

# Android @ Szczecin 2011

Konrad Malawski  
konrad.malawski@java.pl

1 sierpnia 2011



**lunar logic**  
polska



@ktosopl

[github.com/ktoso](https://github.com/ktoso)

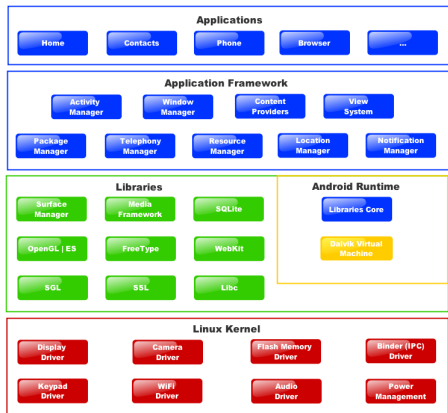


[blog.project13.pl](http://blog.project13.pl)



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

# Architektura



Rysunek: Architektura systemu Android

## Słowa na dziś:

- ▶ Application
- ▶ Activity
- ▶ View
- ▶ Service
- ▶ Intent
- ▶ \_\_\_\_\_ Manager

# Przykłady Managerów

- ▶ **NotificationManager** - dla powiadomień



# Przykłady Managerów

- ▶ **NotificationManager** - dla powiadomień
- ▶ **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)

# Przykłady Managerów

- ▶ **NotificationManager** - dla powiadomień
- ▶ **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- ▶ **AlarmManager** - dla alarmów



# Przykłady Managerów

- ▶ **NotificationManager** - dla powiadomień
- ▶ **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- ▶ **AlarmManager** - dla alarmów
- ▶ **BatteryManager** - stan baterii



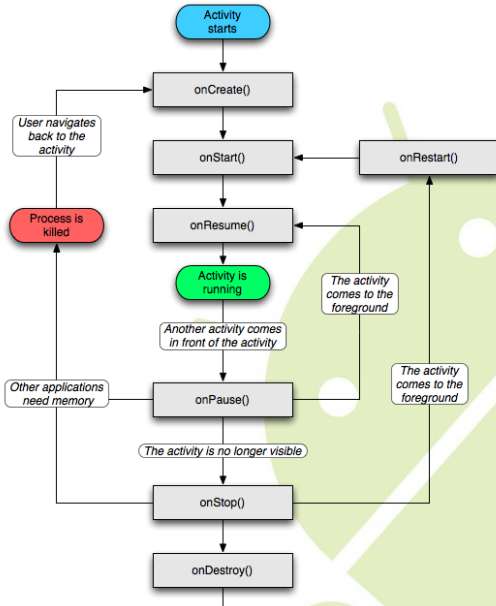
# Przykłady Managerów

- ▶ **NotificationManager** - dla powiadomień
- ▶ **AccountManager** - dla kont użytkownika, np. konto skype / gmail (system-wide)
- ▶ **AlarmManager** - dla alarmów
- ▶ **BatteryManager** - stan baterii
- ▶ **ClipboardManager**, **ConnectivityManager**, **LocationManager**, **FragmentManager**, **DownloadManager**

# Good News, nie będzie tsunami wyliczeń nazw klas

Obiecuję, że nie będzie już slajdów takich jak poprzedni :-)  
**Konkrety, obrazki, kodzenie!**

## Activity lifecycle



# Activity

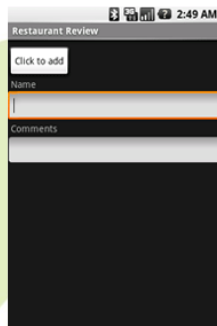
Wołany gdy Activity jest tworzona po raz pierwszy.

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

## LinearLayout + EditText z wrap\_content

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

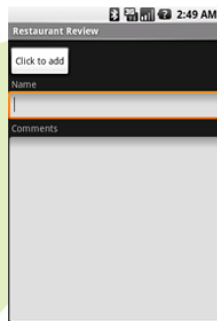
    <TextView android:id="..." ... />
    <EditText android:id="..."
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```



## LinearLayout + TextView 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

## 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 = findViewById(pl.project13.R.id.login);
```

# Strings as Resources

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

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. Rozwiązane jest do wedle 'na czym wołany jest findViewById'.

## R, tips and tricks

- ▶ nazwy wykorzystywane dla np. ID **nie muszą** być unikalne, @+id/login raz może oznaczać ten EditText a raz TextView. Rozwiązane jest do wedle 'na czym wołany jest findViewById'.
- ▶ 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. Rozwiązane jest do wedle 'na czym wołany jest findViewById'.
- ▶ Korzystamy raczej z 'notacji\_z\_podkreśleniami\_tutaj'
- ▶ W kontekście gdzie nie masz **findViewById**, 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](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
```

# 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:

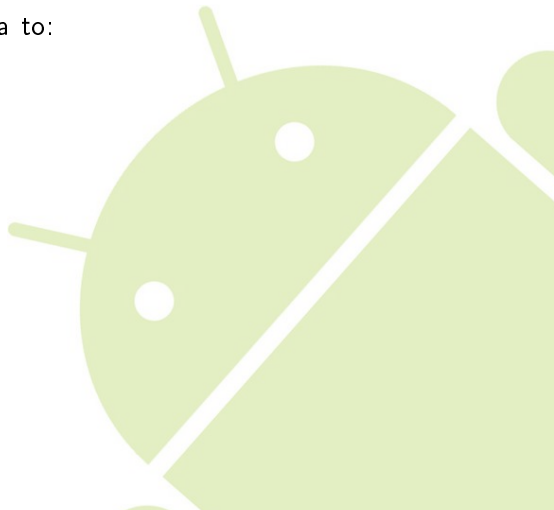
- ▶ `Log.v()` - VERBOSE
- ▶ `Log.d()` - DEBUG
- ▶ `Log.i()` - INFO
- ▶ `Log.w()` - WARN



# 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



# 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 **wewnętrznej 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 **wewnętrznej 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 **wewnętrznej 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 **wewnętrznej 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 korzystać 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 dostać 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.

# 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 savedInstanceState) {  
        prefs = PreferenceManager  
            .getDefaultSharedPreferences(this);  
        mLogin = (EditText) findViewById(R.id.login);  
    }  
}
```

# RoboGuice - co zyskujemy?

Przed:

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

Po:

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

# 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`

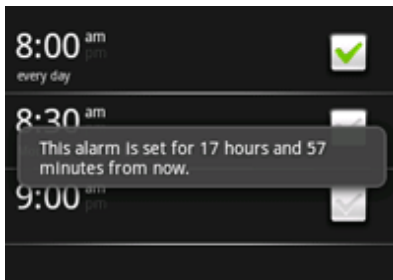
# Zadanie: Kroczki do przodu

- ▶ 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");
```

# Pyszne tosty z masłem (android.widget.Toast)



Przykład użycia:

```
Toast.makeText(getApplicationContext(),  
    "Halo_Szczecin!",  
    Toast.LENGTH_LONG)  
    .show();
```

## Co więcej potrafi Toast?

```
Toast t = Toast.makeText(this, txt, LENGTH_SHORT);
```

## Co więcej potrafi Toast?

```
Toast t = Toast.makeText(this , txt , LENGTH_SHORT);
```

Można mu zmienić pozycję:

```
t.setGravity( Gravity.TOP | Gravity.LEFT , 0 , 0);
```

## Co więcej potrafi Toast?

```
Toast t = Toast.makeText(this , txt , LENGTH_SHORT);
```

Można mu zmienić pozycję:

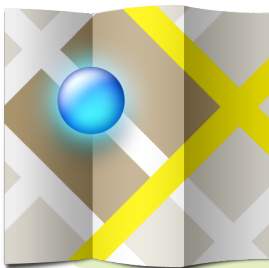
```
t.setGravity( Gravity.TOP| Gravity.LEFT , 0 , 0);
```

lub podmienić widok:

```
View customView = findViewById(R.id.custom_view);  
/**/  
t.setView(customView)
```



# Google Maps



Istnieje pewien "problem" z Google Maps oraz niektórymi innymi API.

**Nie są one dostępne bez odpowiedniego klucza oraz podpisania swojej aplikacji!**

# MapsAPI key sign-up

Rejestrujemy się po klucz na:

<http://code.google.com/intl/pl-PL/android/maps-api-signup.html>

BitLy: **<http://bit.ly/mapsapiandroid>**

## Zdobywanie MD5 klucza 'debug'

```
keytool -list -alias androiddebugkey \  
-keystore <path_to_debug_keystore>.keystore \  
-storepass android -keypass android
```

# Zdobywanie Md5 klucza 'release'

```
keytool -list -keystore /android.keystore
```

```
Keystore type: JKS
```

```
Keystore provider: SUN
```

```
Your keystore contains 1 entry
```

```
android-key, Jul 3, 2011, PrivateKeyEntry,
```

```
Certificate fingerprint (MD5): AA:AA:AA:AA...
```

# Oto co dostaniemy:

Twój klucz to:

```
OX8wqBZbEMsNFtsoHXzufZdWTKB1cUvTWq2OfEg
```

Klucz jest przeznaczony dla wszystkich aplikacji podpisanych Twoim certyfikatem, którego „odciskiem pal

```
A2:D6:8F:12:A6:81:CA:C1:1B:8F:78:8A:E4:FB:A7:FC
```

Poniżej przedstawiono przykład układu xml, który ułatwi rozpoczęcie przygody z mapami:

```
<com.google.android.maps.MapView  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:apiKey="OX8wqBZbEMsNFtsoHXzufZdWTKB1cUvTWq2OfEg"  
/>
```

Więcej informacji znajduje się w [dokumentacji interfejsu API](#).

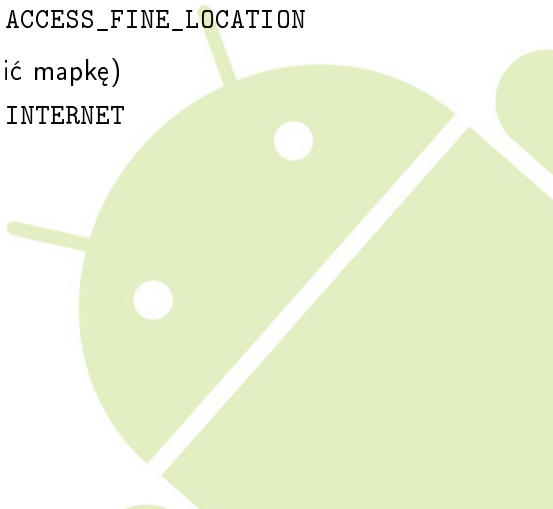
# Permissions

W tym przypadku interesują następujące `<uses-permission/>`:

- ▶ `android.permission.ACCESS_COARSE_LOCATION`
- ▶ `android.permission.ACCESS_FINE_LOCATION`

oraz (skoro chcemy wyświetlić mapkę)

- ▶ `android.permission.INTERNET`



# Permissions

W tym przypadku interesują następujące `<uses-permission/>`:

- ▶ `android.permission.ACCESS_COARSE_LOCATION`
- ▶ `android.permission.ACCESS_FINE_LOCATION`

oraz (skoro chcemy wyświetlić mapkę)

- ▶ `android.permission.INTERNET`

Dodatkowo jeszcze deklarujemy wykorzystanie biblioteki maps:

```
<uses-library android:name="com.google.android.maps" />
```

## Uwaga!

```
<application>  
  <uses-library/>  
</application>  
<uses-permission/>
```

# Co na pewno się przyda?

- ▶ **LocationManager**
- ▶ **MapView**
- ▶ bardzo wygodny jest **MapActivity**
- ▶ tip: dostępny jest **GPS** i **NETWORK** location provider



# Zadanie: Google Maps App

- ▶ mapka, wycentrowana na obecnym położeniu telefonu
- ▶ podczas odświeżenia lokalizacji ma pojawiać się Toast z nową lokalizacją (oraz recentrujemy mapkę)
- ▶ obecne położenie ma być zaznaczone markerem:  
**<http://bit.ly/gmapmark>**
- ▶ dodatkowe: w przypadku oddalenia się od miejsca X (dowolne) należy odpalić **'alarm'**
- ▶ dodatkowe: zaskocz nas czymś!