Работа с ресурсами приложения

Что такое ресурс приложения?

Ресурсы являются необходимой составляющей любого приложения. Ресурсы представляют собой и файлы разметки, и отдельные строки, и звуковые файлы, и файлы изображений и другие.
Хранятся все эти ресурсы в специальном каталоге **res**. Там вы сможете добавлять/удалять различные ресурсы.
@android: означает, что мы используем системные ресурсы вместо своих. Однако нужно учитывать что на разных устройствах будут разные предустановленные ресурсы.
Например, **@android:drawable/ic_search_category_default -** это ссылка на системную иконку поиска в виде лупы. И эта иконка скорее всего будет отличаться на различных устройствах

Типы ресурсов

Ресурс	**Каталог проекта**	**Файл**	**Элемент в файле**
Строки	/res/values/	strings.xml	<string>
Plurals	/res/values/	strings.xml или Произвольное название	<plurals>
Массивы строк	/res/values/	strings.xml или arrays.xml	<string-array>
Логические значения Boolean	/res/values/	bools.xml	<bool>
Цвета	/res/values/	colors.xml	<color>
Список цветов	/res/color/	Произвольное название	<selector>
Размеры (Dimensions)	/res/values/	dimens.xml	<dimen>

Идентификаторы ID	/res/values/	ids.xml	<item>
Целые числа	/res/values/	integers.xml	<integer>
Массив целых чисел	/res/values/	integers.xml	<integer-array>
Графические файлы	/res/drawable/	Файлы с расширением jpg и png	-
Tween-анимация	/res/anim/	Файл XML с произвольным названием	<set>, <alpha>, <rotate>, <scale>, <translate>
Покадровая анимация	/res/drawable/	Файл XML с произвольным названием	<animation-list>
Анимация свойств	/res/animator/	Файл XML с произвольным названием	<set>, <objectAnimator>, <valueAnimator>
Меню	/res/menu/	Файл XML с произвольным названием	<menu>
XML-файлы	/res/xml/	Файл XML с произвольным названием	
Бинарные и текстовые ресурсы	/res/raw/	Файлы мультимедиа (mp3, mp4), текстовые и другие файлы	
Разметка графического интерфейса	/res/layout/	Файл XML с произвольным названием	
Стили и темы	/res/values/	styles.xml, themes.xml	<style>

Давайте вкратце пройдемся по наиболее распространенным видам ресурсов

Немного о strings

Весь текст крайне желательно хранить в ресурсах приложения. Если вам нужно будет изменить этот текст, а он у вас используется в нескольких местах, то вы просто сможете поправить текст в strings.xml и он изменится всюду, где вы использовали этот строковый ресурс.
Также, если будет необходимость перевести приложение на другой язык, достаточно будет изменить несколько файлов со строковыми ресурсами, а не искать текст в коде. Вам нужно для каждой локализации создать отдельный strings.xml. Как это делать мы рассмотрим в следующих уроках.

Задаем текст из строкового ресурса в xml:
```
android:text="@string/example_text"
```

Задаем текст из строкового ресурса в коде:
```
mTextView.setText(R.string.app_name);
```

Немного о drawable

Drawable можно хранить как картинку и как xml-вектор. Картинки надо нарезать под соответствующие типы экранов(mdpi, hdpi, и т.д., подробнее об этом в следующем уроке), вектор может неправильно считаться системой и не везде поддерживается.

С Android 5.0 появилась возможность использовать векторную графику. Также, была добавлена поддержка в Support Library начиная с 7 версии API.

Преимущество использования векторов против png заключается в том, что не нужно постоянно нарезать каждую png для разных типов экрана + векторы выглядят намного четче в отличие от png.

Тут мы с помощью атрибута **src** у ImageView устанавливаем **png drawable** “ic_delete”.
```
android:src="@android:drawable/ic_delete"
```

Но при использовании векторной графики используйте **srcCompat** вместо **src** атрибута.
```
app:srcCompat="@android:drawable/ic_delete_vector"
```

Обращение к drawable ресурсу через метод **getResources()**
```
Drawable drawable = getResources().getDrawable(R.drawable.ic_android_black);
```

или через метод ContextCompat
```
Drawable drawable = ContextCompat.getDrawable(this, R.drawable.ic_android_black);
```

И устанавливаем наш Drawable в наш ImageView
```
mExampleImageView.setImageDrawable(drawable);
```

Немного о dimen

Если вам часто нужны одинаковые значения для каких-то размеров, то dimen - то, что вам нужно. Например, у вас все TextView имеют padding равный 8dp. А потом дизайнер захочет изменить это требование и вместо padding равного 8dp станет 4dp. Вы легко сможете исправить это в самом ресурсе, не меняя это значение в каждом TextView.

Как и другие ресурсы, ресурс dimension определяется в корневом элементе <resources>. Тег <dimen> обозначает ресурс и в качестве значения принимает некоторое значение размера в одной из принятых единиц измерения (dp, sp, pt, px, mm, in). Определение размеров должно находиться в папке res/values в файле с любым произвольным именем. Мы же решили назвать этот файл dimens.
Так он выглядит изнутри:
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <dimen name="text_view_padding">8dp</dimen>
  <dimen name="text_view_text_size">16sp</dimen>
</resources>
```

Обращение к dimen ресурсу из xml:

```
android:padding="@dimen/text_view_padding"
android:textSize="@dimen/text_view_text_size"
```

Обращение к dimen ресурсу из кода:
```
int textViewPadding = (int) getResources().getDimension(R.dimen.text_view_padding);
```

Обратите внимание, что метод **getDimension()** возвращает dp, но не все методы принимают dp на вход. Чаще всего методы принимают на вход пиксели, и для того чтобы автоматом конвертировать dp в пиксели можно воспользоваться методом **getDimensionPixelSize().**

Немного о color

Все цвета хранятся внутри xml-файла res/values/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>
  <color name="textColor">#7e7e7e</color>
```

</**resources**>

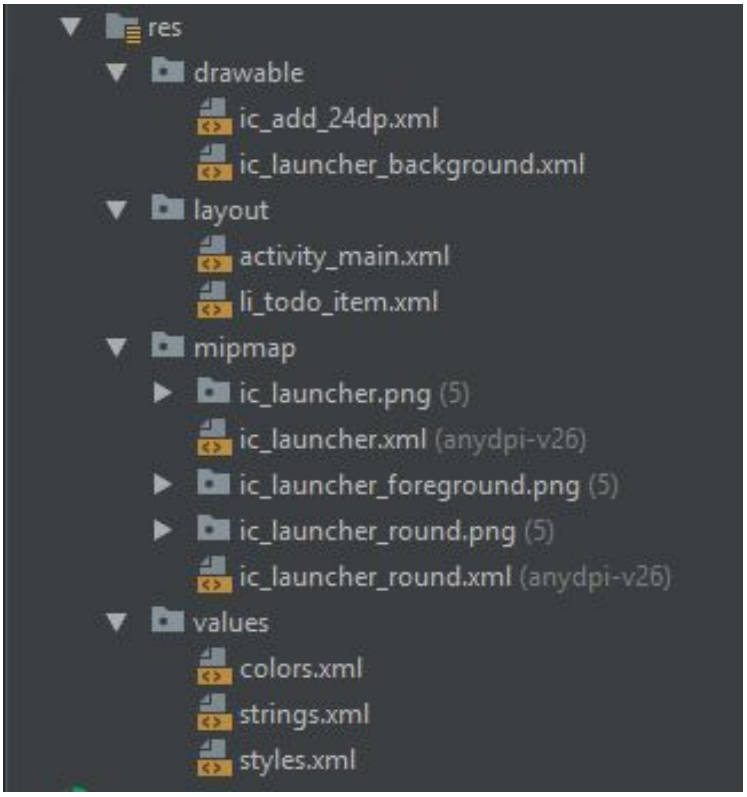
Это обращения к ресурсам цвета из XML:
android:background="@android:color/transparent"
android:textColor="@color/textColor"

Это обращения к ресурсам цвета из кода:
mTextView.setTextColor(ContextCompat.getColor(**this**, R.color.textColor));

Также есть метод **getColor()**, но он **deprecated (**перестанет поддерживаться**)** и поэтому мы не будем им пользоваться.

Пример ресурсов в проекте

Внутри приложения это выглядит так:



По умолчанию здесь есть каталоги не для всех типов ресурсов, которые используются в Android, но, при необходимости, мы можем добавить в папку **res/** нужный каталог, а затем поместить в него ресурс.

Ссылки на ресурсы

Чтобы получить доступ к любому ресурсу необходимо обратиться к нему по его id. Все id хранятся в файле R, который генерируется каждый раз при сборке на основе текущих ресурсов проекта. Редактировать его руками не имеет смысла. Помимо id, в файле R так же находятся и другие типы ссылок на ресурсы:
Типы ссылок ресурсов:

1. R.drawable (ему соответствует тип в XML drawable)
2. R.id (id)
3. R.layout (layout)
4. R.string (string)
5. R.attr (attr)
6. R.plural (plurals)
7. R.array (string-array)
8. R.dimen (dimensions)
9. и другие

Доступ к ресурсам

Есть два способа доступа к ресурсам:

1. В коде
2. В XML

Доступ к ресурсам в коде

Вспомним как мы использовали строковый ресурс в TextView:

```
textView.setText(R.string.app_name);
```

Внутри strings.xml она выглядит так:

```
<string name="app_name">MyApplication</string>
```

Через выражение R.string.your_text мы и ссылаемся на ресурс your_text, где string - тип ресурса, а your_text - имя ресурса.

Доступ к ресурсам в XML

Для задания значений для некоторых атрибутов и элементов XML можно использовать ссылку на существующий ресурс. Это требуется при создании файлов макета при указании строк и изображений для виджетов.

Например, при добавлении в Button необходимо использовать строковый ресурс для текста на кнопке:

```
<Button
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:text="@string/submit" />
```

Если в качестве параметра в атрибут text вы передадите hardcoded текст(это когда вы пишите текст без использования строкового ресурса), то чтобы перенести его в строковый ресурс вы можете воспользоваться комбинацией клавиш Alt+Enter.

Вообще, если вы используете какой-то текст внутри приложения, то желательно, чтобы он был не hardcoded, а хранился именно как строковый ресурс.

getResources()

Для получения ресурсов мы можем использовать метод getResources(), который возвращает объект android.content.res.Resources. Но чтобы получить сам ресурс, нам надо у полученного объекта Resources вызвать один из методов.

- getString(): получает строку из файла strings.xml
- getDimension(): получает числовое значение - ресурс dimen
- getDrawable(): получает графический файл
- getBoolean(): получает значение boolean
- и другие

Пример:

```
String text = getResources().getString(R.id.your_text);
```

Примеры работы с другими ресурсами из кода

Как обращаться к ресурсам из XML мы рассмотрели, теперь давайте узнаем как обращаться к ним из кода.

Простое обращение к layout ресурсу

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

Обращение к id ресурсу, находим наше ImageView с помощью метода findViewById

```
mExampleImageView = findViewById(R.id.iv_example);
```

Дополнительная информация

http://developer.alexanderklimov.ru/android/theory/resources.php