РЕСУРСИ

План:

Робота з ресурсами

Застосування ресурсів

Ресурси рядків

Ресурси Plurals

Ресурси dimension

Переклад з dip в пікселі

Ресурси Color і установка кольору

Робота з ресурсами

Ресурс в додатку Android представляє собою файл, наприклад, файл розмітки інтерфейсу або деяке значення, наприклад, просту рядок. Тобто ресурси представляють собою і файли розмітки, і окремі рядки, і звукові файли, файли зображень і т.д. Всі ресурси знаходяться в проекті в каталозі *res* . Для різних типів ресурсів, визначених у проекті, в каталозі *res* створюються підкаталоги. Підтримувані підкаталоги:

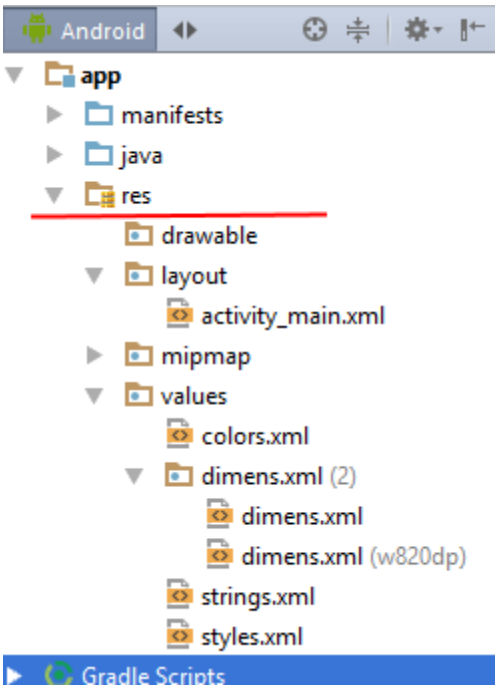
- **animator/** : xml-файли, що визначають анімацію властивостей
- **anim/** : xml-файли, що визначають tween-анімацію
- **color/** : xml-файли, що визначають список квітів
- **drawable/** : Графічні файли (*.png* , *.jpg* , *.gif*)
- **mipmap/** : Графічні файли, які використовуються для іконок додатка під різні дозволи екранів
- **layout/** : xml-файли, що визначають для користувача інтерфейс програми
- **menu/** : xml-файли, що визначають меню програми
- **raw/** : різні файли, які зберігаються в початковому вигляді
- **values/** : xml-файли, які містять різні використовувані в додатку значення, наприклад, ресурси рядків
- **xml/** : Довільні xml-файли

В цілому ми можемо визначити такі типи ресурсів:

Ресурс	Каталог проекту	файл	елемент в файлі
Рядки	/Res/values/	strings.xml	<String>
Plurals	/Res/values/	strings.xml	<Plurals>
масиви рядків	/Res/values/	strings.xml або arrays.xml	<String-array>
Логічні значення	/Res/values/	bools.xml	<Bool>

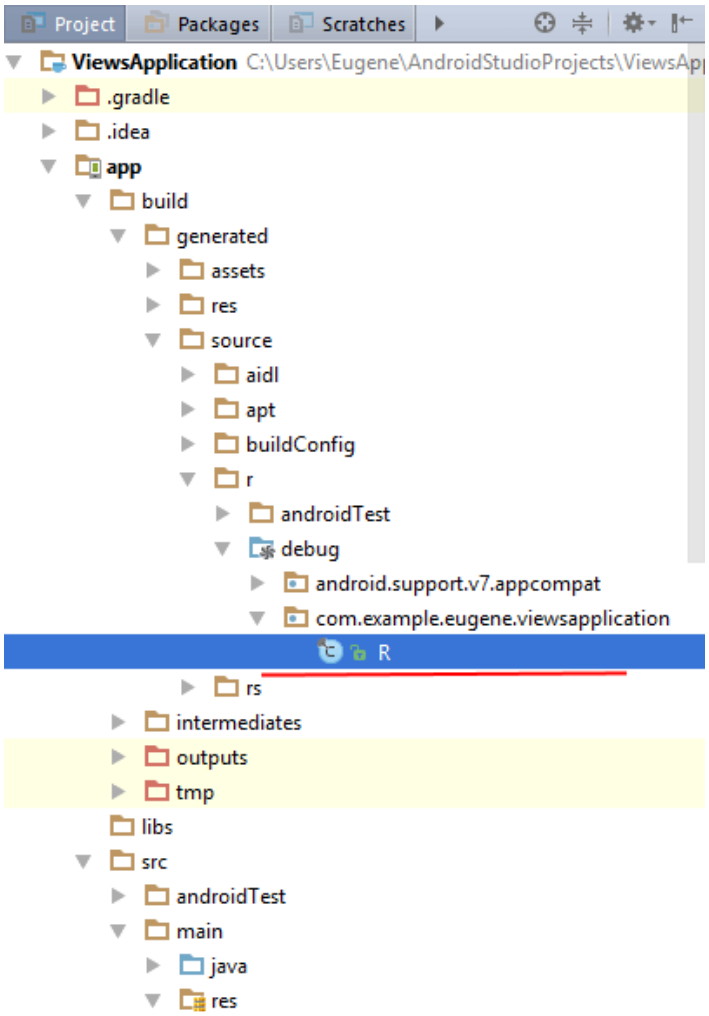
Boolean			
Кольори	/Res/values/	colors.xml	<Color>
список кольорів	/Res/color/	Довільний назву	<Selector>
Розміри (Dimensions)	/Res/values/	dimens.xml	<Dimen>
ідентифікатори ID	/Res/values/	ids.xml	<Item>
Цілі числа	/Res/values/	integers.xml	<Integer>
Масив цілих чисел	/Res/values/	integers.xml	<Integer-array>
графічні файли	/Res/drawable/	Файли з розширенням jpg і png	-
Tween-анімація	/Res/anim/	Файл xml з довільною назвою	<Set", "alpha", "rotate », «scale", "translate>
покадрова анімація	/Res/drawable/	Файл xml з довільною назвою	<Animation-list>
анімація властивостей	/Res/animator/	Файл xml з довільною назвою	<Set "," objectAnimator>, <valueAnimator>
Меню	/Res/menu/	Файл xml з довільною назвою	<Menu>
XML-файли	/Res/xml/	Файл xml з довільною назвою	
Бінарні і текстові ресурси	/Res/raw/	Файли мультимедіа (mp3, mp4), текстові та інші файли	
Розмітка графічного інтерфейсу	/Res/layout/	Файл xml з довільною назвою	
Стилі і теми	/Res/values/	styles.xml, themes.xml	<Style>

Наприклад, якщо ми візьмемо стандартний проект Android Studio, який створюється за замовчуванням, то там можемо помітити наявність вже декількох папок для різних ресурсів в каталозі *res* :



За замовчуванням тут є каталоги не для всіх типів ресурсів, які використовуватимуться в Android, проте при необхідності ми можемо додати в папку *res* потрібний каталог, а в нього потім помістити ресурс.

Коли відбувається компіляція проекту відомості про всі ресурси додаються в спеціальний файл **R.java**, який можна знайти в проекті по шляху *app\build\generated\source\r\debug\[пакет_додатку]*. Щоб побачити його потрібно треба переключитися на повноцінний вид проекту:



У цьому файлі будуть зберігатися всі визначення ресурсів у вигляді числових констант. Наприклад, у проекті за замовчуванням є ресурс розмітки інтерфейсу - файл *activity_main.xml* в папці *res/layout* . Для цього ресурсу в класі R буде створюватися приблизно наступна константа:

```
            public final class R {
.............................
                public static final class layout {
                    public static final int activity_main=0x7f030001;
                }
.....................................
```

Застосування ресурсів

Існує два способи доступу до ресурсів: в файлі вихідного коду і в файлі xml.

Посилання на ресурси в коді

Тип ресурсу в даному записі посилається на одне з просторів (внутрішніх класів), визначених у файлі R.java, які мають відповідні їм типи в xml:

- R.drawable(йому відповідає тип в xml drawable)
- R.id(id)
- R.layout(layout)
- R.string(string)
- R.attr(attr)
- R.plural(plurals)
- R.array(string-array)

Наприклад, для установки ресурсу activity_main.xml в якості графічного інтерфейсу в коді MainActivity в методі onCreate()є такий рядок:

setContentView(R.layout.activity_main);

Через вираз R.layout.activity_mainми і посилаємося на ресурс activity_main.xml, де layout- тип ресурсу, а activity_main- ім'я ресурсу.

Подібним чином ми можемо отримувати інші ресурси. Наприклад, у файлі *res/values/strings.xml* визначено ресурс app_name:

<resources>
  <string name="app_name">ViewsApplication</string>
</resources>

Цей ресурс посилається на рядок. Щоб отримати посилання на даний ресурс в коді java, ми можемо використовувати вираз R.string.app_name.

Доступ в файлі xml

Нерідко виникає необхідність посилатися на ресурс у файлі xml, наприклад, у файлі, який визначає візуальний інтерфейс, наприклад, в activity_main.xml. Посилання на ресурси в файлах xml мають наступну формалізовану форму:@[имя_пакета:]тип_ресурса/имя_ресурса

- Ім’я_пакету представляє ім'я пакету, в якому ресурс знаходиться (вказувати необов'язково, якщо ресурс знаходиться в тому ж пакеті)
- тип_ресурсу представляє підклас, визначений у класі R для типу ресурсу
- Ім’я_ресурсу ім'я файлу ресурсу без розширення або значення атрибута android:nameв XML-елементі (для простих значень).

Наприклад, ми хочемо вивести в елемент TextView рядок, яка визначена у вигляді ресурсу в файлі strings.xml:

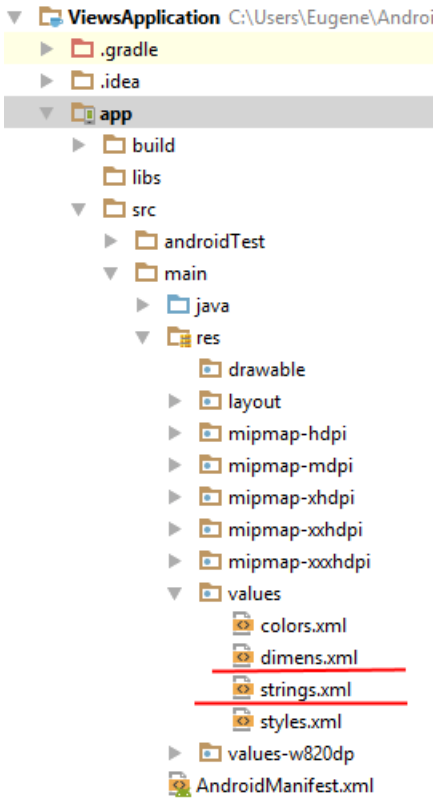
метод getResources

Для отримання ресурсів в класі Activity ми можемо використовувати метод **getResources()** , який повертає об'єкт **android.content.res.Resources** . Але щоб отримати сам ресурс, нам треба у отриманого об'єкта Resources викликати один з методів:

- **getString()** : отримує рядок з файлу strings.xml по числовому ідентифікатору
- **getDimension()** : отримує числове значення - ресурс dimen
- **getDrawable()** : отримує графічний файл
- **getBoolean()** : отримує значення boolean

Це тільки деякі методи. Але коротко розглянемо їх застосування. Візьмемо той же файл *res/values/strings.xml* як джерело ресурсів. Нехай файл **strings.xml** виглядає так:

```
<resources>
<string name="app_name">ViewsApplication</string>
</resources>
```



І змінимо код MainActivity:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        // отримання ресурсів з файлу values/strings.xml
        String app_name = getResources().getString(R.string.app_name);

        TextView textView = new TextView(this);
        textView.setText(app_name);

        setContentView(textView);
    }
}
```

Тут, використовуючи метод getResources()отримуємо все ресурси. Числові ресурси встановлюємо з як відступів всередині елемента TextView, а строковий ресурс - як текст. При запуску програми ми побачимо застосування цього матеріалу:

Ресурси рядків

Ресурси рядків - один з важливих компонентів програми. Ми використовуємо їх при виведенні назви програми, різного тексту, наприклад, тексту кнопок і т.д.

XML-файли, що представляють собою ресурси рядків, знаходяться в проекті в папці *res/values* . За замовчуванням ресурси рядків знаходяться в файлі *strings.xml* , який може виглядати наступним чином:

```
<resources>
    <string name="app_name">ViewsApplication</string>
</resources>
```

У найпростішому вигляді цей файл визначає один ресурс "app_name", який встановлює назву програми. Але природно ми можемо визначити будь-які строкові ресурси. Кожен окремий ресурс визначається за допомогою елемента **string** , а його атрибут nameмістить назву ресурсу.

Для ресурсів рядків в класі Rвизначається внутрішній клас static final class string. Цей клас використовується в якості простору для зберігання ідентифікаторів ресурсів рядків:

```
public static final class string {
    public static final int app_name = 0x7f040000;
}
```

Константа app_nameмає тип не String, а int, а її значення - числовий ідентифікатор ресурсу.

Потім в додатку в файлах коду ми можемо посилатися на ці ресурси:

```
R.string.app_name
```

А ОС Android сама зіставити дані числові ідентифікатори з відповідними ресурсами рядків. наприклад:

```
String application_name = getResources().getString(R.string.app_name);
```

Або в xml-файлі:

```
@string/app_name
```

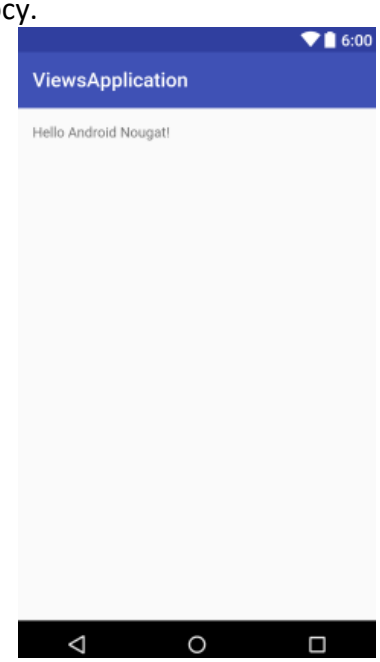
Наприклад, змінимо файл *res/values/strings.xml* наступним чином:

```
<resources>
    <string name="app_name">test_02_theme</string>
    <string name="message">Hello Android Nougat!</string>
</resources>
```

Тут доданий ресурс message зі значенням "Hello Android Nougat!".
Тепер використовуємо ресурс у файлі activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android=http://schemas.android.com/apk/res/android
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <TextView
        android:id="@+id/welcome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/message" />
</RelativeLayout>
```

За допомогою виразу @string/message передаємо атрибуту android:text значення з ресурсу.



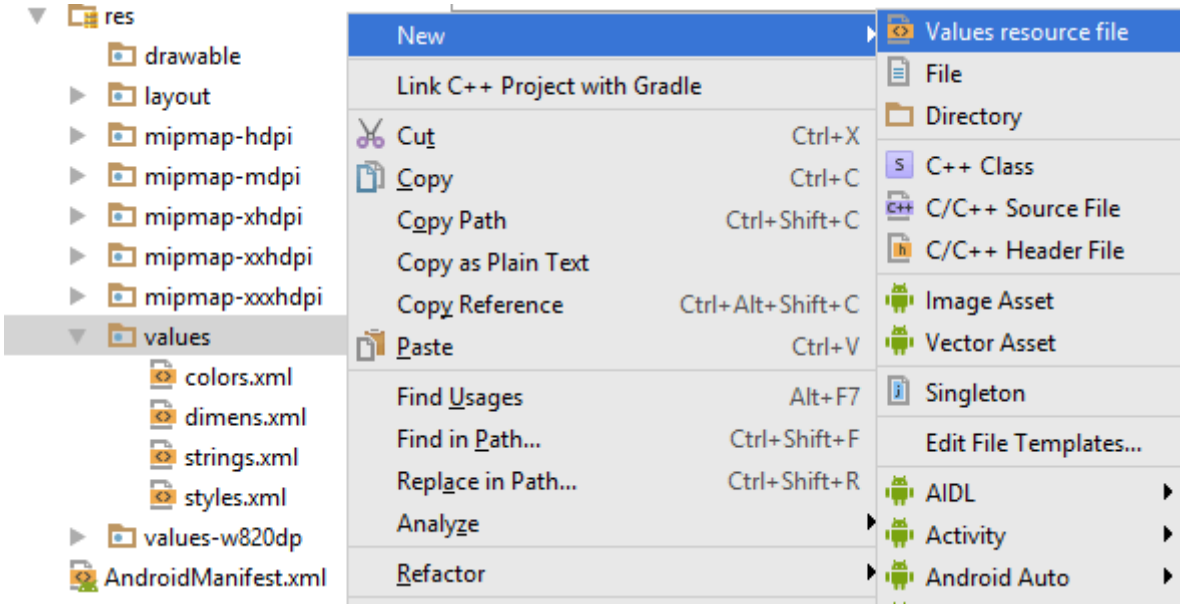
Аналогічно ми могли б використовувати ресурс в коді Activity:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        // встановлюємо в якості інтерфейсу файл second_layout.xml
        setContentView(R.layout.activity_main);

        // отримуємо елемент textView
        TextView textView = (TextView) findViewById(R.id.welcome);
        // отримуємо ресурс
        String message = getResources().getString(R.string.message);
        // перевстановлюємо в нього текст
        textView.setText(message);
    }
}
```

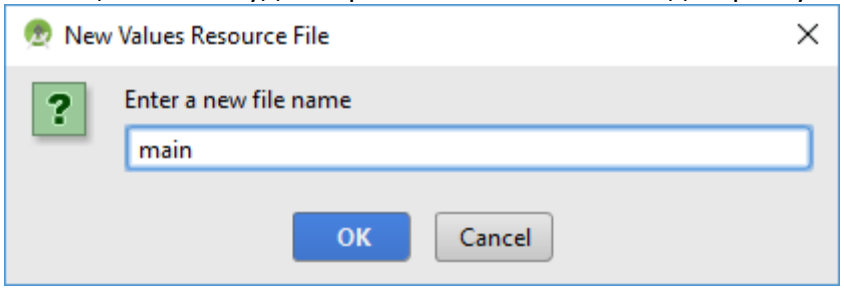
Хоча за замовчуванням для ресурсів рядків застосовується файл strings.xml, але розробники можуть додавати додаткові файли ресурсів в каталог проекту *res/values* . При цьому досить дотримуватися структуру файлу: він повинен мати кореневий вузол <resources> і мати один або кілька елементів <string>.

Так, натиснемо на папку *res/values* правою кнопкою миші і в списку, що з'явився виберемо пункт **New -> Value Resource File** :



Причому слід зазначити, що даний тип файлів буде характерний для будь-якого типу ресурсів, який додається в папку *res/values* .

Після цього нам буде запропоновано визначити для файлу ім'я:



Назвемо, наприклад, main , і в папку *res/values* буде додано новий файл main.xml. Визначимо в ньому пару ресурсів:

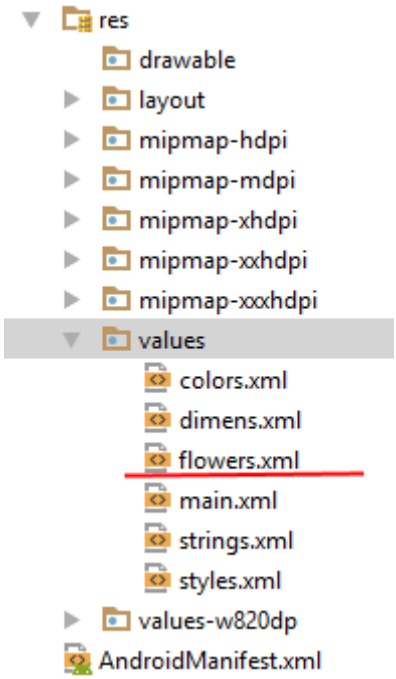
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="welcome">Ласкаво просимо</string>
    <string name="click_button">Натисніть на кнопку</string>
</resources>
```

І після цього ми зможемо використовувати певні тут ресурси в коді Activity або в файлі layout.

Ресурси Plurals

Plurals представляють ще один вид набору рядків. Він призначений для опису кількості елементів. Для чого це треба? Наприклад, візьмемо іменник: нерідко воно змінює закінчення в залежності від числівника, яке з ним вживається: 1 квітка, 2 квітки, 5 квіток. Для подібних випадків і використовується ресурс plurals.

Подивимося на прикладі. Додамо в папку *res/values* новий ресурс. Назвемо його **flowers** :



Змінимо його вміст в такий спосіб:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<plurals name="flowers">
    <item quantity="one">%d квітка</item>
    <item quantity="few">%d квітки</item>
    <item quantity="many">%d квіток</item>
</plurals>
</resources>
```

Для завдання ресурсу використовується елемент <plurals>, для якого існує атрибут name, який одержує в якості значення довільну назву, за яким потім посилаються на даний ресурс.

Самі набори рядків вводяться дочірніми елементами <item>. Цей елемент має атрибут quantity, який має значення, яке вказує, коли цей рядок використовується. Даний атрибут може набувати таких значень:

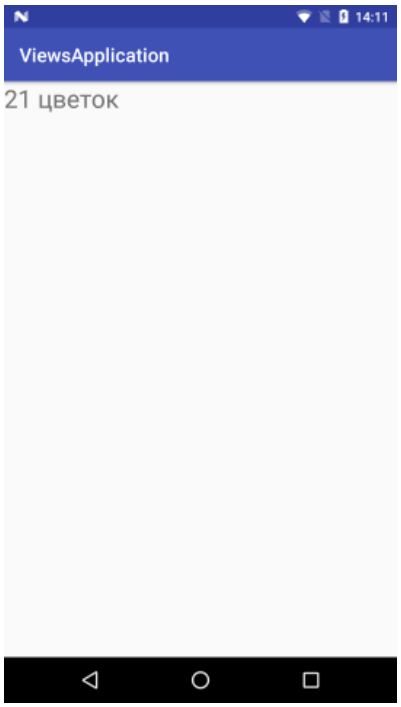
- **zero** : рядок для кількості в розмірі 0
- **one** : рядок для кількості в розмірі 1 (для російської мови - для завдання всіх кількостей, що закінчуються на 1, крім 11)
- **two** : рядок для кількості в розмірі 2
- **few** : рядок для невеликої кількості
- **many** : рядок для великих кількостей
- **other** : всі інші випадки

Причому в даному випадку багато залежить від конкретної мови. А система сама дозволяє визначити, яке значення брати для того чи іншого числа.

Використання даного ресурсу можливо тільки в коді java. Тому змінимо код MainActivity:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        String rose = getResources().getQuantityString(R.plurals.flowers,
21, 21);

        TextView textView = new TextView(this);
        textView.setText(rose);
        textView.setTextSize(26);
        setContentView(textView);
    }
}
```

За допомогою методу getQuantityString ми отримуємо значення ресурсу. Першим параметром передаємо ідентифікатор ресурсу. Другим параметром йде значення. для якого потрібно знайти потрібний рядок. Третій параметр являє собою значення, яке буде вставлятися на місце плейсхолдера % d . Тобто ми отримуємо рядок для числа 21.



Ресурси dimension

Визначення розмірів повинно знаходитися в папці **res/values** в файлі з будь-яким довільним ім'ям. Загальний синтаксис визначення ресурсу наступний:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <dimen name="ім'я ресурсу">використовуваний розмір</dimen>
</resources>
```

Як і інші ресурси, ресурс dimension визначається в кореневому елементі <resources>. Тег <dimen>позначає ресурс і як значний приймає деяке значення розміру в одній з прийнятих одиниць виміру (dp, sp, pt, px, mm, in).

Так, додамо в Android Studio в папку **res/values** новий файл, який назвемо **dimens.xml** . Визначимо в ньому наступне вміст:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <dimen name="activity_horizontal_margin">16dp</dimen>
        <dimen name="activity_vertical_margin">16dp</dimen>
        <dimen name="text_size">16sp</dimen>
</resources>
```

Тут визначені два ресурси для відступів activity_horizontal_margin і activity_vertical_margin, які зберігають значення 16 dp, і ресурс text_size, який зберігає висоту шрифту - 16sp. Назви ресурсів можуть бути довільними.

Використовуємо ресурс у файлі activity_main.xml:

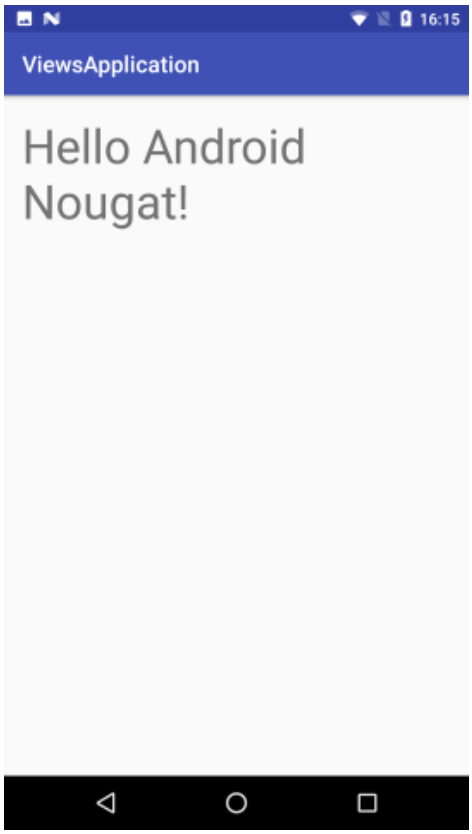
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">
```

```
        <TextView
            android:textSize="@dimen/text_size"
            android:layout_width="wrap_content"
            android:text="Hello Android Nougat!" />
    </RelativeLayout>
```

Ресурси dimension використовуються для таких атріубтов візуальних елементів, які в якості значення вимагають вказівку числового значення. Наприклад, атріубут android:layout_heightабо android:textSize. Для отримання ресурсу в xml після "@dimen /" вказується ім'я ресурсу.

Для отримання ресурсів в коді java застосовується метод **getDimension()** класу Resources:

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        float textSize = getResources().getDimension(R.dimen.text_size);
        int leftPadding =
(int)getResources().getDimension(R.dimen.activity_horizontal_margin);
        int topPadding =
(int)getResources().getDimension(R.dimen.activity_vertical_margin);

        TextView textView = new TextView(this);

        textView.setText("Hello Android Nougat!");
        textView.setTextSize(textSize);
        textView.setPadding(leftPadding, topPadding, leftPadding, topPadding);

        setContentView(textView);
    }
}
```



Переклад з dip в пікселі

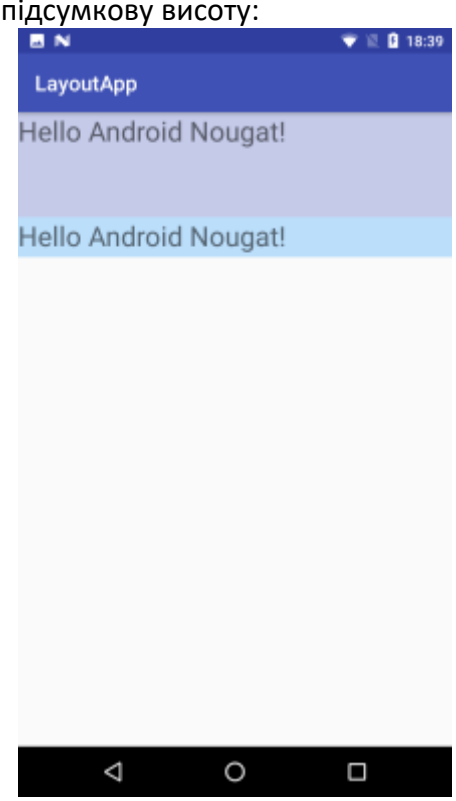
У разі вище ми використовували переважно одиниці dip. Але що, якщо ми хочемо динамічно перейти від одних одиниць вимірювання до інших, наприклад, від dip до пікселів. Для цього ми можемо застосувати клас **android.util.TypedValue** :

```
package com.example.i.test_02_theme;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;
import android.util.TypedValue;

public class MainActivity extends AppCompatActivity {

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

        // умовне значение в dp
        int textViewHeight = 100;

        LinearLayout linearLayout = new LinearLayout(this);
        linearLayout.setOrientation(LinearLayout.VERTICAL);

        TextView textView1 = new TextView(this);
        textView1.setText("Hello Android Nougat!");
        textView1.setTextSize(26);
        textView1.setBackgroundColor(0xffc5cae9);

        // перетворимо розмір з dp в фізичні пікселі
        int height = (int)TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,
```

```
                textViewHeight, getResources().getDisplayMetrics());
        textView1.setHeight(height);

        linearLayout.addView(textView1);

        TextView textView2 = new TextView(this);
        textView2.setText("Hello Android Nougat!");
        textView2.setTextSize(26);
        textView2.setBackgroundColor(0xffbbdefb);
        textView2.setHeight(textViewHeight);
        linearLayout.addView(textView2);

        setContentView(linearLayout);
    }
}
```

За допомогою методу **TypedValue.applyDimension()** здійснюється переклад з одних одиниць в інші. Перший параметр - тип вихідних одиниць вимірювання (в даному випадку dip), другий параметр - кількість для конвертації, а третій параметр - параметр DisplayMetrics, який дозволяє провести конвертацію.

В даному випадку створюється 2 елементи TextView. Для установки висоти для обох елементів використовується значення з змінної textViewHeight, проте в першому випадку це значення піддається преоразованій з dp в пікселі, і відповідно обидва елементи матимуть різну підсумкову висоту:



Ресурси Color і установка кольору

У додатку Android також можна визначати ресурси квітів (Color). Вони повинні зберігатися в файлі по шляху *res/values* і також, як і ресурси рядків, укладені в тег <resources>. Так, за замовчуванням при створенні найпростішого проекту в папку *res/values* додається файл **colors.xml** :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

Колір визначається за допомогою елемента <color>. Його атрибут name встановлює назву кольору, яке буде використовуватися в додатку, а шістнадцяткове число - значення кольору.

Для завдання колірних ресурсів можна використовувати такі формати:

- #RGB (# F00 - 12-бітове значення)
- #ARGB (# 8F00 - 12-бітове значення з додаванням альфа-каналу)
- #RRGGBB (# FF00FF - 24-бітове значення)
- #AARRGGBB (# 80FF00FF - 24-бітове значення з додаванням альфа-каналу)

Щоб простіше було орієнтуватися в колірних схемах і в тому, які кольори краще використовувати, на сторінці https://material.io/guidelines/style/color.html# можна знайти великий список з квітів, рекомендемих компанією Google.

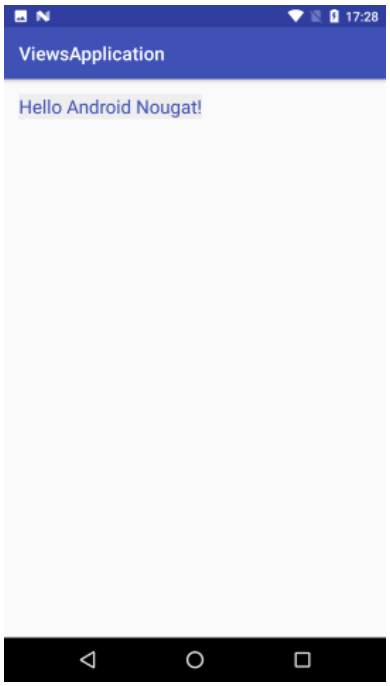
Змінимо файл colors.xml, додавши ще один колір:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="textViewColor">#EEEEEE</color>
</resources>
```

Застосуємо кольору в файлі *activity_main.xml* :

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin">

    <TextView
        android:textSize="20dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textColor="@color/colorPrimary"
        android:background="@color/textViewColor"

        android:text="Hello Android Nougat!" />

</RelativeLayout>
```

За допомогою атрибута android:textColorвстановлюється колір тексту в TextView, а атрибут android:backgroundвстановлює фон TextView. Як значення вони використовують колір, наприклад, в тому ж шестнадцатеричном форматі. Для отримання самого кольору після "@color/" вказується ім'я ресурсу.



Також можна використовувати колірні ресурси в коді MainActivity:

```
package com.example.i.test_02_theme;

import android.support.v4.content.ContextCompat;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

        int textColor = ContextCompat.getColor(this, R.color.colorPrimary);
        int backgroundColor = ContextCompat.getColor(this,
R.color.textViewColor);

        TextView textView = new TextView(this);
        textView.setText("Hello Android Nougat!");
        textView.setTextSize(20);
        textView.setPadding(16, 16, 16, 16);

        textView.setTextColor(textColor);
        textView.setBackgroundColor(backgroundColor);

        setContentView(textView);
    }
}
```

Для отримання кольору застосовує метод **ContextCompat.getColor()** , який в якості першого параметра приймає поточний об'єкт Activity, а другий парамет - ідентифікатор колірного ресурсу.