

# Moar\* Fun with Views

\* sic

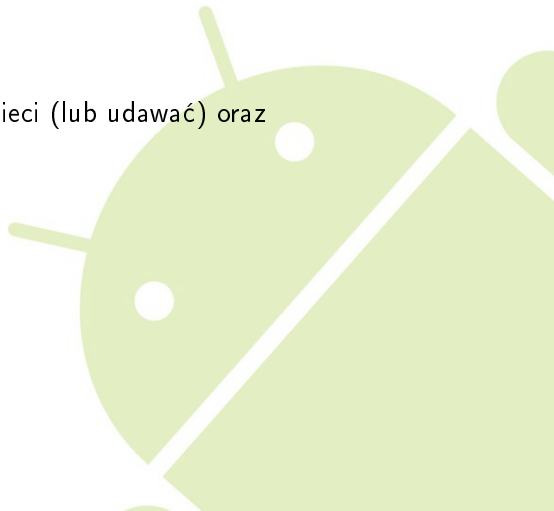
# Zmierzamy w kierunku kolejnego Activity

- Dodamy nowe Activity



# Zmierzamy w kierunku kolejnego Activity

- ▶ Dodamy nowe Activity
- ▶ Będzie pobierać coś z sieci (lub udawać) oraz



# Zmierzamy w kierunku kolejnego Activity

- ▶ Dodamy nowe Activity
- ▶ Będzie pobierać coś z sieci (lub udawać) oraz
- ▶ hint: przyda się jakiś progress bar etc

# Zmierzamy w kierunku kolejnego Activity

- ▶ Dodamy nowe Activity
- ▶ Będzie pobierać coś z sieci (lub udawać) oraz
- ▶ hint: przyda się jakiś progress bar etc
- ▶ utworzy z tych danych ListView

# Zmierzamy w kierunku kolejnego Activity

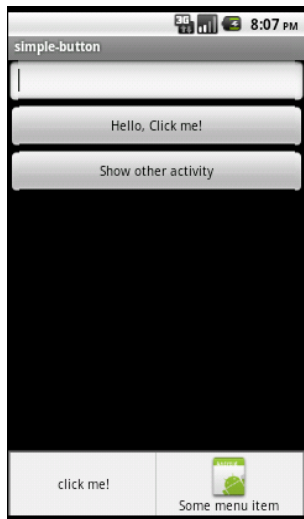
- ▶ Dodamy nowe Activity
- ▶ Będzie pobierać coś z sieci (lub udawać) oraz
- ▶ hint: przyda się jakiś progress bar etc
- ▶ utworzy z tych danych ListView
- ▶ przejdziemy do niego przez **menu** w obecnym Activity

# Menu



# Menu (vide przycisk menu)

Cel:





```
<menu xmlns:android="http://...">

    <item android:id="@+id/click_me_menu_item"
          android:title="click_me!"
          />

    <item android:id="@+id/some_menu_item"
          android:title="Some_menu_item"
          android:icon="@drawable/icon"
          />

</menu>
```

## SomeActivity#onCreateOptionsMenu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();

    inflater.inflate(R.menu.sample_menu, menu);
    return true;
}
```

## SomeActivity#onCreateOptionsMenu

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();

    inflater.inflate(R.menu.sample_menu, menu);
    return true; // true == ma zostac pokazane
}
```

## SomeActivity#onOptionsItemSelected

```
@Override
public boolean onOptionsItemSelected(Menulitem item) {
    int itemId = item.getItemId();

    switch (itemId){
        case R.id.click_me_menu_item:
            doSomething();
            break;
        default:
            Log.i(TAG, "Some weird action was requested");
    }

    return true;
}
```

## SomeActivity#onOptionsItemSelected

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int itemId = item.getItemId();

    switch (itemId){
        case R.id.click_me_menu_item:
            doSomething();
            break;
        default:
            Log.i(TAG, "Some weird action was requested");
            return false;
    }

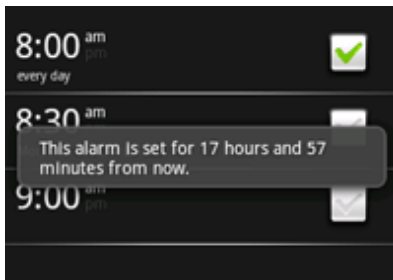
    return true; // true == obsluzylismy event
                //           == no need to bubble it
}
```

# Giving Feedback

(Toasts and Dialogs)



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



Przykład użycia:

```
Toast.makeText(getApplicationContext(), // note: explain  
    "Halo Szczecin!",  
    Toast.LENGTH_LONG)  
    .show();
```

## Co więcej potrafi Toast?

```
Toast t = Toast.makeText(MyActivity.this, txt, LENGTH_S
```



## Co więcej potrafi Toast?

```
Toast t = Toast.makeText(MyActivity.this, txt, LENGTH_S
```

Można mu zmienić pozycję:

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

## Co więcej potrafi Toast?

```
Toast t = Toast.makeText(MyActivity.this, txt, LENGTH_S
```

Można mu zmienić pozycję:

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

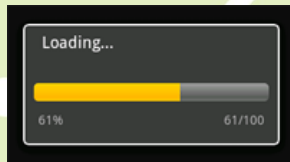
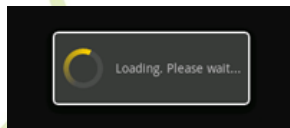
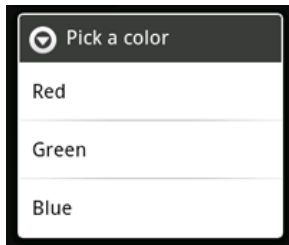
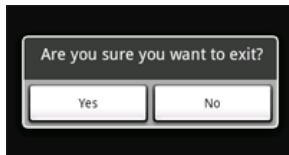
lub podmienić widok:

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

# Dialog

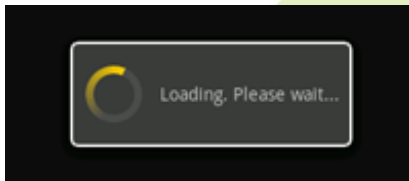


## Dialog - wyskakuje 'nad' Activity



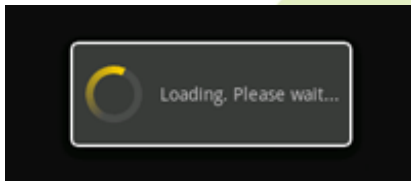
## Progress Dialog (Spinning)

```
ProgressDialog dialog = ProgressDialog  
    .show(MyActivity.this ,  
        "",  
        "Loading. Please wait..."  
        true);
```



## Progress Dialog (Spinning)

```
ProgressDialog dialog = ProgressDialog  
    .show(MyActivity.this ,  
        "" ,  
        "Loading. Please wait..."  
        true);
```



```
dialog.hide();
```

# Sloooooooooow stuff

Dotychczas siedzieliśmy na tzw. „Main Thread”.

- ▶ Zajmuje się on m.in. rysowaniem komponentów
- ▶ Jest współdzielony między Activity oraz Service!
- ▶ „Zajęcie” głównego wątku na zbyt długo spowoduje **UBICIE** naszej aplikacji!

# Sloooooooooow stuff

Introducing: **LazyWorker.java**:

```
public class LazyWorker {  
    List<String> getData() {  
        sleep(10000);  
  
        return data;  
    }  
  
    void sleep(int howLong) { /**/ }  
}
```

Będzie on udawał pobieranie danych z sieci.



# Fun Fact: **NetworkOnMainThreadException**

Od wersji 3.0, Android **wymusza** korzystanie z wątków celem robienia czegokolwiek związanego z siecią.

W przypadku zawołania np. `GET(''http://google.com'')`; na będąc na `MainThread`, zostanie rzucony wyjątek:

**`android.os.NetworkOnMainThreadException`**

GET - fikcyjna implementacja pobierająca content z sieci

# Więc... `new Thread()`?

„My się wątków nie boimy!”  
oświadczył dzielny rycerz.

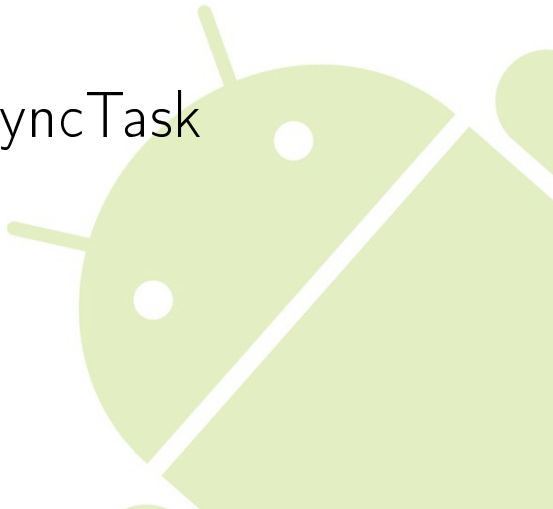


# Więc... `new Thread()`?

„My się wątków nie boimy!”  
oświadczył dzielny rycerz.

Szybko jednak zmienił zdanie, znajduwszy się w paszczy  
Mutexowego smoka.

# AsyncTask



# AsyncTask

## <Params, Progress, Result>



# AsyncTask<Params, Progress, Result>

Jedna z śliczniejszych abstrakcji na zadania asynchroniczne w API Androida.

Podstawowa implementacja wygląda tak:

```
new AsyncTask<Void , Void , Void>(){  
    Void doInBackground(Void ... voids){  
        /**/  
    }  
}.execute();
```

# AsyncTask<Params, Progress, Result>

Jedna z śliczniejszych abstrakcji na zadania asynchroniczne w API Androida.

Podstawowa implementacja wygląda tak:

```
new AsyncTask<Void , Void , Void>(){  
    Void doInBackground(Void ... voids){  
        /**/  
    }  
}.execute();
```

Anyone remember **java.lang.Void**? :-)

# AsyncTask<Params, Progress, Result>

Typowe zastosowanie, „pobieracz” danych:

```
List<String> datas =  
    new AsyncTask<Void, Integer, List<String>>() {  
        @Override  
        protected List<String> doInBackground(Void... voids)  
            return /* get stuff from the internet */;  
    }  
}.execute() // not blocking  
.get(); // blocking
```



## AsyncTask<Params, Progress, Result>

Typowe zastosowanie, „worker” + „zestaw danych”:

```
String[] data = getData();

new AsyncTask<String, Integer, Void>() {

    protected void onPreExecute(){
        Log.i(TAG, "Warning, _will_ download _the_ internet!")
    }

    protected List<String> doInBackground(Void... voids) {
        return /* get stuff from the internet */;
    }

    protected void onPostExecute(Void result) { // result
        Log.i(TAG, "Wow, _we've_ downloaded _the_ web!")
    }
}.execute(data); // varargs!
//.execute("a", "b", "c", "d"); // przypomnienie
```

# AsyncTask + ProgressDialog

```
final ProgressDialog d
    = new ProgressDialog(MyAct.this);
d.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
d.setMessage("Loading details ...");
d.setCancelable(false);

// ...

new AsyncTask<String, Integer, List<Data>>() {
```