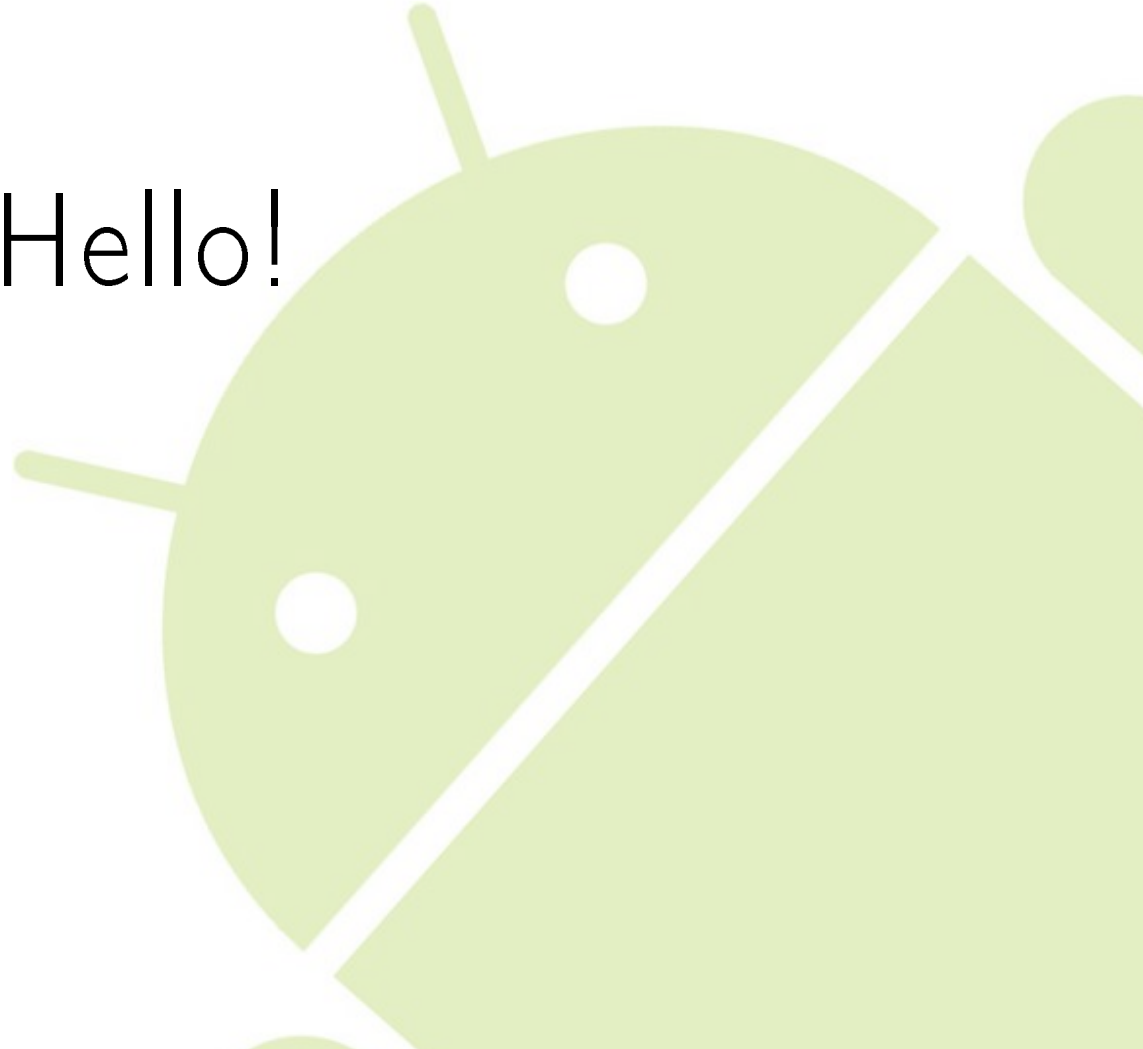
Android @ Szczecin 2011

Konrad Malawski
konrad.malawski@java.pl

4 sierpnia 2011

Hello!



Let's move the Java world!

GeeCON

lunar logic
polska

@ktosopl

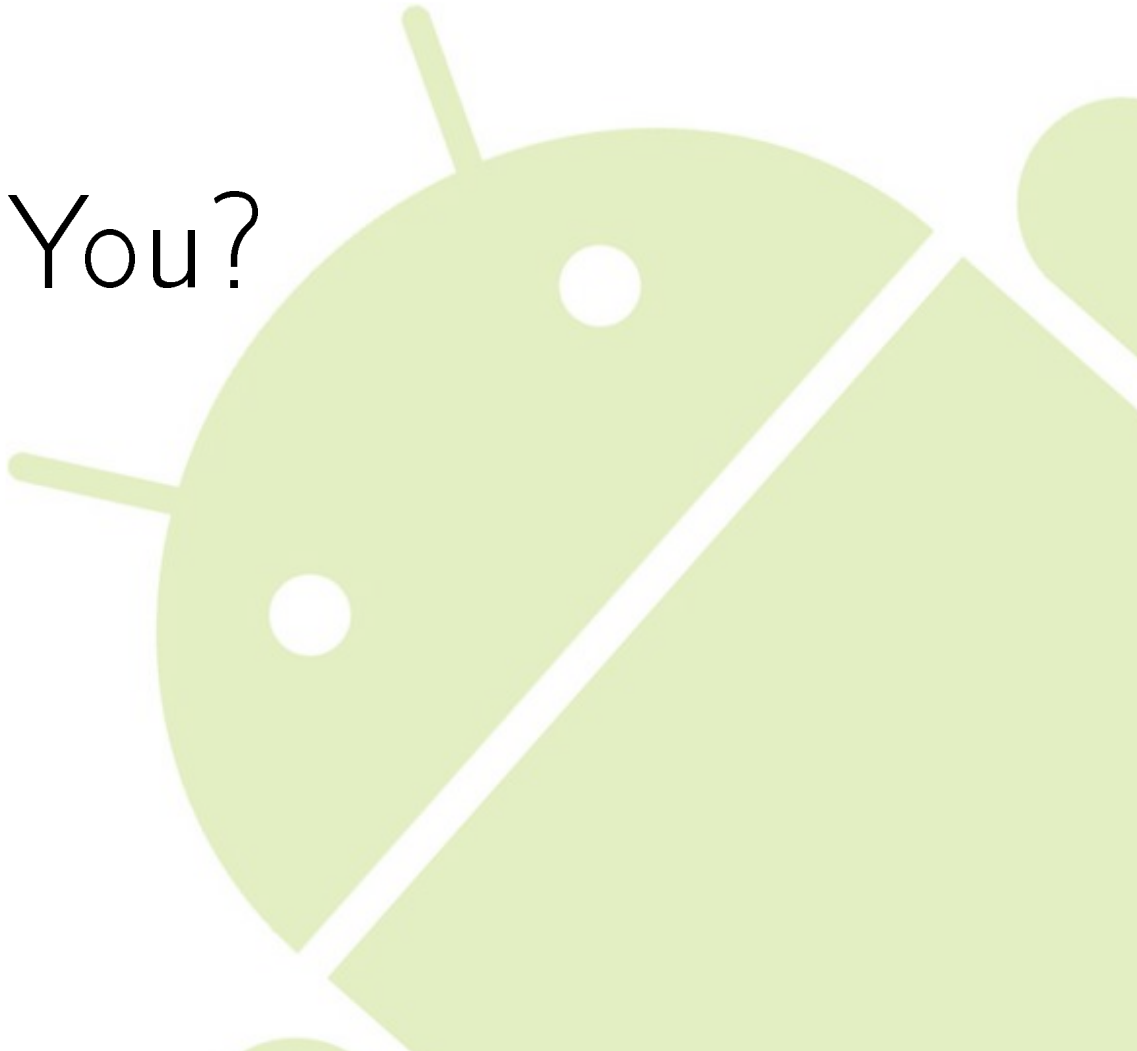
blog.project13.pl

github.com/ktoso

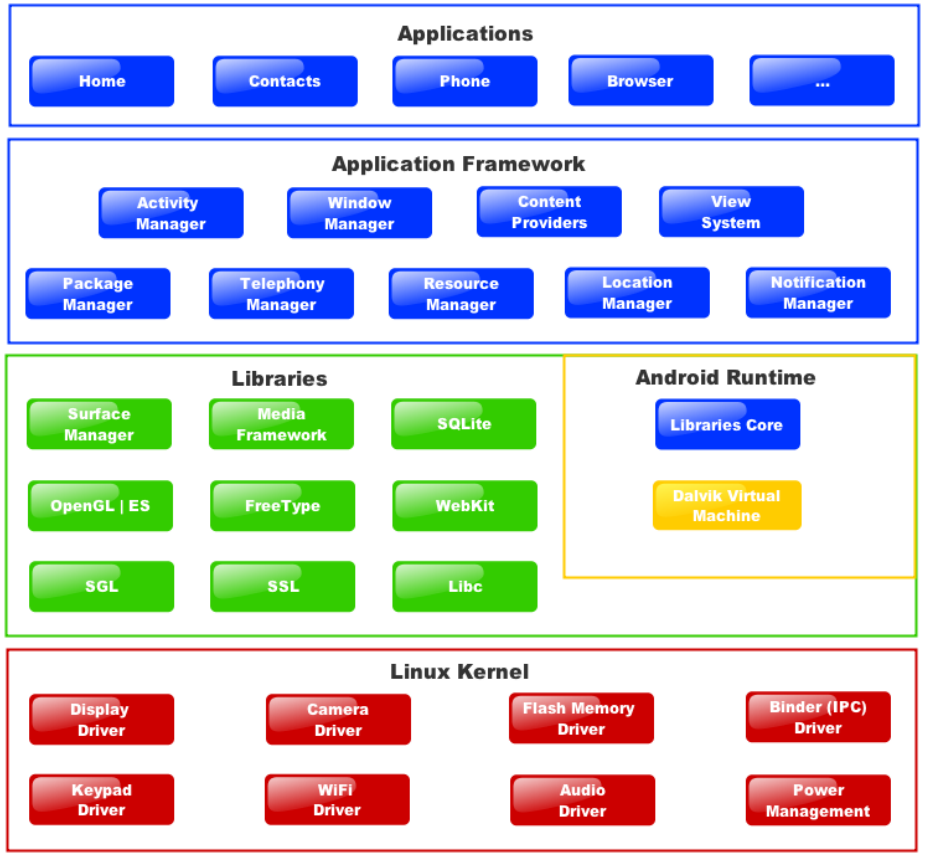
CODERETREAT

KRAKÓW - APRIL 2nd 2011 - START: 08:00

You?



Architektura



Rysunek: Architektura systemu Android

Słowa na dziś:

- Application
- Activity
- View
- Service
- Intent
- _ _ _ _ _ _ _ _Manager

The „Glue”: AndroidManifest.xml

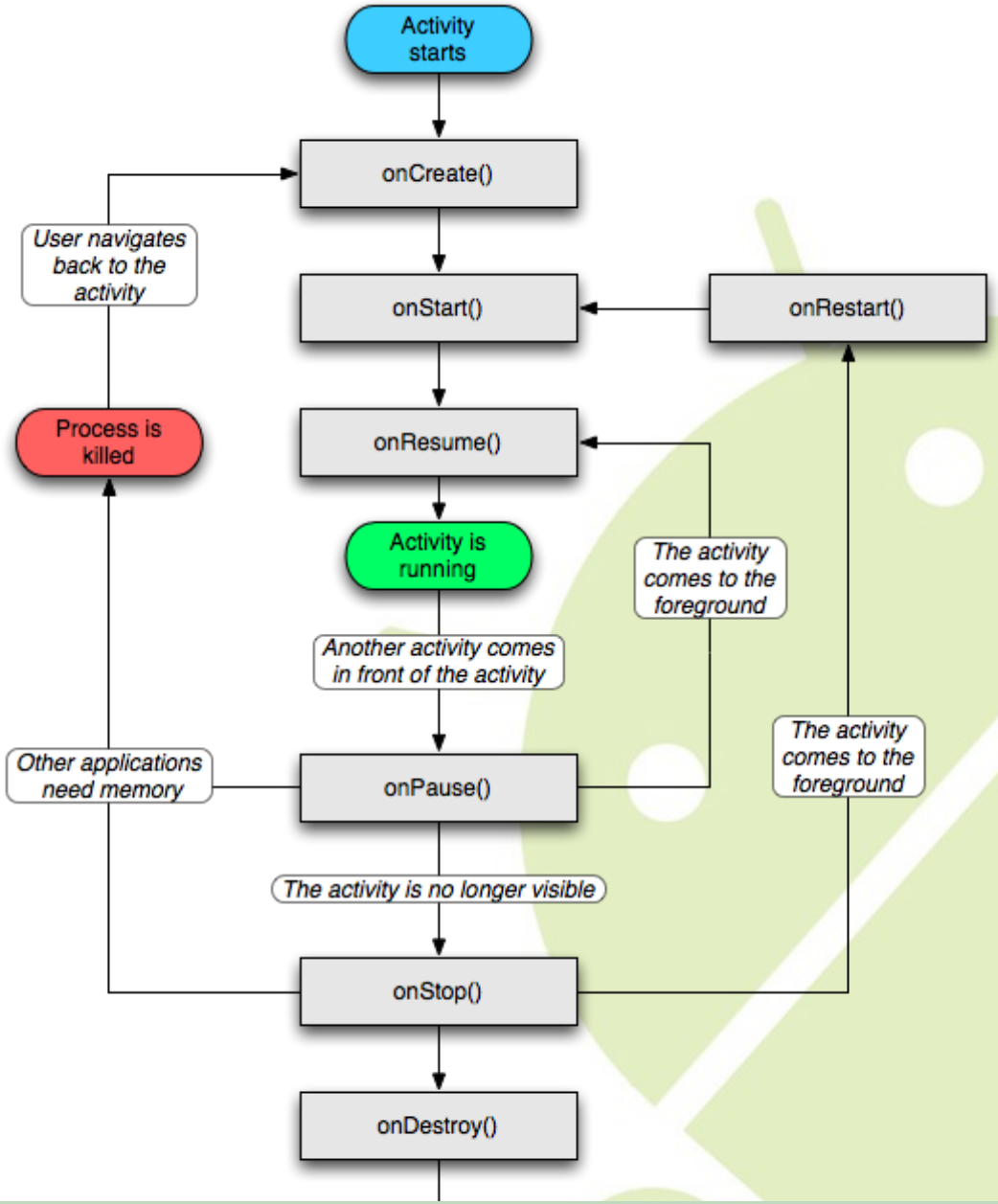
AndroidManifest.xml

```
<manifest xmlns:android="http://..."
      package="pl.project13.android.github.notify"
      android:versionCode="1"
      android:versionName="1.0">

  <application android:name=".guice.GitHubNotifyApplicat
               android:label="@string/app_name"
               android:icon="@drawable/icon"
               android:debuggable="true"
               android:hardwareAccelerated="true">

    <activity android:name=".activity.WatchPreferencesA
              android:icon="@drawable/icon"
              android:label="GitHub Notify">
      <intent-filter>
        <action android:name="android.intent.action.MAIN
        <category android:name="android.intent.category.
      </intent-filter>
    </activity>
```

AndroidManifest.xml

```
    <service android:name=".service.NewCommitNotifierSer
  </application>

  <!-- PERMISSIONS -->
  <uses-sdk android:minSdkVersion="6"
            android:targetSdkVersion="11"/>

  <uses-permission android:name="android.permission.INTE
  <uses-permission android:name="android.permission.ACCES
  <uses-permission android:name="android.permission.RECE
</manifest>
```

Activity lifecycle



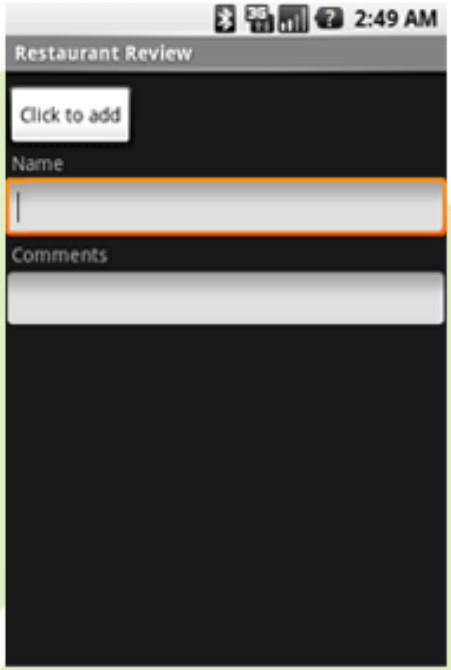
Activity

Wołany gdy Activity jest tworzona po raz pierwszy.

```
public class MainActivity extends Activity {
  public void onCreate(Bundle savedState) {
    /**/
  }
}
```

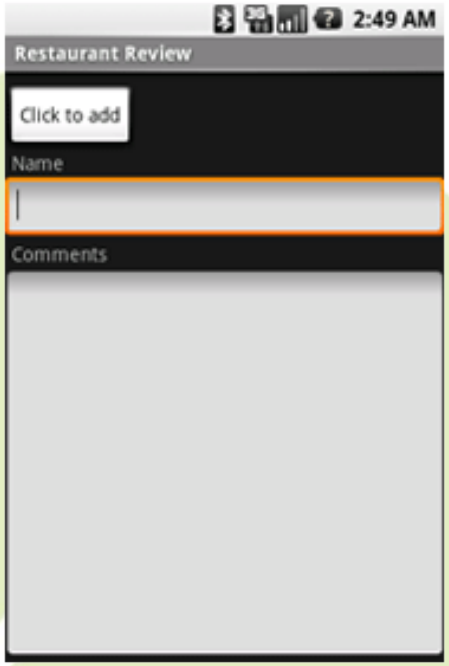
LinearLayout + EditText z wrap_content

```
<LinearLayout
 android:id="@+id/root_layout"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical">
  <!-- ... -->

  <TextView android:id="..." ... />
  <EditText android:id="..."
   android:layout_width="fill_parent"
   android:layout_height="wrap_content"
  />
</LinearLayout>
```



LinearLayout + EditText z fill_parent

```
<LinearLayout
 android:id="@+id/root_layout"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical">
  <!-- ... -->

  <TextView android:id="..." ... />
  <EditText android:id="..."
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
  />
</LinearLayout>
```



Panowanie nad Layoutami

- Istnieje jeszcze **match_parent**, pojawił się w nowszych wersjach Android, jest analogiczny do fill_parent.
- W przypadku gdy chcemy ustalać "jaką część widoku ma zajmować pewien element", korzystamy z **android:layout_weight**, przyjmuje on liczby *(default = 1)*.

Ważne ViewGroups:

- **FrameLayout** - Layout dla tylko jednego elementu, najprostszy
- **LinearLayout** - Wystarczający dla prostych widoków, kombinowanie kilku LinearLayout daje przyjemne efekty
- **ListView** - Sam dba o scrollowanie
- **TabHost** - Może mieć zakładki

Ważne ViewGroups:

- **FrameLayout** - Layout dla tylko jednego elementu, najprostszy
- **LinearLayout** - Wystarczający dla prostych widoków, kombinowanie kilku LinearLayout daje przyjemne efekty
- **ListView** - Sam dba o scrollowanie
- **TabHost** - Może mieć zakładki
- **Spinner**, **Gallery**, **GridView**, **RelativeLayout**, **ScrollView**, **SurfaceView**, **TableLayout**, **ViewFlipper**, **ViewSwitcher**...

R, as in Resource

R, as in Resource

```
package pl.project13;

public final class R {
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int login=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040000;
    }
}
```

R, as in Resource

Dodanie elementu id **wewnątrz widoku**, poprzez **+@id/**

```
<EditText android:id="@+id/login" ... />
```

R, as in Resource

Dodanie elementu id **wewnątrz widoku**, poprzez **+@id/**

```
<EditText android:id="@+id/login" ... />
```

Spowoduje wygenerowanie pola w klasie **R**:

```
public final class R {
  public static final class id {
    public static final int login = 0x7f050000;
  }
}
```

R, as in Resource

Dodanie elementu id **wewnątrz widoku**, poprzez **+@id/**

```
<EditText android:id="@+id/login" ... />
```

Spowoduje wygenerowanie pola w klasie **R**:

```
public final class R {
  public static final class id {
    public static final int login = 0x7f050000;
  }
}
```

A skorzystamy z niego w np. **Activity**:

```
EditText mLogin = findById(pl.project13.R.id.login);
```

Strings as Resources

```
<TextView android:layout_height="fill_parent"
          android:layout_width="fill_parent"
          android:text="Witaj swiecie!"
        />
```

Mieszanie treści z layoutem jest złym pomysłem.

Strings as Resources

```
<TextView android:layout_height="fill_parent"
          android:layout_width="fill_parent"
          android:text="@string/hello_world"
        />
```

```
<resources>
  <string name="hello_world">Witaj swiecie!</string>
</resources>
```

(Tip: W intellij piszemy *@string/hello_world* a następnie ALT+ENTER, aby utworzyć zasób w odpowiednim miejscu.)

Proste I18N (i nie tylko) dzięki klasyfikatorom

res/values-pl/strings.xml

```
<resources>
  <string name="hello_world">Witaj swiecie!</string>
</resources>
```

res/values-en/strings.xml

```
<resources>
  <string name="hello_world">Hello world!</string>
</resources>
```

R, tips and tricks

- nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązywane jest do wedle 'na czym wołany jest findById'.

R, tips and tricks

- nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązywane jest do wedle 'na czym wołany jest findById'.
- Korzystamy raczej z 'notacji_z_podkresleniami_tutaj'

R, tips and tricks

- nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązywane jest do wedle 'na czym wołany jest findById'.
- Korzystamy raczej z 'notacji_z_podkresleniami_tutaj'
- W kontekście gdzie nie masz **findById**, skorzystaj z **android.content.res.Resources.getSystem().get_____()**

Zadanie: Hello Szczecin!

- Prosty widok z przyciskiem oraz tekstem oraz polem tekstowym na czyjeś imię
- Ma być możliwe podanie swojego imienia, wtedy text po wciśnięciu przycisku ma wyglądać „Hello _____!”
- tip: korzystamy z **res/values/strings.xml** oraz **res/layouts/main.xml**
- tip: Przydadzą się widżety: TextView, Button, EditText

Zadanie: Hello Szczecin!

- Prosty widok z przyciskiem oraz tekstem oraz polem tekstowym na czyjeś imię
- Ma być możliwe podanie swojego imienia, wtedy text po wciśnięciu przycisku ma wyglądać „Hello _____!”
- tip: korzystamy z **res/values/strings.xml** oraz **res/layouts/main.xml**
- tip: Przydadzą się widżety: TextView, Button, EditText
- **Uwaga! Skorzystamy z Robolectric to *TDD** aplikacji Androiowej!**

* TDD - Test Driven Development

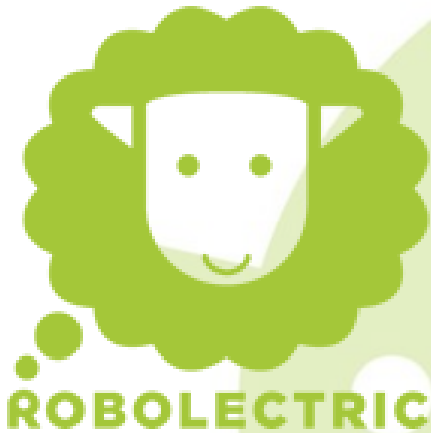
Tak tak, ponieważ zwyczajny hello world byłby nudny! :-)

Testy są już dla Was przygotowane.

Testowanie a sprawa Androida

Jest pewien problem z 'zwyczajnym testowaniem' aplikacji androidowych... Wcześniej czy później wpadnie się na:

```
java.lang.RuntimeException: Stub!
```



http://pivotal.github.com/robolectric/index.html

Robolectric zastępuje implementacje rzucające te wyjątki domyślnymi (`return 0 / null`).

Plain Old Testing

Jakby ktoś faktycznie chciał rzeźbić bardziej niskopoziomowo swoje testy, oto jak to zrobić:

`http://developer.android.com/resources/tutorials/testing/activity_test.html`

Zadanie: Hello Szczecin!

- Prosty widok z przyciskiem oraz tekstem oraz polem tekstowym na czyjeś imię
- Ma być możliwe podanie swojego imienia, wtedy text po wciśnięciu przycisku ma wyglądać „Hello _____!”
- tip: korzystamy z **res/values/strings.xml** oraz **res/layouts/main.xml**
- tip: Przydadzą się widżety: TextView, Button, EditText **Skorzystamy z Robolectric to *TDD** aplikacji Androiowej!**

```
git clone git@github.com:ktoso/szczecin-android.git
cd szczecin-android/projects/simple-button
```