Ресурсы в Андроид

Omnia mea mecum porto

(Все свое ношу с собой)

Содержание

- Предоставление ресурсов
- Группирование типов ресурсов
- Имена каталогов ресурсов
- Предоставление альтернативных ресурсов
- Создание псевдонимов ресурсов
- Алгоритм выбора ресурсов
- Доступ к ресурсам

Предоставление ресурсов

Обязательно необходимо экспортировать из кода ресурсы приложения, такие как изображения и строки, для последующей их независимой обработки.

Следует также обеспечить альтернативные ресурсы для определенных конфигураций устройств, группируя их в каталогах ресурсов со специальными именами.

В режиме выполнения Android использует соответствующие ресурсы с учетом текущей конфигурации. Например, можно предоставлять другой макет пользовательского интерфейса в зависимости от размера экрана или различные строки в зависимости от настройки языка.

Пример: локаль

Для профессионально написанного приложения лучше использовать локализацию.

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.

не будем трогать файл **strings.xml** в каталоге **res/values**

```
<?xml version="1.0" encoding="utf-8"?> <resources>
 <string name="app_name">Locale Application</string>
<string name="hello_world">Hello world!</string>
</resources>
```

русской локализации необходимо создать новый подкаталог **values-ru** в том же каталоге **res: res/values-ru/strings.xml**

```
<?xml version="1.0" encoding="utf-8"?> <resources>
<string name="app_name">Локализованное приложение</string>
<string name="hello_world">Здравствуй, Мир!</string>
</resources>
```

Группирование типов ресурсов

Следует поместить ресурсы каждого типа в определенный подкаталог каталога res/ вашего проекта. В качестве примера приведена иерархия файлов для простого проекта:

- MyProject/
- src/
- MyActivity.java
- res/
- drawable/
- graphic.png
- layout/
- main.xml
- info.xml
- mipmap/
- icon.png
- values/
- strings.xml

Как видно в этом примере, каталог res/ содержит все ресурсы (в подкаталогах): ресурс-изображение, два ресурса-макета, каталоги mipmap/ для значков запуска и файл строк.

Имена каталогов ресурсов

`animator/`	Файлы XML, которые определяют анимации свойств.
`anim/`	Файлы XML, которые определяют анимации преобразований. (Анимации свойств также можно сохранять в этом каталоге, но для анимаций свойств предпочтительнее использовать каталог `animator/`, чтобы различать эти два типа).
`color/`	Файлы XML, которые определяют список состояний цветов. См. раздел Ресурс списка состояний цветов
`drawable/`	Файлы растровых изображений (`.png`, `.9.png`, `.jpg`, `.gif`) или файлы XML, которые составляют следующие подтипы графических ресурсов: · Файлы растровых изображений · Файлы из девяти фрагментов (растровые изображения с возможностью изменения размера) · Списки состояний · Формы · Графические анимации · Другие графические элементы

mipmap/ Графические файлы для значков запуска с различной плотностью. Подробные сведения об управлении значками запуска с помощью папок `mipmap/` см. в разделе Обзор управления проектами.

layout/ Файлы XML, которые определяют макет пользовательского интерфейса. См. раздел Ресурсы макетов.

menu/ Файлы XML, которые определяют меню приложения, такие как меню параметров, контекстные меню или вложенные меню. См. раздел Ресурсы меню.

raw/ Произвольные файлы для сохранения в исходной форме. Чтобы открыть эти ресурсы с помощью`InputStream`, вызовите `Resources.openRawResource()` с идентификатором ресурса, который имеет вид `R.raw.<em>filename</em>`.

Однако, если требуется получить доступ к исходным именам файлов и иерархии файлов, можно сохранять некоторые ресурсы в каталоге `assets/` (вместо каталога `res/raw/`). Файлы в каталоге `assets/` не получают идентификатора ресурса, поэтому их чтение возможно только с помощью `AssetManager`.

values/ Файлы XML, которые содержат простые значения, такие как строки, целые числа и цвета. Тогда как XML-файлы ресурсов в других подкаталогах каталога `res/` определяют отдельные ресурсы на базе имени файла XML, файлы в каталоге `values/` описывают несколько ресурсов. Для файла в этом каталоге каждый дочерний элемент элемента `<resources>` определяет один ресурс.

Например, элемент `<string>` создает ресурс `R.string`, а элемент `<color>` создает ресурс `R.color`.

Так как каждый ресурс определяется с помощью своего собственного элемента XML, можно назначать имя файла по своему усмотрению и помещать ресурсы разных типов в один файл. Тем не мене, может появиться необходимость поместить ресурсы отдельных типов в разные файлы. Например, ниже приведены соглашения для имен файлов ресурсов, которые можно создать в этом каталоге:

- arrays.xml для ресурсов-массивов (массивы с указанием типа)
- colors.xml для значений цветов
- dimens.xml для значений единиц измерений
- strings.xml для строковых значений
- styles.xml для стилей.

См. разделы Строковые ресурсы, Ресурсы стиля и Дополнительные типы ресурсов.

xml/ Произвольные XML-файлы, которые можно читать в режиме выполнения вызовом метода `Resources.getXML()`. Здесь должны сохраняться различные файлы конфигурации XML, например, конфигурация с возможностью поиска.

Предоставление альтернативных ресурсов

Ресурсы, сохраненные в подкаталогах, которые описаны в таблице являются ресурсами **«по умолчанию».**

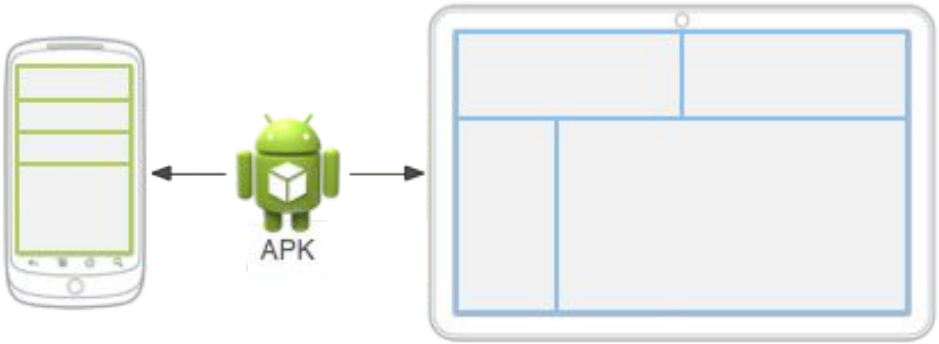
Таким образом, эти ресурсы определяют дизайн и содержимое приложения по умолчанию.

Однако различные типы устройств Android могут вызывать различные типы ресурсов.

Почти каждое приложение должно предоставлять альтернативные ресурсы, чтобы поддерживать определенные конфигурации устройств.

Например, необходимо включить альтернативные графические ресурсы для экранов с разной плотностью растра и альтернативные ресурсы для разных языков.

В режиме выполнения Android определяет конфигурацию устройства и загружает соответствующие ресурсы для приложения.



Чтобы указать альтернативы для конкретных конфигураций набора ресурсов, выполните следующие действия:

Создайте новый каталог в каталоге res/ с именем следующего вида <имя_ресурса>-<квалификатор_конфигурации>.

<resources_name> – имя каталога соответствующих ресурсов по умолчанию (определено в таблице 1).

<qualifier> – имя, которое указывает определенную конфигурацию, для которой должны использоваться эти ресурсы (определено в таблице 2).

Можно добавлять несколько квалификаторов *<qualifier>*.

Разделяйте их знаком дефиса.

Сохраните соответствующие альтернативные ресурсы в этом новом каталоге.

Файлы ресурсов должны иметь имена, точно совпадающие с именами файлов ресурсов по умолчанию.

В качестве примера здесь приведено несколько ресурсов по умолчанию и альтернативных ресурсов:

res/

drawable/

icon.png

background.png

drawable-hdpi/

icon.png

background.png

Комментарий

Квалификатор hdpi указывает, что ресурсы в этом каталоге предназначены для устройств, оснащенных экраном высокой плотности.

Изображения в каждом из этих каталогов для графических объектов имеют размер для определенной плотности экрана, но имена файлов полностью совпадают.

Таким образом, идентификатор ресурса, который указывает на изображение icon.png или background.png, всегда одинаков, но Android выбирает версию каждого ресурса, которая оптимально соответствует текущему устройству, сравнивая информацию о конфигурации устройства с квалификаторами в имени каталога ресурсов.

Android поддерживает несколько квалификаторов конфигурации, позволяя добавлять несколько квалификаторов к одному имени каталога, разделяя квалификаторы дефисом.

В таблице 2 перечислены допустимые квалификаторы конфигурации в порядке приоритета — если используется несколько квалификаторов для каталога ресурсов, необходимо добавлять их к имени каталога в том порядке, в котором они перечислены в таблице.

Квалификаторы для каталога ресурсов

Конфигурация	Значения квалификатора	Описание
MCC и MNC	Примеры: mcc310 mcc310-mnc004 mcc208-mnc00 и т. д.	Код страны для мобильной связи (MCC), за которым может следовать код сети мобильной связи (MNC) из SIM-карты устройства. Например, mcc310 – код США для любого поставщика услуг, mcc310-mnc004 – код США для Verizon и mcc208-mnc00 – код Франции для Orange. Если в устройстве используется радиосвязь (телефон GSM), значения MCC и MNC добываются из SIM-карты. Можно также использовать только код MCC (например, для включения в приложении разрешенных в стране ресурсов). Если требуется указать только язык, используйте квалификатор *язык и регион* (обсуждается ниже). Если принято решение использовать квалификатор MCC и MNC, следует делать это с осторожностью и проверить корректность его работы.

Квалификаторы для каталога ресурсов

Язык и регион	Примеры:
	`en`
	`fr`
	`en-rUS`
	`fr-rFR`
	`fr-rCA`
	и т. д.
		Язык задается двухбуквенным кодом языка ISO 639-1, к которому можно добавить двухбуквенный код региона ISO 3166-1-alpha-2 (которому предшествует строчная буква "`r`").
		
		Коды *не* зависят от регистра; префикс `r` служит для обозначения кода региона. Нельзя указывать только код региона.
		
		Он может измениться за время работы приложения, если пользователь изменяет свой язык в системных настройках. В разделе Обработка изменений в режиме выполнения содержится информация о воздействии таких изменений на приложение во время выполнения.
		
		В разделе Локализация приведено полное руководство по локализации приложения для других языков.
		
		См. также поле конфигурации `locale`, которое

Квалификаторы для каталога ресурсов

Ориентация экрана	`port` `land`	• `port`: Устройство в портретной (вертикальной) ориентации • `land`: Устройство в книжной (горизонтальной) ориентации Ориентация может измениться за время работы приложения, если пользователь поворачивает экран. В разделе Обработка изменений в режиме выполнениясодержится информация о воздействии таких изменений на приложение во время выполнения. См. также поле конфигурации `orientation`, которое указывает текущую ориентацию устройства.

Правила квалификатора имени

Здесь приведены некоторые правила использования имен квалификаторов:

Можно указать несколько квалификаторов для одного набора ресурсов, разделяя их дефисами.

Например, drawable-en-rUS-land применяется к устройствам в США, на английском языке в альбомной ориентации.

Квалификаторы должны идти в том же порядке, в котором они перечислены в таблице

Например:

Неправильно: drawable-hdpi-port/

Правильно: drawable-port-hdpi/

Нельзя использовать вложенные каталоги альтернативных ресурсов. Например, нельзя иметь каталог res/drawable/drawable-en/.

Значения не зависят от регистра букв. Компилятор ресурсов преобразует имена каталогов в нижний регистр перед обработкой, чтобы избежать проблем в файловых системах, не учитывающих регистр.

Прописные буквы в именах служат исключительно для удобочитаемости.

Правила квалификатора имени

Поддерживается только одно значение квалификатора каждого типа.

Например, если требуется использовать одинаковые графические файлы для испанского и французского языков, *нельзя* создавать каталог с именем drawable-rES-rFR/.

Вместо этого необходимо создать два каталога ресурсов, например, drawable-rES/ и drawable-rFR/, которые содержат соответствующие файлы.

Однако не обязательно фактически копировать одинаковые файлы в оба каталога. Вместо этого можно создать псевдоним для ресурса.

После сохранения альтернативных ресурсов в каталоги с именами этих квалификаторов Android автоматически применяет ресурсы в приложении на основе текущей конфигурации устройства.

При каждом запросе ресурсов Android проверяет каталоги альтернативных ресурсов, которые содержат файл запрошенного ресурса, затем находят наиболее подходящий ресурс (обсуждается ниже).

Если нет альтернативных ресурсов, которые соответствуют конкретной конфигурации устройства, Android использует ресурсы по умолчанию (набор ресурсов для конкретного типа ресурсов, которые не содержат квалификатора конфигурации).

Создание псевдонимов ресурсов

Ресурс, предназначенный для нескольких конфигураций устройства (но не являющийся ресурсом по умолчанию), следует помещать только в один каталог альтернативных ресурсов.

Вместо этого можно (в некоторых случаях) создать альтернативный ресурс, действующий в качестве псевдонима для ресурса, сохраненного в каталоге ресурсов по умолчанию.

Например, представьте, что имеется значок приложения, icon.png, и требуется иметь уникальные версии этого значка для разных языков.

Однако в двух языках, канадском английском и канадском французском, требуется использовать одинаковую версию.

Можно предположить, что требуется скопировать одно изображение в каталоги ресурсов для обоих языков, но это неверно.

Вместо этого можно сохранить изображение для обоих языков, как icon_ca.png (любое имя, кроме icon.png), и поместить его в каталог по умолчанию res/drawable/.

Затем создайте файл icon.xml в каталогах res/drawable-en-rCA/ и res/drawable-fr-rCA/ который ссылается на ресурс icon_ca.png с помощью элемента <bitmap>.

Это позволяет хранить только одну версию файла PNG и два маленьких файла XML, которые указывают на него.

Пример файла XML

Графические объекты

Чтобы создать псевдоним для существующего графического объекта, используйте элемент <drawable>. Например:

```
<?xml version="1.0" encoding="utf-8"?>
    <bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:src="@drawable/icon_ca" />
```

Если сохранить этот файл под именем icon.xml (в каталоге альтернативных ресурсов, например,res/drawable-en-rCA/), он компилируется в ресурс, на который можно ссылаться с помощью R.drawable.icon, но фактически он является псевдонимом для ресурса R.drawable.icon_ca (который сохранен в каталоге res/drawable/).

Макет

Чтобы создать псевдоним для существующего макета, используйте элемент <include> , заключенный в теги <merge>.

Например:

```
<?xml version="1.0" encoding="utf-8"?>
    <merge>
        <include layout="@layout/main_ltr"/>
    </merge>
```

Если сохранить этот файл под именем main.xml, он компилируется в ресурс, на который можно ссылаться с помощью R.layout.main, но фактически он является псевдонимом для ресурса R.layout.main_ltr .

Строки и другие простые значения

Чтобы создать псевдоним для существующей строки используйте идентификатор ресурса нужной строки в качестве значения для новой строки.

Например:

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <string name="hello">Hello</string>
        <string name="hi">@string/hello</string>
    </resources>
```

Ресурс R.string.hi теперь является псевдонимом для R.string.hello.

Другие простые значения работают аналогично.

Например, цвет:

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
        <color name="yellow">#f00</color>
        <color name="highlight">@color/red</color>
    </resources>
```

Обеспечение оптимальной совместимости устройства с ресурсами

Для того чтобы приложение поддерживало несколько конфигураций устройств, очень важно всегда предоставлять ресурсы по умолчанию для каждого типа ресурсов, используемых приложением.

Например, если приложение поддерживает несколько языков, всегда включайте каталог values/ (в котором сохранены строки) *без* квалификатора языка и региона.

Если вместо этого поместить все файлы строк в каталоги с квалификаторами языка и региона, приложение закроется с ошибкой при запуске на устройстве, на котором установлен язык, отсутствующий в ваших строках.

Но как только вы предоставили ресурсы values/ по умолчанию, приложение будет работать правильно (даже если пользователь не понимает этого языка, это лучше, чем завершение с ошибкой).

Таким же образом, если вы предоставляете различные ресурсы макета в зависимости от ориентации экрана, следует указать одну ориентацию в качестве ориентации по умолчанию.

Оптимальная совместимость с устройствами

Для обеспечения оптимальной совместимости с устройствами обязательно предоставляйте ресурсы по умолчанию, которые приложение может правильно выполнять.

Затем создайте альтернативные ресурсы для определенных конфигураций устройств с помощью квалификаторов конфигурации.

Из этого правила есть одно исключение: Если в приложении для параметра minSdkVersion установлено значение 4 или выше, *не требуется* предоставлять графические ресурсы по умолчанию при предоставлении альтернативных графических ресурсов с квалификатором плотность экрана.

Даже без графических ресурсов по умолчанию Android может найти наиболее подходящую альтернативную плотность экрана и масштабировать растровые изображения при необходимости.

Однако для оптимальной работы на устройствах всех типов следует предоставить альтернативные графические ресурсы для всех трех типов плотности.

Как Android находит наиболее подходящий ресурс

Когда вы запрашиваете ресурс, для которого предоставлена альтернатива, Android выбирает альтернативный ресурс для использования в режиме выполнения в зависимости от текущей конфигурации устройства.

Чтобы продемонстрировать, как Android выбирает альтернативный ресурс, допустим, что имеются следующие каталоги графических ресурсов, каждый из которых содержит различные версии одинаковых изображений:

drawable/

drawable-en/

drawable-fr-rCA/

drawable-en-port/

drawable-en-notouch-12key/

drawable-port-ldpi/

drawable-port-notouch-12key/

И допустим, что устройство имеет следующую конфигурацию:

Язык = en-GB

Ориентация экрана = port

Плотность пикселов на экране = hdpi

Тип сенсорного экрана = notouch

Основной способ ввода текста = 12key

Алгоритм выбора ресурсов

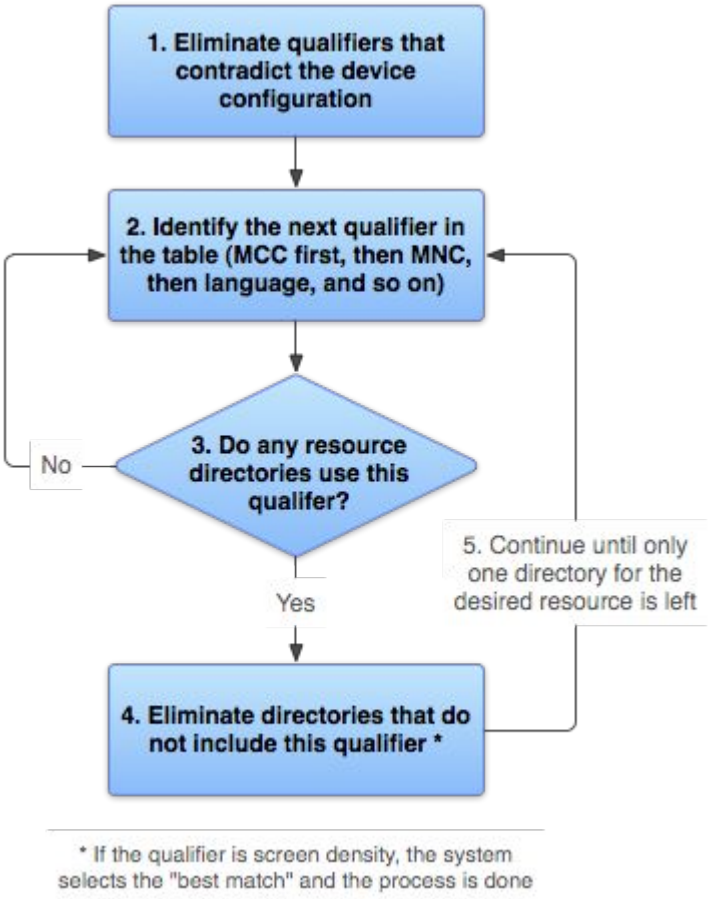
Сравнивая конфигурацию устройства с доступными альтернативными ресурсами, Android выбирает графику из каталога drawable-en-port.

Система приходит к решению об используемых ресурсах на основе следующей логики:

Исключение файлов ресурсов, которые противоречат конфигурации устройства.

Каталог drawable-fr-rCA/ исключается, так как он противоречит языку en-GB.

drawable/
drawable-en/
~~drawable-fr-rCA/~~
drawable-en-port/
drawable-en-notouch-12key/
drawable-port-ldpi/
drawable-port-notouch-12key/



Алгоритм выбора ресурсов

Указание (следующего) квалификатора с высшим приоритетом в списке (таблицы). (Начать с MCC, затем двигаться вниз.)

Содержат ли какие-либо каталоги ресурсов этот квалификатор?

Если Нет, вернуться к шагу 2 и найти следующий квалификатор. (В нашем примере получается ответ «нет», пока не достигнут квалификатор языка.)

Если Да, перейти к шагу 4.

Исключить каталоги ресурсов, которые не содержат этого квалификатора.

В данном примере система исключает все каталоги, которые не содержат квалификатора языка:

~~drawable/~~

drawable-en/

drawable-en-port/

drawable-en-notouch-12key/

~~drawable-port-ldpi/~~

~~drawable-port-notouch-12key~~/

Алгоритм выбора ресурсов

Вернуться и повторять шаги 2, 3 и 4, пока не останется только один каталог. В нашем примере следующим квалификатором, для которого есть совпадения, является ориентация экрана.

Поэтому исключаются ресурсы, не указывающие ориентацию экрана:

~~drawable-en/~~

drawable-en-port/

~~drawable-en-notouch-12key/~~

Остается каталог drawable-en-port.

Хотя эта процедура выполняется для каждого запрошенного ресурса, система дополнительно оптимизирует некоторые вопросы.

Одна из таких оптимизаций состоит в том, что поскольку конфигурация устройства известна, можно исключить альтернативные ресурсы, которые не могут подойти.

Например, если используется конфигурация с английским языком ("en"), все каталоги ресурсов, для которых установлен другой квалификатор языка, никогда не включаются в пул проверяемых ресурсов (хотя каталоги ресурсов *без* квалификатора языка включаются).

Примечание

Приоритет квалификатора (в таблице) более важен, чем число квалификаторов, которые точно соответствуют устройству.

Например, на шаге 4 выше, последний вариант в списке содержит три квалификатора, которые точно соответствуют устройству (ориентация, тип сенсорного экрана и способ ввода), в то время как drawable-en содержит только один подходящий параметр (язык).

Однако язык имеет более высокий приоритет, чем эти остальные квалификаторы, поэтому drawable-port-notouch-12key вычеркивается.

Доступ к ресурсам

После того, как вы предоставили ресурс в вашем приложении этот ресурс можно применить.

Для этого необходимо создать ссылку на идентификатор ресурса.

Для задания всех таких идентификаторов в вашем проекте используется класс R, который автоматически создается инструментом aapt (Android Asset Packaging Tool).

Во время компиляции приложения инструмент aapt создает класс R, в котором находятся идентификаторы для всех ресурсов в каталоге res/.

Для каждого типа ресурсов предусмотрен подкласс R (например, класс R.drawable для элементов дизайна), а для каждого ресурса указанного типа существует статическая целочисленная переменная (например, R.drawable.icon).

Эта переменная как раз и служит идентификатором ресурса, которую можно использовать для его получения.

Идентификатор ресурса

Несмотря на то, что в классе R находятся идентификаторы ресурсов, никогда не обращайтесь к нему для поиска идентификатора ресурса.

Последний состоит из следующих компонентов:

Тип ресурса: ресурсы объединены по типам, таким как string, drawable и layout. Дополнительные сведения о различных типах представлены в разделе Типы ресурсов.

Имя ресурса, в качестве которого выступает либо имя файла (без расширения), либо значение атрибута XML android:name (если ресурс представляет собой простое значение, например строку).

Существует два способа доступа к ресурсу

Из кода: с помощью статической целочисленной переменной из подкласса вашего класса R , например:

R.string.hello

string — это тип ресурса, а hello — это его имя. Существует множество API-интерфейсов Android, которые могут получать доступ к ресурсам, идентификаторы которых указаны в этом формате. См. раздел Доступ к ресурсам из кода.

Из XML: с помощью особого синтаксиса XML, который также соответствует идентификатору ресурса, заданному в классе R, например:

@string/hello

string — это тип ресурса, а hello — это его имя. Этот синтаксис можно использовать в любом фрагменте ресурса XML, где ожидается значение, указанное вами в ресурсе. См. раздел Доступ к ресурсам из XML

Доступ к ресурсам из кода

Чтобы использовать ресурс в коде, можно передать идентификатор ресурса в виде параметра метода. Например, с помощью метода setImageResource() можно указать использование виджетом ImageView ресурса res/drawable/myimage.png:

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

Отдельные ресурсы также можно получать с помощью методов в классе Resources, экземпляр которого можно получить с помощью метода getResources().

Доступ к исходным файлам

В редких случаях может потребоваться получить доступ к исходным файлам и каталогам. В этом случае просто сохранить файлы в каталоге res/ будет недостаточно, поскольку обратиться к ресурсу из папки res/ можно только по его идентификатору. Вместо этого ресурсы можно сохранить в каталоге assets/.

Файлам, которые сохранены в каталоге assets/, *не* присваиваются идентификаторы ресурсов, поэтому вам не удастся сослаться на них с помощью класса R или из ресурсов XML. Вместо этого можно запросить файлы из каталога assets/, как в обычной файловой системе, и считать необработанные данные с помощью AssetManager.

Однако, если вам требуется всего лишь возможность считать необработанные данные (например, видео- или аудиофайл), сохраните требуемый файл в каталоге res/raw/ и считайте поток байтов с помощью метода openRawResource().

Синтаксис

Ниже представлен синтаксис ссылки на ресурс из кода.

[*<package_name>*.]R.*<resource_type>*.*<resource_name>*

<package_name> — это имя пакета, в котором находится ресурс (не требуется при создании ссылок на ресурсы из вашего собственного пакета).

<resource_type> — это подкласс R для типа ресурса.

<resource_name> — это либо имя файла ресурса (без расширения), либо значение атрибута android:name в элементе XML (для простых значений).

Примеры использования

Существует множество методов, которые могут принимать идентификатор ресурса в виде параметра. Для получения ресурсов можно использовать методы, представленные в классе Resources. Можно получить экземпляр Resources с помощью Context.getResources().

Примеры доступа к ресурсам из кода:

```
// Load a background for the current screen from a drawable resource
    getWindow().setBackgroundDrawableResource(R.drawable.my_background_image) ;

    // Set the Activity title by getting a string from the Resources object, because
    //  this method requires a CharSequence rather than a resource ID
    getWindow().setTitle(getResources().getText(R.string.main_title));

    // Load a custom layout for the current screen
    setContentView(R.layout.main_screen);

    // Set a slide in animation by getting an Animation from the Resources object
    mFlipper.setInAnimation(AnimationUtils.loadAnimation(this,
            R.anim.hyperspace_in));

    // Set the text on a TextView object using a resource ID
    TextView msgTextView = (TextView) findViewById(R.id.msg);
    msgTextView.setText(R.string.hello_message);
```

Доступ к ресурсам из XML

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

Например, при добавлении в макет элемента Button необходимо использовать строковый ресурс для надписи на кнопке:

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

Синтаксис

Синтаксис ссылки на ресурс из ресурса XML.

@[*<package_name>*:]*<resource_type>*/*<resource_name>*

<package_name > — это имя пакета, в котором находится ресурс (не требуется при создании ссылок на ресурсы из одного и того же пакета).

<resource_type > — это подкласс R для типа ресурса.

<resource_name > — это либо имя файла ресурса (без расширения), либо значение атрибута android:name в элементе XML (для простых значений).

Дополнительные сведения о каждом типе ресурсов и порядке создания ссылок на них см. в разделе Типы ресурсов.

Примеры использования

В некоторых случаях ресурс необходимо использовать в качестве значения в элементе XML (например, чтобы применить графическое изображение к виджету)

Например, если имеется следующий файл ресурса, включающий цветовой ресурс и строковый ресурс:

```
<?xml version="1.0" encoding="utf-8"?>
    <resources>
      <color name="opaque_red">#f00</color>
      <string name="hello">Hello!</string>
    </resources>
```

Эти ресурсы можно использовать в следующем файле макета для задания цвета текста и надписи:

```
<?xml version="1.0" encoding="utf-8"?>
    <EditText xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:textColor="@color/opaque_red"
       android:text="@string/hello" />
```

В этом случае в ссылке на ресурс не нужно указывать имя пакета, поскольку ресурсы находятся в вашем собственном пакете.

Ссылки на системный ресурс

Однако для создания ссылки на системный ресурс вам потребуется указать имя пакета. Например:

```
<?xml version="1.0" encoding="utf-8"?>
    <EditText
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:textColor="@android:color/secondary_text_dark"
        android:text="@string/hello" />
```

Примечание. Всегда используйте строковые ресурсы, поскольку ваше приложение может потребоваться перевести на другие языки.

Ресурсы в XML для создания псевдонимов.

Вы даже можете использовать ресурсы в XML для создания псевдонимов.

Например, можно создать элемент дизайна, который будет служить псевдонимом для другого элемента дизайна:

```
<?xml version="1.0" encoding="utf-8"?>
    <bitmap
    xmlns:android="http://schemas.android.com/apk/res/android"
        android:src="@drawable/other_drawable" />
```

Это может показаться излишним, однако такой подход очень полезен при использовании альтернативных ресурсов.

Ссылка на атрибуты стиля

Синтаксис для создания ссылки на атрибут стиля практически идентичен обычному формату ресурса, только в этом случае вместо символа@ необходимо указать вопросительный знак (?), а тип ресурса вообще необязательно указывать. Например:

?[*<package_name>*:][*<resource_type>*/]*<resource_name>*

Ниже представлен пример создания ссылки на атрибут для задания цвета текста в соответствии с «основным» цветом текста системной темы оформления:

```
<EditText id="text"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textColor="?android:textColorSecondary"
        android:text="@string/hello_world" />
```

Здесь атрибут android:textColor служит для задания имени атрибута стиля в текущей теме.

Теперь в Android используется значение, примененное к атрибуту стиля android:textColorSecondary в качестве значения для android:textColor в этом виджете.

Поскольку инструменту для работы с системными ресурсами известно, что в этом контексте ожидается ресурс атрибута, вам не нужно явно указывать его тип (который должен быть ?android:attr/textColorSecondary) — тип attr можно исключить.

Доступ к ресурсам платформы

В Android предусмотрен ряд стандартных ресурсов, например, стилей, тем и макетов.

Для доступа к этим ресурсам укажите в ссылке на ресурс имя пакета android.

Например, в Android имеется ресурс макета, который можно использовать для элементов списка в виджете ListAdapter:

setListAdapter(new **ArrayAdapter**<String>(this, **android.R.layout.simple_list_item_1**, myarray));

В этом примере simple_list_item_1 представляет собой ресурс макета, определенный платформой для элементов в виджете ListView.

Вы можете использовать его вместо создания собственных макетов для элементов списка.

Содержание

- http://developer.alexanderklimov.ru/android/locale.php

- https://developer.android.com/guide/topics/resources/providing-resources.html