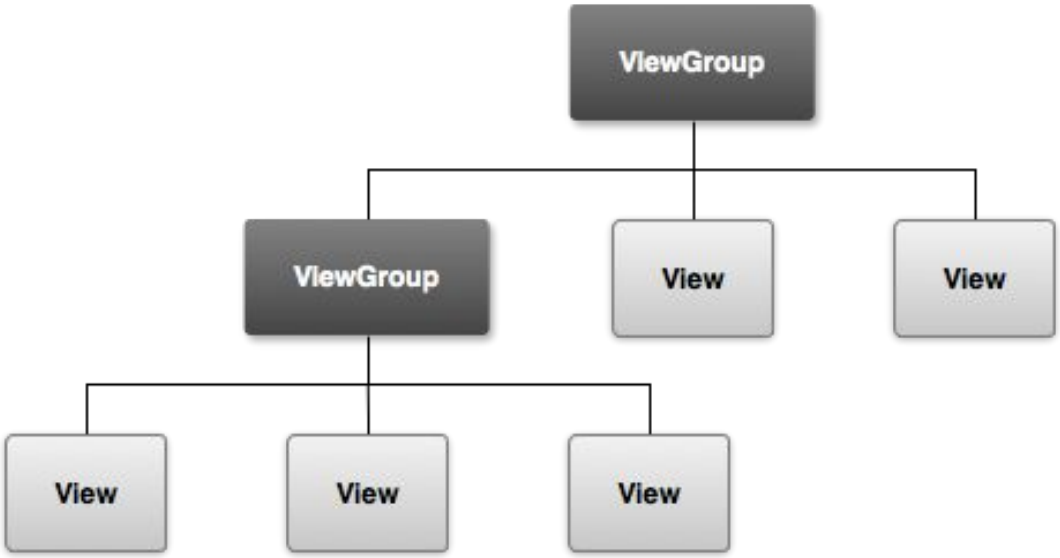
2017 Android of WMNTAA

User Interface

User Interface

The user interface for each component of your app is defined using a hierarchy of View and ViewGroup objects. Each view group is an invisible container that organizes child views, while the child views may be input controls or other widgets that draw some part of the UI.



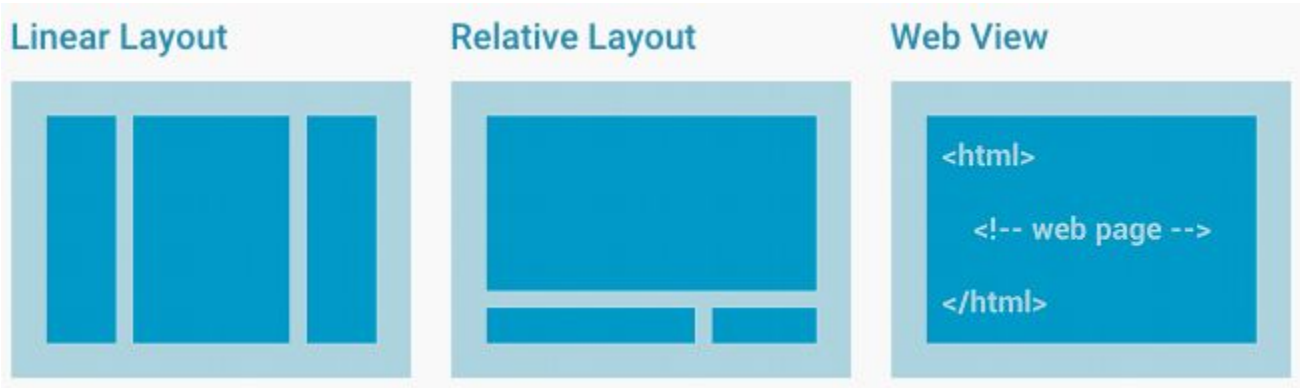
Layout

A layout defines the visual structure for a user interface, such as the UI for an activity or app widget.

Android provides a straightforward XML vocabulary that corresponds to the View classes and subclasses, such as those for widgets and layouts.

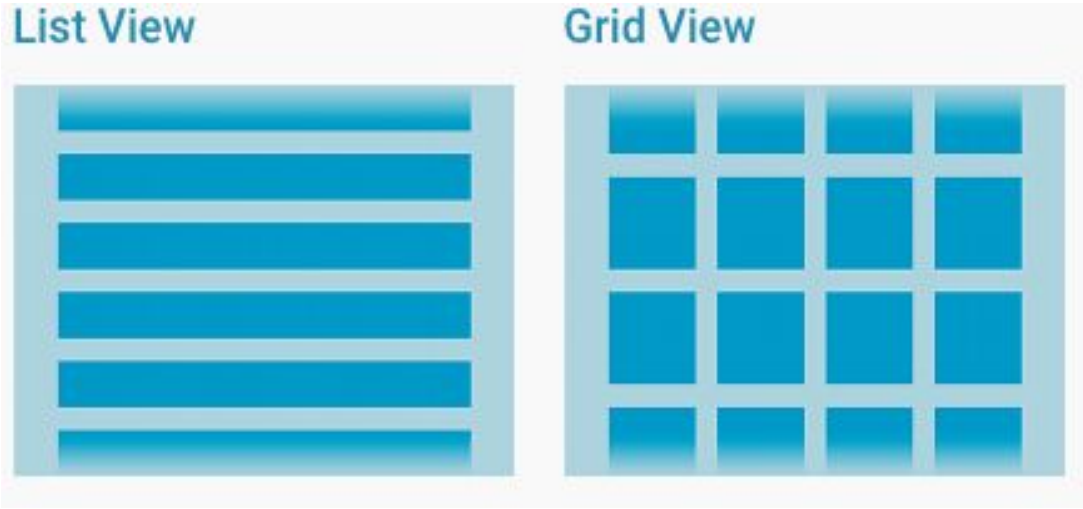
Common Layouts

- Linear Layout: A layout that organizes its children into a single horizontal or vertical row
- Relative Layout: Enables you to specify the location of child objects relative to each other
- Webview: particularly for web pages



Layouts with an Adapter

- Listview: Displays a scrolling single column list
- Gridview: Displays a scrolling multiple columns and rows



Create a Button

Declare a Button in layout file

```
<Button
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:text="@string/self_destruct"
    android:onClick="selfDestruct" />
```

Create a Button

Set Button attributes in onCreate()

```
public class MyActivity extends Activity {
    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        setContentView(R.layout.content_layout_id);

        final Button button = (Button) findViewById(R.id.button_id);
        button.setOnClickListener(new View.OnClickListener() {
            public void onClick(View v) {
                // Perform action on click
            }
        });
    }
}
```

List View

ListView is a view group that displays a list of scrollable items. The list items are automatically inserted to the list using an Adapter that pulls content from a source such as an array or database query and converts each item result into a view that's placed into the list.

Filling ListView With Data

- Binding a ListView instance to an Adapter so that it can retrieve data from an external source and creates a View that represents each data entry
  - ArrayAdapter: Use this adapter when your data source is an array
  - SimpleCursorAdapter: Use this adapter when your data comes from a Cursor

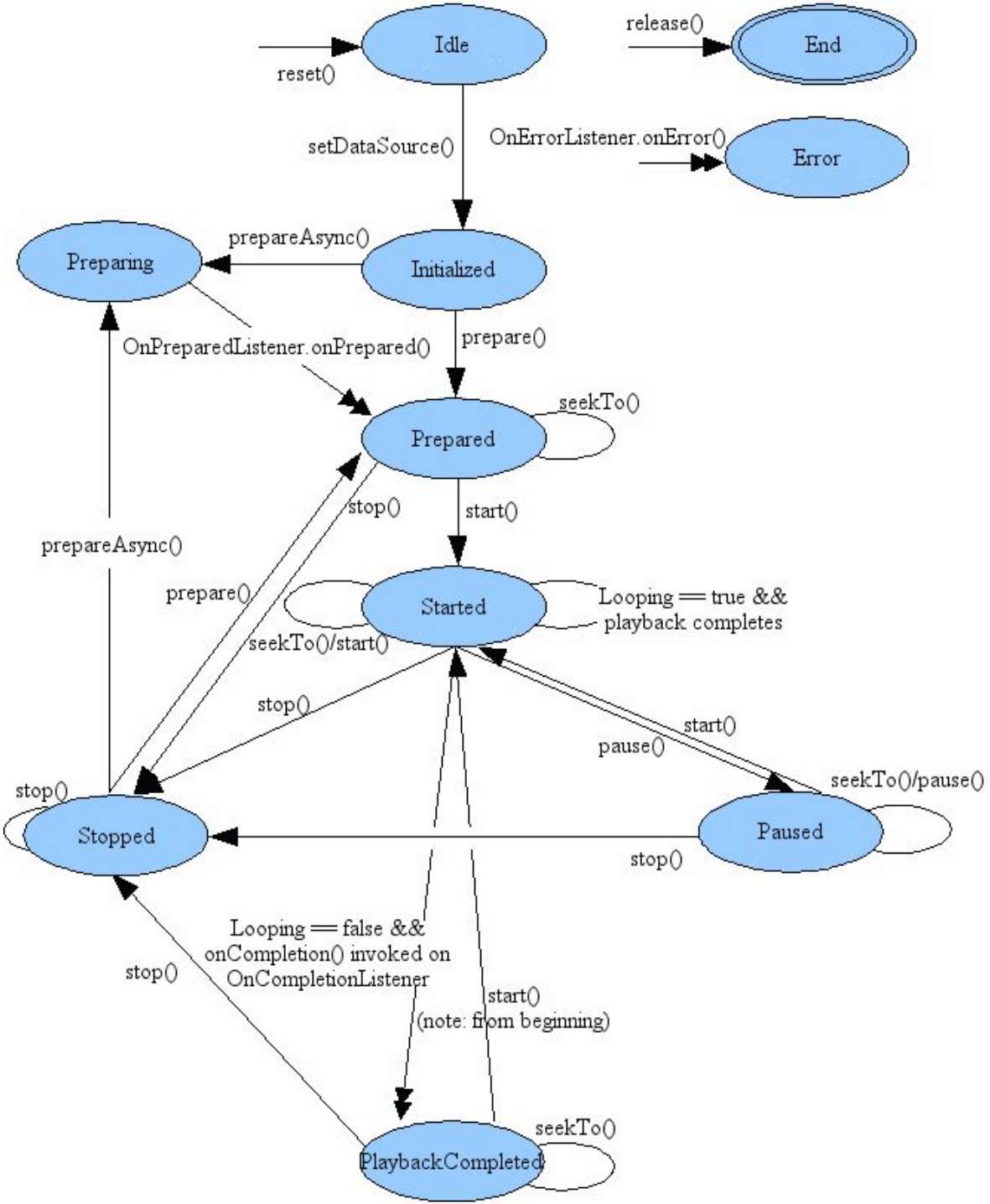
ListView with ArrayAdapter

- For example, if you have an array of strings you want to display in a ListView, initialize a new ArrayAdapter to specify the layout for the string array

```
// Create an empty adapter we will use to display the loaded data.
// We pass null for the cursor, then update it in onLoadFinished()
mAdapter = new SimpleCursorAdapter(this,
        android.R.layout.simple_list_item_1, null,
        fromColumns, toViews, 0);
setListAdapter(mAdapter);
```

Media Player

The Android multimedia framework includes support for playing variety of common media types, so that you can easily integrate audio, video and images into your applications. You can play audio or video from media files stored in your application's resources (raw resources), from standalone files in the filesystem, or from a data stream arriving over a network connection, all using MediaPlayer APIs.


Idle
reset()
release()
End
setDataSource()
OnErrorListener.onError()
Error
Preparing
prepareAsync()
Initialized
OnPreparedListener.onPrepared()
prepare()
seekTo()
Prepared
stop()
start()
prepareAsync()
prepare()
Started
Looping == true &&
playback completes
seekTo()/start()
stop()
start()
pause()
seekTo()/pause()
stop()
Stopped
Paused
stop()
Looping == false &&
onCompletion() invoked on
OnCompletionListener
stop()
start()
(note: from beginning)
seekTo()
PlaybackCompleted

Using MediaPlayer

Here is an example of how to play audio that's available as a local raw resource (saved in your application's res/raw/ directory):

```
MediaPlayer mediaPlayer = MediaPlayer.create(context, R.raw.sound_file_1);
mediaPlayer.start(); // no need to call prepare(); create() does that for you
```

Using MediaPlayer

And here is how you might play from a URI available locally in the system (that you obtained through a Content Resolver, for instance)

```
Uri myUri = ....; // initialize Uri here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(getApplicationContext(), myUri);
mediaPlayer.prepare();
mediaPlayer.start();
```

Using MediaPlayer

Playing from a remote URL via HTTP streaming looks like this

```
String url = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(url);
mediaPlayer.prepare(); // might take long! (for buffering, etc)
mediaPlayer.start();
```

Hands-on Exercise

Write a App which contain a button, playing a audio file with URI while clicked. If you click the button while playing, you should stop the media player first, and replay from head.

What's URI ? : https://developer.android.com/reference/java/net/URI.html

Challenge

Use VideoView and MediaPlayer to play a video file.

Example :

https://github.com/tanchihpin0517/HelloWorld