Tvorba mobilních aplikací

5. seminář

Radek Janoštík

Univerzita Palackého v Olomouci

15. 3. 2022

Opakování

- Aktivity si předávají data pomocí třídy `Intent`
- Spuštění aktivity = vytvoření intentu, nastavení dat (`putExtra`) ve tvaru Klíč:hodnota
- `startActivity(intent)` – spuštění bez čekání na výsledek
- `startActivityForResult(intent, requestCode)` – spuštění s čekáním na výsledek
  - Spuštěná aplikace udělá svou činnost, nastaví návratová data do Intentu
  - Spustí metodu `setResult(RESULT_OK, intent)` a skončí `finish()`
  - V čekající aplikaci se spustí callback `onActivityResult()`, ve kterém zpracujeme vrácená data
- Veškerá data předávána jako klíč-hodnota – Klíče by měly být „statické“ konstanty (ne magické!)

Zobrazení seznamu položek

- Modelová situace: Aplikace pro ukládání poznámek

```
data class Note(val title : String, val text : String)
class MainActivity : AppCompatActivity() {
    val notes: MutableList<Note> = mutableListOf()
...
}
```

  - Máme uloženo několik tisíc poznámek
  - Jak je rozumně zobrazit?

Zobrazení seznamu položek

- Modelová situace: Aplikace pro ukládání poznámek

```
data class Note(val title : String, val text : String)
class MainActivity : AppCompatActivity() {
    val notes: MutableList<Note> = mutableListOf()
...
}
```

  - Máme uloženo několik tisíc poznámek
  - Jak je rozumně zobrazit?
  - Naivní řešení – pro každou poznámku vygeneruji `TextView`
  - Pomocí `ScrollView` je zobrazím

Zobrazení seznamu položek

- Modelová situace: Aplikace pro ukládání poznámek

```
data class Note(val title : String, val text : String)
class MainActivity : AppCompatActivity() {
    val notes: MutableList<Note> = mutableListOf()
...
}
```

  - Máme uloženo několik tisíc poznámek
  - Jak je rozumně zobrazit?
  - Naivní řešení – pro každou poznámku vygeneruji `TextView`
  - Pomocí `ScrollView` je zobrazím

- Asi by fungovalo, ale velká paměťová náročnost (pro každou poznámku `View` prvek)
  - Komplikovanější přidávání
  - Každý prvek vlastní reakce na událost

RecyclerView

- ⇒ Komponenta `RecyclerView`, která umí recyklovat `View` prvky
- „Paměťově šetrný zobrazovací prvek pro kolekce prvků“
- Interakce probíhá za pomocí *Adaptéru*
  - Zajišťuje napojení na interní kolekci
  - Vytvoření UI prvků pro zobrazené prvky
  - Překreslení po změně
- Můžeme jen normálně umístit do aktivity, ale musíme doprogramovat přístup k datům

RecyclerView

- ⇒ Komponenta `RecyclerView`, která umí recyklovat `View` prvky
- „Paměťově šetrný zobrazovací prvek pro kolekce prvků“
- Interakce probíhá za pomocí *Adaptéru*
  - Zajišťuje napojení na interní kolekci
  - Vytvoření UI prvků pro zobrazené prvky
  - Překreslení po změně
- Můžeme jen normálně umístit do aktivity, ale musíme doprogramovat přístup k datům
- Nadefinujeme, jak bude vypadat jedna položka – vytvoříme nový layout `note_item.xml`
  - LinearLayout pro zobrazení titulku a zkráceného textu poznámky
- Do `build.gradle` přidáme `implementation 'com.android.support:recyclerview-v7:28.0.0'`

Vytvoření adaptéru

- Adaptér bude mít k dispozici kolekci zobrazovaných položek
- Definuje, jak z položky udělat View prvek
- Metody:
  - `onCreateHolder()` – volána, když RecyclerView potřebuje novou položku
  - `onBindViewHolder()` – RecyclerView zobrazuje novou položku
  - `getItemCount()` – getter na velikost kolekce „na pozadí“
- Interní třída dědící `RecyclerView.ViewHolder` – reprezentuje jednu položku v seznamu, onClick(), . . .
- Ukázka

Floating Action Button (FAB)

- Aplikace pro android by měly dodržovat pravidla vzledu
- Material Design
  `https://developer.android.com/guide/topics/ui/look-and-feel`
- Soubor doporučení vzhledu, animací, chování, ...
- Jedním z prvků Material Designu je Floating Action Button `https://developer.android.com/guide/topics/ui/floating-action-button` – plovoucí tlačítko pro akci
- Můžeme přidat:

```
<android.support.design.widget.FloatingActionButton
    android:id="@+id/fab" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:layout_gravity="end|bottom"
    android:layout_margin="16dp" android:contentDescription="Pridat"
    android:onClick="fabClick" android:src="@drawable/ic_baseline_add_24"
    app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
```

- Přidání ikony, metody `onClick()`

Jednotky

- Pro definici velikostí ovládacích prvků máme k dispozici více jednotek
- `mm`, `in`, `pt`
  - Doporučuje se nepoužívat. Proč?

Jednotky

- Pro definici velikostí ovládacích prvků máme k dispozici více jednotek
- `mm`, `in`, `pt`
  - Doporučuje se nepoužívat. Proč?
- `px` – různě velké napříč začízeními
  - Také nepoužívat

Jednotky

- Pro definici velikostí ovládacích prvků máme k dispozici více jednotek
- `mm`, `in`, `pt`
  - Doporučuje se nepoužívat. Proč?
- `px` – různě velké napříč začízeními
  - Také nepoužívat
- `dp` – *density independent pixels*
  - Virtuální jednotka – přepočet podle hustoty zobrazovacího zařízení
  - Přepočet např. na `https://dpi.lv/`
  - Zavedeny kategorie hustot `https://developer.android.com/training/multiscreen/screendensities`
    `ldip, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi`
  - Můžeme přizpůsobovat layout dle hustoty – koncovk např. `-xhdpi` v resources
  - Pro `mdpi` $\approx$ 160dpi platí 1dp $\approx$ 1px
  - Zjištění měřítka: resources.displayMetrics.densityDpi

Jednotky

- Pro definici velikostí ovládacích prvků máme k dispozici více jednotek
- `mm`, `in`, `pt`
  - Doporučuje se nepoužívat. Proč?
- `px` – různě velké napříč začízeními
  - Také nepoužívat
- `dp` – *density independent pixels*
  - Virtuální jednotka – přepočet podle hustoty zobrazovacího zařízení
  - Přepočet např. na `https://dpi.lv/`
  - Zavedeny kategorie hustot `https://developer.android.com/training/multiscreen/screendensities`
    `ldip, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi`
  - Můžeme přizpůsobovat layout dle hustoty – koncovk např. `-xhdpi` v resources
  - Pro `mdpi` $\approx$ 160dpi platí 1dp $\approx$ 1px
  - Zjištění měřítka: resources.displayMetrics.densityDpi
- `sp` – podobné jako `dp`, používá se pro velikost písma, zohledňuje nastavení

Barvy

- Při specifikaci barev ovládacích prvků bychom měli udržovat jednotnost
- Mít barvu „napevno“ u elementů není dobrá praxe (změna)
- V souboru `/res/values/colors.xml` můžeme jednotlivé barvy pojmenovat
- → znovupoužitelnost, snadná změna

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="app_background">#FF88AA77</color>
    <color name="app_foreground">#FF555555</color>
</resources>
```

- Přístup z Kotlinu: `R.color.app_background`
- Přístup z XML: `android:textColor="@color/app_background"`

Řetězce

- Statické řetězce pro GUI by se neměly nacházet nikde v kódu
- Přesun do resources → `/res/values/strings.xml`

```
<resources>
    <string name="app_name">My Application</string>
    <string name="title">Title</string>
</resources>
```

- Přístup z Kotlinu: `R.string.title` z XML: `@strings/title`
- Podpora jazykových mutací – rozdělení resources
  - Vytvoření souboru: `/res/values-cs/strings.xml`
  - Systém vybere nejbližší mutaci (dle nastavení)
  - Možnost přepínání v Designeru

Orientace displeje

- Obdobně jako u jazykových mutací a hustoty bodů obrazovky můžeme definovat vzhled na základě orientace
- Může se hodit jinak přeskládat elementy na základě orientace obrazovky
- $\Rightarrow$ Speciální layout

Orientace displeje

- Obdobně jako u jazykových mutací a hustoty bodů obrazovky můžeme definovat vzhled na základě orientace
- Může se hodit jinak přeskládat elementy na základě orientace obrazovky
- $\Rightarrow$ Speciální layout
- Opět konvence pojmenování adresářů $\rightarrow$ `/r/layout-land.xml`, `/r/layout-port.xml`
- Ve statické třídě `R` jen jeden název pro více variant
- Id elementů musí být stejné (může se na ně odkazovat z kódu)

Grafika – bitmapy

- Bitmapové obrázky(`jpeg, png`) bychom měli dopředu přeškálovat
- Umístit do `/res/drawable-*` dle hustoty pixelů
- Přístup z Kotlinu: `R.drawable.obrazek`, z XML: `@drawable/obrazek`
- Systém vybere nejvhodnější obrázek
- Pokud chybí, přeškáluje (ztráta detailů, paměť, výkon)

Grafika – 9-patch bitmapy

- Speciální „roztahovatelné“ PNG soubory
- Vhodné např. pro grafická tlačítka, dialogy, . . .
- Obrázek rozšířen o 1px z každé strany
- Černé pixely vlevo a nahoře vymezují části, které mohou roztahovat
- Černé pixely vpravo a dole vymezují padding elementu (nepovinné)
- Ostatní pixely na krajích musí být buď bíle nebo průhledné
- Vložíme png obrázek→(pravý klik) Create 9-patch file (ukázka)
  - Vložíme tlačítko a nastavíme jako pozadí
  - Můžeme libovolně měnit velikost a určené části se budou roztahovat
- Kompromis mezi bitmapovou a vektorovou grafikou

Grafika – shapedrawables

- XML schéma pro kreslení jednoduchých tvarů (třeba předchozí tlačítko)
- `https://developer.android.com/guide/topics/resources/drawable-resource#Shape`
- Základní tvary: `rectangle, oval, line, ring`

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/purple_200" />
    <corners android:topLeftRadius="150dp"
        android:bottomRightRadius="100dp"/>
    <stroke android:color="@color/black" android:width="2dp"/>
</shape>
```

- Kombinace více tvarů jako
<layer–list><item></item><item></item></layer–list>

Vektorová grafika

- Android nemá nativní podporu pro klasické SVG
- Různé obcházení pomocí knihoven

Vektorová grafika

- Android nemá nativní podporu pro klasické SVG
- Různé obcházení pomocí knihoven
- Plná nativní podpora vektorové grafiky od API 21 (5.0 Lollipop)
- Vlastní formát pro vektorovou grafiku `https://developer.android.com/guide/topics/graphics/vector-drawable-resources`
- Podpora pouze podmnožiny vlastností z SVG

Vektorová grafika

- Android nemá nativní podporu pro klasické SVG
- Různé obcházení pomocí knihoven
- Plná nativní podpora vektorové grafiky od API 21 (5.0 Lollipop)
- Vlastní formát pro vektorovou grafiku `https://developer.android.com/guide/topics/graphics/vector-drawable-resources`
- Podpora pouze podmnožiny vlastností z SVG
- Android Studio obsahuje konvertor: (pravý klik)`res/drawable`→`Vector Asset`
- Můžeme vybrat z *clipartu* nebo ze souboru (který bude převeden)

Styly

- Podobně jako u barev bychom se neměli opakovat u definici vzhledu ovládacích prvků

Styly

- Podobně jako u barev bychom se neměli opakovat u definici vzhledu ovládacích prvků
- Mějme dvě textová pole

```
<EditText
android:id="@+id/PersonName"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@color/black"
android:fontFamily="monospace"
android:layout_margin="10dp"
android:text="Name" />
```

```
<EditText
android:id="@+id/PersonSurname"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@color/black"
android:fontFamily="monospace"
android:layout_margin="10dp"
android:text="Surname" />
```

- Při změně vzhledu bychom museli upravovat více míst

Styly

- Podobně jako u barev bychom se neměli opakovat u definici vzhledu ovládacích prvků
- Mějme dvě textová pole

```
<EditText
android:id="@+id/PersonName"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@color/black"
android:fontFamily="monospace"
android:layout_margin="10dp"
android:text="Name" />
```

```
<EditText
android:id="@+id/PersonSurname"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:background="@color/black"
android:fontFamily="monospace"
android:layout_margin="10dp"
android:text="Surname" />
```

- Při změně vzhledu bychom museli upravovat více míst
- $\Rightarrow$ definice stylů – soubor `/res/values/styles.xml`
- Obdoba kaskádových stylů *CSS*

Styly – ukázka

- Definice (`/res/values/styles.xml`)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="FormInput">
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:background">@color/app_background</item>
        <item name="android:fontFamily">monospace</item>
        <item name="android:layout_margin">10dp</item>
    </style>
</resources>
```

- Aplikace

```
<EditText
    android:id="@+id/PersonName"
    style="@style/FormInput"
    android:text="Name" />
```

```
<EditText
    android:id="@+id/PersonSurname"
    style="@style/FormInput"
    android:text="Name" />
```

Témata

- = globální styly pro celou aplikaci/aktivitu
- Definice v `/res/values/themes.xml`
  - Stejné jako styl
  - Dědí z předdefinovaných témat z SDK

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.MyApplication"
        parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/purple_500</item>
        <item name="colorPrimaryDark">@color/purple_700</item>
        <item name="colorAccent">@color/teal_200</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

- Změna tématu pro celou aplikaci→ `AndroidMaifest.xml` atribut `android:theme`

Úkol

1. Naprogramujte si vlastní RecyclerView pro seznam poznámek (nezobrazovat celý text poznámky)
2. Udělejte hlavní aktivitu se seznamem poznámek a FAB s ikonou +
3. Doprogramujte přidání nové poznámky a zobrazení celé poznámky
4. Veškeré řetězce a barvy mějte v resources
   - Udělejte podporu dvou jazyků (cs, en)
5. Společné vlastnosti vizuálních prvků přesuňte do stylů