SCIENCES SORBONNE UNIVERSITÉ

Maps and Localisation

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

GoogleMap

Not a part of Android

- It's a project from Google
- https://console.developers.google.com

Add your key in res/values/google_map_api.xml

```
<resources>
        <string name="google_maps_key_instructions"
                templateMergeStrategy="replace">
        <string name="google_maps_key"
                templateMergeStrategy="preserve">
             MY_KEY
        </string>
</resources>
```

Dependencies

Android Studio provide ready-to-use project

Install dependencies

- Tools/Android/SDK Manager
- Install Google Play Services
- Install Google Play Repository

Modify Graddle.build

```
compile 'com.google.android.gms:play-services:8.4.0'
```

You can now display a map

GoogleMap: details

Composed of the SupportMapFragment

- A fragment that can display a map
- Acquiring the map should be done through

```
getMapAsync(OnMapReadyCallback);
```

- Implement then the callback

Can be integrated directly

```
<fragment
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:name=
              "com.google.android.gms.maps.SupportMapFragment"
    android:id="@+id/map"
    android:layout_width="match_parent"
    android:layout_height="match_parent"/>
```

Maniplating Markers

Add a marker

```
private GoogleMap mMap;

// …
Marker m = mMap.addMarker(new MarkerOptions()
  .position(point)
  .icon(BitmapDescriptorFactory
          .defaultMarker(BitmapDescriptorFactory.HUE_ORANGE))
 .draggable(true)
 .alpha(0.7f)
 .visible(true)
 .title("Fancy title"));
 .snippet("Snippet for this marker"));
```

Remove a marker

```
m.remove();
```

Grab User Events

Click on the map

- setOnMapClickListener

Long click on the map

- setOnMapLongClickListener

Click on a marker

- setOnMarkerClickListener
- You must keep track of existing markers

Moving a marker

- setOnMarkerDragListern

Many other actions can be captured!

Localisation (Where Am I?)

Exact localisation is complex

- Multiple sources:
  - **GPS, Wifi, Cellular network**
- Various kind of precision
- The user is moving
- Energy saver with best precision

The 3 forms are based on LocationManager

…but Google Play API helps to build efficient applications

Many ways to get the User Localisation

- LocationManager framework
- GooglePlay location API
- Use existing features

```
mMap.setMyLocationEnabled(true);
```

Using Localisation

Modify AndroidManifest.xml

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

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

Two Kind of usages:

- ACCESS_COARSE_LOCATION:
  - approximative localisation
  - Give you the current block
- ACCESS_FINE_LOCATION
  - precise location
  - position in the street

Use Google Localisation API

Based on the latest known position

- enough in most cases
- can specify brand with / energy policy
- Specify Connection Failed
- Intensive use of Callbacks

Must be created and instantiated in onCreate method

```
GoogleApiClient mGoogleApiClient;
protected synchronized void buildGoogleApiClient() {
    mGoogleApiClient = new GoogleApiClient.Builder(this)
        .addConnectionCallbacks(this)
        .addOnConnectionFailedListener(this)
        .addApi(LocationServices.API)
        .build();
}
```

Following User's Moves (1/2)

Build requests

```
LocationRequest mLocationRequest;
protected void createLocationRequest() {
    mLocationRequest = new LocationRequest();
    mLocationRequest.setInterval(1000);
    mLocationRequest.setFastestInterval(1000);
    mLocationRequest.
        setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
}
```

Specify priority

PRIORITY_BALANCED_POWER_ACCURACY	Performances and Precisions WIfi + réseau cellulaire
PRIORITY_HIGH_ACCURACY	Extreme Localisation GPS
PRIORITY_LOW_POWER	City precision energy saving
PRIORITY_NO_POWER	No Consomation Localisation deduced from the other activities

Following User's Moves (2/2)

Then start requesting the position

```
protected void startLocationUpdates() {
    createLocationRequest();
    LocationServices.FusedLocationApi.requestLocationUpdates(
            mGoogleApiClient, mLocationRequest, this);
}
```

The callback onLocationChanged will then be called

- You have to do something in this callback

Using Location Manager (1/2)


```
// Acquire a reference to the system Location Manager
LocationManager locationManager = (LocationManager)
this.getSystemService(Context.LOCATION_SERVICE);

// Define a listener that responds to location updates
LocationListener locationListener = new LocationListener() {
    public void onLocationChanged(Location location) {
      // Called when a new location is found by the
      // network location provider.
      makeUseOfNewLocation(location);
    }

    public void onStatusChanged(String provider,
                                int status, Bundle extras) {}

    public void onProviderEnabled(String provider) {}

    public void onProviderDisabled(String provider) {}
};
```

Using Location Manager (2/2)

Build requests

```
// Register the listener with the Location Manager
// to receive location updates
locationManager.requestLocationUpdates
        (LocationManager.NETWORK_PROVIDER, 0, 0,
         locationListener);
```

You must then

- Define the source : GPS, WIFI, ...
- Define the frequency
- Define refreshing
- ...

Summary

Manipulation of maps is easy

- When you use Google API

In the emulator your can virtually move

- telnet 127.0.0.1 5554
- geo fix 12 40

LocationManager framework

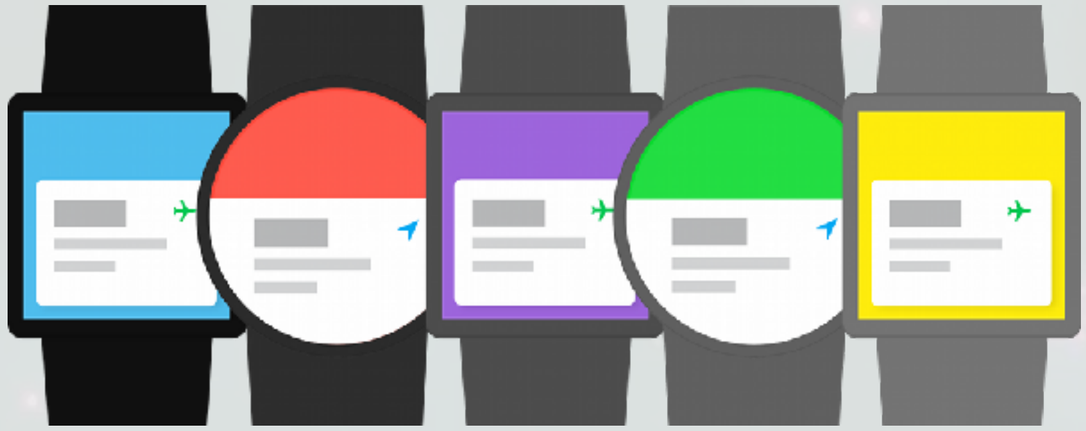
- Is complex to use
- Does not allows to specify energy policies
- BUT ...
- ... independent from Google !

SCIENCES SORBONNE UNIVERSITÉ

Wearables

Renault@lrde.epita.fr

Wearable



- Current trends for Application
- provide quick access to your app
- Easily customizable
- Objectives: ensure that the user will not miss any notifications from its favorite app

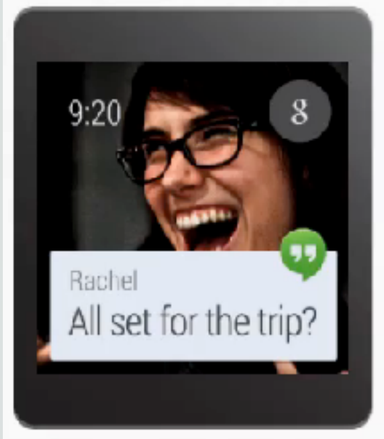
Many kind of wearable exists

- Activity trackers
- Watches
- Glasses
- ...

ContextStream and CueCard

ContextStream

- Smarter Notifications than in the phone
- Notifications are displayed vertically
- On a card, a right swipe displays more informations



CueCard

- open when saying "ok google" or when tapping the home screen
- Is triggered through voice intents

How to build a wearable activity?

Similar to a *classical* application

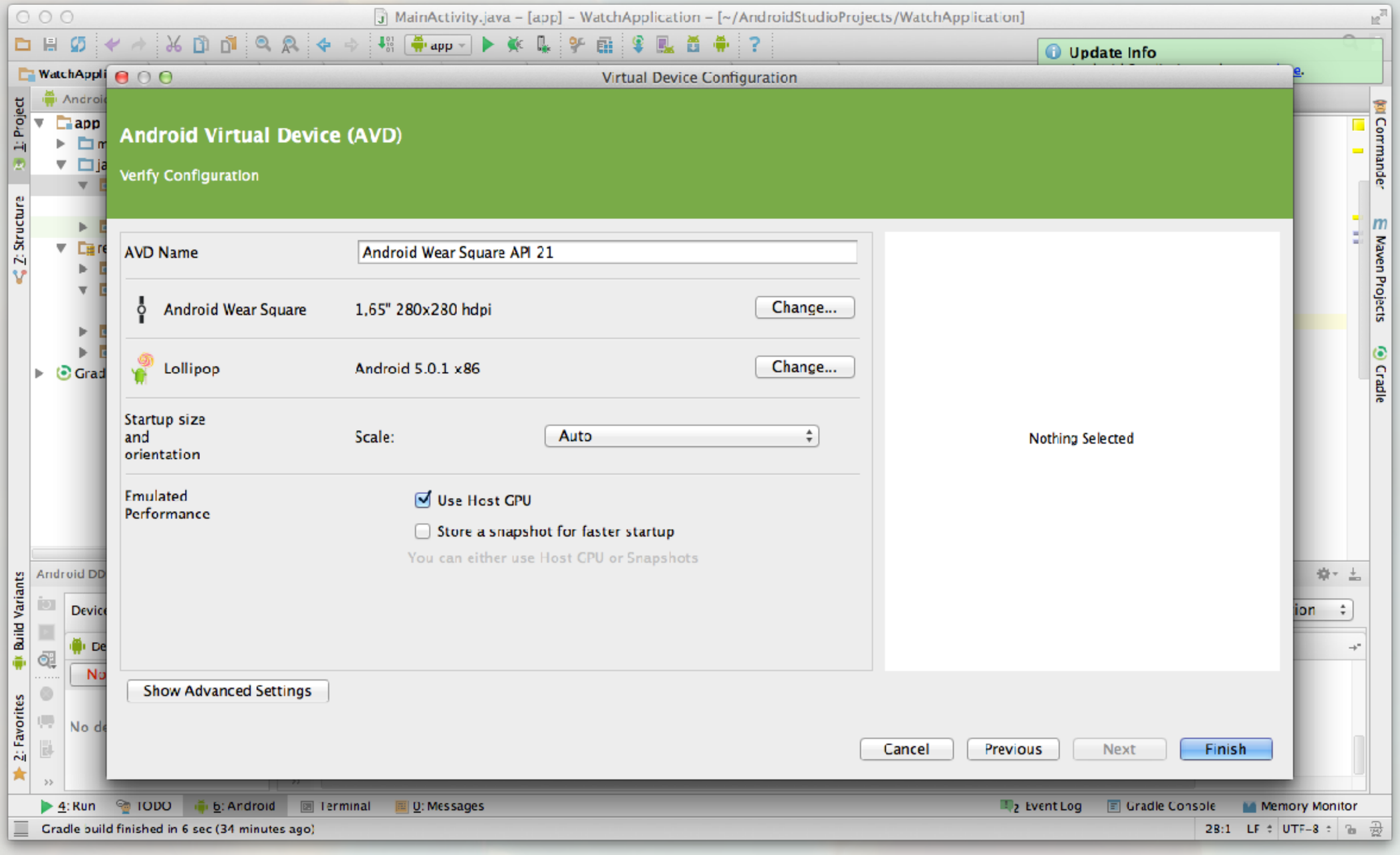
Things to know about:

- After some time the operating system is sleeping
  - **when the user will go back, its application will no longer be the active one**
  - **Home screen will be displayed**
- Small screen size !
  - **Not easy to manipulate a complex UI**
- Some frameworks are not supported
- Apps are not downloaded on the device
  - **The mother app is running on the phone**
  - **The wearable app is displaying on the wearable**

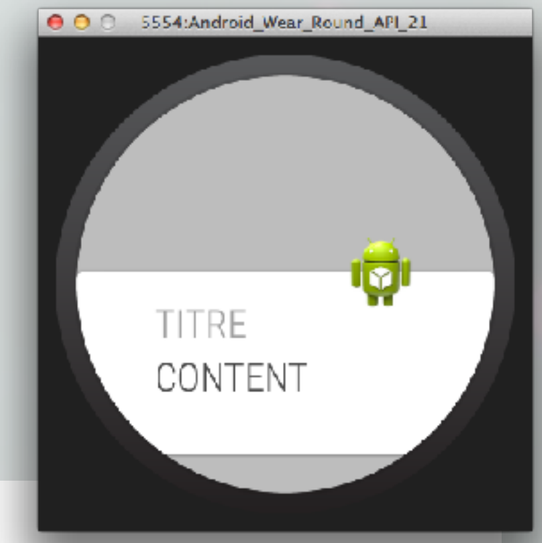
Building a test device

Building a virtual wearable is easy

- AVD manager provides a lot of predefined skills



Building a notification



Notifications are displayed after a top-down move

- An action can be defined when tapping a notification

```
Builder notificationBuilder = new Builder(this)
        .setSmallIcon(R.drawable.ic_launcher)
        .setContentTitle("TITRE")
        .setContentText("CONTENT");

 // Get an instance of NotificationManager Service
NotificationManagerCompat notificationManager =
        NotificationManagerCompat.from(this);

// Build the notification and issues it with
//notification manager.
notificationManager.notify(notificationId,
notificationBuilder.build());
```

Using voice to launch Applications (1/2)

Fundamental mechanisms for wearable

- Running predefined apps:
  - **Take a note, Set an alarm**
- Running third party apps:
  - **Reaction to the "Start" command**
  - **The application is then launched**

Running predefined apps

```
<activity android:name="MyNoteActivity">
    <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category
    android:name="com.google.android.voicesearch.SELF_NOTE" />
</intent-filter>
</activity>
```

Using voice to launch Applications (2/2)

Running third party apps

- Same mechanism
- Only specify the name that must be speak

```
<application>
 <activity android:name="StartRunActivity"
           android:label="My Running App">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category
          android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
 </activity>
</application>
```

Voice interaction (1/2)

The app may interact with voice

- Build an intent to capture user voice
- Result is provided as a list of string

Start listening

```
private static final int SPEECH_REQUEST_CODE = 0;

// Create an intent that can start the Speech Recognizer
private void displaySpeechRecognizer() {
    Intent intent = new
    Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    // Start the activity, the intent will be
    // populated with the speech text
    startActivityForResult(intent, SPEECH_REQUEST_CODE);
}
```

Voice interaction (2/2)

Getting results

```
// This callback is invoked when the Speech Recognizer
// returns. This is where you process the intent and extract
// the speech text from the intent.
@Override
protected void onActivityResult(int requestCode,
                                int resultCode,
                                Intent data) {
    if (    requestCode == SPEECH_REQUEST_CODE
         && resultCode == RESULT_OK) {
         List<String> results = data.getStringArrayListExtra(
                 RecognizerIntent.EXTRA_RESULTS);
         String spokenText = results.get(0);
         // Do something with spokenText
    }
    super.onActivityResult(requestCode, resultCode, data);
}
```

Summary

Developing a wearable activity

- is similar to developing a classical one
- Android studio also provides builtin tools
- can't be done without a phone-side application

Some specificities must be considered

- ContextStream

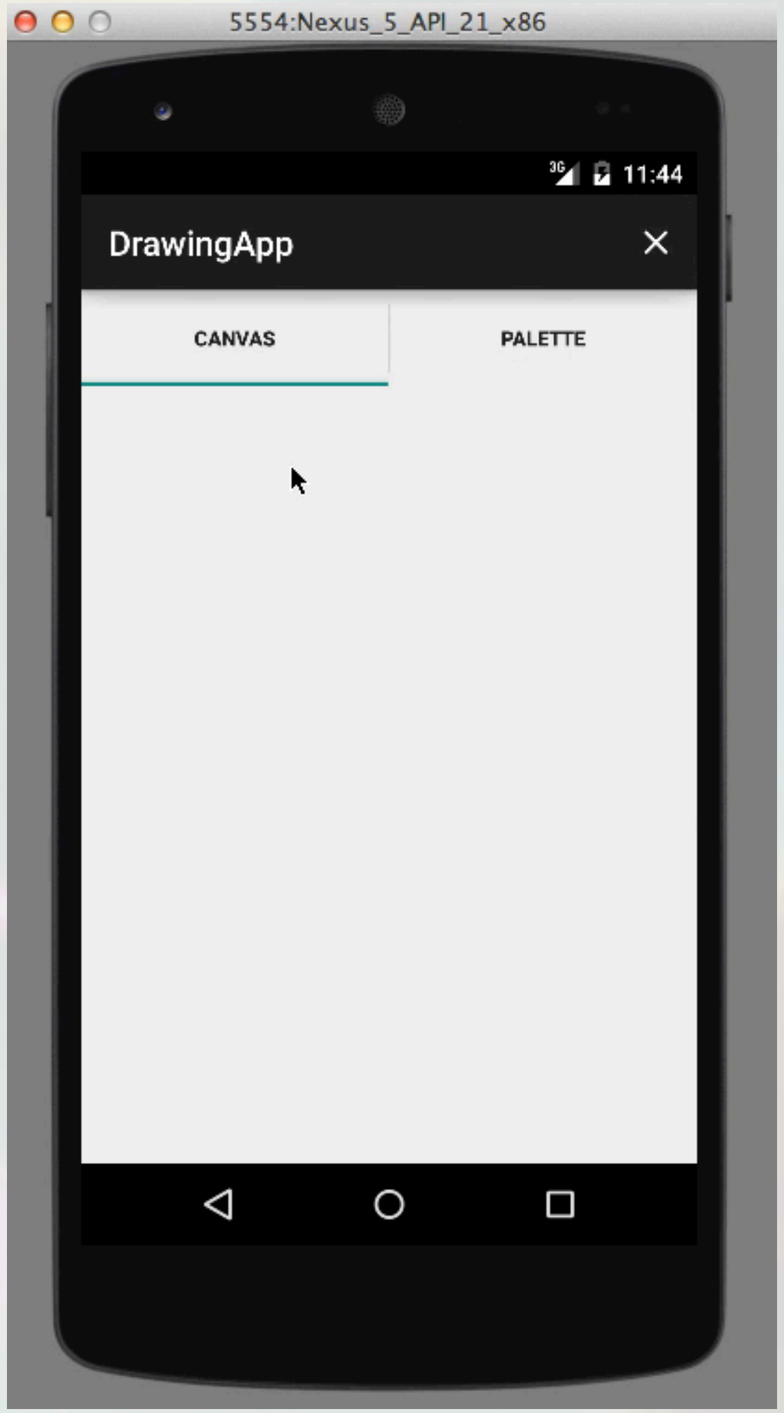
SCIENCES SORBONNE UNIVERSITÉ

Exercise: Drawing Application

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Demo Video



GUI and Implementation details

Multiples implementations for the layout

- Use button and fragments
  - **Two buttons canvas and palette**
  - **A click on a button update the main view with the DrawingCanvas or the List**

- OR Use ActionBar and ViewPager

- OR Use TabHost and TabWidgets

Define a DrawCanvas

- Look to classes Paint and Path

Grab User touches and clicks

Summary

Toward an end-to-end application

- Add persistency for rotation
- Add persistency even when the application is closed
- Add the ability to draw predefined shapes
- Add the ability to share the drawing with the ActionBar
  - **Useful for many applications**
    - **BItstrip**
    - **Snapshat**
    - **doodle.ly**
    - **…**
- Add the ability to create movies from a sequence of images
- Add the ability to explore and save previous drawing

This App teach you how to deal with view and how to move them through the UI

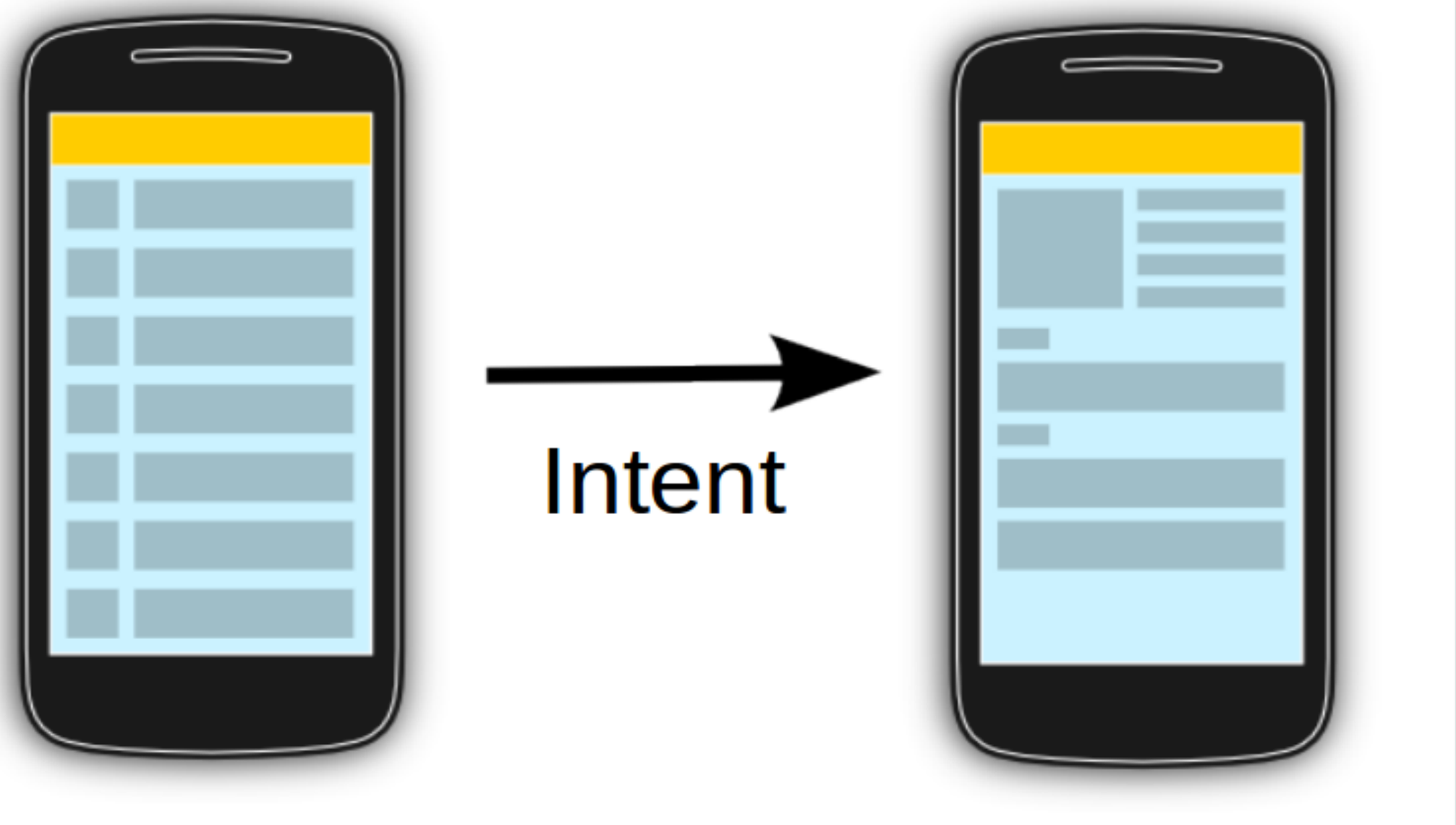
SCIENCES SORBONNE UNIVERSITÉ

Intents and Intent Bus

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Main Idea



What is the goal of Intents?

Two kind of components in Android

- Activities: GUI for the user
  - **Game, mails, etc.**
- Services: background tasks
  - **music player, sync., etc.**

Goals of the Intents

- Ease the communication between these components
- Provide functionalities for the other apps

How intents are managed by the system?

- Asynchronous communication bus
- Intent filters

Structure of an Intent

action	data

Action examples

action	data	description
ACTION_VIEW	content://contact/people/1	Information about people with UID 1
ACTION_VIEW	tel:123	Display keyboard with pre-composed 123 numero
ACTION_DIAL	content://contact/people/1	Display keyboard already filled with the phone number of people with UID1

Extra parameters for Intents

action	data	category	type	component	extra

- **category: extra informations for the actions**
  - ACTION_MAIN+CATEGORY_HOME: launch home screen
- **type: the MIME type in data**
  - may be deduced sometime by the system
- **component: the name of the component targeted by the intent**
- **extra: a bundle of informations (e.g. subject for a mail)**

Target of an Intent

Explicit target:

- The name is specified in the Intent
- The specified target will be triggered

```
Intent intent = new Intent(getApplicationContext(),
                           MyClass.class);
startActivity(intent);
```

Implicit target

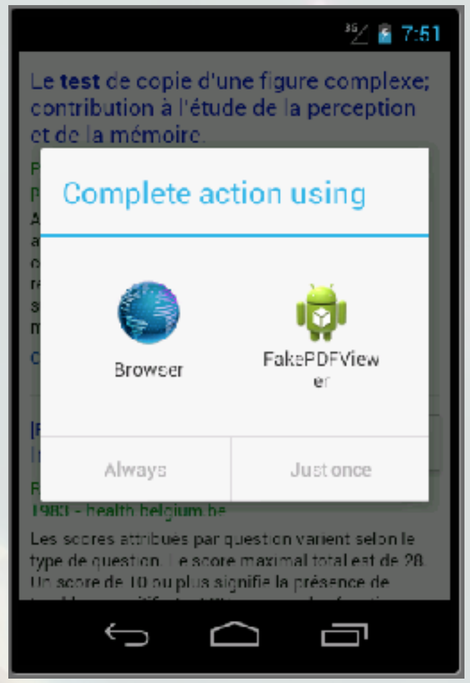
- The system have a lookup to find the best component to trigger
- This lookup is called **Intent Resolution**

```
Intent intent = new Intent(Intent.ACTION_DIAL);
intent.setData(Uri.parse("tel:"+phone));
startActivity(intent);
```

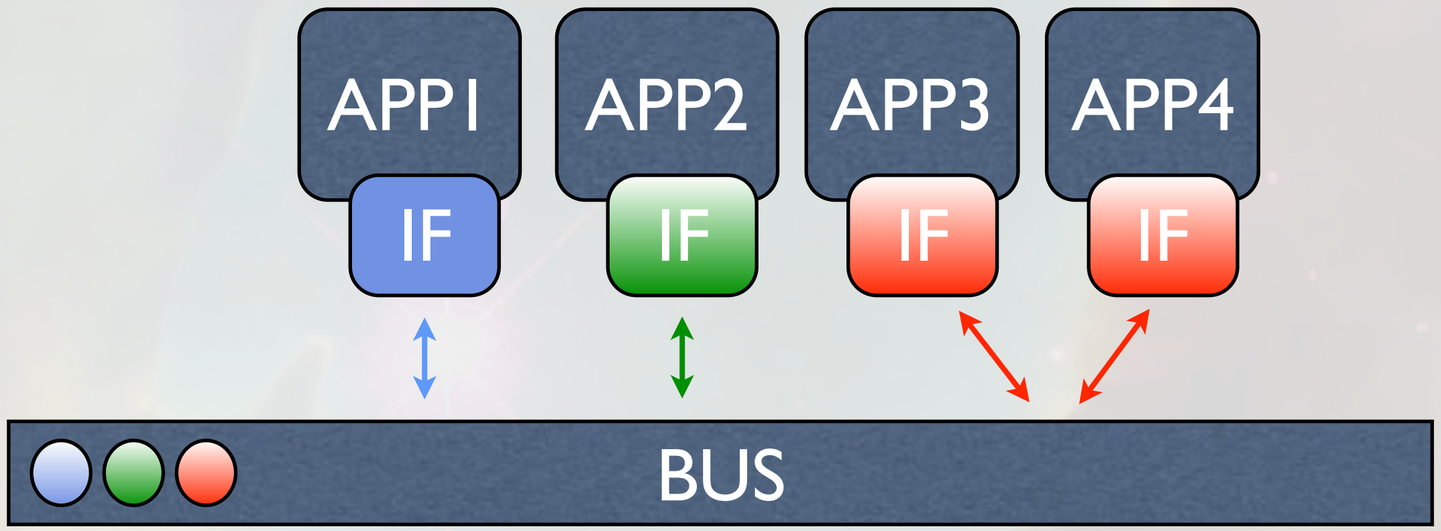
Intent Resolution

Intents Filters

- Each class define its realizable actions
- In the AndroidManifest.xml



The lookup explores all Intent Filters



Declare an Intent Filter

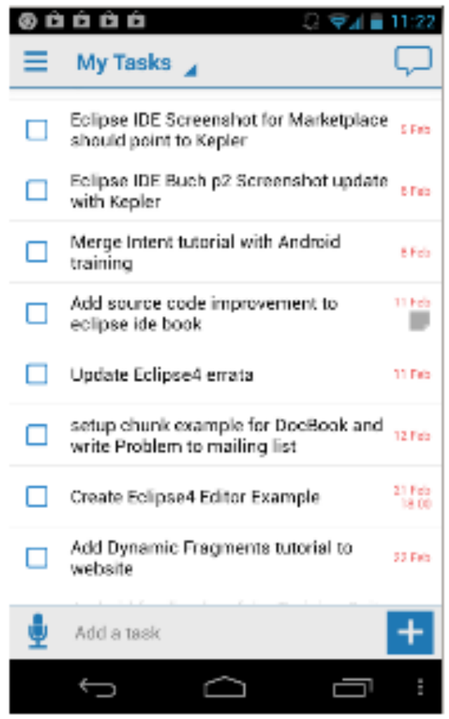
For instance, build an application that may read PDF

- Modify the AndroidManifest.xml

```
<intent-filter>
    <action
        android:name="android.intent.action.VIEW"                    />
    <category
        android:name="android.intent.category.DEFAULT"    />
    <category
        android:name="android.intent.category.BROWSABLE"  />
    <data
        android:mimeType="application/pdf"                           />
</intent-filter>
```

Result from an Intent

Intent + resultCode provided by called activity



onActivityResult(requestCode, resultCode, intent)

RequestCode provided by Android to identify which activity type was started

startActivityForResult:

- Identify a call
- Result is provided in onActivityResult

Summary

Intent Bus

- Helps the communication between components
- Circulation of asynchronous messages

The target of an Intent can be

- Explicit: if the name of the component is specified
- Implicit: the system tries to find the best application
  - **A component can have multiple Intent Filters**
  - **If multiple component can answer, the user choose!**

We can obtain results after the return of a a component triggered by an Intent

SCIENCES SORBONNE UNIVERSITÉ

Services

Renault@lrde.epita.fr

What is the goal of Services?

Background Tasks

- Similar to services on Linux
- No GUI
- Useful for running a background task
  - **Music for instance**

Lifecycle is simplified

A Service can be controlled by an activity or an widget

- in other words, we (most-of-the-time) needs an UI to interact

Services can be shared among multiple components

More details

Warning!

- By default, the service run in the process that triggered the service
- Neither a thread, nor a separated processus by default!

The system only

- instantiate the service
- run the callbacks
- the developper has to allocate ressources, use dedicated thread, etc.

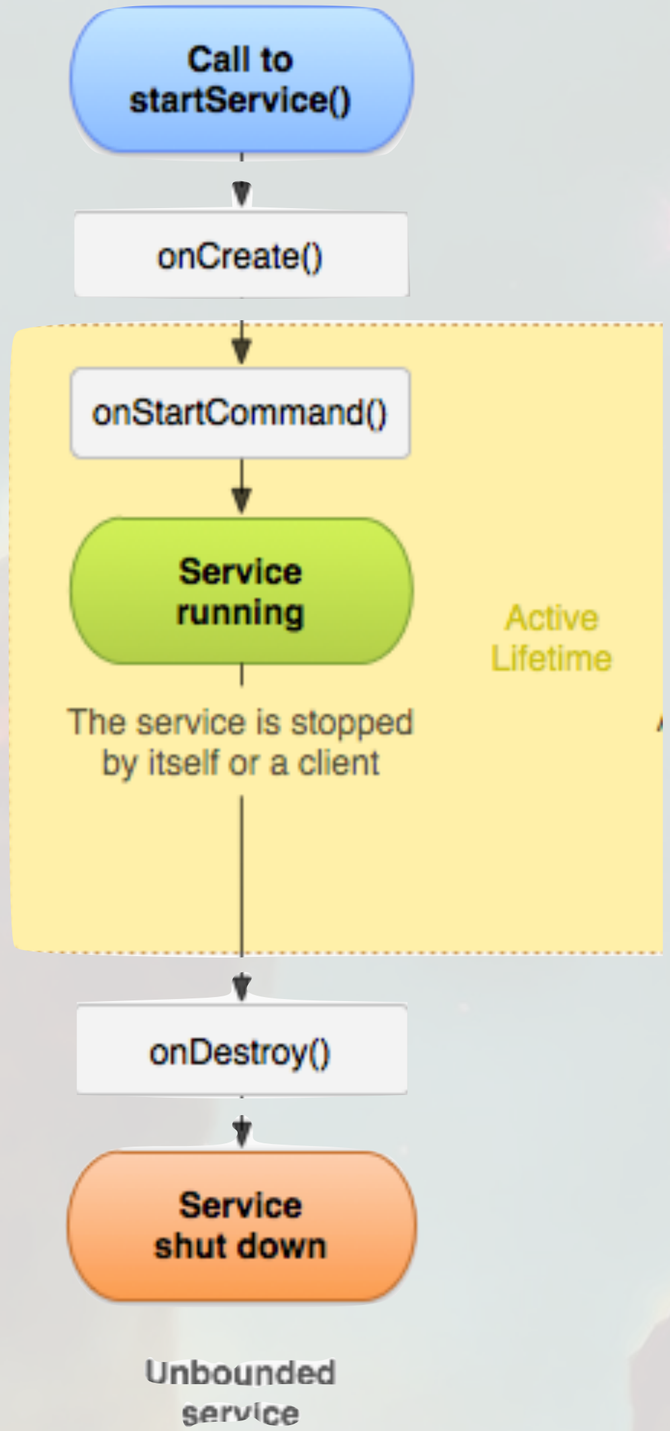
Two kind of services exists

- Bounded:
  - **a single instance**
  - **runs continuously**
- Unbounded:
  - **runs as long as some component use it**

Unbounded Services

Running the service

- onStartService
- Parameters are passed through Intents



One-way communication

Stopping the service

- stopService
- The number of start request is not taking in account for stopping the service

Unbounded Service: Media Player Example (1/2)


```
public class UnboundedService extends Service {
    MediaPlayer mediaPlayer;

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

    @Override
    public void onCreate(){
        mediaPlayer = MediaPlayer.create(this, R.raw.mymusic);
        mediaPlayer.setLooping(false);
    }

    @Override
    public void onDestroy(){
        mediaPlayer.stop();
    }
```

Unbounded Service: Media Player Example (2/2)


```
    @Override
    public int onStartCommand(Intent intent, int flags, int
startID){
        mediaPlayer.start();
        // START_STICKY to continue until the
        // service is stopped
        return START_STICKY;
    }
}
```

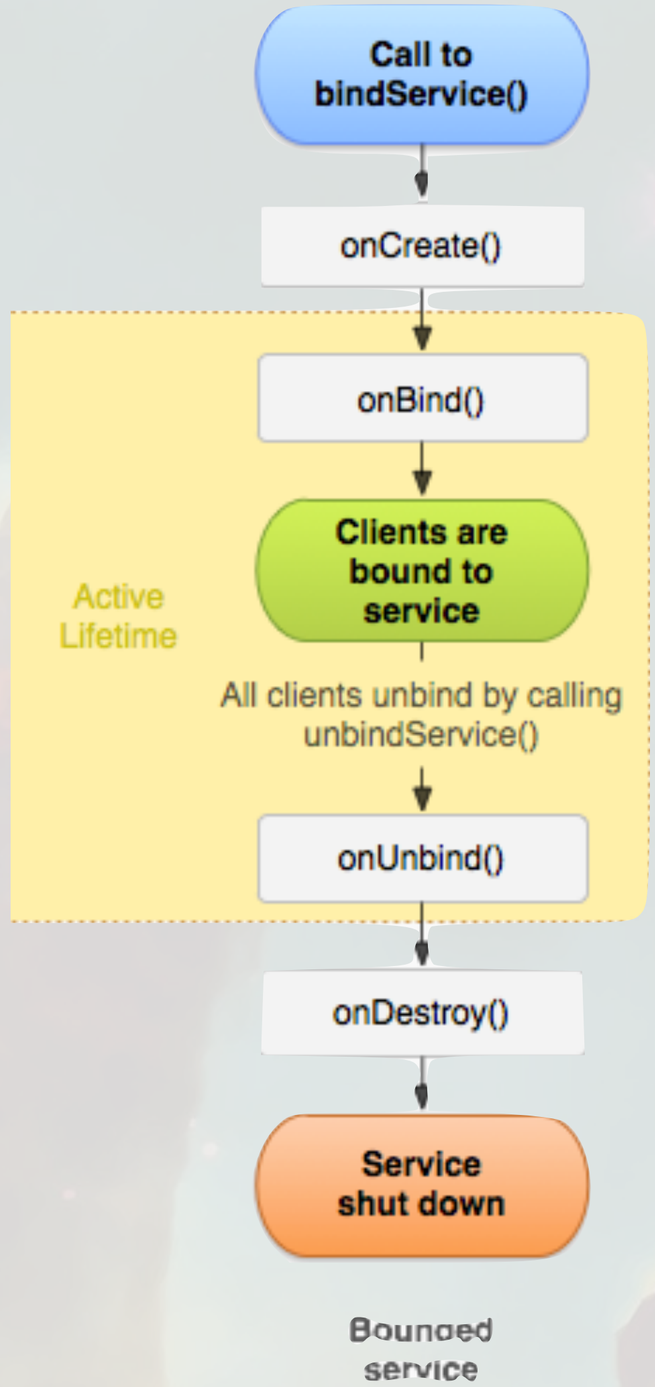
Declaration in the AndroidManifest.xml

```
 <service
        android:name=".UnboundedService"
        android:enabled="true">
</service>
```

Bounded Services

Starting service

- onBind
- returns a binder, close to notion of client/server
- close to IPC
- First client creates the service



The service is manipulated through the binder

Stoping the service

- Clients disconnect
- Last client destroy the service

Bounded Service Example (1/3)

Construct the binder (IBinder)

```
public class IDBinder extends Binder {
    int pseudoRnd = 0;

    public int getNewID(){
        ++pseudoRnd;
        return pseudoRnd;
    }
}
```

Building the service

```
public class BoundedService extends Service {
    @Override
    public IBinder onBind(Intent intent) {
        return myIDBinder;
    }
    IDBinder myIDBinder = new IDBinder();
}
```

Bounded Service Example (2/3)

Connecting with the bounded service

```
boolean isBinded = false;
    IDBinder idService;
    ServiceConnection mConnection = new ServiceConnection() {
        @Override
        public void onServiceConnected(ComponentName name,
                                       IBinder service) {
            idService = (IDBinder) service;
            isBinded = true;
        }
        @Override
        public void onServiceDisconnected(ComponentName name) {
            isBinded = false;
        }
};
```

Bounded Service Example (3/3)

Fix the AndroidManifest.xml

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

Connect to the service

```
getApplicationContext()
    .bindService(new Intent(getApplicationContext(),
                                BoundedService.class),
                 mConnection, BIND_AUTO_CREATE);
```

Disconnect to the service

```
getApplicationContext().unbindService(mConnection);
```

Call the service

```
idService.getNewID();
```

Summary

Two kind of Services

- Bounded or not depending on your needs
- Can be started with different policies in case of early terminaison (START_STICKY)

Services work well with widgets to build background tasks

Services have their own lifecycle

Must handle threads

- in the example, music player handle that

SCIENCES SORBONNE UNIVERSITÉ

Broadcast Receiver

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Definition

- **Component that responds to events**

- **Most of the events are triggered by the system**
  - Low battery, screen locked, picture taken...

- **... BUT some may be triggered by applications**
  - To inform that something is now available

- **BroadcastReceivers:**
  - Don't have a GUI
  - But can launch some notifications

Building a BroadcastReceiver

Modify the AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.admin.mybroadcastapplication" >

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

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

<receiver android:name=".MyReceiver">

   <intent-filter>
     <action android:name="android.permission.READ_SMS"/>
     <action android:name="android.permission.RECEIVE_SMS"/>
   </intent-filter>

</receiver>
```

Registering & Understanding Lifecycle

Register the BroadcastReceiver

- Specify programmatically
  - the receiver
  - the IntentFilter

```
registerReceiver(new MyReceiver(), new IntentFilter
                 ("android.provider.Telephony.SMS_RECEIVED"));
```

Lifecycle

- Simplest lifecycle possible
- Only one action onReceive
  - In this method, computing must be short and without asynchronous processing

Implementation of a Broadcast Receiver


```
public class MyReceiver extends BroadcastReceiver {
   @Override
   public void onReceive(Context context, Intent intent) {
      NotificationCompat.Builder mBuilder =
            new NotificationCompat.Builder(context);
       mBuilder.setContentTitle("Alert, Critical Message!");
       mBuilder.setContentText("This is Notification Detail!");

       // Create
       NotificationManager notificationmanager =
                (NotificationManager) context
                .getSystemService(Context.NOTIFICATION_SERVICE);

       // Build Notification with Notification Manager
       notificationmanager.notify(0, mBuilder.build());
   }
}
```

No direct access to the UI

Broadcasting its own Events

Defining a new event

```
<receiver android:name=".MyReceiver">
  <intent-filter>
  <action android:name="com.example.mybdcastapp.CUSTOM_INTENT"/>
</intent-filter>
```

Register to an event

```
registerReceiver(new MyReceiver(), new
        IntentFilter(« com.example.mybdcastapp.CUSTOM_INTENT"));
```

Broadcasting

Normal Broadcast

(sendBroadcast)

- Asynchronous messages
- No ordering between receivers
- Efficient

```
Intent intent = new Intent();
intent.setAction("com.example.mybdcastapp.CUSTOM_INTENT");
sendBroadcast(intent);
```

Ordered Broadcast

(sendOrderedBroadcast)

- One receiver at a time
- Priority can be fixed through android:priority
- Receiver can stop diffusion

```
Intent intent = new Intent();
intent.setAction("com.example.mybdcastapp.CUSTOM_INTENT");
sendOrderedBroadcast(intent);
```

A word on Security

- **BroadcastReceiver use Context that**
  - allows access to application-specific ressources & classes
  - etc.

- **Ensure that you only work on Intents or string that are in your namespace**

- **Some applications do not respect IntentFilter**
  - This can be fight using android:exported=false

- **All applications can target a broadcast receiver**
  - Fix that by using Android permissions

Summary

- **BroadcastReceiver offer a way to trace all the system events**

- **Two kind of broadcasting**
  - ordered
  - asynchronous

- **LocalBroadcastManager**
  - Do not use shared memory (inter-processes)
  - High security: we can share sensitive information
  - Other application cannot target these managers

SCIENCES SORBONNE UNIVERSITÉ

Content Providers

Renaultᵃlrde.epita.fr

Inter-processes communication

How to exchange information between processes?

Send Intents

- fill the extra with a lot of informations
  - **Management can be hard**

Share information through database

- Not available for inter-process communication

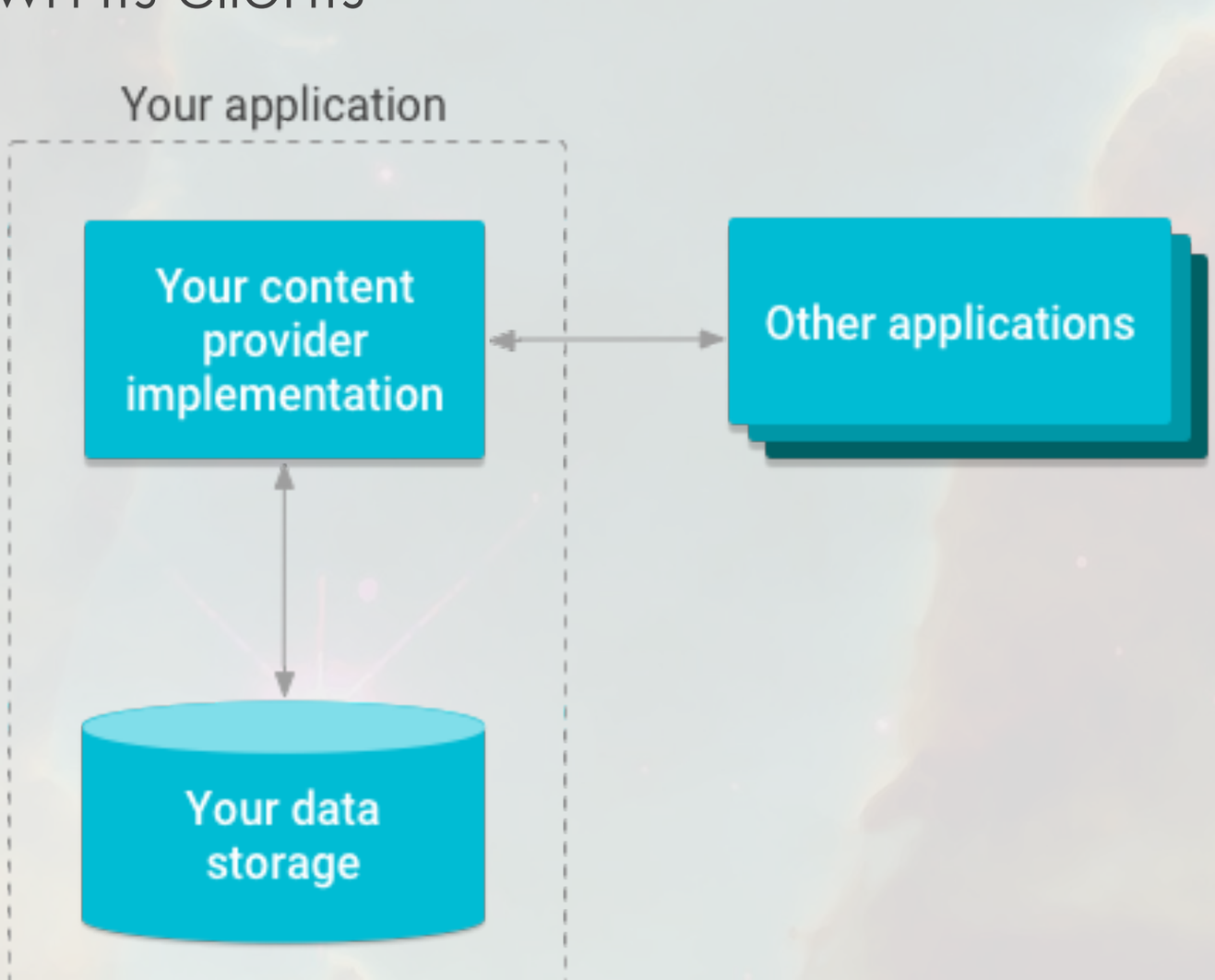
Use files

- Read / Write must be synchronized
- The format must be fixed
- cannot restrict to a specific application

ContentProviders

A provider

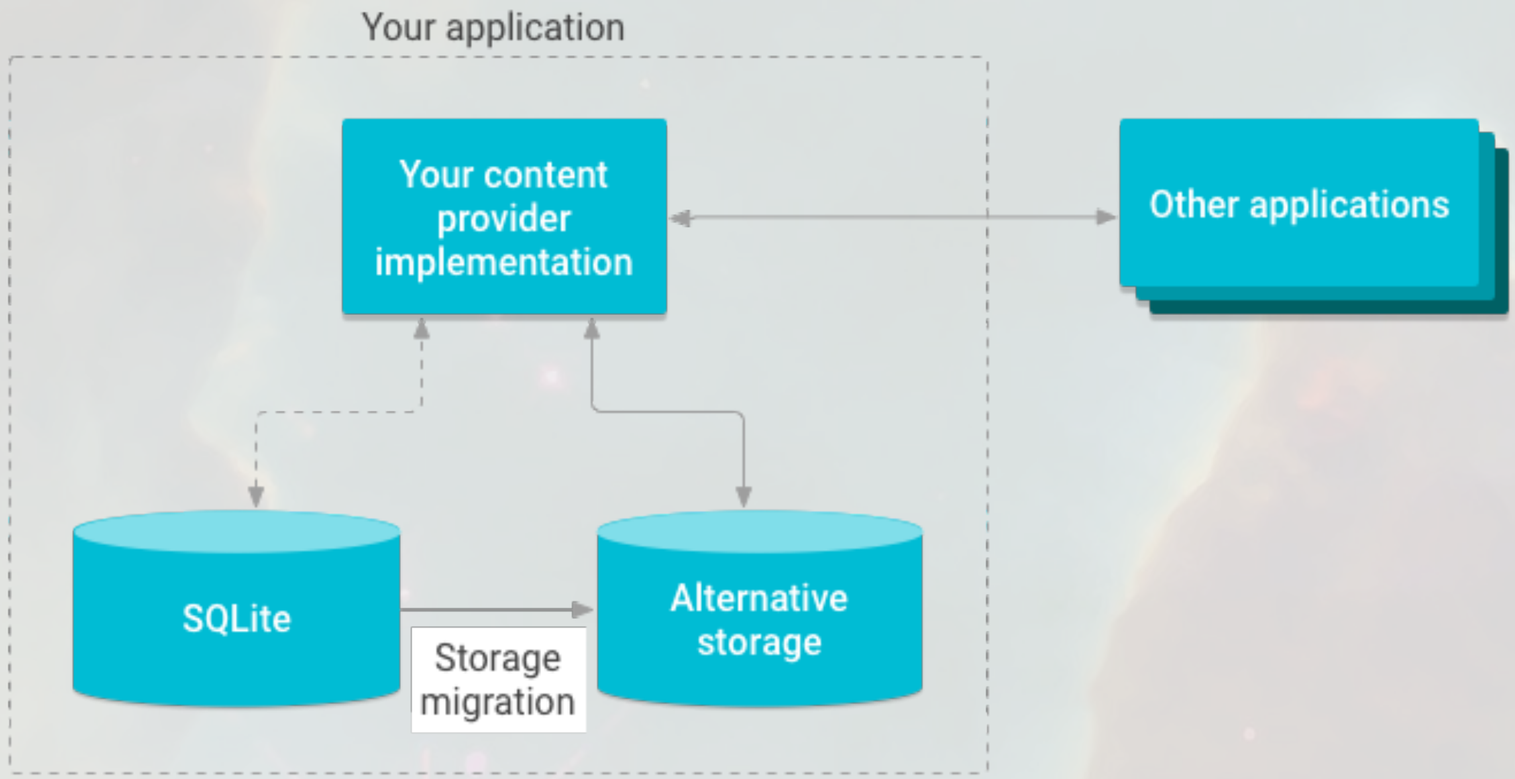
- Handle some content (data)
- Manage this content through a database (for instance)
- Must ensure the validity of its database
- May not known its clients



Advantages of ContentProviders? (1/2)

Nice abstraction

- Changing the underlying representation does not affect other applications



You can use it even if you don't plan to share with other Applications

Advantages of ContentProviders? (2/2)

- **Granular control over the permissions for accessing data**
  - Restrict access
  - Grant blanket
  - Configure read / write permissions

- **Can be used to mask many sources**
  - Your data appears as a single one

- **Nice pattern that abstract your data**
  - SQL or raw files are accessed the same way

Cursors

Wrap around read / write data

- Data is manipulated without knowing the underlying structure
- Can be seen as a direct access to a database raw

Offers CRUD features

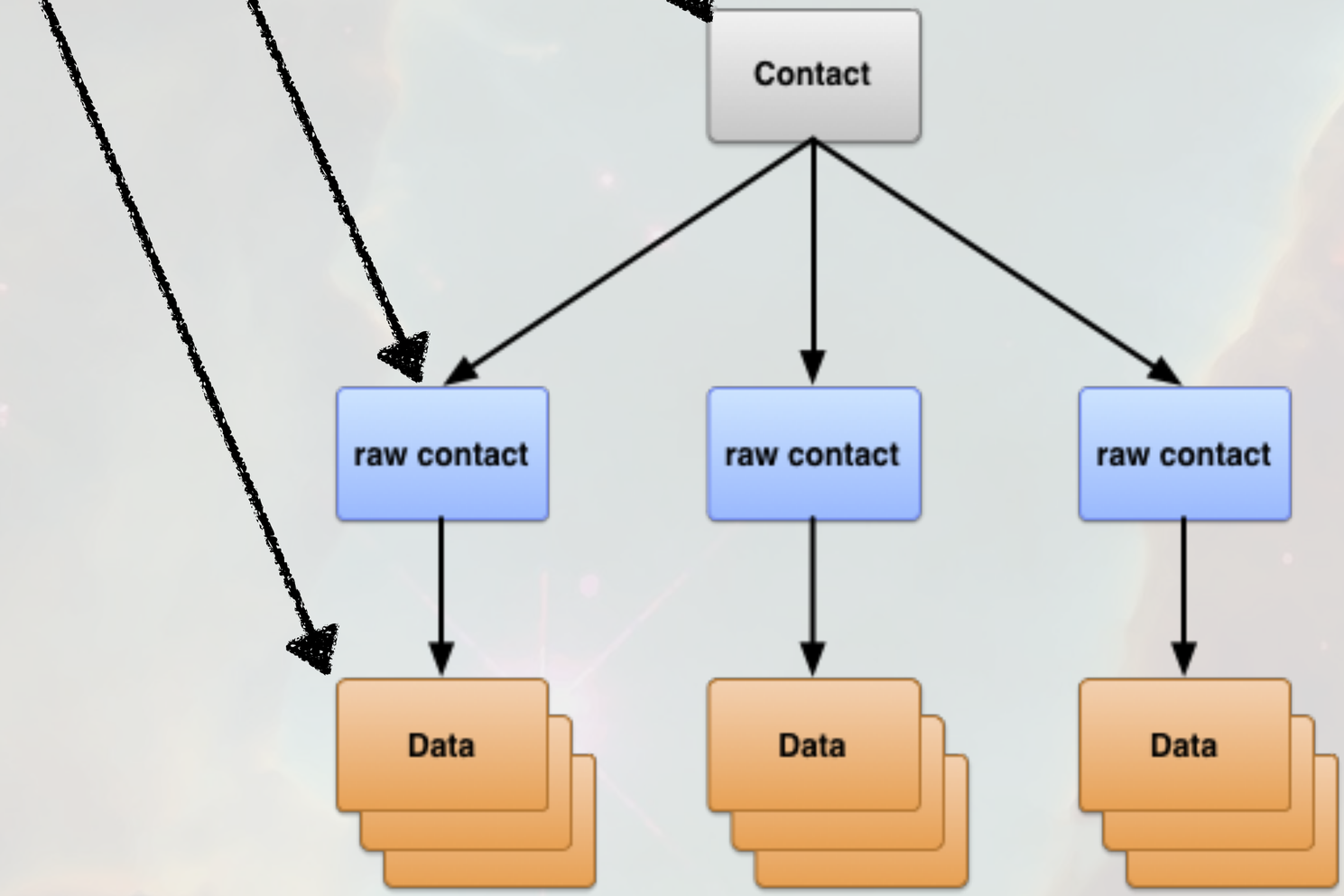
- **C**reate: create a new data
- **R**emove: remove some existing data
- **U**pdate: update existing data
- **D**elete: delete existing data

Implement android.database.Cursor

Using existing ContentProvider (Contacts)

Contacts structure

- ContactContract.Contracts: store all contacts
- ContactContract.RawContact: store summary for each contact
- ContactContract.Data: SMS, mails, etc.



Access to Contacts

Modify the AndroidManifest.xml

```
<manifest
    xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.admin.mycontactapplication"
>
<uses-permission
    android:name="android.permission.READ_CONTACTS"
/>
```

Build the request

```
ContentResolver cr = getContentResolver();
Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI,
                      null, null, null, null);
```

Parameters for filtering etc.

Exploiting the cursor


```
if (cur.getCount() > 0) {
  while (cur.moveToNext()) {
     String name = cur.getString(
          cur.getColumnIndex(
               ContactsContract.Contacts.DISPLAY_NAME));

        Toast.makeText(getApplicationContext(),name,
                       Toast.LENGTH_SHORT).show();
     }
}
```

We can then look in the other bases if there are additional informations

Define a ContentProvider

Define your own URI

```
contents://com.example.admin.mycontactapplication
```

Implement ContentProvider

onCreate()	Prepare ContentProvider
getType(Uri)	Return the MIME type of the URI
delete(…)	Suppress an entry
insert(…)	Insert a new entry
Update(…)	Update a data

Create Database (1/2)


```
private SQLiteDatabase db;

static final String DATABASE_NAME = "mydb";

static final String TABLE_NAME = "names";

static final int DATABASE_VERSION = 1;

static final String CREATE_DB_TABLE =
          " CREATE TABLE "
        + TABLE_NAME
        + " (id INTEGER PRIMARY KEY AUTOINCREMENT, "
        + " name TEXT NOT NULL);";

private static class DatabaseHelper extends SQLiteOpenHelper {
    DatabaseHelper(Context context) {
       super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }
```

Create Database (2/2)


```
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_DB_TABLE);
}

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion,
                          int newVersion) {
      db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
      onCreate(db);
    }
}
```

Define Constants


```
public class MyProvider extends ContentProvider {
    static final String PROVIDER_NAME =
            "com.example.admin.mycontactapplication.MyProvider";
    static final String URL =
            "content://" + PROVIDER_NAME + "/cte";

    static final Uri CONTENT_URI = Uri.parse(URL);

    static final String id = "id";
    static final String name = "name";
    static final int uriCode = 1;

    static final UriMatcher uriMatcher;
    private static HashMap<String, String> values;

    static {
        uriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
        uriMatcher.addURI(PROVIDER_NAME, "cte", uriCode);
        uriMatcher.addURI(PROVIDER_NAME, "cte/*", uriCode);
    }
```

Implement methods (1/4)


```
@Override
public String getType(Uri uri) {
 switch (uriMatcher.match(uri)) {
  case uriCode:
    return "vnd.android.cursor.dir/cte";
  default:
   throw new IllegalArgumentException
                ("Unsupported URI: " + uri);
  }
}

@Override
public boolean onCreate() {
   Context context = getContext();
   DatabaseHelper dbHelper = new DatabaseHelper(context);
   db = dbHelper.getWritableDatabase();
   if (db != null)
     return true;
  return false;
}
```

Implement methods (2/4)


```
@Override
public int delete(Uri uri, String selection,
                  String[] selectionArgs) {
  int count = 0;
  switch (uriMatcher.match(uri)) {
       case uriCode:
         count = db.delete(TABLE_NAME,
                           selection,
                           selectionArgs);
         break;
       default:
        throw new IllegalArgumentException
                           ("Unknown URI " + uri);
  }
  getContext().getContentResolver().notifyChange(uri, null);
  return count;
}
```

Implement methods (3/4)


```
@Override
public Uri insert(Uri uri, ContentValues values) {
   long rowID = db.insert(TABLE_NAME, "", values);

   if (rowID > 0) {
      Uri _uri =
           ContentUris.withAppendedId(CONTENT_URI, rowID);
      getContext().getContentResolver()
                  .notifyChange(_uri, null);
      return _uri;
   }

   throw new SQLException("Failed to add a record into " + uri);
}
```

Implement methods (3/4)


```
@Override
public Cursor query(Uri uri, String[] projection,
                    String selection, String[] selectionArgs,
                    String sortOrder) {
 SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
 qb.setTables(TABLE_NAME);
 switch (uriMatcher.match(uri)) {
      case uriCode:
          qb.setProjectionMap(values);
          break;
      default:
          throw new IllegalArgumentException("Unknown URI " + uri);
 }
 if (sortOrder == null || sortOrder == "") {
      sortOrder = name;
 }
  Cursor c = qb.query(db, projection, selection,
                  selectionArgs, null, null, sortOrder);
  c.setNotificationUri(getContext().getContentResolver(), uri);
  return c;
}
```

Add element to the database

New element to the database

```
public void onClickAddName(View view) {
  ContentValues values = new ContentValues();
  values.put(MyProvider.name, ((EditText)
             findViewById(R.id.txtName)).getText().toString());
  Uri uri = getContentResolver()
            .insert(MyProvider.CONTENT_URI, values);
}
```

Modify the AndroidManifest.xml

```
<provider
  android:name=".MyProvider"
  android:authorities=
        "com.example.admin.mycontactapplication.MyProvider"
  android:exported="true"
  android:multiprocess="true">
</provider>
```

Client (1/2)

Implement LoaderManager.LoaderCallbacks<Cursor>

```
CursorLoader cursorLoader;

public void onClickDisplayNames(View view) {
    getSupportLoaderManager().initLoader(1, null,  this);
}

@Override
public Loader<Cursor>
onCreateLoader(int arg0, Bundle arg1) {
   cursorLoader =
      new CursorLoader(this, Uri.parse("content://
com.example.admin.mycontactapplication.MyProvider/cte"),
                       null, null, null, null);
  return cursorLoader;
}
```

Client (2/2)

Implement LoaderManager.LoaderCallbacks<Cursor>

```
@Override
public void onLoadFinished(Loader<Cursor> arg0,
                           Cursor cursor) {
    cursor.moveToFirst();
    StringBuilder res = new StringBuilder();
    while (!cursor.isAfterLast()) {
      res.append("\n"
          + cursor.getString(cursor.getColumnIndex("id"))
          + "_"
          + cursor.getString(cursor.getColumnIndex("name")));
      cursor.moveToNext();
    }
  resultView.setText(res);
}
```

Summary

Content Providers

- Are used on databases
  - **But not necessarily**
- Efficient way to share informations between threads
  - **permissions can be fixed**
- May be used by a single application to provide an abstraction layer

Access is done through ContentResolver

- Predefined ones: CursorLoader, ...
- URI must be known

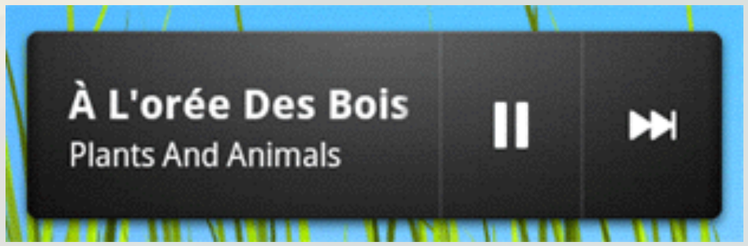
You can now build your own database

SCIENCES SORBONNE UNIVERSITÉ

Widgets

Renaultt@lrde.epita.fr

Description



Miniature Application

- embedded in other application (homescreen for instance)
- that receive periodically updates
- that offer the best GUI for BroadcastReceivers
  - **One can define a widget that count the number of received SMS**

Non-technical details

- **A widget may be associated to an activity that helps its configuration**
  - Refresh rate
  - Appearance
  - Informations to display

- **The widget must be not too small or too large**

- **For the installation**
  - Application -> Widgets -> drag-and-drop on the home screen

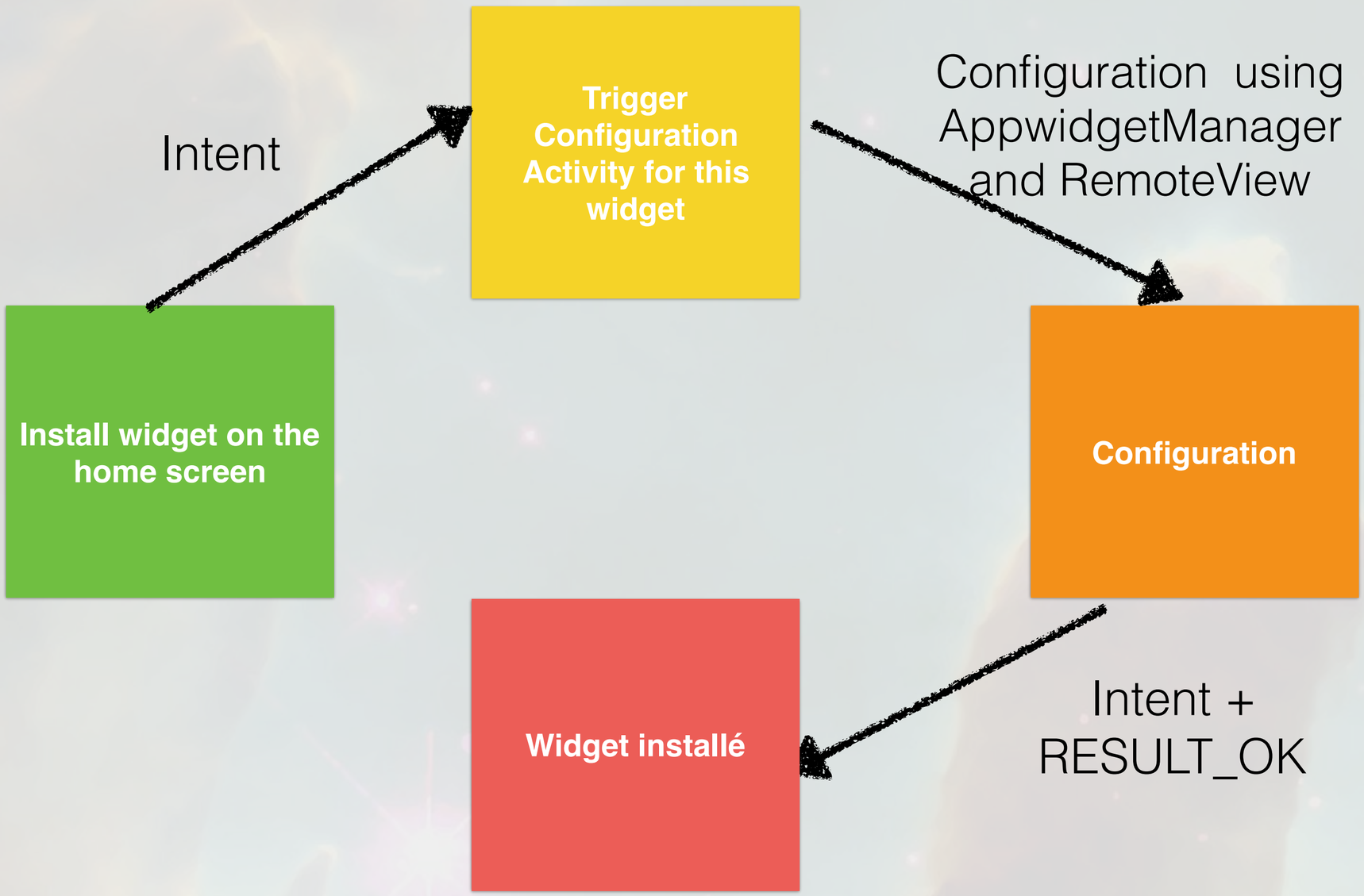
- **<u>Multiple instances of the same widgets can exist at the same time!</u>**

Widgets Lifecycle

Widgets works by callbacks

onEnabled()	Called when installed on the HomeScreen (first instance only)
onDeleted()	Called when a widget is removed from the HomeScreen
onDisabled()	Called when the last widget is removed from the HomeScreen
onUpdate()	Called foreach updates. An identifier helps to detect which instance is concerned

Lifecycle Details



Defining a widget

Modify AndroidManifest.xml

```
<receiver android:name="MyExampleAppWidgetProvider" >
 <intent-filter>
 <action android:name=
          "android.appwidget.action.APPWIDGET_UPDATE" />
 </intent-filter>
 <meta-data android:name="android.appwidget.provider"
          android:resource="@xml/my_example_appwidget_info" />
</receiver>
```

- MyExampleAppWidgetProvider:
  - **Entry point for the widget**
- xml/my_example_appwidget_info:
  - **configuration file for the widget**

Widget's Configuration

Configuration File

```
<?xml version="1.0" encoding="utf-8"?>
<appwidget-provider
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:minWidth="40dp"
    android:minHeight="40dp"
    android:updatePeriodMillis="86400000"
    android:initialLayout="@layout/my_example_appwidget"
    android:resizeMode="horizontal|vertical"
    android:widgetCategory="home_screen|keyguard">
</appwidget-provider>
```

Updates cannot be less than 10 000 milliseconds!

Defining the Main class

Option 1:

- Use a BroadcastReceiver

Option 2:

- Use an AppWidgetProvider
  - Facilities to build widgets
  - Parse automatically relevant fields of the Intent
  - Call hook methods with extras
  - Load the GUI

```
public class MyExampleAppWidgetProvider
        extends AppWidgetProvider {
}
```

- Note: this widgets does nothing except loading its UI

Widget's Configuration Activity (1/2)

Modify AndroidManifest.xml

```
android:configure=
        "com.example.admin.widgetapplication.MainActivity"
```

This activity will be triggered automatically when installing the widget

Think to declare this activity sensible to widget configuration in AndroidFile.xml

```
<intent-filter>
  <action android:name=
            "android.appwidget.action.APPWIDGET_CONFIGURE"/>
</intent-filter>
```

Widget's Configuration Activity (2/2)

Get the ID of the widget

```
private int mAppWidgetId;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Intent intent = getIntent();
    Bundle extras = intent.getExtras();
    if (extras != null) {
        mAppWidgetId = extras.getInt(
                AppWidgetManager.EXTRA_APPWIDGET_ID,
                AppWidgetManager.INVALID_APPWIDGET_ID);
    }
}
```

How to instanciate a new Widget?

Create a RemoteView and specify the layout

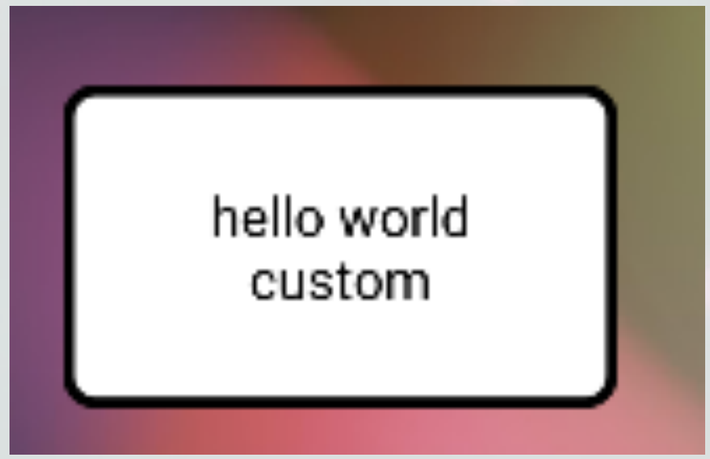
```
AppWidgetManager appWidgetManager =
        AppWidgetManager.getInstance(getApplicationContext());

RemoteViews views = new RemoteViews(
                    getApplicationContext().getPackageName(),
                    R.layout.my_example_appwidget_custom);

appWidgetManager.updateAppWidget(mAppWidgetId, views);
```

Notify the widget that the configuration is done and finish the configuration activity

```
Intent resultValue = new Intent();
resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID,
                     mAppWidgetId);
setResult(RESULT_OK, resultValue);
finish();
```

How to build friendly widgets?

Rounded Widgets are more aesthetic



- Create a file res/drawable/rounded.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android">
    <solid   android:color="#FFFFFF"/>
    <stroke  android:width="3dip" android:color="#B1BCBE" />
    <corners android:radius="10dip"/>
    <padding android:left="0dip" android:top="0dip"
             android:right="0dip" android:bottom="0dip" />
</shape>
```

- and set this as background for the widget layout!

Summary

Widgets are often a betterment for you application

- Quick access to informations
  - **Can be installed in the HomeScreen**
- Can handle buttons to trigger other android components
- Mix well with BroadcastReceivers
  - **Easy to build a counter for some system events**

Guidelines

- (70-n*30) for a cell in the home screen
- Android is providing existing shapes for widgets (among the others)

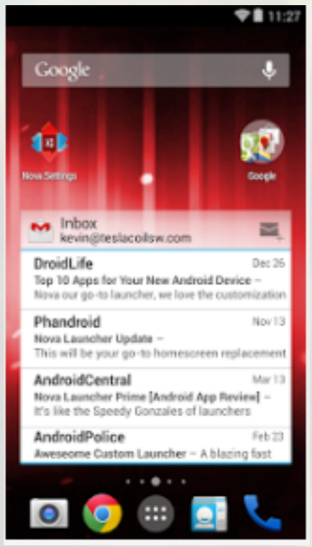
SCIENCES SORBONNE UNIVERSITÉ

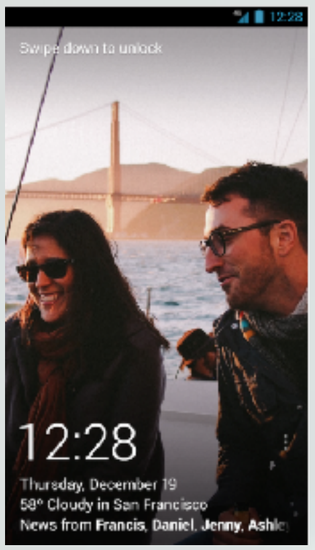
Packages and HomeScreen

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Home Screen





Goals

- Displays Widgets
- Provides access to applications
- Configurable by the user

Offers a screen that display "important" informations

- An important information depend of the user

How to Access to Installed Application?

PackageInstaller

- Interface for installing applications
- Use PackageManager to manage applications

PackageManager

- Linux Daemon
- During the install of an application
  - **Open the APK**
  - **Grab all valuable information**
- Applications are installed in directories
  - **/system/app: the preinstalled applications**
  - **/data/app: user applications**
  - **/data/data/<app_name>: data of each application**

Lifecycle

/init.rc

installd service
UID=root

Zygote
UID=root

/dev/socket/installd

system Server (UID=system)

Activity Manager | Package Manager | Account Manager | etc..

Installer
com.android.server.pm

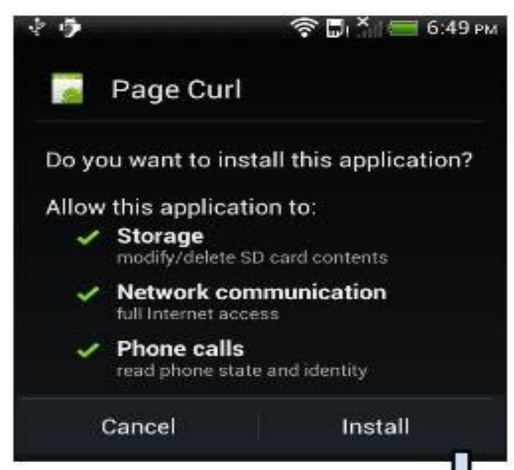
AppDirObServer
PackageManagerService

4. Install Package Process

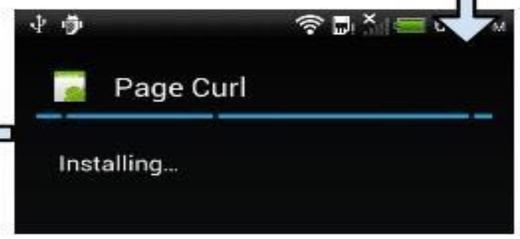
- Waiting
- Add a package to the queue for the installation process
- Determine the appropriate location of the package installation
- Determine installation Install / Update new
- A copy of the apk file to a given directory
- Determine the UID of the app
- Request to installd daemon process
- Create the application directory and set permissions
- Excision of dex code to the cache directory
- To reflect and packages.list / system / data / packages.xml the latest status
- Broadcast to the system along with the name of the effect of the installation is complete package Intent.ACTION_PACKAGE_ADDED:
If the new (Intent.ACTION_PACKAGE_REPLACED): the case of an update

1. Uri to PackageInstaller

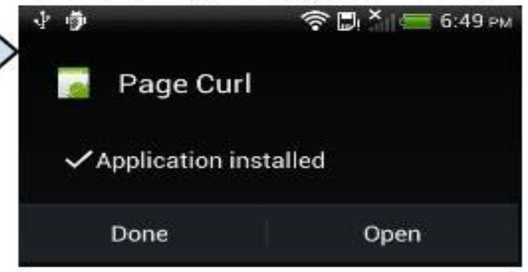
2. PackageInstallerActivity



3. InstallAppProgress



5. InstallAppProgress->Finish



PackageManager Details

Access to a lot of informations

- QueryBroadcastReceiver:
  - **What are the BroadcastReceiver that can reply to an Intent?**
- QueryIntentActivities:
  - **What are the Activities that can reply to an Intent?**
- QueryIntentService:
  - **What are the Servic that can reply to an Intent?**
- GetApplicationEnabled Settings:
  - **Read permission of an Application**
- SetApplicationSettings:
  - **Modify existing permissions**
- …

Accessing to Application's Basic Informations


```
public class AppDetail {
   CharSequence label;  CharSequence name; Drawable icon;
}
private PackageManager manager;
private List<AppDetail> apps;

private void loadApps(){
   manager = getPackageManager();
   apps = new ArrayList<AppDetail>();
   Intent i = new Intent(Intent.ACTION_MAIN, null);
   i.addCategory(Intent.CATEGORY_LAUNCHER);
   List<ResolveInfo> availableActivities =
                        manager.queryIntentActivities(i, 0);
   for(ResolveInfo ri:availableActivities){
      AppDetail app = new AppDetail();
      app.label = ri.loadLabel(manager);
      app.name = ri.activityInfo.packageName;
      app.icon = ri.activityInfo.loadIcon(manager);
      apps.add(app);
   }
```

Define a GUI for displaying this information

Build a simple View

- Here a button that will launch a listView on click

... and modify AndroidManifest.xml

```
<activity
    android:name=".MainActivity"
    android:label="@string/app_name"
    android:theme=
        "@android:style/Theme.Wallpaper.NoTitleBar.Fullscreen"
    android:launchMode="singleTask"
    android:stateNotNeeded="true" >
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.HOME" />
    <category android:name=
              "android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

The "Activity" HomeScreen

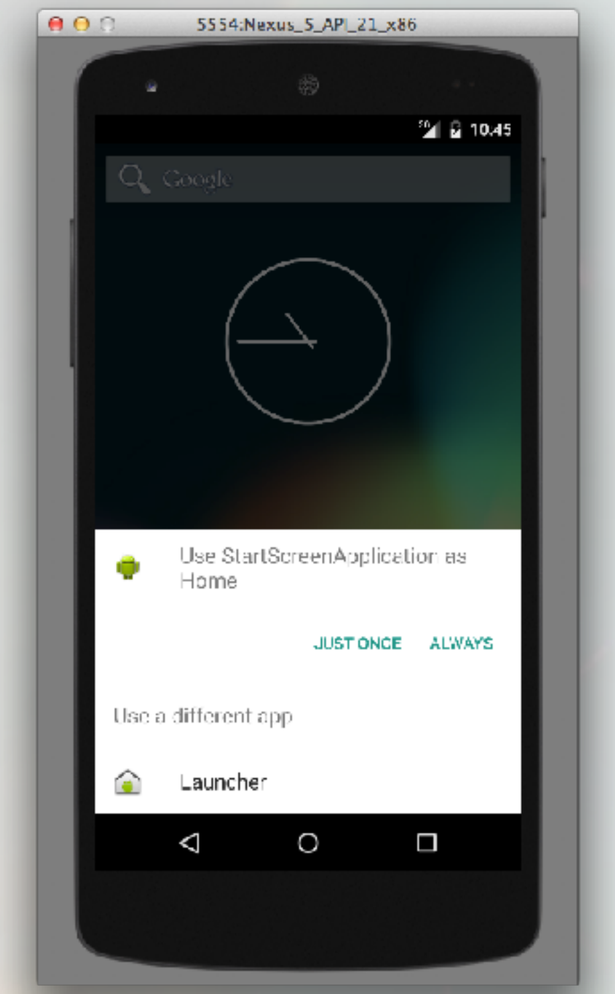
Build a simple App to access all existing Apps

- Fill the list view with some information

```
private ListView list;
private void loadListView(){
    list = (ListView)findViewById(R.id.apps_list);
    ArrayAdapter<AppDetail> adapter =
       new ArrayAdapter<AppDetail>(this,  R.layout.list_item, apps) {
        @Override
       public View getView(int position, View convertView, ViewGroup parent) {
          if(convertView == null){
           convertView = getLayoutInflater().inflate(R.layout.list_item, null);
         }
         ImageView appIcon =
                 (ImageView)convertView.findViewById(R.id.item_app_icon);
         appIcon.setImageDrawable(apps.get(position).icon);
         TextView appLabel =
                 (TextView)convertView.findViewById(R.id.item_app_label);
         appLabel.setText(apps.get(position).label);
         TextView appName =
(TextView)convertView.findViewById(R.id.item_app_name);
         appName.setText(apps.get(position).name);
        return convertView;
      }
  };
```

Setup the Home Screen

- When you launch the Application, you are asked to decide if you want to change your default home screen



Summary

- PackageManager helps to get informations about installed applications

- This informations are useful when we want to build an HomeScreen

- Defining Lock-Screen is possible
  - Same mechanism
  - Register to Events:
    - **BOOT_COMPLETED: to trigger lock-screen after a reboot**
    - **ACTION_SCREEN_OFF, ACTION_SCREEN_ON: to trigger lock-screen activation**
    - **... other events**
  - Be careful with security
    - **You have to propose security schemes**

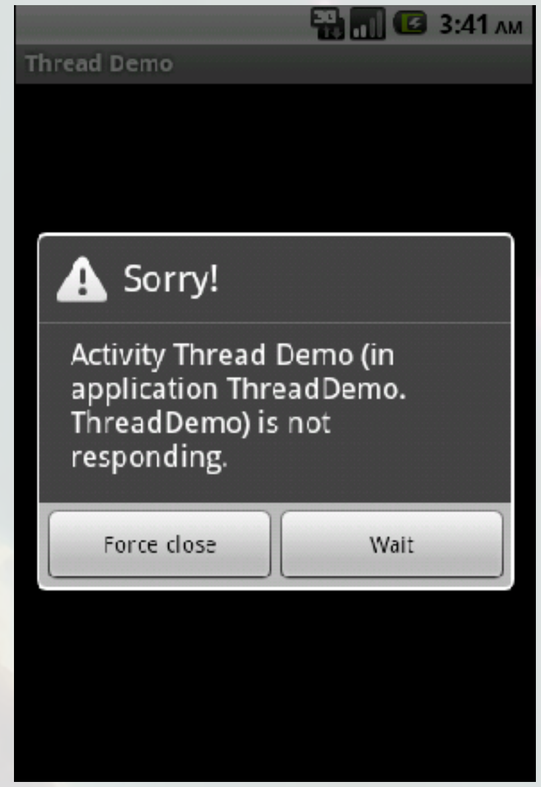
SCIENCES SORBONNE UNIVERSITÉ

Asynchronous Tasks

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Technical Considerations



In an Application, only the UI Thread can update the GUI

- Must **NOT** be blocked

For long processing, 2 options

- Build a new thread that will perform computation
  - **You have to communicate with UI thread if you want to display the result**
  - **may be hard**
  - **Not in this chapter**
- Use asynchronous tasks
  - **Dedicated component for performing background operation with GUI updates**

Asynchronous Tasks

Ease manipulation of the UI thread

- Background operation
- Builtin callbacks to update the GUI
  - **Useful for progress bar for instance**

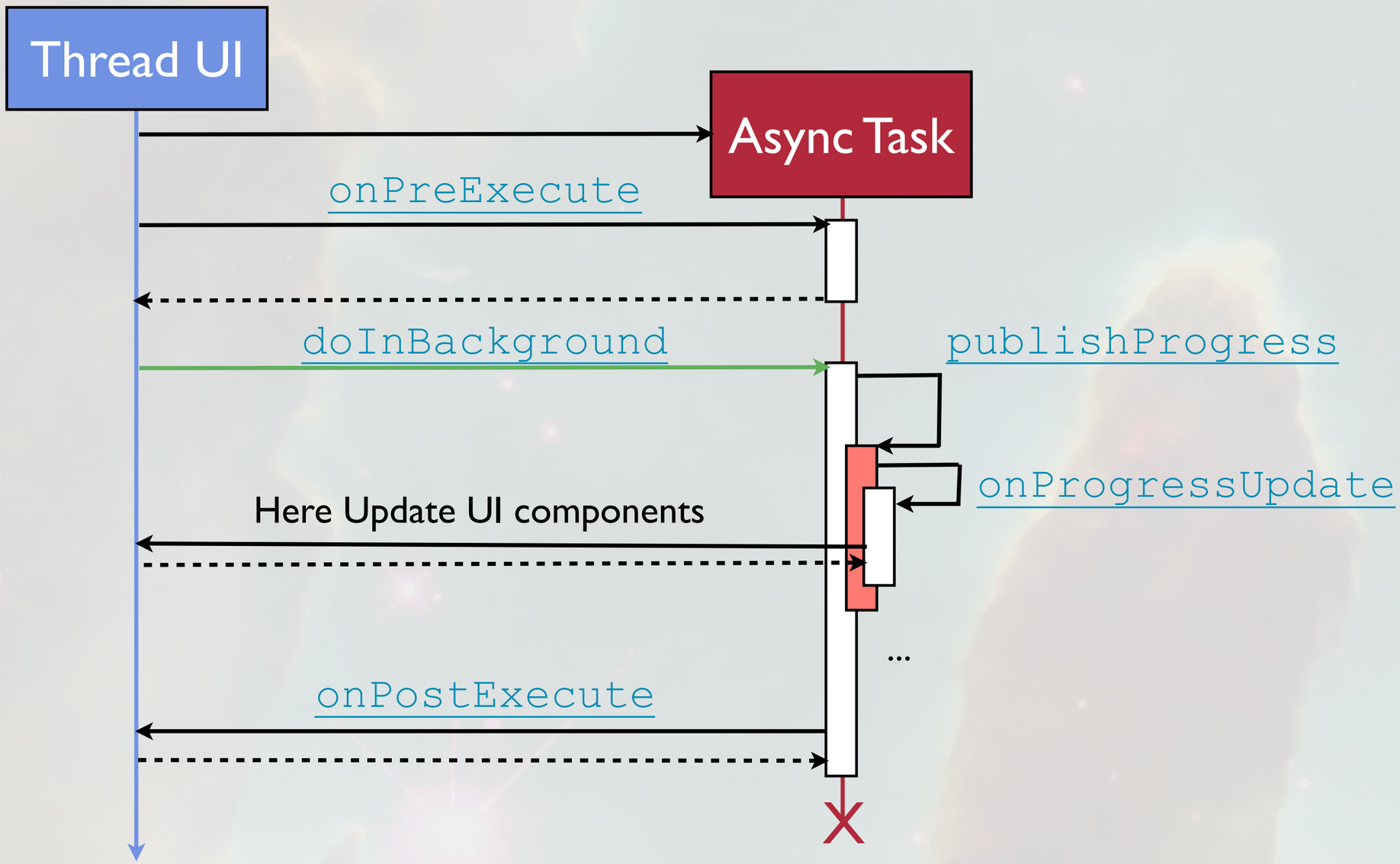
MUST NEVER REPLACE THREADS

- Should not be used for more than few seconds operations

Use three parameters

- Param:
  - **The type of the parameter used for the computation**
- Progress:
  - **The unit used to notify a progress**
- Results:
  - **The type of the result of an asynchronous task**

Lifecycle


Thread UI
Async Task
onPreExecute
doInBackground
publishProgress
onProgressUpdate
Here Update UI components
...
onPostExecute

AsyncTask and ProgressBar (1/2)


```
public class BigCompute
                    extends AsyncTask<Void, Integer, Void> {
  Context mContext;
  ProgressBar mProgressBar;

  BigCompute(Context c, ProgressBar p){
      mContext = c;
      mProgressBar = p;
  }

  @Override
  protected void onPreExecute() {
      Toast.makeText(mContext, "OnPreExecute",
                     Toast.LENGTH_SHORT).show();
      super.onPreExecute();
  }
  @Override
  protected void onPostExecute(Void aVoid) {
      super.onPostExecute(aVoid);
  }
```

AsyncTask and ProgressBar (2/2)


```java
@Override
protected void onProgressUpdate(Integer... values) {
    super.onProgressUpdate(values);
    mProgressBar.setProgress(values[0]);
}
```

```java
@Override
protected Void doInBackground(Void... params) {
    for (int progress = 0; progress < 100; ++progress){
        try{
            Thread.currentThread().sleep(1000);
        }catch (Exception e){}
        publishProgress(progress);
    }
    return null;
}
```

Building an AsyncTask

Instantiation

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  Button b = (Button) findViewById(R.id.button_launch);
  b.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        BigCompute bc = new BigCompute(getApplicationContext(),
              (ProgressBar)findViewById(R.id.progressBar));
        bc.execute();
      }
 });
}
```

AsyncTask and Rotation

AsyncTask does not support rotation as-is

Possible workarounds

- Define the asynchronous task in a Fragment an Apply a setRetainedInstance
- Associate AsyncTasks to Services
- Relaunch the AsyncTask

No best solutions

Summary

Asynchronous Tasks

- Easy way to release UI Thread
- Already predefined handlers
- Not for more than few seconds computing
- Well suited for mixing with ProgressBar

Mix bad with rotation

- Even if some solution exist

Should never replace threads

SCIENCES SORBONNE UNIVERSITÉ

Threads

Renaultⓐlrde.epita.fr

Overview

An Android Application is executed by a process

- A thread, called **main thread** or **UI thread**, is in charge of updating GUI

One thread per component

- The UI Thread of the component at the top of the back stack runs
- Thread UI manage callbacks

An application performing a lot of computing must use different threads

Rules

- 1 - Do not block UI Thread
- 2 - Only UI Thread can modify UI
  - Android Toolkit UI is not thread safe

```
public void onClick(View v) {
  new Thread(new Runnable() {
    public void run() {
      Bitmap b =
        loadImageFromNetwork("http://example.com/image.png");
      mImageView.setImageBitmap(b);
    }
  }).start();
}
```

How to update GUI then ?

To update GUI from another Thread

- Use Asynchronous tasks
- Use dedicated methods that take a thread as parameter
  - **Activity.onRunUI(Runnable)**
  - **View.post(Runnable)**
  - **View.postDelayed(Runnable, long)**

```
public void onClick(View v) {
 new Thread(new Runnable() {
     public void run() {
        final Bitmap bitmap =
            loadImageFromNetwork("http://example.com/image.png");
        mImageView.post(new Runnable() {
             public void run() {
                 mImageView.setImageBitmap(bitmap);
             }
        });
     }
   }).start();
```

Message Queue & Looper

- **A thread holds a message queue**
  - For all action and callback to perform later

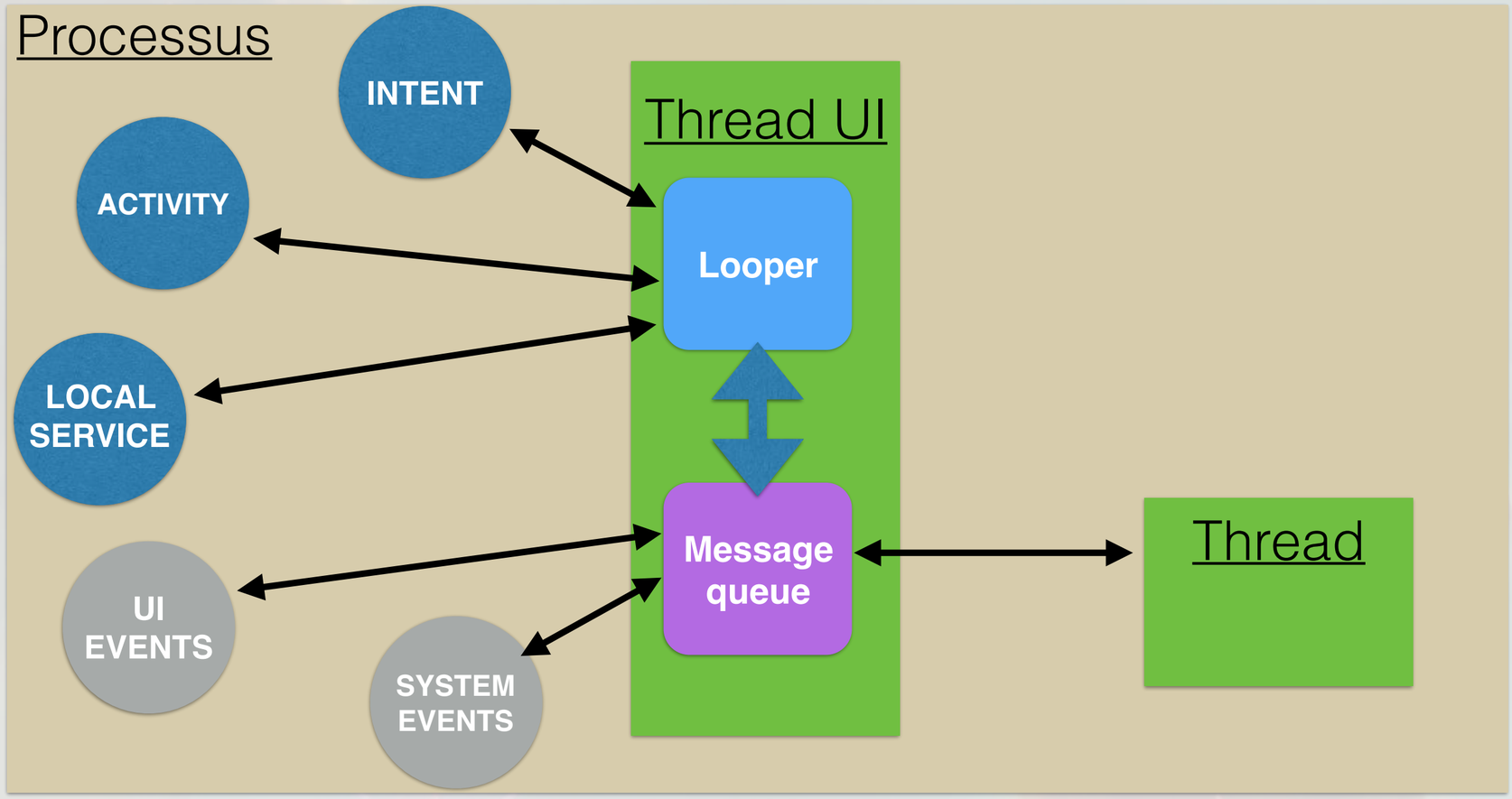
- **This queue is thread-safe**

- **Messages from this queue will be flushed by the Looper**
  - When a message is received, the looper treat it
  - To do so, handlers are used

- **By default, only the UI Thread have a looper and a message queue**

How does the looper work?

- We can force the UI Thread to do something thanks to the looper



Creating its own Looper

We can build a callback looper in every thread

- To do so, use Handlers

```
class LooperThread extends Thread {
  public Handler mHandler;
  public void run() {
    //Initialize the current thread as a looper.
    Looper.prepare();

    //instance a Handler of the current thread
    mHandler = new Handler() {
       // process incoming messages here
      public void handleMessage(Message msg) {
      }
    };

    //Run the message queue in this thread.
    Looper.loop();
}
```

Define Handler in the UI Thread

The handler is associated to a given thread

```
private ThreadCompute mThread;
private Handler mHandler;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mHandler = new Handler(Looper.getMainLooper()) {
        @Override
        public void handleMessage(Message inputMessage) {
            Toast.makeText(getApplicationContext(),
                inputMessage.toString(), Toast.LENGTH_SHORT)
                .show();
        }
    };
    mThread = new ThreadCompute();
    mThread.start();
}
```

Define Handler in the UI Thread

The handler is associated to a given thread

```
void notifyUI(){
    Message completeMessage =  mHandler.obtainMessage();
    completeMessage.sendToTarget();
}
```

```
class ThreadCompute extends Thread {
    @Override
    public void run() {
        try {
            Thread.sleep(10000);
            notifyUI();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
};
```

Priorities

THREAD_PRIORITY_AUDIO	THREAD_PRIORITY_URGENT_AUDIO
THREAD_PRIORITY_BACKGROUND	THREAD_PRIORITY_LOWEST
THREAD_PRIORITY_DISPLAY	THREAD_PRIORITY_URGENT_DISPLAY
THREAD_PRIORITY_FOREGROUND	THREAD_PRIORITY_MOST_FAVORABLE
THREAD_PRIORITY_LESS_FAVORABLE	….

Summary

- **Only the UI Thread can modify the UI**

- **If another component want to modify the UI, it has to trigger an action on the UI Thread**
  - with AsyncTask
  - with predefined methods
  - with handler

- **An application can handle multiple threads**
  - onPrepare should be called to setup the looper

- **Knowing how to manage threads is important**

SCIENCES SORBONNE UNIVERSITÉ

Battery Management

F. Kordon - Sorbonne Université - CC2018

Renault@lrde.epita.fr

Goals and Problems

- **Battery is a sensitive component in mobile devices**
  - Think to it when writing applications
  - The application must adapt to battery level
  - Restrict interaction with networks

- **If all applications were sensitive to battery problem, the lifetime of a charge will no longer be a problem**

- **How to build such an application?**

Monitor the Battery Status

BatteryManager broadcast information about battery

- Prefer updates when the device is charging

```
IntentFilter ifilter =
        new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
Intent batteryStatus = context.registerReceiver(null, ifilter);

// Are we charging / charged?
int status =
    batteryStatus.getIntExtra(BatteryManager.EXTRA_STATUS, -1);
boolean isCharging =
    status == BatteryManager.BATTERY_STATUS_CHARGING ||
    status == BatteryManager.BATTERY_STATUS_FULL;

// How are we charging?
int chargePlug =
    batteryStatus.getIntExtra(BatteryManager.EXTRA_PLUGGED, -1);
boolean usbCharge =
    chargePlug == BatteryManager.BATTERY_PLUGGED_USB;
boolean acCharge =
    chargePlug == BatteryManager.BATTERY_PLUGGED_AC;
```

Detect the exact level of the battery

Exact level

```
int level = batteryStatus
                  .getIntExtra(BatteryManager.EXTRA_LEVEL,-1);
int scale = batteryStatus
                  .getIntExtra(BatteryManager.EXTRA_SCALE, -1);
float batteryPct = level / (float)scale;
```

Modify AndroidManifest.xml

```
<receiver android:name=".PowerConnectionReceiver">
  <intent-filter>
    <action android:name=
          "android.intent.action.ACTION_POWER_CONNECTED"/>
    <action android:name=
          "android.intent.action.ACTION_POWER_DISCONNECTED"/>
  </intent-filter>
</receiver>
```

Detect only important variations

Modify AndroidManifest.xml

```
<intent-filter>
  <action android:name=
           "android.intent.action.ACTION_BATTERY_LOW"/>
  <action android:name=
           "android.intent.action.ACTION_BATTERY_OKAY"/>
</intent-filter>
```

- ACTION_BATTERY_LOW
  - **the battery is low, you should stop energy consumer activities**
- ACTION_BATTERY_OKAY
  - **the battery is OK**

Detecting Docks

You can trigger when the device is docked

- Car, Desk, etc. : to perform specific actions like swapping networks, silent mode, etc

```
IntentFilter ifilter =
                    new IntentFilter(Intent.ACTION_DOCK_EVENT);
Intent dockStatus = context.registerReceiver(null, ifilter);
int dockState = battery.getIntExtra(EXTRA_DOCK_STATE, -1);

boolean isDocked =
                    dockState != Intent.EXTRA_DOCK_STATE_UNDOCKED;
boolean isCar = dockState == EXTRA_DOCK_STATE_CAR;
boolean isDesk= dockState == EXTRA_DOCK_STATE_DESK ||
                    dockState == EXTRA_DOCK_STATE_LE_DESK ||
                    dockState == EXTRA_DOCK_STATE_HE_DESK;
```

Summary

Managing the battery is really important

- to have responsible behavior
- to modify the behavior of the application

Trace also connectivity to optimize battery

- CONNECTIVITY_CHANGE: helps to detect such situations
- Can be combined with connectivity manager

Some receiver can be deactivated runtime

- using PackageManager
- to avoid extra energy consumptions

SCIENCES SORBONNE UNIVERSITÉ

Networks

Renaultᵃ@lrde.epita.fr

Goals

- **Nowadays, most applications require an access to the internet**
  - to advertise
  - to update data
  - to publish on social network
  - …

- **Android provides frameworks for that**
  - Classical HTTP clients
  - Connectivity Manager
  - Volley Framework
  - …

Network Connection

Modify AndroidManifest.xml

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


Choose an HTTP Client

- HttpURLConnection:
  - **Most used component**
  - **Compress Cache**
  - **Post-Froyo applications**
- HttpClient:
  - **Before Froyo and Eclair**
  - **Less configurable**

Test Connectivity

ConnectivityManager

- Check all possible sources (GPS, Wifi, ...)
- Send Intents when connectivity changes

```
ConnectivityManager connMgr = (ConnectivityManager)
                getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo networkInfo = connMgr.getActiveNetworkInfo();
if (networkInfo != null && networkInfo.isConnected()) {
    // Network available ...
} else {
    // Network not available ...
}
```

- isConnected:
  - **check if a connection exists**
- getActiveNetworkInfo:
  - **informations about existing networks**

How to download a Webpage? (1/2)

Use asynchronous task

```
private class DownloadWebpageTask
        extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... urls) {
        // params comes from the execute() call:
        // params[0] is the url.
        try {
            return downloadUrl(urls[0]);
        } catch (IOException e) {
            return "Unable to retrieve URL (maybe invalid).";
        }
    }

    // onPostExecute displays the results of the AsyncTask.
    @Override
    protected void onPostExecute(String result) {
    }
```

How to download a Webpage? (2/2)


```java
private String downloadUrl(String myurl) throws IOException {
    InputStream is = null;
    // Only display the first 500 characters of the retrieved web page content.
    int len = 500;
    try {
        URL url = new URL(myurl);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setReadTimeout(10000 /* milliseconds */);
        conn.setConnectTimeout(15000 /* milliseconds */);
        conn.setRequestMethod("GET");
        conn.setDoInput(true);
        // Starts the query
        conn.connect();
        int response = conn.getResponseCode();
        is = conn.getInputStream();
        // Convert the InputStream into a string
        String contentAsString = readIt(is, len);
        return contentAsString;

    // Makes sure that the InputStream is closed after the app is
    // finished using it.
    } finally {
        if (is != null)
            is.close();
    }
}
```

Volley Framework

Google project dedicated to Android

- Automatic request scheduling
- Simultaneous communications
- Caches
- Request's priority
- Easy-configuration

Adapted for applications

- with a massive use of RPC
- using massively structured datas

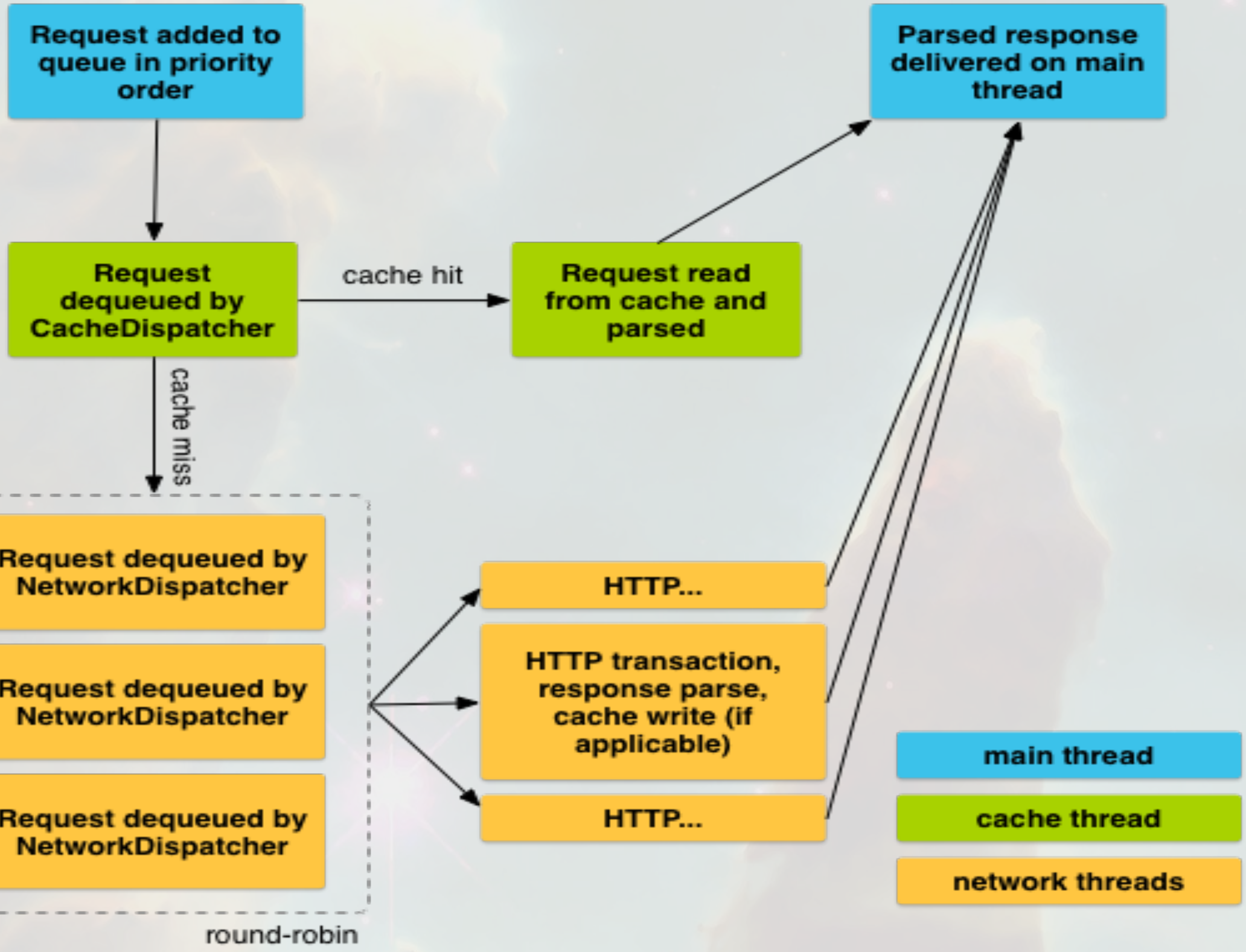
Ready-to-use component

- in graddle: 'compile com.mcxiaoke.volley:library:1.0.6'

Send/Receive Request


```
RequestQueue queue = Volley.newRequestQueue(this);
String url ="http://www.google.com";

// Request a string response from the provided URL.
StringRequest stringRequest =
    new StringRequest(Request.Method.GET, url,
                      new Response.Listener() {
        @Override
         public void onResponse(Object response) {
            // the response without using AsyncTask!
        }
       }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            //      Error!
        }
    });

// Add the request to the RequestQueue.
queue.add(stringRequest);
```

Request Lifecycle



Summary

- **Many ways to connect to network**
  - Classical HTTP clients
  - Modern Frameworks

- **Do not have long computing in the UI Thread**
  - Otherwise the application will crash

- **Think to check connectivity before sending a request**

- **Manage connectivity all around you application**

SCIENCES SORBONNE UNIVERSITÉ

Sensors

Renault@lrde.epita.fr

Description

Sensors

- are device dependent
- produce raw data

Many sensors may be available on devices

- Based on user's movements
  - **Accelerometer, Gyro, rotation sensors, ...**
- Based on the environment
  - **Temperature, barometer, ...**
- Based on positions
  - **Magnetometers, GPS, ...**

Sensor Framework

SensorManager	Helps to calibrate and detect constants
Sensor	An instance of a given sensor
SensorEvent	Information stored after an event on the sensor
SensorEventListener	Listener on the sensor

Detecting Device Sensors

List available sensors

```
SensorManager mSensorManager =
    (SensorManager) getSystemService(Context.SENSOR_SERVICE);
List<Sensor> deviceSensors =
    mSensorManager.getSensorList(Sensor.TYPE_ALL);
for (Sensor s: deviceSensors)
    Toast.makeText(getApplicationContext(), s.toString(),
                   Toast.LENGTH_SHORT).show();
```

Types of Sensors

- Sensor.TYPE_GYROSCOPE
- Sensor.TYPE_LINEAR_ACCELERATION
- Sensor.TYPE_GRAVITY
- Sensor.TYPE_ALL

Using Light Sensor (1/2)


```
public class MainActivity extends ActionBarActivity
                          implements SensorEventListener {
 private SensorManager mSensorManager;
 private Sensor mLight;

 @Override
 public final void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mSensorManager =
      (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    mLight = mSensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
}

@Override
public final void onAccuracyChanged(Sensor sensor,
                                    int accuracy)  {
       // Do something here if sensor accuracy changes.
}
```

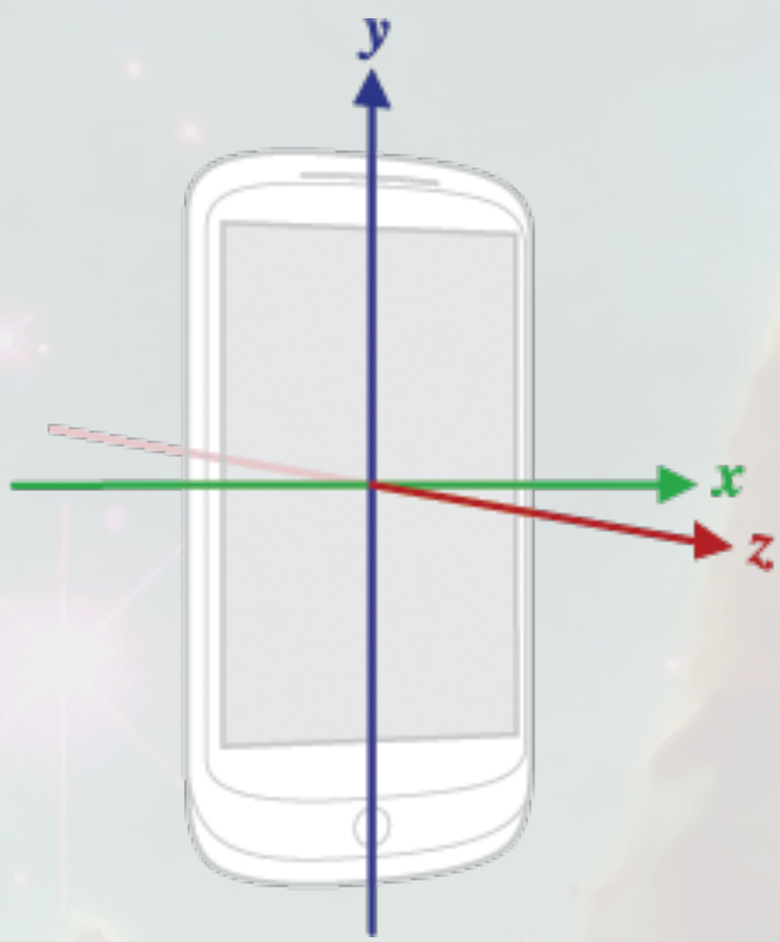
Using Light Sensor (2/2)


```java
@Override
public final void onSensorChanged(SensorEvent event) {
  // The light sensor returns a single value.
  // Many sensors return 3 values, one for each axis.
  float lux = event.values[0];
  TextView v = (TextView) findViewById(R.id.display);
  v.setText("LUX : " + lux );
}
@Override
protected void onResume() {
  super.onResume();
  mSensorManager.registerListener(this, mLight,
                                  SensorManager.SENSOR_DELAY_NORMAL);
}
@Override
protected void onPause() {
  super.onPause();
  mSensorManager.unregisterListener(this);
}
```

Motion Sensor

Works similarly

- Register to SensorEventListener in onResume
- Unregister in onStop
- Register for accelerometer event (for instance)
- onSensorChange provides informations
  - **x in event.values[0]**
  - **y in event.values[1]**
  - **z in event.values[2]**



Runtime Detection of Sensors

- Sensors can be plugged and unplugged anytime
  - Sensors may be broken!

- Detect sensors at runtime

```
private SensorManager mSensorManager;
   ...
mSensorManager =
    (SensorManager) getSystemService(Context.SENSOR_SERVICE);
if (mSensorManager.getDefaultSensor(Sensor.TYPE_PRESSURE)
     != null){
       // Success! There's a pressure sensor.
}
else {
   // Failure! No pressure sensor.
}
```

Summary

Do not forget to declare in AndroidManifest.xml

```
<uses-feature android:name=
                  "android.hardware.sensor.accelerometer"
              android:required="true"
/>
```

Multiple sensors are available on devices BUT

- Choose low refreshing rate to be friendly with battery

All sensors use the same interface

- one can register to all sensors

SCIENCES SORBONNE UNIVERSITÉ

Peer2Peer

Renault@lrde.epita.fr

Main Goals

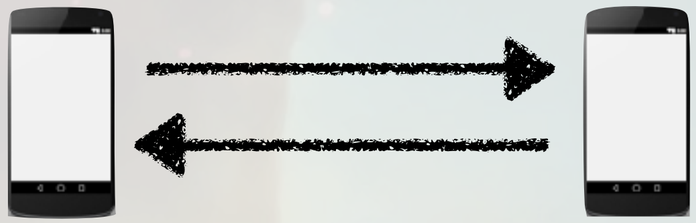
Use local network to exchange information between multiple devices

- Chats
- Content Sharing Applications
- LAN Games

How does it work?

- A device declares a service on the local network
- Look for devices connected to the network
- Peer-2-Peer connection

In this kind of connection devices are services and clients



Connection to local network

Multiple local networks

- NFC (Near Field Communication)
  - **Small distance 10 cm**
  - **Contactless payment**
- Bluetooth
  - **Medium distance 10 m**
  - **Connection with wearable**
  - **Low speed**
- Wifi
  - **Large distance**
  - **High speed**
  - **Internet**

Here we focus on P2P over Wifi

WifiP2Manager

Methods to interact with the Wifi device and to detect and connect with other peer

initialize()	Register a service on a Wifi network
connect()	Start a P2P connection with another device
discoverPeers()	Lookup for other peers
createGroup()	Construct a group with the current device as the group owner
removeGroup()	Remove the current group

How to declare a network service?

Modify AndroidManifest.xml

- ACCESS_WIFI_STATE
  - access to wifi informations
- CHANGE_WIFI_STATE
  - modify Wifi connectivity for the device
- INTERNET
  - allows the application to open sockets

```
<uses-permission
    android:required="true"
    android:name="android.permission.ACCESS_WIFI_STATE"/>
<uses-permission
    android:required="true"
    android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission
    android:required="true"
    android:name="android.permission.INTERNET"/>
```

Registering a Service


```
int SERVER_PORT = 9000;
WifiP2pManager mManager;
WifiP2pManager.Channel channel;
private void startRegistration() {
    Map record = new HashMap();
    record.put("listenport", String.valueOf(SERVER_PORT));
    record.put("buddyname", "John doe (on "
              + android.os.Build.MODEL + ")" );
    record.put("available", "visible");
    WifiP2pDnsSdServiceInfo serviceInfo = WifiP2pDnsSdServiceInfo
            .newInstance("_test", "_presence._tcp", record);
    mManager.addLocalService(channel, serviceInfo,
                             new WifiP2pManager.ActionListener() {
        @Override
        public void onSuccess() {
        }
        @Override
        public void onFailure(int arg0) {

        }
    });
}
```

Detecting existing service (1/3)


```
final HashMap<String, String> buddies =
                new HashMap<String, String>();
private void discoverService() {
    WifiP2pManager.DnsSdTxtRecordListener txtListener =
        new WifiP2pManager.DnsSdTxtRecordListener() {
            @Override
             /* Callback includes:
              * fullDomain: full domain name:
              *                 e.g "tt._ipp._tcp.local."
              * record: TXT record dta as a map of key/value pairs.
              * device: The device running the advertised service.
              */
            public void onDnsSdTxtRecordAvailable(
                    String fullDomain, Map record,
                WifiP2pDevice device) {
                 buddies.put(device.deviceAddress,
                         (String) record.get("buddyname") +
                  "--" + (String) record.get("available"));
    }
};
```

Detecting existing service (2/3)


```
  WifiP2pManager.DnsSdServiceResponseListener servListener =
                new WifiP2pManager.DnsSdServiceResponseListener() {
    @Override
    public void onDnsSdServiceAvailable(String instanceName,
                                        String registrationType,
                                        WifiP2pDevice
resourceType){

    // Update the device name with the human-friendly version
from
    // the DnsTxtRecord, assuming one arrived.
    resourceType.deviceName = buddies
            .containsKey(resourceType.deviceAddress) ?
               buddies.get(resourceType.deviceAddress) :
               resourceType.deviceName;
    }
};

 mManager.setDnsSdResponseListeners(channel,
                                    servListener, txtListener);
```

Detecting existing service (3/3)


```
WifiP2pDnsSdServiceRequest serviceRequest =
                    WifiP2pDnsSdServiceRequest.newInstance();
mManager.addServiceRequest(channel,
        serviceRequest,
        new WifiP2pManager.ActionListener() {
            @Override
            public void onSuccess() {
                // Success!
            }
            @Override
            public void onFailure(int code) {
                // Command failed.
                // Check for P2P_UNSUPPORTED, ERROR, or BUSY
            }
        });
    mManager.discoverServices(channel, new WifiP2pManager.ActionListener() {
        @Override
        public void onSuccess() {
        }
        @Override
        public void onFailure(int reason) {

        }
    });
}
```

Summary

- **We can now discover, create and connect to services**

- **To exchange with a device**
  - just open a socket and exchange data

- **Over other network P2P communication works the same way**
  - Just read the documentation

- **Think to remove your service when you leave the Wifi or the application**

SCIENCES SORBONNE UNIVERSITÉ

Advertising

Renault@lrde.epita.fr

Advertising

What for?

- Most of Android Applications are free
- advertising is one way to have money

Paid Applications can also embed advertising

- 15% to 20% more gains

Advertising must be controlled

- Limit the impact on the advertising
- Control and restrict according to your users

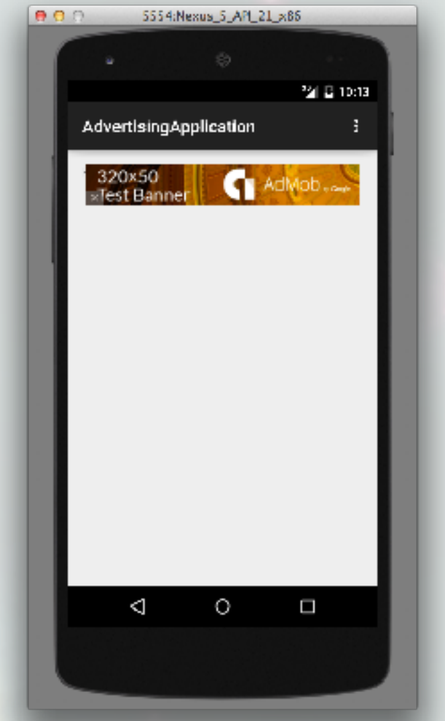
Choose your advertising network

- AdMob, DoubleClickForPublisher, AdWhirl, MobClix, MobFox, MobPub...

AdMob

Widely used service on Android

- More than 700 000 application use it
- Reports for the developers
- Easy filtering



Two kind of advertising

- Banners
  - **A banner is displayed on the top of the screen**
  - **Small impact on the user experience**
- Interstitial
  - **Full screen HTML5**
  - **The user choose to close the ad**
  - **Usually displayed when swapping activities**

Setup (1/2)

Register into AdMob

- Declare then each advertising
- And get AD_UNIT_ID for each

Modify AndroidManifest.xml

- Internet permission

```
<uses-permission
      android:name="android.permission.INTERNET"/>
<uses-permission
      android:name=« android.permission.ACCESS NETWORK STATE"/>
```

- Clickable advertising

```
<activity android:name="com.google.android.gms.ads.AdActivity"
   android:configChanges="keyboard|keyboardHidden|orientation|
         screenLayout|uiMode|screenSize|smallestScreenSize"
   android:theme="@android:style/Theme.Translucent" />
```

Setup (2/2)

Modify AndroidManifest.xml

- Connect to GooglePlay

```
<meta-data
      android:name="com.google.android.gms.version"
      android:value="@integer/google_play_services_version"/>
```

Modify build.graddle

```
compile 'com.google.android.gms:play-services:6.5.+'
```

Declaring a Banner

Integrate AD_UNIT_ID into res/values/string.xml

```
<string name="banner_ad_unit_id">
        ca-app-pub-XXX/YYY
</string>
```

Modify main layout to add the ad

```
<com.google.android.gms.ads.AdView
      android:id="@+id/adView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_centerHorizontal="true"
      ads:adSize="BANNER"
      ads:adUnitId="@string/banner_ad_unit_id">
</com.google.android.gms.ads.AdView>
```

Trigger a Banner (1/2)


```
private AdView mAdView;
private AdRequest adRequest;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mAdView = (AdView) findViewById(R.id.adView);
    adRequest = new AdRequest.Builder()
    //.addTestDevice(AdRequest.DEVICE_ID_EMULATOR)
    //.tagForChildDirectedTreatment(true)
    //.setGender(AdRequest.GENDER_FEMALE)
    //.setBirthday(new GregorianCalendar(1985, 1, 1).getTime())
            .build();
    mAdView.loadAd(adRequest);
}
/** Called when leaving the activity */
@Override
public void onPause() {
    if (mAdView != null)    mAdView.pause();
    super.onPause();
}
```

Trigger a Banner (2/2)


```java
/** Called when returning to the activity */
@Override
public void onResume() {
    super.onResume();
    if (mAdView != null) {
        mAdView.resume();
    }
}

/** Called before the activity is destroyed */
@Override
public void onDestroy() {
    if (mAdView != null) {
        mAdView.destroy();
    }
    super.onDestroy();
}
```

Lifecycle

Interface com.google.android.gms.ad.AdListener

- onAdLoaded
  - triggered when the ad is loaded
- onAdOpened
  - triggered when the ad is open
- onAdClosed
  - triggered when the ad is closed
- onAdLeftApplication
  - triggered when the ad is dismissed
- onAdFailedToLoad
  - triggered when a problem occurs during the loading of an app

Interstitial Advertising

Work the same!

- Instantiation

```
interstitial = new InterstitialAd(this);
interstitial.setAdUnitId("ca-app-pub-XXX/YYY");
adRequest = new AdRequest.Builder().build();

// Begin loading your interstitial.
interstitial.loadAd(adRequest);
displayInterstitial();
```

- Trigger the Ad

```
// Invoke displayInterstitial() when you are ready
// to display an interstitial.
public void displayInterstitial() {
   if (interstitial.isLoaded())  interstitial.show();
   else Toast.makeText(this, "Ad did not load",
                       Toast.LENGTH_SHORT).show();
}
```

Summary

Easy to integrate into your application

To note

- Do not overload your application with ads
- Choose your ad network
- Define the kind of ad you want to have
- Log the lifecycle of your activity

Free-to-play game use a lot of ads