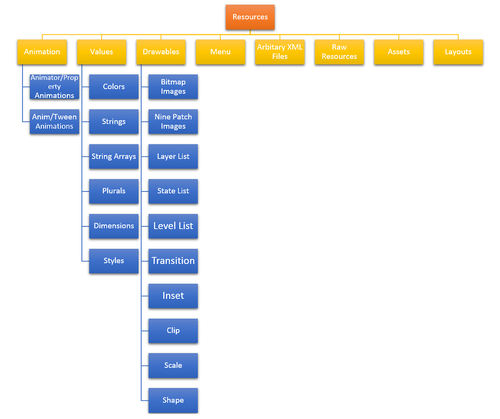
**Ресурсы**

**Основы**



На схеме представлена только часть ресурсов. Система постоянно обновляется и добавляются новые типы.

Ресурсы - один из основных компонентов, с которыми вам придётся работать очень часто. В Android принято держать некоторые объекты - изображения, строковые константы, цвета, анимацию, стили и т.п. за пределами исходного кода. Система поддерживает хранение ресурсов в отдельных файлах. Ресурсы легче поддерживать, обновлять, редактировать.

Каждое приложение на Android содержит каталог для ресурсов **res** и каталог для активов **assets**. Реальное различие между ресурсами и активами заключается в следующем:

* информация в каталоге ресурсов будет доступна в приложении через класс **R**, который автоматически генерируется средой разработки. То есть хранение файлов и данных в ресурсах (в каталоге **res**) делает их легкодоступными для использования в коде программы;
* для чтения информации, помещённой в каталог активов **assets** (необработанный формат файла), необходимо использовать **AssetManager** для чтения файла как потока байтов.

Android умеет динамически выбирать данные из дерева ресурсов, содержащие разные значения для разных конфигураций, языков и регионов. При запуске Android автоматически загрузит нужный ресурс, не требуя ни одной строчки кода.

Ресурсы в Android являются декларативными. В основном ресурсы хранятся в виде XML-файлов в каталоге res с подкаталогами **values, drawable-ldpi, drawable-mdpi, drawable-hdpi, layout**, но также бывают и другие типы ресурсов.

Для удобства система создаёт идентификаторы ресурсов и использует их в файле **R.java** (класс **R**, который содержит ссылки на все ресурсы проекта), что позволяет ссылаться на ресурсы внутри кода программы. Статический класс R генерируется на основе ваших заданных ресурсов и создаётся во время компиляции проекта. При создании класс содержит статические подклассы для всех типов ресурсов, для которых был описан хотя бы один экземпляр. Подробнее о классе **R** рассказано ниже.

Так как файл **R** генерируется автоматически, то не смысла его редактировать вручную, потому что все изменения будут утеряны при повторной генерации.

В общем виде ресурсы представляют собой файл (например, изображение) или значение (например, заголовок программы), связанные с создаваемым приложением. Удобств использования ресурсов заключается в том, что их можно изменять без повторной компиляции или новой разработки приложения. Имена файлов для ресурсов должны состоять исключительно из букв в нижнем регистре, чисел и символов подчёркивания.

Самыми распространёнными ресурсами являются, пожалуй, строки (string), цвета (color) и графические рисунки (bitmap). В приложении не рекомендуется применять жёстко написанные строки кода - вместо них следует использовать соответствующие идентификаторы, что позволяет изменять текст строкового ресурса, не изменяя исходного кода.

Начинающие программисты не всегда до конца правильно понимают процесс создания ресурсов. В Android используются два подхода - первый подход заключается в том, что ресурсы задаются в файле, при этом имя файла значения не имеет. Второй подход - ресурс задаётся в виде самого файла, и тогда имя файла уже имеет значение (при этом нужно учитывать определённые нюансы).

Общая структура каталогов, содержащих ресурсы выглядит следующим образом:

| /res/values/strings.xml  /colors.xml  /dimens.xml  /attrs.xml  /styles.xml  /drawable/\*.png  /\*.jpg  /\*.gif  /\*.9.png  /anim/\*.xml  /layout/\*.xml  /raw/\*.\*  /xml/\*.xml  /assets/\*.\*/\*.\* |
| --- |

Только в **assets** может располагаться любой набор подкаталогов разной вложенности. Файлы, находящиеся в любом другом каталоге, размещаются именно на уровне этого каталога и не глубже.

Перечисление основных ресурсов Android

| **Тип ресурса** | **Размещение** | **Описание** |
| --- | --- | --- |
| Цвета | res/values/имя\_файла | Идентификатор цвета, указывающий на цветовой код. ID таких ресурсов выражаются в R.java как R.color.\*. XML-узел: /resources/color |
| Строки | res/values/имя\_файла | Строковые ресурсы. В их число также входят строки в формате java и html. ID таких ресурсов выражаются в R.java как R.string.\*. XML-узел: resources/string. Можно использовать дополнительное форматирование при помощи стандартных html-тегов <b>, <i> и <u>. Методы, которые будут обрабатывать строковые ресурсы с HTML-форматированием, должны уметь обрабатывать эти теги. |
| Меню | /res/menu/имя\_файла | Меню в приложении можно задать как XML-ресурсы. |
| Параметры | /res/values/имя\_файла | Представляет собой параметры или размеры различных элементов. Поддерживает пиксели, дюймы, миллиметры, не зависящие от плотности экрана пиксели (dip) и пиксели, не зависящие от масштаба. ID таких ресурсов выражаются в R.java как R.dimen.\*. XML-узел: resources/dimen |
| Изображения | /res/drawable/ваши\_файлы | Ресурсы-изображения. Поддерживает форматы JPG, GIF, PNG (самый предпочтительный) и др. Каждое изображение является отдельным файлом и получает собственный идентификатор, который формируется по имени файла без расширения. Такие ID ресурсов представлены в файле R.java как R.drawable.\*. Система также поддерживает так называемые растягиваемые изображения (stretchable image), в которых можно менять масштаб отдельных элементов, а другие элементы оставлять без изменений. |
| Отрисовываемые цвета | /res/values/ваш\_файл или /res/drawable/ваши\_файлы | Представляет цветные прямоугольники, которые используются в качестве фона основных отрисовываемых объектов, например точечных рисунков. Поддержка такой функции обеспечивается тегом значения drawable, находящимся в подкаталоге значений. Такие id ресурсов выражаются в файле R.java как R.drawable.\*. ХМL-узел для такого файла: /resources/drawable. В Android при помощи специальных ХМL-файлов, расположенных в /res/drawable, также поддерживаются скруглённые и градиентные прямоугольники. Корневым ХМL-тегом для drawable является <shape>. Идентификаторы таких ресурсов выражаются в файле R.java как R.drawable.\*. В таком случае, каждое имя файла преобразуется в уникальный id отрисовываемого объекта |
| Анимация | /res/anim/ваш\_файл | Android может выполнить простую анимацию на графике или на серии графических изображений. Анимация включает вращения, постепенное изменение, перемещение и протяжение. |
| Произвольные XML-файлы | /res/xml/\*.xml | В Android в качестве ресурсов могут использоваться произвольные XML-файлы. Они компилируются в aapt. Идентификаторы таких ресурсов также выражаются в файле R.java как R.xml.\* |
| Произвольные необработанные ресурсы | /res/raw/\*.\* | Любые нескомпилированные двоичные или текстовые файлы, например, видео. Каждый файл получает уникальный id ресурса. Идентификаторы таких ресурсов выражаются в файле R.java как R.raw.\* |
| Произвольные необработанные активы | /assets/\*.\*/\*.\* | Можно использовать произвольные файлы в произвольно названных каталогах, которые находятся в подкаталоге /assets. Это не ресурсы, а просто необработанные файлы. В этом каталоге, в отличие от /res, подкаталоги могут располагаться на любой глубине. Для таких файлов не создаются идентификаторы ресурсов. При работе с ними нужно использовать относительное имя пути, начиная с /assets, но не указываея этого каталгоа в имени пути |

**Идентификаторы**

Этот тип ресурсов формируется, как правило, автоматически, и программисты даже не обращают на него внимания. Когда вы размещаете новый элемент на форме, с которым будете взаимодействовать в программе, то ему нужно присвоить идентификатор. Как правило, это происходит в виде **@+id/editText** (часто это происходит автоматически). Знак плюса обозначает, что если идентификатора не существует, то его нужно создать в классе R. В программе вы можете обращаться к элементу **R.id.editText**.

Но можно заранее создать ресурс типа **item** для задания **id**, не связанного ни с каким конкретным ресурсом:

| <resources>      <item type="id" name="text"/>  </resources> |
| --- |

Здесь **type** описывает тип ресурса, в данном случае **id**. Когда **id** будет установлен, будет работать и следующее определение **View**:

| <TextView android:id="@id/textView">      </TextView> |
| --- |

Обычно идентификаторы размещают в отдельном файле **res/values/ids.xml**

**Строковые ресурсы**

Строковые ресурсы помогают упростить процесс создания локализованных версий. Строковые ресурсы обозначаются тегом **<string>.**,

При разработке приложений вы видели, что система создает файл **strings.xml**, в котором хранились строки для заголовка приложения и выводимого сообщения. Вы можете редактировать данный файл, добавляя новые строковые ресурсы. А также вы можете создать новые файлы, которые будут содержать строковые ресурсы. Все эти файлы должны находиться в подкаталоге **/res/values**. Но в большинстве случаев программисты используют для строковых ресурсов стандартное имя **strings.xml**. Типичный файл выглядит следующим образом.

| <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="hello">Здравствуй, Мир!</string>      <string name="app\_name">Hello, World</string>  </resources> |
| --- |

При создании или обновлении файла со строковыми ресурсами среда разработки автоматически создаёт или обновляет класс **R.java**, сообщая уникальные ID для определённых в файле строковых ресурсов (Независимо от количества файлов ресурсов, в проекте содержится только один файл **R.java**). Если открыть данный файл, то можно найти там наши ресурсы в следующем виде:

| public static final int hello = 0x7f040000; public static final int app\_name = 0x7f040001; |
| --- |

В принципе достаточно запомнить, что **R.java** создаёт внутренний статический класс как пространство имён, в котором содержатся ID строковых ресурсов. Два **static final int**, которые используются в переменных **hello** и **app\_name**, являются идентификаторами ресурсов и соответствуют соответствующим строковым ресурсам. Вы можете использовать данные идентификаторы в своем исходном коде, используя следующий формат - **R.string.hello**.

Обратите внимание, что сгенерированные ID указывают на **int**, а не на **String**. Android при необходимости самостоятельно подставляет вместо **int** нужные строки.

Обычно принято хранить строковые ресурсы в файле **strings.xml**, но вы можете использовать несколько файлов. Главное, чтобы XML-файл имел необходимую структуру и находился в подкаталоге **res/values**.

Структура файла для строковых ресурсов довольно проста. Имеется корневой узел <resources>, за которым следуют один или несколько дочерних элементов <string>. Каждый элемент <string> в свою очередь имеет свойство **name**, которое в файле **R.java** представляет собой атрибут **id**.

Если вы создаёте несколько файлов с ресурсами, то следите за уникальностью создаваемых имён. Не выйдет ничего хорошего, если в двух файлах будет одна и та же переменная **app\_name**.

Запомните, что пробелы в начале и в конце строк обрезаются. Если вам так нужны пробелы, то разместите строку в кавычках и строка будет выводиться как есть. Также можно попробовать использовать код **\u0020** вместо пробела.

Такая же проблема и с несколькими пробелами внутри строки - будет выводиться только один пробел (напоминает поведение в html-документе).

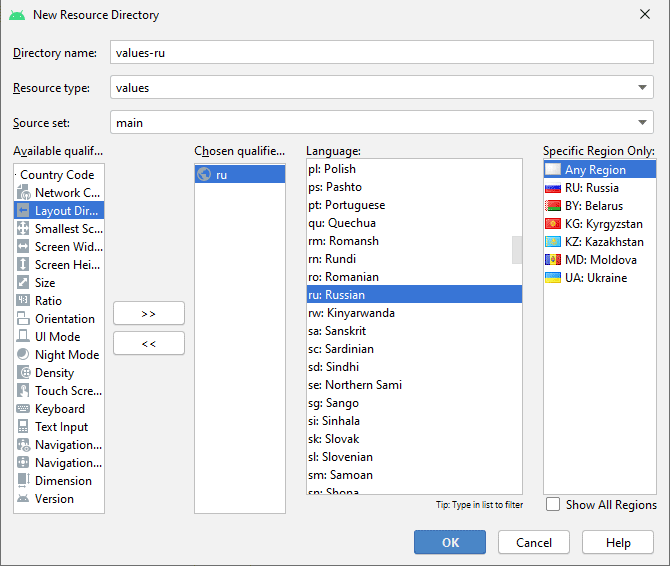
**Локализация.**

Не будем менять строчку **Hello, World**, которая будет считаться строкой по умолчанию, а создадим новый локализованный ресурс. Это даст нам дополнительное преимущество, если пользователь запустит приложение на телефоне с английской локалью, то он увидит текст на знакомом ему языке. Если приложение запустит русский пользователь, то он увидит текст на русском языке.

Напишем приложение, которое будет поддерживать английскую и русскую локализации. По умолчанию будет использоваться английская локализация. В этом случае, если на телефоне не будет найден нужный локализованный ресурс, то будет применяться английский вариант.

Создадим новый проект "LocaleApp", но на этот раз мы не будем трогать файл **strings.xml** в каталоге **res/values**.

Нажимаем правой кнопкой мыши по папке **res** и выбираем **New | Android resource directory**. В диалоговом окне в левой части **Available qualifiers:** выбираем пункт **Locale** и переносим его в правую часть **Chosen qualifiers:** с помощью кнопки с двумя стрелками вправо. В появившейся третьей колонке выбираем нужные языки, например, русский. Вы увидите, что в поле **Directory name** автоматически появится нужное название папки. Дополнительно вы можете указать и регион в колонке **Specific Region Only**



В режиме **Android** вы можете не увидеть созданную папку, поэтому временно переключитесь в режим **Project**. Скопируйте файл **res/values/strings.xml** и вставьте его в новую папку. Можете вернуться в прежний режим. Если папка не пуста, то она уже видна.

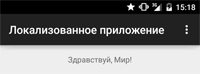
Измените содержание string.xml английской версии

| <**resources**>  <**string name="app\_name"**>Locale Application</**string**>  <**string name="hello\_world"**>Hello world!</**string**> </**resources**> |
| --- |

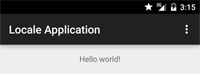
Измените содержание string.xml (ru) русской версии

| *<?***xml version="1.0" encoding="utf-8"***?>* <**resources**>  <**string name="app\_name"**>Локализованное приложение</**string**>  <**string name="hello\_world"**>Здравствуй, Мир!</**string**> </**resources**> |
| --- |

Запустим приложение и увидим, что текст выводится на русском, если у вас на эмуляторе или реальном устройстве выбраны русские настройки.



Как перейти на американскую локаль? Идём в **Настройки→Язык и клавиатура→Выбрать язык**. Выбираем английский язык **English (United States)**. Снова запускаем приложение в эмуляторе или на телефоне. Теперь наше приложение будет выглядеть так:



Ресурсы из каталога **res/values** считаются ресурсами по умолчанию. Они отображаются в том случае, когда система не найдёт ресурсы для текущей локали устройства. В нашем случае, при выборе любого языка, кроме русского будет использоваться **res/values**. Если мы захотим разделить англичан и американцев, тогда придётся создавать два каталога: **res/values-en-rUK** и **res/values-en-rUS**. Обратите внимание, что перед кодом второй части ставится буква 'r'. Иными словами, мы определяем не только язык, но и регион.

Проведем несколько экспериментов. Удалим из файла **res/values-ru/strings.xml** строковый ресурс **hello\_world** с текстом **Здравствуй, Мир!**. Сохраним изменения и запустим проект на устройстве с русским языком. Платформа заменила отсутствующий в локализации ресурс дефолтным английским.

Продолжим опыты и поступим наоборот - удалим теперь ресурс из файла **res/values/strings.xml**. На устройстве выберем какой-нибудь другой язык, например, итальянский. В этом случае вы увидите что-то непонятное, скорее всего вы получите сообщение об ошибке.

Отсюда можно сделать вывод, что вам необходимо следить за ресурсами по умолчанию. Старайтесь проконтролировать, чтобы ваше приложение всегда содержало все необходимые ресурсы по умолчанию, а уже потом можете начинать локализацию приложения. Для этого вам достаточно будет скопировать файл в новый подкаталог и отредактировать его.

На самом деле локализовать можно не только строковые ресурсы. В разных странах принято рисовать какой-нибудь объект в соответствии со своими традициями, например, изображение почтового ящика. В этом случае вы можете создать каталог **/res/drawable-de**, в котором будут находиться изображения, предназначенные для немцев.

В коде вы обращаетесь к нужным ресурсам следующим образом:

| String hello = getResources().getString(R.string.***hello\_world***); |
| --- |

// или так

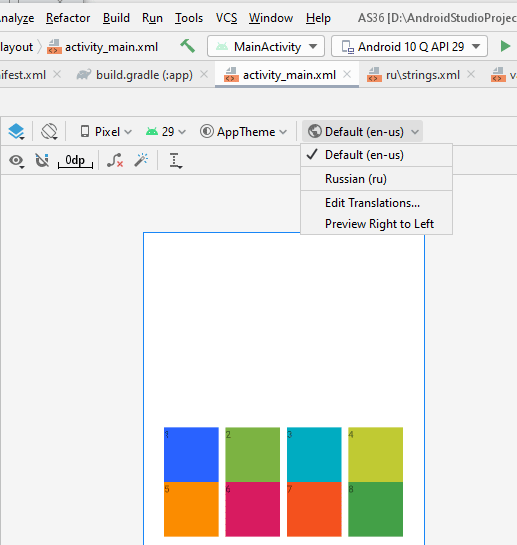
| TextView textView = **new** TextView(**this**); textView.setText(R.string.***hello\_world***); |
| --- |

А вот так указывается какой конкретно язык необходимо использовать

| Locale locale = **new** Locale(**"en"**);  Locale.setDefault(locale);  Configuration configuration = **new** Configuration();  configuration.locale = locale;  getBaseContext().getResources().updateConfiguration(configuration, **null**); *// выводим английский текст на русской локали устройства* setTitle(R.string.***app\_name***); |
| --- |

**Просмотр и создание локализованных строк в режиме дизайна**

Чтобы увидеть, как меняется текст в различных локализованных версиях вашего приложения, необязательно запускать приложение в эмуляторе. Можно в студии на панели инструментов щёлкнуть на значке глобуса и выбрать из выпадающего списка нужный вам пункт (он появится после того, как вы создадите необходимые локализованные ресурсы), и все локализованные строчки сразу поменяются на экране активности.



Можно сразу создать локализованный файл через удобный мастер. Там же в меню есть пункт **Edit Translations...**, который выводит диалоговое окно со списком уже имеющихся строковых ресурсов из вашей программы

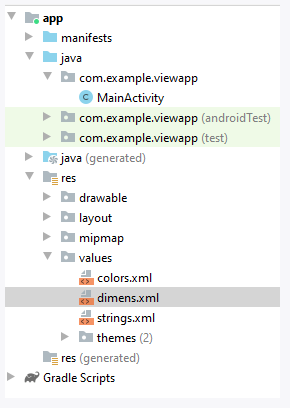
**Ресурсы dimension**

Определение размеров должно находиться в папке **res/values** в файле с любым произвольным именем. Общий синтаксис определения ресурса следующий:

| <?xml version="1.0" encoding="utf-8"?>  <resources>      <dimen name="имя\_ресурса">используемый\_размер</dimen>  </resources> |
| --- |

Как и другие ресурсы, ресурс dimension определяется в корневом элементе <resources>. Тег <dimen> обозначает ресурс и в качестве значния принимает некоторое значение размера в одной из принятых единиц измерения (dp, sp, pt, px, mm, in).

Так, добавим в Android Studio в папку **res/values** новый элемент **Values Resources File**, который назовем **dimens.xml**.



Определим в нем следующее содержимое:

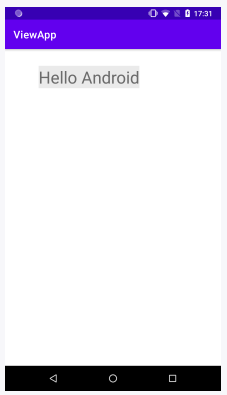
| <?xml version="1.0" encoding="utf-8"?>  <resources>      <dimen name="horizontal\_margin">64dp</dimen>      <dimen name="vertical\_margin">32dp</dimen>      <dimen name="text\_size">32sp</dimen>  </resources> |
| --- |

Здесь определены два ресурса для отступов horizontal\_margin и vertical\_margin, которые хранят соответственно значения 64dp и 32dp, и ресурс text\_size, который хранит высоту шрифта - 32sp. Названия ресурсов могут быть произвольными.

Используем ресурс в файле activity\_main.xml:



Ресурсы dimension используются для таких атриубтов визуальных элементов, которые в качестве значения требуют указание числового значения. Например, атриубут android:layout\_height или android:textSize. Для получения ресурса в xml после "@dimen/" указывается имя ресурса.



Для получения ресурсов в коде java применяется метод **getDimension()** класса Resources. Например, определим в коде Java аналогичный визуальный интерфейс:



При программном получении ресурсов dimen с помощью метода getDimension() следует учитывать, что он возвращает значения в физических пикселях, даже если в файле ресурсов мы определили значения в других величинах (32sp, 64dp). В большинстве случае это не вызовет каких-то трудностей, поскольку большинство методов в Java принимают именно пиксели, а не dp или другие единицы, как например, метод setMargins(), который устанавливает отступы.

Однако метод setTextSize() принимает именно **sp**, поэтому с помощью дополнительного параметра необходимо указать, что в данном случае имеются в виду физические пиксели, а не sp:

| textView.setTextSize(TypedValue.COMPLEX\_UNIT\_PX, textSize); |
| --- |

Либо, как вариант, с помощью класса **TypedValue** программно перевести физические пиксели в sp или другую единицу измерения.

**Задание для самостоятельной работы:**

1. Создайте приложение по заданному макету интерфейса используя везде где это возможно ресурсы, настройте автоматическую локализацию для английского языка.

