LUKÁŠ NOVÁK

# KMI / TMA

## TVORBA MOBILNÍCH APLIKACÍ

7. SEMINÁŘ | 4.11.2020

ZS 2020/2021 | STŘEDA 15:00-17:30

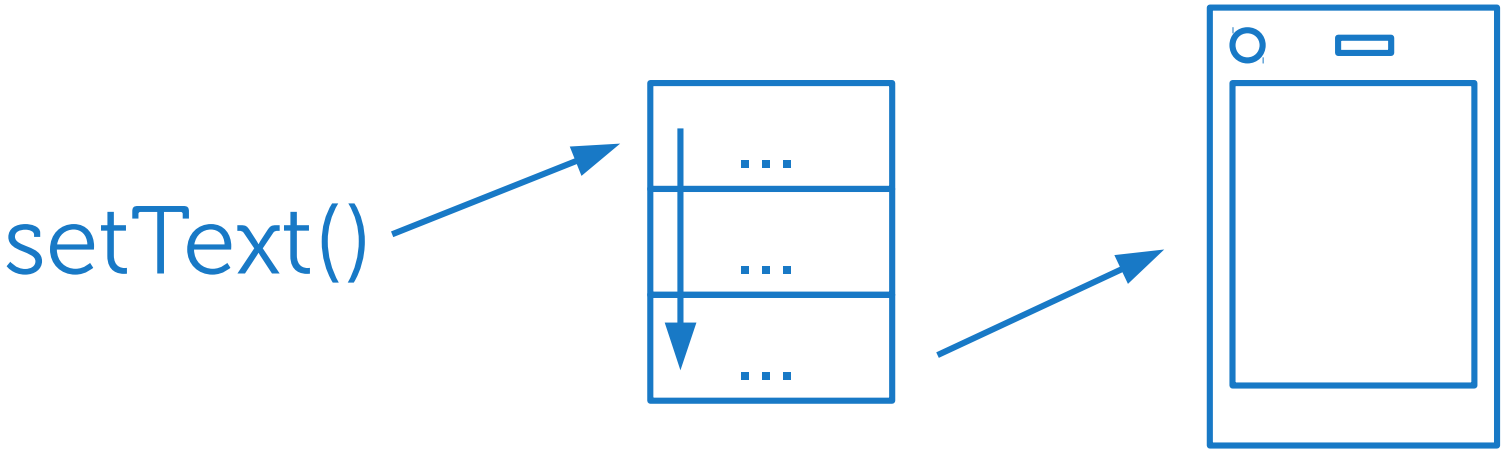
# OBSAH SEMINÁŘE

## PRÁCE NA POZADÍ,
## PRÁCE SE SÍTÍ

# PRÁCE NA POZADÍ

## PROČ JI POTŘEBUJEME?

- veškeré UI operace jsou prováděny na tzv. UI vlákně (někdy také hlavním vlákně/main thread)
- v UI vlákně je obsluhována fronta zpráv s požadavky na úpravy UI



# PRÁCE NA POZADÍ

## KDY JI POTŘEBUJEME?

- pro plynulé UI je nutné, aby na UI vlákně nebyly prováděny pomalé operace
  - síťové operace
  - práce s pamětí (tedy i DB)
  - složité výpočty, cykly, aj.
- ty je potřeba přesunout do jiných vláken
  - z nich však není možné měnit UI
  - jak z toho ven?

# PRÁCE NA POZADÍ

## JAKÉ MÁME MOŽNOSTI?

- třídy z **java.util.concurrent**
  - jen pro speciální případy
  - runOnUiThread(...), View.post(...)
- **AsyncTask:**
  - jednoduché asynch. operace
- **Handler:**
  - zpracování zpráv UI vláknem
- **Service, IntentService**
  - služba na pozadí, často overkill
- **Knihovny:** RxJava, Kotlin Coroutines, apod.

# PRÁCE NA POZADÍ

## ASYNCTASK

- **pomocná třída** pro řešení krátkých operací na pozadí, které spolupracují s UI
- pomocí **AsyncTask** můžeme definovat operace, které se provedou **před** provedením, **v průběhu** provádění a **po** provedení operace na pozadí

# PRÁCE NA POZADÍ

## ASYNCTASK

- vytvoříme potomka abstraktní třídy **AsyncTask**
- **AsyncTask** je generická, je potřeba definovat datové typy vstupu, průběhu a výstupu

```
public class MyBackgroundTask extends AsyncTask<String, Void, Integer> {
```

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- je nutné implementovat metodu **doInBackground(T...)**
  - tělo metody je vykonáno na pozadí bez blokování **UI vlákna**
  - **nelze** provádět **UI operace!**
  - přijímá variabilní počet argumentů prvního definovaného generického typu (**T**)
  - vrací instanci třetího definovaného generického typu (**V**)

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- lze implementovat metodu **onPreExecute()**
  - tělo metody je vykonáno na **UI vlákně** před metodou **doInBackground**
  - lze provádět UI operace
  - např. pro spuštění progress baru, zamezení opětovného provedení operace, aj.

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- lze implementovat metodu **onPostExecute()**
  - tělo metody je vykonáno po provedení **doInBackground**
  - přijímá výsledek metody **doInBackground**
  - lze provádět UI operace
  - např. pro zpracování výsledku, zastavení progress baru, apod.

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- **doInBackground(T...)**
  - z této metody je možné vícekrát volat metodu **publishProgress(U)**, která přijímá argumenty druhého definovaného generického typu (**U**)
  - pro aktualizaci progress baru, postupnou aktualizaci UI dle získaných výsledků, aj.

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- lze implementovat **onProgressUpdate(U)**
  - metoda vykonána po zavolání **publishProgress** z metody **doInBackground**
  - přijímá argumenty druhého definovaného generického typu (**U**)
  - lze provádět UI operace

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- lze implementovat metodu **onCancelled()**
  - tělo metody je vykonáno pokud je na instanci zavolána metoda **cancel(boolean interrupt)**
  - pokud je interrupt=true, pak lze zjistit, zda byla tato metoda zavolána pomocí metody **isCancelled** v **doInBackground**

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- po vytvoření instance potomka **AsyncTask** zavoláme metodu **execute** s variabilním počtem argumentů dle prvního definovaného generického typu
- tím jsou spuštěny metody **onPreExecute**, **doInBackground**, příp. **onProgressUpdate** a **onPostExecute**

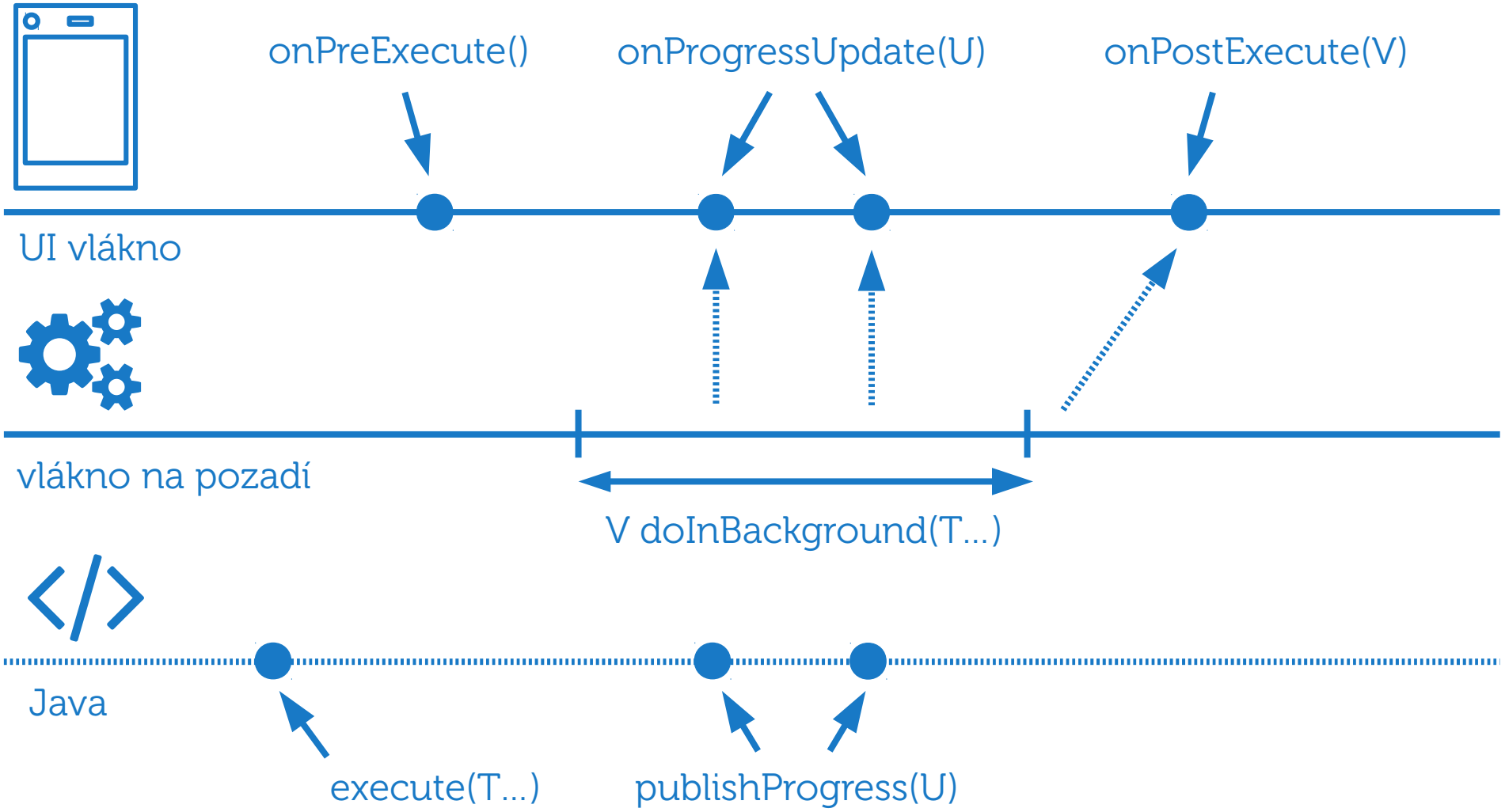
# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

- je nutné dbát na možnost restartování aktivity např. v případě změny orientace
- pokud budeme volat v **doInBackground** **getActivity** nebo si uložíme **Activity** do členské proměnné **AsyncTask**, musíme si dát pozor, zda tato aktivita žije a není to původní aktivita před změnou orientace
  - tj. podmínka != null, **!isDestroyed()**, apod.

# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>



# PRÁCE NA POZADÍ

## ASYNCTASK<T,U,V>

```
public class TmaTask extends AsyncTask<String, Integer, Integer> {

    @Override
    protected void onPreExecute() {
        // prepare UI before processing
        super.onPreExecute();
    }

    @Override
    protected Integer doInBackground(String... params) {
        // process Strings and return Integer
        return 0;
    }

    @Override
    protected void onPostExecute(Integer integer) {
        // process result Integer in UI
        super.onPostExecute(integer);
    }

    @Override
    protected void onProgressUpdate(Integer... values) {
        // update UI with Integers
        super.onProgressUpdate(values);
    }

}
```

# PRÁCE SE SÍTÍ

## JAKÉ JSOU MOŽNOSTI?

- **HttpUrlConnection**
  - získání dat pomocí HTTP
- **WebView/Chrome Custom Tabs**
  - „jednoduché“ zobrazení webu
- **DownloadManager**
  - stahování objemných souborů
- **3rd klienti**
  - OkHttp/Retrofit, Vollley, Apache HTTP Client, ...

# PRÁCE SE SÍTÍ

## HTTPURLCONNECTION

- HttpURLConnection
  - dostupná v SDK
  - komplexní třída pro práci se sítí
  - téměř vše, co je potřeba
    - SSL
    - HTTP autentizace
    - cookies
    - cache
    - ...

# PRÁCE SE SÍTÍ

## HTTPURLCONNECTION

- instanci vytvoříme metodou **openConnection** na instanci **URL**
- následně získáme proud pomocí **getInputStream**, se kterým můžeme pracovat pomocí standardních **java.io** možností (**BufferedInputStream**, **BufferedReader**, apod.)
- po dokončení potřeba uzavřít pomocí **disconnect()**

# PRÁCE SE SÍTÍ

## HTTPURLCONNECTION

```
private void downloadData() {

    try {
        HttpURLConnection connection = (HttpURLConnection)
            (new URL("http://novaklukas.cz/tma").openConnection());
        try {
            InputStream is = connection.getInputStream();
            InputStreamReader isr = new InputStreamReader(is);
            BufferedReader br = new BufferedReader(isr);
            // do something with reader
        } catch (IOException e) {
            Log.e(TAG, "Can't read file from URL", e);
        } finally {
            connection.disconnect();
        }
    } catch (MalformedURLException e) {
        Log.e(TAG, "Malformed URL", e);
    } catch (IOException e) {
        Log.e(TAG, "Can't create connection", e);
    }

}
```

# PRÁCE SE SÍTÍ

## BA-DUM-TSSS

```
ndroidRuntime: FATAL EXCEPTION: main
               Process: cz.upol.inf.kma.tmatodo, PID: 2714
               java.lang.RuntimeException: Unable to start activity ComponentInfo{cz.upol.inf.kma
                   at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:2416)
                   at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:2476)
                   at android.app.ActivityThread.-wrap11(ActivityThread.java)
                   at android.app.ActivityThread$H.handleMessage(ActivityThread.java:1344)
                   at android.os.Handler.dispatchMessage(Handler.java:102)
                   at android.os.Looper.loop(Looper.java:148)
                   at android.app.ActivityThread.main(ActivityThread.java:5417) <1 internal calls
                   at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:
                   at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:616)
                Caused by: android.os.NetworkOnMainThreadException
                   at android.os.StrictMode$AndroidBlockGuardPolicy.onNetwork(StrictMode.java:127
                   at java.net.InetAddress.lookupHostByName(InetAddress.java:431)
                   at java.net.InetAddress.getAllByNameImpl(InetAddress.java:252)
                   at java.net.InetAddress.getAllByName(InetAddress.java:215)
                   at com.android.okhttp.internal.Network$1.resolveInetAddresses(Network.java:29)
                   at com.android.okhttp.internal.http.RouteSelector.resetNextInetSocketAddress(R
                   at com.android.okhttp.internal.http.RouteSelector.nextProxy(RouteSelector.java
                   at com.android.okhttp.internal.http.RouteSelector.next(RouteSelector.java:100)
```

# PRÁCE SE SÍTÍ

## JE POTŘEBA PRACOVAT NA POZADÍ

- systém nepovolí prácí se sítí na hlavním UI vlákně i kdyby bylo připojení superrychlé
- potřeba pracovat se sítí v jiném vlákně
  - AsyncTask
  - služba na pozadí
  - ...

# PRÁCE SE SÍTÍ

## BA-DUM-TSSSS

```
do E/AndroidRuntime: FATAL EXCEPTION: AsyncTask #1
                     Process: cz.upol.inf.kma.tmatodo, PID: 7437
                     java.lang.RuntimeException: An error occurred while executing doInBackground()
                         at android.os.AsyncTask$3.done(AsyncTask.java:309)
                         at java.util.concurrent.FutureTask.finishCompletion(FutureTask.java:354)
                         at java.util.concurrent.FutureTask.setException(FutureTask.java:223)
                         at java.util.concurrent.FutureTask.run(FutureTask.java:242)
                         at android.os.AsyncTask$SerialExecutor$1.run(AsyncTask.java:234)
                         at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1113)
                         at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:588)
                         at java.lang.Thread.run(Thread.java:818)
                      Caused by: java.lang.SecurityException: Permission denied (missing INTERNET permission?)
                         at java.net.InetAddress.lookupHostByName(InetAddress.java:464)
                         at java.net.InetAddress.getAllByNameImpl(InetAddress.java:252)
                         at java.net.InetAddress.getAllByName(InetAddress.java:215)
                         at com.android.okhttp.internal.Network$1.resolveInetAddresses(Network.java:29)
                         at com.android.okhttp.internal.http.RouteSelector.resetNextInetSocketAddress(RouteSelector.ja
                         at com.android.okhttp.internal.http.RouteSelector.nextProxy(RouteSelector.java:157)
                         at com.android.okhttp.internal.http.RouteSelector.next(RouteSelector.java:100)
                         at com.android.okhttp.internal.http.HttpEngine.createNextConnection(HttpEngine.java:357)
                         at com.android.okhttp.internal.http.HttpEngine.nextConnection(HttpEngine.java:340)
                         at com.android.okhttp.internal.http.HttpEngine.connect(HttpEngine.java:330)
                         at com.android.okhttp.internal.http.HttpEngine.sendRequest(HttpEngine.java:248)
                         at com.android.okhttp.internal.huc.HttpURLConnectionImpl.execute(HttpURLConnectionImpl.java:4
                         at com.android.okhttp.internal.huc.HttpURLConnectionImpl.getResponse(HttpURLConnectionImpl.ja
```

# PRÁCE SE SÍTÍ

## SYSTÉM MUSÍ MÍT POVOLENÍ

- pro práci se sítí je třeba přidat oprávnění do **AndroidManifest.xml**
- do těla **<manifest...>** přidat tag:

```
<uses-permission android:name="android.
permission.INTERNET" />
```

# JSON

## BTW.

- webové služby často vrací data ve formátu JSON (JavaScript Object Notation)
- jde o zápis dat „podobný“ JS definici objektu
- v Androidu je možné použít třídy z **org.json**
  - **vytvoření JSON pole ze String**
    - JSONArray arr = new JSONArray(String)
  - **získání velikosti pole**
    - int length = arr.length()
  - **získání JSON objektu z pole**
    - JSONObject obj = arr.getJSONObject(index)
  - **získání hodnot z JSON objektu**
    - int val = obj.getInt(key), String val = obj.getString(key)

# ÚKOL 7. SEMINÁŘE

1) do toolbar menu přidat položku pro stažení TODO položek z URL: http://www.novaklukas.cz/tma/2019/seminars/07/todo_data.php
2) při stahování zobrazit standardní progress bar
3) po stažení zpracovat položky a přidat je do seznamu

**Tipy pro řešení:**

1) progress bar skrývejte/zobrazujte pomocí setVisibility(View.VISIBLE/View.GONE)
2) dejte si pozor, abyste neaktualizovali seznam v doInBackground (nelze měnit UI komponenty v jiném než UI vlákně)

# OTÁZKY

## PTEJTE SE!