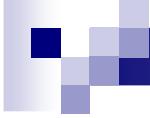




NTI / PAA PROGRAMOVÁNÍ MOBILNÍCH APLIKACÍ

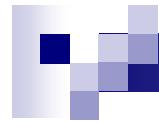
9. Services, Notifications

Ing. Igor Kopetschke – TUL, NTI
<http://www.nti.tul.cz>



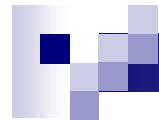
Android – Service

- Další z mechanismů práce na pozadí v Androidu
- Doposud jsme si představili Handler, vlákna a AsyncTask
- Vlákna a AsyncTask jsou primárně určeny
 - Jednorázové specializované činnosti, které jsou krátké v čase
 - Předpokládá se interakce s UI
- Service mají jiný účel
 - V principu se podobají aktivitám, ale bez UI
 - Určeny hlavně pro dlouhou kontinuální činnost
 - Přehrávání hudby, dlouhé stahování dat aj.
 - Nevadí jim přesunutí do pozadí a práce s jinou aplikací
 - Mají vlastní životní cyklus nezávislý na aktivitě



Android – Service

- Méně náchylné k násilnému ukončení systémem než activity na pozadí
- Pokud jsou přeci jen ukončeny, systém je po získání potřebných prostředků restartuje
- Jsou spouštěny, zastavovány a kontrolovány z jiných komponent aplikace (např. jiné Service, Activity nebo BroadcastReceiveru)
- Je nutné je registrovat v AndroidManifestu

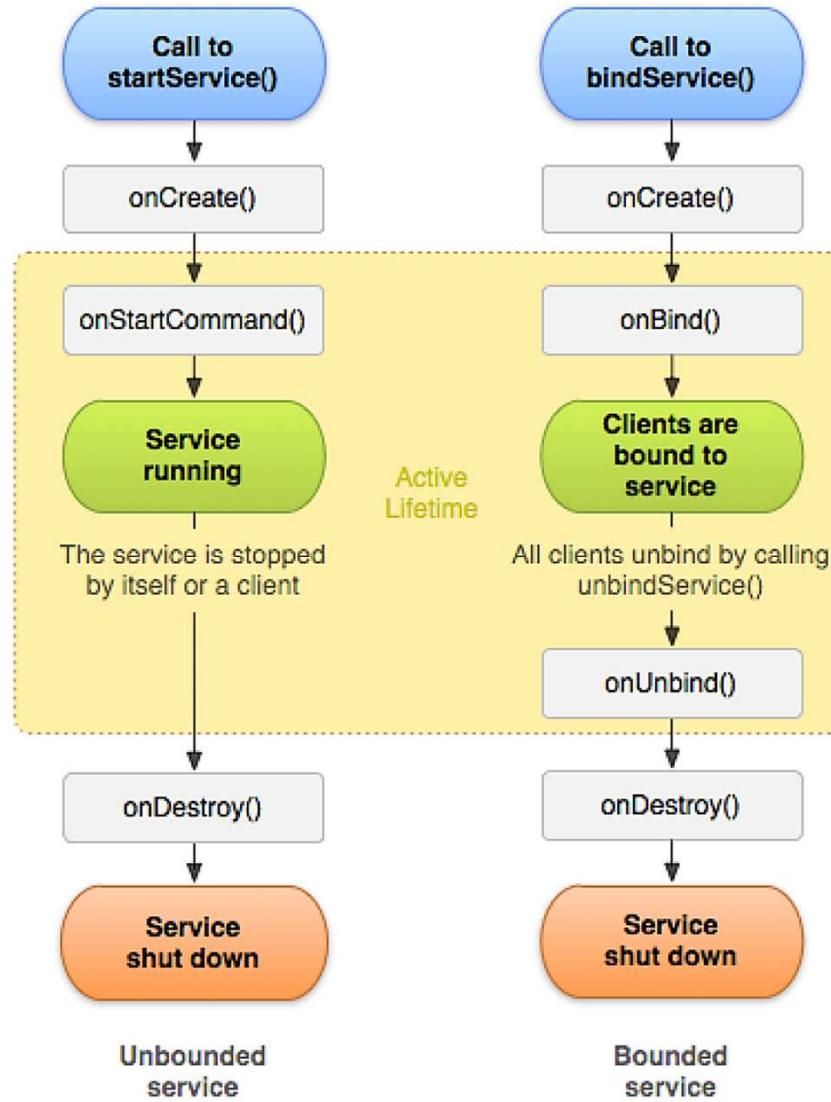


Android – Service

Životní cyklus Service

- `onCreate()` – zavoláno, když je Service vytvořena jakýkoliv způsobem
- `onStartCommand()` – zavoláno, když je vyvolána metoda `startService()`
- `onBind()` – zavoláno, pokud klient zavolá `bindService()`
- `onDestroy()` – zavoláno, když má být Service odstraněna
 - Není zaručeno, že tato metoda bude zavolána

Android – Service



Android – Service

MyService.java

```
1. public class MyService extends Service{  
2.     @Override  
3.     public int onStartCommand(Intent intent, int flags, int startId) {  
4.         //provedení potřebného úkonu  
5.         return START_STICKY;  
6.     }  
7.     @Override  
8.     public IBinder onBind(Intent intent) {  
9.         return null;  
10.    }  
11. }
```

MainActivity.java

```
1. startService(new Intent(this, MyService.class));  
2. stopService(new Intent(this, MyService.class));
```

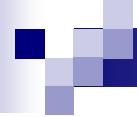
AndroidManifest

```
1. <service android:enabled="true" android:name=".MyService" />
```

Android – Service

Service – onStartCommand()

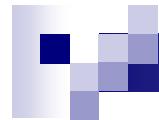
- `onStartCommand()` vrací konstanty, které určují, jak se bude služba chovat, když ji systém ukončí, kvůli nedostatku prostředků
 - Zavedeno od Android 2.0
 - Dříve používaná metoda `onStart()` je deprecated
- *START_STICKY*
 - Standardní chování
 - Služba se vždy restartuje, ale Intent v `onStartCommand()` bude null
 - Typické pro služby, které jsou explicitně spouštěny a zastavovány pomocí `startService()` a `stopService()`



Android – Service

Service – onStartCommand()

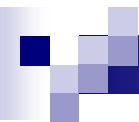
- *START_NOT_STICKY*
 - Služba se restartuje, pouze pokud čekají na vyřízení další požadavky. Jinak zůstane služba zastavena
 - Typické pro služby, které se spouští periodicky a po skončení požadovaného úkonu se samy ukončí pomocí `stopSelf()`
- *START_REDELIVER_INTENT*
 - Podobné *START_NOT_STICKY*
 - Služba se restartuje, pouze pokud čekají na vyřízení další požadavky nebo pokud před ukončením nedokončila služba svoji práci. Byla ukončena před zavoláním `stopSelf()`
 - Po restartu je předán do `onStartCommand()` původní Intent
 - Typické pro služby, kdy je potřeba zajistit vykonání operace



Android – Service

Service – běh

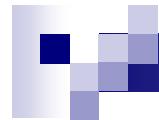
- Na rozdíl od Activity má daná služba vždy maximálně jednu instanci
- Po opětovném zavolání `startService()` se pouze znova zavolá `onStartCommand()` v již běžící službě
- Service zůstává spuštěna dokud není explicitně zastavena z jiné komponenty pomocí `stopService()`, nezavolá `stopSelf()` nebo není ukončena systémem



Android – Service

Service – flags

- Pro zjištění, jak se Service spustila, slouží atribut flags, který může nabývat těchto hodnot:
 - *START_FLAG_REDELIVERY* – indikuje, že Intent je znovu doručením předchozího Intentu (*START_REDELIVER_INTENT*), ale Service byla ukončena před zavoláním `stopSelf()`
 - *START_FLAG_RETRY* – indikuje, že Service byla restartována systémem (kvůli nedostatku prostředků) a měla nastaveno *START_STICKY*



Android – Service

Provázání z Activity do Service

- `startService()`
 - Jako parametr je Intent s potřebnými Extras
 - Volání je asynchronní
 - Je zavolána metoda `onCreate()`
 - Intent je předán do metody `onStartCommand()`
 - Služba běží dokud není (explicitně) zastavena

Android – Service

Provázání z Activity do Service

- bindService ()
 - Způsob, jak vystavit určité API služby ostatním komponentám (klientům)
 - Volání je asynchronní
 - Služba vrací tzv. `IBinder`, který reprezentuje určitý objekt ze služby
 - Současně je potřeba vytvořit `ServiceConnection` v daném klientovi
 - Pokud je jako parametr předáno `BIND_AUTO_CREATE`, je vytvořena nová služba, jinak se vrací `false`
 - Neprovede se však operace `onStartCommand()`

Android – Service

Provázání z Activity do Service

- V dané službě vytvoříme třídu která dědí od Binder a vrací odkaz na danou instanci služby
- Odkaz na instanci této třídy vrátíme v onBind

```
1. private final IBinder binder = new MyBinder();
2. public class MyBinder extends Binder {
3.     MyService getService() {
4.         return MyService.this;
5.     }
6. }
7. @Override
8. public IBinder onBind(Intent intent) {
9.     return binder;
10. }
```

Android – Service

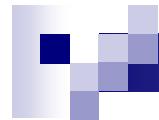
Provázání z Activity do Service

- V aktivitě vytvoříme třídu dědící od ServiceConnection

```
1. // odkaz na sluzbu
2. private MyService service;
3. // ma na starosti spojeni mezi sluzbou a activitou
4. private ServiceConnection mConnection = new ServiceConnection() {
5.     // spusti se jakmile se povede navazat spojeni
6.     public void onServiceConnected(ComponentName className, IBinder service) {
7.         //ziskame instanci nasi sluzby
8.         service = ((MyService.MyBinder) service).getService();
9.     }
10.    // spusti se jakmile se spojeni ukonci
11.    public void onServiceDisconnected(ComponentName className) {
12.        service = null;
13.    }
14.};
```

- Její instanci poté předáme do bindService()

```
1. Intent i = new Intent(this, MyService.class);
2. bindService(i, mConnection, BIND_AUTO_CREATE);
```



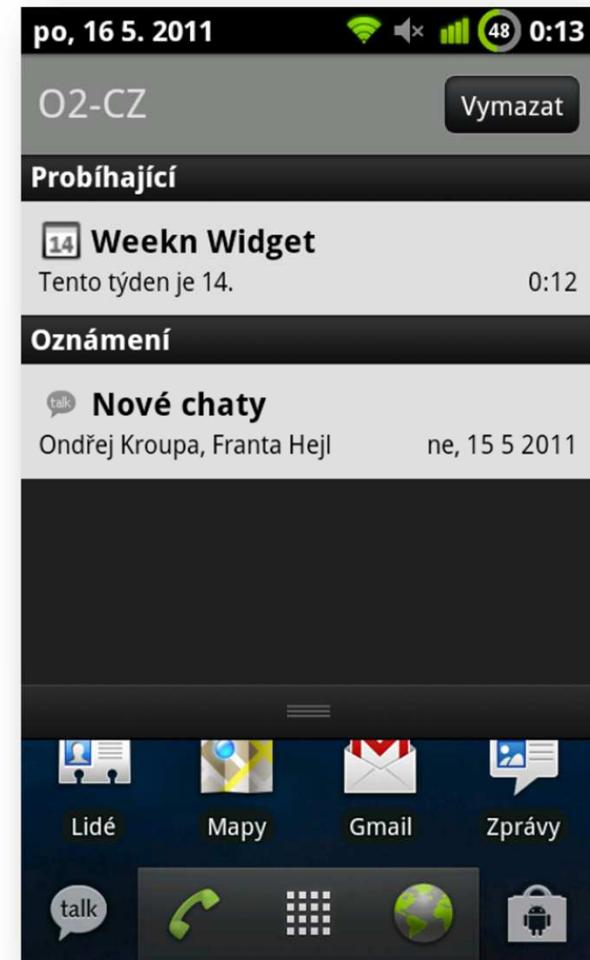
Android – Service

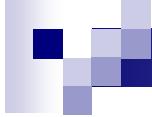
Provázání ze Service do Activity

- Callback object nebo Listener
 - Klient poskytne objekt, přes který bude služba komunikovat
 - Pozor na memory leaky a ošetření životního cyklu
- Broadcast Intents
 - Využití Broadcast Receiveru pro zachycení Intentu
- Pending Results
 - Zapouzdření Intentu a vykonání příslušné akce
 - Viz. dále

Android – Notification

- Slouží k upozornění uživatele na nějakou událost bez použití Activity
- Indikují také probíhající služby (služby, které mají zvýšenou prioritu)
- Mohou mít tyto funkce:
 - Zobrazení ikonky ve Statusbaru
 - Zobrazení podrobnějších informací a spouštění Intentu po rozbalení Statusbaru
 - Blikání notifikačních LED diod, zvukové upozornění a další HW záležitosti





Android – Notification Manager

- Systémová služba starající se o notifikace
- Umožňuje vytvářet nové notifikace, upravovat staré nebo odstranit již nepotřebné

```
1. String nsName = Context.NOTIFICATION_SERVICE;  
2. NotificationManager notificationManager;  
3. notificationManager = (NotificationManager) getSystemService(nsName);
```

Android – Notification

Notification – vytvoření

- Vytvoření notifikace a nastavení základních parametrů

```
1. // Vybereme ikonku, ktera se bude zobrazovat u notifikace
2. int icon = R.drawable.icon;
3. // Text ktery se objevi pri aktivace notifikace
4. String tickerText = "Notifikace";
5. // Cas, podle ktereho se urcuje poradi notifikaci
6. long when = System.currentTimeMillis();
7. Notification notification = new Notification(icon, tickerText, when);
8. //Nastavime pocet, ktery bude zobrazen pres ikonku notifikace
9. notification.number=10;
```

Android – Notification

Notification – rozšiřující informace

- Nastavení defaultního View po rozbalení Statusbaru a Intentu, který se spustí po kliku na notifikaci

```
1. Context context = this;  
2. // Text ktery se zobrazi po roztahnuti statusbaru  
3. String expandedText = "Podrobny text notifikace";  
4. // Titulek notifikace po roztahnuti statusbaru  
5. String expandedTitle = "Titulek notifikace";  
6. // Intent ktery spusti activitu po kliknuti na notifikaci  
7. Intent intent = new Intent(this, MainActivity.class);  
8. PendingIntent pi = PendingIntent.getActivity(context, 0, intent, 0);  
9. //Nastavime defaultni View  
10.notification.setLatestEventInfo(context, expandedTitle, expandedText, pi);
```

- Zobrazení notifikace

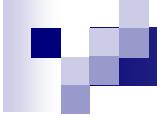
```
1. int id=0; //jedinecne id notifikace v ramci aplikace  
2. notificationManager.notify(id, notification);
```

Android – Notification

PendingIntent

- Zapouzdří Intent, který pak může být použit systémem nebo jinými aplikacemi
- Existují 3 typy podle toho, co se následně provede:
 - Spustí aktivitu (jako zavolání `Context.startActivity(Intent)`)
 - Spustí službu (jako zavolání `Context.startService(Intent)`)
 - Vyšle broadcast (jako zavolání `Context.sendBroadcast()`)
- Obdrží se zavoláním patřičné metody

```
1. pi = PendingIntent.getActivity(context, requestCode, intent, flags);  
2. pi = PendingIntent.getService(context, requestCode, intent, flags);  
3. pi = PendingIntent.getBroadcast(context, requestCode, intent, flags);
```



Android – Notification

Notification – další parametry

- `notification.contentView` – nastavení vlastního layoutu notifikace po rozbalení statusbaru
- `notification.defaults` – možnosti upozornění (`DEFAULT_LIGHTS` | `DEFAULT_SOUND` | `DEFAULT_VIBRATE`)
- `notification.flags` – upřesnění notifikace (viz dokumentace)
- `notification.sound` – Uri souboru pro zvukové upozornění
- `notification.vibrate` – Pole s hodnotami, v jaké frekvenci bude telefon vibrovat
- `notification.ledARGB` – Barva LED diody

Android – Notification

Notification – startForeground()

- Ve výchozím nastavení služba běží na pozadí
- Pokud potřebujeme zvýšit prioritu služby je potřeba službu dostat na popředí
- Je **téměř** zaručeno, že systém tuto službu předčasně neukončí
- Není použit `NotificationManager`, ale speciální metoda, která automaticky po skončení služby odstraní notifikaci

```
1. startForeground (id, notification);
```

Android – Notification

Priority procesu

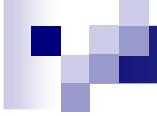
- Pět různých úrovní (priorit) pro proces. Určuje „důležitost“ procesu pro uživatele a ukončení při nedostatku paměti
 - Níže daný výpis je částečně zjednodušen
- Foreground process – např.
 - Activita na popředí (její `onResume()` metoda byla zavolána)
 - Service, která zavolala `startForeground()`
 - Service, která je využívána (bounded) aktivitou na popředí
 - Broadcast Receiver zpracovávající metodu `onReceive()`
- Visible process – stále viditelný uživateli např.
 - Activita, jejíž `onPause()` metoda byla zavolána
 - Service, která je využívána (bounded) visible aktivitou



Android – Notification

Priority procesu

- Service process
 - Service, která byla nastartována pomocí `startService()` a nespadá do výše uvedených kategorií
- Background process
 - Activita, jejíž `onStop()` metoda byla zavolána
 - Proces neovlivňuje to, co uživatel dělá nebo vidí
 - Pro ukončení se používá LRU algoritmus
 - Důležité správně implementovat životní cyklus
- Empty process
 - Neobsahuje žádné aktivní komponenty
 - Používán pro cachování a zrychlení startup time



Použité a doporučené zdroje

- <http://developer.android.com/>
- <http://www.zdrojak.cz/serialy/vyvijime-pro-android/>
- <http://www.itnetwork.cz/java/android>
- <https://users.fit.cvut.cz/cermaond/dokuwiki>
- Google...



.. A to je pro dnešek vše

DĚKUJI ZA POZORNOST