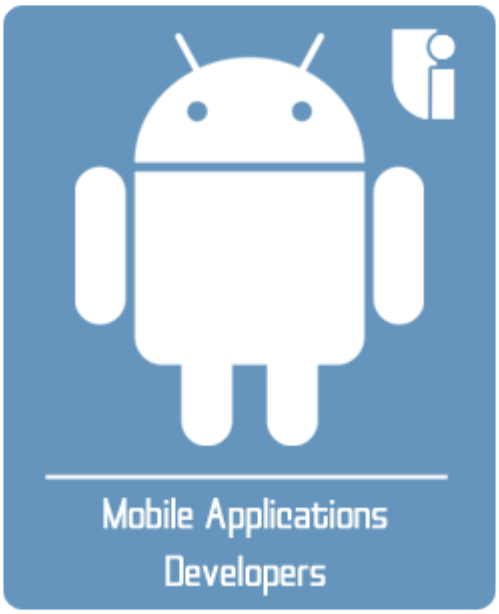
akademia androida



Service, BroadcastReceiver, ContentProvider

część IV

agenda

1. BroadcastReceiver
2. Service
3. ContentProvider
4. Zadanie 1.
5. Zadanie 2 (domowe).

1. BroadcastReceiver

`BroadcastReceiver` jest jednym z czterech komponentów Androida (obok komponentów `Activity`, `Service` oraz `ContentProvider`). Jest to komponent, który przechwytuje intencje rozgłoszone globalnie. W poprzedniej części intencje służyły do poruszania się między aktywnościami lub uruchomienia żądanej akcji, w tym wypadku informują one, że jakaś akcja (uruchomienie telefonu, SMS, połączenie przychodzące) miała miejsce w systemie. Jeżeli nasza aplikacja będzie posiadać komponent `BroadcastReceiver` wtedy możemy na tego typu wydarzenia odpowiednio zareagować.

Aby stworzyć `BroadcastReceiver` w pliku `AndroidManifest.xml` dodajemy element `<receiver>` wraz z filtrem intencji określającym na jakie intencje będzie on reagował. Nazwa to nazwa klasy, która dziedziczy po `BroadcastReceiver`. Przykładowo:

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
...
<receiver android:name=".MySmsReceiver" >
    <intent-filter>
       <action android:name="android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
</receiver>
```

Klasa która, będzie dziedziczyła po `BroadcastReceiver` musi implementować metodę abstrakcyjną `onReceive()`, w której przechwytujemy dostarczoną intencje.

```
public class MySmsReceiver extends BroadcastReceiver {
      @Override
      public void onReceive(Context ctx, Intent i) {
            if (i.getAction().equals("android.provider.Telephony.SMS_RECEIVED")) {
                  Toast.makeText(ctx, "Dostałem sms", Toast.LENGTH_SHORT).show();
            }
      }
}
```

Sam komponent `BroadcastReceiver` nie powinien wykonywać skomplikowanych operacji, jeżeli takie operacje są potrzebne powinien on uruchomić komponent `Service`, który służy do wykonywania tego typu zadań.

Standardowe akcje systemu Android w klasie `Intent`:

- `ACTION_TIME_TICK`
- `ACTION_TIME_CHANGED`
- `ACTION_TIMEZONE_CHANGED`
- `ACTION_BOOT_COMPLETED`
- `ACTION_PACKAGE_ADDED`
- `ACTION_PACKAGE_CHANGED`
- `ACTION_PACKAGE_REMOVED`
- `ACTION_PACKAGE_RESTARTED`
- `ACTION_PACKAGE_DATA_CLEARED`
- `ACTION_UID_REMOVED`
- `ACTION_BATTERY_CHANGED`
- `ACTION_POWER_CONNECTED`
- `ACTION_POWER_DISCONNECTED`
- `ACTION_SHUTDOWN`

http://developer.android.com/reference/android/content/Intent.html
Akapit : **Standard Broadcast Actions**

2. Service

Komponenty `Service` jak już wcześniej wspomniałem służą do wykonywania długotrwałych operacji w tle, same w sobie nie posiadają UI. Najczęściej są uruchamiane poprzez komponent `BroadcastReceiver`. Do uruchamiania służy metoda `Context.startService(Intent)`. Należy również zaimplementować interfejs `Runnable`, aby operacje wykonywały się w osobnym wątku. `Service` dodajemy do projektu poprzez dopisanie w pliku `AndroidManifest.xml` elementu `<service>` oraz utworzenie nowej klasy, przykłady poniżej.

AndroidManifest.xml:

```
<service android:name=".MyUselessService" />
```

Nasz `Service`:

```
public class MyUselessService extends Service implements Runnable {
	@Override
	public void onCreate() {
		super.onCreate();
		Thread aThread = new Thread(this);
		aThread.start();
	}
	public void run() {
		//jakies operacje
		stopSelf();
	}
	@Override
	public IBinder onBind(Intent arg0) {
		return null;
	}
}
```

`Service` może pokazać powiadomienie, które pojawi się w pasku powiadomień systemu Android. Powiadomienie tworzy się za pomocą klasy `NotificationCompat.Builder` dla API < 11 (wymaga Android Support Library http://developer.android.com/training/basics/fragments/support-lib.html) oraz `Notification.Builder` dla wyższych wersji API. Przykład:

```
int NOTIFICATION_ID = 123456;
NotificationManager nm =
                    (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

NotificationCompat.Builder builder = new NotificationCompat.Builder(this);
builder
        // aktywność do uruchomienia po kliknięciu w powiadomienie
       .setContentIntent(PendingIntent)
       .setSmallIcon(int) //np R.drawable.icon
       .setLargeIcon(BitmapFactory.decodeResource(this.getResources(),int))
       .setTicker(String)
       .setWhen(System.currentTimeMillis()).setAutoCancel(true)
       .setContentTitle(String)
       .setContentText(String);
Notification n = builder.build();
nm.notify(NOTIFICATION_ID, n);
```

3. ContentProvider

ContentProvider jest to komponent systemu Android, który pozwala na udostępnianie danych z naszej aplikacji do innych aplikacji. ContentProvider może również dostarczać dane do tej samej aplikacji w której został zaimplementowany. Może on dawać dostęp do dowolnego mechanizmu przechowywania danych, którego używa się w systemie Android (baza SQLite, XML etc.). Dane dostępne są poprzez omówione na poprzednich zajęciach URI, które są zdefiniowane w naszej klasie dziedziczącej po ContentProvider jako stałe.

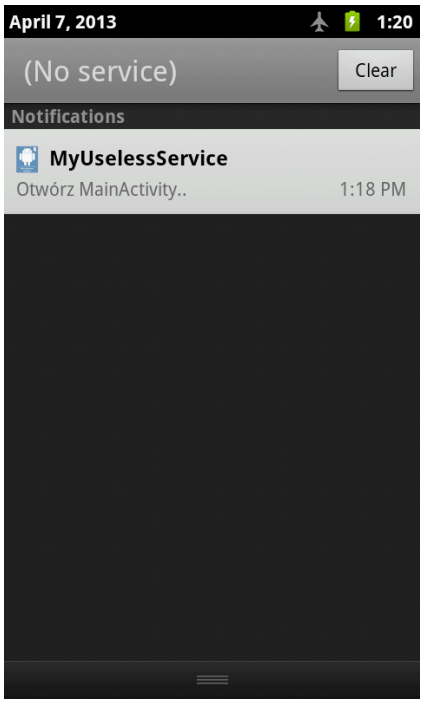
Przykłady:

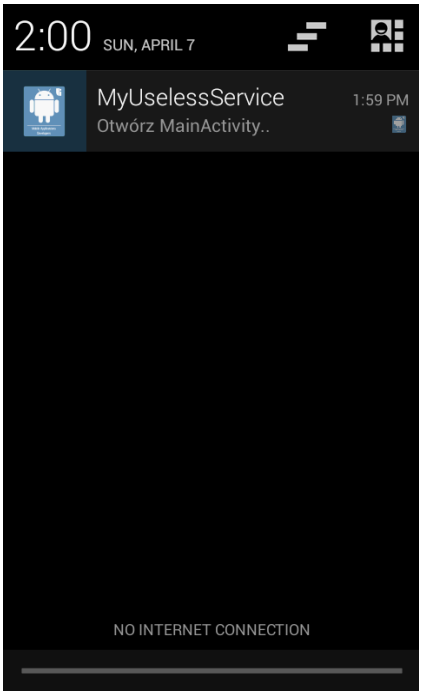
`content://furniture/parts/`	Zwraca listę części z providera udostępniającego `content://furniture`
`content://furniture/chairs/`	Zwraca listę krzeseł z providera udostępniającego `content://furniture`
`content://furniture/chairs/1`	Zwraca konkretne krzesło o ID 1 z providera udostępniającego `content://furniture`

Więcej na temat tego komponentu dowiecie się podczas zajęć na temat składowania danych w Androidzie.

4. Zadanie 1.

- Stwórz własny `BroadcastReceiver` oraz `Service`.
- `BroadcastReceiver` powinien podczas przejścia w tryb samolotowy uruchomić stworzony `Service`.
- `Service` powinien pokazać powiadomienie, które po kliknięciu na nie przeniesie do nas do głównej aktywności.
- Ponadto `Service` po 2 sekundach działania za pomocą metody `Log.d(String,String)` wypisuję w logach dowolny tekst.





Tip:

```
Akcja : Intent.ACTION_AIRPLANE_MODE_CHANGED
<action android:name="android.intent.action.AIRPLANE_MODE" />
```

5. Zadanie 2 (domowe).

Przerobić aplikację z zadania pierwszego w taki sposób aby `Service` uruchamiał się co określony czas. Można wykorzystać do tego klasę `AlarmManager`.

Dziękuję za uwagę !

Kontakt:

- sswierczek@wi.zut.edu.pl
- mad@zut.edu.pl