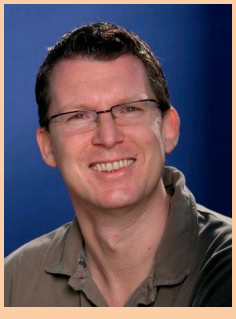
Hands-On

In een serie van drie artikelen laat Johan Pelgrim zien hoe je een Android-app kunt ontwikkelen. De applicatie heet 'Office Hours' en houdt zich bezig met het registreren van werktijd ten behoeve van de urenadministratie van jou als Java-professional. Dit eerste artikel is vooral als introductie bedoeld om bekend te raken met een aantal Android concepten.

Kantooruren in beeld met eigen Android app



Johan Pelgrim
Mobile Software Engineer bij VX Company.

Android SDK

Om met het voorbeeld in dit artikel aan de slag te kunnen heb je op zijn minst de Android Software Development Kit (SDK) nodig (1). Het pad waarin je de SDK hebt geïnstalleerd, wordt aangeduid met <ANDROID_SDK> in de rest van dit artikel. Je moet ook even de Hello World Tutorial (2) uitvoeren om te kijken of alles werkt. Hierin wordt ook beschreven hoe je een Android Platform installeert en een Android Virtual Device (AVD) moet aanmaken. Voor de app in deze serie artikelen heb je Android Platform versie 2.2 (API Level 8) nodig.

De start

Maak een Android-project aan met de volgende eigenschappen:

- **Build Target:** "Android 2.2". Dit is onze doelversie. Voer "<ANDROID_SDK>/tools/android list targets" uit om te kijken welke targets er mogelijk zijn.
- **Package:** "nl.javamagazine.officehours". Dit is een unieke identificatie waarmee de app bekend zal staan binnen het Android systeem.
- **Activity:** "TimerActivity". Objecten van het type Activity zijn vaak schermvullende user interfaces waarmee de gebruiker een activiteit kan uitvoeren. In de TimerActivity wil je een tijdregistratie kunnen starten en weer stoppen.

Via de command line ziet het creëren van een project er zo uit (3):

```
<ANDROID_SDK>/tools/android create project --target 7
 --name OfficeHours --path ~/officehours --activity
 TimerActivity --package nl.javamagazine.officehours
```

Probeer je project te bouwen in je project map met commando "ant debug" (4). Start de Android SDK manager op met: "<ANDROID_SDK>/tools/android". Selecteer **Tools | Manage AVDs** om je AVD aan te maken, te selecteren en te starten. Ga nu naar de bin map in je project map (~/officehours/bin in het voorbeeld hierboven) en installeer het OfficeHours-debug.apk bestand op de draaiende AVD met commando:

```
<ANDROID_SDK>/platform-tools/adb install -r OfficeHours-
debug.apk
```

In je project map zijn de src en res mappen en de AndroidManifest.xml en project.properties bestanden het meest belangrijk. Dit zijn ook typisch de zaken die je opslaat in je versiebeheersysteem. In de src map staat je broncode. In de res map staan alle bestanden die je nodig hebt voor je app, maar welke geen broncode bestanden zijn. In het AndroidManifest.xml bestand staat je app-definitie, welke het Android systeem gebruikt om je app te registreren. In project.properties staat de Android doelversie.

In de src/nl/javamagazine/officehours map is een class met naam TimerActivity aangemaakt. In deze class is de onCreate methode al geïmplementeerd. Dit is een activity lifecycle callback methode.

```
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
}
```

Activity lifecycle

De activity lifecycle van onze app wordt door de Activity Manager in het Android systeem beheerd. De drie lifecycle callback methodes die je het meest gebruikt zijn onCreate, onResume en onPause. De onCreate methode is voor eenmalige initialisatie en wordt vaak gebruikt om je user interface op te bouwen of te koppelen met een layout XML-bestand. In

onResume worden de dingen geïnitialiseerd die je nodig hebt voordat je activity op het scherm wordt getoond (bijvoorbeeld achtergrondmuziek starten). In onPause stop je vaak de dingen die je in onResume gestart hebt voordat je activity van het scherm wordt gehaald. De activity lifecycle is een essentieel onderdeel van de Android architectuur dus ik kan je aanraden hierover verder te lezen op de Android developers webiste (5).

Layout resources

In de gegenereerde onCreate methode wordt met een aanroep naar setContentView een View als top-level view aan de content window van onze Activity verbonden. Het argument "R.layout.main" is een verwijzing naar een layout resource. Android gebruikt deze bestanden om de daadwerkelijke top-level view object te creëren door deze layout resources "op te blazen" (inflate).

Hoewel je ook (complexe) user interfaces in code kunt maken, wordt geadviseerd om zoveel mogelijk van deze layout resources gebruik te maken. Je kunt namelijk met qualifiers meerdere layout resources maken die voor verschillende situaties worden ingezet7. Als je bijvoorbeeld ook een main.xml bestand in een nieuwe res/layout-land map plaatst, zal de user interface die daarin wordt gedefinieerd worden getoond wanneer je app in de landschapsmodus draait (zie voorbeeld in de broncode (10)).

Wanneer je main.xml opent zie je dat er een top-level node met naam LinearLayout beschreven staat, met een enkele TextView node daarin. Een View is de basis voor user interface componenten, zoals knoppen, lijsten, labels en dergelijke. Elke view heeft eigenschappen om de hoogte en de breedte aan te geven. Deze worden gezet met de android:layout_height respectievelijk android:layout_width attributen en kunnen twee waarden bevatten: "fill_parent", hiermee wordt alle ruimte benut om de view binnen de randen van de parent uit te vullen, of "wrap_content", hiermee wordt de hoogte of breedte van de view beperkt tot de hoogte of breedte van de inhoud van de view.

De LinearLayout is een layout manager die overerft van ViewGroup, die weer overerft van View. Een ViewGroup is een hiërarchie van View (en dus ook ViewGroup) objecten. Een layout manager zorgt ervoor dat de views op een bepaalde manier op het scherm getoond worden. De LinearLayout in het main.xml bestand doet dat recht onder elkaar. Dit wordt aangegeven in het android:orientation attribuut.

De gegenereerde R-class in de gen map bevat gegeneerde referenties naar de resources in de res map. Dit kunnen geluiden, afbeeldingen en videobestanden zijn, maar Android heeft ook uitgebreide mogelijkheden om (internationale) strings, kleuren, dimensies, arrays en meer los van de code te trekken en in deze res map te plaatsen (6).

Het TimerActivity scherm

Je hebt nu voldoende kennis op zak om de user interface van het eerste scherm van de Office Hours app te bouwen. Open het res/layout/main.xml bestand. Je kunt de bestaande TextView hergebruiken. Pas het android:text attribuut aan naar "@string/elapsed_time_label". Hiermee verwijs je naar een resource van type string. Het is aan te raden alle applicatieteksten buiten de layout XML-bestanden en code te houden. Plaats de volgende inhoud in het strings.xml bestand in de res/values map:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Office Hours</string>
    <string name="elapsed_time_label">Elapsed time
</string>
    <string name="elapsed_time">00:00:00</string>
    <string name="project_label">Project</string>
    <string name="timer_off">Start</string>
    <string name="timer_on">Stop</string>
</resources>
```

Dit zijn alle applicatieteksten die je in deze versie van de app nodig hebt. Je kunt nu ook heel gemakkelijk enkele teksten aanpassen voor gebruikers die Nederlands als taal hebben ingesteld op hun telefoon. Plaats een XML-bestand, weer met naam strings.xml, in een nieuwe map met qualifier (7) "nl" (res/values-nl) met de volgende inhoud:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="elapsed_time_label">Verstreken tijd
</string>
</resources>
```

Voeg een nieuwe TextView toe onder de Elapsed time-label met de volgende attributen:

```
<TextView
    android:id="@+id/elapsed_time"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:gravity="center_horizontal"
    android:text="@string/elapsed_time"
    android:textSize="80sp" />
```

Het "android:id" attribuut wordt gebruikt om in onze code naar een view, binnen onze layout, te kunnen refereren. Het plus-teken ("+") geeft aan dat je een id wilt laten genereren. Dat gebeurt eenmalig bij de definitie van de view in het layout XML-bestand. Zonder deze "+" geef je aan dat je naar een andere view wilt refereren binnen een layout XML-bestand.
Door het android:gravity attribuut de waarde "center_horizontal" te geven valt de tekst in het midden. De waarde "80sp" voor het android:textSize attribuut staat voor 80 scale independent pixels. Dit is een manier om schermonafhankelijke groottes

Een view is de basis voor user interface componenten.

aan te geven waardoor de tekst verhoudingsgewijs dezelfde grootte heeft op verschillende Android apparaten. De waarde sp of sip gebruik je voor tekst, dp of dip (density independent pixels) gebruik je voor grafische elementen (8).

Voeg nu een projectlijst toe door een combinatie van een TextView als label en een Spinner view te definiëren.

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/project_label" />
<Spinner
    android:id="@+id/project_spinner"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:entries="@array/project_list" />
```

De inhoud van de spinner kun je vullen met een array van strings.

De inhoud van de Spinner kun je vullen met een array van strings die je ook als XML-bestand in de res map kunt opslaan. Plaats een bestand arrays.xml met de volgende inhoud in de map res/values en voeg je eigen projecten als <item> nodes toe.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="project_list">
        <item>Java Magazine - Office Hours</item>
    </string-array>
</resources>
```

Voeg nu als laatste geen gewone knop, maar een ToggleButton toe.

```
<ToggleButton
  android:id="@+id/timer_button"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_weight="1"
  android:textOff="@string/timer_off"
  android:textOn="@string/timer_on"
  android:textSize="80sp" />
```

Met het android:layout_weight="1" attribuut zeg je dat de ToggleButton meer gewicht heeft dan de andere views in deze layout en zal de layout manager de resterende ruimte gebruiken om de views naar hun gewicht uit te vullen. Een ToggleButton heeft twee statussen, de checked state (de knop is ingedrukt) en de unchecked state. Als de knop niet ingedrukt is, wil je de tekst "Start" laten zien. Wanneer de knop ingedrukt is en de timer loopt, wil je de tekst "Stop" laten zien.

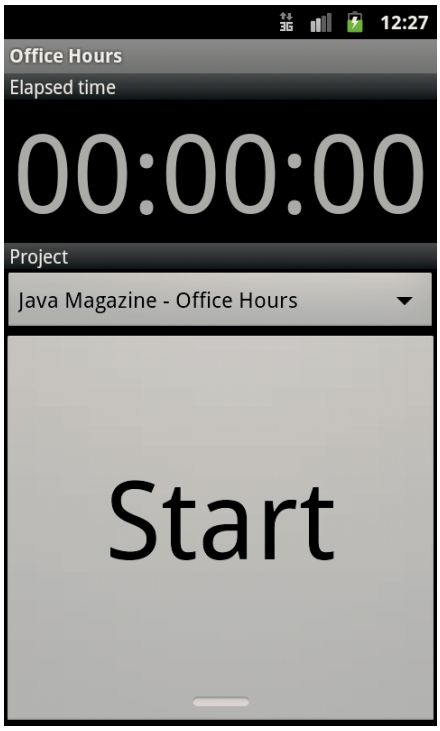
Voor de Elapsed time- en Project-labels ga je de achtergrond wijzigen. Dit kan een plaatje zijn, maar in Android kun je ook grafische elementen beschrijven in XML. Je gaat een shape drawable met een verloop in kleur van lichtgrijs naar donkergrijs maken6. Voeg een XML-bestand met naam label_background.xml en de volgende inhoud toe aan een nieuwe map res/drawable.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/
res/android"
  android:shape="rectangle">
  <gradient
    android:angle="90"
    android:centerX="0"
    android:centerY="0"
    android:endColor="#555"
    android:startColor="#222"
    android:type="linear"/>
  <padding
     android:left="4dp"/>
</shape>
```

Voeg attribuut android:background toe aan de Elapsed time- en Project-labels met waarde "@drawable/label_background".
Voeg ook nog een kleur toe als externe resource. Plaats een bestand met de naam colors.xml in de res/values map met de volgende inhoud:

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

Voeg attribuut android:textColor="@color/label_font" toe aan de Elapsed time- en Project-labels. Dit zijn slechts een paar voorbeelden van wat je allemaal als resource kunt definiëren. Meer voorbeelden vind je op de Android developers website6.
Bouw en run je Android-app. Als het goed is, ziet je layout er nu zo uit.



Je hebt de user interface gemaakt. Nu is het tijd om de achterliggende code te schrijven om de timer te starten en te stoppen. Ga terug naar het TimerActivity.java bestand.

Implementeren van timer logica

Wanneer je op de startknop drukt, wil je dat je app de starttijd onthoudt. Na elke seconde moet de text in de elapsed time-textview aangepast worden en als je weer op stop drukt moet dit stoppen. Dit kan

met een eigen thread, maar beter nog door het plaatsen van berichten op een Android MessageQueue.

User interfaces worden getekend in de main-thread, ook wel UI-thread genoemd. Elke thread heeft een MessageQueue van waaruit berichten worden opgepakt en uitgevoerd op de bijbehorende thread. Android kent de Handler class die berichten kan registreren op een MessageQueue die op een gegeven moment worden uitgevoerd op de thread waarin de Handler instantie is aangemaakt. Gebruik de postAtTime(Runnable r, long upTimeMillis) methode om elke seconde de elapsed time-textview aan te passen.
Introduceer een aantal private variabelen: mStartTime van type long en mHandler van type android.os.Handler. Introduceer ook twee private variabelen voor de toggle button en de elapsed time-textview. Deze heb je nodig om te luisteren naar een verandering in de checked state van de knop, respectievelijk het elke seconde updaten van de verstreken tijd.

```
public class TimerActivity extends Activity {

  private long mStartTime;
  private Handler mHandler;
  private TextView mElapsedTime;
  private ToggleButton mToggleTimerButton;

  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mStartTime = 0L;
    mHandler = new Handler();

    mElapsedTime = (TextView) findViewById(R.id.elapsed_
time);
    mToggleTimerButton = (ToggleButton) findViewById
(R.id.timer_button);
    (...)
```

Zet nu een onCheckChangeListener op de toggle button. Hiermee ga je bij het starten van de timer het mStartTime veld vullen en het eerste bericht via de mHandler variabele plaatsen op de message queue.

```
    (...)
mToggleTimerButton.setOnCheckedChangeListener(new
 OnCheckedChangeListener() {
      public void onCheckedChanged(CompoundButton
 buttonView, boolean isChecked) {
        if (isChecked) {
          mStartTime = System.currentTimeMillis();
          mHandler.removeCallbacks(mUpdateTimeTask);
          mHandler.post(mUpdateTimeTask);
        } else {
          mStartTime = 0L;
          mHandler.removeCallbacks(mUpdateTimeTask);
        }
      }
    });
  }
```

Met removeCallbacks worden alle nog uitstaande berichten van ons object uit de message queue verwijderd. Er wordt gerefereerd naar een instantie met naam mUpdateTimeTask van type Runnable. Maak deze aan als private variable met de volgende code:

```
private Runnable mUpdateTimeTask = new Runnable() {
  public void run() {
    final long timerStartTime = mStartTime;
    final long currentTimeMillis = System.
currentTimeMillis();

    final long elapsedTime = currentTimeMillis -
 timerStartTime;
    int scs = (int) (elapsedTime / 1000) % 60;
    int mns = (int) (elapsedTime / 1000 / 60) % 60;
    int hrs = (int) (elapsedTime / 1000 / 60 / 60) %
24;

    final long uptimeMillis = SystemClock.
uptimeMillis();
    final long systemStartTime = currentTimeMillis -
uptimeMillis;

    mElapsedTime.setText(String.format("%02d:%02d:%02d",
hrs, mns, scs));

    final long nextUpdateTime = elapsedTime -
(systemStartTime - timerStartTime) + 1000;
    mHandler.postAtTime(this, nextUpdateTime);
  }
};
```

In het kort wordt de tijd berekend tussen de starttijd van onze timer en de huidige tijd. De methode postAtTime vraagt echter om een tijd in milliseconden die relatief is aan de tijd wanneer het toestel voor het laatst is opgestart.

Bouw en run de app. Druk op de startknop en zie dat de timer gaat lopen. Eigenlijk loopt er niets, je onthoudt alleen de starttijd en je ververst de inhoud van de textview elke seconde. Draai je telefoon eens in landschapsmodus en zie wat er gebeurt (in de emulator doe je dat met de knoppen ctrl-F11).

Je ververst de inhoud van de textview elke seconde.

Persisteren van de starttijd

Bij elke rotatieverandering wordt het huidige Activity object weggegooid en een nieuwe opgebouwd in landschapsmodus. Met andere woorden, onze mStartTime variabele wordt op dat moment weer op 0L gezet. Je kunt dit oplossen door de starttijd te persisteren. Android heeft daar verschillende mogelijkheden voor (9). Dit keer gebruik je de SharedPreferences wat het mogelijk maakt om key-value paren van primitieve data op te slaan. Je opent een SharedPreferences bestand (wat uiteindelijk gewoon een XML-bestand is op je telefoon) met een naam en een modus. De app is de enige gebruiker van dit bestand, dus open deze in de privé modus (int met waarde 0). Maak een private variabele aan van type SharedPreferences en initialiseer deze in de onCreate methode:

```
private SharedPreferences mSharedPreferences;
public void onCreate(Bundle savedInstanceState) {
  (...)
  mSharedPreferences = getSharedPreferences("OfficeHour
s", 0);
```

Zet de waarde van de mStartTime variabele nu op de waarde uit de shared preferences door de metho-

In het volgende artikel: Opslaan in een SQLite database.

de getLong(String key, long default) te gebruiken.

```
mStartTime = mSharedPreferences.getLong("mStartTime",
 0L);
```

Zodra je op de startknop drukt (dit event wordt afgevangen in de if constructie in de onCheckedChanged methode die je in onCreate hebt toegekend), schrijf je de nieuwe starttijd weg naar het shared preferences bestand. Hiervoor heb je een SharedPreferences.Editor ojbect nodig. Deze krijg je door de methode edit() aan te roepen op onze SharedPreference instantie. Hiermee kun je data toevoegen of wijzigen en de wijzigingen committen.

```
if (isChecked) {
  if (mStartTime == 0) { // Alleen zetten wanneer er
niet al een timer "loopt"
    mStartTime = System.currentTimeMillis();
    Editor editor = mSharedPreferences.edit();
    editor.putLong("mStartTime", mStartTime);
    editor.commit();
  }
  mHandler.removeCallbacks(mUpdateTimeTask);
  mHandler.post(mUpdateTimeTask);
} (...)
```

Als je de stopknop indrukt (dit event wordt afgevangen in de else constructie in de onCheckedChanged methode die je in onCreate hebt toegekend) kun je de mStartTime variabele weer op 0L zetten en de waarde in het shared preferences bestand verwijderen.

```
(...) else {
  mStartTime = 0L;
  Editor editor = mSharedPreferences.edit();
  editor.remove("mStartTime");
  editor.commit();
  mHandler.removeCallbacks(mUpdateTimeTask);
}
```

Implementeer nu de onResume lifecycle callback methode om alvast wat zaken te initialiseren wanneer je een een actieve timer hebt (mStartTime > 0).

```
protected void onResume() {
  super.onResume();
  if (mStartTime > 0) {
    mToggleTimerButton.setChecked(true);
    mHandler.removeCallbacks(mUpdateTimeTask);
    mHandler.post(mUpdateTimeTask);
  }
}
```

Als laatste is het goed om in de onPause methode alle berichten op de message queue van onze mHandler op te ruimen. De onPause wordt aangeroepen voordat onze user interface van het scherm wordt gehaald. Het is dan toch niet meer van belang om de elapsed time-textview aan te passen.

```
protected void onPause() {
  super.onPause();
  mHandler.removeCallbacks(mUpdateTimeTask);
}
```

Tot slot

Je hebt in sneltreinvaart gezien hoe je een user interface kunt opbouwen met behulp van een layout XML-bestand. Ook heb je een aantal zaken zoals teksten, kleuren en een string-array buiten de code gehouden door ze als resource op te slaan. Je hebt kennis gemaakt met de Handler class om de user interface elke seconde aan te passen. Je hebt de SharedPreferences class gebruikt om de starttijd op te slaan en zo de timer robuust te maken en fictief te laten doorlopen, ook als bijvoorbeeld het toestel uit en weer aan wordt gezet.

In het volgende artikel laat ik je zien hoe je tijdregistraties kunt opslaan in een locale SQLite database en ga je de opgeslagen tijden uit deze database tonen in een lijst op het scherm.

Ik hoop dat dit artikel je aanmoedigt om meer te doen met Android. Als je vragen hebt over dit artikel kun je me mailen op e-mail adres jpelgrim@vxcompany.com. Hier kun je ook terecht als je interesse hebt in de Android Workshops die ik geef. Wanneer je het leuk vindt om andere Android ontwikkelaars te ontmoeten wil ik je uitnodigen om je aan te melden bij de Nederlandse Android gebruikersgroep via http://www.dutchaug.org. «

Referenties

(1) Android SDK
http://developer.android.com/sdk
(2) Hello World tutorial
http://developer.android.com/resources/tutorials/hello-world.html
(3) Projects via Cmd Line
http://developer.android.com/guide/developing/projects/projects-cmdline.html
(4) Running via Cmd Line
http://developer.android.com/guide/developing/building/building-cmdline.html
(5) Activity Lifecycle
http://developer.android.com/guide/topics/fundamentals/activities.html#Lifecycle
(6) Resource Types
http://developer.android.com/guide/topics/resources/available-resources.html
(7) Providing Resources
http://developer.android.com/guide/topics/resources/providing-resources.html
(8) Density independence
http://developer.android.com/guide/practices/screens_support.html
(9) Data Storage
http://developer.android.com/guide/topics/data/data-storage.html
(10) Broncode:
https://github.com/downloads/jpelgrim/office-hours/officehours-artikel1.zip