



Developpement mobile – Les intents

AKKA ZEMMARI

ZEMMARI@LABRI.FR

Intents

- ▶ Les Intents permettent de gérer l'envoi et la réception de messages afin de faire coopérer les applications.
 - ▶ Le but des Intents est de déléguer une action à un autre composant, une autre application ou une autre activité de l'application courante.
- ▶ Classification des Intent
 - ▶ Intent explicite : indique l'identifiant du composant destinataire
 - ▶ Intent implicite : indique le type d'action demandée et optionnellement l'URI correspondante
- ▶ URI : Uniform Resource Identifier
 - ▶ Généralise la notion d'URL
 - ▶ Schéma générique = `scheme://host:port/path`
 - ▶ Schémas usuels : `http`, `mailto`, `tel`, `mms`, `geo`, `file`, ...

Intents

- ▶ Un objet Intent contient les informations suivantes:
 - ▶ le nom du composant ciblé (facultatif)
 - ▶ l'action à réaliser, sous forme de chaîne de caractères
 - ▶ les données : contenu MIME et URI
 - ▶ des données supplémentaires sous forme de paires de clé/valeur
 - ▶ une catégorie pour cibler un type d'application
 - ▶ des drapeaux (information supplémentaire)

Intent Explicite

- ▶ Intent explicite : indique l'identifiant du composant destinataire
- ▶ Demande la création d'une nouvelle activité en lui passant le nom de la classe en paramètre. Le système démarre une nouvelle instance de la classe
- ▶ `startActivity(this, ActivityTwo.class);`

- ▶ Exemple :

```
Intent login = new Intent(this, GiveLogin.class);  
startActivity(login);
```


Intent implicite

- ▶ Intent implicite : n'indique pas directement l'identifiant du composant destinataire; indique le type d'action demandée et optionnellement l'URI correspondante.
- ▶ Demande la création d'une activité capable d'effectuer l'action demandée.
- ▶ Le système cherche une application ayant la capacité à effectuer l'action demandée.
- ▶ Le système démarre une nouvelle instance de la classe.
- ▶ Exemple :

```
Intent intent= new Intent(ACTION, [uri]);  
startActivity(intent);
```


Intent implicite

```
Button b = (Button)findViewById(R.id.Button01);  
b.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        Uri telnumber = Uri.parse("tel:0248484000");  
        Intent call = new Intent(Intent.ACTION_DIAL, telnumber);  
        startActivity(call);  
    }  
});
```

Demande l'ouverture d'un service capable de composer un numéro de téléphone et lui passe l'URI correspondant au schéma reconnu
tel: <numerodetel>

Liste des actions possibles

- ▶ Plusieurs actions natives existent par défaut sur Android.
- ▶ La plus courante est l'action `Intent.ACTION_VIEW` qui permet d'appeler une application pour visualiser un contenu dont on donne l'URI.
- ▶ Les autres actions sont :
 - ▶ `ACTION_CALL` (`ANSWER`, `DIAL`) : passer/réceptionner/afficher un appel
 - ▶ `ACTION_EDIT` (`DELETE`) : éditer/supprimer une donnée
 - ▶ `ACTION_SEND` : envoyer des données par SMS ou E-mail
 - ▶ `ACTION_WEB_SEARCH` : rechercher sur internet

Data

- ▶ Il s'agit de *Uniform Resource Identifier* (URI).

- ▶ Un URI se construit comme suit :

- ▶ `Uri webpage = Uri.parse("http ://duckduckgo.com/ ?android") ;`

- ▶ `Uri tel = Uri.parse("tel :+336123456789") ;`

- ▶ `Uri geo = Uri.parse("geo :52.312405,20.918330") ;`

Type

- ▶ Le type de la donnée peut être ajouté aux données elles-mêmes pour aider Android à trouver les app les plus pertinentes pour les traiter
 - ▶ MIME Type : toujours en minuscules
 - ▶ text/html video/avi applica- tion/msword application/x-gzip image/jpeg audio/x-mpeg-3 video/mpeg application/pdf image/png etc.

Exemple 1

```
Intent emailIntent = new Intent(android.content.Intent.ACTION_SEND);
String[] recipients = new String[]{"my@email.com", ""};
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL, recipients);
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT, "Test");
emailIntent.putExtra(android.content.Intent.EXTRA_TEXT, "Message");
emailIntent.setType("text/plain");
startActivity(Intent.createChooser(emailIntent, "Send mail..."));
finish();
```

L'activité peut forcer le système à la dépiler avec finish()

Exemple 2

```
▶ Uri page = Uri.parse("http://www.google.fr");  
  Intent intent = new Intent(Intent.ACTION_VIEW);  
  intent.setData(page);  
  startActivity(intent);
```


Choix de l'application

- ▶ le mécanisme en charge de trouver toutes les app aptes à exécuter l'action requise est appelé intention resolution :
 - ▶ si une seule activité répond aux critères, elle est automatiquement lancée
 - ▶ si plusieurs activités peuvent répondre, Android construit un chooser basé sur la valeur de l'Action, sur les Data et sur le Type.
 - ▶ si aucune app présente ne peut répondre, le système génère une exception : `ActivityNotFoundException`.

Retour d'une activité

- ▶ Lorsque le bouton retour est pressé, l'activité courante prend fin et revient à l'activité précédente.
- ▶ Cela permet par exemple de terminer son appel téléphonique et de revenir à l'interface ayant initié l'appel.
- ▶ Au sein d'une application, une activité peut vouloir récupérer un code de retour de l'activité "enfant".
- ▶ On utilise pour cela la méthode
`startActivityForResult(Intent i, int requestCode)`
qui envoie un code de retour à l'activité enfant.
- ▶ Lorsque l'activité parent reprend la main, il devient possible de filtrer le code de retour dans la méthode
`onActivityResult(int requestCode, int resultCode, Intent data)`
pour savoir si l'on revient ou pas de l'activité enfant.

Retour d'une activité

- ▶ Code d'appel dans l'activité appelante

```
Intent login = new Intent(this, GivePhoneNumber.class);  
startActivityForResult(login, 48);
```

- ▶ Code de traitement par l'activité appelante

```
protected void onActivityResult(int resultCode, Intent data){  
    if (requestCode == 48){  
        Toast.makeText(this, "Coup de tel terminé", Toast.LENGTH_LONG).show()  
    }  
}
```

Filtre sur le numéro de
requête

Transfert de données

- ▶ Les Intent peuvent aussi transporter des données qu'une application veut transmettre à une autre application
- ▶ Activité appelante : `intent.putExtra(key,value);`

```
Intent login = new Intent(this, GivePhoneNumber.class);  
login.putExtra("nom", "toto");  
login.putExtra("pwd", "$$toto31");  
startActivityForResult(login,48);
```

- ▶ Activité appelée: `intent.getExtras().getXXX(key);`

```
Bundle extra = intent.getExtras();  
String nom = extra.getString("nom");  
String pwd = extra.getString("pwd");
```


Transfert de données

- ▶ Les types de base (+array) sont gérés
 - ▶ `putExtra(String key String val)`
 - ▶ `putStringArrayListExtra(String key, ArrayList<String> value)`
- ▶ Les types complexes (c-à-d les classes) doivent implémenter `Parcelable`, OU `Serializable`
- ▶ Récupération du données via
`Bundle bundle = getIntent().getExtras();`
- ▶ Récupération par la méthode `bundle.getXXX()` propre à chaque type
 - ▶ `bundle.getDouble()`
 - ▶ `bundle.getString()`
 - ▶ `bundle.getArray()`
 - ▶ ...

Retour d'une activité

- ▶ L'activité appelée peut :
 - ▶ transmettre un code de retour via `setResult (codeRetour)`
 - ▶ transmettre des données supplémentaires
 1. Construire un nouvel Intent via `new Intent ()`
 2. Ajouter des données dans l'Intent via `putExtra (String)`
 3. Envoyer le résultat via `setResult (int resultCode, Intent data)`
 4. Le code de la requête est ajouté automatiquement
 5. Le nouvel Intent est envoyé à l'application appelante

Retour d'une activité

- L'activité appelée fabrique un nouvel Intent

```
Intent intent = new Intent();  
intent.putExtra("resultat", ""+result);  
setResult(Activity.RESULT_OK, resultIntent);
```

RESULT_OK = constante
prédéfinie

- L'activité appelante récupère les données

```
public void onActivityResult(int requestCode, int resultCode, Intent  
data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    switch(requestCode) {  
        case (48) : {  
            if (resultCode == Activity.RESULT_OK) {  
                // TODO Extract the data } break; }  
        }  
    }  
}
```


Retour d'une activité

- L'activité appelée peut forcer le retour en appelant `finish()`

```
// méthode onCreate de l'activité pour implémenter
// un bouton qui « referme » l'activité
Button finish = (Button)findViewById(R.id.finish);
finish.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        setResult(50);
        finish();
    }
});
```

Code différent de
`RESULT_OK` pour signaler
un cas particulier

Intents explicites : résumé

Sans données et sans retour :

Activity A

```
Intent i = new Intent (Context,B.class)  
startActivity(i)
```

Activity B

[OnCreate ()]



Intents explicites : résumé

Avec données et résultats en retour :

Activity A

```
Intent i = new Intent(Context,B.class)
i.putExtra(key1,value)1;
startActivityForResult(i,code);
```

Activity B

```
[OnCreate()]
Intent i = getIntent()
val=i.getStringExtra(key1)
```

```
Intent resIntent = new Intent()
resIntent.putExtra(key2,val2)
setResult(returnStatus,resIntent)
```

```
onActivityResult(code,returnStatus,Intent resIntent)
val=resIntent.getStringExtra(key2,default_val)
```


TP 03 : Intents

- ▶ A récupérer ici :
 - ▶ <http://www.labri.fr/~zemmari/pam>