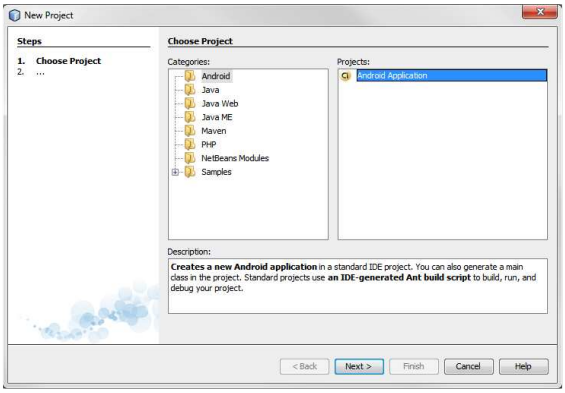
[Android] Podstawy programowania

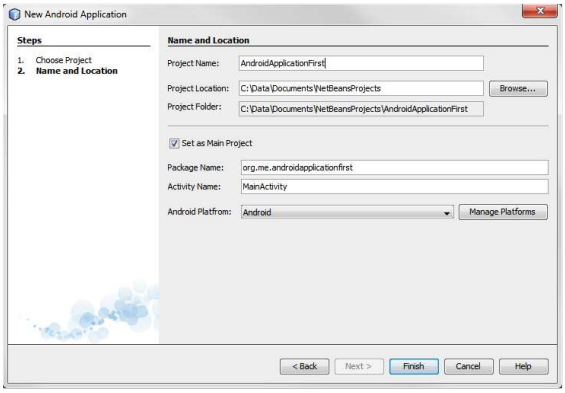
Linki[Przewodnik z przykładami](#)**Zasoby**[Krótka prezentacja wprowadzająca do budowy systemu](#)[Prosta aplikacja z menu i dialogami, którą utworzymy tutaj krok po kroku](#)

Nowy projekt w NetBeans

Standardowa procedura przy tworzeniu nowego projektu, z tym że wybieramy kategorię projektu **Android**



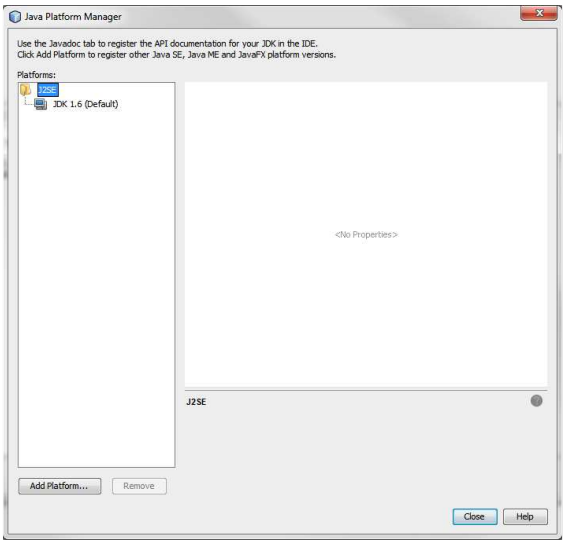
Wybieramy nazwę dla naszego projektu, np. *AndroidApplicationFirst*.
Tu możemy wybrać nazwę Aktywności, domyślnie *MainActivity*.



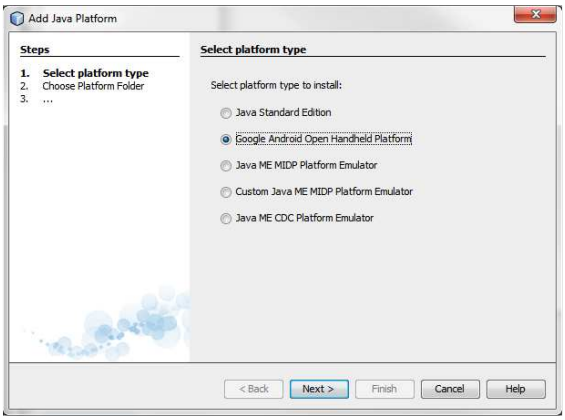
Jeśli mamy na liście dostępna platformę, to znaczy że możemy ominać kolejny punkt.

Dodawanie platformy

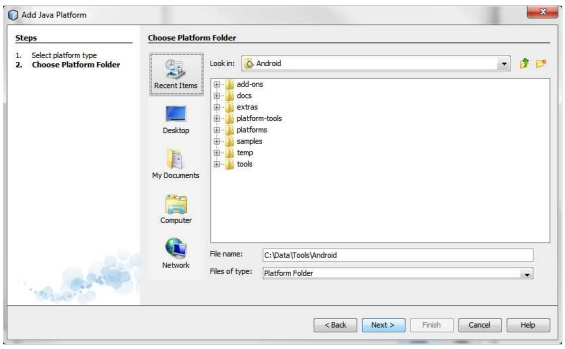
Jeżeli nie dodaliśmy wcześniej platformy klikamy w **Manage Platforms**
(To samo menu pojawia się po wyborze z menu **Tools > Java Platforms**)



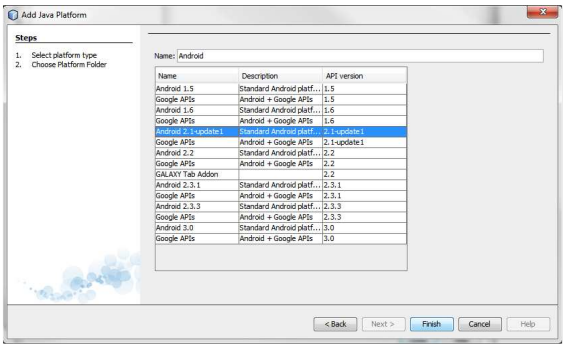
Następnie **Add Platform ...**



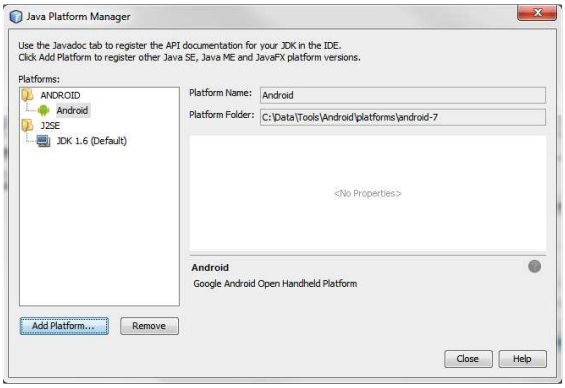
Lista typów może być różna w zależności od zainstalowanych dodatków do NetBeans'a. Zaznaczamy **Google Android Open Handheld Platform**



Znajdujemy katalog zainstalowanego pakietu Android SDK, **Next**,



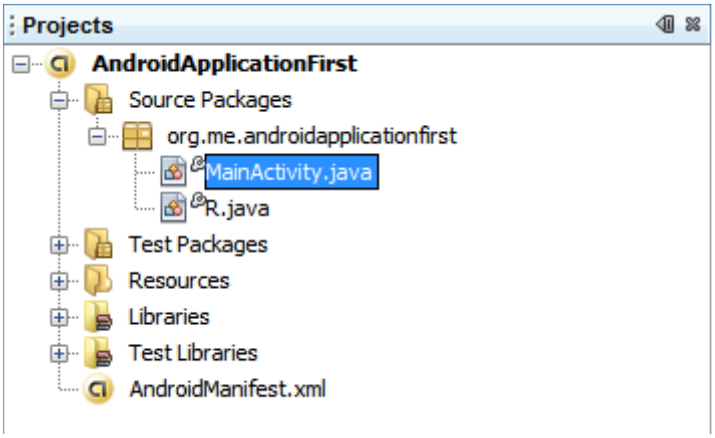
musimy wybrać platformę na którą będziemy tworzyć aplikacje.



Po zakończeniu będzie widoczna platforma.

- ## Struktura katalogów nowego projektu

Kreator tworzy automatycznie kilka plików.



- **src** - Source Packages
  - **MainActivity.java** - główna "aktywność" naszej aplikacji - plik startowy.
  - **R.java** - ten plik jest regenerowany po każdym przebudowaniu, zawiera informacje o zasobach aplikacji - skorelowane z katalogiem **Resources**.
- **res** - Resources
  - **layout** - katalog z plikami wyglądu aktywności.
    main.xml odwołanie **R.layout.main**
  - **values** - katalog z treścią aplikacji,

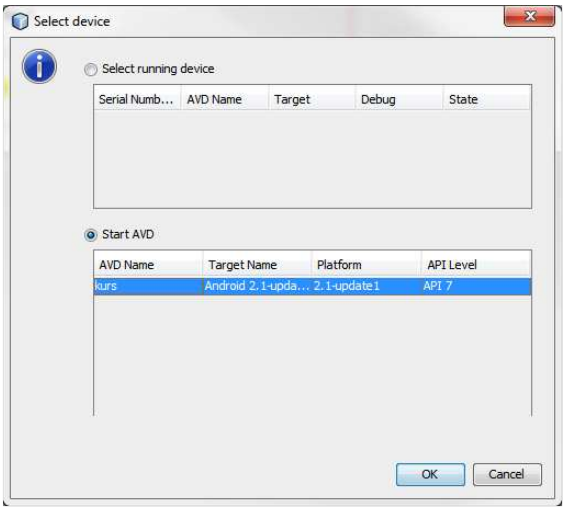
np. **strings.xml** - zawierający napisy do których można odwołać się przez **R.string.*NAME***

- ## **Hello Android from NetBeans** czyli pierwsze uruchomienie

Po wciśnięciu *F6* niestety byśmy niewiele zobaczyli, więc dodajmy jedną linijkę do metody **onCreate**

```
setContentView(R.layout.main);
```

To sprawi że nasza aktywność pokaże szablon zapisany w **res/layout/main.xml**



Jeśli nie mamy uruchomionego aktulanie emulatora to wyskoczy nam okienko by wystartować.

- ## Nowy przycisk

Dodamy przycisk do naszej aplikacji, zmodyfikujemy plik **layout/main.xml**

```
<?xml version="1.0" encoding="UTF-8"?> <LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical" android:layout_width="fill_parent"
android:layout_height="fill_parent"> <TextView
android:id="@+id/textView1" android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Hello Android
from NetBeans"/><Button android:id="@+id/button1"
android:layout_width="fill_parent"
android:layout_height="wrap_content" android:text="Hello, I am a
Button" /> </LinearLayout>
```

@+id/button1 oznacza to dodanie **button1** do listy zasobów typu **id**

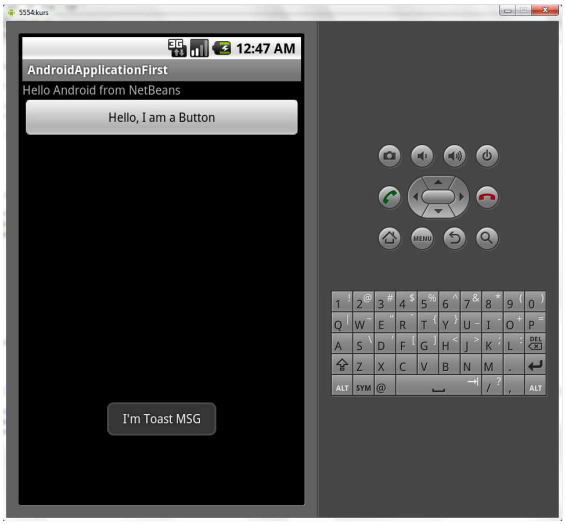
By obsłużyć obiekt znajdujący się na "formie" trzeba go najpierw znaleźć:

```
button1 = (Button) findViewById(R.id.button1);
```

Dalej programuje się "normalnie*" jak w javie ;]

Sprawmy by po wciśnięciu guzika pojawił się **Toast**.
Kod po przeróbkach będzie wyglądał następująco:

```
/* * To change this template, choose Tools | Templates * and open
the template in the editor. */ package
org.me.androidapplicationfirst; import android.app.Activity;
import android.os.Bundle; import android.view.View; import
android.view.View.OnClickListener; import android.widget.Button;
import android.widget.TextView; import android.widget.Toast; /** *
* @author arti */ public class MainActivity extends Activity {
private Button button1; private TextView textView1; private
OnClickListener buttonListner = new OnClickListener() { public
void onClick(View arg0) { //throw new UnsupportedOperationException
("Not supported yet."); Toast.makeText(getApplicationContext
(), "I'm Toast MSG", Toast.LENGTH_LONG).show(); } }; /** Called
when the activity is first created. */ @Override public void
onCreate(Bundle icicle) { super.onCreate(icicle); // ToDo add your
GUI initialization code here setContentView(R.layout.main);
button1 = (Button) findViewById(R.id.button1); textView1 =
(TextView) findViewById(R.id.textView1); button1.setOnClickListener
(buttonListner); } }
```



Menu aplikacji

Dodamy teraz menu do naszej aplikacji.
Najpierw należy stworzyć katalogi do zasobów

- **drawable** - katalog z grafikami - odwołanie **R.drawable.*NAME***

umieściłem tam pliki: newspaper_32  help_32  delete_32

- **menu** - katalog ze strukturą menu - **R.menu.*NAME*** plik: **menu.xml**

```
<?xml version="1.0" encoding="utf-8"?> <menu
xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/news"
android:icon="@drawable/newspaper_32"
android:title="@string/news"> <menu> <item
android:id="@+id/news_create"
android:title="@string/news_create"/> <item
android:id="@+id/news_delete"
android:title="@string/news_delete" /> </menu> </item> <item
android:id="@+id/help" android:icon="@drawable/help_32"
android:title="@string/help" /> <item android:id="@+id/exit"
android:icon="@drawable/delete_32"
android:title="@string/exit" /> </menu>
```

Widać tu inny sposób odwołania do zasobów **@TYP_ZASOBU/NAZWA_ZASOBU**.
Zmodyfikujmy zatem plik **values/strings.xml**

```
<?xml version="1.0" encoding="UTF-8"?> <resources> <string
name="app_name">AndroidApplicationFirst</string> <string
name="news">Nowości</string> <string
name="news_create">Stwórz</string> <string
name="news_delete">Kasuj</string> <string
name="help">Pomoc</string> <string
name="exit">Koniec</string> <string name="Yes">Tak</string>
<string name="No">Nie</string> <string
name="news_delete_msg">Czy chcesz usunąć?</string>
</resources>
```

By wyświetlić menu należy dodać metodę **onCreateOptionsMenu**

```
@Override public boolean onCreateOptionsMenu(Menu menu) { //return
super.onCreateOptionsMenu(menu); MenuInflater inflater =
getMenuInflater(); inflater.inflate(R.menu.menu, menu); return
true; }
```

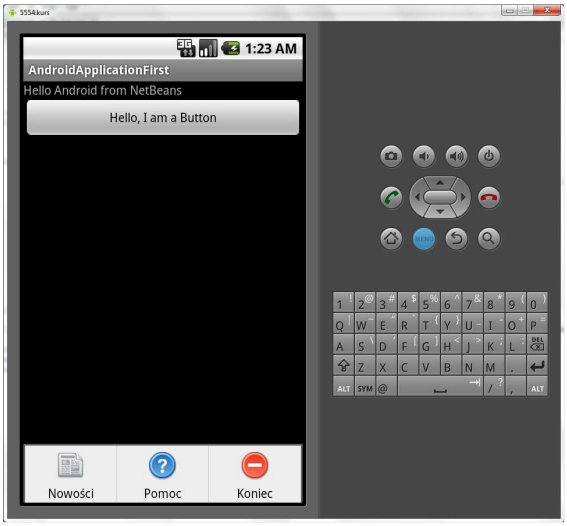
By reagować na akcje **onOptionsItemSelected**

```
@Override public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) { case R.id.news_create: newsCreate();
return true; case R.id.news_delete: newsDelete(); return true;
case R.id.help: return true; case R.id.exit: finish(); return
true; default: return super.onOptionsItemSelected(item); } }
```

finish() - zamyka aktywność

```
private void newsCreate() { Toast.makeText(getApplicationContext
(), "News created :P", Toast.LENGTH_LONG).show(); } private void
newsDelete() { AlertDialog.Builder builder = new
AlertDialog.Builder(this); builder.setMessage
(R.string.news_delete_msg).setCancelable(false).setPositiveButton
(R.string.Yes, new DialogInterface.OnClickListener() { public void
```

```
onClick(DialogInterface dialog, int id) { MainActivity.this.finish
(); } }).setNegativeButton(R.string.No, new
DialogInterface.OnClickListener() { public void onClick
(DialogInterface dialog, int id) { dialog.cancel(); } });
AlertDialog alert = builder.create(); alert.show(); }
```



W aplikacji do pobrania reagujemy jeszcze na wybranie pomocy przez otwarcie dialogu - na podstawie: przewodnika