BroadcastReceiver

Programming the Android Platform

Broadcast Receiver

- Base class for components that receive and react to events
  - Events are represented as Intents
  - Events are broadcast system-wide
  - Interested BroadcastReceivers receive Intent via onReceive()
- BroadcastReceivers have no user interface

Use Case

- BroadcastReceivers get registered to receive specific Intents
- Some component Broadcasts an Intent
- Android identifies appropriate recipients & delivers event by calling BroadcastReceiver.onReceive()
- Event handled in on Receive()

Registering BroadcastReceivers

- Can register in two ways
- Statically via AndroidManifest.XML
- Dynamically via Context.registerReceiver()

Static Registration

- Include <receiver> in AndroidManifest.xml

```
<application>
   <receiver receiver_specs >
      <intent-filter> event_specs </intent-filter>
   </receiver>
</application>
```

- Receiver registered at boot time or when application package is added at runtime

Static Registration (cont.)

```
<application ...>
     <activity android:name=".SimpleBroadcast" ...>  ...  </activity>
      <receiver android:name=".Receiver2">
          <intent-filter android:priority="5">
              <action android:name=
                    "course.examples.BroadcastReceiver.intent.action.TEST2">
              </action>
          </intent-filter>
      </receiver>
 </application>
 <uses-permission
     android:name="android.permission.VIBRATE"></uses-permission>
```

Dynamic Registration

- Create an IntentFilter
- Create a BroadcastReceiver
- Register BroadcastReceiver to receive Intents that match the IntentFilter using Context .registerReceiver()
- As appropriate call Context.unRegisterReceiver() to unregister BroadcastReceiver

Dynamic Registration (cont.)

```
public class SingleBroadcast extends Activity {
   public static final String CUSTOM_INTENT =
    "course.examples.BroadcastReceiver.intent.action.TEST1";
   public void onCreate(Bundle savedInstanceState) {
      ...
      registerReceiver(new Receiver1(),
                                    new IntentFilter(CUSTOM_INTENT));
    }
}
```

Event Broadcast

- Several broadcast methods supported
- Normal vs. Ordered
  - Normal: processing order undefined
  - Ordered: sequential processing in priority order
- Sticky vs. Non-Sticky
  - Sticky: Store Intent after initial broadcast
  - Non-Sticky: Discard Intent after initial broadcast
- With or without receiver permissions

Normal Broadcasts

```
//public abstract class Context …

// send Intent to interested BroadcastReceivers
void sendBroadcast (Intent intent)

// send Intent to interested BroadcastReceivers
// if they have the specified permissions
void sendBroadcast (Intent intent, String receiverPermission)
```

Normal Broadcasts (cont.)

```
public class SimpleBroadcast extends Activity {
  public static final String CUSTOM_INTENT =
    "course.examples.BroadcastReceiver.intent.action.TEST2";
  public void onCreate(Bundle savedInstanceState) {
     …
     Button button = (Button) findViewById(R.id.button);
     button.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            sendBroadcast(new Intent(CUSTOM_INTENT),
                                       android.Manifest.permission.VIBRATE);
        }
     });
…
```

Ordered Broadcasts

```
//public abstract class Context …

// send Intent to interested BroadcastReceivers in priority order
void sendOrderedBroadcast (Intent intent,
                           String receiverPermission)

// send Intent to interested BroadcastReceivers in priority order
// sender can provide various parameters for greater control
void sendOrderedBroadcast (Intent intent,
                           String receiverPermission,
                           BroadcastReceiver resultReceiver,
                           Handler scheduler,
                           int initialCode,
                           String initialData,
                           Bundle initialExtras)
```

Ordered Broadcasts (cont.)

```
public class CompoundOrderedBroadcast extends Activity {
  ...
  public static final String CUSTOM_INTENT =
         "course.examples.BroadcastReceiver.intent.action.TEST4";
  public void onCreate(Bundle savedInstanceState) {
    ...
    Button.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        sendOrderedBroadcast(new Intent(CUSTOM_INTENT),
                                    android.Manifest.permission.VIBRATE);
      }
  });
  ...
```

Ordered Broadcasts (cont.)

```
public class CompOrdBcastWithResultReceiver extends Activity {
  public void onCreate(Bundle savedInstanceState) {
  ...
    button.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        sendOrderedBroadcast(new Intent(CUSTOM_INTENT), null,
            new BroadcastReceiver() {
              public void onReceive(Context context, Intent intent) {
                System.out.println("Final Result is:" + getResultData());
              }
            },  null, 0, null, null);
      }
    });
...
```

Sticky Broadcasts

- Sticky Intents are cached by Android
  - New Intents overwrite older Intents they match
- When BroadcastReceivers are dynamically registered
  - Cached sticky Intents matching the specified IntentFilter are broadcast to the BroadcastReceiver
  - One matching sticky Intent is returned to the caller

Sticky Broadcasts (cont.)

```
//public abstract class Context …
// send sticky Intent to interested BroadcastReceivers
void sendStickyBroadcast (Intent intent)

// send sticky Intent to interested BroadcastReceivers in priority order
// sender can provide various parameters for greater control

void sendStickyOrderedBroadcast (Intent intent,
                                 BroadcastReceiver resultReceiver,
                                 Handler scheduler,
                                 int initialCode,
                                 String initialData,
                                 Bundle initialExtras)
```

- Broadcaster must have BROADCAST_STICKY permission to send sticky Intents

Intent Filter Resolution

- Similar to resolution for Activities & Services
- Some debugging tips
  - Log BroadcastReceivers that match an Intent
    - Intent.setFlag(FLAG_DEBUG_LOG_RESOLUTION)
  - List BroadcastReceivers registered to receive intents
    - Dynamic registration
      - % adb shell dumpsys activity b
    - Static registration
      - % adb shell dumpsys package

Event Delivery

- Events delivered by calling onReceive() and passing Intent as a parameter

Event Handling in onReceive()

- onReceive() should be short-lived
  - Hosting process has high priority while onReceive() is executing & is often terminated when onReceive() returns
- If event handling is lengthy, consider starting a Service, rather than performing complete operation in onReceive()
- BroadcastReceivers can’t start asynchronous operations
  - e.g., showing a dialog, binding to a Service, starting an Activity via startActivityForResult

Handling a Normal Broadcast

```
public class Receiver1 extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    System.out.println(this + ":GOT THE INTENT");
    // emulator doesn't support vibration
    Vibrator v = (Vibrator) context.getSystemService(
                                      Context.VIBRATOR_SERVICE);
    v.vibrate(500);
  }
}
```

Handling an Ordered Broadcast

- Passing results

```
public class Receiver1 extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    String tmp = getResultData() != null ? getResultData() : "";
    setResultData(tmp + ":Receiver 1:");
  }
}
```

Handling an Ordered Broadcast

- Aborting a broadcast

```
public class Receiver2 extends BroadcastReceiver {
  public void onReceive(Context context, Intent intent) {
    if (isOrderedBroadcast()) {
      abortBroadcast();
    }
    System.out.println(this + ":GOT THE INTENT");
    // emulator doesn't support vibration
    Vibrator v = (Vibrator) context.getSystemService(
                                    Context.VIBRATOR_SERVICE);
    v.vibrate(500);
  }
}
```

Handling a Sticky Broadcast

```
public class StickyIntentBroadcastReceiverActivity extends Activity {
  public void onCreate(Bundle savedInstanceState) {
    registerReceiver(new BroadcastReceiver() {
      public void onReceive(Context context, Intent intent) {
        if (intent.getAction().equals(
                              Intent.ACTION_BATTERY_CHANGED)) {
          String age = "Reading taken recently";
          if (isInitialStickyBroadcast()) { age = "Reading may be stale”; }
            state.setText("Current Battery Level" +  String.valueOf(
              intent. getIntExtra(BatteryManager.EXTRA_LEVEL, -1))   + "\n" + age);
        }
      }
    }, new IntentFilter(Intent.ACTION_BATTERY_CHANGED));
  }
}
```

Source Code Examples

- BroadcastReceiverCompoundBroadcast
- BroadcastReceiverCompoundOrderedBroadcast
- BroadcastReceiverCompoundOrderedBroadcast
  WithResultReceiver
- BroadcastReceiverSingleBroadcast
  DynamicRegistration
- BroadcastReceiverSingleBroadcastStaticRegistration
- BroadcastReceiverStickyIntent