Logger

Android Logger

Logujemy przy pomocy **android.util.Log**.
Korzystamy z niego następująco:

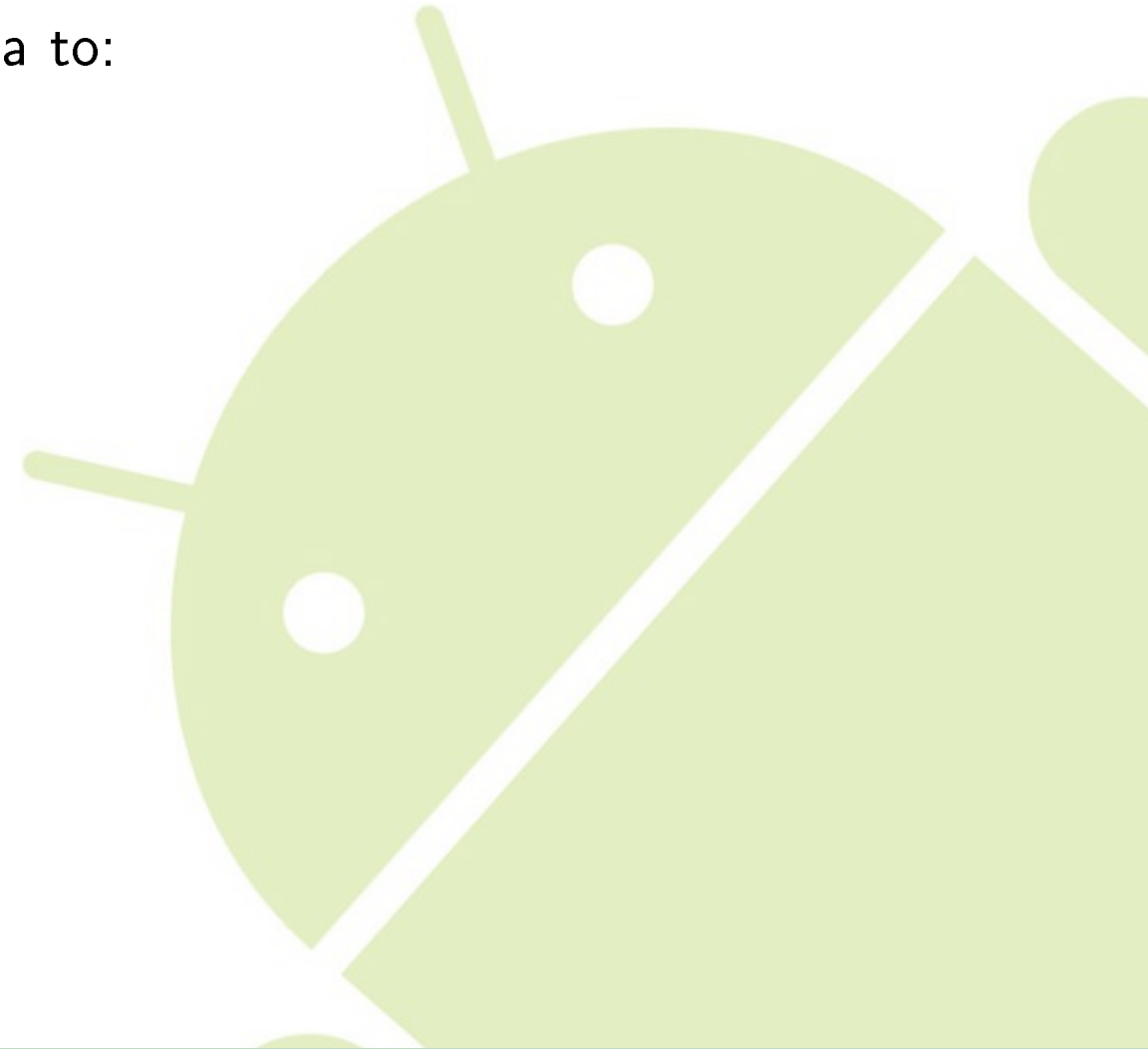
```
class MyClass {
  private final String TAG = getClass()
                               .getSimpleName();

  void doStuff() {
    Log.d(TAG, "doing␣stuff...");  // debug
  }
}
```

Android Log levels

Dostępne poziomy logowania to:

- `Log.v()` - VERBOSE
- `Log.d()` - DEBUG



Android Log levels

Dostępne poziomy logowania to:

- `Log.v()` - VERBOSE
- `Log.d()` - DEBUG
- `Log.i()` - INFO

Android Log levels

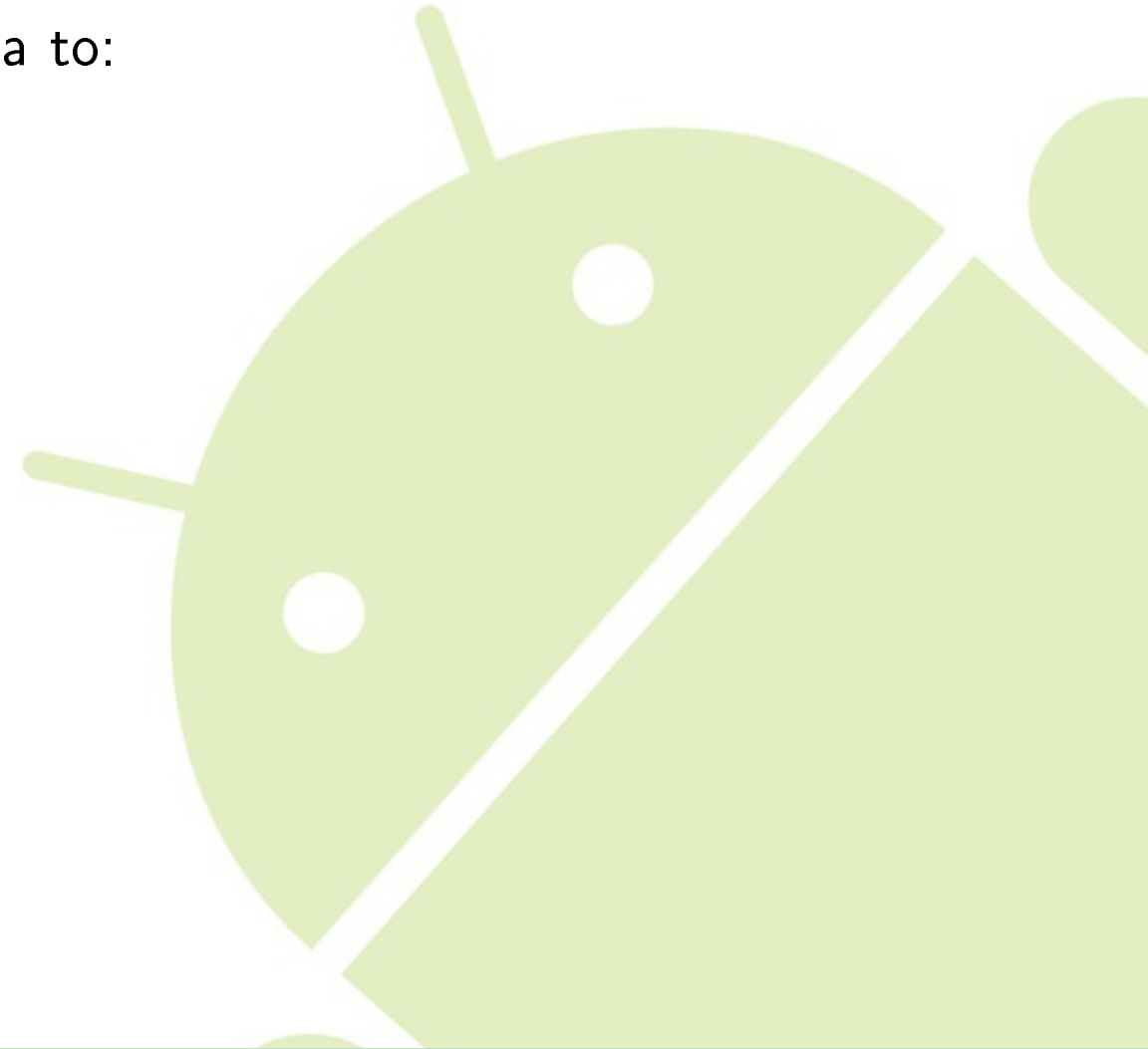
Dostępne poziomy logowania to:

- `Log.v()` - VERBOSE
- `Log.d()` - DEBUG
- `Log.i()` - INFO
- `Log.w()` - WARN

Android Log levels

Dostępne poziomy logowania to:

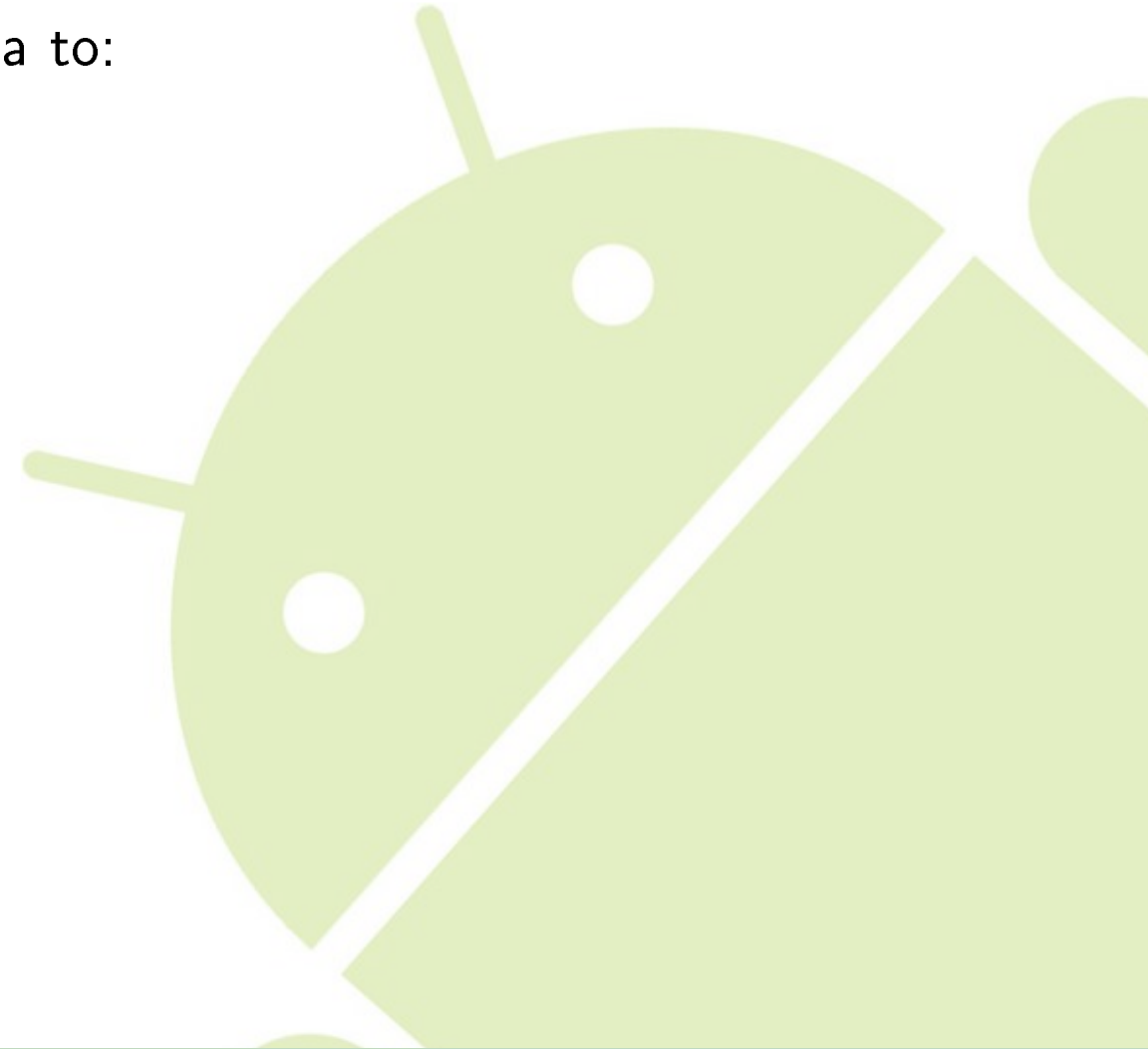
- `Log.v()` - VERBOSE
- `Log.d()` - DEBUG
- `Log.i()` - INFO
- `Log.w()` - WARN
- `Log.e()` - ERROR



Android Log levels

Dostępne poziomy logowania to:

- `Log.v()` - VERBOSE
- `Log.d()` - DEBUG
- `Log.i()` - INFO
- `Log.w()` - WARN
- `Log.e()` - ERROR
- `Log.wtf()`



Android Log levels

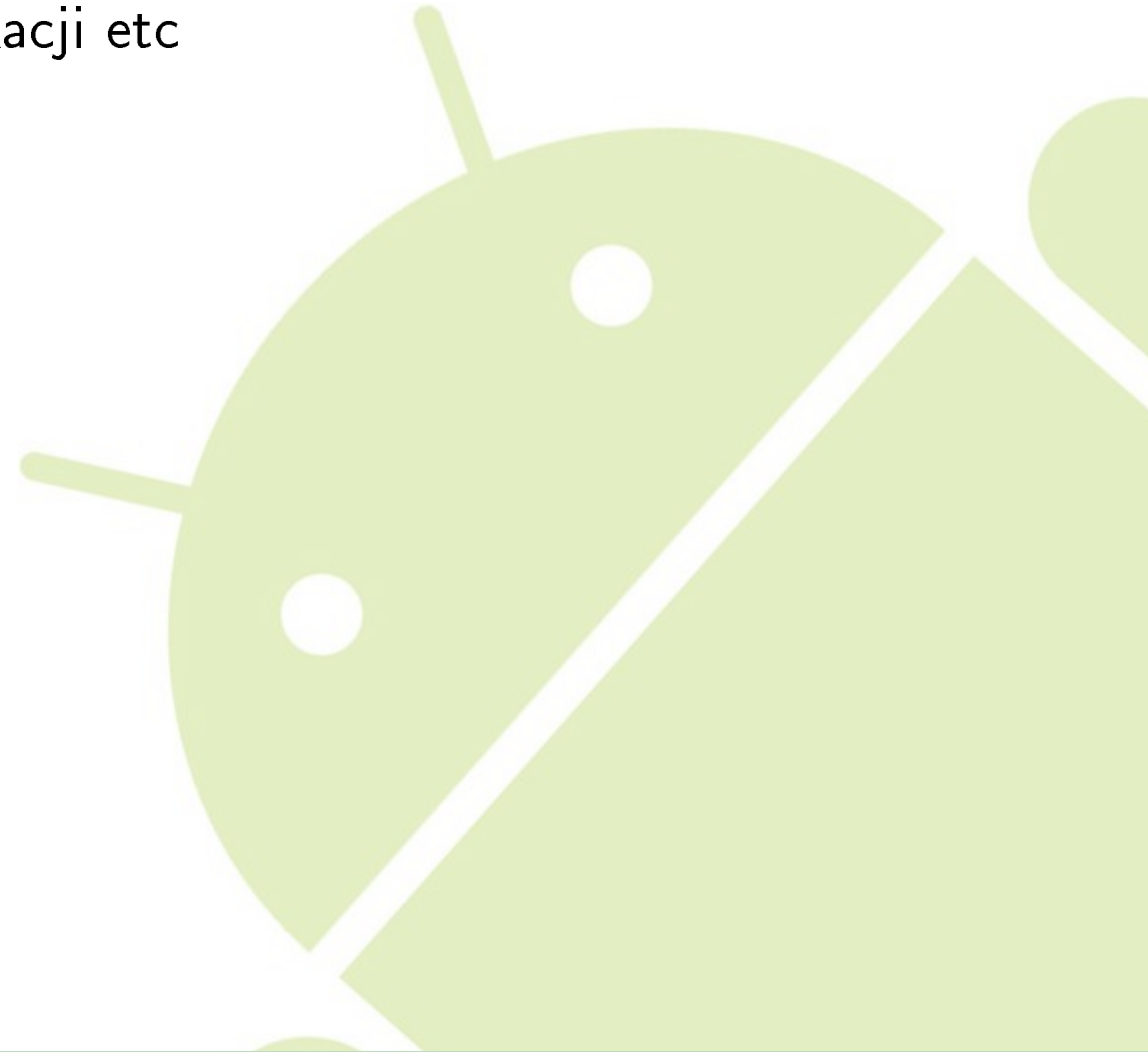
Dostępne poziomy logowania to:

- `Log.v()` - VERBOSE
- `Log.d()` - DEBUG
- `Log.i()` - INFO
- `Log.w()` - WARN
- `Log.e()` - ERROR
- `Log.wtf()` - **W**hat a **T**errible **F**ailure

Data Storage

Sposoby przechowywania danych

- **Shared Preferences** - Prosty KeyValue store, idealny dla prostych ustawień aplikacji etc



Sposoby przechowywania danych

- **Shared Preferences** - Prosty KeyValue store, idealny dla prostych ustawień aplikacji etc
- **Internal Storage** - Zapisywanie plików w swoim formacie na **wewnątrznej pamięci** urządzenia (dobre dla cache obrazków etc)

Sposoby przechowywania danych

- **Shared Preferences** - Prosty KeyValue store, idealny dla prostych ustawień aplikacji etc
- **Internal Storage** - Zapisywanie plików w swoim formacie na **wewnątrznej pamięci** urządzenia (dobre dla cache obrazków etc)
- **External Storage** - Zapisywanie plików w swoim formacie na np. **karcie SD** (dobre dla cache obrazków etc)

Sposoby przechowywania danych

- **Shared Preferences** - Prosty KeyValue store, idealny dla prostych ustawień aplikacji etc
- **Internal Storage** - Zapisywanie plików w swoim formacie na **wewnątrznej pamięci** urządzenia (dobre dla cache obrazków etc)
- **External Storage** - Zapisywanie plików w swoim formacie na np. **karcie SD** (dobre dla cache obrazków etc)
- **SQLite Database** - Zwyczajna instancja bazy danych SQLite - m.in. tym sposobem dostajemy informacje o kontaktach

Sposoby przechowywania danych

- **Shared Preferences** - Prosty KeyValue store, idealny dla prostych ustawień aplikacji etc
- **Internal Storage** - Zapisywanie plików w swoim formacie na **wewnątrznej pamięci** urządzenia (dobre dla cache obrazków etc)
- **External Storage** - Zapisywanie plików w swoim formacie na np. **karcie SD** (dobre dla cache obrazków etc)
- **SQLite Database** - Zwyczajna instancja bazy danych SQLite - m.in. tym sposobem dostajemy informacje o kontaktach
- **Cloud Storage** - Nie trzymamy danych lokalnie, pchamy i pobieramy wszystko z chmurki

Shared Preferences

SharedPreferences - API

Uzyskanie instancji:

```
// zawsze zadziala:
Context ctx = getApplicationContext();

// w Activity lub Service jest latwiej:
// ctx = this;

SharedPreferences prefs = PreferenceManager
                    .getDefaultSharedPreferences(ctx);
```

SharedPreferences - Read API

Przykład **odczytania** zmiennej:

```
// oto jak korzystac z @strings/ w Activity
String key = getString(R.string.key_sound_notif);

String s = preferences.getString(key, "undefined");

// analogicznie dla Integer/Long/Double/StringSet
```

SharedPreferences - Write API

Przykład **zapisania** zmiennej:

```
SharedPreferences.Editor editor = preferences.edit();

editor.putString(key, "new value");
// analogicznie dla Integer/StringSet/Double ...

editor.commit();
```

Shared Preferences

Shared w sensie „wewnątrz **naszej** aplikacji”, nie między wieloma.
SharedPreferences zapisywane są w:

```
/data/data/pl.project13.myapp/shared_prefs
```

Shared Preferences

Shared w sensie „wewnątrz **naszej** aplikacji”, nie między wieloma.
SharedPreferences zapisywane są w:

```
/data/data/pl.project13.myapp/shared_prefs
```

Można się tam dobrać gdy się jest **root**:

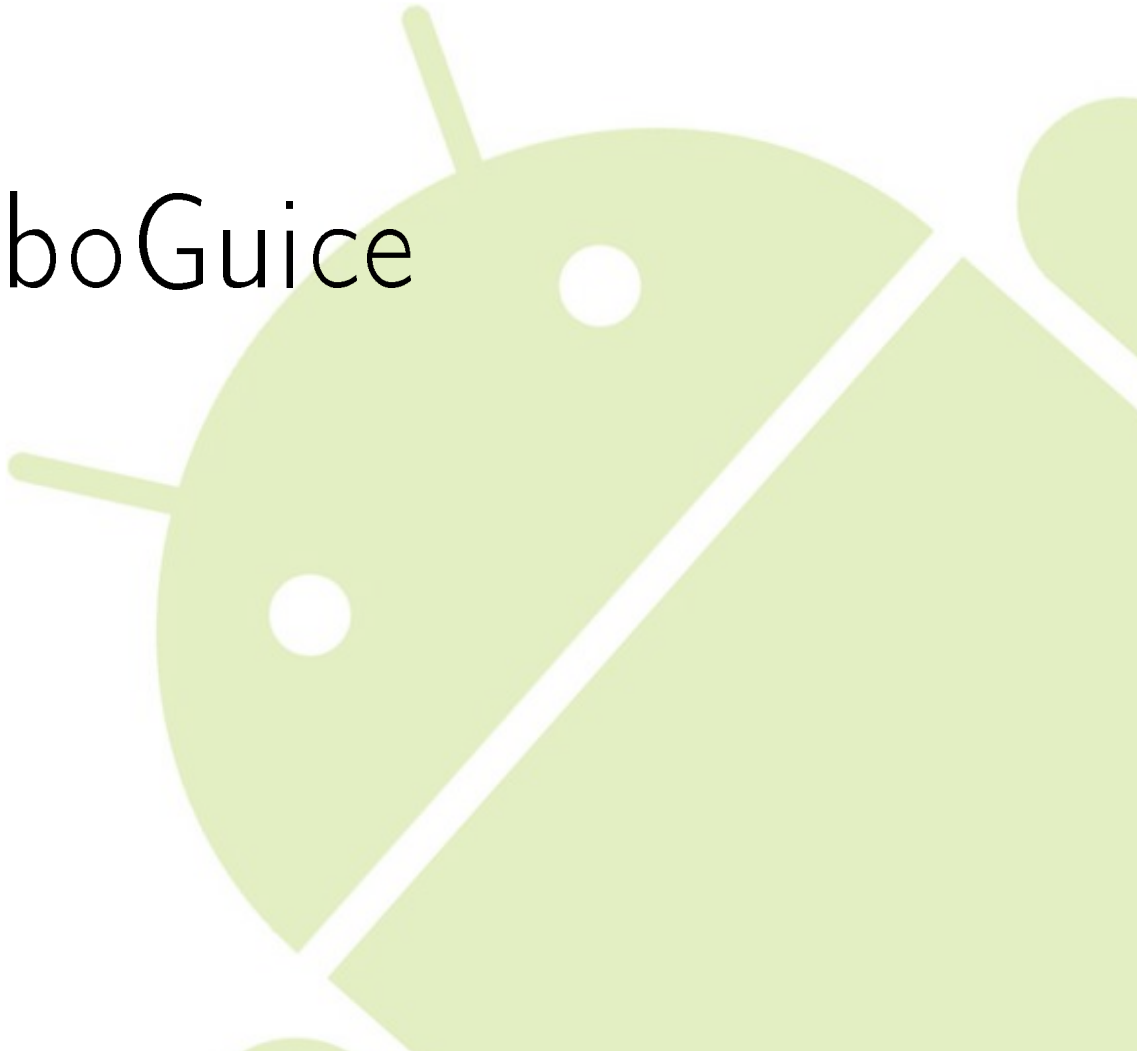
```
$ abd shell
> cat /data/data/pl.project13.myapp/shared_prefs

<map>
 <string name="pass">zomg_its_plain_text</string>
 <!-- ... -->
</map>
```

Security a SharedPreferences

Wniosek jest prosty:
Szyfrujemy ważne rzeczy trzymane gdziekolwiek na komórce.

RoboGuice



Robo Guice - Google Guice for Android

Google Guice = **JSR-330** Dependency Injection for Java

RoboGuice - co zyskujemy?

Przed:

```
class Act extends Activity {
 SharedPreferences prefs;
 EditText mLogin;
 public void onCreate(Bundle savedState) {
  prefs = PreferenceManager
          .getDefaultSharedPreferences(this);
  mLogin = (EditText) findById(R.id.login);
 }
}
```

RoboGuice - co zyskujemy?

Przed:

```
class Act extends Activity {
 SharedPreferences prefs;
 EditText mLogin;
 public void onCreate(Bundle savedState) {
  prefs = PreferenceManager
          .getDefaultSharedPreferences(this);
  mLogin = (EditText) findById(R.id.login);
 }
}
```

Po:

```
class Act extends RoboActivity {
 @Inject SharedPreferences prefs;
 @InjectView(R.id.login) EditText mLogin;
 public void onCreate(Bundle savedState){ /**/ }
}
```

Zapięcie RoboGuice w 4 krokach:

1. własny **App extends RoboApplication** gdzie nadpisujemy **#addApplicationModules**

Zapięcie RoboGuice w 4 krokach:

1. własny **App extends RoboApplication** gdzie nadpisujemy **#addApplicationModules**
2. własny **SzczecinModule extends AbstractAndroidModule**, który dodajemy powyżej

Zapięcie RoboGuice w 4 krokach:

1. własny **App extends RoboApplication** gdzie nadpisujemy **#addApplicationModules**
2. własny **SzczecinModule extends AbstractAndroidModule**, który dodajemy powyżej
3. dodanie `<application android:name=".App"` dla naszej aplikacji w **AndroidManifest.xml**

Zapięcie RoboGuice w 4 krokach:

1. własny **App extends RoboApplication** gdzie nadpisujemy **#addApplicationModules**
2. własny **SzczecinModule extends AbstractAndroidModule**, który dodajemy powyżej
3. dodanie `<application android:name=".App"` dla naszej aplikacji w **AndroidManifest.xml**
4. zamiana `MyActivity extends Activity` na: `MyActivity extends RoboActivity`

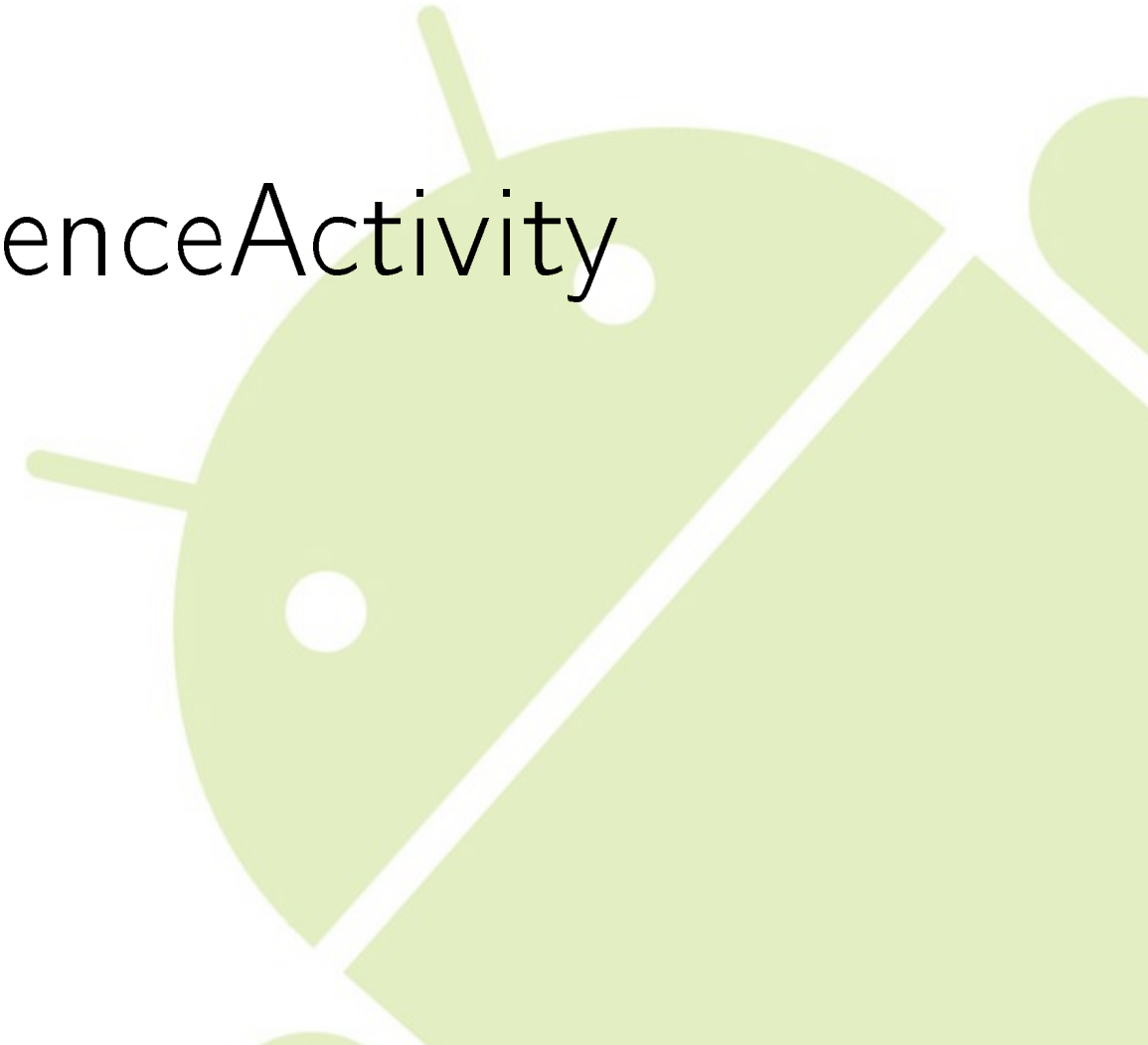
Zadanie: Kroczki do przodu

- podpinamy **RoboGuice**
- zapisujemy **imię użytkownika** w **SharedPreferences**
- podpinamy **res/menu/menu.xml** (1 element menu, o nazwie 'settings') (tip: robi się to w **onCreateOptionsMenu()**)
-

tip:

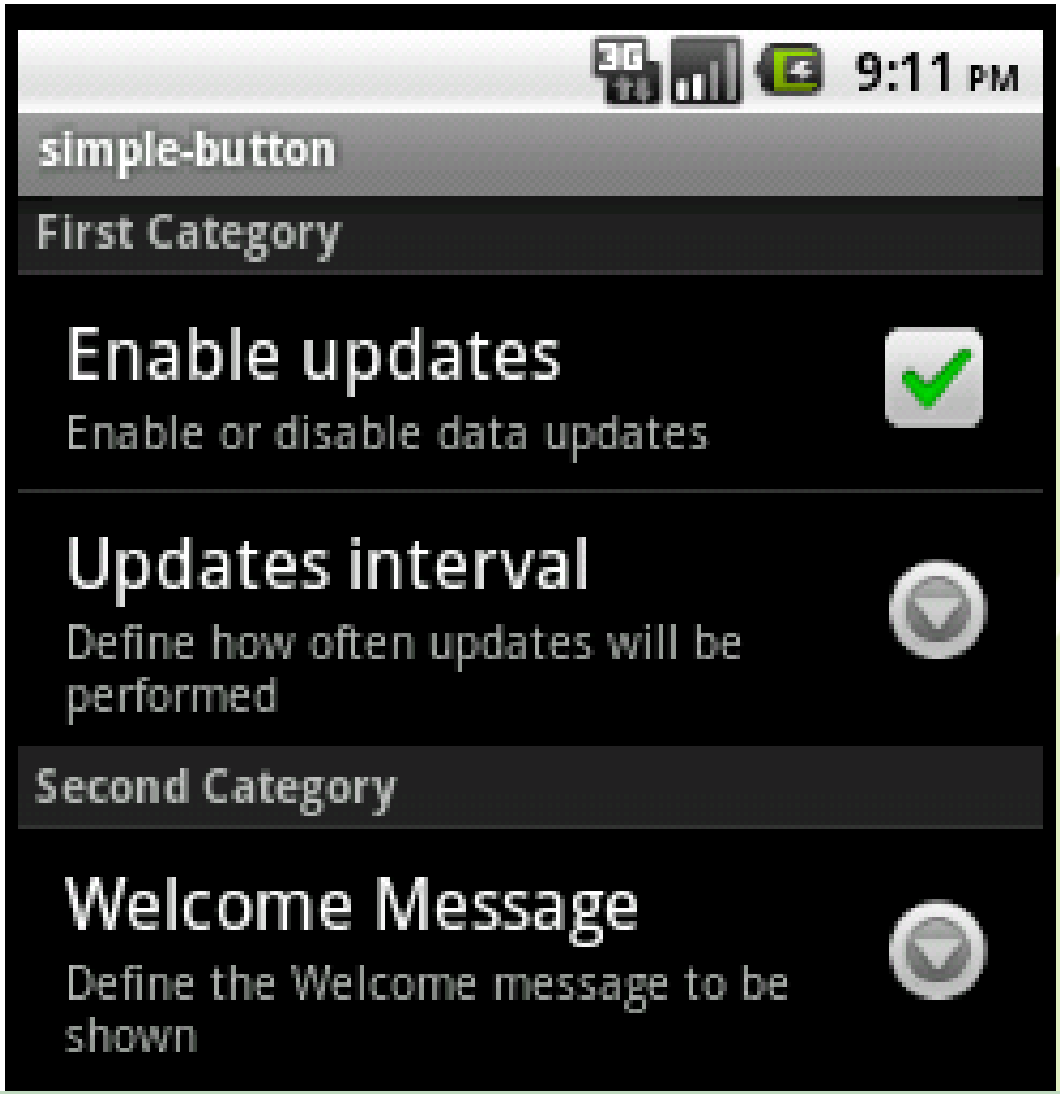
```
bindConstant()
.annotatedWith(SharedPreferencesName.class)
.to("pl.project13");
```

PreferenceActivity



PreferenceActivity

Ręczne edytowanie SharedPreferences szybko robi się nudne...
Wtedy właśnie korzystamy z PreferenceActivity:



/res/**xml/preferences.xml**

```
<PreferenceScreen xmlns:android="http://...">
  <PreferenceCategory
      android:title="Ustawienia Usera"
      android:key="user_preferences">

    <CheckBoxPreference
        android:key="@string/pref_key_notify_user"
        android:summary="Powiadamiaj uzytkownika"
        android:title="Wlacz powiadomienia"
        android:defaultValue="true"
    />

  </PreferenceCategory>
<!-- ... -->
```

/res/**xml/preferences.xml**

```
<!-- ... -->
  <PreferenceCategory
        android:title="Ustawienia pozostale"
        android:key="other_preferences">

    <EditTextPreference
          android:key="@string/pref_key_welcome_msg"
          android:title="Wiadomosc powitalna"
          android:summary="Wiadomosc jaka zostanie
                           powitany uzytkownik"
          android:dialogTitle="Wiadomosc powitalna"
          android:dialogMessage="Wpisz wiadomosc:"
          android:defaultValue="Jak sie masz, " />
  </PreferenceCategory>
</PreferenceScreen>
```

PreferenceActivity - implementacja

W przeciwieństwie do powyższych 2 slajdów xml, tutaj kodu jest malutko:

```
public class SettingsActivity
                              extends PreferenceActivity {

  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    addPreferencesFromResource(R.xml.preferences);
  }
}
```

Jak uruchomić inną Activity?

I w tym miejscu czas poznać: Incencje

Intent

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie. Poprzez intent działają chociażby:

- Otworzenie linka

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie. Poprzez intent działają chociażby:

- Otworzenie linka
- Wysłanie/odebranie SMSa

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie. Poprzez intent działają chociażby:

- Otworzenie linka
- Wysłanie/odebranie SMSa (Intent albo przez SMSManager)

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie.
Poprzez intent działają chociażby:

- Otworzenie linka
- Wysłanie/odebranie SMSa (Intent albo przez SMSManager)
- Uruchomienie usługi (Service)

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie. Poprzez intent działają chociażby:

- Otworzenie linka
- Wysłanie/odebranie SMSa (Intent albo przez SMSManager)
- Uruchomienie usługi (Service)
- Uruchomienie Activity

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie. Poprzez intent działają chociażby:

- Otworzenie linka
- Wysłanie/odebranie SMSa (Intent albo przez SMSManager)
- Uruchomienie usługi (Service)
- Uruchomienie Activity
- Przekazanie danych innej części aplikacji

Intent - czyli „Intencja wykonania X”

Intent'y wykorzystywane są bardzo bardzo intensywnie w androidzie. Poprzez intent działają chociażby:

- Otworzenie linka
- Wysłanie/odebranie SMSa (Intent albo przez SMSManager)
- Uruchomienie usługi (Service)
- Uruchomienie Activity
- Przekazanie danych innej części aplikacji
- nasłuchiwanie na „**system-wide**” zdarzenia

Intent - czyl „Intencja wykonania X”

Intent = „Chciałbym zrobić X”.

Intent - czyl „Intencja wykonania X”

Intent = „Chciałbym zrobić X”.

Chciałbym otworzyć przeglądarkę www:

```
String action = Intent.ACTION_VIEW;
Uri uri = Uri.parse("http://www.geecon.org")

Intent viewIntent = new Intent(action, uri);
startActivity(viewIntent); // uruchom
```

Intent - przykłady cd.

Chcę wysłać SMSa, przy pomocy **jakiejś aplikacji**,
która potrafi się tym zająć.

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);

sendIntent.putExtra("sms_body", "Halo Szczecin!");
sendIntent.setType("vnd.android-dir/mms-sms");

startActivity(sendIntent);
```

Intent - przykłady cd.

Chcę wysłać SMSa, przy pomocy **jakiejś aplikacji**,
która potrafi się tym zająć.

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);

sendIntent.putExtra("sms_body", "Halo Szczecin!");
sendIntent.setType("vnd.android-dir/mms-sms");

startActivity(sendIntent);
```

SMSy również można wysyłać przy pomocy **SMSManager**.

Intent-Filter - „Słuchacze”

Aby „słuachć” na globalne Intent trzeba dodać w **AndroidManifest.xml**:

```
<application ...>
  <receiver android:name=".SmsReceiver">
    <intent-filter>
      <action android:name=
        "android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
  </receiver>
</application>
```

Intent-Filter - „Słuchacze”

Aby „słuachć” na globalne Intent trzeba dodać w **AndroidManifest.xml**:

```
<application ...>
  <receiver android:name=".SmsReceiver">
    <intent-filter>
      <action android:name=
        "android.provider.Telephony.SMS_RECEIVED" />
    </intent-filter>
  </receiver>
</application>
```

MAIN oraz **LAUNCHER** dla Activity, definiujące główne Activity naszej aplikacji również rejestrujemy przez Intent-Filtry!

IntentReciever - przykład dla SMS_RECEIVED

```
public class SmsReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent)
    Bundle bundle = intent.getExtras();
    if(bundle == null) return;

    Object[] pdus = (Object[]) bundle.get("pdus");

    for (Object aPdus : pdus) {
      SmsMessage msg;
      msg = SmsMessage.createFromPdu((byte[]) aPdus);

      String from = msg.getOriginatingAddress();
      String body = msg.getMessageBody().toString();
      Log.i(TAG, format(%s: %s, from, body));
    }
  }
}
```

Intent - fun facts

- **Intent** może być obsłużony przez dowolną aplikację - nie tylko „nasze”

Intent - fun facts

- **Intent** może być obsłużony przez dowolną aplikację - nie tylko „nasze”
- W przypadku gdy **Intent** natrafi 2+ „Recievery”, pyta użytkownika którego ma użyć. Przykład:

Intent - fun facts

- **Intent** może być obsłużony przez dowolną aplikację - nie tylko „nasze”
- W przypadku gdy **Intent** natrafi 2+ „Recievery”, pyta użytkownika którego ma użyć. Przykład:
  I: „Otwórz ten link.”

Intent - fun facts

- **Intent** może być obsłużony przez dowolną aplikację - nie tylko „nasze”
- W przypadku gdy **Intent** natrafi 2+ „Recievery”, pyta użytkownika którego ma użyć. Przykład:
  I: „Otwórz ten link.”
  A: „W Operze czy w Firefoxie?”

Intent - fun facts

- **Intent** może być obsłużony przez dowolną aplikację - nie tylko „nasze”
- W przypadku gdy **Intent** natrafi 2+ „Recievery”, pyta użytkownika którego ma użyć. Przykład:
  I: „Otwórz ten link.”
  A: „W Operze czy w Firefoxie?”
- **Intent** może nieść ze sobą masę dodatkowych informacji oraz flag. Vide metody klasy Intent.