AppWidgets

AppWidget

- Good news: Bardzo proste!
- AppWidget = specjalny BroadcastReciever
- a rozmiar etc, deklarujemy w **res/xml/my_widget.xml**

AppWidgetProvider

AppWidgetProvider, musi zostać zarejestrowany w **AndroidManifest.xml** (w <application/>):

```
<receiver android:name=".ui.appwidgets.MyWidgetProvider"
  <intent-filter>
    <action android:name="android.appwidget.action.APPWI
  </intent-filter>
  <meta-data android:name="android.appwidget.provider"
             android:resource="@xml/my_widget"/>
</receiver>
```

res/xml/**my_widget.xml**

```
<appwidget-provider xmlns:android="http://schemas.androi
    android:minWidth="294dp"
    android:minHeight="72dp"
    android:updatePeriodMillis="86400000"
    android:previewImage="@drawable/preview_widget"
    android:initialLayout="@layout/widget">
</appwidget-provider>
```

Deklarowanie tego w XML jest wygodniejsze - mamy filtrowanie folderów (-v11).

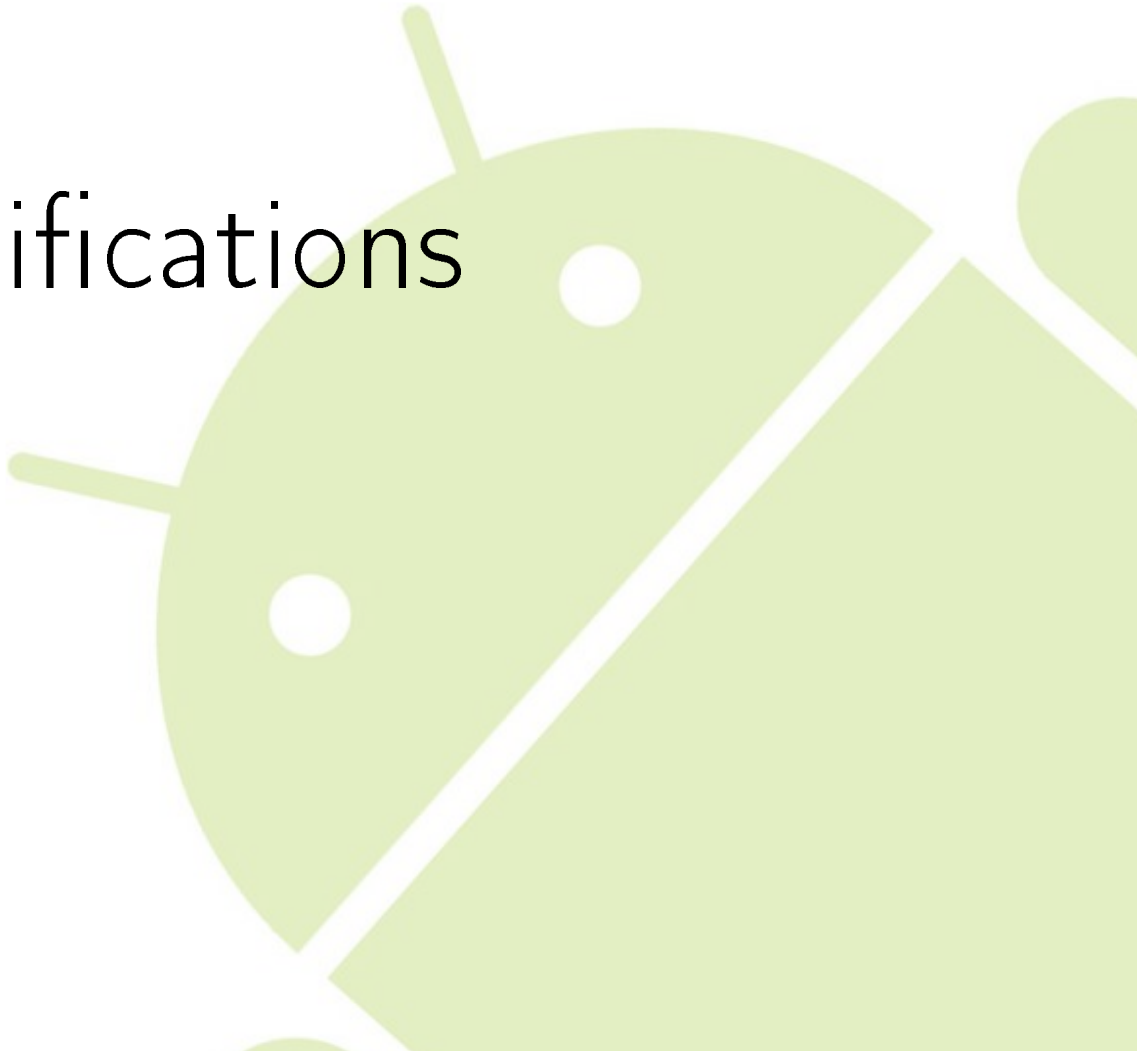
AppWidget - implementacja

```
public class MyWidgetProvider extends AppWidgetProvider

  @Override
  public void onUpdate(Context context,
                       AppWidgetManager appWidgetManager
                       int[] appWidgetIds) {

    // Provider obsluguje WIELE (N) widzetow!
    final int N = appWidgetIds.length;

    // aktualizujemy kazgego
    for (int i = 0; i < N; i++) {
      int appWidgetId = appWidgetIds[i];
      populateView(context, appWidgetManager,
                           appWidgetId);

    }
  }
```

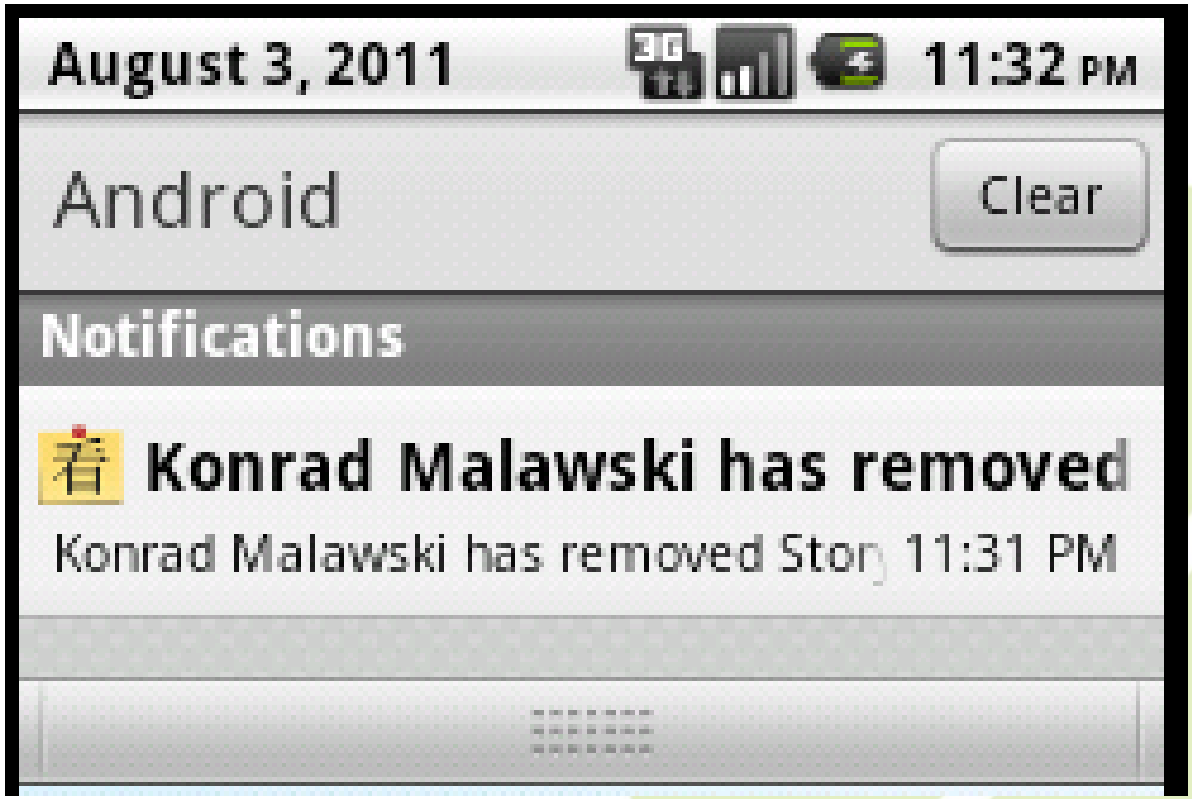
AppWidget - implementacja

```
private void populateView(Context context, AppWidgetMana
    // Przygotowujemy intent do odpalenia "on click"
    Intent intent = new Intent(context, ViewDetailsActivit
    PendingIntent pendingIntent = PendingIntent.getActivit

    // rejestrujemy onClickListener'a troszke inaczej:
    RemoteViews views = new RemoteViews(context.getPackageN
    views.setOnClickPendingIntent(R.id.container, pendingI

    // aktualizujemy widok widzetu (prosimy menagera o to)
    appWidgetManager.updateAppWidget(appWidgetId, views);
}
} // end of class
```

Notifications



Notificaion - przykład



NorificationManager

```
class MyActivity extends RoboActivity {

   @Inject
   NotificationManager notificationManager;

}
```

NorificationManager

```
class MyActivity extends RoboActivity {

   @Inject
   NotificationManager notificationManager;

}
```

Albo oczywiście **Service**.

NorificationManager - 1/3

```
int icon = R.drawable.ic_kanbanery;
long when = System.currentTimeMillis();

Notification notification = new Notification(icon, title

// ...
```

NorificationManager - 2/3

```
int icon = R.drawable.ic_kanbanery;
long when = System.currentTimeMillis();

Notification notification = new Notification(icon, title

Intent notificationIntent = new Intent(this, ColumnsAct
PendingIntent onClickIntent = PendingIntent.getActivity(

// ...
```

NotificationManager - 3/3

```
int icon = R.drawable.ic_kanbanery;
long when = System.currentTimeMillis();

Notification notification =
                    new Notification(icon, title, when);

Intent notificationIntent = new Intent(this,
                                       ColumnsActivity.class);
PendingIntent contentIntent = PendingIntent
        .getActivity(this, 0, notificationIntent, 0);

notification.setLatestEventInfo(context, title,
                                    msg, contentIntent);
notification.flags = Notification.FLAG_AUTO_CANCEL;


notificationManager.notify(ACTION_ID, // explain
                                  notification);
```

Service

Service

- Service uruchamiany jest na tym samym głównym wątku co Activity (UIThread)

Service

- Service uruchamiany jest na tym samym głównym wątku co Activity (UIThread)
- Oczywiście nie wolno mu go „zablokować”

Service

- Service uruchamiany jest na tym samym głównym wątku co Activity (UIThread)
- Oczywiście nie wolno mu go „zablokować”
- Service działa „w tle”, nie ma UI

Service

- Service uruchamiany jest na tym samym głównym wątku co Activity (UIThread)
- Oczywiście nie wolno mu go „zablokować”
- Service działa „w tle”, nie ma UI
- Service to **NIE** osobny proces

Service

- Service uruchamiany jest na tym samym głównym wątku co Activity (UIThread)
- Oczywiście nie wolno mu go „zablokować”
- Service działa „w tle”, nie ma UI
- Service to **NIE** osobny proces
- Service to **NIE** osobny wątek

Service

- Service uruchamiany jest na tym samym głównym wątku co Activity (UIThread)
- Oczywiście nie wolno mu go „zablokować”
- Service działa „w tle”, nie ma UI
- Service to **NIE** osobny proces
- Service to **NIE** osobny wątek
- Kontunuuje działanie nawet po zamknięciu wszelkich Activity aplikacji

2 „rodzaje” Service (choć ta sama klasa)

Mimo, że zawsze mówimy o **android.app.Service**, możemy mieć na myśli 2 „typy” Service:

2 „rodzaje" Service (choć ta sama klasa)

Mimo, że zawsze mówimy o **android.app.Service**, możemy mieć na myśli 2 „typy" Service:

- „**Uruchamiany**" celem zrobienia czegoś przez nas - **startService(Intent)**

2 „rodzaje" Service (choć ta sama klasa)

Mimo, że zawsze mówimy o **android.app.Service**, możemy mieć na myśli 2 „typy" Service:

- „**Uruchamiany**" celem zrobienia czegoś przez nas - **startService(Intent)**
- „**Udostępniany**" celem udostępnienia komuś API, nawet zewnętrznym aplikacjom! - tutaj mowa o **IBinder** i **onBind()**. (Bardzo zaawansowane rzeczy można tutaj robić, vide AIDL)

2 „rodzaje" Service (choć ta sama klasa)

Mimo, że zawsze mówimy o **android.app.Service**, możemy mieć na myśli 2 „typy" Service:

- „**Uruchamiany**" celem zrobienia czegoś przez nas - **startService(Intent)**
- „**Udostępniany**" celem udostępnienia komuś API, nawet zewnętrznym aplikacjom! - tutaj mowa o **IBinder** i **onBind()**. (Bardzo zaawansowane rzeczy można tutaj robić, vide AIDL)

Nas interesuje jedynie pierwszy rodzaj serwisu.

Service lifecycle

- **onCreate()** - gdy jeszcze nie był utworzony, a zawołano **startService()**

Service lifecycle

- **onCreate()** - gdy jeszcze nie był utworzony, a zawołano **startService()**
- **onStart()** - przy uruchomieniu Serwisu, po onCreate() - tutaj umieszczamy „logikę”

Service lifecycle

- **onCreate()** - gdy jeszcze nie był utworzony, a zawołano **startService()**
- **onStart()** - przy uruchomieniu Serwisu, po onCreate() - tutaj umieszczamy „logikę”
- **onBind()** - w przypadku „uruchomienia” przez **bindService()**, na razie nas nie interesuje - możemy zwracać **null**

Service lifecycle

- **onCreate()** - gdy jeszcze nie był utworzony, a zawołano **startService()**
- **onStart()** - przy uruchomieniu Serwisu, po onCreate() - tutaj umieszczamy „logikę”
- **onBind()** - w przypadku „uruchomienia” przez **bindService()**, na razie nas nie interesuje - możemy zwracać **null**
- **onStartCommand()** - wołany za każdym radem gdy ktoś startuje service. *OnCreate() nie zostałby zawołany jak 2 razy pod rząd zastartujesz serwis!*

Service lifecycle

- **onCreate()** - gdy jeszcze nie był utworzony, a zawołano **startService()**
- **onStart()** - przy uruchomieniu Serwisu, po onCreate() - tutaj umieszczamy „logikę”
- **onBind()** - w przypadku „uruchomienia” przez **bindService()**, na razie nas nie interesuje - możemy zwracać **null**
- **onStartCommand()** - wołany za każdym radem gdy ktoś startuje service. *OnCreate() nie zostałby zawołany jak 2 razy pod rząd zastartujesz serwis!*
- **onDestroy()** - wiadomo, gdy servis zostaje zatrzymywany

Service lifecycle

- **onCreate()** - gdy jeszcze nie był utworzony, a zawołano **startService()**
- **onStart()** - przy uruchomieniu Serwisu, po onCreate() - tutaj umieszczamy „logikę”
- **onBind()** - w przypadku „uruchomienia” przez **bindService()**, na razie nas nie interesuje - możemy zwracać **null**
- **onStartCommand()** - wołany za każdym radem gdy ktoś startuje service. *OnCreate() nie zostałby zawołany jak 2 razy pod rząd zastartujesz serwis!*
- **onDestroy()** - wiadomo, gdy servis zostaje zatrzymywany
- ciekawostka: **onLowMemory()** - gdy zaczyna brakować pamięci w systemie. Po zwróceniu z tej metody android przeprowadzi Garbage Collection.

Service - AndroidManifest.xml

```
<application>
<!-- ... -->
<service android:name=".service.MyService"/>
</application>
```

Service - implementacja

```
public class MyService extends RoboService {
  public IBinder onBind(Intent intent) {
    return null;
  }

  @Override
  public void onCreate() {
    // ...
  }
}
```

Service - implementacja

Częsta implementacja:

```
public class MyService extends RoboService {
  // ...

  Timer myTimer;

  @Override
  public void onCreate() {
    // ...
    myTimer = new Timer();
    myTimer.schedule(new DoStuffTimerTask(/**/), M, M);
  }
}
```

Service - sendBroadcast(Intent)

Jedno z popularniejszych zastosowań - service pracuje w tle, a następnie powiadamia „zainteresowanych” że skończył.

```
class DoStuffTimerTask {
  public void run(){
    int number = random.nextInt();

    Intent intent = new Intent("pl.llp.NEW_NUMBER")
    intent.putExtra("number", number);
    sendBroadcast(intent);
  }
}
```

Być może zadanko?

- **AppWidget** który będzie pokazywał wartość którą generuje cyklicznie **Service**

Być może zadanko?

- **AppWidget** który będzie pokazywał wartość którą generuje cyklicznie **Service**
- Service ma powiadamiać AppWidget poprzez sendBroadcast()

Być może zadanko?

- **AppWidget** który będzie pokazywał wartość którą generuje cyklicznie **Service**
- Service ma powiadamiać AppWidget poprzez sendBroadcast()
- AppWidget (jest reciever'em) i w AndroidManifest musi również słuchać na nasz nowy intent

Być może zadanko?

- **AppWidget** który będzie pokazywał wartość którą generuje cyklicznie **Service**
- Service ma powiadamiać AppWidget poprzez sendBroadcast()
- AppWidget (jest reciever'em) i w AndroidManifest musi również słuchać na nasz nowy intent
- implementujemy w nim onRecieve - również oprócz onUpdate!

Być może zadanko?

- **AppWidget** który będzie pokazywał wartość którą generuje cyklicznie **Service**
- Service ma powiadamiać AppWidget poprzez sendBroadcast()
- AppWidget (jest reciever'em) i w AndroidManifest musi również słuchać na nasz nowy intent
- implementujemy w nim onRecieve - również oprócz onUpdate!

Bardzo możliwe że traficie na kilka „ale powinno działać" - ping me w razie problemów.

Hackaround... w razie problemów.

```
public void onReceive(Context context, Intent intent) {
  super.onReceive(context, intent);

  if (intent.getAction()
      .equals(RandomNumbersService
              .NUMBER_INTENT_ACTION_NAME)) {

    Bundle extras = intent.getExtras();
    // ...

    // force update !
    AppWidgetManager mngr = AppWidgetManager
                                .getInstance(context);
    int[] appWidgetIds =
            mngr.getAppWidgetIds(intent.getComponent());

    onUpdate(context, mngr, appWidgetIds);
  }
}
```