TD Android

Les Intents

Romain Raveaux
IUT de La Rochelle

Pour faire suite à un “Hello World” un peu atypique façon Google Maps, nous abordons un système important et même essentiel à Android : les `Intents`.

Android offre un système de communication trés ingénieux permettant de faire passer l’information entre `Activitys` ou plus généralement composants applicatifs . Ce système, déjà mentionné dans une partie précédente, est connu sous le nom d’`Intent`. Nous verrons donc dans ce TD comment transmettre l’information d’une `Activity` à une autre à l’aide de ce mécanisme d’`Intents`. Nous défricherons aussi la différence subtile entre `Activity` et `View` seront probablement éclairé par la lecture des lignes suivantes.

Introduction

Android sandbox les applications. Le mot “sandbox” est un anglicisme qui signifie simplement “bac à sable”. Il peu sembler étrange de parler de “bac à sable” dans un problème d’informatique mais c’est pourtant bel et bien la traduction adéquate dans le contexte. Le sandboxing est une pratique de plus en plus courante dans la téléphonie mobile qui consiste à séparer presque totalement les applications entre elles. Lorsque Android exécute une application, il restreint cette dernière à des actions bien définies (accès mémoire, accès sur les capteurs, etc…). Cette pratique permet de protéger le système au maximum en évitant de laisser des applications faire comme bon leurs semble.

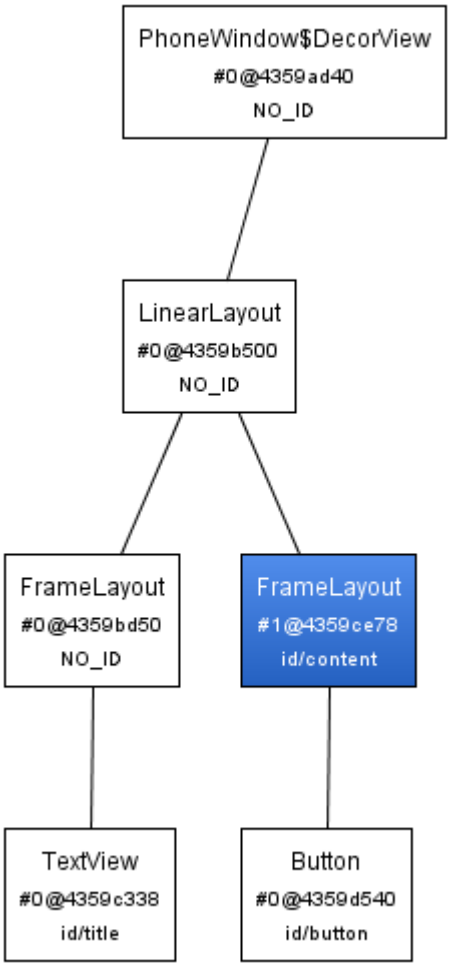
Malgré l’énorme apport sur la sécurité, le sandoxing restreint fortement la communication entre applications. C’est dans l’optique de contourner ce “problème”, que les Intents ont été conçus. Pour faire simple, un Intent est un ensemble de données qui peut être passé à un autre composant applicatif (de la même application ou non) de façon implicite (requête pour une action – lire de la musique ou scanner un code barre par exemple) ou explicite (lancement d’une classe précise).

Démarrage d’une Activity grâce aux Intents

Pour bien décrire le système des Intents, nous allons créer une petite application (sans aucun but réel, je l’accorde) composée de 2 écrans : le premier dispose d’un bouton permettant de démarrer une nouvelle Activity de façon explicite. La seconde Activity affichera simplement un champ de recherche permettant d’effectuer une recherche sur Google. Le code de cette application est disponible dans ce dossier zippé.

Démarrage d'une activité de façon explicite

Pour commencer, créons une première Activity qui affichera un unique bouton central. Le code XML ci-dessous est composé d'une unique balise Button. Cette simplicité de l'interface graphique provient du fait que l'espace disponible au développeur pour afficher son application (l'intégralité de l'écran moins la barre de tâches et la barre de nom) est disponible sous la forme d'un FrameLayout. L'image ci-dessous montre la hiérarchie des vues générées avec ce fichier XML. On aperçoit facilement (en bleu) un FrameLayout.



Les paramètres android:layout_gravity="center" et android:text="Cliquez ici pour démarrer" permettent respectivement de centrer le bouton dans sa vue parente et de définir le texte du bouton.

```
<?xml version="1.0" encoding="utf-8"?>
<Button
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="Cliquez ici pour démarrer" />
```

Notre interface graphique étant prête, il suffit maintenant de créer notre première `Activity`. Pour faciliter la compréhension, j'ai préféré directement commenter le code. N'oubliez pas d'inclure cette nouvelle `Activity` dans le fichier Manifest.xml pour qu'Android autorise son exécution.

Le manifest :

```
<?xml version="1.0" encoding="utf-8" ?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="
    fr.pm4.android.HelloIntent " android:versionCode="1" android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".EntryPoint" 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=".GoogleSearch" android:label="@string/app_name">
<intent-filter>
 <action android:name="android.intent.action.DEFAULT" />
   </intent-filter>
   </activity>
   </application>
 <uses-sdk android:minSdkVersion="3" />
   </manifest>
```

Le code de la première activity :

```
package fr.pm4.android.HelloIntent;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

/*
 * La classe EntryPoint implémente l'interface View.OnClickListener. Cela
permet ainsi
 * d'enregistrer l'activité auprès des vues pour qu'elle puisse recevoir
les évènements de "clic".
 */
public class EntryPoint extends Activity implements View.OnClickListener {

        /*
         * Cette variable permettra de conserver une référence sur le
bouton
         * de l'interface
         */
        private Button mButton;

        /*
         * La redéfinition de la méthode onCreate(Bundle) permet
d'effectuer des actions
         * supplémentaires à l'Activity de base.
         */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*
         * On créé l'interface graphique en utilisant le fichier entry.xml.
         * Ce dernier étant disponible comme ressource (dans /res/layout),
il suffit d'utiliser
         * le fichier R.java et sa sous-classe layout.
         */
        setContentView(R.layout.entry);
        /*
         * Récupère une référence sur le bouton en utilisant son
identifiant
         */
        mButton = (Button)findViewById(R.id.button);
        /*
         * On enregistre l'activité auprès du bouton pour recevoir les
évènements
         * "clic" provoqué par l'utilisateur.
         */
        mButton.setOnClickListener(this);
    }
```

```
    /*
     * La méthode onClick(View) provient de l'interface
View.OnClickListener.
     */
        @Override
        public void onClick(View v) {
                if (v == mButton) {
                        /*
                         * Nous sommes maintenant sûr que la vue ayant été
cliquée est le bouton
                         * de notre interface. Il suffit donc de créer un
nouvel Intent pour démarrer
                         * la seconde activité.
                         */
                        Intent intent = new Intent(EntryPoint.this,
GoogleSearch.class);
                        startActivity(intent);
                }
        }
}
```

Notre première Activity est maintenant terminée. Le lancement d'une nouvelle Activity de façon explicite s'effectue en deux lignes. La première, Intent intent = new Intent(EntryPoint.this, GoogleSearch.class); crée simplement un nouvel Intent dont le contexte de départ est l'Activity courante (EntryPoint.this) et l'Activity de destination se nomme GoogleSearch.

Démarrage d'une activité de façon implicite

Il ne nous reste maintenant plus qu'à définir l'Activity GoogleSearch en utilisant des principes similaires à ceux précédemment décrits. Commençons, tout d'abord par créer l'interface graphique :

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:padding="10px">

        <TextView
                android:text="Entrez la recherche à effectuer dans le champ
ci-dessous puis cliquez sur le bouton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginBottom="10px"
                android:gravity="center_horizontal" />

        <EditText
                android:id="@+id/editText"
                android:layout_marginBottom="5px"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content" />
```

```
        <ImageButton
                android:id="@+id/imageButton"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/logo_google"
                android:layout_gravity="center"></ImageButton>

</LinearLayout>
```

Il ne me semble pas très important de décrire le fichier XML précédent car il peut être considéré comme classique. Il est à noter, tout de même, une légère différence avec les fichiers XML de notre HelloWorld : les ressources de type chaines de caractères, dimensions, etc. ne sont pas externalisés dans un fichier externe. Dans notre cas cela n'a pas réellement d'impact puisque nous n'allons pas internationaliser l'application.

```
package fr.pm4.android.HelloIntent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.ImageButton;

public class GoogleSearch extends Activity implements View.OnClickListener
{

        private ImageButton mImageButton;
        private EditText mEditText;

        @Override
        protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.google);
                mImageButton = (ImageButton)findViewById(R.id.imageButton);
                mEditText = (EditText)findViewById(R.id.editText);
                mImageButton.setOnClickListener(this);
        }

        @Override
        public void onClick(View view) {
                if (view == mImageButton) {
                        final String requete =
"http://www.google.fr/search?q=" + mEditText.getText();
                        Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(requete));
                        startActivity(intent);
                }
        }
}
```

Un clic sur le bouton “Google” créé un nouvel Intent à l’aide du constructeur Intent(String,Uri). La chaine de caractère passée en paramètre est une action à effectuer (ici ACTION_VIEW : action définie par le framework qui consiste à démarrer un navigateur web sur l’Uri donnée). On démarre enfin l’activité par un simple startActivity(Intent). Une action ne définissant pas une application en particulier, Android va tenter de chercher une application s’étant définie comme capable de répondre à l’action ACTION_VIEW. C’est cette “non certitude” sur l’application à ouvrir qui explique le nom de cette méthode : implicite. Généralement le navigateur utilisé est celui inclus de base dans le téléphone.

Conclusion

Les Intents sont des composants essentiels de l’architecture Android. Le démarrage explicite ou implicite d’une Activity à l’aide d’Intents n’est qu’une petite infime partie des possibilités de ce mécanisme. Cette partie fait donc office d’introduction aux Intents et vous permet de mieux comprendre la communication entre composants applicatifs Android. Vous êtes maintenant prêts à coder vos premiers programmes à écrans multiples !

Remerciement

Ce TD a pu voir le jour grâce à l’aide de Cyril Mottier et de son livre. Développez pour Android de Cyril Mottier et Ludovic Perrier