



Mobile Applikationen

Kommunikation

von Tobias Muth



Inhalt

- Intents
 - Empfänger von Intents
 - Intent Filter
 - Kommunikation zwischen Activities
 - Rückgabewerte
- Broadcast Receiver
 - Implementierung
 - Manifestdatei



Intents

- Nachrichten werden in Intents gekapselt
- Intents koppeln nicht nur Activities, sondern auch Services und Broadcast Receiver
- binden Komponenten einer oder mehrerer Apps zur Laufzeit
- Sie enthalten entweder abstrakte Operationsanweisungen oder,
- Sie informieren über eintretende Ereignisse
- Übermitteln Empfänger welche Aktion ausgeführt werden soll
- Liefern die hierfür benötigten Daten mit
- Kategorie eines Intents wird von Android selber gefiltert
 - beschreibt welche Komponenten ein Intent erhalten / behandeln können
 - beschreibt wie aufrufende Komponenten ggf. zu starten sind



Empfänger von Intents

- Empfänger werden durch setzen des Komponentennamens angegeben
 - Voll qualifizierter Klassenname der Zielkomponente (Activity, Service, Broadcast Receiver)
 - Paketname der App, die die Komponente enthält
- explizite Intents
 - benennen Zielkomponente
 - üblicherweise für anwendungsinternen Austausch genutzt
- implizite Intents
 - benennen kein Ziel
 - üblicherweise für Austausch mit anderen Apps genutzt



Intent Filter

- System muss ermitteln welche Komponenten am besten für Empfang / Auswertung eines Intents geeignet sind
- Können Activities / Services sowie Broadcast Receiver sein
- Inhalt des Intents wird mit jeweiligen Intent Filtern verglichen
- Intent Filter werden in die Manifestdatei eingetragen
 - beschreiben welche impliziten Intents eine Komponente empfangen / verarbeiten kann
- Wurzelelemente (z.B. `<activity />`) werden mit der Kindkomponente `<intent-filter />` versehen
 - `<intent-filter />` enthält Kindelemente die zum Filtern impliziter Intents genutzt werden
- Ohne Filter empfängt eine Komponente nur explizite Intents



Intent Filter - Action & Category

- Element `<action />` muss mindestens einmal vorhanden sein
- Speichert die Aktionen die ein empfangenes Intent auslösen soll
 - z.B Anruf tätigen, Suche im Web durchführen
- Klasse Intent hält Konstanten für diese Aktionen fest
- Konstanten beginnen mit dem Präfix `ACTION_`
- Attribut `android:name` des `<action />` Elements wird `android.intent.action.<KONSTANTENNAME>` zugewiesen
 - Konstantenname ohne das Präfix `ACTION_`
 - `<action android:name="android.intent.action.WEB_SEARCH" />`
- Kategorien von Intents legen fest an welche Art von Komponente diese übermittelt werden
- Attribut `android:name` des `<intent-filter />` Elements in Manifestdatei wird `android.intent.category` zugewiesen
 - `<category android:name="android.intent.category.BROWSABLE" />`



Kommunikation zwischen Activities

- Benutzer startet und beendet während Navigation durch die App mehrere Activities
- Activities rufen durch übergabe eines Intents an die Methode “startActivity()” Nachfolger auf

```
Intent intent = new Intent(this, SimpleActivity.class);  
startActivity(intent);
```

- !Jede Activity die mit startActivity() aufgerufen werden soll, muss in der Manifestdatei eingetragen sein!



Kommunikation zwischen Activities - implizite Intents

```
Intent in = new Intent(Intent.ACTION_INSERT,
    ContactsContract.Contacts.CONTENT_URI);
in.putExtra("finishActivityOnSaveCompleted", true);
in.putExtra(ContactsContract.Intents.Insert.NAME,
    "Max Mustermann");
in.putExtra(ContactsContract.Intents.Insert.PHONE,
    "+49 (123) 45 67 89");
in.putExtra(ContactsContract.Intents.Insert.PHONE_TYPE,
    ContactsContract.CommonDataKinds.Phone.TYPE_WORK);
startActivity(in);
```

- Methode “putExtra()” übergibt einem Intent beliebig viele Nutzdaten
- “putExtra()” überschreibt vorhandene Werte



Kommunikation zwischen Activities - implizite Intents, ContentValues

```
ArrayList<ContentValues> data = new ArrayList<>();  
...  
in.putParcelableArrayListExtra(  
    ContactsContract.Intents.Insert.DATA, data);  
ContentValues v1 = new ContentValues();  
v1.put(ContactsContract.Contacts.Data.MIMETYPE,  
    ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);  
v1.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "123");  
v1.put(ContactsContract.CommonDataKinds.Phone.TYPE,  
    ContactsContract.CommonDataKinds.Phone.TYPE_HOME);  
data.add(v1);  
ContentValues v2 = new ContentValues();  
v2.put(ContactsContract.Contacts.Data.MIMETYPE,  
    ContactsContract.CommonDataKinds.Phone.CONTENT_ITEM_TYPE);  
v2.put(ContactsContract.CommonDataKinds.Phone.NUMBER, "456");  
v2.put(ContactsContract.CommonDataKinds.Phone.TYPE,  
    ContactsContract.CommonDataKinds.Phone.TYPE_MOBILE);  
data.add(v2);
```



Rückgabewerte

- !Activities warten nicht auf Beendigung von nachgestarteten Activities!
 - Methode “startActivityResult()” nutzen
- Methode “onActivityResult()” ist für Rückgabewerte zuständig
- “startActivityResult()” wird ein Request Code als Parameter übergeben
 - Best Practice: RQ als Konstante in der aufrufenden Klasse definieren

```
private static final int RQ_INSERT_CONTACT = 1234;
```

- RQ gibt an, wer eine Activity gestartet hat

```
Intent in = new Intent(...);
```

```
...
```

```
startActivityResult(in, RQ_INSERT_CONTACT);
```




Rückgabewerte

```
@Override
protected void onActivityResult(int requestCode,
                                int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RQ_INSERT_CONTACT) {
        if (resultCode == RESULT_OK) {
            if (data != null) {
                Intent i =
                    new Intent(Intent.ACTION_VIEW,
                               data.getData());
                startActivity(i);
            }
        }
    }
}
```



Broadcast Receiver

- Reagieren auf Systemweit gesandte Nachrichten
 - z.B. niedriger Batteriestand, ausgeschalteter Bildschirm
- Haben keine Bedienoberfläche, senden allerdings Nachrichten in die Statuszeile
- Schnittstellen zu anderen Komponenten, sollten daher wenig Logik enthalten
- Kann unabhängig von anderen Komponenten einer App deaktiviert werden



Broadcast Receiver - Implementierung

```
@Override
public void onReceive(Context context, Intent intent) {
    if (Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction())) {
        // Benachrichtigung zusammenbauen
        String msg =
            DateFormat.getDateTimeInstance().format(new Date());
        Notification.Builder builder =
            new Notification.Builder(context);
        builder.setSmallIcon(
            R.drawable.ic_launcher).
            setTitle(
                context.getString(R.string.app_name)).
            setTextColor(msg).
            setWhen(System.currentTimeMillis());
        Notification n = builder.build();
        // anzeigen
        NotificationManager m = (NotificationManager) context
            .getSystemService(Context.NOTIFICATION_SERVICE);
        m.notify(ID, n);
    }
}
```



Broadcast Receiver - Implementierung

- Broadcast Receiver leiten sich von Klasse `android.content.BroadcastReceiver` ab und müssen die Methode `“onReceive()”` überschreiben
- `“onReceive()”` wird aufgerufen sobald der Broadcast Receiver einen Intent empfängt
- !Nach 10 Sekunden kann Android den zugehörigen Prozess blockieren!
- !Receiver-Objekt ist nach dem Verlassen der Methode `“onReceive()”` evtl. nicht mehr vorhanden!



Broadcast Receiver - Manifestdatei

```
<receiver android:name=".BootCompletedReceiver">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED" />
    </intent-filter>
</receiver>
```

- Attribut android:name verweist auf Klasse, die den Receiver implementiert
 - Ist es nicht gesetzt, wird das Paket, welches im <manifest /> Element definiert ist verwendet
- android:enabled gibt an, ob ein Receiver instanziiert werden kann
- android:exported gibt an, ob ein Receiver Nachrichten von außerhalb der eigenen App erhalten kann
 - Hat ein Receiver keine Intent-Filter ist der Standardwert "false"
 - Hat ein Receiver Intent-Filter ist der Standardwert "true"
- android:icon gibt ein individuelles Symbol an
- android:label gibt eine individuelle Beschreibung an



Broadcast Receiver - Manifestdatei

- android:permission gibt an, welche Berechtigung Komponenten haben müssen um dem Receiver eine Nachricht senden zu können
 - ist das Attribut nicht gesetzt wird die Berechtigung, die inner des <application /> Elements gesetzt ist genutzt
- android:process gibt den Namen des Prozesses an, in welchem der Broadcast Receiver ausgeführt werden sollte
 - Kann mehreren Elementen vergeben werden um App auf mehrere Prozesse aufzuteilen



Vielen Dank für eure Aufmerksamkeit

Noch Fragen?