KAPITOLA 6

Notifikace, alarmy

V této kapitole:
- Notifikace
- Alarmy

Notifikace

Notifikace jsou zprávy, někdy i rozsáhlejší, které potřebuje aplikace ve vhodné situaci zobrazit uživateli mimo běžné uživatelské rozhraní aplikace.

Android podporuje dva různé druhy upozornění. Zprávy Toast, jejichž název je odvozen od toho, že zpráva „vyskočí" na obrazovku aplikace doslova jako topinka z topinkovače. Zprávy Toast jako nejjednodušší prostředek na výpis textu jste už mnohokrát použili v aplikacích v předchozích kapitolách. Druhým typem je oznámení v oznamovací oblasti nebo panelu oznámení.

Tato oznámení jsou viditelná a mohou přistupovat k systémům řízení oznamovací oblasti v horní části zařízení.

Poznámka: Notifikace se často používají k upozornění uživatele, že nějaká činnost, která trvá déle, byla dokončena. Typickým příkladem je stahování souborů z Internetu.

Příklad – zobrazení toastu s vlastním designem

Klasický toast je text na pozadí, který se zobrazí v okénku. V tomto příkladu vytvoříte vlastní návrh okna toastu a kromě textu zobrazíte i obrázek. S návrhem layoutu si nemusíte dělat žádné starosti, stačí na plochu hlavní aktivity přidat tlačítko.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical" >

  <Button
    android:id="@+id/btToast"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
```

```
    android:text="Toast"/>

</LinearLayout>
```

Více pozornosti budeme věnovat návrhu vlastního layoutu toastu v souboru *muj_toast.xml*. Layout obsahuje prvek `ImageView` na zobrazení ikony a `TextView` na zobrazení textu oznámení.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/toast_layout_root"
  android:orientation="horizontal"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:padding="10dp"
  android:background="#7777" >
  <ImageView android:id="@+id/image"
    android:layout_width="64dp"
    android:layout_height="64dp"
    android:layout_marginRight="10dp"
    android:src="@drawable/ic_launcher"
    android:layout_centerVertical="true"
    android:contentDescription="popis" />
  <TextView android:id="@+id/text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#FFF"
    android:text="Text oznámení!"
    android:layout_centerVertical="true"
    android:layout_toRightOf="@id/image"
    android:textSize="40sp" />
</RelativeLayout>
```

Obrázek 6.1: Zobrazení toastu s vlastním layoutem

Kód hlavní aktivity po stisknutí tlačítka zobrazí toast. V metodě `setView` se naplní layout toastu z XML souboru.

```
public class MainActivity extends Activity
{

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }
  public void onClick(View View)
  {
    Toast toast = new Toast(getApplicationContext());
    toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
    toast.setDuration(Toast.LENGTH_LONG);
    toast.setView(getLayoutInflater().inflate(R.layout.muj_toast,null));
    toast.show();
  }
}
```

Příklad – poslání notifikace

Nevýhodou toastu je, že se na definovanou dobu zobrazí a potom zmizí. Pokud se uživatel právě na displej nedívá, notifikaci neuvidí. V následujícím příkladu pošle aplikace notifikaci, která se zobrazí v oznamovací oblasti, kde si ji uživatel může kdykoliv zobrazit.

Pomocí panelu oznámení mohou aplikace i Android samotný informovat uživatele o různých událostech. Panel oznámení je možné gestem tažení prstu směrem dolů vysunout a zobrazit další informace, které byly umístěny v oznamovací oblasti.

Upozornění: Z procesu běžícího na pozadí, bez ohledu na to, zda se jedná o službu nebo Broadcast Receiver, nespouštějte aktivity přímo. Určitě byste se uživateli nezavděčili, spíš byste jej rozzlobili, jelikož nevíte, co právě dělá nebo co má v úmyslu. Namísto toho vytvořte notifikaci, která mu poskytne požadovanou informaci, a uživatel se sám rozhodne, zda spustí nějakou aplikaci.

Aby bylo možné přehrání zvuku, vytvořte ve složce *res* složku *raw* a do ní nakopírujte soubor ve formátu MP3 se zvukem, který má zaznít při upozornění. Soubor se zvukem snadno získáte z Internetu.

Layout hlavní aktivity obsahuje jedno tlačítko, podobně jako v předchozí aplikaci. Liší se jen textem, jelikož neposíláme toast, ale notifikaci.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical" >

  <Button
    android:id="@+id/btNotifikace"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="onClick"
    android:text="Notifikace"/>
</LinearLayout>
```

Abychom demonstrovali co nejvíce funkcionality týkající se oznámení, vytvoříme další aktivitu. Tato se spustí, když uživatel klepne na oznámení v panelu oznámení. Aktivita v tomto příkladu vypíše konstantní textové oznámení. V reálné aplikaci by vypsala podrobnosti o události, která notifikaci vyvolala.

Layout v souboru *activity_notif.xml*:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical" >

  <TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
    android:text="Aktivita spuštěna z oznámení"
    android:textSize="24sp">
  </TextView>
</LinearLayout>
```

Soubor *NotifActivity.java*:

```
public class NotifActivity extends Activity
{
  @Override
  protected void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_notif);
  }
}
```

Nezapomeňte tuto aktivitu přidat do seznamu aktivit v souboru *AndroidManifest.xml*.

```
<activity android:name=".NotifActivity" />
```

V kódu hlavní aktivity definujeme texty nadpisu a textu oznámení a také zvuk a vzor vibrací, které uživatele na oznámení upozorní. V metodě `onCreate` vytvoříme záměr, který se odešle po klepnutí na oznámení v panelu oznámení. Záměr spustí aktivitu `NotifActivity`.

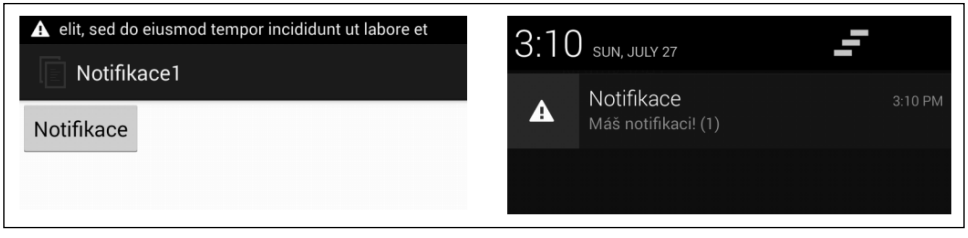
Ke správě notifikací je určena systémová služba `NotificationManager`. Notifikace bude definovaná prostřednictvím objektu `Notification.Builder`.

```
public class MainActivity extends Activity
{
  private static final int NOTIF_ID = 1;
  private int pocetNotifikaci;

  private Intent notifIntent;
  private PendingIntent pendingIntent;

  private final CharSequence ticker = "Lorem ipsum dolor sit amet...!";
  private final CharSequence nadpis = "Notifikace";
  private final CharSequence oznam = "Máš notifikaci!";

  private Uri soundURI =
    Uri.parse("android.resource://com.example.notifikace1/" + R.raw.alarm);
  private long[] vibrace = { 0, 300, 200, 300 };

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    notifIntent = new Intent(getApplicationContext(), NotifActivity.class);
    pendingIntent = PendingIntent.getActivity(getApplicationContext(), 0,
      notifIntent, Intent.FLAG_ACTIVITY_NEW_TASK);
  }
```

```
  public void onClick(View View)
  {
    Notification.Builder notificationBuilder = new Notification.Builder(
      getApplicationContext())
      .setTicker(ticker)
      .setSmallIcon(android.R.drawable.stat_sys_warning)
      .setAutoCancel(true)
      .setContentTitle(nadpis)
      .setContentText(oznam + " (" + ++pocetNotifikaci + ")")
      .setContentIntent(pendingIntent).setSound(soundURI)
      .setVibrate(vibrace);

    NotificationManager mNotificationManager = (NotificationManager)
      getSystemService(Context.NOTIFICATION_SERVICE);
    mNotificationManager.notify(NOTIF_ID,
      notificationBuilder.build());
  }
}
```

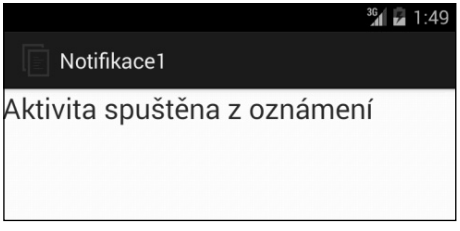
Po stisknutí tlačítka aplikace vyšle systému notifikaci. Ta se projeví zvukem, vibrací, zobrazením ikony v oznamovací oblasti a postupným rolováním textu. Úmyslně jsme zadali dlouhý text, aby se rolování projevilo. Po odrolování textu zůstane zobrazená pouze ikona notifikace.

Po zobrazení panelu oznámení se zobrazí oznámení vaší aplikace. To sestává z ikony, názvu, textu oznámení a data. Oznámení také obsahuje číslo v závorkách, které udává, kolikrát bylo oznámení přijato. Zkuste poslat oznámení z aplikace znovu a toto číslo v závorkách se bude inkrementovat.



Obrázek 6.2: Test aplikace posílající notifikaci

Pokud uživatel klepne na oznámení v okně notifikačního centra, spustí se druhá aktivita aplikace, která vypíše textové oznámení.



Obrázek 6.3: Aktivita spuštěná po klepnutí na notifikaci

Alarmy

V dosud probíraných příkladech aplikace realizovala činnost, případně sekvenci činností, ihned po spuštění. Často se však vyskytuje požadavek, aby aplikace dělala něco v přesně stanovený čas, například aby za 15 minut uživatele na něco upozornila, případně aby něco dělala periodicky. K tomuto účelu slouží alarmy.

Poznámka: Alarm na platformě Android je mechanizmus umožňující poslání objektu `intent` do budoucnosti, do přesně definovaného časového okamžiku nebo do více časových okamžiků.

Úlohou aplikace je vytvořit a nastavit alarmy. K tomuto účelu je k dispozici rozhraní `Alarm Manager`, umožňující nastavení a zrušení alarmů. Po vytvoření a nastavení alarmů a ukončení aplikace, která to realizovala, může zařízení přejít do režimu spánku a aktivuje se až v definovaný časový okamžik.

Upozornění: Alarmy nastavené aplikacemi budou zrušeny po úplném vypnutí, případně restartování zařízení.

Vytvoření alarmu

Pomocí metody

```
getSystemService(Context.ALARM_SERVICE)
```

musí aplikace nejprve získat referenci na `AlarmManager`.

Jednorázový alarm nastavíte pomocí metody

```
void set(int typ, long casAktivace, PendingIntent operace)
```

K nastavení alarmu opakujícího se v periodických časových intervalech použijete metodu

```
void setRepeating(int typ, long casAktivace, long interval,
  PendingIntent operace)
```

Nebo můžete použít metodu

```
void setInexactRepeating(int typ, long casAktivace, long interval,
  PendingIntent operace)
```

Poznámka: Počínaje verzí API 19 (KitKat) jsou všechny alarmy aktivované jako `inexact`, což ve verzích API <=18 zabrání opakovanému buzení zařízení při více těsně za sebou jsoucích alarmech. Jak vyplývá z názvu `inexact` (nepřesný), alarm se nemusí aktivovat v přesně nastaveném čase, ale například zařízení se probudí a alarm aktivuje současně s jiným alarmem nastaveným na vhodnou dobu. Třeba když je dvojice alarmů `inexact` naplánovaná s 15sekundovým zpožděním, systém je aktivuje současně.

Takovéto nastavení se dá použít tam, kde na přesném čase nezáleží, například u připomínky, že je potřeba nakrmit rybičky v akváriu a podobně.

K definování intervalu je v metodě `setInexactRepeating` potřeba použít předdefinované konstanty:

- INTERVAL_FIFTEEN_MINUTES
- INTERVAL_HALF_HOUR
- INTERVAL_HOUR
- INTERVAL_HALF_DAY
- INTERVAL_DAY

Parametr typ určuje, jakým způsobem se interpretuje parametr casAktivace: buď jako reálný čas, kdy hodnota parametru udává počet milisekund od půlnoci 1. ledna 1970 (RTC_...), nebo jako relativní čas od posledního spuštění systému (ELAPSED_...). Tento parametr zároveň určuje chování systému v případě, že nastal čas alarmu. Jednou z možností je probudit zařízení v okamžiku alarmu a doručit záměr (RTC_WAKEUP), druhou pokračovat ve stavu sleep a záměr doručit při nejbližším probuzení zařízení (RTC).

Příklad – ověření funkcionality alarmů

Na praktické ověření jednotlivých typů alarmů vytvoříme cvičný příklad. Jeho uživatelské rozhraní tvoří čtyři tlačítka. První tři aktivují alarmy a třetí slouží ke zrušení opakování alarmu.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:orientation="vertical" >

  <Button
    android:id="@+id/bt_single"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/nastav_single" >
  </Button>

  <Button
    android:id="@+id/bt_repeating"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/nastav_repeat" />

  <Button
    android:id="@+id/bt_inexact"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/nastav_inexact" />

  <Button
    android:id="@+id/bt_zrus"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/zrus_repeat" />
</LinearLayout>
```

Obrázek 6.4: Uživatelské rozhraní aplikace

Pro úplnost uvádíme i texty definované v souboru *strings.xml*.

```
<resources>
  <string name="app_name">Alarm Demo</string>
  <string name="text_upozorneni">Nakrm rybičky!</string>
  <string name="nastav_single">Nastav Single Alarm</string>
  <string name="nastav_repeat">Nastav Repeating Alarm</string>
  <string name="nastav_inexact">Nastav Inexact Repeating Alarm</string>
  <string name="zrus_repeat">Zruš Repeating Alarm</string>
</resources>
```

Dříve než přistoupíte k programování kódu hlavní aktivity, vytvořte třídu odvozenou od `BroadcastReceiver`, která bude přijímat záměry a zobrazovat notifikace. Abychom demonstrovali i přehrání zvuku, vytvořte ve složce *res* složku *raw* a do ní nakopírujte soubory ve formátu MP3 se zvukem, který má zaznít při alarmu. Soubor se zvukem alarmu snadno získáte z Internetu. Alarm se projeví i jako sekvence vibrací.

```
public class NotifReceiver extends BroadcastReceiver
{
  private static final int NOTIF_ID = 1;
  private static final String TAG = "NotifReceiver";

  private Intent notifIntent;
  private PendingIntent pendIntent;

  //zvuk a vibrace
  private Uri soundURI
    = Uri.parse("android.resource://com.example.alarmdemo/" + R.raw.alarm);
  private long[] mVibratePattern = { 0, 300, 100, 300 };

  @Override
  public void onReceive(Context context, Intent intent)
  {

    notifIntent = new Intent(context, MainActivity.class);
    pendIntent = PendingIntent.getActivity(context, 0,
      notifIntent, Intent.FLAG_ACTIVITY_NEW_TASK);

    Notification.Builder notificationBuilder = new
      Notification.Builder(context).setTicker("Upozornění!")
      .setSmallIcon(android.R.drawable.stat_sys_warning)
      .setAutoCancel(true).setContentTitle("Nezapomeň")
      .setContentText("nakrmit rybicky!!").setContentIntent(pendIntent)
      .setSound(soundURI).setVibrate(mVibratePattern);

    NotificationManager notifManag = (NotificationManager)
      context.getSystemService(Context.NOTIFICATION_SERVICE);
    notifManag.notify(NOTIF_ID, notificationBuilder.build());

    Log.i(TAG,"Notifikace:" +
      DateFormat.getDateTimeInstance().format(new Date()));

  }
}
```

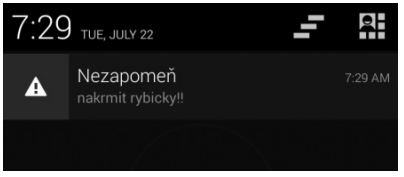
V kódu hlavní aktivity je inicializace objektu `AlarmManager`, vytvoření záměru pro notifikační `BroadcastReceiver` a v obsluze událostí klepnutí na tlačítka nastavení jednotlivých typů alarmů. V reálné aplikaci byste nastavení alarmu oznámili uživateli, například přes toast.

```
public class MainActivity extends Activity
{
  private AlarmManager alarmManag;
  private Intent notifIntent;
  private PendingIntent notifPendingIntent;
  private static final long SPOZDENI = 30000L;

  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    alarmManag = (AlarmManager) getSystemService(ALARM_SERVICE);
    notifIntent = new Intent(MainActivity.this, NotifReceiver.class);
    notifPendingIntent = PendingIntent.getBroadcast(MainActivity.this, 0,
      notifIntent, 0);

    final Button btSingle = (Button) findViewById(R.id.bt_single);
    btSingle.setOnClickListener(new OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
        alarmManag.set(AlarmManager.RTC_WAKEUP,
          System.currentTimeMillis() + SPOZDENI,
          notifPendingIntent);
      }
    });

    final Button btRepeating = (Button) findViewById(R.id.bt_repeating);
    btRepeating.setOnClickListener(new OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
        alarmManag.setRepeating(AlarmManager.ELAPSED_REALTIME,
          SystemClock.elapsedRealtime() + SPOZDENI,
          AlarmManager.INTERVAL_FIFTEEN_MINUTES,
          notifPendingIntent);
      }
    });

    final Button btInexact = (Button) findViewById(R.id.bt_inexact);
    btInexact.setOnClickListener(new OnClickListener()
    {
      @Override
      public void onClick(View v)
```

```
      {
        alarmManag.setInexactRepeating(
          AlarmManager.ELAPSED_REALTIME,
          SystemClock.elapsedRealtime() + SPOZDENI,
          AlarmManager.INTERVAL_FIFTEEN_MINUTES,
          notifPendingIntent);
      }
    });

    final Button cancelRepeatingAlarmButton = (Button)
      findViewById(R.id.bt_zrus);
    btSingle.setOnClickListener(new OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
         alarmManag.cancel(notifPendingIntent);
      }
    });
  }
}
```

Nezapomeňte do souboru *AndroidManifest.xml* přidat `receiver`:

```
<receiver android:name=".NotifReceiver">
</receiver>
```



Obrázek 6.5: Notifikace