreisz.com>
Date:   Mon Oct 13 12:04:36 2014 -0400

    dispatch to camera to get picture

commit 049942786f435314f0112c6519b6439daa8b32f4
Author: Wesley Reisz <wes@wesleyreisz.com>
Date:   Mon Oct 13 12:00:19 2014 -0400

    added camera layout activity
```

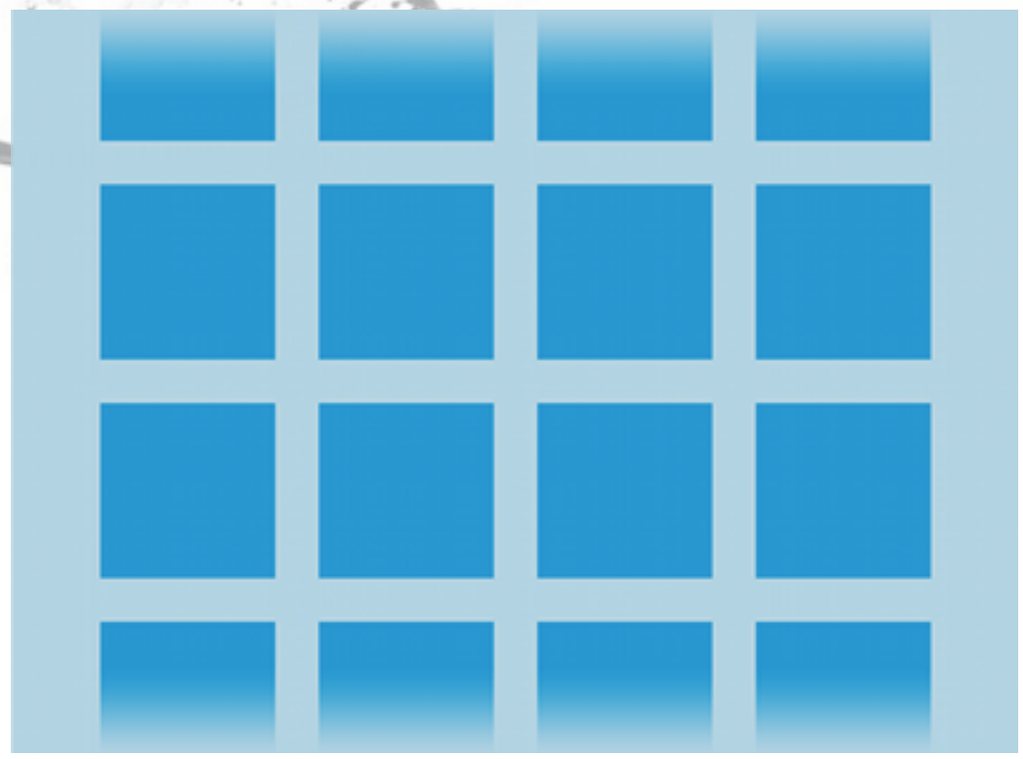
```java
private void dispatchTakePictureIntent() {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);
    }
}
```

```java
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bundle extras = data.getExtras();
        Bitmap imageBitmap = (Bitmap) extras.get("data");
        mImageView.setImageBitmap(imageBitmap);
    }
}
```

# ListView / GridView



# ListView / GridView



WESLEY REISZ

# ListView / GridView

Adapters are basically used to deliver content. One adapter you probably have in every application is the CursorAdapter which enables you to deliver content given by a cursor from a database query. A ListView has nearly always some sort of Adapter.

Basic Example:
http://www.javacodegeeks.com/2013/06/android-listview-tutorial-and-basic-example.html

This is a part of the android OS. Here is the actual version of the defined XML file.

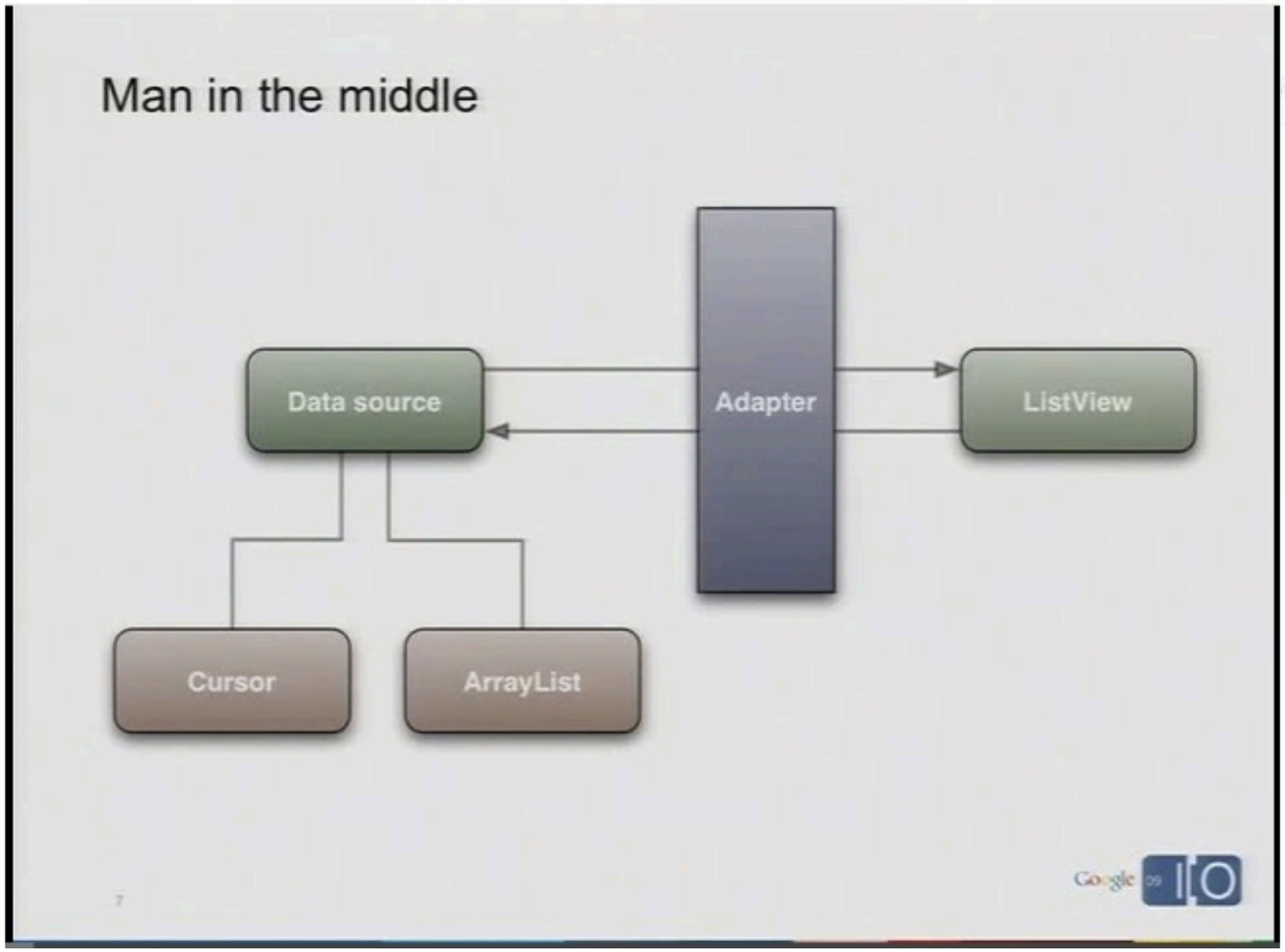
simple_list_item_1:

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@android:id/text1"
        style="?android:attr/listItemFirstLineStyle"
        android:paddingTop="2dip"
        android:paddingBottom="3dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

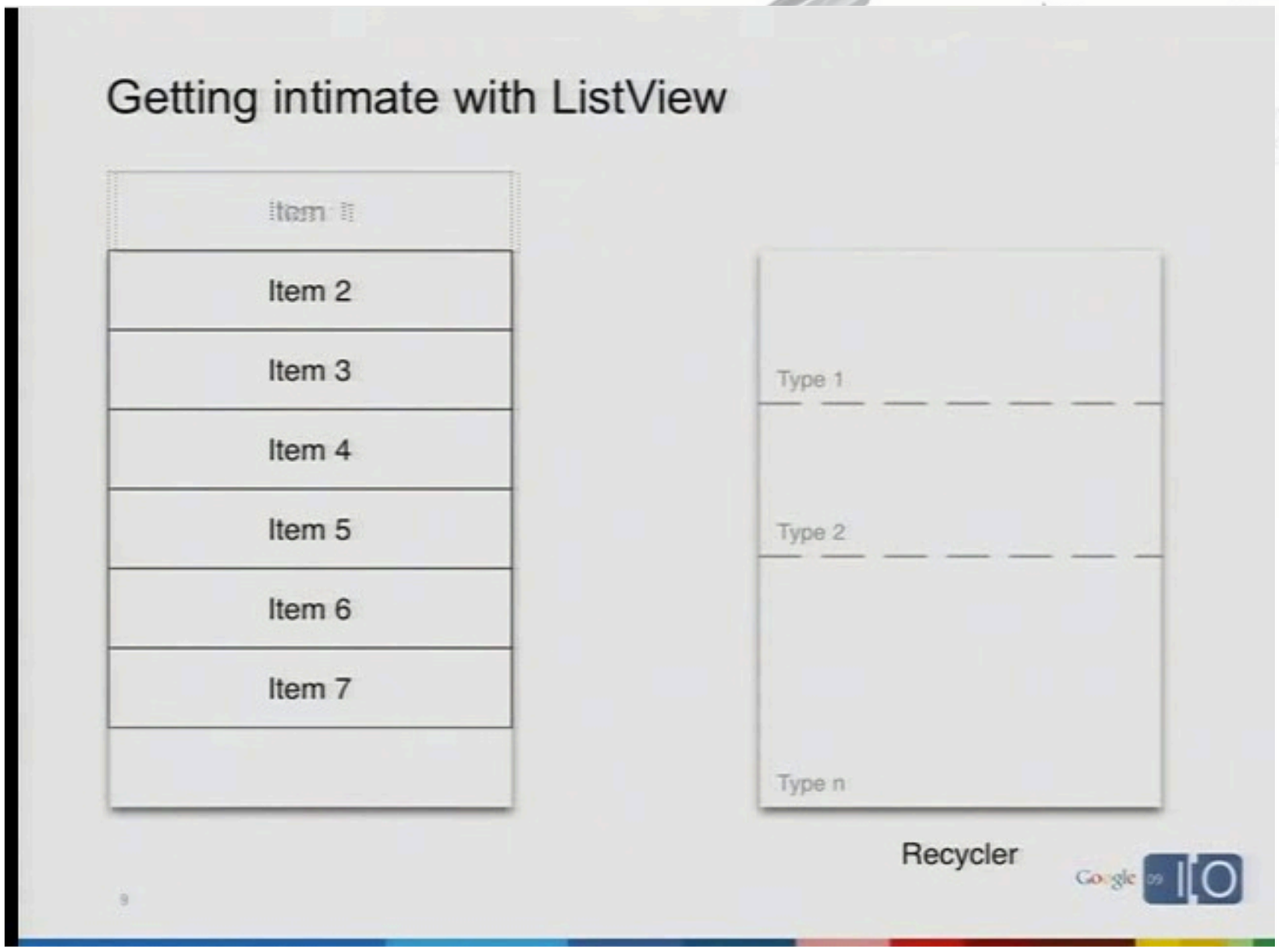
simple_list_item_2:

```
    <TextView android:id="@android:id/text1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            style="?android:attr/listItemFirstLineStyle"/>

    <TextView android:id="@android:id/text2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@android:id/text1"
            style="?android:attr/listItemSecondLineStyle" />
```

# ListView / GridView



# ListView / GridView



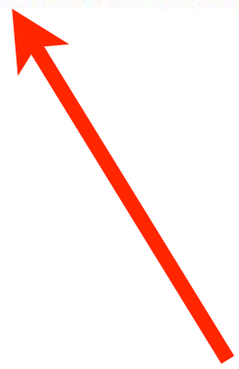
```java
package com.wesleyreisz.mymusiclist;

import ...


public class MusicListActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_music_list);

        ListView lv = (ListView)findViewById(R.id.listViewSongs);
        List<Song> songs = new MusicService().findAll();
        SongAdapter songAdapter = new SongAdapter(this, R.layout.activity_music_list,songs);
        lv.setAdapter(songAdapter);
    }


    @Override
```

```java
package com.wesleyreisz.mymusiclist;

import ...

public class MusicListActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_music_list);

        ListView lv = (ListView)findViewById(R.id.listViewSongs);
        List<Song> songs = new MusicService().findAll();
        SongAdapter songAdapter = new SongAdapter(this, R.layout.activity_music_list,songs);
        lv.setAdapter(songAdapter);
    }


    @Override
```

MusicListActivity.java | SongAdapter.java | listview_for_each_song.xml | activity_music_list.xml | MusicService.java | Song.java

```java
package com.wesleyreisz.mymusiclist;

import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.List;

/**
 * Created by wesleyreisz on 10/13/14.
 */
public class MusicService {
    private List<Song> songs;
    {
        songs = new ArrayList<Song>();
        songs.add(new Song("I'm Yours","Jason Mraz","We Sing, We Dance, we Steal Things",getDate(2008, 05, 15)));
        songs.add(new Song("Kryptonite","3 Doors Down","The Better Life",getDate(2001, 01, 17)));
        songs.add(new Song("Timber","Pit Bull","The Better Life",getDate(2013, 10, 7)));
        songs.add(new Song("Dark Horse","Katy Perry","Single",getDate(2013, 12, 17)));
        songs.add(new Song("Counting Stars","One Republic","Single",getDate(2013, 6, 14)));
        songs.add(new Song("Demons","Imagine Dragons","Single",getDate(2013, 10, 22)));
        songs.add(new Song("Drink A Beer","Luke Bryan","Crash My Party",getDate(2013, 11, 11)));
        songs.add(new Song("Burn","Ellie Goulding ","Halcyon",getDate(2013, 03, 13)));
        songs.add(new Song("Story Of My Life","One Direction ","Story Of My Life",getDate(2013, 11, 25)));
        songs.add(new Song("Let Her Go","Passenger","All the Little Lights",getDate(2012, 7, 12)));
    }

    public List<Song> findAll(){
        return songs;
    }

    public Song findOne(String name){
        for(Song song:songs){
            if(song.getName().equals(name)){
                return song;
            }
        }
        return new Song();
    }

    private static Date getDate(int year, int month, int day) {
        Calendar c = Calendar.getInstance();
        c.set(year,month,day,0,0);
        return c.getTime();
    }

}
```

```java
package com.wesleyreisz.mymusiclist;

import ...


public class MusicListActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_music_list);

        ListView lv = (ListView)findViewById(R.id.listViewSongs);
        List<Song> songs = new MusicService().findAll();
        SongAdapter songAdapter = new SongAdapter(this, R.layout.activity_music_list,songs);
        lv.setAdapter(songAdapter);
    }


    @Override
```



```java
/**
 * Created by wesleyreisz on 10/13/14.
 */
public class SongAdapter extends ArrayAdapter<Song>{
    private SimpleDateFormat df = new SimpleDateFormat("MMM d, yyyy (EEE");
    private Context mContext;
    private List<Song>mSongs;

    public SongAdapter(Context context, int textViewResourceId, List<Song> songs) {
        super(context, textViewResourceId, songs);
        this.mContext = context;
        this.mSongs = songs;

    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        if(convertView==null){
            LayoutInflater inflater = (LayoutInflater)mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            convertView = inflater.inflate(R.layout.listview_for_each_song, parent, false);
        }

        final Song song = mSongs.get(position);

        TextView textViewTitle = (TextView)convertView.findViewById(R.id.textViewSongTitle);
        textViewTitle.setText(song.getName() + " (" + song.getArtist() + ")");

        TextView textViewAlbum = (TextView)convertView.findViewById(R.id.textViewSongArtist);
        textViewAlbum.setText(song.getAlbum());

        TextView textViewPublishedDate = (TextView)convertView.findViewById(R.id.textViewSongDate);
        textViewPublishedDate.setText(df.format(song.getPublishedDate()));

        return convertView;
    }
}
```

# ListView / GridView

Don't

```
public View getView(int position, View convertView, ViewGroup parent) {
    View item = mInflater.inflate(R.layout.list_item_icon_text, null);

    ((TextView) item.findViewById(R.id.text)).setText(DATA[position]);
    ((ImageView) item.findViewById(R.id.icon)).setImageBitmap(
            (position & 1) == 1 ? mIcon1 : mIcon2);

    return item;
}
```

10

Do

```
public View getView(int position, View convertView, ViewGroup parent) {
    if (convertView == null) {
        convertView = mInflater.inflate(R.layout.item, null);
    }

    ((TextView) convertView.findViewById(R.id.text)).setText(DATA[position]);
    ((ImageView) convertView.findViewById(R.id.icon)).setImageBitmap(
            (position & 1) == 1 ? mIcon1 : mIcon2);

    return convertView;
}
```

11

SongAdapter.java | listview_for_each_song.xml | activity_music_list.xml

```xml
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="64dp"
    android:paddingRight="64dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".MusicListActivity">


    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/listViewSongs"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true" />
</RelativeLayout>
```

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

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/textViewSongTitle"
        android:textSize="16dp"
        android:textColor="#686868"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="TextView"
        android:padding="5dp"
        android:paddingLeft="10dp"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <TextView
            android:id="@+id/textViewSongArtist"
            android:textSize="12dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView"
            android:textStyle="italic"
            android:paddingLeft="20dp"
            />

        <TextView
            android:id="@+id/textViewSongDate"
            android:textSize="12dp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="TextView"
            android:textStyle="italic"
            android:paddingLeft="12dp"
            android:paddingBottom="10dp"
            android:layout_weight="1"/>
    </LinearLayout>


</LinearLayout>
```

WESLEY REISZ

# ListView

- Lab2: My Music List Video 4 of 9
  https://www.youtube.com/watch?v=BvTUGCZ-LMY

- Lab2: My Music List Video 5 of 9
  https://www.youtube.com/watch?v=UpEbO5YUmWk

# Agenda

- Review Test
- Android Manifest
- Simple Data Sharing
- ListView / GridView