

# ANDROID - UI

3. týden, KIV/MKZ 2016

L. Pešička

# EMULÁTOR

## klávesové zkratky:

*C:\sw\android\sdk\tools>emulator -help-keys*

Klávesová zkratka	činnost
Ctrl+F11, Ctrl+F12	Emulátor na výšku / šířku (na další či předchozí layout)
Home	Tlačítko Home
Escape	Tlačítko Back
F2, PageUp	Tlačítko Menu
Alt+Enter	fullscreen

Povídání ohledně rychlosti emulátoru:

<https://infinum.co/the-capsized-eight/articles/is-your-android-emulator-just-too-slow>

# EMULÁTOR - GENYMOTION

- ◉ <https://www.genymotion.com/features/>
- ◉ Individual - Basic - Free

## Camera

Use your laptop webcam as the video source of Android of your camera.

## Plugins / Testing tools

Genymotion is compatible with Android SDK tools, Eclipse and Android Studio.

## Battery

Test your app with various charge levels and see how it handles those use cases.

## Browser

Test your website in various Android browsers: Webkit for Android, Firefox for Android and many more!

## GPS

Use the GPS widget to easily develop and test your geolocation-based apps.

## Perfectly adapted

Genymotion works on Linux, Windows, Mac OS X.



# REÁLNÉ ZAŘÍZENÍ

## Windows

- ⦿ USB driver pro konkrétní telefon
- ⦿ ze stránek výrobce, seznam:

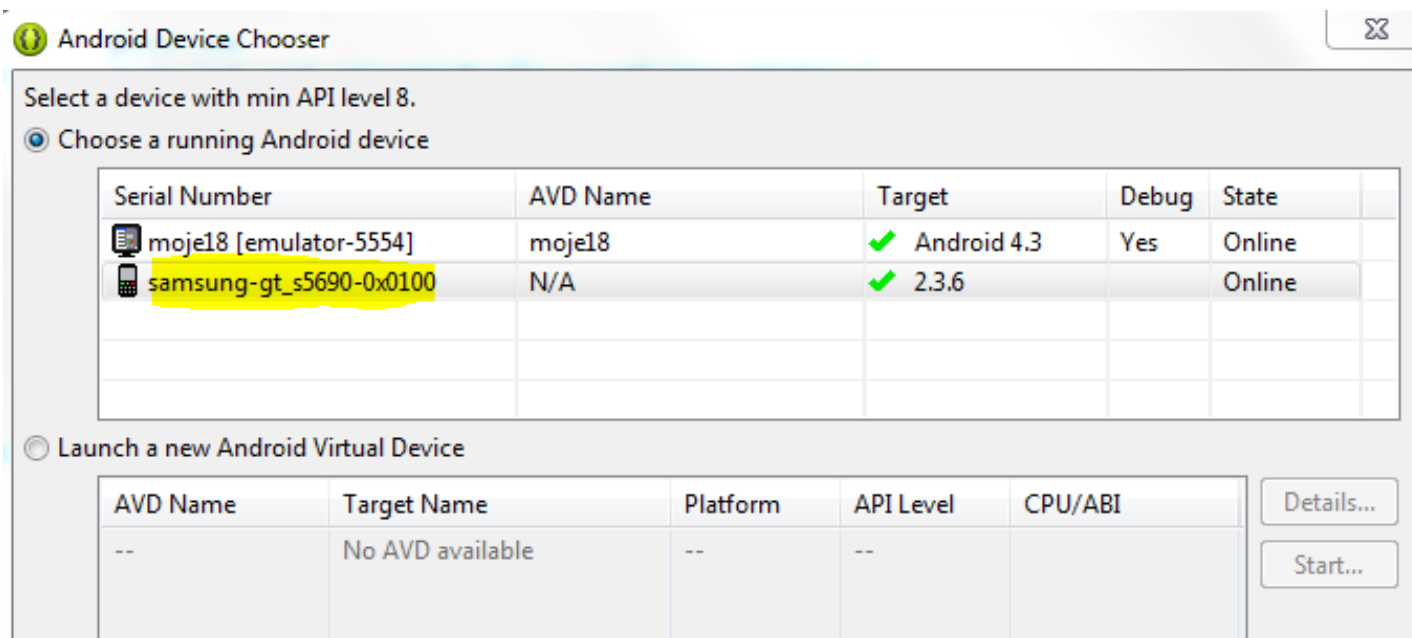
<http://developer.android.com/tools/extras/oem-usb.html>

## Linux

- ⦿ viz návod na oficiálních stránkách:
- ⦿ důležitý parametr Vendor ID
- ⦿ viz:

<http://developer.android.com/tools/device.html#VendorIds>

# VÝBĚR MEZI EMULÁTOREM A REÁLNÝM ZAŘÍZENÍM



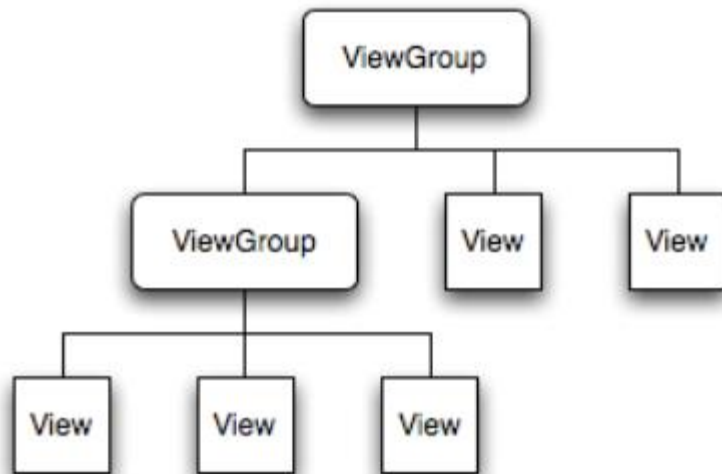
zde můžeme volit mezi reálným zařízením Samsung s Androidem 2.3.6, nebo emulátorem s Androidem 4.3

# OBSAH

- Prvky UI
  - Widgety, Layouty
  - Alternativní resources
- Notifikace
- Menu
- Kontextové menu
- Spuštění jiné aktivity

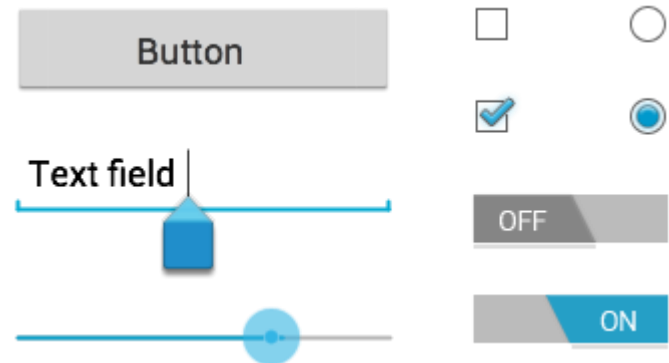
# USER INTERFACE

- ◉ objekty **ViewGroup** a **View**
- ◉ **ViewGroup** - slouží pro layouts „rozvržení“
- ◉ **View** - pro widgety, UI objekty



pro zobrazení  
hierarchie objektů  
zavoláme  
**setContentView()**

# VIEW (WIDGET)



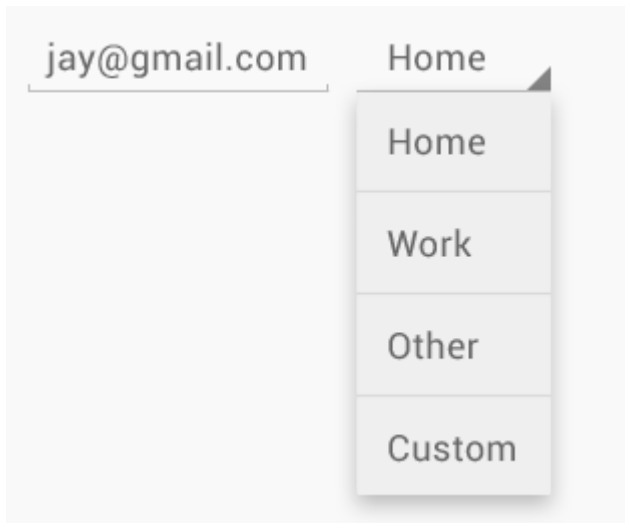
- ◉ View objekt
- ◉ možnost vytvořit vlastní View
  
- ◉ **TextView** - nápis (label)
- ◉ **ImageView** - obrázek
- ◉ **Button** - tlačítko
- ◉ **EditText** - pole pro uživatelský vstup (editbox)
- ◉ **CheckBox**, **RadioButton**
- ◉ **AnalogClock**
- ◉ další: výběr data, zoom control aj.



# VYBRANÉ INPUT CONTROLS

třída	popis
Button	Tlačítko, lze stisknout
EditText	Editovatelné textové pole
AutoCompleteTextView	Viz předchozí včetně návrhů
CheckBox	On/off switch, pro množinu neexkluzivních voleb
RadioButton, RadioGroup	Přepínač, exkluzivní volba
ToggleButton	On/Off tlačítko
Spinner	Výběr jedné hodnoty z drop-down nabídky
DatePicker, TimePicker	Výběr data, času

# PŘÍKLAD - SPINNER



Volby, z kterých volíme - [SpinnerAdapter](#) - čerpá z [ArrayAdapter](#) (hodnoty v poli nebo XML string-array) nebo [CursorAdapter](#) (z databáze)

Odkaz: <http://developer.android.com/guide/topics/ui/controls/spinner.html>

# PŘÍKLAD - AUTOCOMplete



Ukazuje návrhy zatímco uživatel píše

Seznam návrhů lze zavřít tlačítkem back

Lze nadefinovat množství znaků, po kterém se uživateli objeví návrhy

Odkaz:

[http://www.tutorialspoint.com/android/android\\_autocomplete\\_textview\\_control.htm](http://www.tutorialspoint.com/android/android_autocomplete_textview_control.htm)

# WIDGET - PŘÍKLAD IMAGEVIEW

<ImageView

```
    android:id="@+id/imageView1"  
    android:layout_height="wrap_content"  
    android:layout_width="wrap_content"  
    android:src="@drawable/and"  
    android:onClick="click_obrazek">  
</ImageView>
```

do *projektu res/drawable-mdpi* přidáme  
obrázek (**and.jpg**)

lze mít různé velikosti obrázků dle rozlišení

# ANDROID:ID

- ⦿ unikátní identifikace
- ⦿ `android:id="@+id/imageView1"`
  - @ říká parseru: zbytek bude id resource
  - +: nový resource, vytvořit a přidat do R.java
  - v programu `R.id.imageView1`
- ⦿ zdroje už nabízené Android frameworkem:
  - `android:id="@android:id/empty"`
  - není +
  - přidáný jmenný prostor

# VLOŽENÍ SOUBORU S OBRÁZKEM

- ◉ více adresářů **drawable**:
  - /res/drawable-hdpi ... high density screens
  - /res/drawable-mdpi ... medium density screens
  - /res/drawable-ldpi ... low density screens
  - /res/drawable-xhdpi, xxhdpi
- ◉ pokud nemáme více verzí:
  - /res/drawable
- ◉ přidáme soubor s obrázkem (.png, .jpg,...)
  - např. **and.jpg** (např. Ctrl+c, Ctrl+v)
  - automaticky se updatuje generovaná třída R
- ◉ přístup z Javy: **R.drawable.and**

# JAK VYTVOŘÍME RŮZNÉ RESOURCE ADRESÁŘE V ANDROID STUDIO?

- Kdo si má pamatovat zkratky?
- Kliknu na **drawable** - pravá myš -  
**New drawable resource file**
- Modifikátor **density** - a zvolit vhodnou hodnotu

# Drawable - new - Drawable resource - modifikátor density

The screenshot shows an IDE interface with the following components:

- Project Structure:** A tree view on the left shows the project structure. The `res` folder is expanded, and the `drawable` sub-folder is highlighted with a red arrow. Other folders include `manifests`, `java`, `layout`, `menu`, `mipmap`, `values`, and `Gradle Scripts`.
- Main Activity Code:** The main editor shows the `MainActivity.java` file with the following code:

```
public void tlacitko_dialog (View v) {  
  
    final Context c1 = v.getContext();  
  
    AlertDialog a1 = new AlertDialog.Builder(thi  
a1.setTitle("Nápověda");  
    a1.setMessage("Pomoc si sám!");  
}
```
- New Resource File Dialog:** A dialog box titled "New Resource File" is open. It has the following fields:
  - File name:** (empty)
  - Source set:** `main`
  - Directory name:** `drawable-xhdpi` (circled in red)
- Qualifiers:** The dialog shows a list of "Available qualifiers" on the left and "Chosen qualifiers" on the right. The "Chosen qualifiers" list contains `X-High Density` (highlighted in green). The "Density" list on the right contains the following options: `X-High Density`, `XXX-High Density`, `XX-High Density`, `X-High Density` (highlighted in blue), `High Density`, `Medium Density`, `Low Density`, `Any Density`, and `No Density`. A red circle highlights the "Density" list.



# POUŽITÍ V PROGRAMU

## Přiřazení UI aktivitě:

```
setContentView(R.layout.main);
```

- ⦿ v metodě onCreate
- ⦿ použije soubor `main.xml`
- ⦿ z adresáře `res/layout`  
nebo dle modifikátorů např. `res/layout-land`

## Získání reference na View:

```
TextView tv = (TextView) findViewById(R.id.napis);
```

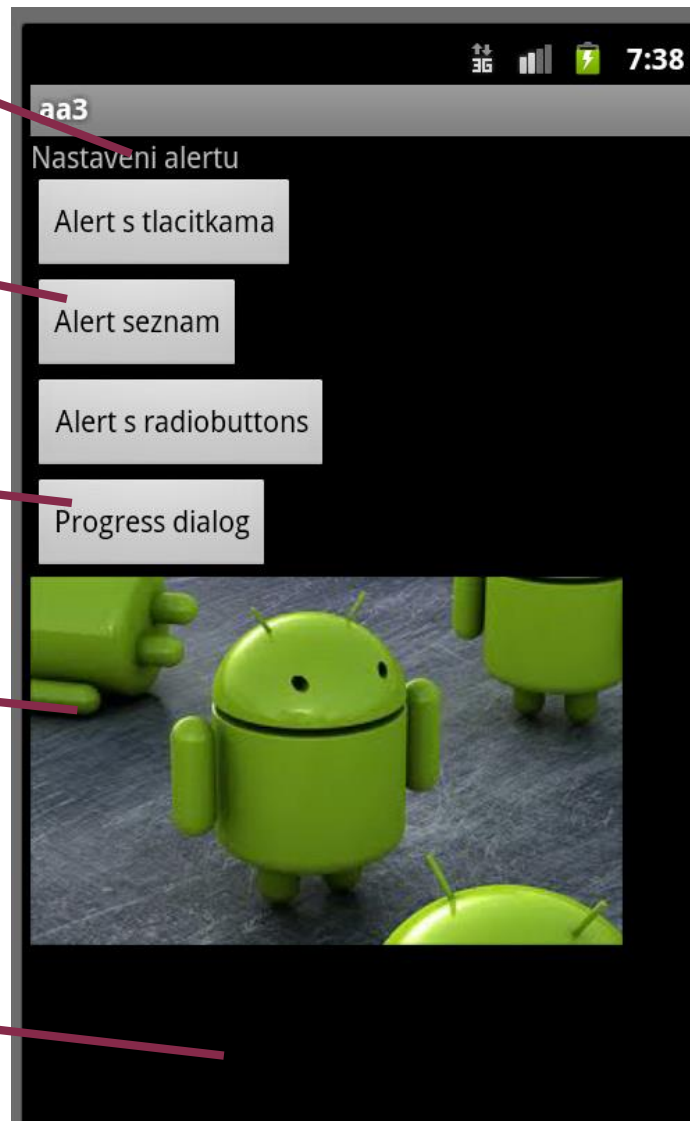
TextView

Button

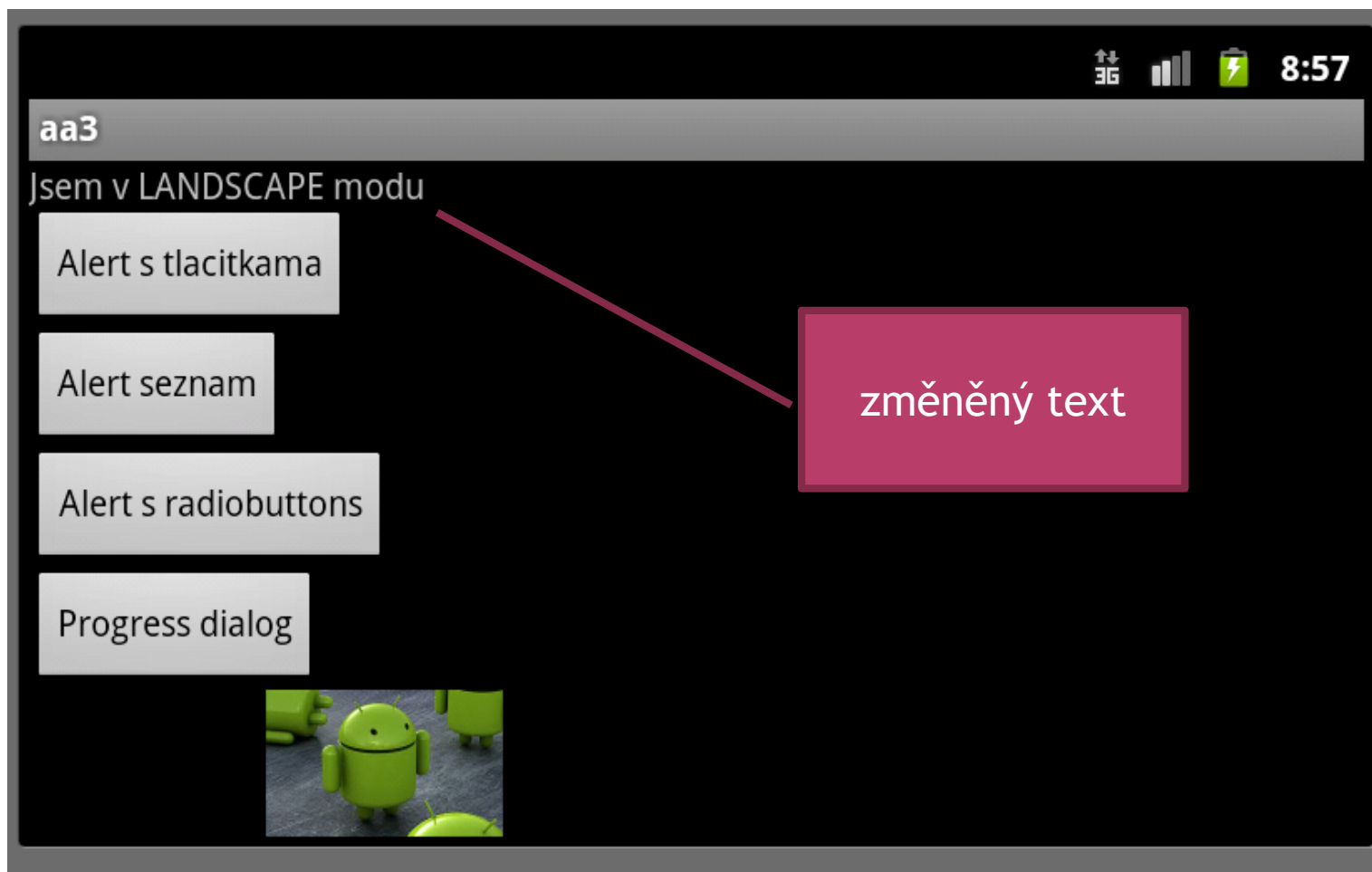
dlouhý stisk  
tlačítka vyvolá  
kontextové menu

ImageView

Tlačítko Menu  
zobrazí zde menu  
na Android 2.x,  
jinak ActionBar  
3.x,4.x nahoře



# ZOBRAZENÍ V LANDSCAPE MÓDU



# ALTERNATIVNÍ ZDROJE

- více verzí
  - obrázků
    - dle možností obrazovky (density)
  - layoutů
    - otočení zařízení do landscape módu
    - nejenom „hezčí“ zobrazení, ale i přidaná funkcionality
  - řetězců (strings.xml)
    - vícejazyčné aplikace
- cíl - v dané situaci využít nejvhodnější zdroje

# ALTERNATIVNÍ ZDROJE - KVALIFIKÁTORY

## layout

- ◉ standardní adresář `/res/layout/neco.xml`
- ◉ lze vytvořit `/res/layout-land/neco.xml`

## řetězce

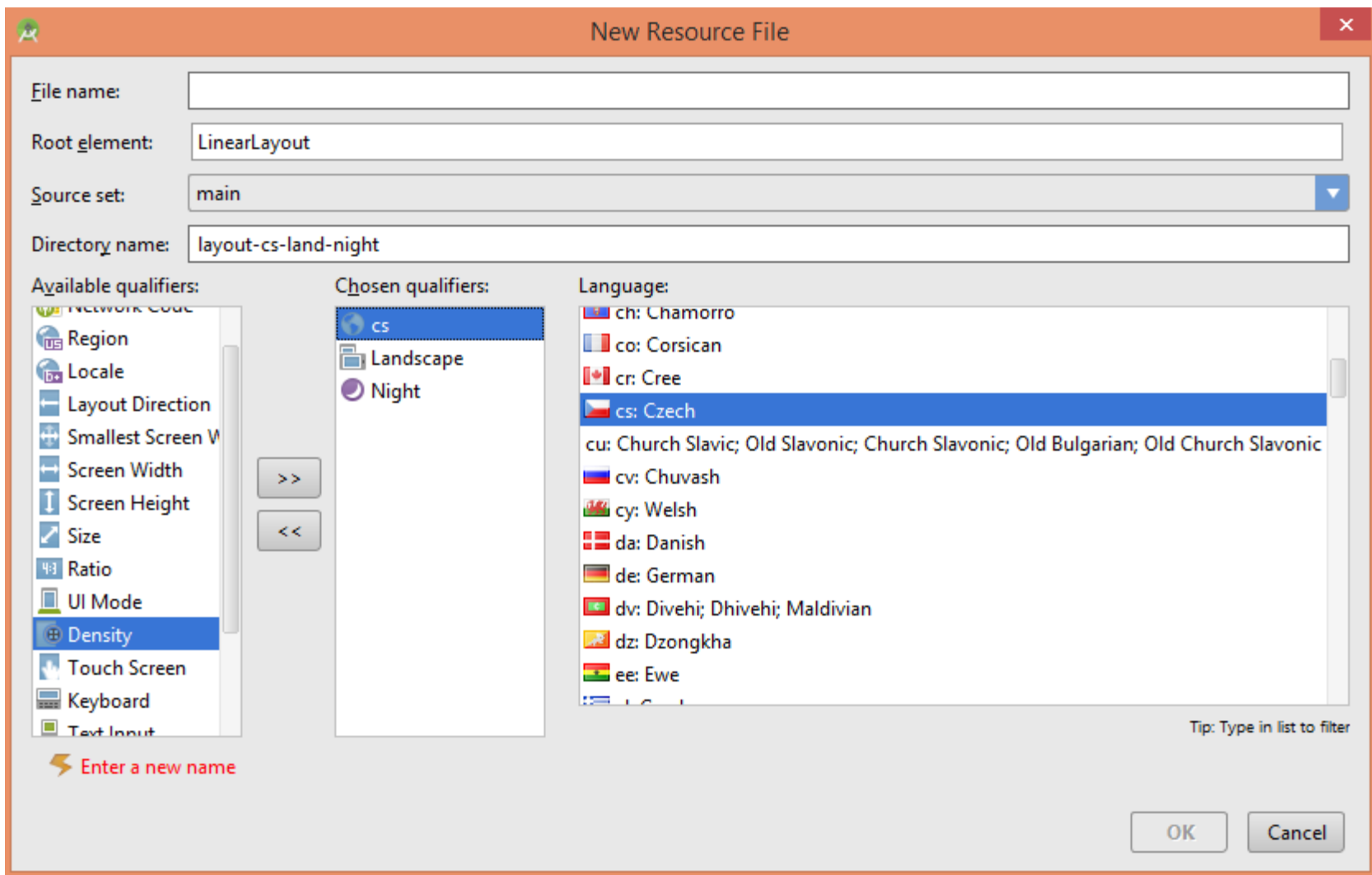
- ◉ standardní `/res/values/strings.xml`
- ◉ např. `/res/values-fr-rCA/strings.xml`
  - `-fr` .. jazyk (francouzština)
  - `-rCA` .. region Kanada

nenajde-li vhodný zdroj, využije defaultní

# ALTERNATIVNÍ ZDROJE

- ◉ při tvorbě pomůže wizard  
PraváMyš -> New -> **Layout Resource File**
- ◉ **konfigurace** (country code, language, orientation, keyboard)
- ◉ **upřesnění** hodnoty

např.: layout -> orientation -> landscape



seznam modifikátorů:

<http://developer.android.com/guide/topics/resources/providing-resources.html#AlternativeResources>

# POZNÁMKY

Přístup k řetězcům:

```
Resources res = getResources();
```

```
String
```

```
    text = res.getString(R.string.retezec_age, age);
```

```
tv.setText(text);
```

```
ed1.setText("");
```

```
ed1.requestFocus();
```

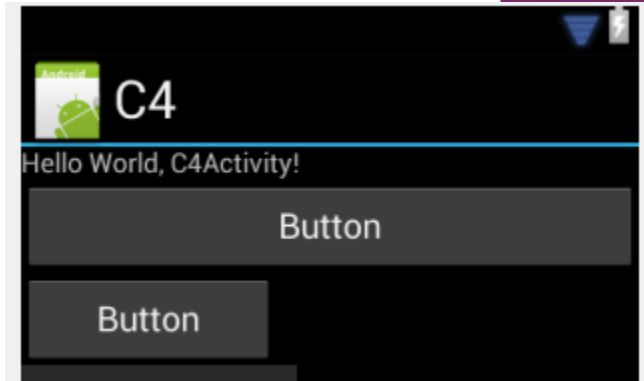


# VLASTNOSTI VIEW (WIDGETU)

- **layout\_height, layout\_width**
  - kolik prostoru žádá od rodiče pro své zobrazení
  - **match\_parent** - všechen dostupný prostor od rodiče (dříve **fill\_parent** do API 8)
  - **wrap\_content** - jen tolik, aby zobrazil co chce
  - exaktní hodnota - nedoporučuje se
- **layout\_weight**
- **layout\_gravity** - centrování prvku
- **gravity** - centrování uvnitř prvku (nápis v tlačítku)
- **text** - text nebo *@strings/nazev*
- **id** - ve tvaru *@+id/jmeno*

# UKÁZKA XML

1. `<Button android:id="@+id/button1"`
2. `android:layout_width="match_parent"`
3. `android:layout_height="wrap_content"`
4. `android:text="Button" />`



5. `<Button`
6. `android:id="@+id/button2"`
7. `android:layout_width="132dp"`
8. `android:layout_height="wrap_content"`
9. `android:onClick="volejDomu"`
10. `android:text="Button" />`

# JEDNOTKY, PŘENOSITELNOST

- ◉ velikost displeje (fyzická)
- ◉ rozlišení
- ◉ pixel density  
(100dpi - 1 palec displeje 100bodů)

dp .. density independent pixel

výhodné použít pro View

$px = dp * (dpi / 160)$

240 dpi screen, 1 dp equals 1.5 physical pixels

Používat **dp** pro všechno, **sp** pro velikosti textu. (!!)

Následující tabulky viz:

<http://stackoverflow.com/questions/2025282/difference-between-px-dp-dip-and-sp-on-android>

# JEDNOTKY - POJMY

jednotky	popis
px	Odpovídá aktuálním pixelům na obrazovce
in	Podle fyzické velikosti obrazovky, 1 inch = 2.54cm
mm	Na základě fyzické velikosti obrazovky, milimetry
pt	Points - 1/72 of an inch based on the physical size of the screen.
dp or dip	Density-independent Pixels an abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi screen, so <b>one dp is one pixel on a 160 dpi screen</b> . The <b>ratio</b> of dp-to-pixel will <b>change with the screen density</b> , but not necessarily in direct proportion.
sp	Scale-independent Pixels - this is like the dp unit, but it is also scaled by the user's <b>font size</b> preference. It is recommend you use this unit when specifying font sizes, so they will be adjusted for both the screen density and user's preference.

# JEDNOTKY - SROVNÁNÍ

Unit	Description	Units Per Physical Inch	Density Independent	Same Physical Size On Every Screen
px	Pixels	Varies	No	No
in	Inches	1	Yes	Yes
mm	Millimeters	25.4	Yes	Yes
pt	Points	72	Yes	Yes
dp	Density Independent Pixels	~160	Yes	No
sp	Scale Independent Pixels	~160	Yes	No

# CO JE CÍLEM?



Figure 2. Example application without support for different densities, as shown on low, medium, and high-density screens.

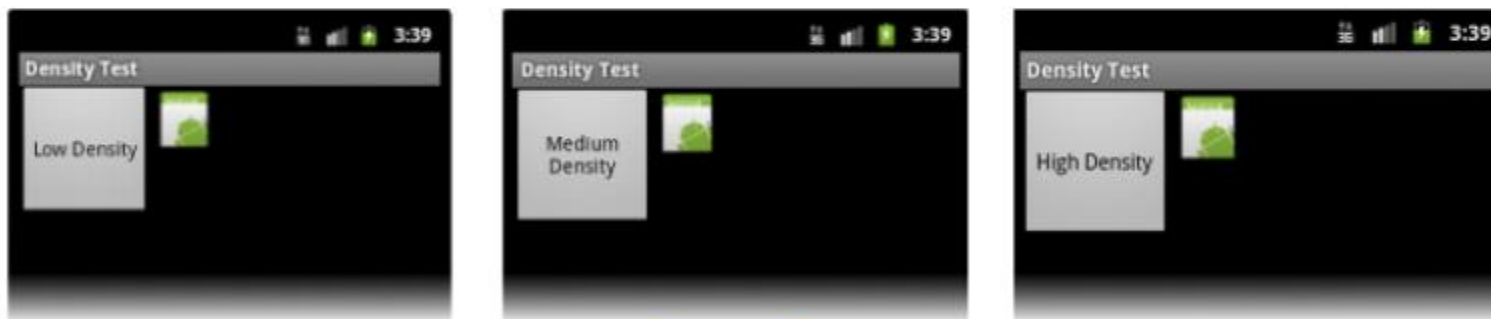
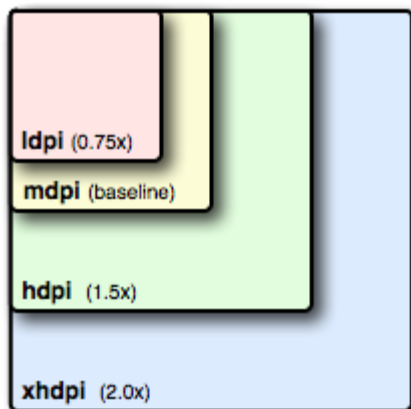


Figure 3. Example application with good support for different densities (it's density independent), as shown on low, medium, and high density screens.

Důležitý odkaz:

[http://developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html)

# VELIKOSTI



Odkaz:

[http://developer.android.com/guide/practices/screens\\_support.html](http://developer.android.com/guide/practices/screens_support.html)

Poměr velikostí:

3:4:6:8:12:16 scaling ratio

36x36 (0.75x) for low-density

48x48 (1.0x baseline) for medium-density

72x72 (1.5x) for high-density

96x96 (2.0x) for extra-high-density

180x180 (3.0x) for extra-extra-high-density

192x192 (4.0x) for extra-extra-extra-high-density

# KVALIFIKÁTORY PRO VELIKOST OBRAZOVKY

res/layout/main\_activity.xml

# For handsets (smaller than 600dp available width)

res/layout-sw600dp/main\_activity.xml

# For 7" tablets (600dp wide and bigger)

res/layout-sw720dp/main\_activity.xml

# For 10" tablets (720dp wide and bigger)

Modifikátor nejmenší dostupná šířka (-swCislo)

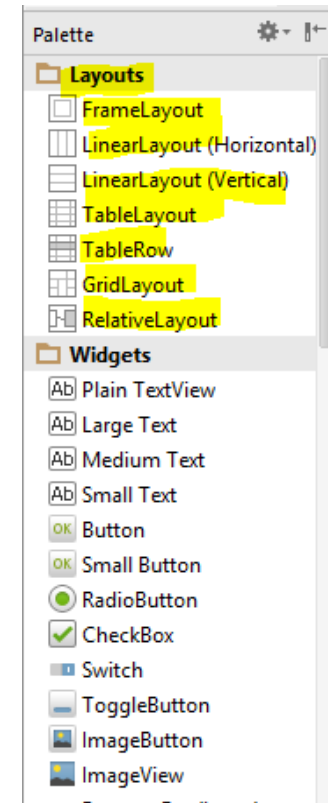
- kolik bude pro náš layout dostupné, bez ohledu na orientaci zařízení



# LAYOUT

odpovědná za alokaci prostoru pro své potomky  
hlavní layouty:

- ◉ **Linear** Layout
- ◉ **Table** Layout
- ◉ **Frame** Layout
- ◉ **Relative** Layout
- ◉ **(Absolute** Layout)
- ◉ **Grid** Layout (novější)



Ukázky layoutů a widgetů:

<http://developer.android.com/resources/tutorials/views/index.html>

# LINEAR LAYOUT

- ◉ často používaný
- ◉ skládá objekty za sebe, **horizontálně** nebo **vertikálně**, záleží na pořadí objektů
- ◉ ptá se, kolik prostoru bude child objekt potřebovat - dřívější child může spotřebovat téměř vše...

android:orientation="vertical"

android:orientation="horizontal"

```
<LinearLayout ... >  
  <TextView ... />  
  <Button ... />  
</LinearLayout >
```

# XML - LINEAR LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
```

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

```
<TextView
```

```
  android:layout_width="fill_parent"  
  android:layout_height="wrap_content"  
  android:text="@string/hello" />
```

```
<Button
```

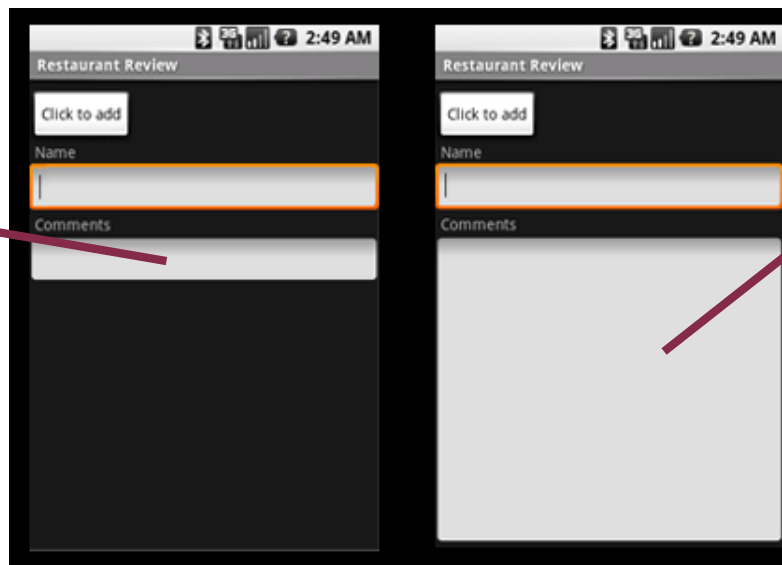
```
  android:id="@+id/button1"  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content"  
  android:layout_gravity="center"  
  android:layout_weight="0.5"  
  android:text="Button" />
```

```
</LinearLayout>
```

# LINEAR LAYOUT - POKRAČOVÁNÍ

- **layout\_gravity** ( = zarovnání )
  - left, right, center\_horizontal
- **layout\_weight** ( = váha, důležitost )
  - integer, defaultní hodnota 0
  - důležitější dostane „větší prostor“ k vykreslení

weight  
všude 0



nastavená weight 1,  
všichni ostatní 0

# WEIGHT - POČÍTÁNÍ

přidělený prostor dítěti = (weight dítěte) /  
(suma weight všech dětí v Linear Layout)

př. 3 TextBoxy, z toho 2 weight 2, a jeden 1

1. text box =  $2 / (2+2+1)$

2. text box =  $2 / (2+2+1)$






3. text box =  $1 / (2+2+1)$

viz <http://stackoverflow.com/questions/3995825/what-does-androidlayout-weight-mean>

# TABLE LAYOUT

- ◉ umístění objektů do tabulky
- ◉ **TableRow** - reprezentuje řádku, obsahuje další widgety, ty jsou horizontálně vedle sebe
- ◉ **android:stretchColumns="1"**
  - lze použít \*
  - roztažení dle dostupného prostoru

```
<TableLayout ... android:stretchColumns="1">  
  <TableRow> ... </TableRow>  
</TableLayout>
```

	Feb 7	Feb 8	Feb 9	Feb 10	Feb 11
Day High	28°F	26°F	23°F	17°F	19°F
Day Low	15°F	14°F	3°F	5°F	6°F
Conditions					

# FRAME LAYOUT

- ◉ umístí potomka nad předchozího (nejnověji umístěný zakrývá předchozí...)
- ◉ prázdný prostor na obrazovce, který vyplníme jedním objektem
- ◉ „přišpendlen“ k levému hornímu rohu obrazovky, další child kresleny přes předchozí (můžeme je umístit např. pomocí `android:layout_gravity="bottom|left"`)
- ◉ jako placeholder pro další přidané programově

# FRAME LAYOUT PŘÍKLAD

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

```
<ImageView  
    android:id="@+id/imageView1"  
    android:layout_width="393dp"  
    android:layout_height="match_parent"  
    android:src="@drawable/and" />
```

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:orientation="vertical" >
```

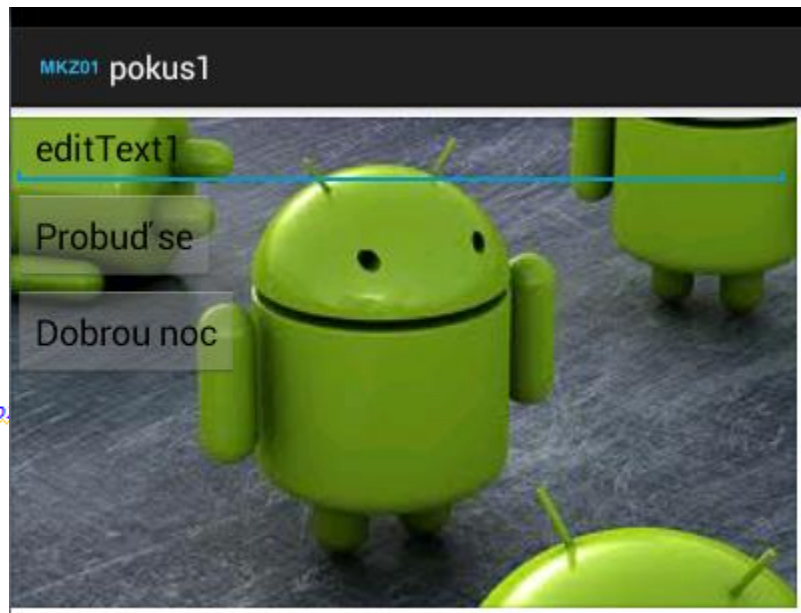
```
<EditText  
    android:id="@+id/editText1"  
    android:layout_width="390dp"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:text="Editační pole na obrázkovém po
```

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Probud' se" />
```

```
<Button  
    android:id="@+id/button2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Dobrou noc" />
```

```
</LinearLayout>
```

```
</FrameLayout>
```





# RELATIVE LAYOUT

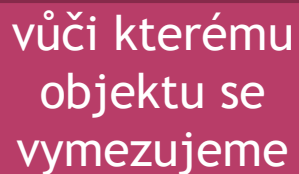
- pozicovat potomka relativně k ostatním nebo ke kontejneru
- potomci jsou renderovány v uvedeném pořadí

`android:layout_below="@id/label"`

`android:layout_toLeftOf="@id/ok"`

`android:layout_alignTop="@id/ok"`

vůči kterému  
objektu se  
vymezujeme



# RELATIVE LAYOUT

## - VŮČI KONTEJNERU

`android:layout_alignParentBottom`

spodek elementu na spodek kontejneru

`android:layout_alignParentLeft`

levá strana na levý okraj kontejneru

`android:layout_alignParentRight`

`android:layout_alignParentTop`

`android:layout_centerHorizontal`

`android:layout_centerVertical`

`android:layout_centerInParent`

centruje obojí

# RELATIVE LAYOUT

## - VŮČI OSTATNÍM PRVKŮM

⦿ použijeme id elementu

`android:layout_above`

`android:layout_below`

`android:layout_toLeftOf`

`android:layout_toRightOf`

`android:layout_alignBaseline`

`android:layout_alignBottom`

`android:layout_alignLeft`

zarovná levou hranu s daným elementem

# ABSOLUTE LAYOUT

- ◉ umístění na absolutní souřadnice na obrazovce
- ◉ není flexibilní  
=> problém přenositelnost na různá zařízení

# ANDROID LAYOUT TUTORIAL

<http://www.learn-android.com/2010/01/05/android-layout-tutorial/>

ukázky XML definic včetně výsledného zobrazení vrstvy

# NÁSTROJ: ANDROID DEVICE MONITOR

pro analýzu UI je k dispozici nástroj  
**HierarchyViewer**, který je součástí SDK

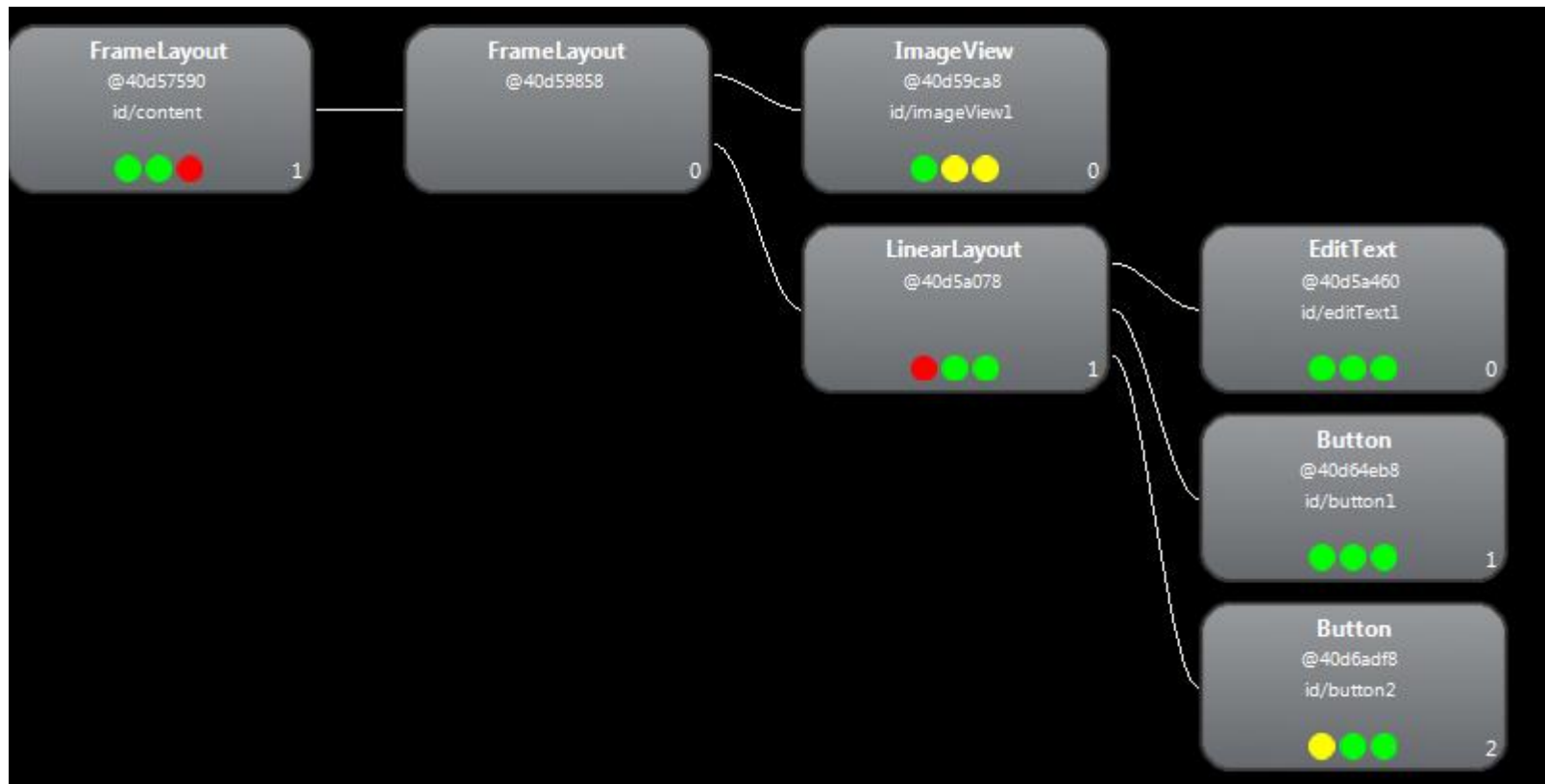
použití:

- ◉ běží nám aplikace v emulátoru
- ◉ z příkazové řádky (cmd.exe pod Win)  
**c:\sw\android\sdk\tools\monitor.bat**

<http://code.tutsplus.com/tutorials/android-tools-using-the-hierarchy-viewer--mobile-6997>

(podívat se)

# HIERARCHYVIEWER - UKÁZKA



ukázka části z frame-layout příkladu

# BARVY, POZADÍ

- ◉ `android:background="@drawable/background"`
- ◉ `android:textColor="@color/barva1"`
  - viz další slide
- ◉ `android:background="#cfff"`
- ◉ standardní RGB + Alpha kanál (transparentnost)
- ◉ každá složka 0..255, #0..#FF
- ◉ AARRGGBB
- ◉ #4B8C .. #44BB88CC
  
- ◉ obrázek pozadí, barva textu, barva pozadí textboxu (TextEdit)



# BARVY


- v popisu layoutu:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_centerVertical="true"
    android:text="@string/hello_world"
    android:textColor="@color/barva1" />
```

- v souboru /res/values/color.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="barva1">#FF0000</color>
</resources>
```

# TOAST



Jsem toast :-)

- nebere aktivitě focus
- zprávy, kterým není potřeba věnovat velká pozornost (oznámení např. *download skončil*)
- nepoužívat na důležité zprávy
- časem sám zmizí

```
Toast.makeText (this, "Privezli jidlo!",  
    Toast.LENGTH_LONG).show();
```

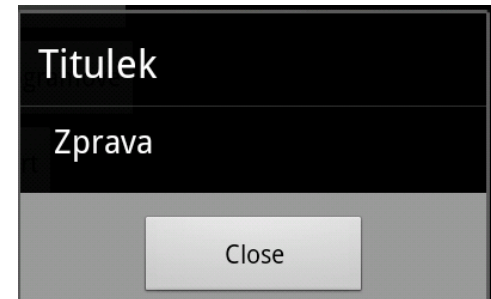
```
Toast.makeText(this, R.string.hlaseni_jidlo,  
    Toast.LENGTH_LONG).show();
```

# ALERT DIALOG

- ⦿ `setPositiveButton` -- yes, save
- ⦿ `setNegativeButton` -- close
- ⦿ `setNeutralButton` -- no

```
new AlertDialog.Builder(this).setTitle(„Titulek”).  
setMessage(„Zprava”).setNeutralButton("Close",  
null).show();
```

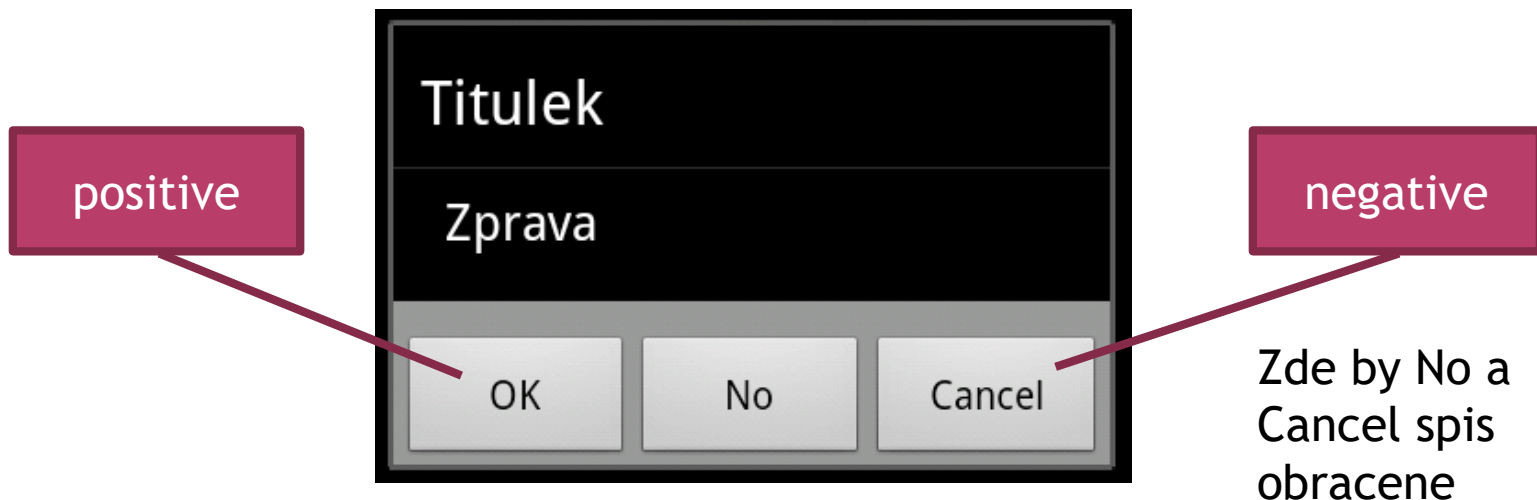
```
new AlertDialog.Builder(this).setTitle(„Titulek”).  
setMessage(„Zprava”).setIcon(R.drawable.icon).  
setNeutralButton("Close", null).show();
```



# ALERT DIALOG SE 3 TLAČÍTKY

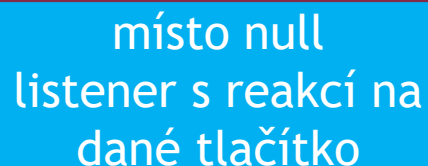
## ○ new

```
AlertDialog.Builder(this).setTitle(„Titulek“).  
setMessage(„Zprava“).  
setNegativeButton("Cancel", null).  
setPositiveButton("OK", null).  
setNeutralButton("No", null).show();
```



# ALERT DIALOG, OBSLUHA

```
1. new AlertDialog.Builder(this)
2. .setTitle("Titulek")
3. .setMessage("Zprava")
4. .setNegativeButton("Cancel",new DialogInterface.OnClickListener() {
5.     public void onClick(DialogInterface dialog, int id) {
6.         finish(); // ukončení aktivity
7.     }
8. }
9. )
10. .setNeutralButton("No", null)
11. .setPositiveButton("OK",null).show();
```



místo null  
listener s reakcí na  
dané tlačítko

# ALERT DIALOG - SETBUTTON

```
AlertDialog a1 = new AlertDialog.Builder(this).create();
a1.setTitle("Od cisnika");
a1.setMessage("Zaplat a jdi.");
a1.setButton(RESULT_OK, "Jdu", new
DialogInterface.OnClickListener() {
```

@Override

```
public void onClick(DialogInterface dialog, int which) {
    Toast.makeText(c1, „Cau", Toast.LENGTH_LONG).show();
}
});
a1.setIcon(R.drawable.ic_launcher);
a1.show();
```

# ALERT - SEZNAM (SETITEMS)

```
1. final CharSequence[] items = {"Jaro", "Leto", "Podzim", "Zima"};
2. AlertDialog.Builder builder = new AlertDialog.Builder(this);
3. builder.setTitle("Vyber oblíbenou část roku:");
4. builder.setItems(items, new DialogInterface.OnClickListener() {
5.     public void onClick(DialogInterface dialog, int item) {
6.         Toast.makeText(getApplicationContext(), items[item],
7.         Toast.LENGTH_SHORT).show();
8.     }
9. });
10. AlertDialog alert = builder.create();
    alert.show();
```

využijeme index  
vrácené položky

Vyber oblíbenou část roku:

Jaro

Leto

Podzim

Zima

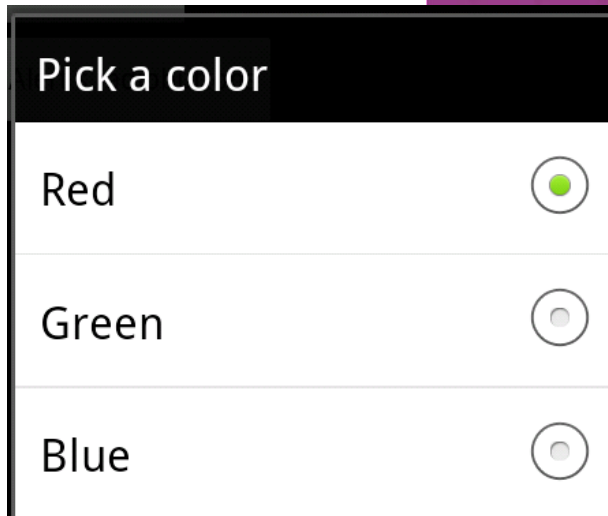
# ALERT - RADIO BUTTONS

```
final CharSequence[] items = {"Red", "Green", "Blue"};
```

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);  
builder.setTitle("Pick a color");
```

```
builder.setSingleChoiceItems(items, -1, new  
DialogInterface.OnClickListener() {  
    public void onClick(DialogInterface dialog, int item) {  
        Toast.makeText(getApplicationContext(), items[item],  
Toast.LENGTH_SHORT).show();  
    }  
});  
AlertDialog alert = builder.create();
```

na začátku žádný  
vybraný





# ZPRACOVÁNÍ UI UDÁLOSTÍ

- zachytit událost z daného View objektu
- rozhraní event listeners,
- `View.set...Listener()`
- callbacky:

událost	Popis, příklad
<code>onClick()</code>	Např. stisk tlačítka
<code>onLongClick()</code>	Dlouhý stisk
<code>onFocusChange()</code>	Navigace na/z položky
<code>onKey()</code>	Stisk / uvolnění klávesy
<code>onTouch()</code>	Stisknutí, uvolnění, pohyb po obrazovce
<code>onCreateContextMenu()</code>	Při vytvoření kontextového menu

# REGISTRACE POSLUCHAČE PRO TLAČÍTKO (!)

```
1. // anonymní implementace OnClickListener
2. private OnClickListener mujPosluchac = new OnClickListener() {
3.     public void onClick(View v) {
4.         // zareagujeme na stisk tlačítka
5.     }
6. };

7. protected void onCreate(Bundle savedInstanceState) {
8.     ...
9.     // tlačítko z layoutu
10.    Button button = (Button)findViewById(R.id.button01);
11.    // zaregistrujeme posluchače
12.    button.setOnClickListener(mujPosluchac);
13.    ...
14. }
```

# LISTENER V RÁMCI AKTIVITY

```
1. public class ExampleActivity extends Activity
    implements OnClickListener {
2.     protected void onCreate(Bundle savedInstanceState) {
3.         ...
4.         Button button = (Button)findViewById(R.id.button01);
5.         button.setOnClickListener(this);
6.     }
7.     // implementace callbacku
8.     public void onClick(View v) {
9.         // zareagujeme na stisk tlačítka
10.    }
11. }
```

# NÁVRATOVÁ HODNOTA NĚKTERÝCH LISTENERŮ

- ◉ *true* - obsloužili jsme událost
- ◉ *false* - událost pokračuje do dalšího zpracování

příklady listenerů:

- `onLongClick()`
- `onKey()`
- `onTouch()`

# VYTVÁŘENÍ MENU

- ◉ options menu
  - když uživatel stiskne tlačítko Menu (ActionBar)
- ◉ context menu
  - dlouhý stisk daného objektu
- ◉ submenu
  - menu obsahuje další vnořené položky

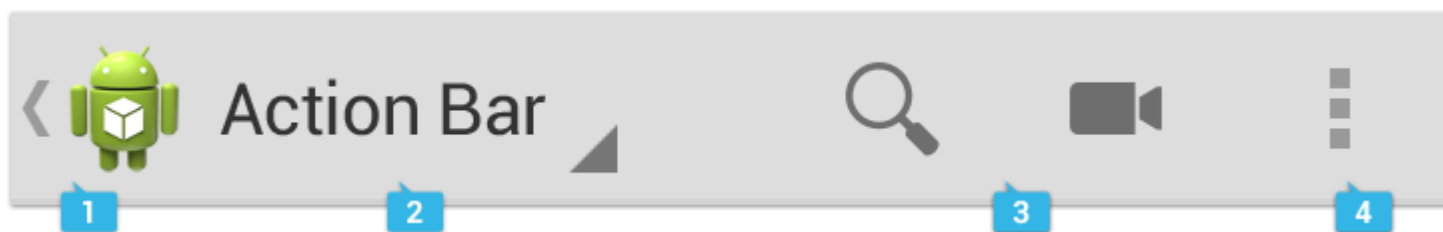
# ACTION BAR

- ⊙ Konzistentní navigace napříč aplikací
- ⊙ Priorita důležitých akcí
- ⊙ Adaptace na konfiguraci obrazovky (port-land)
- ⊙ Přidávání widgetů (search, share)
- ⊙ Navigace mezi obrazovkami
- ⊙ <http://developer.android.com/design/patterns/actionbar.html>

hezký tutoriál:

<http://www.vogella.com/tutorials/AndroidActionBar/article.html>

# ACTION BAR



pole	název	poznámka
1	Ikona aplikace	Využití pro navigaci
2	View control	
3	Action buttons	Nejdůležitější akce; kdo se sem nevejde, jde do overflow
4	Action overflow	Pro méně časté akce

# ACTION BAR

na vrchu aktivity

- titulek aktivity
- ikona
- akce, které lze vyvolat
- další interaktivní položky
- navigace v rámci aplikace



povolen pro API  $\geq 11$  (od And3.0), lze jej zakázat  
pro API  $< 11$  .. options menu

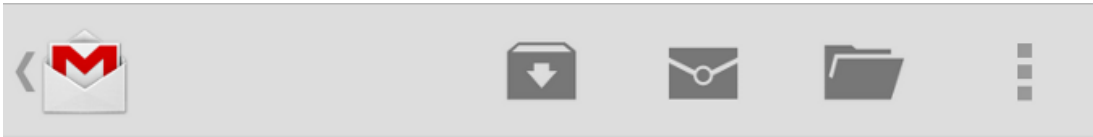
lze využít Support Library (API 7 a výše)

také se používá <http://actionbarsherlock.com/>

action bar je viditelný, zatímco menu v 2.x je až na  
požádání (stisknutí tlačítka Menu)



# ACTION BAR



3 akční tlačítka a overflow button (další položky)

aby se položka menu (ne)objevila v action baru, v XML definici menu u položky:

`showAsAction="ifRoom"`

`showAsAction= "always"`

`showAsAction= "never"`

# ACTION BAR

```
ActionBar actionBar = getActionBar();
```

```
// skrýt
```

```
actionBar.hide();
```

```
// zobrazit
```

```
actionBar.show();
```

```
// změna ikony
```

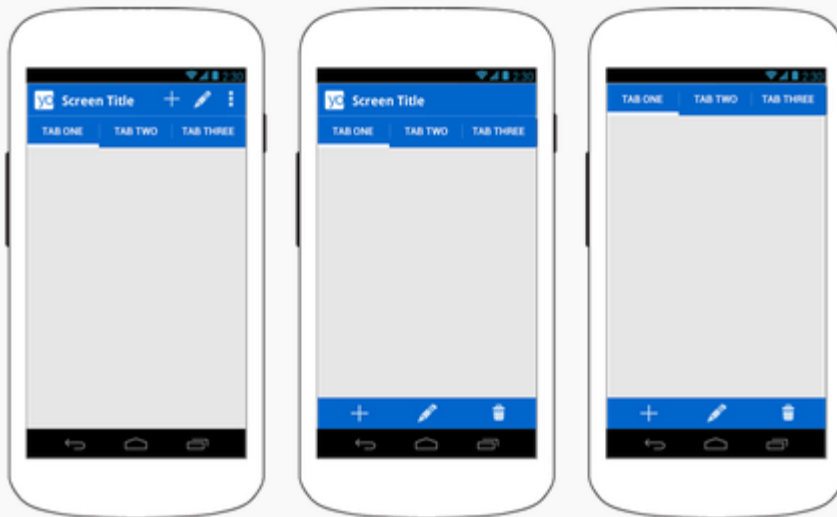
```
actionBar.setIcon(R.drawable.ico_myico);
```

# SPLIT ACTION BAR

Oddělená část ve spodní části obrazovky

```
<manifest ...>
  <activity uiOptions="splitActionBarWhenNarrow" ... >
    <meta-data android:name="android.support.UI_OPTIONS"
              android:value="splitActionBarWhenNarrow" />
  </activity>
</manifest>
```

uiOptions .. Od API14  
Metadata .. Pro starší



**Figure 3.** Mock-ups showing an action bar with tabs (left), then with split action bar (middle); and with the app icon and title disabled (right).

# NAVIGATION DRAWER

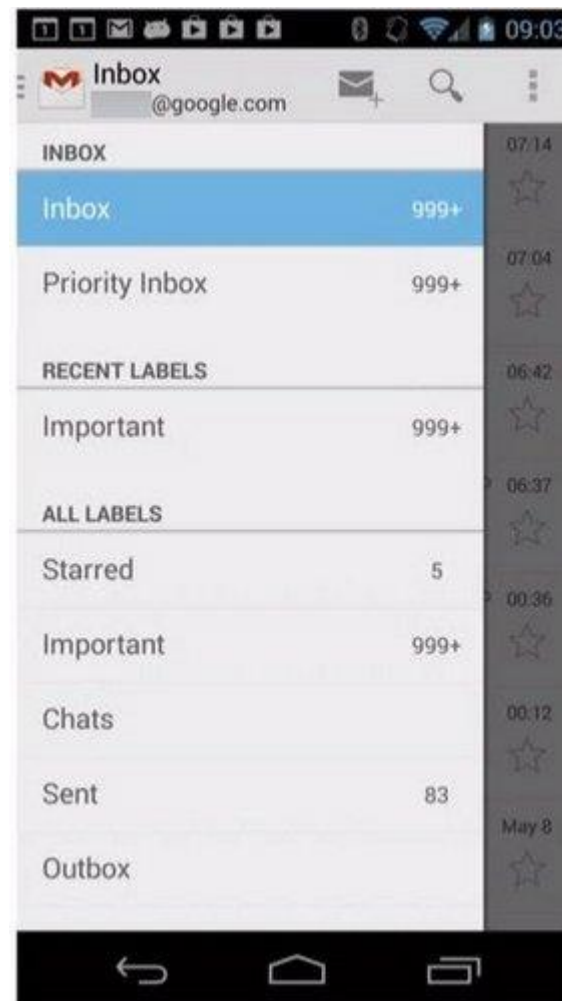
panel zobrazující možnosti navigace  
na levé straně obrazovky

defaultně skrytý  
lze jej potáhnout z levé strany  
nebo kliknutím na ikonu aplikace

může obsahovat:

- titulky
- ikony
- čítače

obr. viz deVogella tutorial



# JAK NA MENU

1. tvorba v XML menu resource
2. nahrát menu jako objekt

## XML:

- <menu> root element
- <item> jedna položka menu
- <group> možnost grupovat položky

## atributy:

- android:id            unikátní identifikátor
- android:icon        ikona položky menu
- android:title        zobrazený text

# MENU - XML

```
<?xml version="1.0" encoding="utf-8"?>
  <menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
      android:icon="@drawable/ic_new_game"
      android:title="@string/new_game" />
    <item android:id="@+id/help"
      android:icon="@drawable/ic_help"
      android:title="@string/help" />
  </menu>
```

menu s dvěmi položkami: nová hra a nápověda

# POZNÁMKY, ANDROID 2.X ZOBRAZENÍ MENU

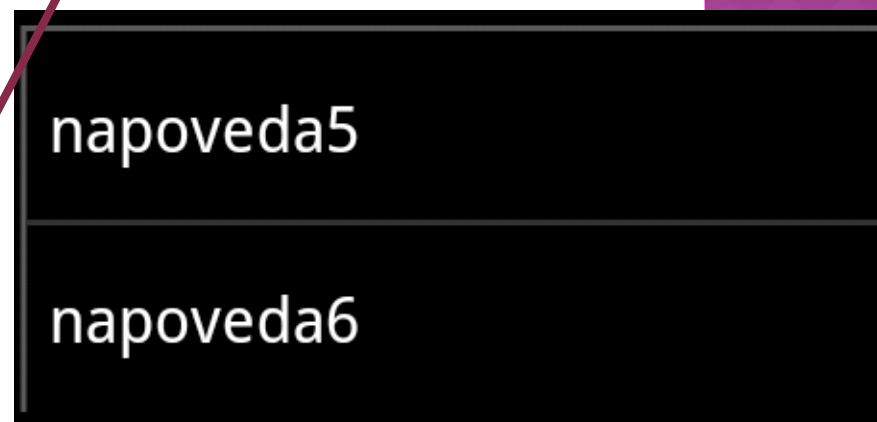
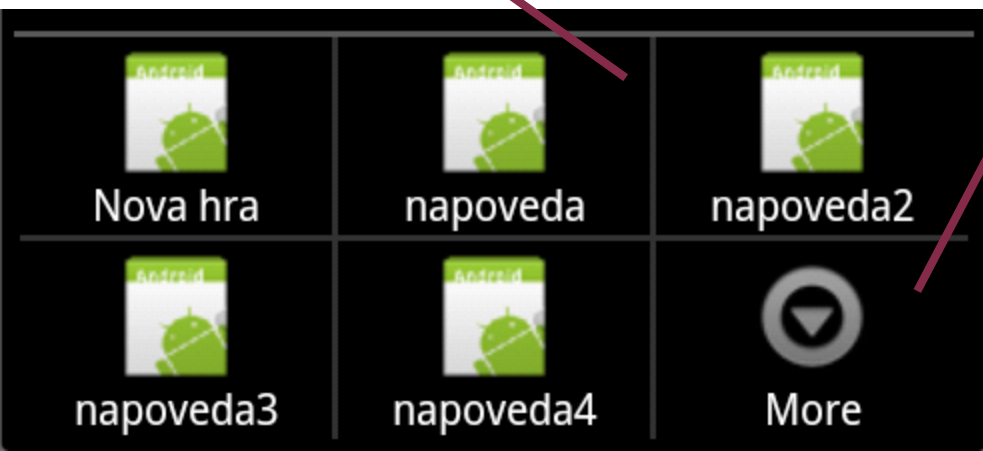
**inflate** menu resource

=

konvertovat menu resource do  
programovatelného objektu

options menu se 7  
položkami

další položky se zobrazí  
po stisknutí More



# INFLATING MENU

máme XML soubor `hra_menu.xml`

zvoláno při prvním vytvoření menu

```
1. public boolean onCreateOptionsMenu(Menu menu) {  
2.     MenuInflater inflater = getMenuInflater();  
3.     inflater.inflate(R.menu.hra_menu, menu);  
4.     return true;  
5. }
```

naplníme menu z XML souboru

Ve starších androidech Menu, v novějších Action bar

menu lze plnit i programově `add()`



# MENU - REAKCE NA AKCI UŽIVATELE

- ⊙ reakce na výběr:
  - systém zavolá *onOptionsItemSelected()*
- ⊙ co uživatel vybral:
  - poznáme dle *getItemId()*
- ⊙ Vytvoření ActionBaru (menu):
  - volá *onCreateOptionsMenu()*
  - *onPrepareOptionsMenu()* - menu dropdown

# MENU - REAKCE NA UŽIVATELE

```
1. public boolean onOptionsItemSelected(MenuItem item) {  
2.     // určíme vybranou položku  
3.     switch (item.getItemId()) {  
4.         case R.id.new_game:  
5.             newGame();  
6.             return true;  
7.         case R.id.help:  
8.             showHelp();  
9.             return true;  
10.        default:  
11.            return super.onOptionsItemSelected(item);  
12.        }  
13.    }
```

true = reakci na menu  
jsme obsloužili

předáme nadřazené  
třídě místo rovnou  
vrátit false

# CONTEXT MENU - REGISTRACE

registrace View pro **ContextMenu**, např. v metodě *onCreate()*:

```
Button button4 = (Button)findViewById(R.id.button4);  
registerForContextMenu(button4);
```

button4

kontextové  
menu

kontextové menu  
nepodporuje ikony ani  
shortcuty

Progress dialog

cont1

cont2

# CONTEXT MENU - VYTVOŘENÍ

## ⦿ vytvoření kontextového menu:

```
public void onCreateContextMenu( ContextMenu menu,  
    View v, ContextMenuInfo menuInfo) {
```

```
    super.onCreateContextMenu(menu, v, menuInfo);  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.context_menu, menu);
```

```
}
```

# CONTEXT MENU - OBSLUHA

```
1. public boolean onContextItemSelected(MenuItem item) {  
2.     AdapterContextMenuInfo info = (AdapterContextMenuInfo)  
   item.getContextMenuInfo();  
3.     switch (item.getItemId()) {  
4.         case R.id.edit:  
5.             ...  
6.             return true;  
7.         case R.id.delete:  
8.             ...  
9.             return true;  
10.        default:  
11.            return super.onContextItemSelected(item);  
12.        }  
13.    }
```

# SUBMENU

- lze přidat do libovolného menu (kromě submenu)
- položka menu:

```
<item android:id="@+id/help7"  
      android:title="vnorena" >
```

```
<menu>
```

```
<item android:id="@+id/help8"  
      android:title="vnorena8" />
```

```
<item android:id="@+id/open"  
      android:title="vnorena9" />
```

```
</menu>
```

```
</item>
```

jedna z položek  
menu

vnořené  
menu -  
submenu

vnorena

vnorena8

vnorena9

# ZKRATKY (SHORT CUTS)

- ◉ na zařízeních s hw klávesnicí
- ◉ nejdou v kontextovém menu

atributy v <item>:

- ◉ `android:alphabeticShortcut`
- ◉ `android:numericShortcut`

programově:

- ◉ `setAlphabeticShortcut(char)`
- ◉ `setNumericShortcut(char)`

# SPUŠTĚNÍ JINÉ AKTIVITY

```
private void call() {  
    try {  
        Intent callIntent = new Intent(Intent.ACTION_DIAL);  
        callIntent.setData(Uri.parse("tel:9785551212"));  
        startActivity(callIntent);  
    }  
    catch (ActivityNotFoundException activityException) {  
        ...  
    }  
}
```



# APLIKACE S VÍCE AKTIVITAMI

- ◉ deklarace aktivit **v manifestu**
- ◉ spuštění další aktivity **přes Intent**
- ◉ můžeme počkat na výsledek spuštěné aktivity a dle něj reagovat
  
- ◉ `startActivity()`
- ◉ `startActivityForResult()`

# APLIKACE S VÍCE AKTIVITAMI

// toto je soubor **MojePrvni.java**

// v adresáři je i další soubor **Druha.java**

```
Intent i = new Intent(this, Druha.class);  
startActivity(i);
```

# PŘEDÁNÍ DAT AKTIVITĚ

první aktivita:

```
Intent intent = new Intent(this, trida.class);  
intent.putExtra("keyname",age);  
startActivity(intent);
```

druhá aktivita:

```
Intent i = getIntent();  
int age = i.getIntExtra("keyname", -1);
```

# PŘEDÁNÍ DAT AKTIVITĚ

```
Intent intent = new Intent(this, RecievingActivity.class);  
intent.putExtra("keyName", value);  
startActivity(intent);
```

-----

```
Bundle extras = intent.getExtras();  
if(extras != null)  
    String data = extras.getString("keyName");
```

Nebo:

```
String data =  
getIntent().getStringExtra().getString("keyName", "defaultKey");
```

Zdroj:

<http://stackoverflow.com/questions/5265913/how-to-use-putextra-and-getextra-for-string-data>

# APLIKACE Z VÍCE AKTIVIT

Opakování ze cvičení

Chceme-li vytvořit aplikaci z více aktivit, budeme potřebovat upravit následující soubory:

<b>název</b>	<b>komentář</b>
prvni.java	Hlavní aktivita, z které pustíme druhou
druha.java	Druhá aktivita
main.xml	Layout první aktivity
druha.xml	Layout druhé aktivity
AndroidManifest.xml	Přidat informaci o druhé aktivitě

# PRVNI.JAVA

```
1. public class prvni extends Activity {  
2.     @Override  
3.     public void onCreate(Bundle savedInstanceState) {  
4.         super.onCreate(savedInstanceState);  
5.         setContentView(R.layout.main);  
6.     }  
7.
```

obsluha stisku  
tlačítka

```
8.     public void tlacitko_prepni (View v) {  
9.
```

```
10.         Intent i = new Intent(prvni.this, druha.class);  
11.         startActivity(i);  
12.     }  
13. }
```

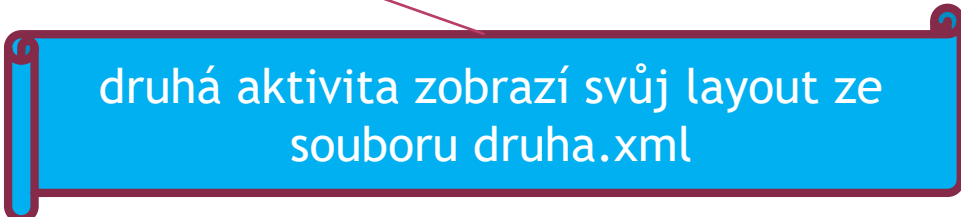
spuštění druhé  
aktivity

kontext

třída s novou  
aktivitou

# DRUHA.JAVA

1. package cz.pesi.vicaktivit;
2. import android.app.Activity;
3. import android.os.Bundle;
4. import android.view.View;
5. public class **druha** extends Activity {
6. @Override
7. public void onCreate(Bundle savedInstanceState) {
8. super.onCreate(savedInstanceState);
9. setContentView(R.layout.**druha**);
10. }
11. }



druhá aktivita zobrazí svůj layout ze souboru druha.xml

# MAIN.XML

- layout první (hlavní aktivity)
    - červený nápis „Aplikace z více aktivit“
    - tlačítko „Přepni na druhou aktivitu“
      - ošetřena událost onClick: tlacitko\_prepni
- 

<TextView

```
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="Aplikace z více aktivit"  
    android:textColor="#FF0000"/ >
```

```
<Button android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Přepni na druhou aktivitu"  
    android:onClick="tlacitko_prepni" ></Button>
```



# DRUHA.XML


- obsahuje jen nápis „Jsem druhá aktivita“
  - jak vytvoříme?
    - pravá myš na našem projektu
    - New - Android XML File
    - název: druhá
    - typ: layout  
(automaticky vybere úložiště /res/layout)
- 

```
<TextView  
  android:text="Jsem druhá aktivita"  
  android:id="@+id/textView1"  
  android:layout_width="wrap_content"  
  android:layout_height="wrap_content">  
</TextView>
```

# ÚPRAVA MANIFESTU

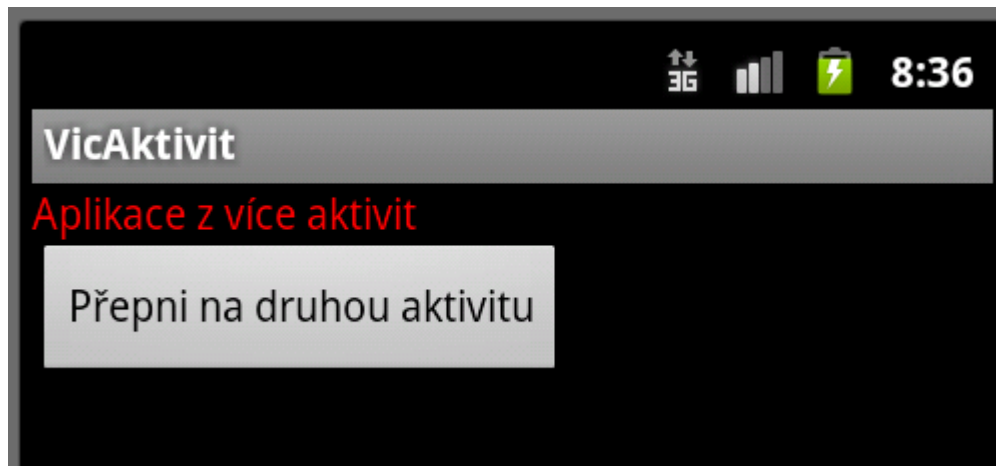
přidáme informaci o druhé aktivitě:

1. `<?xml version="1.0" encoding="utf-8"?>`
2. `<manifest xmlns:android="http://schemas.android.com/apk/res/android"`
3. `package="cz.pesi.vicaktivit"`
4. `android:versionCode="1"`
5. `android:versionName="1.0">`
  
6. `<application android:icon="@drawable/icon" android:label="@string/app_name">`
7. `<activity android:name=".prvni"`
8. `android:label="@string/app_name">`
9. `<intent-filter>`
10. `<action android:name="android.intent.action.MAIN" />`
11. `<category android:name="android.intent.category.LAUNCHER" />`
12. `</intent-filter>`
13. `</activity>`
14. `<activity android:name=".druha"></activity>`
  
15. `</application>`
16. `</manifest>`

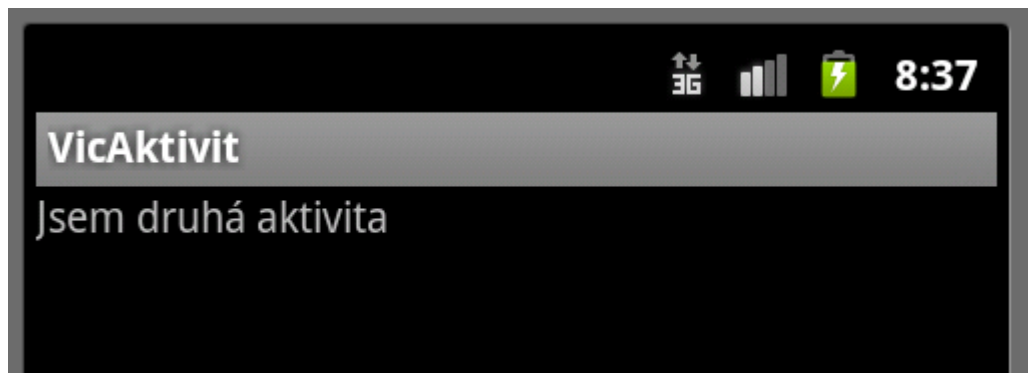


`.druha`  
vezme package  
+ tento řetězec

# APLIKACE Z VÍCE AKTIVIT



Z první aktivity se na druhou dostaneme stiskem vytvořeného tlačítka



Z druhé aktivity se na první dostaneme stiskem systémového tlačítka *Back*

# POUŽITÁ LITERATURA

- ⦿ <http://developer.android.com/>
- ⦿ [http://mobile.tutsplus.com/tutorials/android/android-sdk\\_table-layout/](http://mobile.tutsplus.com/tutorials/android/android-sdk_table-layout/)