10. SERVICE PATTERNS A APP WIDGETS

BI-AND

10. Přednáška

- Service Patterns
- Timer
- AlarmManager
- App Widgets

Service – Downloader a Messenger

- Downloader – např.
  - Stahování aktualizací z Google Play
  - Stahování více souborů z cloudbased úložišť
- Pouze pokud není možné použít DownloadManager z Android 2.3 Gingerbread
- Messenger
  - Oznámení pro activitu, že bylo stahování dokončeno
  - Zasílání zpráv do Handleru activity
  - Messenger je Parcelable
  - Viz. 9. přednáška

Service – Downloader a Messenger

- Intent Service – rozšiřuje standardní Service
  - Startuje se pomocí `startService()`
  - Přijímá požadavky v podobě Intentů
  - Více požadavků je řazeno do fronty
  - Vždy se zpracovává pouze jediný požadavek
  - Vytváří si vlákno na zpracování požadavků
  - Automaticky se ukončí po zpracování všech požadavků

Timer

- Umožňuje spouštět úlohu `TimerTask` v daný čas/časový interval
- Úloha se spustí v nově vytvořeném vlákně, které přísluší danému Timeru

```
1. timerTask = new TimerTask() {
2.       @Override
3.       public void run() {
4.                ...
5.       }
6. };
7. timer = new Timer();
8. //spousteni dane ulohy kazdou 1s
9. timer.schedule(timerTask, 0, 1000);
10.
11....
12.
13.//zruseni timeru
14.timer.cancel();
```

Handler

- Pro zpožděné vykonání úkolu lze využít i metody Handleru

- Pro zprávy
  - `sendMessageAtTime()`
  - `sendMessageDelayed()`

- Pro Runnable
  - `postAtTime()`
  - `postDelayed()`

Vhodnější pro některé úkony než AlarmManager

Alarms

- Umožňují odesílat Intents v předem stanovený čas či intervalech
- Jsou odesílány mimo rámec aplikace
  - Na rozdíl od Service umožňují spouštět události aplikace, ačkoliv již byla před tím ukončena
- Ideální pro provádění jednorázových a pravidelných úkonů jako např.
  - Pravidelné stahování aktualizací na pozadí
  - Spuštění upozornění v daný čas
  - Spuštění časově náročných úkonů v době, kdy se nepředpokládá používání telefonu (v noci)

AlarmManager – vlastnosti

- Alarmy jsou řízeny AlarmManager
  - Systémová služba, k níž získáme přístup pomocí `getSystemService`

```
1. AlarmManager am =
2.   (AlarmManager) context.getSystemService(Context.ALARM_SERVICE);
```

- Alarmy jsou zadávány v podobě PendingIntents
- Mohou být prováděny v různých intervalech (viz.dále)
- Všechny Alarmy jsou zrušeny při vypnutí telefonu
  - Alarm lze opět nastavit pomocí BroadcastReceiveru zaregistrovanému na `Intent.ACTION_BOOT_COMPLETED`
  - Nutná permission

AlarmManager – druhy alarmů

- Jednorázový – spustí se jednou v daný čas

```
am.set(type, triggerAtTime, operation);
```

- Opakovaný – spouští se opakovaně přesně v daných časech (např. každý den v 13:00)

```
am.setRepeating(type, triggerAtTime, interval, operation);
```

- Nepřesný opakovaný – spouští se opakovaně, ale není zaručený přesný čas
  - Méně náročný na baterii, spouští se více alarmů současně

```
am.setInexactRepeating(type, triggerAtTime, interval, operation);
```

AlarmManager – vytvoření alarmu

- U každého Alarmu musíme definovat tyto tři hodnoty
- Čas - určuje, kdy se spustí alarm
  - U opakovaných Alarmů je to čas prvního spuštění
  - Pokud je čas v minulost, spustí se Alarm hned po nastavení
- Operace – PendingIntent, který se provede
- Typ - určuje, jestli zadaný čas určuje specifický nebo uběhlý čas a případně probuzení zařízení

AlarmManager – vytvoření alarmu

- `AlarmManager.`*`RTC`*
  - Čas znamená počet ms od 1.1.1970
    - `System.`*`currentTimeMillis();`*
  - Neprobudí zařízení – daný PendingIntent se provede až po probuzení

- `AlarmManager.`*`RTC_WAKEUP`*
  - Probudí zařízení

- `AlarmManager.`*`ELAPSED_REALTIME`*
  - Čas znamená počet ms od posledního startu zařízení včetně doby spánku
    - `SystemClock.elapsedRealtime();`

- `AlarmManager.`*`ELAPSED_REALTIME_WAKEUP`*

AlarmManager – nastavení alarmu

- Opakovaným nastavením stejného alarmu se zruší předchozí a nastaví nový
  - if (oldPendingIntent.equals(newPendingIntent) == true)
    - Vrací true, pokud oba představují stejnou operaci ze stejného balíčku

```
1.  AlarmManager am =
2.      (AlarmManager) context.getSystemService(ALARM_SERVICE);
3.  long time = System.currentTimeMillis() + 1000*3600*2;
4.  long interval = 1000*3600*2;
5.  Intent intent = new Intent(context, MyReceiver.class);
6.  PendingIntent pendingIntent =
7.      PendingIntent.getBroadcast(context, 0, intent, 0);
8.  int type = AlarmManager.RTC_WAKEUP;
9.  //nastaveni opakovaneho alarmu
10. am.setRepeating(type, time, interval, pendingIntent);
11. ...
12. //zruseni alarmu
13. am.cancel(pendingIntent);
```

Problém spícího zařízení

- Spící zařízení má vypnutý displej, hlavní CPU, klávesnici atd.
- Běží pouze nejnutnější systémové služby
  - Zařízení probudí např. příchozí hovor nebo AlarmManager
- AlarmManager garantuje pouze to, že zařízení bude probuzené po dobu provádění akce onReceive()
- PendingIntent umožňuje spustit (resp. vyslat broadcast)
  - Activitu
  - Service
  - Brodcast Receiver

Service nebo Activita se nemusí vůbec stihnout spustit!

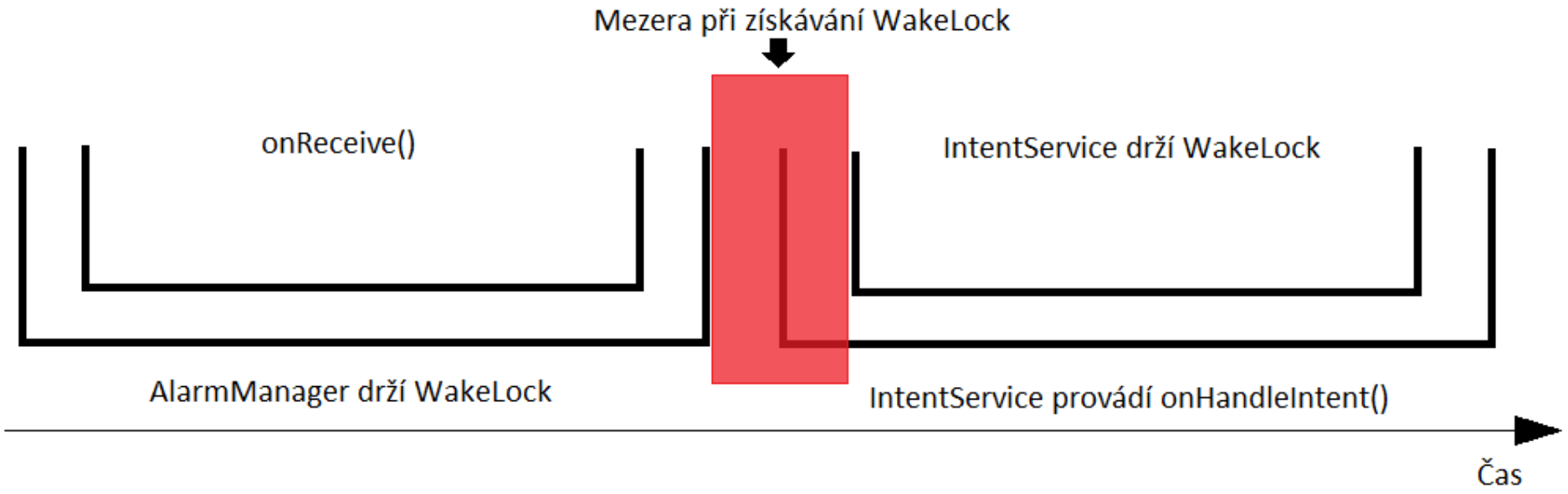
Problém spícího zařízení – řešení?

- AlarmManager spustí alarm a drží tzv. WakeLock
- PendingIntent je zachycen pomocí Broadcast Receiveru
- Brodcast Receiver spustí IntentService

Stejný problém jako v minulém případě!

- Následně je spuštěna IntentService na pozadí
- Je dokončena metoda `onReceive()` a AlarmManager uvolní WakeLock

Problém spícího zařízení – řešení?



Problém spícího zařízení – řešení!

- Je nutné si vytvořit vlastní WakeLock ještě předtím, než je spuštěna Service
- Vše musí proběhnout již při vykonávání `onReceive()`
- Další problém - WakeLock není Parcelable
- Možné řešení
  - Použít sdílenou statickou proměnnou
  - Nutnost vyřešit několik problémů

Pro WakeLock je potřebné oprávnění

Získání WakeLock

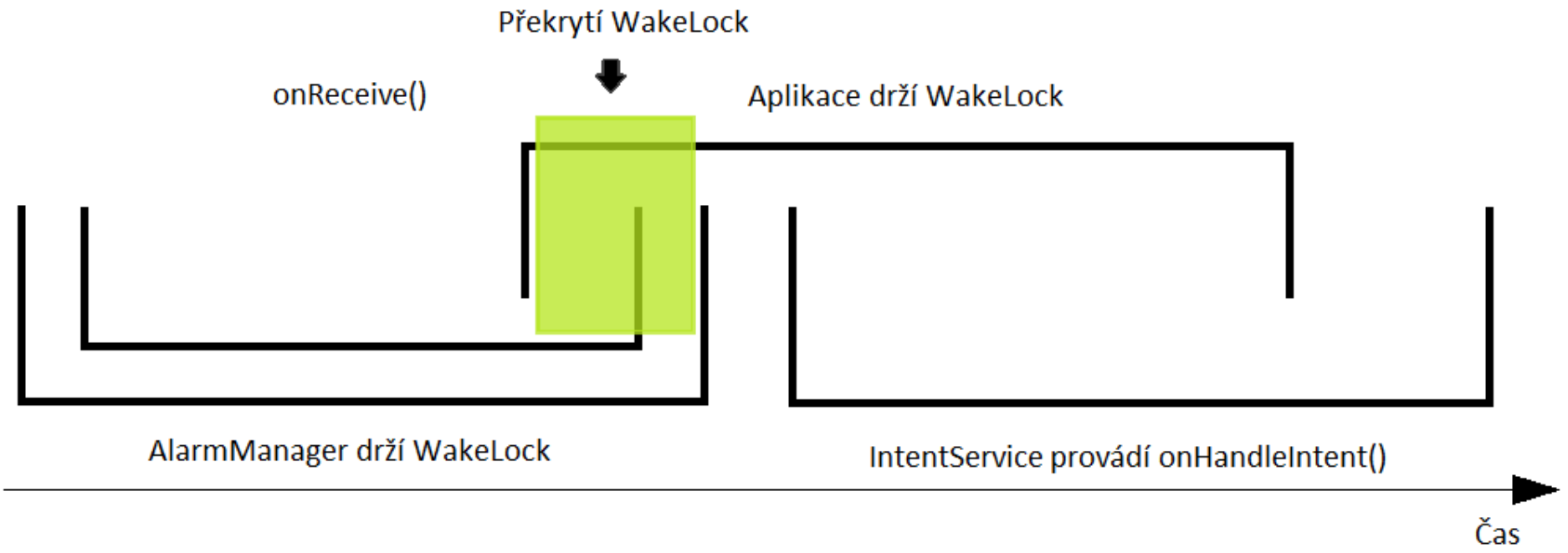
```
1.  private static final String LOCK_NAME_STATIC = "cz.cvut.and";
2.  private static PowerManager.WakeLock lockStatic = null;
3.  synchronized private static PowerManager.WakeLock
       getLock(Context context) {
4.   if (lockStatic == null) {
5.     PowerManager mgr = (PowerManager)
       Context.getSystemService(Context.POWER_SERVICE);
6.     lockStatic = mgr.newWakeLock(
       PowerManager.PARTIAL_WAKE_LOCK, LOCK_NAME_STATIC);
7.     lockStatic.setReferenceCounted(true);
8.    }
9.   return (lockStatic);
10.}
```

Získání WakeLock

- V `onReceive()` zavolání statické metody, která
  - Zavolá `getLock()` a na získaném zámku zavolá `acquire()`
  - Nastartuje IntentService

- IntentService v metodě `onHandleIntent()` musí v části `finally` zajistit navrácení zámku
  - `getLock(this).release()`

Překrytí WakeLock



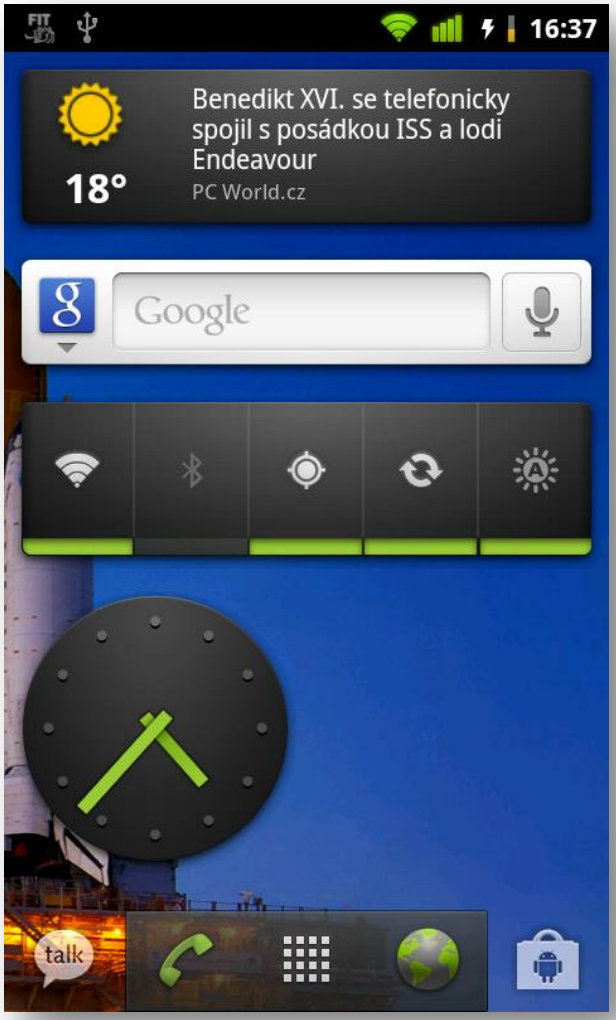
Background Data Setting

- Uživatel může nastavit, aby se data na pozadí nestahovala
- Service by měl toto nastavení zkontrolovat pomocí `getBackgroundDataSetting()` u systémové služby ConnectivityManager
- Lze rovněž odchytit broadcast `ConnectivityManager.`*`ACTION_BACKGROUND_DATA_SETTING_CHANGED`* a úplně zrušit nastavení alarmu

App Widgets

Přidáno až v Android 1.5, kvůli vyřešení problému se zabezpečením

- Vizuální komponenty aplikace, které mohou být přidány do domovské obrazovky
- Běží v procesu domovské obrazovky
- Skládají se z těchto 3 částí:
  - Layout
  - XML definující parametry widgetu
  - BroadcastReceiver, který ovládá widget



App Widgets – omezení architektury

- Nelze použít všechny kontejnery a widgety
- Jediný uživatelský vstup pochází od Button / ImageButton
- Nelze použít OnClickListener. Náhradou jsou PendingIntens
- Samotný vzhled je v podobě tzv. RemoteViews, které jsou vytvářeny vždy znovu
- Omezení vyplývající z BroadcastReceiveru

App Widgets – layout

- Lze použít pouze tyto kontejnery:
  - FrameLayout
  - LinearLayout
  - RelativeLayout
- Lze použít pouze tyto widgety:
  - AnalogClock
  - Button
  - Chronometer
  - ImageButton
  - ImageView
  - Progressbar
  - TextView
  - ViewFlipper

Seznam rozšířen
s příchodem Honeycomb
např. o ListView, GridView,
ViewFlipper

App Widgets – IntentReceiver

- Widgety se do AndroidManifest.xml přidávají jako běžné BroadcastReceivery, pouze se přidají navíc dva tagy do části meta-data
  - Intent-filter pro *android.appwidget.action.APPWIDGET_UPDATE*
  - Odkaz na XML s parametry widgetu

```
1. <receiver
2.     android:name=".ExampleWidget"
3.     android:label="Muj widget">
4.     <intent-filter>
5.         <action
6.             android:name="android.appwidget.action.APPWIDGET_UPDATE" />
7.     </intent-filter>
8.     <meta-data
9.         android:name="android.appwidget.provider"
10.        android:resource="@xml/example_widget_provider" />
11. </receiver>
```

AppWidgetProviderInfo

- Widget je definován XML souborem uloženým v *res/xml/*

```
1. <appwidget-provider
2.       xmlns:android="http://schemas.android.com/apk/res/android"
3.       android:minWidth="144dp"
4.       android:minHeight="72dp"
5.       android:updatePeriodMillis="1800000"
6.       android:configure="cz.cvut.examplewidget.Configure"
7.       android:initialLayout="@layout/main" />
```

- `minWidth`, `minHeight` – minimální šířka a výška, kterou zabírá widget
  - Systém se bude snažit RemoteView „roztáhnout“ do buněk na homescreenu
  - Standardní homescreen je rozdělen na 4x4 buněk
- `updatePeriodMillis` – Interval v ms, jak často bude systém widget aktualizovat
  - Zavolá se `onUpdate()` - negarantován přesný
  - Minimální hodnota je 30 minut

Nastavit na nula a použít AlarmManager

AppWidgetProviderInfo

- `initialLayout` – odkaz na layout definován v res/layout/
  - Layout použitý, než se zavolá poprvé metoda onUpdate()
  - Lze nastavit na invisible a zvolit vhodný vzhled podle „prostředí“
- `configure` – activita, která se spustí při přidání widgetu (volitelné)
  - Musí být celá cesta i se jménem balíčku
  - Viz. dále
- Lze přidat i více elementů *meta-data* do *Androidmanifest.xml* pro širší paletu widgetů

AppWidgetProvider

- Widgety jsou aktualizovány pomocí BroadcastReceiveru s Intent filtry, které reagují na:
  - `AppWidgetManager.`*`ACTION_APPWIDGET_UPDATE`*,
  - `AppWidgetManager.`*`ACTION_APPWIDGET_DELETED`*
  - `AppWidgetManager.`*`ACTION_APPWIDGET_ENABLED`*
  - `AppWidgetManager.`*`ACTION_APPWIDGET_DISABLED`*

- `AppWidgetProvider`
  - Usnadňuje zpracovávání těchto akcí
  - Potomek BroadcastReceiveru

AppWidgetProvider

```
1.  public class ExampleWidget extends AppWidgetProvider {
2.        @Override
3.        public void onUpdate(Context context,
4.                  AppWidgetManager appWidgetManager,
5.                  int[] appWidgetIds) {
6.              //aktualizace widgetu
7.        }
8.  }
```

AppWidgetProvider - metody

- `onUpdate(context, appWidgetManager, appWidgetIds)`
  - Volá se v pravidelném intervalu definovaném pomocí `updatePeriodMillis` v AppWidgetProviderInfo XML
  - Pokud není deklarována konfigurační activita, zavolá se i po přidání widgetu na plochu
- `onDeleted(context, appWidgetIds)`
  - Zavolá se pokaždé, když je widget odebrán z plochy

AppWidgetProvider - metody

- `onEnabled(context)`
  - Zavolá se po prvním přidání widgetu na plochu
  - Nastavení společné pro všechny widgety
  - Pokud uživatel přidá dva widgety, zavolá se pouze při přidání prvního
- `onDisabled(context)`
  - Zavolá se po odebrání posledního widgetu
- `onReceive(context, intent)`
  - Volá se při každém broadcastu před výše zmíněnými metodami

RemoteViews

- Třída umožňující určitou „manipulaci“ s Views zobrazenými v jiné aplikaci
- Vždy vytvořen nový RemoteView – náročná operace

```
1.  RemoteViews views = new RemoteViews(context.getPackageName(),R.layout.main);
2.
3.  // Aktualizace textu v TextView
4.  views.setTextViewText(R.id.textView, "Novy text");
5.  views.setTextColor(R.id.textView, Color.BLUE);
6.  // Nastaveni obrazku v ImageView
7.  views.setImageViewBitmap(R.id.imageView, myBitmap);
8.  // Aktualizace ProgressBaru
9.  views.setProgressBar(R.id.progressbar, 100, 50, false);
10. // Aktualizace Chronometru
11. views.setChronometer(R.id.chronometer,
12.              SystemClock.elapsedRealtime(), null, true);
13.
14. //Spusteni activity Configure po kliku na imageButton
15. Intent i = new Intent(context, Configure.class);
16. PendingIntent pendingIntent = PendingIntent.getBroadcast(context, 0, i, 0);
17. views.setOnClickPendingIntent(R.id.imageButton, pendingIntent);
```

Nelze použít přímo findViewById()

AppWidgetProvider – onUpdate()

- AppWidgetProvider předá do `onUpdate()` tyto parametry
- `Context` – context, ve kterém běží BroadcastReceiver
  - Pozor na omezení plynoucí z BrodcastReceiveru
- `AppWidgetIds` – pole s ID aktuálně zobrazovaných widgetů
- `AppWidgetManager` – stará se o předání RemoteView
  - Zavoláním `appWidgetManager.updateAppWidget(appWidgetId, remoteViews)` se aktualizuje UI widgetu
  - Instanci lze získat z jiné metody pomocí `AppWidgetManager.getInstance(context)`

AppWidgetProvider – onUpdate()

```
1.  @Override
2.  public void onUpdate(Context context,
3.                  AppWidgetManager appWidgetManager, int[] appWidgetIds) {
4.      final int N = appWidgetIds.length;
5.      // Projdeme vsechny widgety
6.      for (int i = 0; i < N; i++) {
7.              //ID aktualne prochazeneho widgetu
8.              int appWidgetId = appWidgetIds[i];
9.
10.             // Nastavime nas layout do RemoteViews
11.             RemoteViews views = new RemoteViews(context.getPackageName(),
12.                                                 R.layout.main);
13.             // nastavime text
14.             views.setTextViewText(R.id.text, appWidgetId + " text");
15.
16.             // Aktualizujeme UI widgetu
17.             appWidgetManager.updateAppWidget(appWidgetId, views);
18.     }
19.     super.onUpdate(context, appWidgetManager, appWidgetIds);
20. }
```

Pro náročnější operace spustit Service

Konfigurační activita

- Automaticky se spustí při vytvoření widgetu
- Umožňuje uživateli nakonfigurovat nastavení widgetu
- Definuje se v *AndroidManifest.xml* jako běžná activita s intent filtrem `android.appwidget.action.APPWIDGET_CONFIGURE`

```
1. <activity android:name=".Configure">
2.    <intent-filter>
3.        <action android:name=
4.             "android.appwidget.action.APPWIDGET_CONFIGURE" />
5.    </intent-filter>
6. </activity>
```

- Musí být také nastavena v AppWidgetProviderInfo XML

```
1. <appwidget-provider
2.      ...
3.      android:configure="cz.cvut.examplewidget.Configure"
4.      ... />
```

Konfigurační activita

- V Intentu, který spustil tuto activitu najdeme ID přidávaného widgetu pod extra polem `AppWidgetManager.`*`EXTRA_APPWIDGET_ID`*
- Pokud je nastavena konfigurační activita, při vytvoření widgetu se nezavolá `onUpdate()`
  - UI je potřeba nastavit zde
- Pokud konfigurační activita vrátí výsledek *`RESULT_CANCELED`*, widget se nevytvoří

Konfigurační activita

```
1.  AppWidgetManager appWidgetManager =
2.      AppWidgetManager.getInstance(this);
3.
4.  //ziskani ID pridavaneho widgetu
5.  Intent intent = getIntent();
6.  int mAppWidgetId = intent.getIntExtra(
7.              AppWidgetManager.EXTRA_APPWIDGET_ID,
8.              AppWidgetManager.INVALID_APPWIDGET_ID);
9.
10. //nastaveni UI
11. RemoteViews views = new RemoteViews(getPackageName(), R.layout.main);
12. views.setTextViewText(R.id.text, "text");
13. appWidgetManager.updateAppWidget(mAppWidgetId, views);
14.
15. //nastaveni vysledku a ukonceni activity
16. Intent resultValue = new Intent();
17. resultValue.putExtra(AppWidgetManager.EXTRA_APPWIDGET_ID
18.                     , mAppWidgetId);
19. setResult(RESULT_OK, resultValue);
20. finish();
```

Další zdroje

- http://developer.android.com/resources/articles/timed-ui-updates.html
- http://developer.android.com/guide/practices/ui_guidelines/widget_design.html
- http://developer.android.com/guide/topics/appwidgets/index.html