http://www.android.com/

Using files

TelephonyManager

Services

Record Camera and Video

Media Player/Recorder and TTS/STT

Data storage overview

- Assets (assets) and Resources (res/raw)
  - Private data installed with the application (**read-only**)
  - Use the **android.content.res.xxx** classes/methods (Resources and AssetManager)
- Shared Preferences
  - Private primitive application data in key-value pairs
- Internal storage
  - Private data on the device memory (files)
- External storage
  - Public data on the shared external storage (files), **corrected in API-19**
  - http://www.androidcentral.com/kitkat-sdcard-changes?pg=2
- SQLite Databases
  - Structured data in a private databases
- Network Connection
  - Data on the web; read write to server (e.g. a DB)
- When app is uninstalled or ”resetted” all private data is removed

Reading files from resources

- **Raw files** are processed by aapt and must be referenced from the application using a resource identifier in the R class
- **Asset files** are compiled into the .apk file as-is and you can navigate this directory in the same way as a typical file system

```
private static CharSequence readResourceOrAsset(Activity activity /* Context ctx */) {
   BufferedReader reader = null;
   try {
      reader = new BufferedReader(new InputStreamReader(activity.getAssets().open("EULA.txt")));
reader = new BufferedReader(new InputStreamReader(activity.getResources().openRawResource(R.raw.instructions)));
      String line;
      StringBuilder buffer = new StringBuilder();
      while ((line = reader.readLine()) != null)
            buffer.append(line).append('\n');
      return buffer;
   } catch (IOException e) {
            return "";
   } finally {
      closeStream(reader);
   }
}

private static void closeStream(Closeable stream) {
      if (stream != null) {
            try {
                  stream.close();
            } catch (IOException e) {
                  // Ignore
            }
      }
}
```

Files and SD Cards

- All of the usual Java file I/O routines from the **java.io** package are available (Output/InputStreamReader, FileReader/Writer etc.)
- Every application can read and write from/to the SD card (external memory) but needs permissions in AndroidManifest
  - Write needs the WRITE_EXTERNAL_STORAGE permission
  - Read needs the READ_EXTERNAL_STORAGE permission
- Some helper methods are provided by the Context class for the applications **private** internal storage area (and private external area?)
  - Only internal access to **/data/data/<packagename>/files** subdirectory with perhaps limited storage

Helper methods for the private storage area

```
// Returns the absolute path to the private directory
Context.getFilesDir();
// Create or access a directory
Context.getDir(String name, int mode);
// Delete a private file. Returns true if it worked, false otherwise
Context.deleteFile(String name);
// Return a list of all files in the application's private area in a String array
Context.fileList();
// Open a private file for reading. Returns a java.io.FileInputStream
Context.openFileInput(String name);
// Open a private file for writing. Returns a java.io.FileOutputStream
Context.openFileOutput(String name, int mode);
```

Internal read/write file example

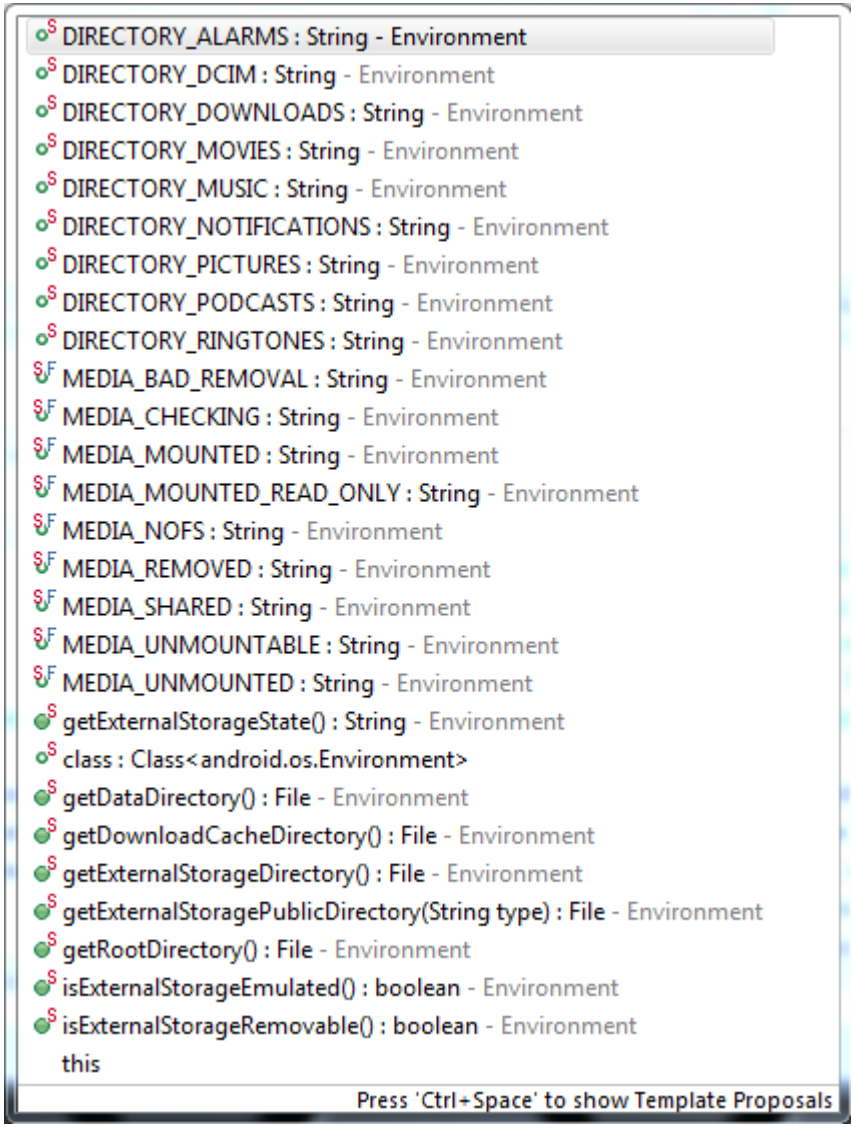
- The permissions are
  - MODE_PRIVATE - No access for other applications
  - MODE_WORLD_READABLE - Read access for other applications
  - MODE_WORLD_WRITABLE - Write access for other applications
  - MODE_WORLD_READABLE | MODE_WORLD_WRITABLE - Read / Write access

```
private void writeFileToInternalStorage() {
       String eol = System.getProperty("line.separator");
       BufferedWriter writer = null;
       try {
          writer = new BufferedWriter(
                    new OutputStreamWriter(
                      openFileOutput("myfile",
                      MODE_WORLD_WRITEABLE)));
          writer.write("This is a test1." + eol);
          writer.write("This is a test2." + eol);
       } catch (Exception e) {
          e.printStackTrace();
       }
       finally {
          if (writer != null) {
             try {
                    writer.close();
              } catch (IOException e) {
                    e.printStackTrace();
              }
          }
     }
}
```

```
private void readFileFromInternalStorage() {
       String eol = System.getProperty("line.separator");
       BufferedReader input = null;
       try {
              input = new BufferedReader(
                           new InputStreamReader(
                             openFileInput("myfile")));
              String line;
              StringBuffer buffer = new StringBuffer();
              while ((line = input.readLine()) != null) {
                    buffer.append(line + eol);
              }
       } catch (Exception e) {
              e.printStackTrace();
       } finally {
              if (input != null) {
                    try {
                        input.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
              }
       }
}
```

External file (Environment)

- getExternalStorageState()
  - Use with MEDIA_*** to get the state of the external storage system (it can be unmounted, removed, unformatted etc.)
- getExternalStoragePublicDirectory()
  - Use with DIRECTORY_*** to get the standard directory locations
- getExternalStorageDirectory()
  - Returns external root folder - on emulator /storage/sdcard
- isExternalStorage***()



External read file example

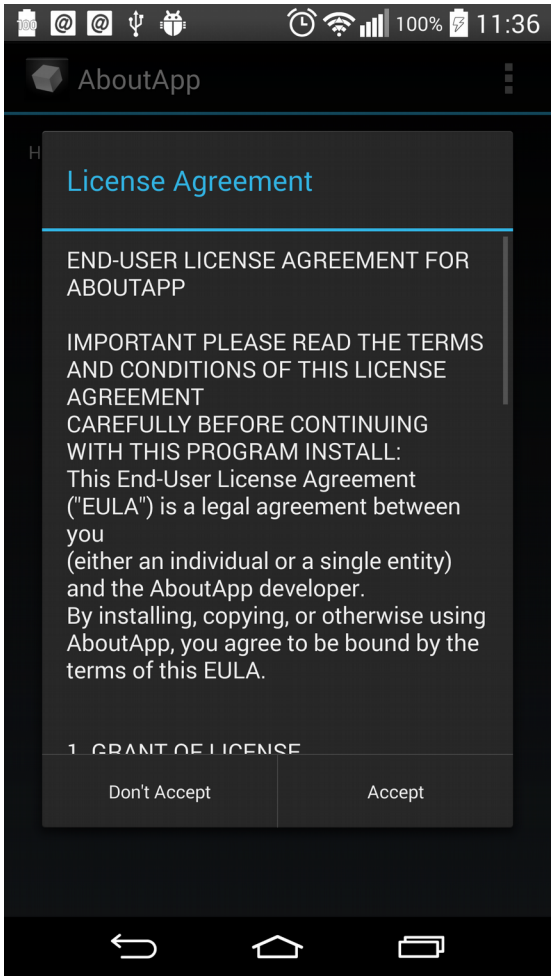
- Via the following method call you can check the state of the external storage system (it can be unmounted, removed, unformatted etc.)
  - Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)

```
private void readFileFromSDCard() {
      File directory = Environment.getExternalStorageDirectory();
      // Assumes that a file article.rss is available on the external storage
      File file = new File(directory + "/article.rss");
      if (!file.exists()) {
            throw new RuntimeException("File not found");
      }
      Log.d("Testing", "Starting to read");
      BufferedReader reader = null;
      try {
            reader = new BufferedReader(new FileReader(file));
            StringBuilder builder = new StringBuilder();
            String line;
            while ((line = reader.readLine()) != null) {
                  builder.append(line);
            }
      } catch (Exception e) {
            e.printStackTrace();
      } finally {
            if (reader != null) {
                  try {
                        reader.close();
                  } catch (IOException e) {
                        e.printStackTrace();
                  }
            }
      }
```

```
External write is similar to
WriteFileToInternalStorage()
Just change the file that is going to be opened
writer =
   new BufferedWriter(new FileWriter(file, append));
```

AboutApp 1

- Example of EULA, about dialog and recent changes
  - Reading files with html tags from res/raw



See the AboutApp example

AboutApp 2

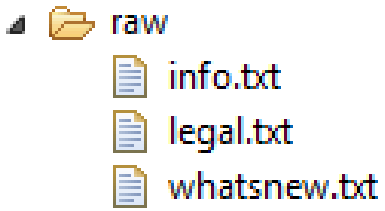
```
public class MainActivity extends Activity {

	private static final boolean LIGHT_THEME = false;

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		// must read theme from settings then recreate(); if it should propagate at once after change
		if(LIGHT_THEME)
			setTheme(R.style.CustomThemeLight);
		else
			setTheme(R.style.CustomThemeDark);

		super.onCreate(savedInstanceState);

		// http://stackoverflow.com/questions/9286822/how-to-force-use-of-overflow-menu-on-devices-with-menu-button
		// in combination with: http://www.intertech.com/Blog/Post/Android-Action-Bar-from-the-Options-Menu.aspx
		try {
			ViewConfiguration config = ViewConfiguration.get(this);
			Field menuKeyField = ViewConfiguration.class.getDeclaredField("sHasPermanentMenuKey");
			if(menuKeyField != null) {
				menuKeyField.setAccessible(true);
				menuKeyField.setBoolean(config, false);
			}
		}
		catch (Exception ex) {
			// Ignore
		}

		setContentView(R.layout.activity_main);
		// Set up the action bar to show a drop down list
		final ActionBar actionBar = getActionBar();
		actionBar.setDisplayShowTitleEnabled(true);
		actionBar.setTitle(R.string.app_name);

		// only shows once
		Eula.show(this);
	}


	@Override
	public boolean onCreateOptionsMenu(Menu menu) {
		// Inflate the menu; this adds items to the
		// action bar if it is present.
		getMenuInflater().inflate(R.menu.main, menu);
		return true;
	}
```

```
<!-- added in res/values/styles.xml -->
<style name="CustomThemeLight" parent="android:Theme.Holo.Light">
</style>
<style name="CustomThemeDark" parent="android:Theme.Holo">
</style>
```

```
	@Override
	public boolean onOptionsItemSelected(MenuItem item) {
		// Handle action bar item clicks here. The action bar will
		// automatically handle clicks on the Home/Up button, so long
		// as you specify a parent activity in AndroidManifest.xml.
		int id = item.getItemId();
		if (id == R.id.action_about) {
			AboutDialog about = new AboutDialog(this);
			about.setTitle(getResources().getString(R.string.app_name)
			+ "  v" + getVersionName());
			about.show();
			return true;
		}
		return super.onOptionsItemSelected(item);
	}

}
```

AboutApp 3

```
public class AboutDialog extends Dialog{

	private static Context mContext = null;

	public AboutDialog(Context context) {
		super(context);
		mContext = context;
	}

	/**	* http://www.techrepublic.com/blog/software-engineer/a-reusable-about-dialog-for-your-android-apps/
	* http://commonsware.com/blog/Android/2010/05/26/html-tags-supported-by-textview.html	*/
	@Override
	public void onCreate(Bundle savedInstanceState) {
		setContentView(R.layout.about);

		TextView tv = (TextView) findViewById(R.id.legal_text);
		Spanned ss = Html.fromHtml(readRawTextFile(R.raw.legal));
		tv.setText(ss);
		Linkify.addLinks(tv, Linkify.ALL);

		tv = (TextView) findViewById(R.id.info_text);
		ss = Html.fromHtml(readRawTextFile(R.raw.info));
		tv.setText(ss);
		Linkify.addLinks(tv, Linkify.ALL);

		tv = (TextView) findViewById(R.id.whatsnew_text);
		ss = Html.fromHtml(readRawTextFile(R.raw.whatsnew));
		tv.setText(ss);
		Linkify.addLinks(tv, Linkify.WEB_URLS);
	}
```

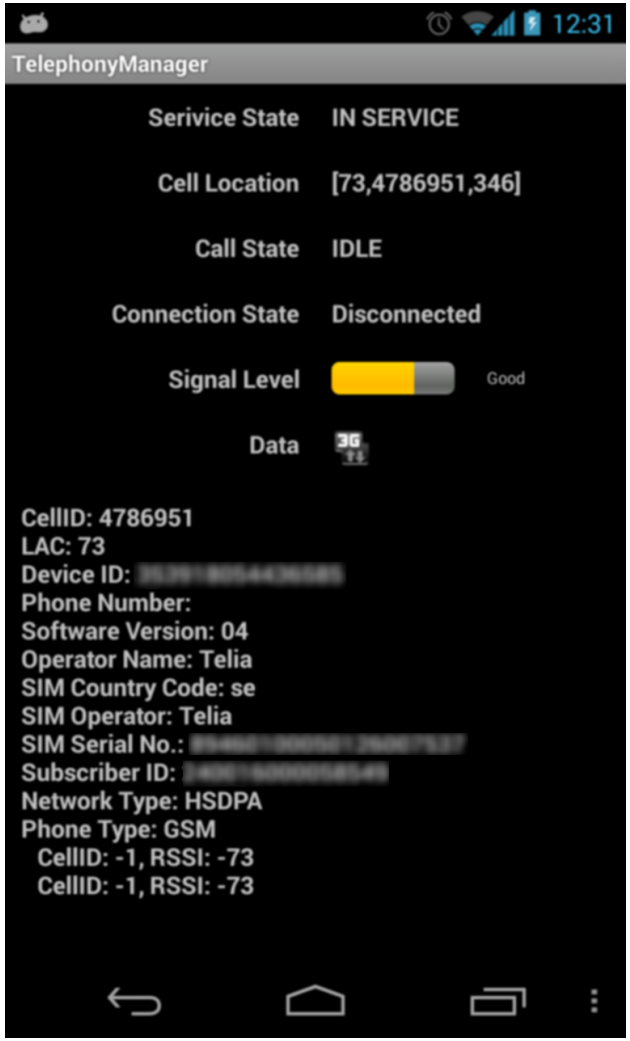



```
public static String readRawTextFile(int id) {
    InputStream inputStream = mContext.getResources().openRawResource(id);
    InputStreamReader in = new InputStreamReader(inputStream);
    BufferedReader buf = new BufferedReader(in);
    String line;
    StringBuilder text = new StringBuilder();
    try {
       while (( line = buf.readLine()) != null) text.append(line);
     } catch (IOException e) {
        return null;
     }
     return text.toString();
 }
```

TelephonyManager

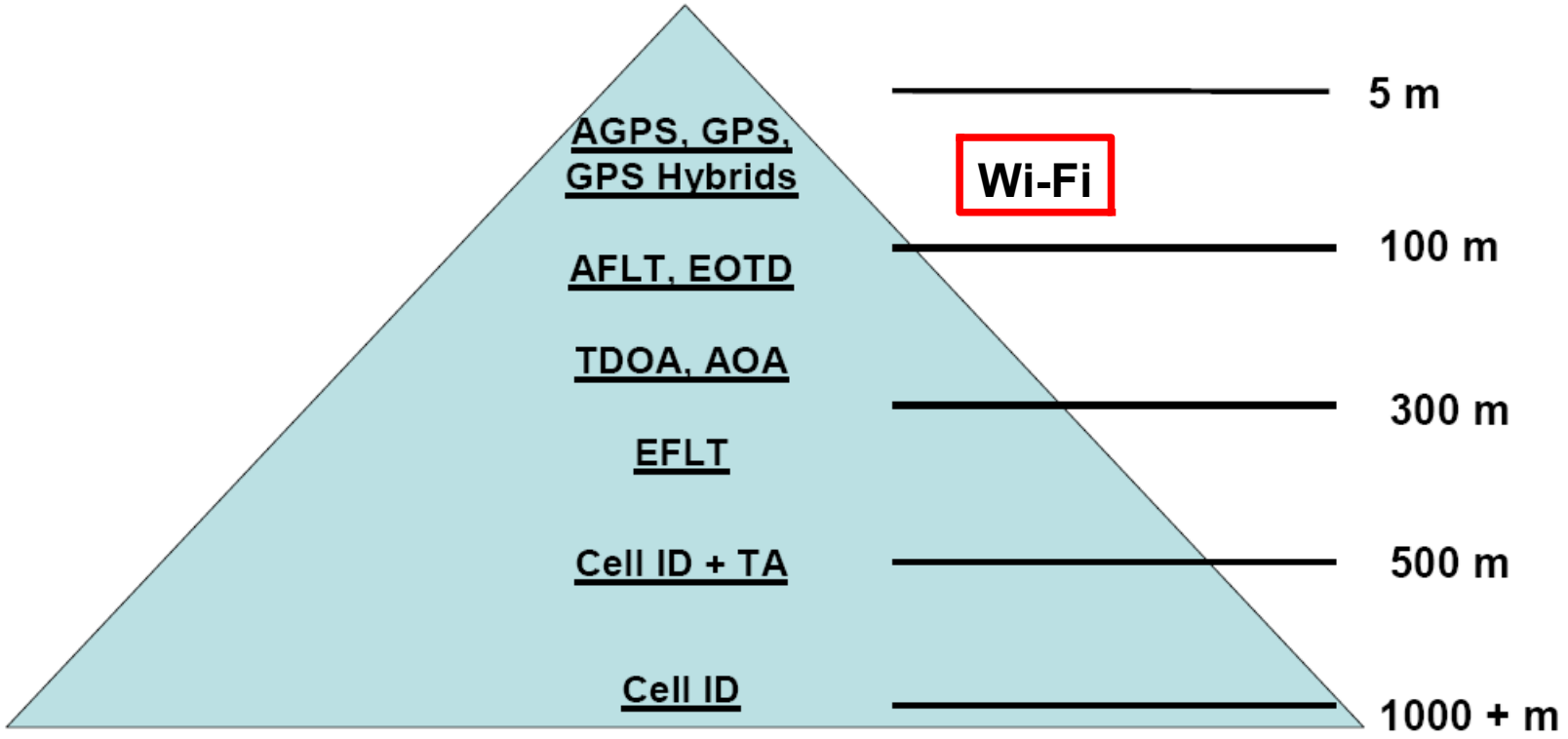
- The Android TelephonyManager can be used to obtain different statistics about network status and telephone information
  - See the PhoneStateSampleActivity

```
private void startSignalLevelListener() {
	TelephonyManager tm =
		(TelephonyManager) getSystemService(TELEPHONY_SERVICE);
	int events = PhoneStateListener.LISTEN_SIGNAL_STRENGTHS
			| PhoneStateListener.LISTEN_DATA_ACTIVITY
			| PhoneStateListener.LISTEN_CELL_LOCATION
			| PhoneStateListener.LISTEN_CALL_STATE
			| PhoneStateListener.LISTEN_CALL_FORWARDING_INDICATOR
			| PhoneStateListener.LISTEN_DATA_CONNECTION_STATE
			| PhoneStateListener.LISTEN_MESSAGE_WAITING_INDICATOR
			| PhoneStateListener.LISTEN_SERVICE_STATE;
	tm.listen(phoneStateListener, events);
}
private final PhoneStateListener phoneStateListener = new PhoneStateListener()
{
	@Override
	public void onCallStateChanged(int state, String incomingNumber) {
		// TODO Auto-generated method stub
	@Override
	public void onCellLocationChanged(CellLocation location) {
		// TODO Auto-generated method stub
	@Override
	public void onDataActivity(int direction) {
		// TODO Auto-generated method stub
	@Override
	public void onDataConnectionStateChanged(int state) {
		// TODO Auto-generated method stub
	@Override
	public void onSignalStrengthsChanged(SignalStrength signalStrength) {
		// TODO Auto-generated method stub
```



Cell ID with TelephonyManager

Network-based, handset-based, and hybrid solutions are available



The Android telephony API provides a way to monitor basic phone information, such as the network type, connection state, and utilities for manipulating phone number strings

```
TelephonyManager tm = (TelephonyManager) getSystemService(Context.TELEPHONY_SERVICE);
String networkOperator = tm.getNetworkOperator(); // obtain the device's MCC and MNC
GsmCellLocation location = (GsmCellLocation) tm.getCellLocation();
int cellID = location.getCid(); // obtain the device's cell ID and LAC (Location Area Code)
int lac = location.getLac();    // and LAC (Location Area Code)
```

FileWriter (chars) example

- CellIDWriter class that writes the Cell ID, Location Area Code, Mobile Country Code and Mobile Network Code to a file on the SD card

```
public class CellIDWriter implements Runnable
{
    private static boolean isWriteRunning = false;
    private static File uri = Environment.getExternalStorageDirectory();
    private final String fname = "/cellid.txt";
    private String mCID, mLAC, mMCC, mMNC;

    public void run()
    {
        try{
            FileWriter fw = new FileWriter(uri.toString() + fname, true);
            StringBuilder sb = new StringBuilder();
            Date dateTime = new Date(System.currentTimeMillis());
            sb.append(mCID + ";" + mLAC + ";" + mMCC + ";" + mMNC + ";" + dateTime.toString() + "\r\n");
            Log.v("CellID", sb.toString());

            // append at end of file
            fw.append(sb.toString());
            // android does flush() in close()
            fw.close();
        }
        catch(Exception e){
            System.err.println(e.toString() +
            " : " + e.getMessage());
            e.printStackTrace();
        }
        finally{
            isWriteRunning = false;
        }
    }
```

```
// call the class with something like
// and it will do the work in a thread
new CellIDWriter().write(String.valueOf(mcellid),
                         String.valueOf(mlac), "240");
```

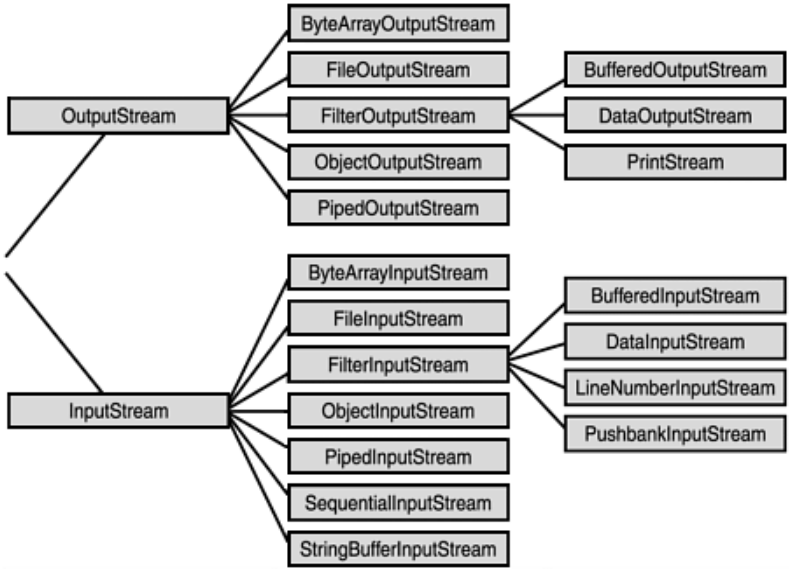
```
    // CellIDWriter class continues here
    public void write(String CID, String LAC, String MCC,
      String MNC){
        mCID = CID; mLAC = LAC; mMCC = MCC, mMNC = MNC;
        if(!isWriteRunning){
            isWriteRunning = true;
            new Thread(this).start();
        }
    }
}
```

```
// synchronized alternative
private static final Object mObjLock = new Object();
synchronized(mObjLock){
      // threaded code which access shared resources
}
```

File I/O (byte) streams

- FileInputStream, FileOutputStream, InputStream and OutputStream

```
// Writing to a File
DataOutputStream dos = new DataOutputStream(openFileOutput("file.dat", MODE_PRIVATE));
dos.writeUTF("Hello, I'm text in a file!")

// Reading from a File
try {
     DataInputStream dis = new DataInputStream(openFileInput("file.dat"));
     dis.readUTF(); //should return "Hello, I'm text in a file!" if the file has been written.
} catch (FileNotFoundException ex) {
     //the file has not yet been written
}

// Example of how to serialize an object to a File:
FileOutputStream fos = context.openFileOutput(SAVENAME,
     context.MODE_PRIVATE);
ObjectOutputStream oos = new ObjectOutputStream(fos);
oos.writeObject(this);
oos.close();

// How to inflate the object back from the file:
FileInputStream fis = context.openFileInput(SAVENAME);
ObjectInputStream ois = new ObjectInputStream(is);
Flow f = (Flow) ois.readObject();

// The files in the resources directories can also be opened. For example,
// to open myrawfile.txt located in the res/raw folder, use the following:
InputStream is = this.getResources().openRawResource(R.raw.myrawfile);
```



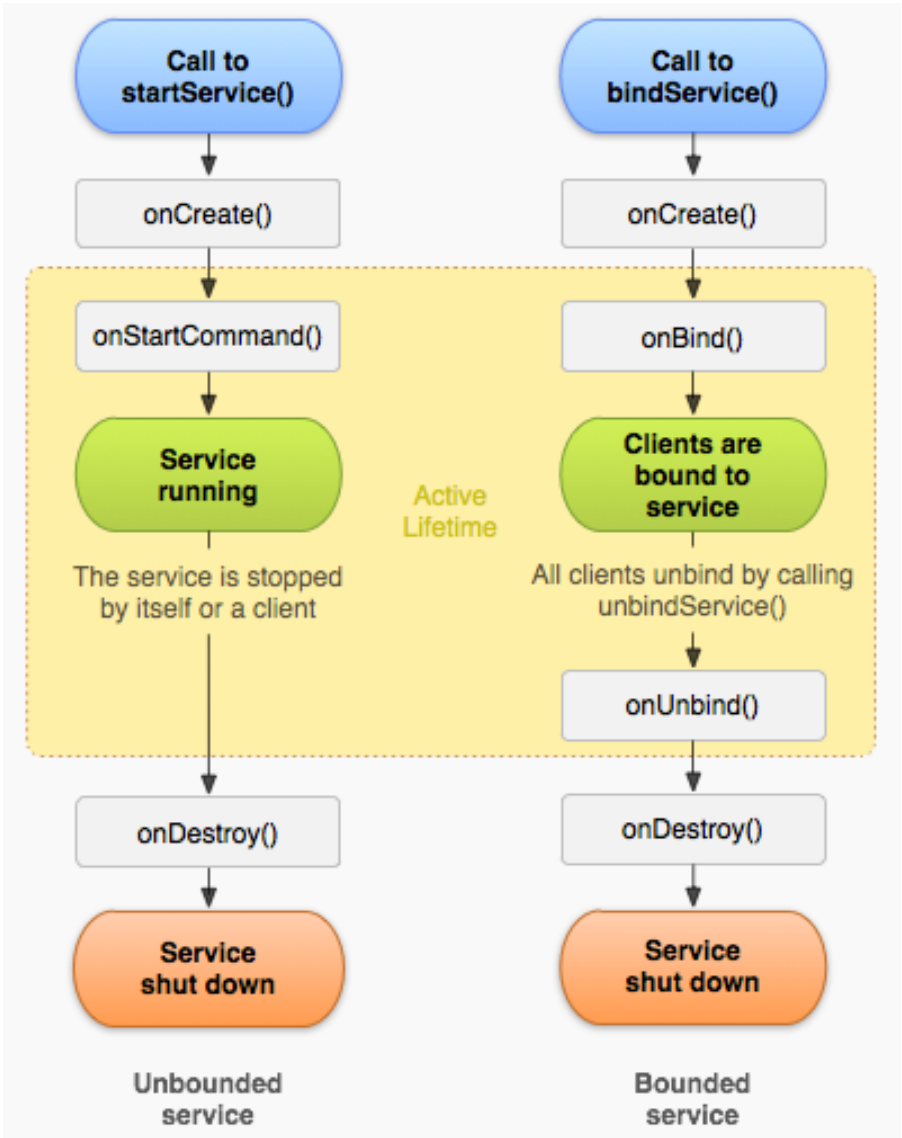
Service 1

- A service is an Android component that runs in the background without any user interface
- When Android needs to kill applications to save resources, Services has high priority, only second to foreground Activities
- Other components can start and stop the service, it can also stop itself
- While it is running other components can bind to it (if implemented)
- Consider using a Service when the application
  - performs a lengthy or intensive processing, not requiring user interaction
  - performs certain tasks at regular intervals, e.g. downloading updates of some content
  - performs lengthy operations that shouldn't be canceled if the application exits, e.g. downloading a large file
  - needs to provide data or information services to other applications (without an user interface)

Service 2

- To create a service in your application
  - you must create a subclass of Service (or one of its existing subclasses, e.g. IntentService)
  - you need to override some callback methods that handle key aspects of the service lifecycle and provide a mechanism for components to bind to the service (if appropriate)
  - each service class must have a corresponding <service> declaration in the application's manifest file
- A service runs in the main thread of its hosting process
  - Any CPU intensive work or blocking operations **should** be performed by spawning a worker thread!
  - The IntentService however has its own thread where it schedules the work to be done

Service lifecycle



- A service can essentially take two forms
  - Started (or unbound) - when an application component starts it by calling **startService()** - runs until it stops itself with stopSelf() or another component stops it by calling **stopService()**
  - Bound - when an application component binds to it by calling **bindService()** - the service runs until the last client component is unbound from it

Creating a simple service

- Declare the service in your Androidmanifest inside the application tag

```
<service android:name=".SimpleService"></service>
```

- Create the source file with a class that extends Service and override at least the **onCreate()** and **onDestroy()** methods
  - onCreate() - perform one-time setup procedures
  - OnStartCommand – called by the system when startService() is called
  - onDestroy() - clean up resources, threads, listeners, receivers, etc.
- Also override the **onBind()** method for cases when a new component binds to this service after it have been created
- Start, stop or bind to the service from an external component

```
startService(new Intent(LocationTracker.this, SimpleService.class));
stopService(new Intent(LocationTracker.this, SimpleService.class));
bindService(Intent service, ServiceConnection conn, int flags);
```

- The service will not stop when the activity is destroyed, paused or the screen orientation is changed

Simple service

```
public class SimpleService extends Service
{
	int[] notes =
		{R.raw.c5, R.raw.b4, R.raw.a4, R.raw.g4};
	int NOTE_DURATION = 500; //millisec
	MediaPlayer m_mediaPlayer;
	boolean paused = false;

	@Override
	public IBinder onBind(Intent intent) {
		return null;
	}

	@Override
	public void onCreate() {
		super.onCreate();
		Toast.makeText(this,"Service created ...",
			Toast.LENGTH_LONG).show();
		paused = false;
		Thread initBkgdThread =
			new Thread(new Runnable() {
			public void run() {
				play_music();
			}
		});
		initBkgdThread.start();
	}

	@Override
	public void onDestroy() {
		super.onDestroy();
		Toast.makeText(this, "Service destroyed ...",
			Toast.LENGTH_LONG).show();
		paused = true;
	}
```

```
// SimpleService class continue here

	private void play_music()
	{
		int i=0;
		for(int i=0; i<120; i++) {
	// check to ensure main activity not paused or destroyed
			if(!paused)
			{
				if(m_mediaPlayer != null)
					m_mediaPlayer.release();

				m_mediaPlayer =
				MediaPlayer.create(this, notes[i%4]);
				m_mediaPlayer.start();

				try {
					Thread.sleep(NOTE_DURATION);
				}
				catch (InterruptedException e) {
					e.printStackTrace();
				}
				i++;
			}
		}
	}
} // end SimpleService class
```

http://developer.android.com/reference/android/app/Service.html

URL above got more advanced examples and information how to bind to running services etc.

IntentService example

- Subclass of Service that uses a worker thread to handle all start requests, one at a time
  - This is the best option if you don't require that your service handles multiple requests simultaneously (most services do not)
  - All you need to do is implement **onHandleIntent()**, which receives the intent for each start request so it can do the background work

```
public class HelloIntentService extends IntentService {
// A constructor is required, and must call the super IntentService(String) constructor with a name for the worker thread
// every other callback is fixed by the default implementation – but its possible to override if super.xxx() is called
      public HelloIntentService() {                                    // xxx = corresponding lifecycle method
            super("HelloIntentService");
      }
      // The IntentService calls this method from the default worker thread with the intent that started
      // the service. When this method returns, IntentService stops the service, as appropriate
      @Override
      protected void onHandleIntent(Intent intent) {
            // Normally we would do some work here, like download a file.
            // For our sample, we just sleep for 5 seconds.
            long endTime = System.currentTimeMillis() + 5*1000;
            while (System.currentTimeMillis() < endTime) {
                  synchronized (this) {
                        try {
                              wait(endTime - System.currentTimeMillis());
                        } catch (Exception e) {
                        }
                  }
            }
      }
}
```

See the example NotificationActivityService

Service example

Performs same work as previous IntentService

```
public class HelloService extends Service {
  private Looper mServiceLooper;
  private ServiceHandler mServiceHandler;

  // Handler that receives messages from the thread
  private final class ServiceHandler extends Handler {
    public ServiceHandler(Looper looper) {
      super(looper);
    }

    @Override
    public void handleMessage(Message msg) {
// Normally we would do some work here, like download a
// file. For our sample, we just sleep for 5 seconds.
      long endTime = System.currentTimeMillis() + 5*1000;
      while (System.currentTimeMillis() < endTime) {
        synchronized (this) {
          try {
            wait(endTime - System.currentTimeMillis());
          } catch (Exception e) {
          }
        }
      }
// Stop the service using the startId, so that we don't
// stop the service in the middle of handling another job
      stopSelf(msg.arg1);
    }
  } // end ServiceHandler class

  @Override
  public void onDestroy() {
    Toast.makeText(this, "service done",
      Toast.LENGTH_SHORT).show();
  }
// developer.android.com/guide/components/services.html
```

```
  @Override
  public void onCreate() {
// Start up the thread running the service. Note that we create
// a separate thread because the service normally runs in the
// process's main thread, which we don't want to block. We also
// make it background priority so CPU-intensive work will not
// disrupt our UI.
    HandlerThread thread = new HandlerThread("ServiceStartArgs",
      Process.THREAD_PRIORITY_BACKGROUND);
    thread.start();
    // Get the HandlerThread's Looper and use it for our Handler
    mServiceLooper = thread.getLooper();
    mServiceHandler = new ServiceHandler(mServiceLooper);
  }

  @Override
  public int onStartCommand(Intent intent, int flags,
    int startId) {
    Toast.makeText(this, "service starting",
      Toast.LENGTH_SHORT).show();
// For each start request, send a message to start a job and
// deliver the start ID so we know which request we're stopping
// when we finish the job
    Message msg = mServiceHandler.obtainMessage();
    msg.arg1 = startId;
    mServiceHandler.sendMessage(msg);
    // If we get killed, after returning from here, restart
    return START_STICKY;
  }

  @Override
  public IBinder onBind(Intent intent) {
    // We don't provide binding, so return null
    return null;
  }
}
```

Bound Service example 1

- When a component is bound to a Service it maintains a reference to the Service instance (some overrides are missing)

```
public class LocalService extends Service {
   // Binder given to clients
   private final IBinder mBinder = new LocalBinder();
   // Random number generator
   private final Random mGenerator = new Random();

   // Class used for the client Binder. Because we know this service always
   // runs in the same process as its clients, we don't need to deal with IPC
   public class LocalBinder extends Binder {
      LocalService getService() {
         // Return this instance of LocalService so clients can call public methods
         return LocalService.this;
      }
   }

   @Override
   public IBinder onBind(Intent intent) {
      return mBinder;
   }

   // method for clients
   public int getRandomNumber() {
      return mGenerator.nextInt(100);
   }
}  // http://developer.android.com/guide/components/bound-services.html
```

Bound Service example 2

- The client component need to implement a ServiceConnection

```
public class BindingActivity extends Activity
{
  LocalService mService;
  boolean mBound = false;

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

  @Override
  protected void onStart() {
    super.onStart();
    // Bind to LocalService
    Intent intent = new Intent(this,
      LocalService.class);
    bindService(intent, mConnection,
      Context.BIND_AUTO_CREATE);
  }

  @Override
  protected void onStop() {
    super.onStop();
    // Unbind from the service
    if (mBound) {
      unbindService(mConnection);
      mBound = false;
    }
  }
```

```
// Called when a button is clicked (the button in the layout file
// attaches to this method with the android:onClick attribute)
  public void onButtonClick(View v) {
     if (mBound) {
// Call a method from the LocalService. However, if this call were
// something that might hang, then this request should occur in a
// separate thread to avoid slowing down the activity performance.
        int num = mService.getRandomNumber();
        Toast.makeText(this, "number: " + num,
            Toast.LENGTH_SHORT).show();
     }
  }

   // Defines callbacks for service binding, passed to bindService()
  private ServiceConnection mConnection = new ServiceConnection()
  {
     @Override
     public void onServiceConnected(ComponentName className,
        IBinder service) {
        // We've bound to LocalService, cast the IBinder and
        // get a LocalService instance
        LocalBinder binder = (LocalBinder) service;
        mService = binder.getService();
        mBound = true;
     }

     @Override
     public void onServiceDisconnected(ComponentName arg0) {
        mBound = false;
     }
   };
}
```

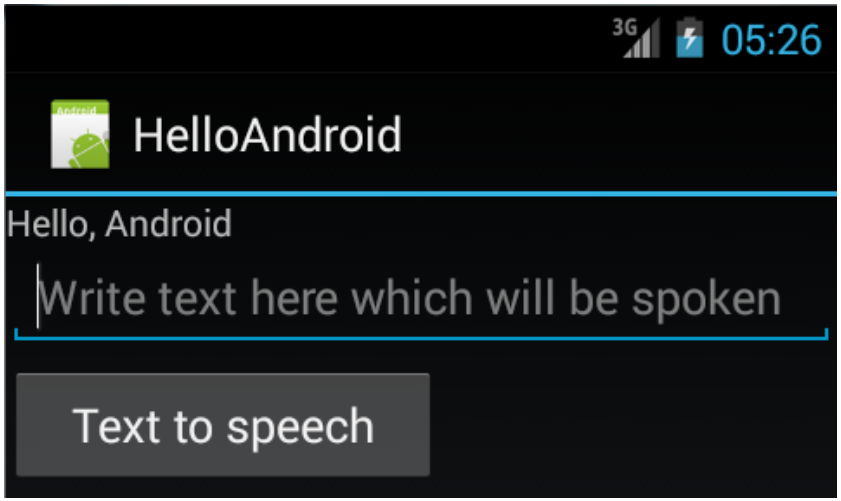
Conclusion

	Thread	AsyncTask	Service	IntentService
When to use ?	- Long task in general. - For tasks in parallel use Multiple threads (traditional mechanisms)	- Long task having to communicate with main thread.	Task with no UI, but shouldn't be too long. Use threads within service for long tasks.	- Long task usually with no communication to main thread - Limitation: tasks executed sequentially
Trigger	Thread start() method	Call to method execute()	Call to method onStartService()	Intent
Triggered From (thread)	Any Thread	Main Thread	Any thread	Main Thread
Runs On	Its own thread	Worker thread. However, Main thread methods may be invoked in between to publish progress.	Main Thread	Separate worker thread is automatically spawned

http://techtej.blogspot.se/2011/03/android-thread-constructspart-4.html

TTS (Text to speech)

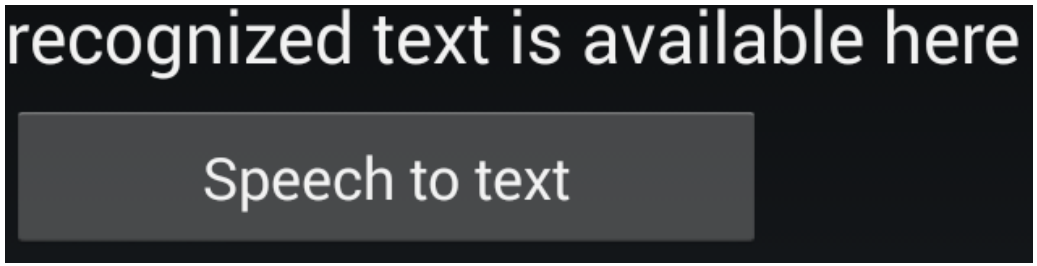
- See the HelloAndroid example
- Send a TextToSpeech.Engine.ACTION_CHECK_TTS_DATA intent to check if TTS is available with the startActivityForResult() method
- In the onActivityResult() method check if the resultcode is equal to the member TextToSpeech.Engine.CHECK_VOICE_DATA_PASS
- If so create the TTS instance **TextToSpeech**(Context this, TextToSpeech.OnInitListener this) which automatically will call the TextToSpeech.OnInitListener method with name **onInit**(int status) when the TextToSpeech engine has initialized
- if status == TextToSpeech.SUCCESS in the onInit() method you can begin use TTS
  - mTts.speak(myText1, TextToSpeech.QUEUE_FLUSH, null);
  - mTts.speak(myText2, TextToSpeech.QUEUE_ADD, null);



STT (Speech to text)

- See the HelloAndroid example as well
- Send a RecognizerIntent.ACTION_RECOGNIZE_SPEECH intent with the startActivityForResult() method (some putExtra is required as LANGUAGE_MODE etc.)
- In the onActivityResult() method check if the resultcode is equal to RESULT_OK
- Get the resulting matching strings and display it in a TextView
  - ArrayList<String> matches = data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);

Note! The recorded speech will be sent to a web service which will do all the recognizing work



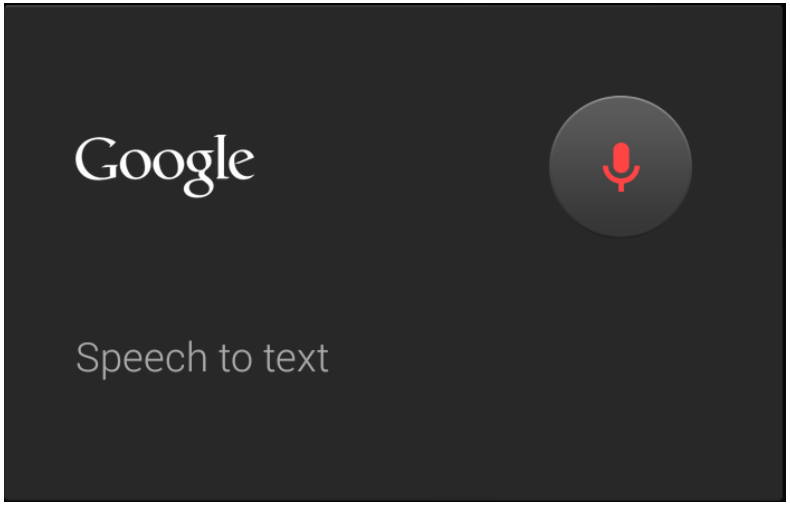
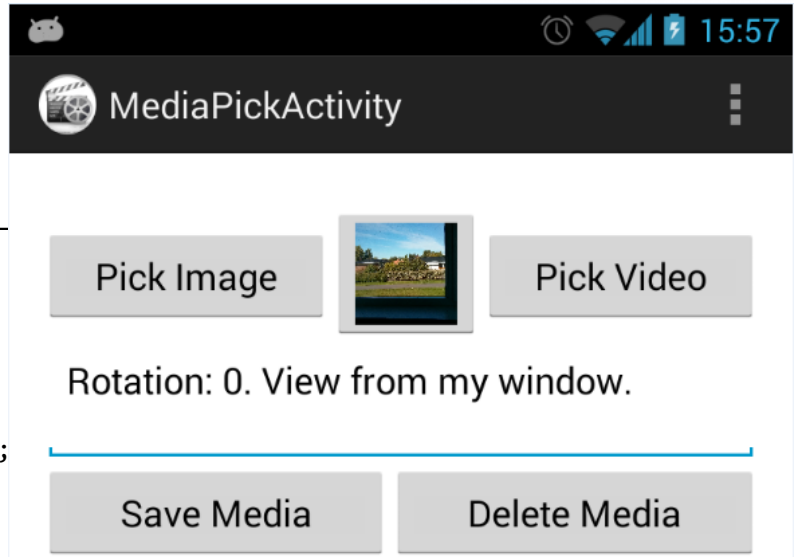


Photo and Video 1

- See the MediaPicker example
- Grabbing photos and videos using the built in media via Intents




```
public void pickImage(View View) {
	Intent cameraIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
	Intent videoIntent = new Intent(MediaStore.ACTION_VIDEO_CAPTURE);
	String dateTime = dateFormat.format(new Date());
	String fname =  dateTime + ".jpg";
	File photo = new File(DIR_TMP, fname);
	cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(photo));
	mMediaUri = Uri.fromFile(photo);
	startActivityForResult(cameraIntent, CAMERA_PIC_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
	super.onActivityResult(requestCode, resultCode, data);

	mFileString = mMediaUri.toString();
	mFileString = mFileString.substring(7, mFileString.length());

	if (requestCode == CAMERA_PIC_REQUEST && resultCode == Activity.RESULT_OK)
		createImageThumbnail();

	else if (requestCode == CAMERA_VIDEO_REQUEST && resultCode == Activity.RESULT_OK) {
		mDateTimeEndVideo = dateFormat.format(new Date());
		createVideoThumbnail();
	}
	else if (resultCode == Activity.RESULT_CANCELED)    // user went back without adding data
	{
		...
```

Photo and Video 2

http://developer.android.com/reference/android/hardware/Camera.html

- See the MediaPicker example
- Grabbing photos and videos using the Camera class via callbacks
  - Important: Call release() to release the camera for use by other applications. Applications should release the camera immediately in onPause() and re-open() it in onResume()

```
// take the photo, no surface is viewable, PhotoHandler class will receive the callback
public void onClickQuickPhoto(View view) {
      if (mCamera != null){
            PhotoHandler ph = new PhotoHandler(this);
            Camera.Parameters params = mCamera.getParameters();
            params.setPictureSize(size.width, size.height);
            ...
            mCamera.takePicture(null, null, ph);
      }
}
public class PhotoHandler implements PictureCallback {
   @Override
   public void onPictureTaken(byte[] data, Camera camera)
   {
      if(data != null)
      {
            File pictureFileDir = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
            String photoName = "picture_" + date + ".jpg";
            try {
                  FileOutputStream fos = new FileOutputStream(pictureFile);
                  fos.write(data);
                  fos.close();
                  Toast.makeText(context, "New Image saved:\n" + photoName, Toast.LENGTH_LONG).show();
                  ...
            }
```

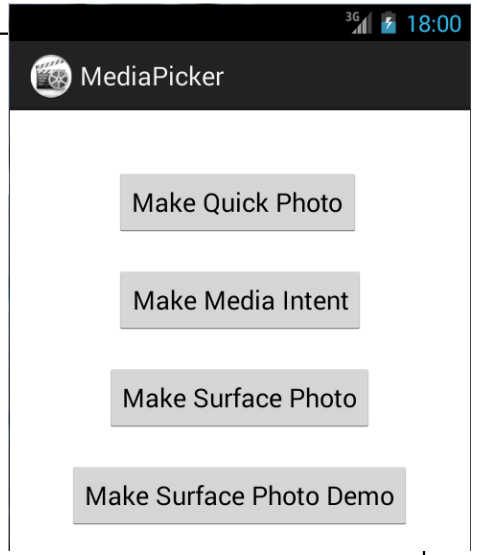
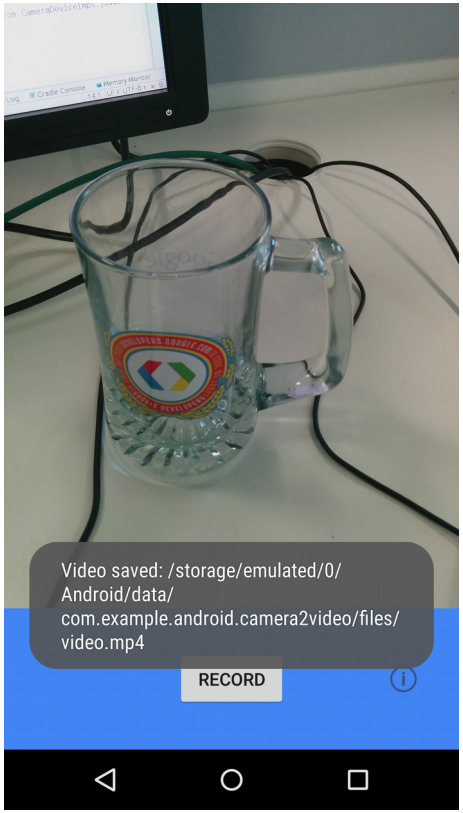


Photo and Video 3

- The Camera2 API is the current Android API
- Updated samples are maintanded at Google Samples
  - https://github.com/googlesamples?utf8=%E2%9C%93&query=camera
- Android-Camera2Video
- Android-Camera2Basic
- It can be **VERY** complicated to get apps to work on many devices since the manufacturers may have implemented less/various API support or have bugs in the implementation!



Portrait or landscape, which side is up?

- There are some classes that can be used to find out
  - **Display** - Provides information about the size and density of a logical display
  - **Configuration** - This class describes all device configuration information that can impact the resources the application retrieves
  - **Surface** - Handle onto a raw buffer that is being managed by the screen compositor

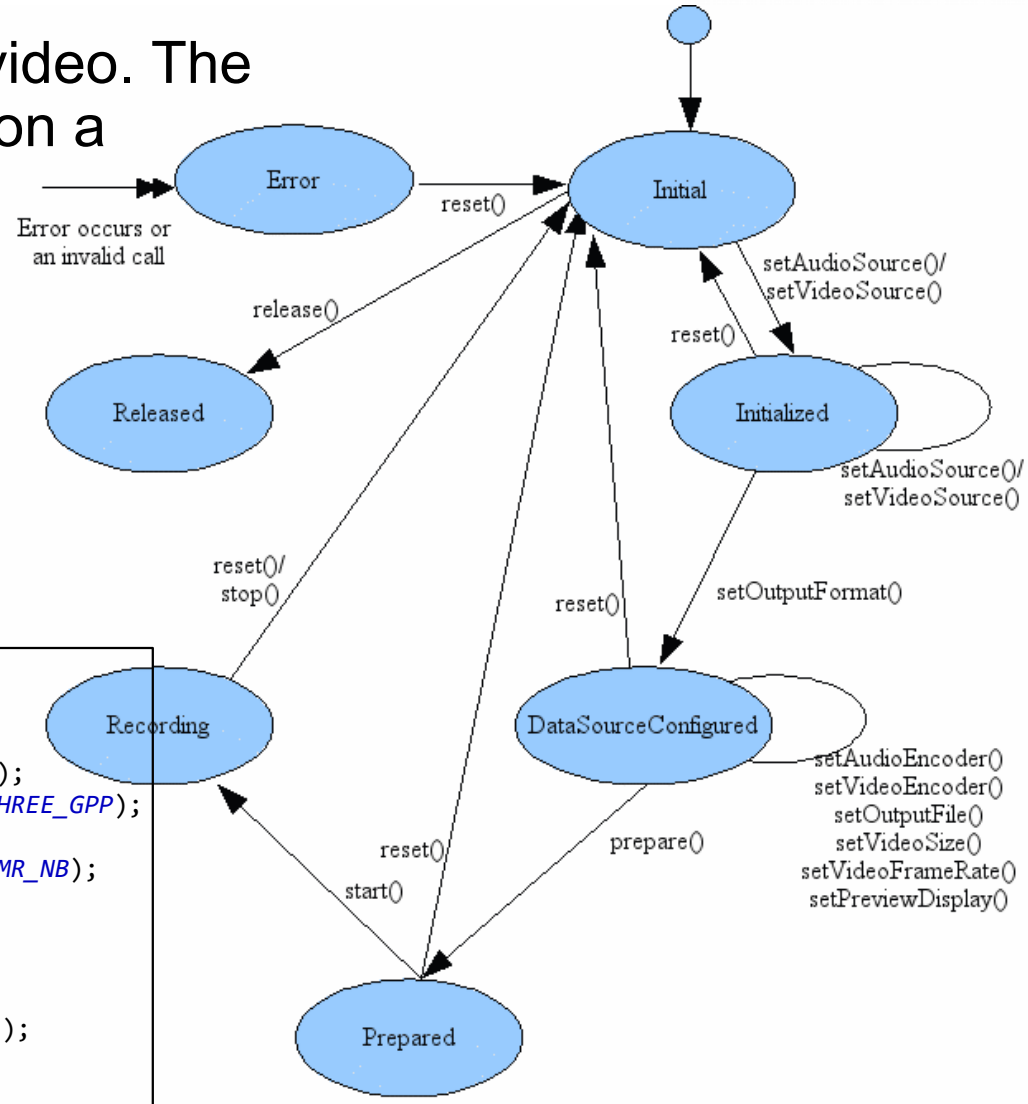
```
// Overall orientation of the screen
// May be one of Configuration.ORIENTATION_LANDSCAPE, Configuration.ORIENTATION_PORTRAIT.
int orientation = getResources().getConfiguration().orientation; // 1 = portrait, 2 = landscape

if(orientation == Configuration.ORIENTATION_PORTRAIT)
{ ... }
// we use this in accelerometer onSensorChanged(SensorEvent event)
Display display = ((WindowManager) getSystemService(Context.WINDOW_SERVICE)).getDefaultDisplay();
switch (display.getRotation())
{
   case Surface.ROTATION_0:
      x = event.values[0]; y = event.values[1];
      break;
   case Surface.ROTATION_90:
      x = -event.values[1]; y = event.values[0];
      break;
   case Surface.ROTATION_180:
      x = -event.values[0]; y = -event.values[1];
      break;
   case Surface.ROTATION_270:
      x = event.values[1]; y = -event.values[0];
      break;
}
```

Check out the API Demos samples for more sensor examples

More about sensors
http://developer.android.com/guide/topics/sensors/sensors_overview.html

MediaRecorder

- Used to record audio and video. The recording control is based on a simple state machine
- State diagram
  - http://developer.android.com/reference/android/media/MediaRecorder.html
- See the AudioRecordTest project

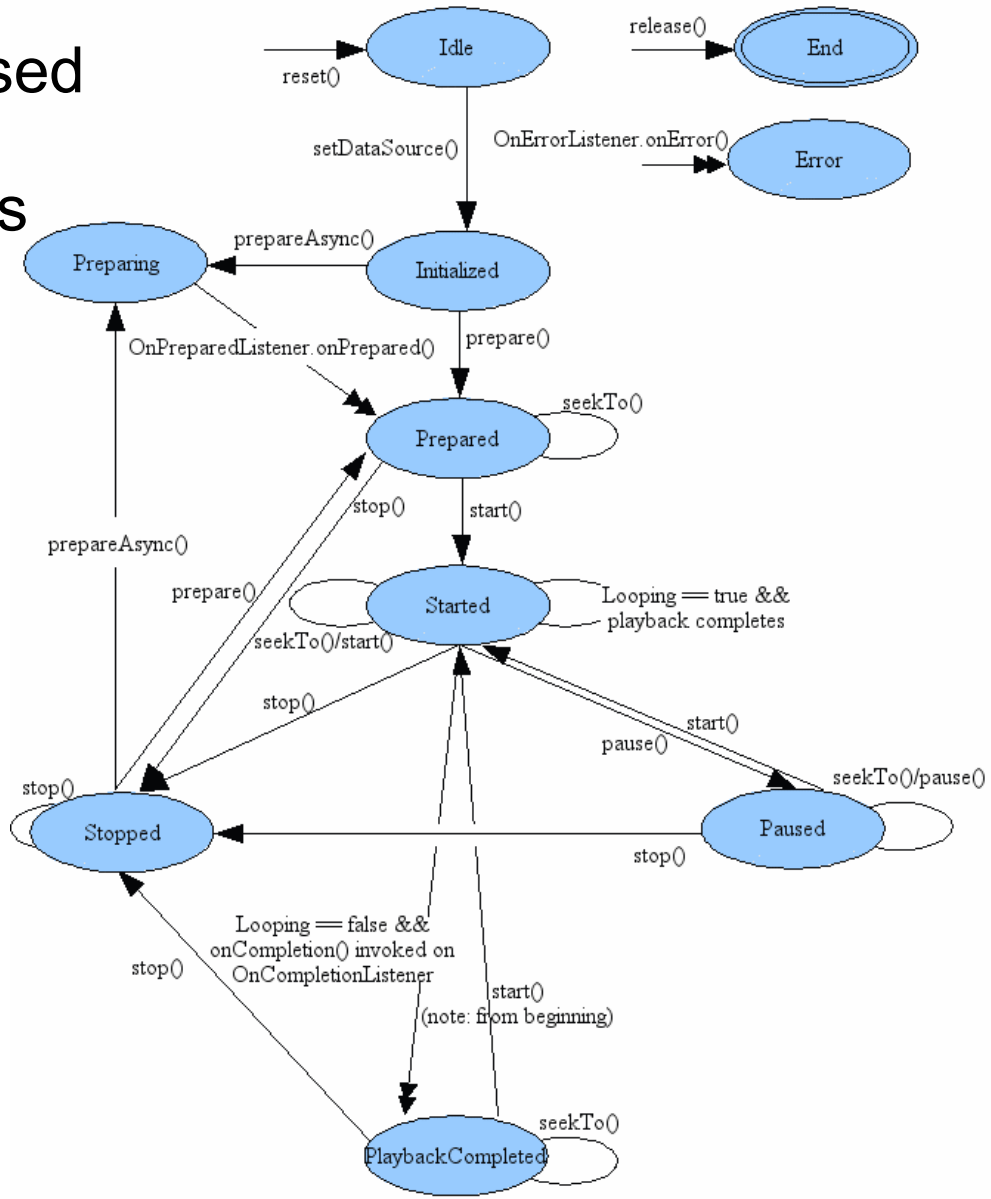


MediaRecorder state diagram

```
private void startRecording() {

  mRecorder = new MediaRecorder();
  mRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
  mRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
  mRecorder.setOutputFile(mFileName);
  mRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);

  try {
      mRecorder.prepare();
  } catch (IOException e) {
      Log.e(LOG_TAG, "startRecording()>prepare() failed");
  }

  mRecorder.start();
}
```

MediaPlayer 1

- MediaPlayer class can be used to control playback of audio/video files and streams
- State diagram
  - http://developer.android.com/reference/android/media/MediaPlayer.html
- See the AudioRecordTest ...




```
single arrow head represent
synchronous method calls,
double arrow head represent
asynchronous method calls
```

MediaPlayer 2

```
// from resources with static, player is already prepared when static create have been used
MediaPlayer player = MediaPlayer.create(this, R.raw.my_audio);
player.start();

// from a URL
String myUrl = "http://........"; // your URL here
MediaPlayer mediaPlayer = new MediaPlayer();
mediaPlayer.setAudioStreamType(AudioManager.STREAM_MUSIC);
mediaPlayer.setDataSource(this, myUrl);
mediaPlayer.prepare();
mediaPlayer.start();

// from a Uri
private void startPlaying() {
      mPlayer = new MediaPlayer();
      mPlayer.setOnCompletionListener(this);
      try {
         mPlayer.setDataSource(mFileName);
         mPlayer.prepare();
         mPlayer.start();
      } catch (IOException e) {
         Log.e(LOG_TAG, "startPlaying()>prepare() failed");
      }
}

@Override
public void onCompletion(MediaPlayer mp) {
    // when no longer needed
      mPlayer.release();
      mPlayer = null;
}
```

MediaPlayer examples

MediaController and VideoView

- MediaController is a view containing controls for a MediaPlayer. Typically contains the buttons like "Play/Pause", "Rewind", "Fast Forward" and a progress slider. It takes care of synchronizing the controls with the state of the MediaPlayer
  - http://developer.android.com/reference/android/widget/MediaController.html
- The VideoView class can load images from various sources (such as resources or content providers), takes care of computing its measurement from the video so that it can be used in any layout manager, and provides various display options such as scaling and tinting
  - http://developer.android.com/reference/android/widget/VideoView.html

```
VideoView videoView = (VideoView) findViewById(R.id.videoView1);
videoView.setKeepScreenOn(true);
// Alternatively, you can use videoView.setVideoPath(<path>); // videoView.setVideoPath("/sdcard/test2.3gp");
videoView.setVideoURI(Uri.parse("android.resource://" + getPackageName() + "/" + R.raw.videoviewdemo));
videoView.setMediaController(new MediaController(this));
videoView.requestFocus();
// other method examples
if (videoView.canSeekForward()) {
   videoView.seekTo(videoView.getDuration()/2);
}
videoView.start();
// ...
videoView.stopPlayback();
```

VideoView implements MediaController.MediaPlayerControl

YouTube Android Player API

- Powerful API to play Youtube videos in your app
- An API client library for your app interacts with a service that is distributed as a part of the YouTube app
  - Users need to run version 4.2.16 or higher of the mobile YouTube app to use the API
- Procedure to get and use it is similar to using the Google Maps Android API in Google Play services
  - Register with your to Gmail address and Android developer key to get an Android API key at the ”Google APIs Console” page
  - Put the YouTubeAndroidPlayerApi.jar in the lib folder
- YouTube Android Player API demo
- Begin here

https://developers.google.com/youtube/android/player/