Tvorba mobilních aplikací

4. seminář

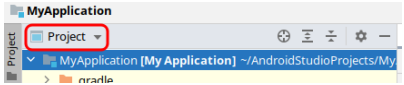
Radek Janoštík

Univerzita Palackého v Olomouci

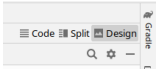
8. 3. 2022

Orientace v Android Studiu (1 / 2)

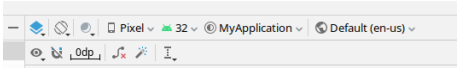
- Základní UI AS jistě intuitivně ovládnete (IntelliJ IDEA)
- Některé důležité prvky však můžete přehlédnout
- Vlevo – Project explorer window – doporučuji přepnout z „Android“ na Project



- V editoru uživatelské rozhraní dvě důležité části
  - Přepínání mezi kódem a designem

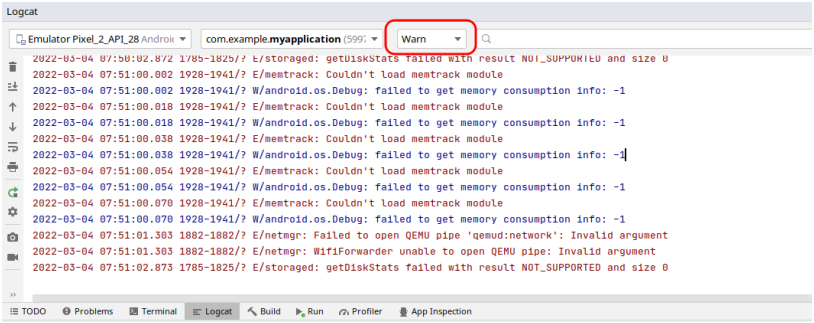


  - Přepínání zobrazení, témat, jazyků, api, zařízení



Orientace v Android Studiu (2 / 2)

- Velmi často bude potřeba podávat informace ze zařízení jinak, než v grafickém rozhraní
- Přístup do logovací konzole tzv. *Logcat*
- Nepotřebujeme ale vidět všechny zprávy (velké množství) → filtrace podle důležitosti



Struktura projektu

- Vybrali jsme *Empty acitivity project* $\rightarrow$ projekt ale není prázdný

Struktura projektu

- Vybrali jsme *Empty acitivity project* → projekt ale není prázdný
- Předdefinovaná šablona se spustitelným projektem a prázdnou *aktivitou*
- Čistě servisní adresáře – `.gradle`(nástroj pro překlad), `.idea`, `build`(sestavené soubory)

Struktura projektu

- Vybrali jsme *Empty acitivity project* → projekt ale není prázdný
- Předdefinovaná šablona se spustitelným projektem a prázdnou *aktivitou*
- Čistě servisní adresáře – `.gradle`(nástroj pro překlad), `.idea`, `build`(sestavené soubory)
- `app` hlavní adresář našeho *modulu* s aplikací (knihovny, zdrojové soubory, resources, testy)
  - `AndroidManifest.xml` – hlavní konfigurační soubor aplikace(název, ikona, počáteční aktivita, témata, oprávnění, . . .)
  - Do adresáře `java` budeme umísťovat náš kód v Kotlinu
- Jak v modulu, tak v top-level máme konfigurační soubory sestavovacího nástroje *Gradle*

Gradle

- Open-source nástroj pro automatizaci procesu sestavení aplikací
- Běží nad JVM, ale může automatizovat sestavení i jiných jazyků
- Řešení závislostí (Maven a Ivy)
- Sestavení modelováno pomocí DAG dílčích kroků(*task*) sestavení
- Napsán a konfigurován v jazyce Groovy
- Okno Gradle (schované úplně vpravo)
- „Díky Gradle sestavíme aplikaci na jedno kliknutí“ (více tasků na pozadí)

Gradle

- Open-source nástroj pro automatizaci procesu sestavení aplikací
- Běží nad JVM, ale může automatizovat sestavení i jiných jazyků
- Řešení závislostí (Maven a Ivy)
- Sestavení modelováno pomocí DAG dílčích kroků(*task*) sestavení
- Napsán a konfigurován v jazyce Groovy
- Okno Gradle (schované úplně vpravo)
- „Díky Gradle sestavíme aplikaci na jedno kliknutí“ (více tasků na pozadí)
- Jaké znáte jiné nástroje pro sestavování(i z jiných jazyků)?

Konfigurace gradle

- Top level konfigurace – týká se celého projektu
  - Nastavení repositářů (odkud stahovat knihovny)
  - Nastavení závislostí (potřebných modulů)
- Konfigurace modulu – samotné naší aplikace
  - Potřebné pluginy a závislosti
  - *compileSdk* jakým SDK bylo kompilováno
  - *applicationId* – jednoznačný identifikátor v Google Play, uživatel se s ním téměř nesetká, nemusí se shodovat s názvem *package*
  - *minSdk* – minimální potřebná SDK pro běh
  - *targetSdk* – verze API, na které byla aplikace testována(systém uplatňuje změny pro kompatibilitu)
  - *versionCode* – kód verze – musí být neklesající (při publikaci)
  - *versionName* – textové označení verze, cokoliv
- + další kompilační nastavení (cílová verze JVM, obfuskace kódu při *release*, ...
- Testy, podepisování balíčky, co dělat po sestavení aplikace, ...

Aktivity

- =hlavní stavební prvek aplikace pro Android

Aktivity

- =hlavní stavební prvek aplikace pro Android
- Aplikace tvořena několika (> 1) aktivitami
- Měla by provádět jednu konkrétní akci uživatele (formulář, výběr položky, udělání fotky, . . .)
  - Uživatel se soustředí na jeden jasný úkol
- Zobrazuje informace z aplikace a umožňuje ovládat aplikace

Aktivity

- =hlavní stavební prvek aplikace pro Android
- Aplikace tvořena několika (> 1) aktivitami
- Měla by provádět jednu konkrétní akci uživatele (formulář, výběr položky, udělání fotky, . . .)
  - Uživatel se soustředí na jeden jasný úkol
- Zobrazuje informace z aplikace a umožňuje ovládat aplikace
- Aktivity na sebe navazují, předávají si data
- Aktivita $\approx$ jedna obrazovka
- Je vybrána aktivita, která se spustí při startu aplikace
- Implementace vlastní aktivity rozšířením třídy `Activity`

Služby – *Services*

- Běh déle trvajících operací na pozadí
- Nemají uživatelské rozhraní
- Např.: Přehrávání hudby, stahování dat, sledování polohy, ...
- Mohou být spuštěny aktivitami, mohou s nimi komunikovat (zobrazovat stav)
- Dva typy
  - Started services – systém je nechá běžet, dokud nedokončí svoji práci (dva přístupy – informovanost uživatele vs. služba na pozadí, o které neví)
  - Bound services – spuštěny jako závislost jiné aplikace (poskytují API, službu)
- Implementace vlastní služby rozšířením třídy `Service`

Broadcast receivers

- V systému mohou nastat *události* (např. přijetí SMS, vybití baterie, hovor, . . .)
- Princip *publish/subscribe* – aplikace mohou vyvolat událost nebo se přihlásit k jejímu příjmu
- Nastane-li událost je vysílána(broadcast) a mohou ji zachytit Broadcast receivers
- Nemají UI, mohou vyvolat notifikaci do status baru
- Další vstupní bod do aplikace
- Vzniká rozšířením třídy `BroadcastReceiver`
- Vysílání je doručeno jako `Intent` (záměr)

Životní cyklus aplikace – motivace

- Desktop: Aplikace je načtena v paměti a běží „stále“

Životní cyklus aplikace – motivace

- Desktop: Aplikace je načtena v paměti a běží „stále“
- Máme omezené prostředky (RAM, baterie)
- Systém se snaží prostředky šetřit uspáváním, ukončováním nepotřebných aplikací/aktivit

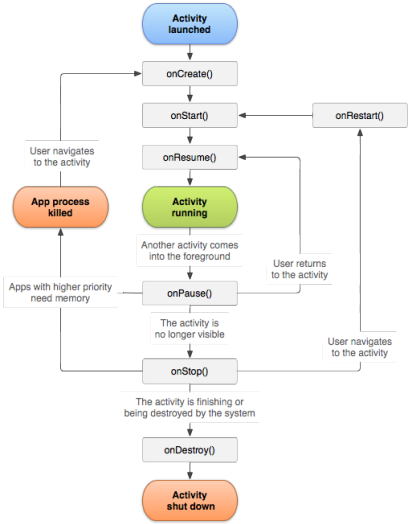
Životní cyklus aplikace – motivace

- Desktop: Aplikace je načtena v paměti a běží „stále“
- Máme omezené prostředky (RAM, baterie)
- Systém se snaží prostředky šetřit uspáváním, ukončováním nepotřebných aplikací/aktivit
- Nepoužívané (nezobrazené) aktivity se odsunují na *BackStack*
- V případě nedostatku systémových prostředků mohou být aktivity kdykoliv pozastaveny, ukončeny
- Aktivita nemá možnost samovolně měnit svůj stav

Životní cyklus aplikace – motivace

- Desktop: Aplikace je načtena v paměti a běží „stále“
- Máme omezené prostředky (RAM, baterie)
- Systém se snaží prostředky šetřit uspáváním, ukončováním nepotřebných aplikací/aktivit
- Nepoužívané (nezobrazené) aktivity se odsunují na *BackStack*
- V případě nedostatku systémových prostředků mohou být aktivity kdykoliv pozastaveny, ukončeny
- Aktivita nemá možnost samovolně měnit svůj stav
- $\Rightarrow$ Systém událostí, které mohou od OS přijít

Životní cyklus aktivity – schéma



Obrázek: developer.android.com/guide/components/activities/activity-lifecycle

Životní cyklus aplikace – stavy a callbacky

- Aplikace se může nacházet ve stavech:
  - `ACTIVE` – Spuštěná a zobrazená aplikace. Aktivity ve stavech `Created`, `Started` nebo `Resumed`. Uživatel s ní může interagovat. Nejvyšší priorita, nebude zabita.
  - `PAUSED` – Při dialogových „oknech“ – nemůžeme interagovat s UI aplikace. Aktivita ve stavu `Paused` *Většinou* nebude zabita
  - `STOPPED` – Při nahrazení jinou aplikací nebo stisk *Home* tlačítka(přes PAUSED). Aktivita ve stavu `Stopped`. *Může* být zabita při nedostatku zdrojů.
  - `DESTROYED` – Aplikace/aktivita byla zabita. Dále již není viditelná.
- Při přechodu mezi stavy je aplikace informována pomocí *callbacků*
  - onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy()
  - Může zareagovat na dané události (uložit si stav, odhlásit odběr dat ze senzorů, . . .)

Aktivita v kódu

- Rozdělena na kód(`MainActivity.kt`) a design (`res/layout/activity_main.xml`)

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

- Metoda onCreate() musí být implementována
  - Nastavení UI (layout)
  - Otevření datových zdrojů, získání dat z *Intentu*
  - Načtení automaticky uloženého stavu (v argumentu)

Design v kódu

- V metodě onCreate() jsme definovali, odkud se má vzít layout aktivity
- Soubor `activity_main.xml` definuje vzhled a rozložení prvků
- Kořenový element `ConstraintLayout` slouží jako kontejner pro ovládací prvky
- Elementy by měly mít Id (odkazování z kódu, pozicování pomocí *omezení(constraints)*
- Relativní pozicování elementů `https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout`
- Ukázka: Přidání dalšího elementu z kódu a jeho pozicování

Design z návrháře

- Podobně jako v ostatních UI Frameworcích můžeme tvořit design klikáním
- *Palette*(vlevo) seznam všech možných ovládacích prvků
  - Drag&Drop na místo, kde bychom prvek chtěli
  - Pomocí ikony  za nás AS odvodí omezení
  - Vyhýbejte se absolutnímu pozicování (proč?)
- *Attributes*(vpravo) – v okně můžeme editovat všechny atributy prvků
- Změna se promítá do XML a naopak

Tlačítko a události

- Vybereme z palety tlačítko a přesuneme do aktivity
- Necháme odvodit omezení a nastavíme mu `id` a `text`
- U tlačítka je důležité, co se má dělat po stisku
  - Nastavíme název metody pro obsluhu událost `onClick()`
- Teď již můžeme implementovat metodu:

```
fun magicClick(view: View) {
    Toast.makeText(this, "Kliknuto na tlacitko",
        Toast.LENGTH_LONG).show()
    Log.i("info", "Uzivatel kliknul na tlacitko")
}
```

Nová aktivita (1 / 2)

- Podle vzoru MainActivity vytvoříme novou
  - Vytvoříme novou Kotlin třídu
  - Nastavíme dědičnost a výchozí konstruktor
  - Přepíšeme metodu onCreate(), kde nadefinujeme layout

    setContentView(R.layout.activity_another)
- Vytvoříme nový layout s vhodným root elementem
- Ukázka: LinearLayout

Nová aktivita (1 / 2)

- Podle vzoru MainActivity vytvoříme novou
  - Vytvoříme novou Kotlin třídu
  - Nastavíme dědičnost a výchozí konstruktor
  - Přepíšeme metodu onCreate(), kde nadefinujeme layout

    setContentView(R.layout.activity_another)
- Vytvoříme nový layout s vhodným root elementem
- Ukázka: LinearLayout
- Nastavíme, aby se po kliknutí na tlačítko spustila nová aktivita

```
val textBox = findViewById<EditText>(R.id.textBox)
val zadanyText = textBox.text.toString()

val intent = Intent(this, AnotherActivity::class.java).apply {
    putExtra("Klic", zadanyText)
}
startActivity(intent)
```

Nová aktivita (2 / 2)

- Nyní by nám aplikace spadla, protože nová aktivita není zaregistrovaná

Nová aktivita (2 / 2)

- Nyní by nám aplikace spadla, protože nová aktivita není zaregistrovaná
- Přidáme novou aktivitu do Manifestu

  <activity android:name=".AnotherActivity">

- V metodě onCreate nové aktivity zpracujeme předaná data a nastavíme je do prvků GUI

```
override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_another)
        Log.i("info", "Vytvorena AnotherActivity")
        val txt = intent.getStringExtra("Klic")
        val textView = findViewById<TextView>(R.id.displayer)
        textView.text = txt
    }
```

Aktivita s návratovou hodnotou (1 / 2)

- V předchozím příkladu jsme pouze vytvořili aktivitu a předali jí data
- Může se hodit, aby „volaná aktivita“ vrátila „původní“ nějaká data
- Nemáme obdobu return – aktivity si pošlou zprávu a zareaguje na ni callback

Aktivita s návratovou hodnotou (1 / 2)

- V předchozím příkladu jsme pouze vytvořili aktivitu a předali jí data
- Může se hodit, aby „volaná aktivita“ vrátila „původní“ nějaká data
- Nemáme obdobu return – aktivity si pošlou zprávu a zareaguje na ni callback
- Spustíme novou aktivitu s *request kódem*

```
startActivityForResult(intent, 1234)
```

- Nadefinujeme reakci v callbacku:

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode==1234) {
        val sendedData = data?.getIntArrayExtra("RESULT")
        Log.w("Warning", sendedData.contentToString())
    }
}
```

Aktivita s návratovou hodnotou (2 / 2)

- Volaná aktivita může nastavit intent jako výsledek a skončit
- (Ukončení po stisku tlačítka)

```
fun anotherButtonClicked(view: View) {
    intent = Intent()
    intent.putExtra("RESULT", intArrayOf(1, 2, 3, 5))
    setResult(RESULT_OK, intent)
    finish()
}
```

- Data se předávají jako klíč - hodnota
- Všechny klíče by měly být jako statické

Úkol

1. Vytvořte aplikaci se 3 aktivitami
   - První pouze zobrazí uvítací obrazovku s tlačítkem
   - Ve druhé bude formulář obsahující textové pole pro jméno, datum narození a checkbox (souhlas s podmínkami)
   - Třetí tato data pouze zobrazí
2. Aktivitám implementujte všechny callbacky(změna stavu)s výpisem do logu
   - ověřte, zda se chovají v souladu s diagramem
3. Vyzkoušejte i spuštění aktivity a zpracování její návratové hodnoty