

Android alkalmazásfejlesztés

Broadcast

Service

Widget

OE-NIK

2013. május 6.

Sicz-Mesziár János

sicz-mesziar.janos@
nik.uni-obuda.hu



Broadcast

- ⊙ Mi is ez? – Aszinkron, üzenetszórásos értesítés
- ⊙ Azok akik felregisztráltak rá értesülnek róla 😊
- ⊙ Két fő csoport:
 - Normal broadcast
 - teljesen aszinkron,
 - „véletlen” sorrend
 - Hatékonyabb, de korlátok: nincs visszatérési érték, visszavonás
 - Ordered broadcast
 - egyszerre csak egy „receiver” fut
 - visszavonható – abort
 - Prioritásokat adhatunk (android:priority)
- ⊙ onReceive() csak egyszer fut le - rövid életű
 - Hosszan futó műveletekre alkalmatlan (timeout ~10sec)
 - Popup dialog meghívására sem jó (objektum már nem él)

BroadcastReceiver-re példák

- ⦿ Akkumulátor szint változik
 - `android.intent.action.BATTERY_CHANGED`
- ⦿ Automatikus indítás? – Rendszer betöltődött
 - `android.intent.action.BOOT_COMPLETED`
- ⦿ SMS érkezett
 - `android.provider.Telephony.SMS_RECEIVED`
- ⦿ Bluetooth felderítés
 - `android.bluetooth.adapter.action.DISCOVERY_STARTED`
 - `android.bluetooth.adapter.action.DISCOVERY_FINISHED`
 - ...
- ⦿ Bejövő hívás, kimenő hívás
 - `android.intent.action.PHONE_STATE`
 - `android.intent.action.NEW_OUTGOING_CALL`
- ⦿ Hálózat változik
- ⦿ Képernyőt lekapcsolták

Broadcast megvalósítás

◎ Regisztrálás statikusan

- BroadcastReceiver osztály implementálása
- AndroidManifest.xml-ben <receiver> megadása

◎ Regisztrálás dinamikusán, futási időben

- BroadcastReceiver osztály implementálása
- Context.registerReceiver()-el regisztrálunk → onResume()
- Context.unregisterReceiver() leiratkozunk → onPause()

```
private BroadcastReceiver mBatInfoReceiver = new  
BroadcastReceiver() {  
    public void onReceive(Context arg0, Intent intent) {  
        int level = intent.getIntExtra("level", 0);  
        contentTxt.setText("" + level + "%");  
    }  
};
```

◎ Broadcast kibocsátás

- Context.sendBroadcast(Intent intent)

Service

⦿ Egy alkalmazás komponens:

- hosszan futó műveletek végrehajtása
- háttérben fut, nincs UI

**Nem külön szálon fut!
Nem egy külön folyamat!**



⦿ Két formája van:

■ Started

- egy komponens (pl.: Activity) elindítja a `startService()` eljárással
- Végtelenségig fut, akkor is ha az őt elindító komponens megsemmisül. (Nincs visszatérési érték) → **stopService()**
- Általában egy műveletet hajt végre, majd megsemmisíti magát.
Pl.: letöltés / feltöltés az internet irányába

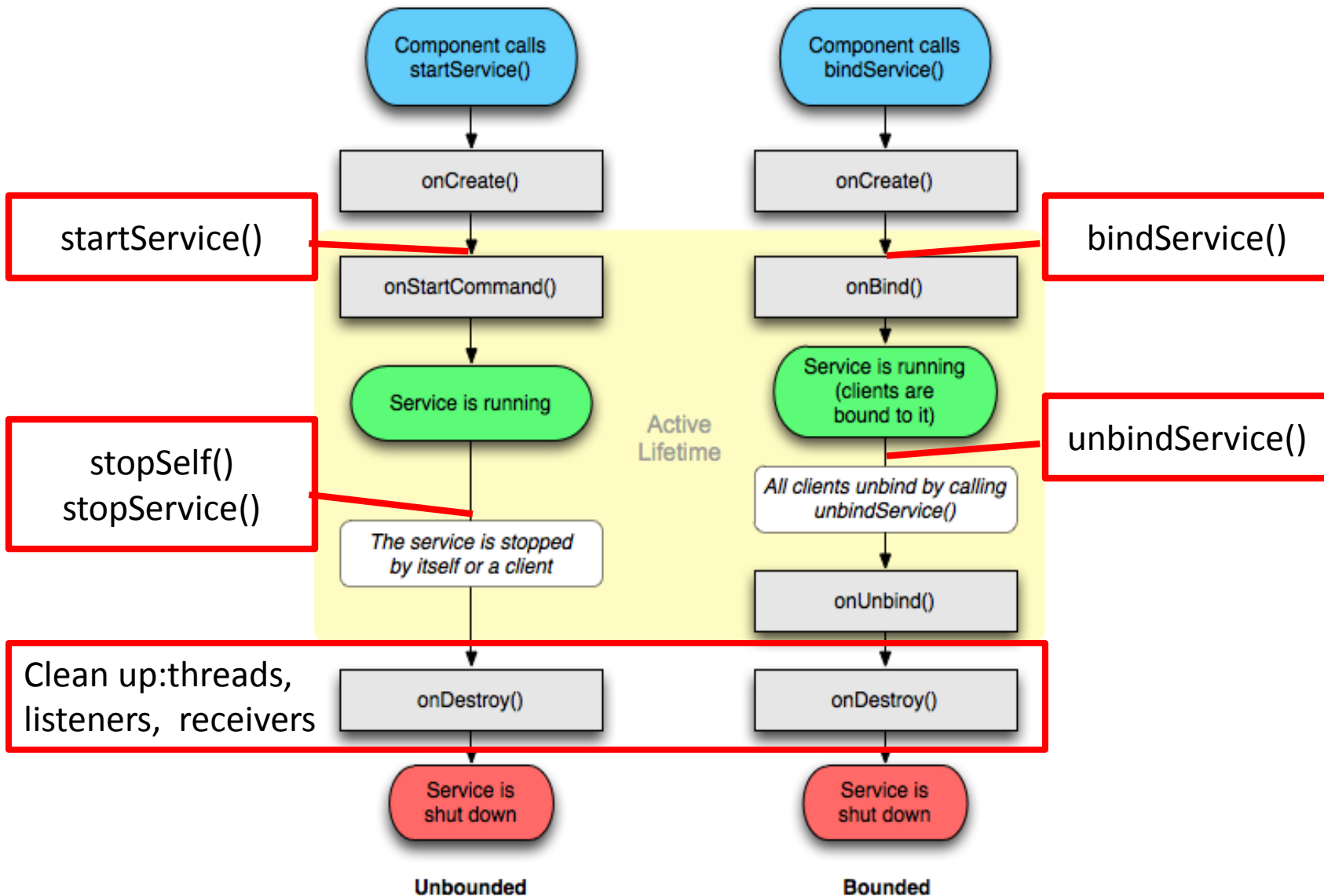
■ Bound

- Egy komponens `bindService()`-al kötődik
- Kliens-szerver felület a komponens és service között (request, results, ...)
- Addig fut amíg az összeköttetés él. (Pl.: zenelejátszás)

**Együtt is
alkalmazható**

Service (2)

Bármelyik komponens használhatja a service-t!
De priváttá tehető!



Service - gyakorlatban

- ◎ Regisztrálás az AndroidManifest.xml-ben!
- ◎ Service leszármaztatása

```
public class MyService extends Service{
    public void onCreate(){};

    public int onStartCommand(Intent intent, int
    flag, int id){};

    public void onDestroy(){};

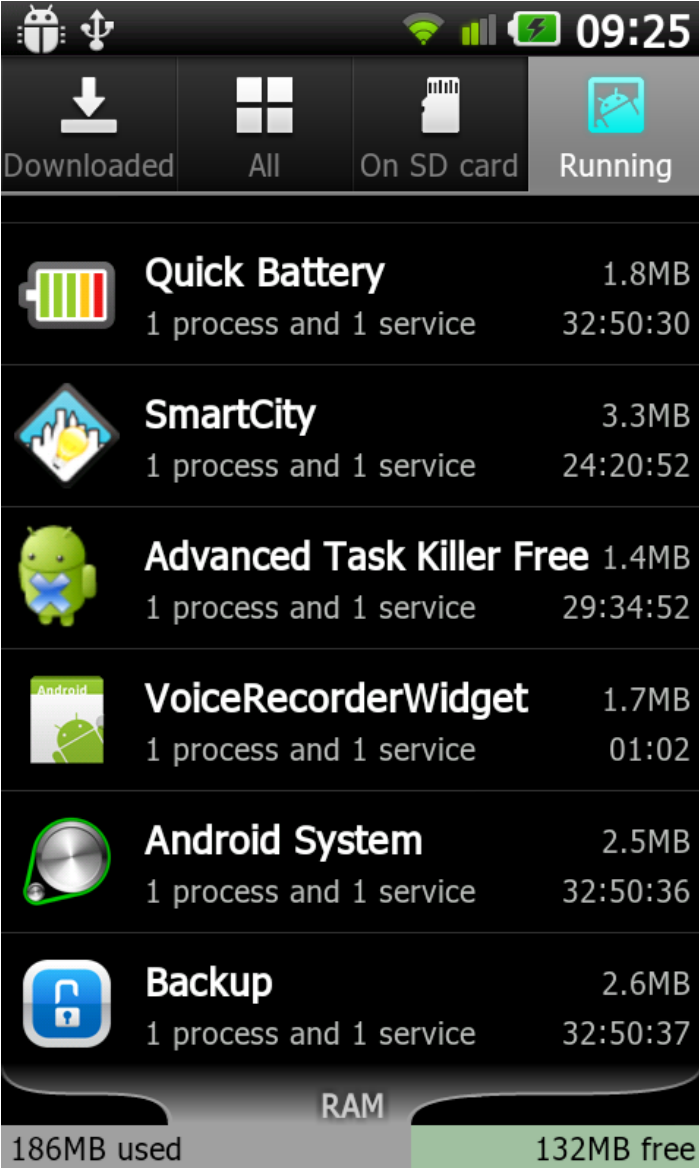
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Service – gyakorlatban (2)

Honnan tudjuk, hogy működik?

Futó service-ok:

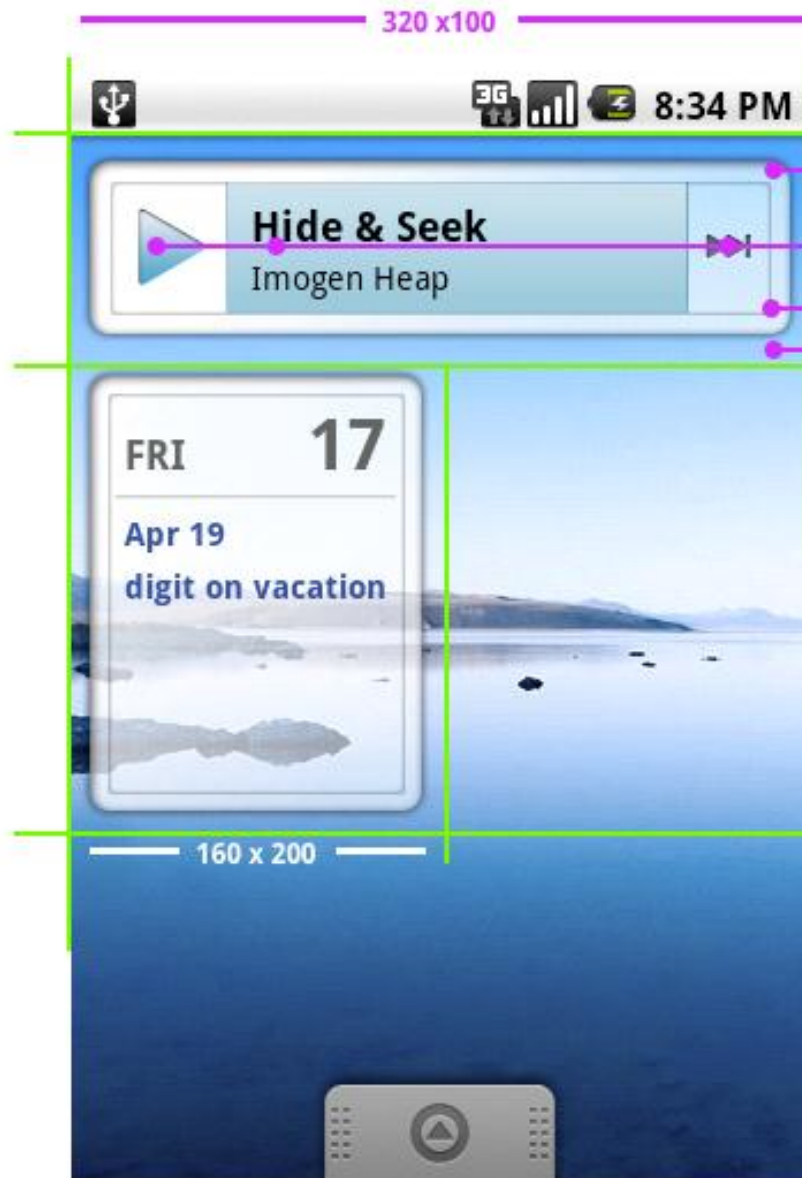
Settings > Application >
Running services



Service Name	Memory Usage	Processes/Services	Time
Quick Battery	1.8MB	1 process and 1 service	32:50:30
SmartCity	3.3MB	1 process and 1 service	24:20:52
Advanced Task Killer Free	1.4MB	1 process and 1 service	29:34:52
VoiceRecorderWidget	1.7MB	1 process and 1 service	01:02
Android System	2.5MB	1 process and 1 service	32:50:36
Backup	2.6MB	1 process and 1 service	32:50:37

RAM
186MB used | 132MB free

Widget



Frame with shadow

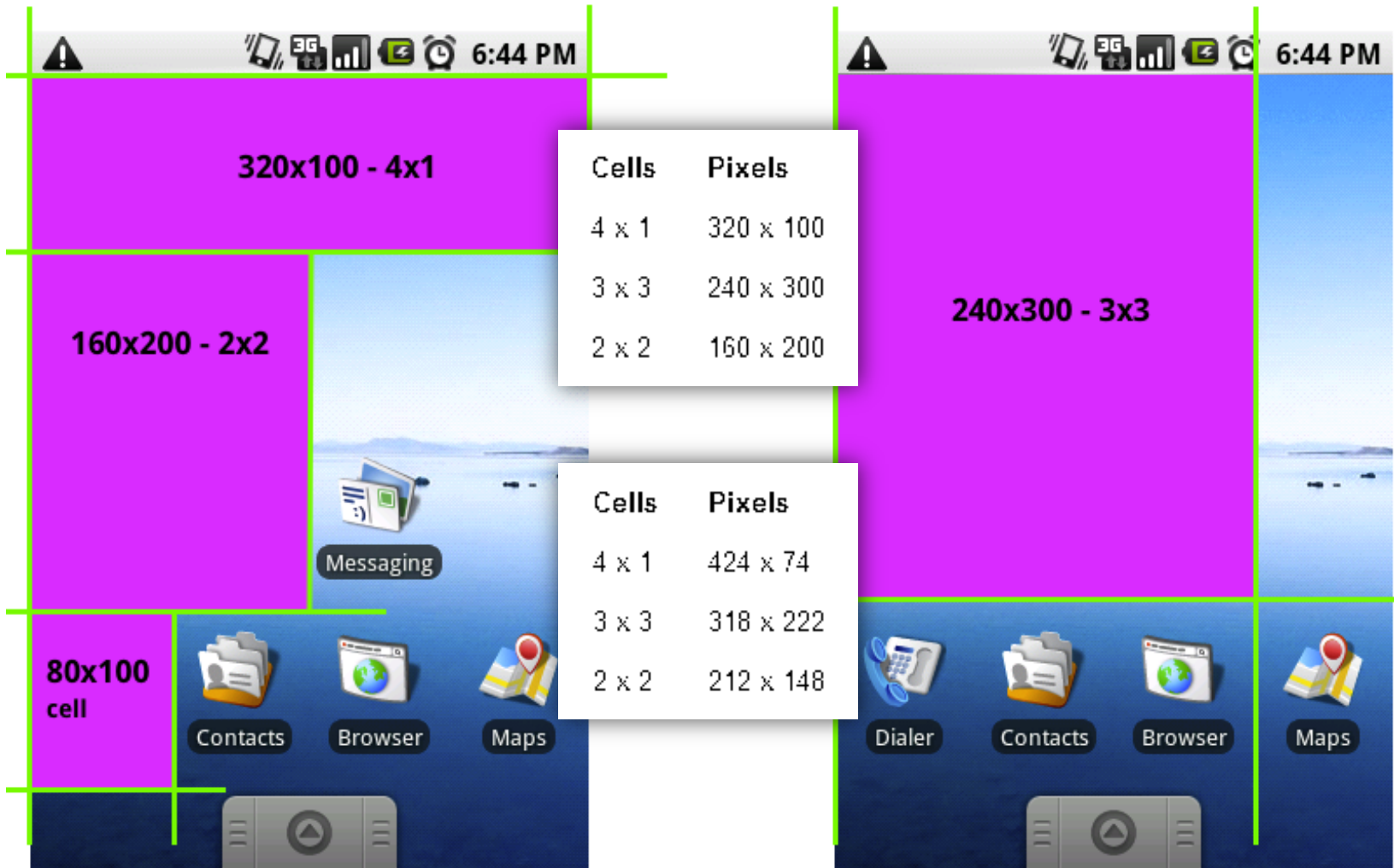
Graphic elements including text with padding for contents

Inner edge

Bounding box

- ◎ Mini alkalmazás
- ◎ Beágyazható másik alkalmazásba (Home)
- ◎ Rendszeres időközönként frissül

Widget – szabvány méretei



Forrás: http://developer.android.com/guide/practices/ui_guidelines/widget_design.html#sizes

Widget - gyakorlatban


◎ App Widget definiálás AndroidManifest-ben

```
<receiver android:name="ExampleAppWidgetProvider" >  
  <intent-filter>  
    <action android:name="android.appwidget.action.APPWIDGET_UPDATE" />  
  </intent-filter>  
  <meta-data android:name="android.appwidget.provider,,  
    android:resource="@xml/example_appwidget_info" />  
</receiver>
```



◎ AppWidgetProviderInfo Metadata

```
<appwidget-provider xmlns:android="http://schemas.android.com/apk/res/android"  
  android:minWidth="294dp"  
  android:minHeight="72dp"  
  android:updatePeriodMillis="86400000,,  
  android:initialLayout="@layout/example_appwidget,,>  
</appwidget-provider>
```



Új > Android XML > AppWidget Provider

Widget – gyakorlatban (2)

◎ App Widget Layout (res/layout/)

- Tipikus layout építés (xml, grafikus szerkesztő, ...)
- DE, RemoteViews-ra alapszik, → nem támogat minden layout-ot vagy view-t!

◎ Támogatott layout-ok:

- FrameLayout
- LinearLayout
- RelativeLayout

◎ Támogatott widget-ek:

- Button, ImageButton
- TextView, ImageView
- ProgressBar
- Chronometer, ViewFlipper, AnalogClock

Widget – gyakorlatban (3)

- ◎ `AppWidgetProvider` osztály implementálása
- ◎ `BroadcastReceiver` leszármazott
- ◎ Értesítés kapunk, ha :

- `onUpdate()` időszakos frissítés (*updatePeriodMillis*)
- `onDeleted()` widgetet eltávolították
- `onEnabled()` első widget létrehozáskor
- `onDisabled()` utolsó widget eltávolításakor
- ~~`onReceive()`~~ minden broadcast beérkezésekor

