Corso di Laboratorio di Programmazione per Sistemi Mobile e Tablet

Docente: Dr. Mauro Dragoni

Dipartimento di Ingegneria e Scienze dell’Informazione
Anno Accademico 2022/2023

in questa lezione...

- Broadcasts
- Inviare un broadcast personalizzato
- Ricezione di broadcast
- Broadcasts e restrizioni
- Best practices

Cosa sono i Broadcasts?

- I Broadcasts sono messaggi inviati dal sistema di backend di Android o da altre applicazioni quando si verifica un evento di interesse.

- I Broadcasts sono inclusi in un oggetto di tipo `Intent`. L'oggetto `Intent` contiene i dettagli dell'evento.

- Esempio: la costante `android.intent.action.HEADSET_PLUG` quando gli auricolari vengono inseriti o tolti dallo smartphone.

Tipi di broadcasts:

- broadcasts di sistema;
- broadcasts personalizzati.

Broadcasts di sistema

I broadcasts di sistema sono dei messaggi inviati dal sistema operativo Android quando un evento di sistema, che puo' influire sul funzionamento della vostra applicazione, occorre.

Un paio di esempi:

- Un `Intent` avente azione di tipo `ACTION_BOOT_COMPLETED` viene generato quando lo smartphone viene avviato.
- Un `Intent` avente azione `ACTION_POWER_CONNECTED` viene generato quanto lo smartphone viene connesso alla rete elettrica.

Broadcasts personalizzati

- Gli broadcasts personalizzati sono quelli generati dalla propria applicazione, in modo similare a ciò che avviene con il sistema Android.

- Esempio: vogliamo informare le altre applicazioni presenti sullo smartphone che la nostra applicazione ha appena finito di scaricare dei dati, e questi sono disponibili per il loro utilizzo.

- Android fornisce tre modi per inviare un broadcast:
  - Ordered broadcast.
  - Normal broadcast.
  - Local broadcast.

Ordered Broadcast

- Gli “ordered broadcast” vengono inviati ad un ricevente per volta.
- Per inviare un “ordered broadcast” e’ necessario utilizzare il metodo `sendOrderedBroadcast()`.
- Ciascun ricevente, puo’ propagare il risultato al ricevente successivo oppure abortire il broadcast del messaggio.
- Per controllare l’ordine dei riceventi, utilizzare l’attributo `android:priority` all’interno del manifesto.
- I riceventi con la stessa priorita’ vengono chiamati in un ordine arbitrario.

Normal Broadcast

- I messaggi vengono inviati nello stesso momento a tutti i riceventi registrati in un ordine non definito, in quanto non importante.
- E’ la via piu’ efficiente per inviare un broadcast.
- I riceventi non possono propagare il risultato dell’elaborazione del messaggio e neppure interrompere il broadcast.
- Il metodo `sendBroadcast()` viene utilizzato per inviare un normale broadcast.

Local Broadcast

- Invia un messaggio di broadcast che puo’ essere recepito solamente all’interno della propria applicazione.
- Non vi e’ alcun problema legato ad aspetti di sicurezza da dover gestire in quanto non vengono istanziate delle comunicazioni inter-processo.
- Per inviare un “local broadcast”:
  - Creare un’istanza di `LocalBroadcastManager`.
  - Chiamare il metodo `sendBroadcast()` sull’istanza stessa.

```
LocalBroadcastManager.getInstance(this)
                     .sendBroadcast(customBroadcastIntent);
```

Cos’e’ un broadcast receiver?

- Un broadcast receiver (il ricevente) e’ un componente dell’applicazione.
- Deve essere registrato per essere in grado di captare i diversi tipi di broadcast che vengono inviati.
- Ogni broadcast receiver viene notificato attraverso un oggetto di tipo Intent:
  - dal sistema, quando il broadcast viene generato a seguito di una tipologia di evento per la quale la nostra applicazione e’ registrata;
  - da un’altra applicazione, inclusa la propria, nel caso il ricevente sia registrato per ricevere eventi personalizzati.

Register your broadcast receiver

I broadcast receivers possono essere registrati in due modi:

- Riceventi statici:
  - registrati all’interno del file manifesto `AndroidManifest.xml`, chiamati anche Manifest-declared receivers.

- Dynamic receivers
  - registrati utilizzando all’applicazione o tramite i `context` delle activities, chiamati anche Context-registered receivers.

Ricevere un broadcast di sistema

- A partire da Android 8.0 (API level 26), i receivers statici non sono piu’ in grado di ricevere molti broadcast.
- E’ necessario quindi utilizzare un dynamic receiver per registrarsi a tali broadcasts.
- Quindi, se vi registrate per questi broadcast tramite il manifesto, Android non sara’ in grado di inviare messaggi alla vostra applicazione.
- Alcuni broadcast sono comunque esenti da questa limitazione. Potete trovare la lista all’interno della documentazione di Android alla voce implicit broadcast exceptions.

Creare un broadcast receiver

- Ereditare la classe `BroadcastReceiver` ed effettuare l'override del metodo `onReceive()`.

- Registrare il broadcast receiver e specificare il relativo intent-filters:
  - staticamente nel manifesto;
  - dinamicamente tramite il metodo `registerReceiver()`.

Creare un broadcast receiver

In Android studio, **File** > **New** > **Other** > **BroadcastReceiver**

```
public class CustomReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Questo metodo viene eseguito quando il BroadcastReceiver
        // sta’ ricevendo un Intent generato da un broadcast.
        throw new UnsupportedOperationException(“Non implementato.");
    }
}
```

Come usare gli Intent-filters

- Gli Intent-filters specificano i tipi di intenti che un broadcast receiver puo' ricevere.
- Essi filtrano gli intenti entranti nell'applicazione in base al valore del `action` agganciato all'oggetto di tipo `Intent`.
- Per aggiungere un intent-filter:
  - all'interno del file `AndroidManifest.xml`, utilizzare il tag `<intent-filter>`.
  - all'interno di del codice Java, utilizzare l'oggetto `IntentFilter`.

Implementare il metodo onReceive()

Esempio di implementazione del metodo `onReceive()` per gestire la connessione e disconnessione dell’energia elettrica..

```
@Override
public void onReceive(Context context, Intent intent) {
    String intentAction = intent.getAction();
    switch (intentAction){
        case Intent.ACTION_POWER_CONNECTED:
            break;
        case Intent.ACTION_POWER_DISCONNECTED:
            break;
    }
}
```

Registrazione statica all’interno del manifesto

- elemento `<receiver>` all’interno del tag `<application>`.
- elemento `<intent-filter>` per registrare il receiver per specifici intenti.

```
<receiver
    android:name=".CustomReceiver"
    android:enabled="true"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.BOOT_COMPLETED"/>
    </intent-filter>
</receiver>
```

Registrazione dinamica

- Registrare il receiver all'interno dei metodi `onCreate()` o `onResume()`.

```
// Registrare il receiver usando il contensto della activity.
   this.registerReceiver(mReceiver, filter);
```

- De-registrarlo all'interno dei metodi `onDestroy()` o `onPause()`.

```
// De-registrazione del receiver
this.unregisterReceiver(mReceiver);
```

Local broadcast receiver

- Registrare un local receiver dinamicamente in quando la registrazione statica nel manifesto non e’ possibile per i local broadcasts.

REGISTRAZIONE

- Creare un’instanza di `LocalBroadcastManager`.
- Chiamare il metodo `registerReceiver()`.

```
LocalBroadcastManager.getInstance(this).registerReceiver
(mReceiver,
  new IntentFilter(CustomReceiver.ACTION_CUSTOM_BROADCAST));
```

Local broadcast receiver

- Registrare un local receiver dinamicamente in quando la registrazione statica nel manifesto non e’ possibile per i local broadcasts.

DE-REGISTRAZIONE

- Creare un’instanza di `LocalBroadcastManager`.
- Chiamare il metodo `LocalBroadcastManager.unregisterReceiver()`.

```
LocalBroadcastManager.getInstance(this)
                     .unregisterReceiver(mReceiver);
```

Implementare delle restrizioni per i broadcasts

- Implementare delle restrizioni nei propri broadcast e’ molto raccomandato.
- Un broadcast senza restrizioni puo’ sollevare dei problemi di sicurezza.
- Esempio banale: se la vostra applicazione contenente informazioni sensibili non implementa delle restrizioni sui broadcast, una seconda applicazione contenente dei malware puo’ registrarsi e ricevere i vostri dati.

Modi per implementare le restrizioni

- Se possibile, utilizzare un `LocalBroadcastManager`, il quale mantiene le informazioni all'interno della vostra applicazione evitando cosi' problemi di sicurezza.

- Utilizzare il metodo `setPackage()` utilizzando il package name desiderato. In questo modo il broadcast sara' ricevuto solamente dalle applicazioni che contengono il package name specificato.

- Permessi di accesso possono essere forzati sia da chi invia il messaggio sia da chi lo riceve.

Forzare i permessi da chi invia/riceve il messaggio

Per forzare un permesso quando si invia un broadcast:

- Fornire un permesso non-null come argomento per il metodo `sendBroadcast()`.
- Solo i riceventi che richiedo il permesso specificato tramite il tag <uses-permission> nel loro file `AndroidManifest.xml` possono ricevere broadcast.

Per forzare un permesso quando si riceve un broadcast:

- se il receiver e' stato registrato in modo dinamico, fornire un permesso non-null al metodo `registerReceiver()`.
- se il receiver e' stato registrato in modo statico, utilizzare l'attributo `android:permission` all'interno del tag `<receiver>` all'interno del file `AndroidManifest.xml`.

Best practices

- Siate sicuri che i namespace che utilizzate siano unici.
- Implementate le restrizioni nei broadcast receivers.
- Altre applicazioni possono rispondere ai broadcasts che inviate, utilizzate i permessi per controllare cio’.
- Preferite receivers dinamici a quelli statici.
- Non eseguite mai delle operazioni troppo lunghe all’interno dei vostri broadcast receiver.

Usare i broadcast (e non solo) per aggiornare la UI dal backend

- I LocalBroadcast si prestano bene come metodo per aggiornare la UI dopo l’avvenimento di eventi nel parte back-end della vostra applicazione.

- Cosa ci serve:
  - un BroadcastReceiver che cattura l’evento generale;
  - un LocalBroadcast registrato nella Activity di interesse in grado di catturare l’evento ed accedere alla UI.

Usare i broadcast (e non solo) per aggiornare la UI dal backend

```
public class MyReceiver extends BroadcastReceiver {

        @Override
        public void onReceive(Context context, Intent intent) {

                if (intent.getExtras() != null) {

                        .....
                        Intent in = new Intent("com.test.example");
                        Bundle extras = new Bundle();
                        extras.putString("com.test.example.data", myData);
                        in.putExtras(extras);
                        context.sendBroadcast(in);
                }
        }
}
```

Usare i broadcast (e non solo) per aggiornare la UI dal backend

```
public class MyActivity extends AppCompatActivity {

        private BroadcastReceiver broadcastReceiver;

        @Override
         protected void onCreate(@Nullable Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                registerReceiver();
        }

        private void registerReceiver() {
                broadcastReceiver = new BroadcastReceiver() {
                        @Override
                        public void onReceive(Context context, Intent intent) {
                                String myData = intent.getStringExtra("com.test.example.data");

                                 /* UPDATE UI */
                         }
                };
                registerReceiver(broadcastReceiver, new IntentFilter("com.test.example"));
        }

        @Override
        protected void onStop() {
                super.onStop();
                if(broadcastReceiver != null) { unregisterReceiver(broadcastReceiver); }
        }
}
```

Approfondimenti

Sul sito ufficiale:

- BroadcastReceiver Reference
- Intents and Intent Filters Guide
- LocalBroadcastManager Reference
- Broadcasts overview