

Android

Programmation

Sylvain Jubertie
sylvain.jubertie@univ-orleans.fr

1 Logs

2 Intents

3 Stockage

4 Capteurs

Organisation du module

8 semaines

- Semaines 1-7 : Cours/TD/TP
- Semaine 8 : Cours/examen écrit/soutenance de projets

1 Logs

2 Intents

3 Stockage

4 Capteurs

Logs

La classe `android.util.Log` permet d'ajouter des messages dans les logs exploitables à partir de la commande `adb logcat`.

Les logs peuvent être de différents types :

```
Log.v( tag , message ); // verbose
Log.d( tag , message ); // debug
Log.i( tag , message ); // information
Log.w( tag , message ); // warning
Log.e( tag , message ); // error
Log.a( tag , message ); // assert
```

L'argument `tag` (généralement le nom de l'activity) est le critère permettant de filtrer les logs.

Exemple

- `Log.e("MyActivity", "File_not_found.");`
- `Log.i("MyActivity", "Connecting_to_...");`

Filtrage

- Uniquement les messages correspondant au tag "MyActivity" :
`adb logcat MyActivity:* *:S`
- Uniquement les erreurs correspondant au tag "MyActivity" :
`adb logcat MyActivity:E *:S`

`*:S` permet de supprimer les messages ne correspondant pas au critère.

1 Logs

2 Intents

3 Stockage

4 Capteurs

Intents

Les *intents* permettent de faire communiquer des *activities*. Par exemple une application de gestion de contacts peut permettre de démarrer l'application de téléphonie pour composer le numéro d'un contact.

Les *intents* sont implantées par la classe `android.content.Intent`. Des informations sur les *activities* susceptibles d'être appelées dans une application doivent être ajoutées dans le fichier `AndroidManifest.xml`.

Exemple: appel à une autre *activity* de l'application

```
public class MainActivity extends Activity
{
    ...
    // Evènements sur un bouton.
    public void startAnotherActivity(View view)
    {
        Log.i(" UserActivity", " button_pressed" );
        Intent intent = new Intent(this, SubActivity.class);
        startActivity( intent );
    }
}
```

Exemple: code de l'*activity* appelée

Code de base de l'autre *activity* :

```
public class SubActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.another);
    }
}
```

Pour l'instant, pas d'échange de données...

AndroidManifest.xml correspondant

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android=...
    ...
    <application ...
        <activity android:name="MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action
                    android:name="android.intent.action.MAIN"/>
                <category
                    android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
        <activity android:name="SubActivity"
            android:label="@string/subactivity_name"
            android:parentActivityName="MainActivity">
        </activity>
    </application>
```

Passage de données entre *activity*

Il est possible de passer des messages dans l'*intent* par un système de clés-valeurs.

Envoi d'une information :

```
intent.putExtra( String key , ... message );
```

Récupération de l'information :

```
intent.get...Extra( String key );
```

Côté émetteur

```
public class MainActivity extends Activity
{
    ...
    public void startAnotherActivity(View view)
    {
        Intent intent = new Intent(this, SubActivity.class);
        String message = ((EditText)findViewById(R.id.et0))
                        .getText().toString();
        intent.putExtra("text", message);
        startActivity(intent);
    }
}
```

Coté récepteur

```
public class SubActivity extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.another2);
        Intent intent = getIntent();
        String message = intent.getStringExtra("text");
        ((EditText)findViewById(R.id.et0)).setText(message);
    }
}
```

Application Camera

- 1 Lancement de l'application Camera depuis l'appli :
méthode `startActivityForResult`
- 2 Prise de photo avec l'*activity* Camera
- 3 Récupération de l'image dans l'appli :
surcharge de la méthode `onActivityResult`

Bouton de prise de photo

...

```
public void takePhoto(View view) {  
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    startActivityForResult(intent, 0);  
}
```


Surcharge

@Override

```
protected void onActivityResult(int requestCode ,
                                int resultCode ,
                                Intent intent) {

    if (resultCode != 0) {
        Bundle extras = intent.getExtras();
        bitmap = (Bitmap) extras.get("data");
        ImageView image_view =
            (ImageView) findViewById(R.id.image_view);
        image_view.setImageBitmap(bitmap);
    }
    else {
        Toast.makeText(getApplicationContext(), "Cancel.",
                        Toast.LENGTH_SHORT).show();
    }
}
```

1 Logs

2 Intents

3 **Stockage**

4 Capteurs

Stockage de données

Suivant la taille, le type des données et la manière de les manipuler, on peut utiliser différentes options :

- fichiers : stockage interne, externe,
- système clé-valeurs (fichiers .ini, .json),
- base de données,
- cloud,
- ...

Fichiers sur stockage interne

- Accès privé par défaut : accès uniquement par l'application.
- Supprimés à la désinstallation de l'application.

Écriture

```
FileOutputStream fos = openFileOutput("filename"  
                                     , Context.MODE_PRIVATE);  
  
String s = "abcdef";  
fos.write(s.getBytes()); // Écriture d'octets  
fos.close();
```

Lecture

```
FileInputStream fis = openFileInput("filename");  
StringBuffer str = new StringBuffer("");  
byte[] buffer = new byte[1024];  
int n;  
while( n = fis.read( buffer )) {  
    str.append( new String( buffer , 0, n ) );  
}  
fis.close();
```

SharedPreferences

Système clé-valeur intégré par défaut.

Utile pour stocker de petites données.

Exemple : login, adresse web, IP/port serveur, dernier numéro composé, ...

Exemple : chargement au lancement de l'Activity

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

    // Récupération du fichier de préférences par défaut.
    SharedPreferences settings =
        getPreferences(MODE_PRIVATE);

    // Récupération de la chaîne associée à la clé nommée "text".
    // Si la clé n'existe pas, la chaîne "empty" est retournée.
    String text = settings.getString("text", "empty");

    // La chaîne récupérée est mise dans une view pour affichage.
    editview = (EditText)findViewById(R.id.editview);
    editview.setText(text);
}
```

Exemple : sauvegarde à l'arrêt de l'Activity

```
@Override
public void onStop()
{
    super.onStop();

    // Récupération du fichier de préférences par défaut.
    SharedPreferences settings =
        getPreferences(MODE_PRIVATE);
    // Récupération de l'Editor pour modifier les entrées.
    SharedPreferences.Editor editor = settings.edit();
    // La chaîne est associée à la clé "text".
    editor.putString("text",
        ( editview.getText() ).toString() );
    // Effectue les modifications.
    editor.commit();
}
```


SQLite

- Base de données intégrée par défaut.
- Requêtes SQL.
- Exemple : liste de contacts, bookmarks, ...

Méthode recommandée

Héritage de la classe `android.database.sqlite.SQLiteOpenHelper`.

Tout le code SQL doit être centralisé dans cette classe.

- constructeur : création ou ouverture de la base si existante.
- méthode `onCreate` : appelée si la base n'existe pas.
- méthode `onUpgrade` : appelée si changement de version de base (migration).
- autres méthodes pour effectuer les différentes requêtes.

Exemple simple

2 classes :

- [DBOpenHelper](#) : gère la DB + interface pour accès à la DB.
- [SQLiteDemoActivity](#) : utilise les méthodes de [DBOpenHelper](#) pour accéder à la base. PAS DE SQL ICI.

Classe DBOpenHelper

```
public class DBOpenHelper extends SQLiteOpenHelper {
    private static final String DB_NAME = "dbname";
    private static final int DB_VERSION = 2;
    private SQLiteDatabase db;
    DBOpenHelper(Context context) {
        super(context, DB_NAME, null, DB_VERSION);
        db = getWritableDatabase();
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("create _table_"
            + DB_TABLE_NAME
            + " (_id _integer _primary
            + "key _autoincrement, _value _text _not _null);")
    }
    ...
}
```

Classe DBOpenHelper suite...

```
...  
// Insertion d'une chaîne dans la table.  
public void insertValue(String value) {  
    // La chaîne n'est pas directement ajoutée dans la base.  
    // Il faut passer par la création d'une structure intermédiaire ContentValues.  
    ContentValues content = new ContentValues();  
    // Insertion de la chaîne dans l'instance de ContentValues.  
    content.put("value", value);  
  
    // Insertion dans la base de l'instance de ContentValues contenant la chaîne.  
    db.insert(DB_TABLE_NAME, null, content);  
}  
...
```

Classe `DBOpenHelper` fin.

```
// Récupération des chaînes de la table.  
public List<String> getValues() {  
    List<String> list = new ArrayList<String>();  
    String[] columns = {"value"};  
    // Exécution de la requête pour obtenir les chaînes  
    // et récupération d'un curseur sur ces données.  
    Cursor cursor = db.query(DB_TABLE_NAME, columns  
                                , null, null, null  
                                , null, null);  
    // Curseur placé en début des chaînes récupérées.  
    cursor.moveToFirst();  
    while (!cursor.isAfterLast()) {  
        // Récupération d'une chaîne et insertion dans une liste.  
        list.add(cursor.getString(0));  
        cursor.moveToNext(); // Passage à l'entrée suivante.  
    }  
    cursor.close(); // Fermeture du curseur.  
    return list;  
}
```

Classe SQLiteDemoActivity

```
public class SQLiteDemo extends Activity {  
    private DBHelper dbHelper;  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        dbHelper = new DBHelper(this);  
        dbHelper.insertValue("abcdef"); //Insertion  
        List<String> list = dbHelper.getValues();  
        // Concaténation des chaines pour affichage  
        String values = "";  
        for(int i = 0 ; i < list.size() ; ++i) {  
            values += list.get(i) + " ";  
        }  
        ((TextView)findViewById(R.id.tv0)).setText(values);  
    }  
}
```

1 Logs

2 Intents

3 Stockage

4 Capteurs

Sensors

- Capteurs de position: GPS, boussoles, magnétomètres, ...
- Capteurs de mouvement: accéléromètres, gyroscopes, ...
- Capteurs environnementaux: luminosité, température, ...

Framework général

- **SensorManager** : gestionnaire de capteurs
- **Sensor** : classe pour tous les types de capteurs
- **SensorEvent** : classe représentant un évènement généré par un capteur
- **SensorEventListener** : interface pour la gestion des évènements

Utilisation d'un capteur

- 1 Indication des capteurs utilisés dans le *manifest*
- 2 Implantation de l'interface [SensorEventListener](#)
- 3 Récupération du [SensorManager](#)
- 4 Récupération du [Sensor](#) suivant le type de capteur souhaité
- 5 Surcharge des méthodes [onSensorChanged](#) et [onAccuracyChanged](#)
- 6 Enregistrement/désenregistrement du capteur dans les méthodes [onStart](#), [onPause](#), [onResume](#) suivant besoins

Accéléromètre

Capteur utilisé uniquement si l'application est au premier plan :

- Enregistrement dans la méthode [onResume](#)
- Désenregistrement dans la méthode [onPause](#)

Modification du manifest

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

Exemple

```
public class AccelerometerDemoActivity extends Activity
    implements SensorEventListener
{
    private SensorManager sensorManager;
    private Sensor accelerometer;
    ...
    @Override
    public void onCreate(Bundle savedInstanceState) {
        ...
        sensorManager = (SensorManager)
            getSystemService(SENSOR_SERVICE);
        accelerometer = sensorManager.getDefaultSensor(
            Sensor.TYPE_ACCELEROMETER);
        ...
    }
}
```

Exemple

```
@Override  
protected void onPause()  
{  
    super.onPause();  
    sensorManager.unregisterListener(this);  
}
```

```
@Override  
protected void onResume()  
{  
    super.onResume();  
    sensorManager.registerListener(  
        this ,  
        accelerometer ,  
        SensorManager.SENSOR_DELAY_UI  
    );  
}
```

Exemple

```
// Changement de la précision du capteur.
public void onAccuracyChanged( Sensor sensor ,
                               int accuracy )
{

// Dés qu'un nouvel évènement est produit par le capteur.
public void onSensorChanged( SensorEvent event )
{
    // Filtrage si plusieurs capteurs.
    if ( event.sensor.getType()
        == Sensor.TYPE_ACCELEROMETER )
    {
        ... event.values[0] ...;
        ... event.values[1] ...;
        ... event.values[2] ...;
    }
}
```

GPS

- `LocationManager` + `LocationListener` + `Location`
- Capteur utilisé même en arrière plan pour suivre le déplacement :
 - Enregistrement dans la méthode `onCreate`
 - Désenregistrement dans la méthode `onStop`

Modification du manifest

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


Exemple

```
public class GPSActivity extends Activity
    implements LocationListener
{
    LocationManager locationManager;
    ...
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        ...
        locationManager = (LocationManager)
            getSystemService(LOCATION_SERVICE);

        locationManager.requestLocationUpdates(
            LocationManager.GPS_PROVIDER, 0, 0, this
        );
        ...
    }
}
```

Exemple

```
@Override
protected void onPause()
{
    super.onPause();
    locationManager.removeUpdates(this);
}

@Override
protected void onResume()
{
    super.onResume();
    locationManager.requestLocationUpdates(
        locationManager.GPS_PROVIDER, 0, 0, this);
}
```

Exemple

```
public void onProviderDisabled( String s)
{

}

public void onProviderEnabled( String s)
{

}

public void onStatusChanged( String provider , int status , B
{

}

public void onLocationChanged( Location location )
{
    ... location.getLongitude() ...;
    ... location.getLatitude() ...;
    ... location.getAltitude() ...;
}
```