Ранее в предыдущих уроках я неоднократно упоминал такое понятие, как ресурсы. Давайте разберемся, что это такое.

В ресурсах хранятся следующие вещи:

* Файлы верстки
* Строки, массивы строк
* Цвета
* Размеры
* Стили
* Анимации
* Меню
* И несколько других типов данных

На самом деле, вы уже работали с ресурсами, когда меняли верстку нашего пока что единственного экрана. Возможно, вы заметили, что Android Studio показывает предупреждения, например, для атрибута android:text у TextView:

Hardcoded string "Hello", should use @string resource

Все потому что строки в Android принято хранить в ресурсах. Почему? Потому что такой подход позволяет легко обрабатывать различные конфигурации. Под конфигурациями я имею в виду такие вещи, как, к примеру, язык (англоязычному пользователю показываем английский текст, русскоязычному — русский), плотность пикселей на экране (для дисплеев с высоким разрешением используем картинки в большем разрешении), тип устройства (для телефонов используем одну верстку, для планшетов — другую) и т.д. Также использование ресурсов позволяет избежать дублирования: вы можете использовать одну и туже строку (или, к примеру, цвет) в нескольких местах.

Класс R

Вы уже могли заметить, что мы используем некий таинственный класс R при обращении, например, к View по идентификатору. Класс R (от слова Resources) генерируется автоматически при сборке проекта (Android Studio генерирует его динамически — а раньше нужно было пересобирать проект). При помощи него мы можем обращаться к ресурсам (идентификатор View — тоже ресурс). Если вы его откроете, то увидите что-то вроде этого:

public final class R {

public static final class anim {

public static final int abc\_fade\_in=0x7f010000;

public static final int abc\_fade\_out=0x7f010001;

public static final int abc\_grow\_fade\_in\_from\_bottom=0x7f010002;

public static final int abc\_popup\_enter=0x7f010003;

public static final int abc\_popup\_exit=0x7f010004;

public static final int abc\_shrink\_fade\_out\_from\_bottom=0x7f010005;

public static final int abc\_slide\_in\_bottom=0x7f010006;

public static final int abc\_slide\_in\_top=0x7f010007;

public static final int abc\_slide\_out\_bottom=0x7f010008;

public static final int abc\_slide\_out\_top=0x7f010009;

public static final int tooltip\_enter=0x7f01000a;

public static final int tooltip\_exit=0x7f01000b;

}

public static final class attr {

//

}

}

Изменять вручную этот файл нельзя. Точнее, можно, но он сгенерируется заново, и все ваши изменения пропадут.

Строки

Начнем с самого часто используемого: со строк. Строковые ресурсы хранятся в файле res/values/strings.xml. На самом деле, можно использовать несколько файлов, или переименовать файл — это не важно, главное, чтобы строки были внутри директории res/values.

Если вы откроете этот файл, то увидите, что там уже есть одна строка: это название нашего приложения.

<resources>

<string name="app\_name">Lessons</string>

</resources>

Новые строки добавляются внутри тега <resources/>. У строки есть атрибут name, в котором нужно обязательно указать уникальный идентификатор строки. Правила те же, что и для идентификаторов у View: латинские символы, цифры и знак подчеркивание, имя должно начинаться с буквы.

Внутри тега указываем, собственно, строку. Давайте перенесем значение атрибута android:text нашего TextView в строковые ресурсы. Добавьте новую строку в файл strings.xml, чтобы он выглядел вот так:

<resources>

<string name="app\_name">Lessons</string>

<string name="hello\_text">Hello</string>

</resources>

Теперь зададим эту строку в самом TextView (в файле activity\_main.xml). Для того, чтобы сослаться на строковый ресурс, нужно использовать следующий синтаксис: @string/name, где name — имя строки в strings.xml. Таким образом, TextView теперь выглядит вот так:

<TextView

android:id="@+id/hello\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="@string/hello\_text"

android:textSize="28sp" />

Форматирование строк

В Java-коде мы устанавливаем текст в TextView:

textView.setText("Hi, " + name);

На самом деле, тут тоже лучше использовать ресурсы, и не складывать строки, а использовать форматирование. В строковом ресурсе мы можем задать "плейсхолдеры", которые будут заменены на переданный нами в качестве аргумента текст (не обязательно текст. можно использовать, например, числа). Пример:

<string name="name\_text\_format">Hi, %1$s</string>

В данном случае %1$s будет заменен на текст. %1 — порядковый номер (если мы будем форматировать 2 аргумента, то второй будет начинаться с %2), а $s — формат. У нас это строка, если мы будем использовать число — то это будет $d, если число с плавающей точкой — то $f (можно также указать точность, например $.2f — 2 знака после запятой).

Итак, добавьте эту строку в strings.xml, и замените Java-код на следующий:

textView.setText(getString(R.string.name\_text\_format, name));

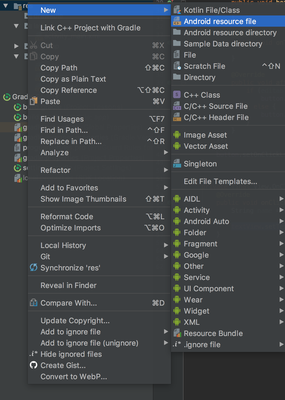
Метод getString() получает строку из ресурсов, и форматирует её, используя переданные аргументы. Если строку не нужно форматировать, то передается только идентификатор строки.

Запустите приложение, чтобы убедиться, что все работает.

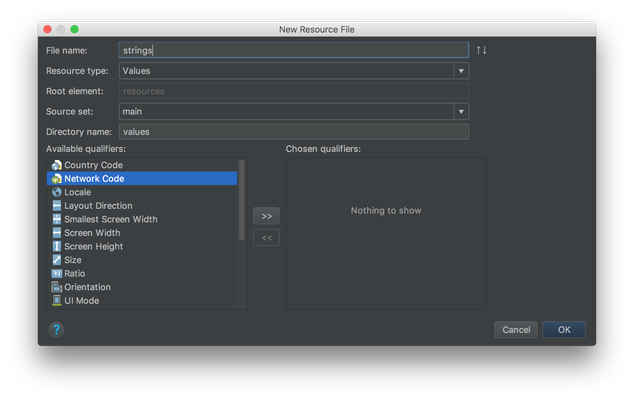
Локализация строк

Немаловажной задачей является локализация. Будет не очень хорошо, если вы покажете английский текст русскоязычному пользователю, не так ли?

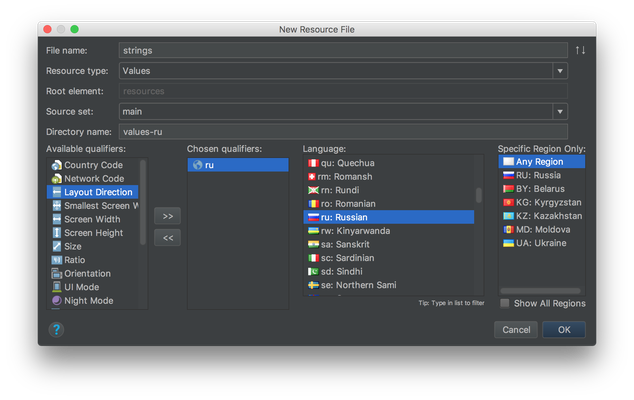
Давайте переведем текст на русский язык. Нажмите правой кнопкой на директорию res, во всплывающем окне выберите **New**, затем **Android resource file**:

[](https://skillberg.com/media/uploads/2017/11/17/a6d2aef98ab14ccc8c238ac5084af623.png)Создание нового файла ресурсов в Android Studio

В появившемся окне в поле **File name** введите "strings". **Resource type** — "values", **Source set** — "main", **Directory name** — "values":

[](https://skillberg.com/media/uploads/2017/11/17/237a826ac0e84d0bb794aaa8b503c0ef.png)Создание файла ресурсов со строками

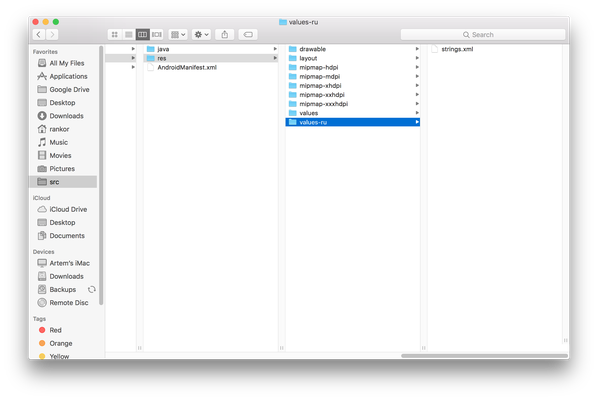
Снизу слева вы можете увидеть различные варианты, по которым мы можем дифференцировать конфигурацию пользовательского устройства. На данный момент нам нужен пункт **Locale**. Выберите его и нажмите на кнопку **>>**чуть правее. Появятся два дополнительных списка, **Language** и **Specific Region Only**. В списке **Language** выберите **ru: Russian**:

[](https://skillberg.com/media/uploads/2017/11/17/acdd1075d20741209a3029eecfb478a8.png)Выбор конфигурации для файла ресурсов

Регион оставьте **Any Region**. В таком случае любой пользователь, у которого в настройках устройства выбран русский язык, будет видеть текст на русском.

Выбор региона нужен для того, чтобы использовать ресурс только для пользователей с заданным языком из определенного региона. К примеру, если вы выберите Belarus, то строки на русском будут показываться только тем, кто живет в Беларуси, всем остальным они будут показываться на другом языке, даже если в настройках выбран русский язык.

После того, как вы нажмете на кнопку **ОК**, создастся новый файл:

[](https://skillberg.com/media/uploads/2017/11/17/30b955e38d73416393fcbe68a5014b67.png)Созданный файл строковых ресурсов

Как видите, он создался в директории res/values-ru, то есть values с русской локалью.

Файл пока почти что пуст: по умолчанию существующие строки добавляться в него не будут. Есть два варианта добавления строк для перевода:

1. Добавить строку вручную.
2. Воспользоваться встроенным редактором переводов.

В первом случае просто скопируйте строку из оригинального файла и вставьте в новый, после чего переведите. Например, я скопирую первые две строки — имя приложения и приветственный текст. Теперь мой файл выглядит так:

<?xml version="1.0" encoding="utf-8"?>

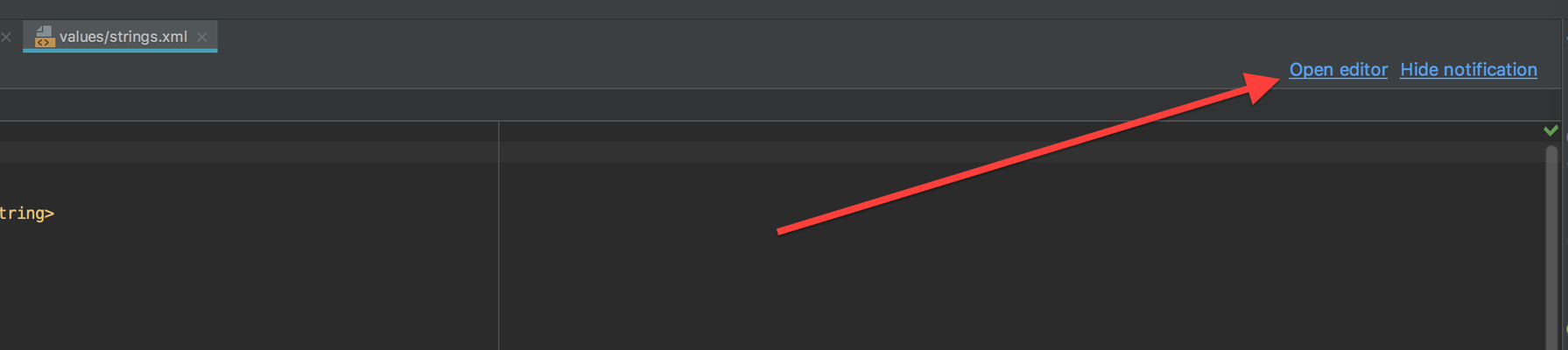
<resources>

<string name="app\_name">Уроки</string>

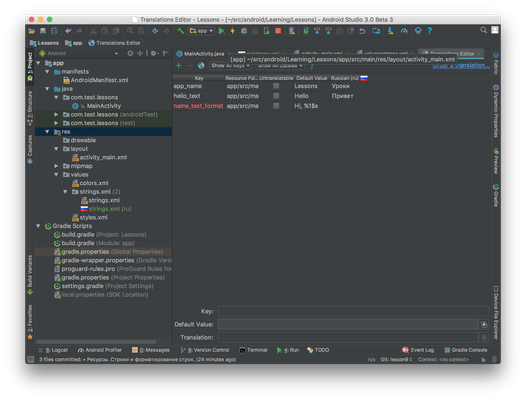
<string name="hello\_text">Привет</string>

</resources>

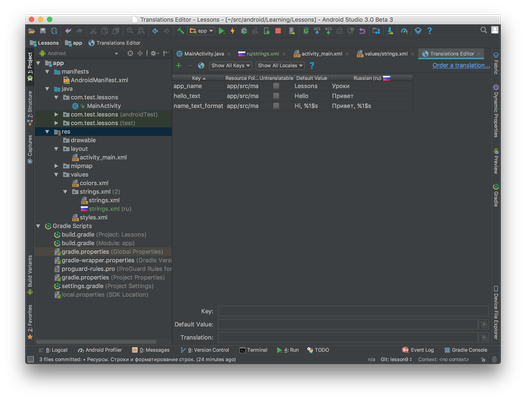
Второй вариант в теории должен быть удобнее, на практике же его реализация на данный момент оставляет желать лучшего. Откройте любой strings.xml. Вверху справа вы увидите кнопку **Open editor**:

[](https://skillberg.com/media/uploads/2017/11/17/15832196478f44ab9429db8299923ade.png)Кнопка "Open Editor"

Нажмите на нее. Откроется редактор строк:

[](https://skillberg.com/media/uploads/2017/11/17/a1de8e6b819c4b13a82ddd67f0e64b14.png)Редактор строк в Android Studio

Непереведённые строки подсвечиваются красным. Давайте переведем. Для этого поставьте курсор в соответствующую ячейку и введите текст. Получится вот так:

[](https://skillberg.com/media/uploads/2017/11/17/007eef0b8b0e4187a304eb7fa4404090.png)Редактор строк в Android Studio

А русский файл строк теперь выглядит так:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">Уроки</string>

<string name="hello\_text">Привет</string>

<string name="name\_text\_format">Привет, %1$s</string>

</resources>

Почему я сказал, что реализация оставляет желать лучшего, ведь это так удобно? Да, это удобно, когда вы работаете с такими маленькими строками и когда у вас всего две локали. Когда строки перестают умещаться в экран и количество локалей превышает 3, работа с этим инструментом превращается в ад, и приходится редактировать файлы строк вручную. На самом деле, это не так страшно, так что не пугайтесь и обязательно всегда переводите приложение :)

Точно так же локализация работает и для других ресурсов: создайте директорию в ресурсах (либо методом, показанным выше, либо вручную) с соответствующим названием и скопируйте туда нужные локализованные ресурсы.

По умолчанию (если для конфигурации пользователя не найдены ресурсы в приложении) берутся стандартные ресурсы. То есть, например, в нашем случае, пользователь с белорусской локалью увидит текст на английском.

Цвета

Цвета хранятся в директории res/values в файле colors.xml. Откройте этот файл:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<color name="colorPrimary">#3F51B5</color>

<color name="colorPrimaryDark">#303F9F</color>

<color name="colorAccent">#FF4081</color>

</resources>

Как видите, в нем уже есть три цвета. Эти цвета сгенерировала Android Studio при создании проекта.

Они нужны для стилизации приложения — вы могли заметить, что первый цвет — это цвет верхней части приложения, второй — цвет статусбара, а третьим подсвечивается поле для ввода текста.

Синтаксис добавления цвета очень похож на строки, только в качестве контента указывается не текст, а цвет в одном из следующих форматов:

* #RGB
* #RRGGBB
* #ARGB
* #AARRGGBB

Давайте добавим текст к нашему TextView. Пусть у него будет черный фон и белый текст. Сперва добавим два новых цвета:

<?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="colorHelloBackground">#000</color>

<color name="colorHelloText">#FFFFFF</color>

</resources>

Первый цвет — для фона, второй — для текста. Теперь откройте файл лэйаута. У каждого View есть атрибут android:background, в котором задается фон View. Он может быть задан не только цветом, но об этом потом. Добавьте цвет фона:

<TextView

android:id="@+id/hello\_tv"

android:layout\_width="wrap\_content"

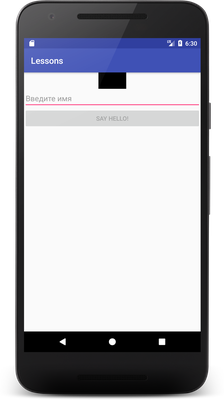
android:layout\_height="wrap\_content"

android:background="@color/colorHelloBackground"

android:text="@string/hello\_text"

android:textSize="28sp" />

Здесь colorHelloBackground — атрибут name у нашего цвета. Запустив приложение, вы увидите это:

[](https://skillberg.com/media/uploads/2017/11/17/3058581cbdaa4b8d9008bb3559e92b30.png)Цвет фона у TextView

Упс, текст пропал. Просто он был серым, и поэтому его не видно. Давайте сделаем цвет текста белым. За цвет текста у TextView (и его производных) отвечает атрибут android:textColor:

<TextView

android:id="@+id/hello\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

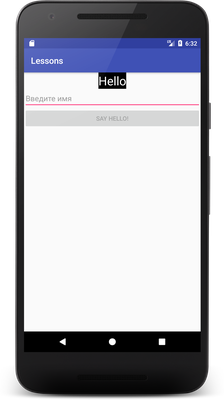
android:background="@color/colorHelloBackground"

android:text="@string/hello\_text"

android:textColor="@color/colorHelloText"

android:textSize="28sp" />

Теперь все в порядке:

[](https://skillberg.com/media/uploads/2017/11/17/085fbe5e393745c78ee97262087d0243.png)Цвет текста у TextView

Из кода мы можем получить цвет с помощью метода getColor(), передав ему идентификатор цвета. Например:

getColor(R.color.colorHelloBackground);

Локализация цветов работает так же, как и для строк: например, в values-ru создаете новый файл colors.xml и добавляете туда нужные цвета.

Размеры

Очень важный пункт, о котором вам обязательно нужно знать — это поддержка различных экранов у устройств.

В прошлых уроках я задавал размеры View в dp и размеры текста в sp. Пришло время разобраться, что это такое.

Предположим, у вас есть устройство с диагональю экрана 10 дюймов и разрешением 1080\*1920. Также у вас есть приложение, в котором есть View шириной в 100 пикселей. А у одного из ваших пользователей устройство с той же диагональю 10 дюймов, но с разрешением уже 2160\*3840, и то самое View в вашем приложении физически стало в два раза меньше по ширине. Упс! Что же делать?

Как известно, есть такое понятие, как плотность пикселей (DPI, Dots per Inch или PPI, Points per Inch). В примере выше у устройства вашего пользователя плотность пикселей в два раза выше, чем у вашего устройства. Для того, чтобы размеры элементов приложений не зависели от плотности пикселей на дисплее пользователя, в Android существует такой юнит измерения, как dp — Density-independent Pixel (пиксель, не зависящий от плотности). Один dp равен одному пикселю на дисплее с плотностью 160 dpi (для сравнения, сейчас есть устройства с 640 dpi). Система самостоятельно переводит dp в пиксели по следующей формуле: px = dp \* (dpi / 160). Например, на дисплее с плотностью 240 dpi 1 dp будет равен 1.5 физическим пикселям.

**Запомните**: никогда не указывайте размеры в пикселях! **Всегда** используйте dp.

Теперь об sp — это Scale-independed Pixel (пиксель, не зависящий от масштабирования). На самом деле, это то же самое, что и dp, однако результат будет автоматически масштабироваться системой в соответствии с настройками размера текста у пользователя.

**Запомните**: всегда указывайте размеры текста в sp. Пользователь должен иметь возможность увеличить или уменьшить текст.

Плотность пикселей

Android может автоматически использовать ресурсы, основываясь на плотности пикселей у дисплея устройства.

Существуют следующие варианты плотности пикселей:

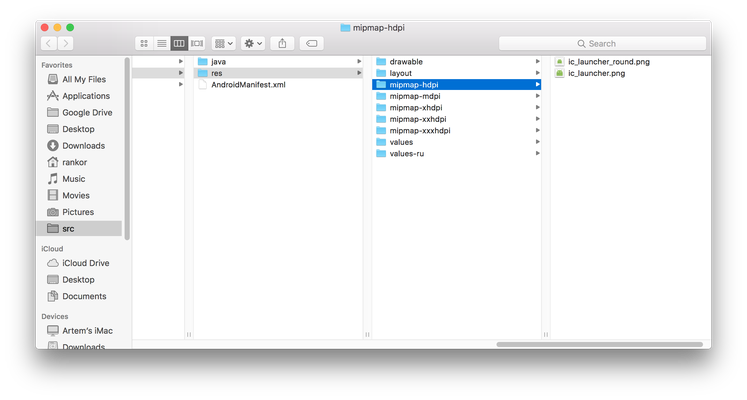
* ldpi — низкая плотность, ~120 DPI.
* mdpi — средняя плотность, ~160 DPI (базовая плотность).
* hdpi — высокая плотность, ~240 DPI.
* xhdpi — экстра-высокая плотность, ~320 DPI.
* xxhdpi — экстра-экстра-высокая плотность, ~480 DPI.
* xxxhdpi — экстра-экстра-экстра-высокая плотность, ~640 DPI.
* tvdpi — плотность для умных телевизоров, ~213 DPI.

Вы можете создать директорию для ресурсов, которые будут использоваться на устройствах с нужной вам плотностью, и положить туда необходимые ресурсы.

К примеру, если вы захотите показывать другой текст на устройствах с плотностью xxxhdpi, создайте директорию values-xxxhdpi и положите туда файл strings.xml с нужными вам строками.

Чаще всего это используется для картинок: для более высокой плотности дисплея используются картинки в высоком разрешении, для более низкой — в более низком разрешении.

Такие ресурсы уже есть в каждом приложении после создания проекта: откройте директорию res/ и увидите там директории mipmap-hdpi, mipmap-xhdpi и т.д. В них хранится иконка приложения в разных разрешениях:

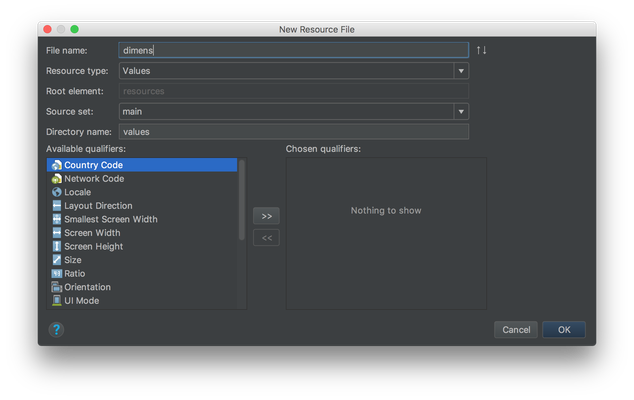
[](https://skillberg.com/media/uploads/2017/11/17/826afbff2a1140b48007abe788b39312.png)Иконки Android-приложения

Работу с изображениями мы рассмотрим в следующих уроках, а сейчас — о хранении размеров в ресурсах.

dimens.xml

Давайте перенесем размер текста TextView в файлы ресурсов.

Создайте новый файл ресурсов как было показано в примере с локализацией строк, но выберите вот такие параметры:

[](https://skillberg.com/media/uploads/2017/11/17/54dcfc31b7e14ac298b2debfe0de8cd8.png)Создание файла dimens.xml

Свежесозданный файл измените следующим образом:

<?xml version="1.0" encoding="utf-8"?>

<resources>

<dimen name="hello\_text\_size">28sp</dimen>

</resources>

Осталось лишь присвоить ресурс нашему TextView:

<TextView

android:id="@+id/hello\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:background="@color/colorHelloBackground"

android:text="@string/hello\_text"

android:textColor="@color/colorHelloText"

android:textSize="@dimen/hello\_text\_size" />

Готово!

На этом урок заканчивается, работу с остальными типами ресурсов мы разберем в следующих уроках.

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/LessonsAndroid/tree/lesson9)

ДОМАШНЕЕ ЗАДАНИЕ

1. Перенесите весь текст в ресурсы.<br/>2. Переведите весь текст на русский язык.<br/>3. Добавьте любой другой язык и переведите весь текст на него (можно через Google Translate).