



NTI/PAA - PROGRAMOVÁNÍ MOBILNÍCH APLIKACÍ

3. Aktivita, Intent, UI, View, layouty, fragmenty

Ing. Igor Kopetschke – TUL, NTI

<http://www.nti.tul.cz>

Android – Activity

- Základní komponenta pro zobrazení aplikace
- Bez existující aktivity nelze spustit aplikaci !
- Aplikace může mít i několik aktivit
- Prezentační vrstva aplikace
- Analogie s MVC/P – Android ale nemá View
- Data zobrazená z nižších vrstev je prezentována uživateli
- Životní cyklus je řízen pomocí **ActivityManager**, který pracuje se zásobníkem a zodpovídá za konkrétní stav aktivit
- Je vždy potomkem třídy `android.app.Activity`

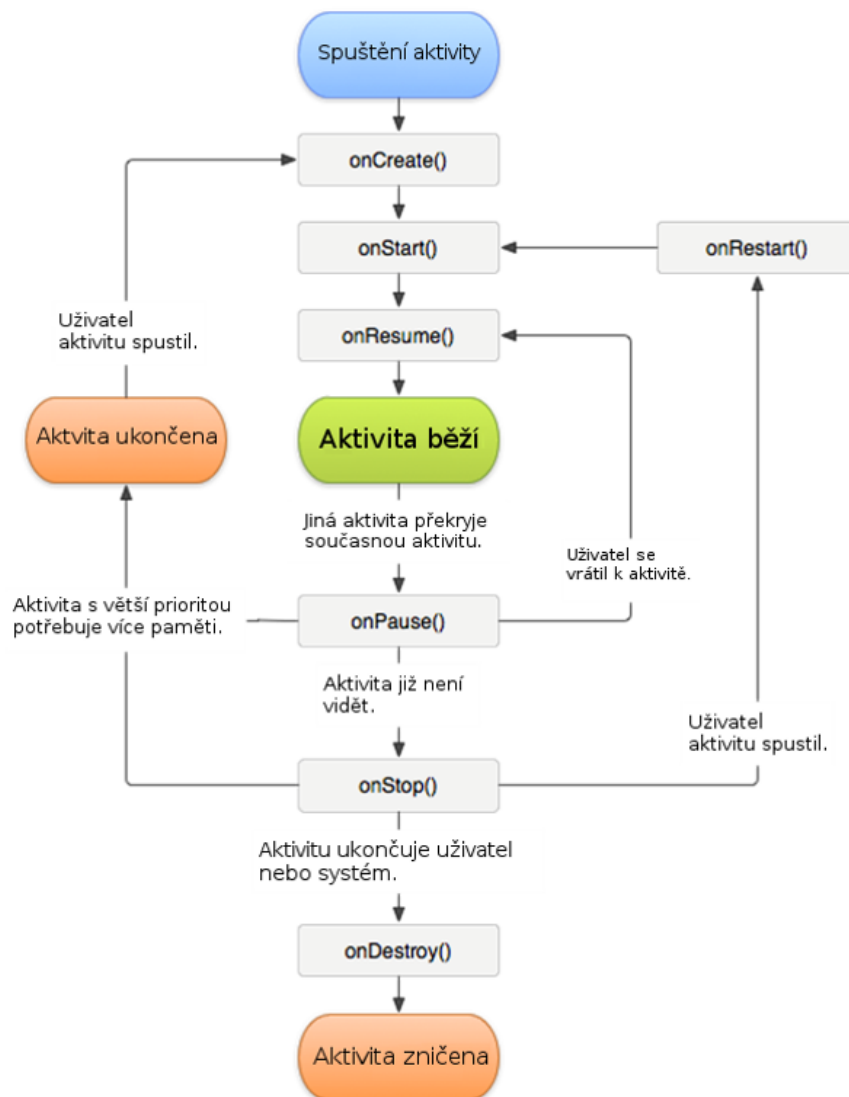
Android – životní cyklus aktivit

- Aktivita má 5 fází životního cyklu, v každém cyklu se volají metody zděděné z třídy **Activity**

Jednotlivé fáze životního cyklu

- **Aktivita spuštěna** – došlo ke spuštění aplikace
- **Aktivita běží** – aktivita spuštěna a je v popředí
- **Aktivita v pozadí** – je vidět, ale překryta jinou aplikací (příchozí SMS, hovor, jiná notifikace ..)
- **Aktivita zastavená** – není vidět, bez přístupu, ale není úplně zničena
- **Aktivita ukončená** – úplné ukončení aktivity

Android – životní cyklus aktivit



Android – metody Activity

■ onCreate()

- Volána při spuštění aktivity, která nebyla spuštěna nebo byla předtím zničena / odstraněna z paměti
- Zde se zavádí např. úvodní UI, načítá konfigurace atd.

■ onStart()

- Následuje metodu onCreate() nebo když je aktivita opět aktivována po svém skrytí – např. po vyřízení příchozí SMS

■ onResume()

- Volána těsně předtím, než je aktivita posunuta do popředí

■ onPause()

- Volána před přechodem aktivity do pozadí
- Systém získává pravomoc aktivitu násilně ukončit



Android – metody Activity

■ **onStop()**

- Aktivita již není viditelná
- Volána při zastavení aktivity.

■ **onDestroy()**

- Volána těsně předtím, než je aktivita úplně odstraněna/zrušena
- Vhodné jako „hook“ pro úklid atd ..

■ **onRestart()**

- Volána po metodě onStop() při restartu aplikace



Android – vytvoření Activity

- Jediná povinná metoda v naší aktivitě je **onCreate()**
- Tuto metodu je nutno vždy překrýt a implementovat
- Parametr typu **Bundle** slouží při znovuzavedení aktivity z pozadí k obnovení stavu
- Rodičovská třída **Activity** poskytuje celou řadu metod k překrytí – budeme se jimi zabývat později v jednotlivých kontextech

Android – vytvoření Activity

```
package com.tul.android.hello;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

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

```
package com.tul.android.hello
```

```
import android.app.Activity
```

```
import android.os.Bundle
```

```
class HelloKotlin() : Activity() {  
    protected override fun onCreate(savedInstanceState : Bundle?) {  
        super<Activity>.onCreate(savedInstanceState)  
        setContentView(R.layout.main)  
    }  
}
```


Android – Intent

- Základní asynchronní komunikační nástroj mezi prvky aplikací
- Třída, která obsahuje popis a data nějakého záměru
- Objekt s definicí cílového procesu s možností zaslat mu data
- Záměr může být implicitní či explicitní

- **Explicitní intent** – má informaci o konkrétní třídě, kterou chce spustit

```
Intent i = new Intent(context, MojeActivity.class);
```

- **Implicitní intent** – má pouze info o záměru (např. chci psát email) a případná data k předání. Nechá na systému, kterou aplikaci (aktivitu) spustí nebo nabídne.

```
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
```

Android – Intent

- Intent zpravidla obsahuje:
 - **Action** – záměr, který je potřeba vykonat
 - **Category** – jedna/více kategorií (jaký typ aktivity je potřeba)
 - CATEGORY_BROWSABLE – aktivita spouštěná prohlížečem
 - CATEGORY_LAUNCHER – zobrazena v seznamu app
 - CATEGORY_HOME – domovská obrazovka aplikace
 - **Extras** – key/value páry hodnot předaných intentu
 - URI a MIME typ
- Intent se zasílá pomocí
 - `startActivity(intent)` – odeslání záměru bez callbacku
 - `startActivityForResult(intent, requestCode)`
a zachytí se pomocí
 - `onActivityResult(requestCode, resultCode, intentData)`

Android – Intent a přenos dat

- S intentem se data předávají jako instance Bundle
- Bundle zapouzdřuje mapu:
 - Klíč – String
 - Hodnota
 - Primitivní typ – int, double, char, boolean ...
 - Objektový typ – jakýkoli serializovatelný objekt

■ Odeslání dat s intentem

```
Intent i = new Intent(this, TargetActivity.class);  
i.putExtra("klic1", 25);
```

nebo

```
i.getExtras().putByteArray("obrazek", pic);
```

■ Získání dat

```
i.getExtras().getByteArray("obrazek");
```

Android – Intent a přenos dat - příklad

■ Zdrojová aktivita

```
...  
int REQUEST_CODE = 2;  
Intent i = new Intent(this, TargetActivity.class);  
i.putExtra("cislo", 25);  
startActivityForResult(i, REQUEST_CODE);  
...
```

```
@Override  
protected void onActivityResult(int requestCode, int resultCode,  
Intent data) {  
    if (resultCode == RESULT_OK && requestCode == REQUEST_CODE) {  
        if (data.hasExtra("vysledek")) {  
            ... Zpracuj data ...  
        }  
    }  
}
```

Android – Intent a přenos dat - příklad

■ Cílová aktivita

- Zachytí a zpracuje předaná data
- A také zašle odpověď
- Odpověď je možno implementovat do některé překryté metody aktivity

```
...
@Override
public void finish() {
    Intent data = new Intent();
    data.putExtra("vysledek", vypocet);
    setResult(RESULT_OK, data);
    super.finish();
}
...
```

Android – Intent - příklady

■ Webová stránka

```
new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.tul.cz"));
```

■ Volání čísla

```
new Intent(Intent.ACTION_CALL, Uri.parse("tel: (+420) 485353111"));
```

■ Vytočení čísla

```
new Intent(Intent.ACTION_DIAL, Uri.parse("tel: (+420) 485353111"));
```

■ GEO data

```
new Intent(Intent.ACTION_VIEW, Uri.parse("geo:50.125,8.65?z=17"));
```

■ Otevření kontaktů

```
new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/"));
```

■ Otevření konkrétního kontaktu

```
new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/1"));
```

■ Získání obrázku

```
new Intent("android.media.action.IMAGE_CAPTURE");
```

Android – Intent – získání kontaktu

■ Získání dat kontaktu z adresáře telefonu

```
private static final int ACTIVITY_PICK_CONTACT = 485353111;

i = new Intent(Intent.ACTION_PICK, ContactsContract.Contacts.CONTENT_URI);
startActivityForResult(i, ACTIVITY_PICK_CONTACT);

...

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
        case ACTIVITY_PICK_CONTACT :
            if (resultCode == Activity.RESULT_OK) {
                //hotovo, máme kontakt
                Uri pickedContact = data.getData();
                return;
            }
            break;
    }
}
```

Android – <intent-filter>

- Element v manifestu
- Říká, na jaký Intent umí komponenta reagovat
- Není potřebná u komponent volaných explicitně
- Stejně jako Intent i zde se definuje
 - <action> – záměr, na který se reaguje
 - <category> – jedna/více kategorií
 - <data> – explicitní data

Android – <intent-filter> - příklad

■ Aktivita typu browser

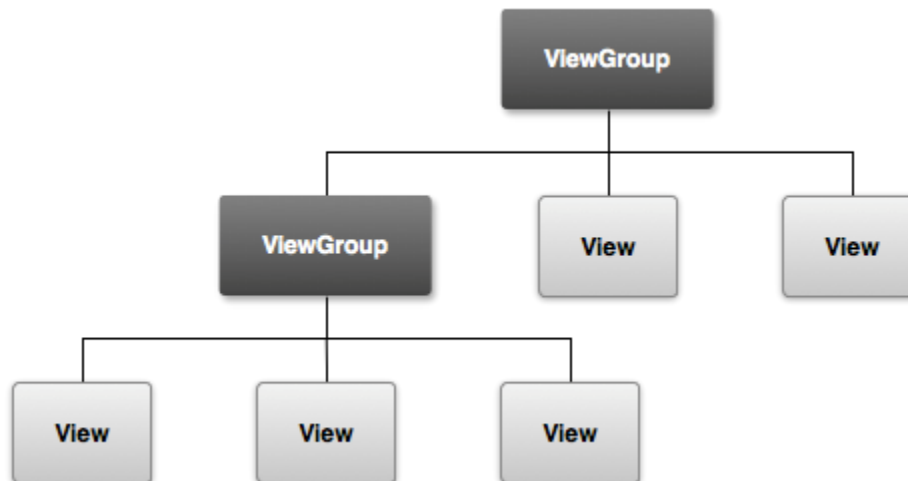
```
<activity android:name=".BrowserActivity">
  <intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="http" />
  </intent-filter>
</activity>
```

■ Aktivita pro konkrétní MIME typ

```
<activity android:name=".TextActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="text/plain" />
  </intent-filter>
</activity>
```

Android – základy UI

- Každý prvek UI je potomkem třídy **View**
- Z ní je odvozena důležitější třída **ViewGroup**
- **View** reprezentuje **právě jeden** konečný prvek UI a poskytuje důležité metody a atributy
- **ViewGroup** zapouzdřuje další View a ViewGroup





Android – atributy View

■ **android:id**

- ID grafického prvku, dohledatelné pomocí **findViewById()**

■ **android:alpha**

- Nastavení průhlednosti v rozmezí 0-1 (double)

■ **android:background, android:foreground**

- Resource nebo color pro pozadí / popředí

■ **android:padding**

- Vnitřní okraj View – možno též `paddingLeft`, `paddingTop` ...

■ **android:visibility**

- Viditelnost View – `visible`, `invisible`

Android – metody View

■ View findViewById(int id)

- Nalezne a vrátí **View** podle ID
- Jako **id** lze použít **R.id.hledane_id**

■ Context getContext()

- Vrátí instanci **Context** v jehož rámci **View** existuje.
- Užitečné v případě, že znáte pouze **View** ale potřebujete i **Context**

■ View rootView()

- Vrátí kořenové **View**

■ getX() , setX()

- Gettery a settery pro všechny atributy

Android – metody View

■ **void setOnXXXListener(View.onXXXListener)**

- Implementace celé rodiny listenerů

- Listener – posluchač událostí

- **Jsou to například :**

- **OnClickListener**
- **OnCreateContextMenuListener**
- **OnDragListener**
- **OnFocusChangeListener**
- **OnHoverListener**
- **OnKeyListener**
- **OnLongClickListener**
- **OnScrollChangeListener**
- **OnTouchListener**
- ...

Android – View a zachycení události

- Řada prvků UI umí reagovat na události
- Existují 3 základní způsoby, jak událost zachytit a ošetřit:
 1. Překrytí specializované metody zděděné z **Activity**
 2. Vytvoření specializovaného **listeneru** + předchozí slide
 3. Vytvoření **vlastní** metody + deklarace události přímo v **XML layoutu** u konkrétního UI prvku
- Záleží jen na logice aplikace a vývojáři, který způsob si v daném případě zvolí

Android – View a zachycení události

Překrytí specializované metody Activity

- Toto znáte z cvičení, kdy jsme zachytávali události z menu
- `boolean onOptionsItemSelected(int featureId, Menu menu)`
 - zachycení události, kdy je některé menu otevíráno
 - Identifikace menu dle předaných parametrů
 - nezapomeňte na konci metody vrátit **true / false**
- `boolean onOptionsItemSelected(MenuItem item)`
 - Zachycení vybrané položky menu dle **item**
 - Interně volá **onMenuItemSelected()**
 - Opět je na konci nutno vrátit **true / false**

Android – View a zachycení události

Vytvoření a zaregistrování listeneru

■ Naimplementujete si listener

□ Na úrovni Activity

```
public class MyActivity extends Activity
    implements OnClickListener
...
public void onClick(View v) { ... }
```

□ Vytvoříte si vlastní vnitřní třídu implementující listener

```
class MyListener implements OnClickListener {
    public void onClick(View v) { ... }
}
...
button.setOnClickListener( new MyListener() );
```


Android – View a zachycení události

Vytvoření vlastní metody + XML atribut v layoutu

- Naimplementujete vlastní metodu

```
public void necoDelej(View source) {  
    Button b = (Button)source;  
    ...  
}
```

- V layout XML definici u UI prvku:

```
<Button  
    android:width      = ...  
    android:height     = ...  
    Android:onClick   = "necoDelej"  
/>
```

Android – potomci View

■ TextView

- Pouhé zobrazení needitovatelného textu
- Má některé důležité atributy:
 - **android:gravity** – obdoba text-align v CSS (zarovnání textu)
 - **android:lines** – nastavení počtu řádků
 - **android:text** – resource na text nebo samotný text
 - **android:textcolor** – resource nebo barva

■ EditText

- Editovatelný text (formulářové pole)
- Má některé důležité atributy:
 - **android:inputtype** – text, date, time, number, textPassword, ...
 - **android:enabled** – true / false

Android – potomci View

■ ImageView

- Zobrazení obrázku
- Má některé důležité atributy:
 - **android:src** – resource obrázku
 - **android:layout_width, android:layout_height** – znáte z cvičení, hodnoty `wrap_content`, `fill_parent`, ...

■ ProgressBar

- Zobrazení průběhu činnosti
- Má některé důležité atributy:
 - **android:progress** – počáteční hodnota, zpravidla 0
 - **android:max** – maximální (konečná) hodnota

Android – ViewGroup

- Nejdůležitější potomek View
- Obsahuje / zapouzdřuje další prvky UI
- Stará se o jejich pořadí, rozmístění a vykreslení
- Důležité metody:
 - **addView(View v)** – přidá nové View na konec skupiny
 - **getChildCount()** – vrací počet View ve skupině
 - **getChildAt(int index)** – vrátí View na dané pozici
 - **indexOfChild(View v)** – vrací index View ve skupině
 - **removeView(View v)** – odebere View ze skupiny
 - **removeViewAt(int index)** – odebere View na pozici index
 - **removeAllViews()** – odebere všechny členy skupiny

Android – Layouts

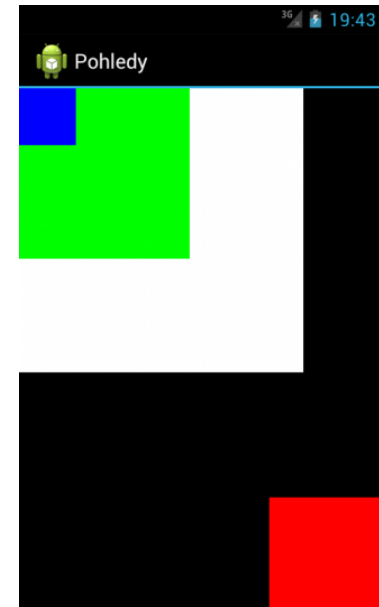
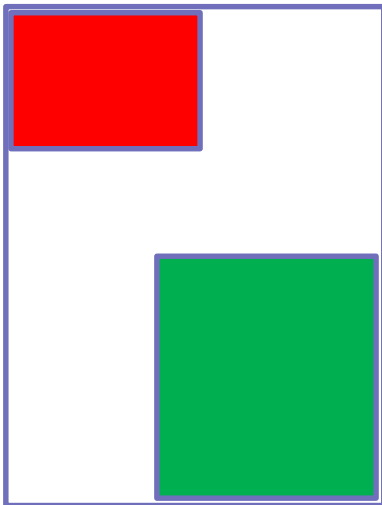
- Důležitými potomky **ViewGroup** jsou tzv. **layouts**
- Jedná se o rozložení a vzhled obrazovky
- Zpravidla tvoří kořenový element pro definici vzhledu pro aktivitu
- Jen pro připomenutí – jednotlivé definice vzhledu aktivity jsou v XML souboru v adresáři **res/layout**
- Odkaz na ně vede přes **R.layout.nazev_souboru**
- Na dalších stránkách si ukážeme nejvýznamnější layouts

Android – AbsoluteLayout

- Založen na principu umísťování potomků na konkrétní absolutní souřadnice
 - android:layout_x
 - android:layout_y
 - android:layout_width
 - android:layout_height
- Nepružný, nepraktický, nevhodný, nepoužívat
- Ve světě rozmanitých velikostí obrazovek a rozlišení je k ničemu
- Zavržen od API 3

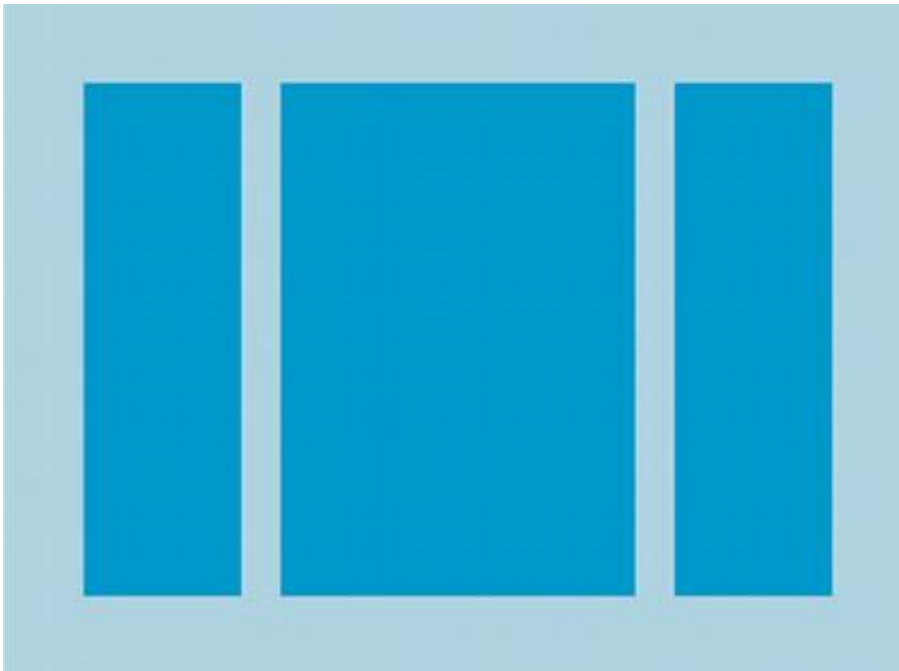
Android – FrameLayout

- Nejjednodušší a nejprimitivnější layout
- Zabere celou obrazovku a umisťuje potomky na sebe
- Což samo o sobě nevypadá moc prakticky
- Rozložení (zarovnání) v layoutu pomocí
 - **android:gravity** – kombinace vertikální | horizontální
 - top|left, bottom|right, center aj.



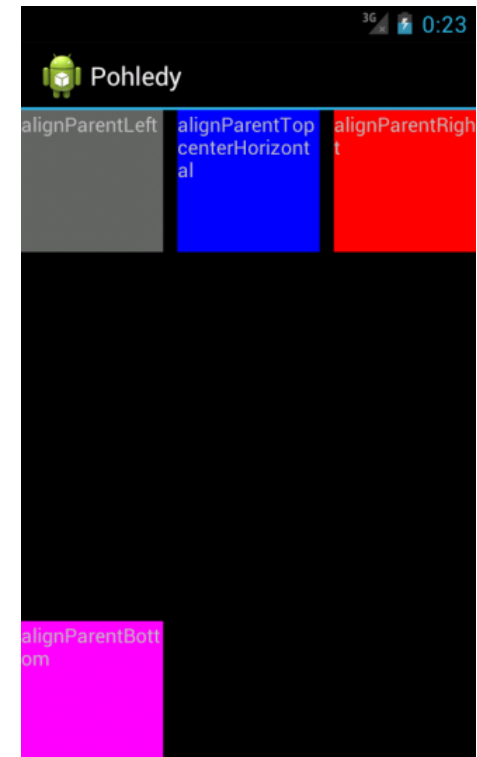
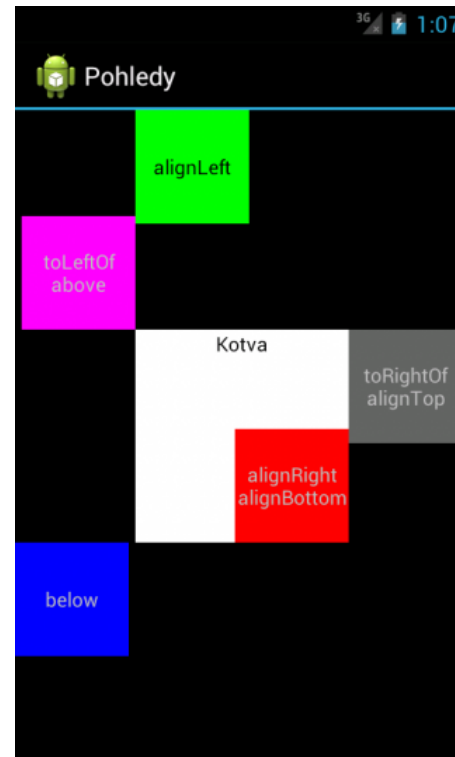
Android – LinearLayout

- Skládá prvky pod sebe nebo vedle sebe
- Rozložení (zarovnání) v layoutu pomocí
 - **android:orientation** – **horizontal** nebo **vertical**
 - **android:gravity** – to už známe



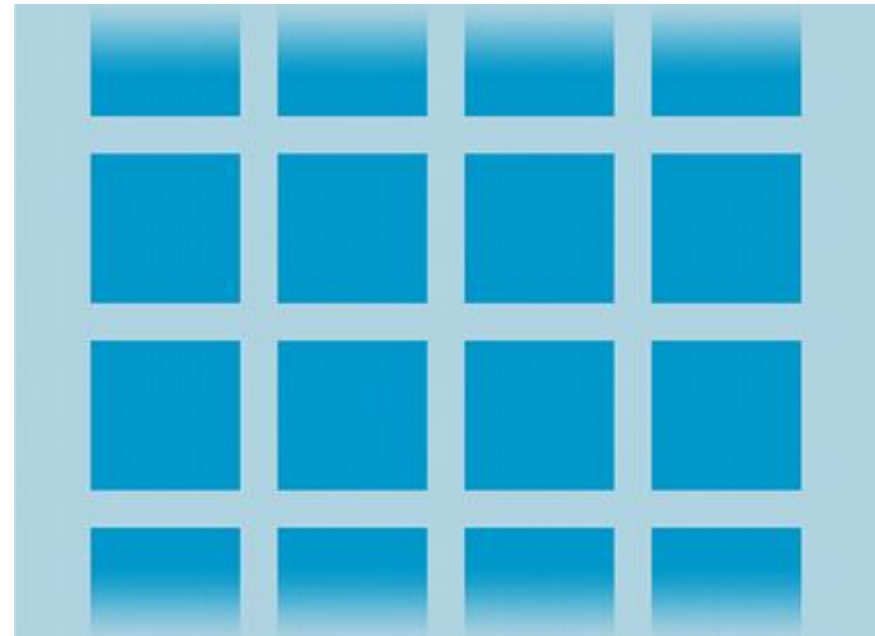
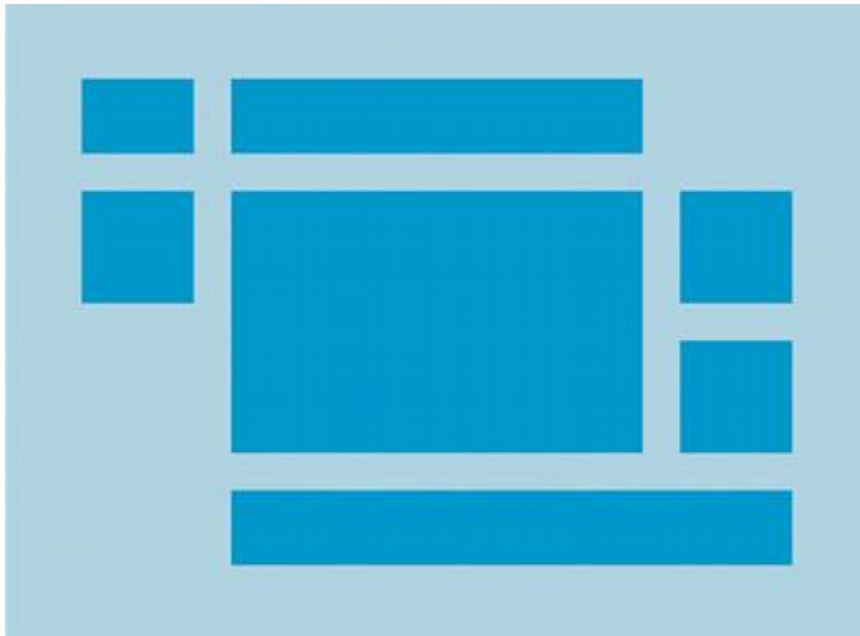
Android – RelativeLayout

- Skládá prvky relativně vůči ostatním
- Lze třeba říci, že A má být vedle B, dále C nad A atd..
- Atributy pro centrování a umístění



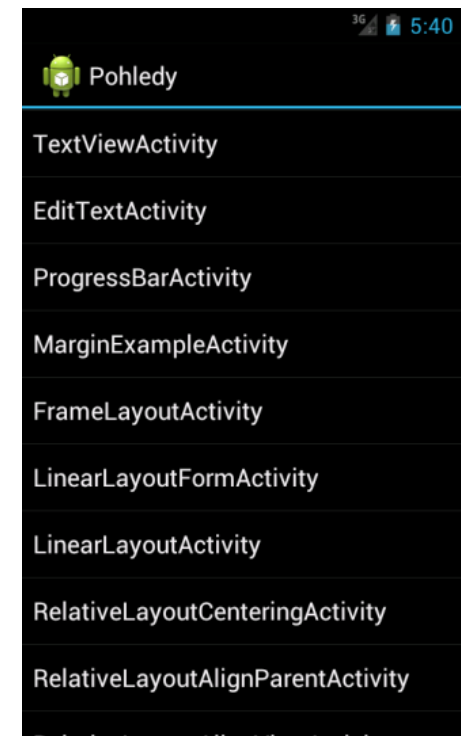
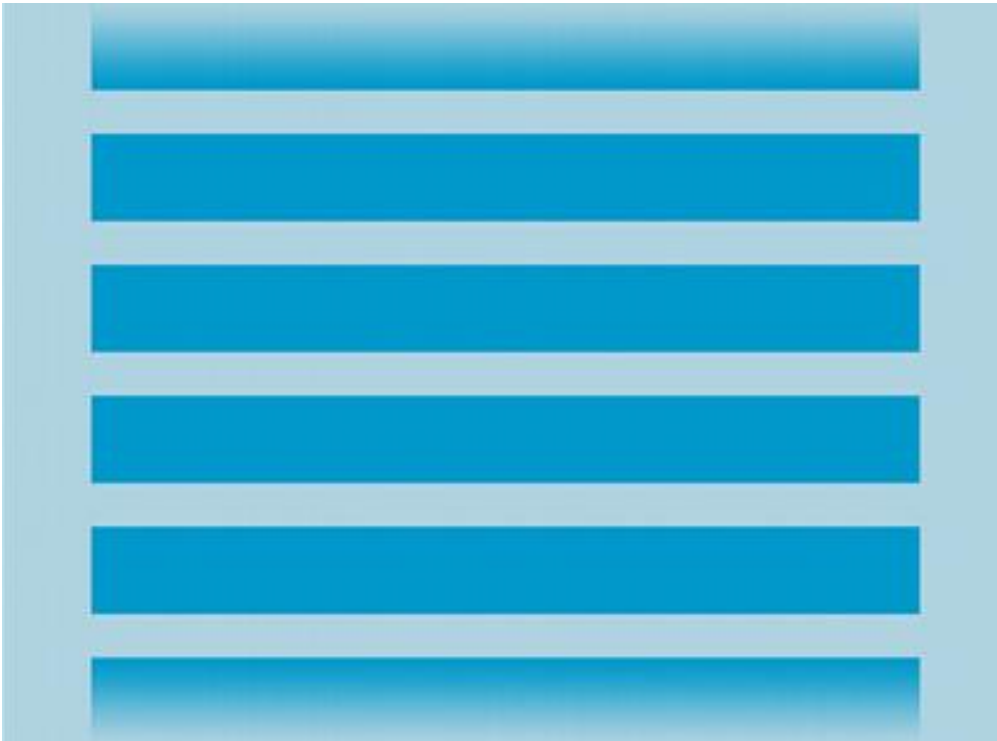
Android – TableLayout, GridView

- Skládá prvky do tabulky, resp. mřížky
- TableLayout – sloupce a řádky, slučování
- GridView – klasická mřížka, možno rolovat



Android – ListView

- Zobrazení potomků jako položky pod sebou
- Toto View podporuje rolování
- Nový prvek přidán na konec listu





Android – ScrollView

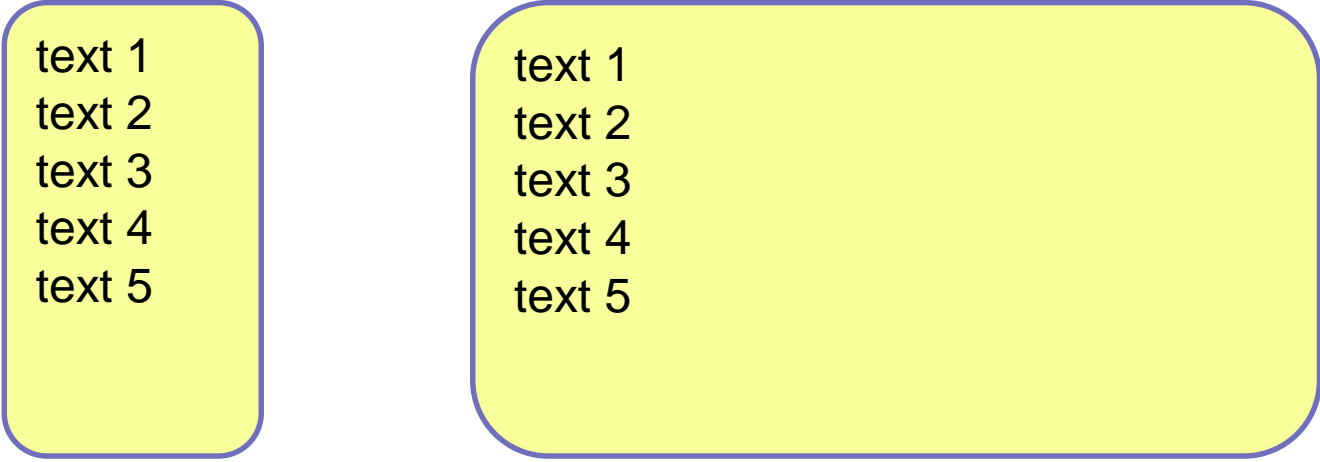
- Potomek FrameLayoutu
- Toto View podporuje rolování
- HorizontalScrollView, VerticalScrollView

Android – vybrané speciální View

- DatePicker, TimePicker
- Formulářové položky (tlačítka, checkbox, aj.)
- WebView
- AnalogClock, DigitalClock
- Gallery
- VideoView

Android – Fragmenty

- Původní Androidu koncept byl založen na Activity jako základním prvkem UI
- Každá Activity má nastavená svou View jako definici zobrazení
- Zlom přišel s tablety a Androidem 3.x
- Nutnost změny UI při zobrazení na šířku – původní aplikace často nevypadaly dobře



text 1
text 2
text 3
text 4
text 5

text 1
text 2
text 3
text 4
text 5

Android – Fragmenty

- Například aplikace se seznamem položek. Při kliknutí se zobrazil detail položky
- Varianta na výšku má 2 UI, varianta na šířku zvládne zobrazit vše najednou

item 1
item 2
item 3
item 4
item 5

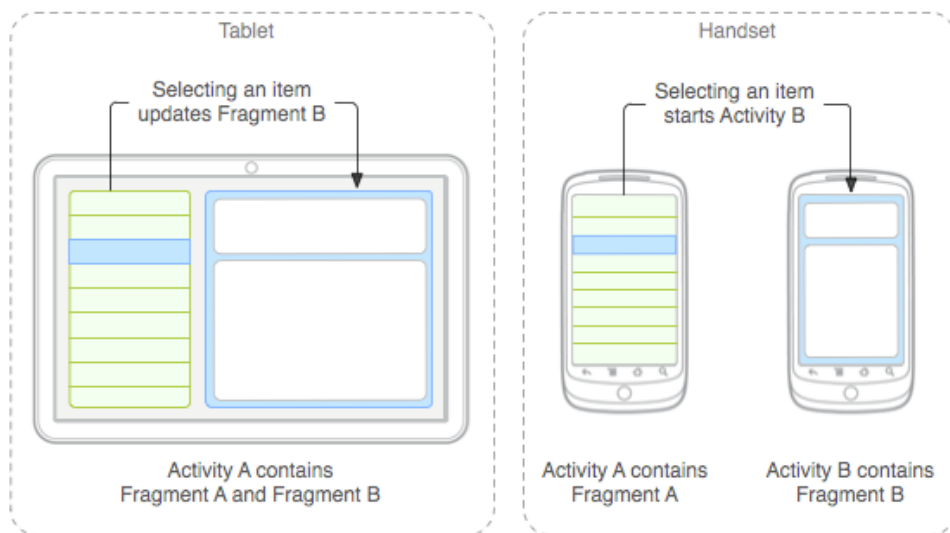
Detail 2
podrobný popis
detailu zvolené
položky, prostě
nějaký text ...

item 1 item 2 item 3 item 4 item 5	Detail 2 podrobný popis detailu zvolené položky, prostě nějaký text ...
---	--

- Existuje špatné a dobré řešení

Android – Fragmenty

- Špatné (zastaralé) řešení obnáší například:
 - Speciální aktivity pro portrait / landscape – nejen UI v layoutu, ale i logiku
 - Vytvářet a překreslovat UI nikoli v layoutech, ale přímo programově v kódu aktivity
 - Ve všech případech se nevyhnete duplicitám v kódu aktivit
 - A nebo to vůbec neřešit – aplikace potom může vypadat dost zoufale
- Správné řešení je použití tzv. fragmentů

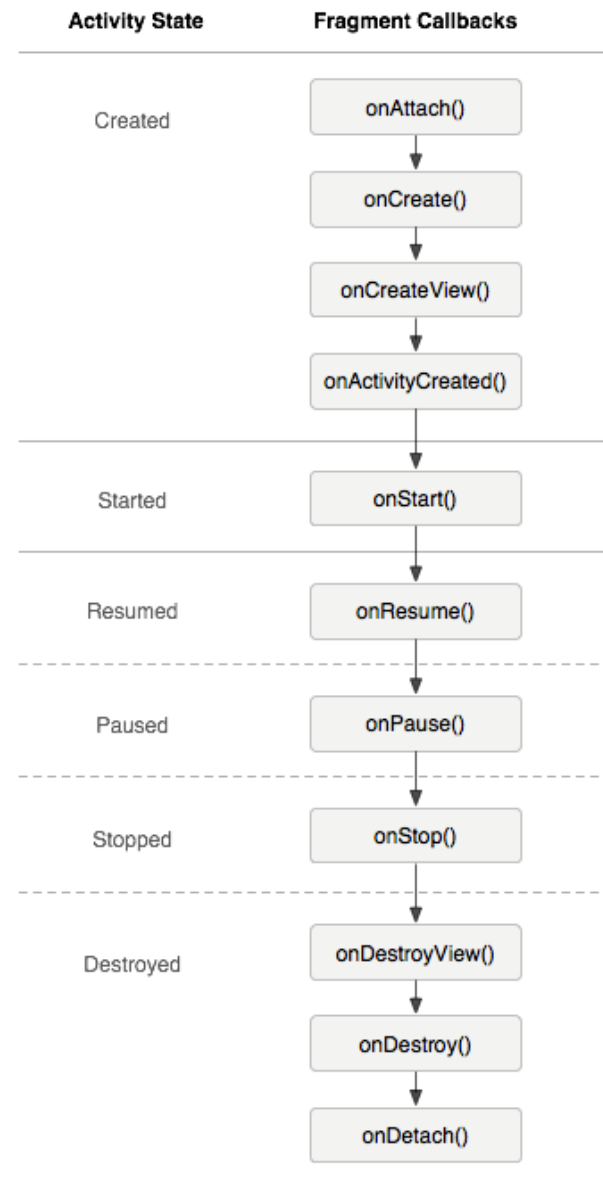


Android – Fragmenty

- Fragment je vrstva, která je vložena mezi Activity a View
- Reprezentuje část UI rozhraní včetně příslušných metod
- Fragment je možno vložit do příslušných Activity nebo dalších fragmentů
- Výhodou je flexibilita znovupoužitelnost jako jednotky UI
- Bez duplikace kódu lze programovat optimalizovaná UI pro různé varianty displejů
- Jejich vývoj je podobný jako u Activity – ale jen podobný
 - Rodičem je Fragment, nikoli Activity
 - Má mírně odlišné metody životního cyklu
 - Instance se vytváří přímo bez Intent
 - Data (Bundle) se fragmentu předávají jinou metodou, často jako součást statické tovární metody pro vytvoření instance fragmentu

Android – Fragmenty – životní cyklus

- Schéma zobrazuje fáze životního cyklu Activity a související callback metody zděděné z třídy Fragment
- Nejčastěji používané jsou:
 - `void onAttach(Activity a)` – volaná ve chvíli, kdy je fragment asociován s aktivitou
 - `View onCreateView(inflater, vgroup, state)` – vytváří a vrací UI související s fragmentem. Zde je vhodné asociovat Fragment s layoutem
 - `void onActivityCreated()` – volána v okamžiku, kdy byla ukončena metoda onCreate() související aktivity
 - `void onDestroyView()` – volána v okamžiku, kdy je “zničeno” View připojené k fragmentu
 - `void onDestroy()` – opak onAttach(), fragment ztrácí asociaci s aktivitou



Android – Fragmenty

- Fragmenty lze vytvářet nejen děděním přímo z třídy `Fragment`
- Android poskytuje specializované potomky
 - **BrowseFragment** – procházení položek, zobrazení detailního obsahu
 - **DialogFragment** – fragment pro dialogové “okno”
 - **ListFragment** – zobrazení seznamu (listu) položek včetně podpory pro callback výběru
 - **ErrorFragment** – zobrazení výskytu chyby
 - **PreferenceFragment** – fragment pro strom preferencí
 - **SearchFragment** – fragment pro vyhledávání
 - **WebViewFragment** – zobrazuje `WebView`
 - atd ...
- Specializované fragmenty poskytují nejen optimalizované UI, ale i dasy metod a listenerů pro práci s nimi

Android – Fragmenty

```
public class MyFragment extends Fragment {
    public static final String POZICE = "index";
    protected static String[] polozky = new String[] { //... };

    public static MyFragment newInstance(int index) {
        MyFragment f = new MyFragment();
        Bundle args = new Bundle();
        args.putInt(POZICE, index);
        f.setArguments(args);
        return f;
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        View v = inflater.inflate(R.layout.myfragment, container, false);
        int index = getArguments().getInt(INDEX, 0);
        TextView tv = (TextView) v.findViewById(R.id.details);
        tv.setText(polozky[index]);
        return v;
    }
}
```

Android – Fragmenty

- Fragmenty lze asociovat s Activity 2 způsoby
 - Přidáním **<fragment>** do layoutu Activity
 - Programově dynamicky pomocí FragmentManager
- Asociace pomocí **<fragment>**

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment class="cz.tul.paa.MyFragment
        android:id="@+id/seznam"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</FrameLayout>
```

Android – Fragmenty

- Dynamické vložení pomocí **FragmentManager**

```
Fragment fragment = MyFragment.newInstance();
FragmentManager fManager = getSupportFragmentManager();
FragmentTransaction fTransaction = fManager.beginTransaction();
// fcontainer - ViewGroup, do které je vlozen fragment
fTransaction.add(fragment, R.id.fcontainer);
fTransaction.commit();
...
// nebo zkracena varianta
getSupportFragmentManager().beginTransaction()
                                .add(R.id.fcontainer, f).commit();
```

- S Fragmenty se ještě setkáme na příští přednášce a hlavně na cvičení



Použité a doporučené zdroje

- <http://developer.android.com/>
- <http://www.zdrojak.cz/serialy/vyvijime-pro-android/>
- <http://www.itnetwork.cz/java/android>
- Google...



.. A to je pro dnešek vše

DĚKUJI ZA POZORNOST