**Ресурсы, активы и локализация приложений**

**Ресурсы и активы** — неотъемлемая часть ***Android-приложения***. Это внешние элементы, которые включаются в приложение: изображения, аудио, видео, строки, разметки, темы и т. д. Каждое приложение содержит каталог для ресурсов **res/** и каталог для активов **assets/**.

**Ресурсы используются чаще, чем активы. 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, помещенной в каталог активов assets/ (необработанный формат файла), необходимо использовать AssetManager для чтения файла как потока байтов.

В этом разделе вы получите информацию о стандартных ресурсах, которые обычно используются в***Аndrоid-приложении***, и о том, как обращаться к ним в коде программы.

* [Активы](http://androidfanclub.ru/programming/%D0%B0%D0%BA%D1%82%D0%B8%D0%B2%D1%8B)
* [Доступные типы ресурсов](http://androidfanclub.ru/programming/%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%BD%D1%8B%D0%B5-%D1%82%D0%B8%D0%BF%D1%8B-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2)
* [Использование ресурсов в коде программы](http://androidfanclub.ru/programming/%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2-%D0%B2-%D0%BA%D0%BE%D0%B4%D0%B5-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D1%8B)
* [Локализация приложений](http://androidfanclub.ru/programming/%D0%BB%D0%BE%D0%BA%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9)
* [Создание ресурсов](http://androidfanclub.ru/programming/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2)
* [Ссылки на ресурсы](http://androidfanclub.ru/programming/%D1%81%D1%81%D1%8B%D0%BB%D0%BA%D0%B8-%D0%BD%D0%B0-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D1%8B)
* [Стили и темы](http://androidfanclub.ru/programming/%D1%81%D1%82%D0%B8%D0%BB%D0%B8-%D0%B8-%D1%82%D0%B5%D0%BC%D1%8B)

**Активы**

**Активы** — это файлы произвольного типа и содержания, располагаемые в каталоге ***assets/***. Этот каталог находится на одном уровне с каталогом *res/* в дереве проекта. Особенностью этого каталога является то, что файлы в***assets/*** не генерируют идентификаторы ресурса в **классе R**. В программном коде необходимо явно определять путь к файлу для доступа к нему. Путь к файлу — это относительный путь, начинающийся с ***assets/***. Для обращения к этим файлам используется **класс AssetManager**. Этот каталог, в отличие от подкаталога res/, позволяет произвольную глубину подкаталогов и произвольные имена файлов и подкаталогов.

Приведем практический пример: необходимо разработать приложение, в котором будут использоваться шрифты стороннего производителя, не входящие в стандартную **библиотеку шрифтов Android**. Самый простой способ сделать это состоит в том, чтобы упаковать нужные шрифты вместе с вашим приложением. Чтобы сделать это, необходимо создать каталог **assets/** в корне каталога проекта и поместить в этот каталог свои файлы шрифтов.

В качестве примера приложения для работы со шрифтами, загружаемыми в качестве активов в приложение, создайте новый проект и в окне *Create New Project*введите следующие значения:

* Project name — ResFontsApp;
* Application name — Load fonts from assets sample;
* Package name — сот. samples. fonts;
* Create Activity — FontsActivity.

Код в файле разметки **main.xml** состоит из ***контейнера LinearLayout*** и четырех дочерних ***виджетов TextView***, которые будут отображать текст с использованием загружаемых шрифтов.

Код XML-файла разметки деятельности приложения**main.xml** представлен в листинге.

**Файл разметки main.xml**

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

<LinearLayout

xmlns:android="<http://schemas.android.com/apk/res/android>"

android:id="@+id/LinearLayout01"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical"

android:gravity="center">

<TextView

android:text="Libertine Italic"

android:id="@+id/text1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="36sp"

android:padding="10sp"/>

<TextView

android:text="Abaddon font"

android:id="@+id/text2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="36sp"

android:padding="10sp"/>

<TextView

android:text="a Papa font"

android:id="@+id/text3"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="36sp"

android:padding="10sp"/>

<TextView

android:text="A Yummy Apology"

android:id="@+id/text4"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textSize="36sp"

android:padding="10sp"/>

</LinearLayout>

Для примеров были использованы свободно распространяемые шрифты: **Linux-шрифт Libertine** и несколько свободных шрифтов, доступных на сайте [**http://www.urbanfonts.com/freefonts.htm**](http://www.urbanfonts.com/freefonts.htm)*.*

В классе деятельности мы загружаем из ресурсов ***объект Textview***, а затем создаем ***объект Typeface***, используя вызов статического***метода Typeface.createFromAsset()***. ***Mетод createFromAsset()*** принимает два параметра:

1. объект AssetManager, который можно получить вызовом метода getAssets();
2. путь к файлу актива.

Например, загрузить шрифт дотекстового поля **Textview** можно следующим способом:

TextView text (TextView)findViewByld(R.id.text\_i);

Typeface face Typeface.createFromAsset(

getAssets(),"fonts/libertine\_it.ttf");

text.setTypeface(face);

Полный код класса деятельности **FontsActivity** приводится в листинге.

**Класс деятельности FontsActivity**

package com.samples.fonts;

import android.app.Activity;

import android.graphics.Typeface;

import android.os.Bundle;

import android.widget.TextView;

public class FontsActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final TextView text1 = (TextView)findViewById(R.id.text1);

text1.setTypeface(Typeface.createFromAsset(

getAssets(), "fonts/libertine\_it.ttf"));

final TextView text2 = (TextView)findViewById(R.id.text2);

text2.setTypeface(Typeface.createFromAsset(

getAssets(), "fonts/abaddon.ttf"));

final TextView text3 = (TextView)findViewById(R.id.text3);

text3.setTypeface(Typeface.createFromAsset(

getAssets(), "fonts/apapa.ttf"));

final TextView text4 = (TextView)findViewById(R.id.text4);

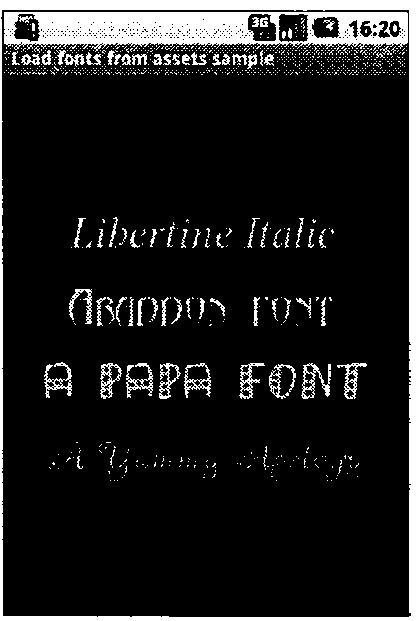
text4.setTypeface(Typeface.createFromAsset(

getAssets(), "fonts/ayummyapology.ttf"));

}

}

Загрузка шрифтов, сохраненных в каталоге ***assets/***



*Внешний вид приложения, запущенного в эмуляторе мобильного устройства*

**Доступные типы ресурсов**

**Android предоставляет разнообразные типы ресурсов, которые вы можете использовать в разрабатываемых приложениях.**

* **Простые значения** — ресурсы могут быть выражены как строки с использованием различных форматирований, чтобы однозначно указать тип создаваемого ресурса.
* **Цвет** — кодировка цвета в шестнадцатеричном выражении определяет значение RGB и alpha-аканал.
* **Строки с дополнительным форматированием**. Можно добавлять дополнительное форматирование для строки, используя три стандартных HTML-тега: ***<b>, <i> и <u>***. Методы, которые будут обрабатывать строковые ресурсы с HTMLформатированием, должны уметь обрабатывать эти теги.
* **Графические ресурсы**. Есть множество видов графических ресурсов, которые можно создавать и использовать в приложении:
* **файлы растровой графики в различных форматах**: \*.png (который наиболее предпочтителен для использования), \*.jpg и \*.gif;
* **PaintDrawable** — объекты, которые являются четырехугольником цвета с произвольно округленными углами. Этот элемент может быть определен в любом из файлов внутри каталога ***res/values/;***
* **графика NinePatch** — изображение PNG, которое можно растягивать. В Android графика **NinePatch** используется для прорисовки виджетов — кнопок, закладок и т. д.
* **Анимация** — Android может выполнить простую анимацию на графике или на серии графических изображений. Анимация включает вращения, постепенное изменение, перемещение и протяжение.
* **Меню**— меню и подменю также могут быть определены как XML ресурсы и загружены в приложение.
* **XML-файлы разметки**. Этот вид ресурса вы уже хорошо изучили ранее.
* **Стили**— используются для элементов. Это один или более атрибутов, относящихся к одному элементу. Стиль применен как атрибут к элементу в файле разметки.
* **Темы** — это один или более атрибутов, относящихся к целому экрану. Например, можно применить готовую тему, определенную в Android, — Theme. dialog к деятельности, разрабатываемой для плавающих диалоговых окон.

**Использование ресурсов в коде программы**

Во время компиляции генерируется **класс R**, который является оболочкой ресурсов и содержит идентификаторы всех ресурсов в программе. Класс r имеет несколько вложенных классов, один для каждого типа ресурса, поддерживаемого системой Android, и для которого в проекте существует файл ресурса. Класс r может содержать следующие вложенные классы:

**R.anim**— идентификаторы для файлов из каталога *res/anim/*;

**R.array**— идентификаторы для файлов из каталога *res/values/*;

**R.bool**— идентификаторы для битовых массивов в файлах **arrays.xml** из каталога *res/values/*;

**R.color**— идентификаторы для файлов **colors.xml** из каталога *res/values/*;

**R.dimen** — идентификаторы для файлов **dimens.xml**из каталога *res/values/*;

**R.drawable** — идентификаторы для файлов из каталога *res/drawable/*;

**R.id**— идентификаторы представлений и групп представлений для файлов XMLразметки из каталога *res/layout/*;

**R.integer**— идентификаторы для целочисленных массивов в файлах **arrays.xml**из каталога *res/values/*;

**R.layout** — идентификаторы для файлов разметки из каталога *res/layout/*;

**R.raw** — идентификаторы для файлов из каталога *res/raw/*;

**R.string** — идентификаторы для файлов **strings.xml** из каталога *res/values/*;

**R. style**— идентификаторы для файлов **styles.xml**из каталога *res/values/*;

**R.xml** — идентификаторы для файлов из каталога *res/xml/*.

Каждый вложенный класс содержит один или несколько идентификаторов для откомпилированных ресурсов, которые используются в коде программы для загрузки ресурса. Далее приведен синтаксис для обращения к ресурсу:

R.resource\_type.resource\_name

В приложениях также можно использовать системные ресурсы. Все системные ресурсы определены под **классом android.R**. Например, стандартный значок приложения на экране можно отобразить так:

android.R.drawable.sym\_def\_app\_icon

Или загрузить стандартный стиль:

android.R.style.Theme\_Black

* [Загрузка XML-документов](http://androidfanclub.ru/programming/%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0-xml-%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2)
* [Загрузка простых типов из ресурсов](http://androidfanclub.ru/programming/%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0-%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%8B%D1%85-%D1%82%D0%B8%D0%BF%D0%BE%D0%B2-%D0%B8%D0%B7-%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81%D0%BE%D0%B2)
* [Загрузка файлов произвольного типа](http://androidfanclub.ru/programming/%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0-%D1%84%D0%B0%D0%B9%D0%BB%D0%BE%D0%B2-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE-%D1%82%D0%B8%D0%BF%D0%B0)
* [Создание меню в XML](http://androidfanclub.ru/programming/%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D0%B5-%D0%BC%D0%B5%D0%BD%D1%8E-%D0%B2-xml)

**Загрузка XML-документов**

Библиотеки Android имеют набор классов для чтения и записи *XML-документов* с произвольной структурой и содержанием. Чтобы упаковать статический *XML-документ* с вашим приложением, поместите его в каталог **res/xml/**, и тогда вы получите возможность обращаться в коде программы к этому документу.

Рассмотрим загрузку *XML-документа* произвольной структуры из ресурсов в код программы. Создадим приложение, способное читать список контактов, определенных в XMLфайле. В окне New Android Project заполните поля:

* Project name — FilesXml;
* Application name — XMLResource Sample;
* Package name — соm.samples.resxmi;
* Create Activity — ResXmlActivity.

В каталоге **res/** создайте подкаталог **xml/**, в котором будет располагаться *XML-файл* контактов. В этом файле мы напишем список контактов, уже неоднократно применяемый в приложениях, и сохраним его под именем **contacts.xml**. Полный код файла **contacts.xml** представлен в листинге.

**XML-документ contacts.xml**

<contacts>

<contact

first\_name="Jacob"

last\_name="Anderson"

phone="412412411"/>

<contact

first\_name="Emily"

last\_name="Duncan"

phone="161863187"/>

<contact

first\_name="Michael"

last\_name="Fuller"

phone="161863187"/>

<contact

first\_name="Emma"

last\_name="Greenman"

phone="161863187"/>

<contact

first\_name="Joshua"

last\_name="Harrison"

phone="161863187"/>

<contact

first\_name="Madison"

last\_name="Johnson"

phone="891863187"/>

<contact

first\_name="Matthew"

last\_name="Cotman"

phone="161863187"/>

<contact

first\_name="Olivia"

last\_name="Lawson"

phone="161863187"/>

<contact

first\_name="Andrew"

last\_name="Chapman"

phone="161863187"/>

<contact

first\_name="Daniel"

last\_name="Honeyman"

phone="161863187"/>

<contact

first\_name="Isabella"

last\_name="Jackson"

phone="161863187"/>

<contact

first\_name="William"

last\_name="Patterson"

phone="687699693"/>

<contact

first\_name="Joseph"

last\_name="Godwin"

phone="965467575"/>

<contact

first\_name="Samantha"

last\_name="Bush"

phone="907865645"/>

<contact

first\_name="Christopher"

last\_name="Gateman"

phone="896874556"/>

</contacts>

**Файл разметки main.xml**

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

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent" >

<ListView

android:id="@android:id/list"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:drawSelectorOnTop="false"/>

</LinearLayout>

**Загрузить файл contacts.xml, созданный ранее, можно следующим образом:**

XmlPullParser parser getResources().getXml(R.xml.contacts);

Метод **getxml()** возвращает **XmlPullParser**, используя который можно прочитать загруженный *XML-документ* в цикле while:

while (parser.getEventType()!= XmlPullParser.END\_DOCUMENT)

if (parser.getEventType() = XmlPullParser.START\_TAG

&& parser.getName().equals("contact")){

list.add(parser.getAttributeValue(O)

+ parser.getAttributeValue(l)

+ "\n" + parser.getAttributeValue(2));

parser.next();

**Полный код класса главной деятельности приложения ResXmlActivity (файл класса деятельности ResXmlActivity .Java)**

package com.samples.resxml;

import java.util.ArrayList;

import org.xmlpull.v1.XmlPullParser;

import android.app.ListActivity;

import android.os.Bundle;

import android.widget.ArrayAdapter;

import android.widget.Toast;

public class ResXmlActivity extends ListActivity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

ArrayList<String> list = new ArrayList<String>();

try {

XmlPullParser parser = getResources().getXml(R.xml.contacts);

while (parser.getEventType()!= XmlPullParser.END\_DOCUMENT) {

if (parser.getEventType() == XmlPullParser.START\_TAG

&& parser.getName().equals("contact")) {

list.add(parser.getAttributeValue(0) + " "

+ parser.getAttributeValue(1) + "\n"

+ parser.getAttributeValue(2));

}

parser.next();

}

}

catch (Throwable t) {

Toast.makeText(this,

"Error loading XML document: " + t.toString(), 4000)

.show();

}

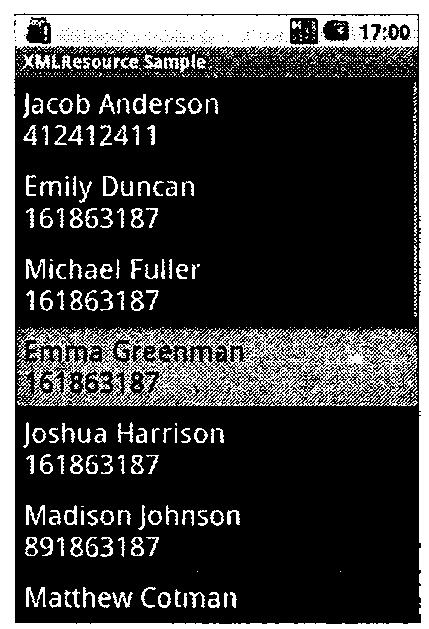
setListAdapter(new ArrayAdapter<String>(

this, android.R.layout.simple\_list\_item\_1, list));

}

}

Внешний вид приложения, запущенного в эмуляторе мобильного устройства, показан на рисунке:



*Чтение XMLдокумента из ресурсов*

**Загрузка простых типов из ресурсов**

Чтобы изучить на практике загрузку разнообразных простых типов ресурсов и их вызов из программного кода, создадим в*Eclipse* новый проект:

* Project name — SimpleValues;
* Application name — Resource Simple;
* Package name — com. samples. simplevalues;
* Create Activity — SimpleValuesActivity.

При создании проекта в ***каталоге res/values/*** мастер генерирует единственный файл **strings.xml**. Добавьте в этот каталог следующие XMLфайлы:

* **arrays.xml** — для массивов строк и целочисленных значений;
* **colors.xml** — для значений цвета;
* **dimen.xml** — для значений размеров текста;
* **drawables.xml** — для графических примитивов.

Заполните эти файлы разнообразными данными.

**Файл ресурсов res/values/arrays.xml**

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

<resources>

<string-array name="names">

<item>Andrew</item>

<item>Helen</item>

<item>Jack</item>

</string-array>

<integer-array name="digits">

<item>1999</item>

<item>2002</item>

<item>2010</item>

</integer-array>

</resources>

**Файл ресурсов res/values/colors.xml**

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

<resources>

<color name="textColor">#0000FF</color>

<color name="backgroundColor">#FF0000</color>

</resources>

**Файл ресурсов res/values/dimen.xml**

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

<resources>

<dimen name="textPointSize">18pt</dimen>

</resources>

**Файл ресурсов res/values/drawables.xml**

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
<drawable name="grayDrawable">#DDD</drawable>  
</resources>

**Файл ресурсов res/values/strings.xml**

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

<resources>

<string name="app\_name">Resource Sample</string>

<string name="some\_text">Some Text</string>

</resources>

В файле разметки создадим структуру с текстовыми полями для отображения на экране загружаемых ресурсов. Файл разметки деятельности приложения **main.xml** приведен в листинге.

**Файл разметки main.xml**

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

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent">

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<TextView

android:layout\_height="wrap\_content"

android:text="String array:"

android:layout\_width="wrap\_content"

android:paddingRight="5px"

android:textColor="@color/textColor"/>

<TextView

android:id="@+id/text\_strings"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:padding="5px"

android:textColor="@color/textColor"/>

</LinearLayout>

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<TextView

android:layout\_height="wrap\_content"

android:text="Int array:"

android:layout\_width="wrap\_content"

android:paddingRight="20px"

android:padding="5px"

android:textColor="@color/textColor"/>

<TextView

android:id="@+id/text\_digits"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textColor="@color/textColor"/>

</LinearLayout>

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content">

<TextView

android:id="@+id/text\_style"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"/>

</LinearLayout>

</LinearLayout>

В классе деятельности ***simpleValuesActivity*** показаны варианты загрузки ресурсов в зависимости от их типа с помощью сгенерированного плагином **класса R**.

**Файл класса деятельности SiinpleValuesAetivity**

package com.samples.simplevalues;

import android.app.Activity;

import android.graphics.drawable.ColorDrawable;

import android.os.Bundle;

import android.view.Window;

import android.widget.TextView;

public class SimpleValuesActivity extends Activity {

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final TextView textStrings = (TextView)findViewById(R.id.text\_strings);

String[] names = getResources().getStringArray(R.array.names);

for(int i = 0; i < names.length; i++) {

textStrings.append("Name[" + i + "]: "+ names[i] + "\n");

}

final TextView textDigits = (TextView)findViewById(R.id.text\_digits);

int[] digits = getResources().getIntArray(R.array.digits);

for(int i = 0; i < digits.length; i++) {

textDigits.append("Digit[" + i + "]: "+ digits[i] + "\n");

}

final TextView textStyle = (TextView)findViewById(R.id.text\_style);

textStyle.setText(

getResources().getText(R.string.some\_text));

textStyle.setTextColor(

getResources().getColor(R.color.textColor));

textStyle.setTextSize(

getResources().getDimension(R.dimen.textPointSize));

textStyle.setBackgroundColor(

getResources().getColor(R.color.backgroundColor));

Window w = this.getWindow();

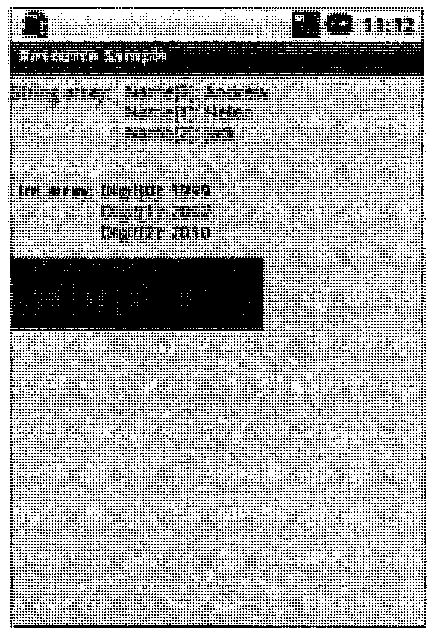
w.setBackgroundDrawable((ColorDrawable)getResources().getDrawable(

R.drawable.grayDrawable));

}

}

После компиляции и запуска проекта внешний вид приложения должен быть таким:



**Загрузка простых типов ресурсов**

**Загрузка файлов произвольного типа**

**Каталог res/raw/** предназначен для хранения файлов произвольного типа, которые сжимаются при компиляции проекта и копируются на устройство в "нормальном" виде. Например, в этот каталог можно поместить какой либо текстовый файл, который будет загружаться в приложение.

При загрузке файла с произвольным типом данных используются методы работы с потоками ввода-вывода. Однако, в отличие от примера (текстового редактора), файлы, размещаемые в **каталоге res/raw/**, генерируют идентификатор ресурса, и в коде программы к ним можно обращаться не по имени файла, а по идентификатору ресурса.

Чтобы показать на практике вариант использования загрузки файлов из каталога res/raw/, создадим новый проект:

* Project name — ContactLauncher;
* Application name — Contact Launcher Sample;
* Package name — соm.samples.contactlauncher;
* Create Activity — ContactLauncherActivity.

В файле разметки **main.xml** создайте единственный ***элемент TextView***, как показано в листинге.

**Файл разметки main.xml**

<?xml version="1.0" encoding="utf8"?>

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:gravity="center">

<TextView

android:id="@+id/text"

android: layout\_height="wrap\_content"

android:text="@string/hello"

android:layout\_width="wrap\_content"

android:gravity="leftlcenter"/>

</LinearLayout>

Загрузка файла из ресурсов аналогична загрузке, приведенной в приложении для работы с файлами. Код класса деятельности, в котором производится загрузка ресурса, приведен в листинге.

**Файл класса деятельности RawActivity.java**

package com.samples.raw;

import java.io.DataInputStream;

import java.io.InputStream;

import android.app.Activity;

import android.os.Bundle;

import android.widget.TextView;

public class RawActivity extends Activity

@Override

public void onCreate(Bundle savedInstanceState)

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

final TextView text (TextView)findViewById(R.id.text);

InputStream iFile getResources().openRawResource(R.raw.file);

try {StringBuffer sBuffer new StringBuffer();

DataInputStream dataIO = new DataInputStream(iFile);

String strLine null;

while((strLine=dataIO.readLine())null){sBuffer.append(strLine+"\n");

dataIO.close();

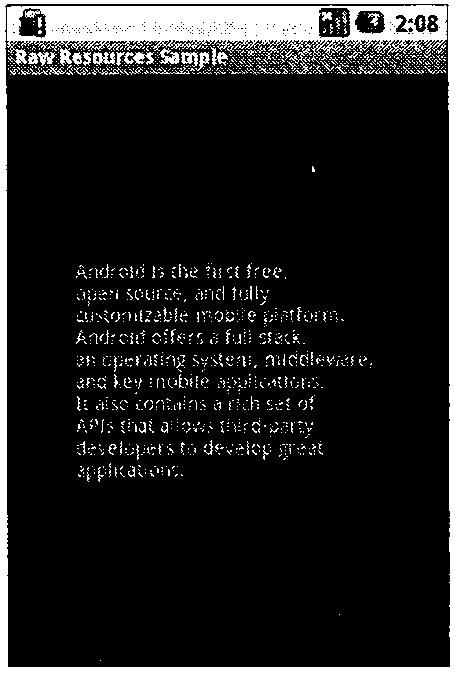
iFile.close();

text.setText(sBuffer.toString());

catch (Exceptionе)

text.setText("Error loading file:+e.getMessage());

Сделаем компиляцию и запустим проект на выполнение. Приложение должно загрузить текстовый файл из каталога ресурсов и вывести на экран его содержимое:



*Загрузка ресурсов из файла в каталоге res/raw/*

**Создание меню в XML**

В предыдущих приложениях мы создавали для окон деятельности схемы разметки в виде *XML-документов*, сохраняемых в каталоге res/layouts/. Аналогично можно определять меню в *XML-документах* и загружать их в приложение. Файлы меню сохраняются в отдельном каталоге **res/menu/.**

**В XML-файле меню есть три элемента:**

* **<menu>** — корневой элемент файла меню;
* **<group>** — контейнерный элемент, определяющий группу меню;
* **<item>** — элемент, определяющий пункт меню.

Элементы **<item>**и **<group>** могут быть дочерними элементами **<group>**. Конечно, корневой узел любого файла должен быть элементом меню.

Как пример приложения с загрузкой меню из XML мы определим то же самое меню со значками, созданное нами ранее. Создадим в Eclipse новый проект и в диалоге Create New Project заполним поля:

* Project пате — IconMenuApp;
* Application пате — Load Menu from resource;
* Package пате — com. samples. resmenuxmi;
* Create Activity — ResMenuXmlActivity.

В ***XML-файле***, который будет определять меню, создадим пять пунктов меню — **Open, Save, Edit, Help**и**Exit** и сохраним этот файл под именем **options.xml** в каталоге ***res/menu/***проекта. Полный код файла **options.xml** представлен в листинге.

**Файл меню options.xml**

<menu xmlns:android="<http://schernas.android.com/apk/res/android>">

<item

android:id="@+id/open"

android:title="Open"

android:icon="@drawable/ic\_menu\_open"

android: orderInCategory="l"/>

<item

android:id="@+id/save"

android:title="Save"

android:icon="@drawable/ic\_menu\_save"

android:orderInCategory="2"/>

<item

android:id="@+id/edit"

android:title="Edit"

android:icon="@drawable/ic\_menu\_edit"

android:orderInCategory="3"/>

<item

android:id="@+id/help"

android:title="Help"

android:icon="@drawable/ic\_menu\_help"

android:orderInCategory="4"/>

<item

android:id="@+id/exit"

android:title="Exit"

android:icon="@drawable/ic\_menu\_exit"

android:orderInCategory="5"/>

</menu>

Файл разметки для деятельности приложения с единственным виджетом TextView приведен в листинге.

**Файл разметки main.xml**

<?xml version="1.0" encoding="utf8"?>

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fi1l\_parent">

<TextView

android:layout\_width="fillparent"

android:layout\_he ight="wrap\_content"

android:text="Press MENU button..

android: gravity="center"

android:textStyle="bold"/>

</LinearLayout>

Загрузка меню из ресурса несколько отличается от создания меню в коде приложения. В коде метода обратного вызова **onCreateOptionsMenu()** достаточно получить ссылку на ресурс *XML файла*, определяющего меню:

@Override

public boolean onCreateOptionsMenu (Menu menu)

getMenuInflater().inflate(R.menu.options, menu);

return true;

При загрузке меню из *XML-файла* идентификаторы меню определяют в файле, и, следовательно, нет необходимости объявлять их в коде программы. В **классе R** будут сгенерированы константы, к которым можно обращаться в коде, как и к идентификаторам остальных ресурсов. Эти идентификаторы можно использовать в методе обратного вызова **onOptionsitemSelected()**, например, таким образом:

@Override

public boolean onOptionsItemSelected (MenuItem item)

switch(item.getItemId())

case R.id.open:

break;

case R.id.save:

break;

default:

return false;

return true;

Полный код класса главной деятельности ResMenuXmlActivity показан в листинге.

**Файл класса деятельности ResMenuXml Activity Java**

package com.samples.resmenuxml;

import android.app.Activity;

import android.os.Bundle;

import android.view.Gravity;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.Toast;

public class ResMenuXmlActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.main);

}

// загрузка меню из XML-файла

@Override

public boolean onCreateOptionsMenu(Menu menu) {

getMenuInflater().inflate(R.menu.options, menu);

return true;

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

CharSequence message;

switch (item.getItemId()) {

case R.id.open:

message = "Open item selected";

break;

case R.id.save:

message = "Save item selected";

break;

case R.id.help:

message = "Help item selected";

break;

case R.id.edit:

message = "Edit item selected";

break;

case R.id.exit:

message = "Exit item selected";

break;

default:

return false;

}

Toast toast = Toast.makeText(this, message, Toast.LENGTH\_SHORT);

toast.setGravity(Gravity.CENTER, 0, 0);

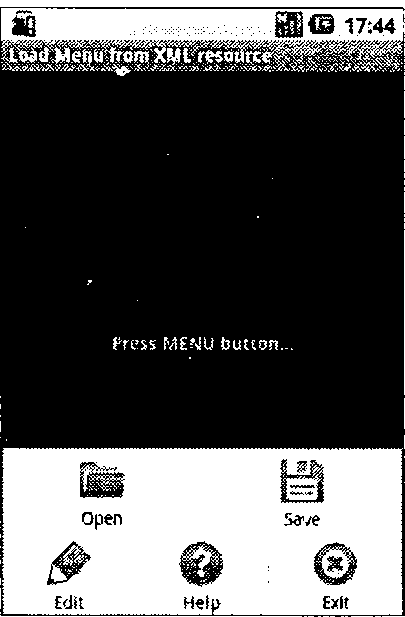
toast.show();

return true;

}

}

Скомпилируйте и запустите проект на выполнение. Внешний вид приложения с загрузкой меню из *XML-файла* не отличается от меню, созданного в коде программы.



*Меню, загруженное из XML-файла*

# Локализация приложений

**Приложения для платформы Android** могут работать на устройствах во многих регионах. Чтобы привлечь больше пользователей, ваше приложение должно . обрабатывать текстовые, аудиофайлы, числа, валюту и графику способами, соответствующими настройкам или языкам тех регионов, где ваше приложение будет использоваться.

Xорошая практика программирования под Android заключается в использовании файлов ресурсов, чтобы отделить содержание приложения от логики работы приложения. Обычно локализуют строковые свойства элементов пользовательского интерфейса, но в принципе можно локализовать и другие компоненты приложения, например разметку, создав отдельную разметку для каждого региона или языка.

# Ресурсы, заданные по умолчанию

  Всякий раз, когда приложение запускается с настройками региона (или языка), для которого вы не создали локализованные строковые ресурсы, **система Android** загружает заданные по умолчанию строки из **файла res/values/strings.xml**. Если заданный по умолчанию ресурс отсутствует или в нем отсутствует требуемая строка, то приложение не запустится и сгенерирует исключение.

Чтобы предотвратить запуск приложения с ошибкой, удостоверьтесь, что ***файл res/values/strings.xml*** существует и определяет каждую необходимую строку. Данное требование применимо ко всем типам ресурсов, а не только к строкам: вы должны создать набор заданных по умолчанию файлов ресурса, содержащих все ресурсы, которые ваше приложение загружает, — разметки, изображения, анимацию и т. д.

При создании локализованного приложения исследуйте в программном коде каждую ссылку на ресурсы. Удостоверьтесь, что заданный по умолчанию ресурс определен для каждого. Также удостоверьтесь, что заданный по умолчанию строковый файл содержит все необходимые строки. Локализованный строковый файл может содержать подмножество строк, но заданный по умолчанию строковый файл должен содержать их все — если какаялибо строка отсутствует в локализованном файле, загрузится версия строки, определенная в ресурсах по умолчанию.

# Создание локализованных ресурсов

  Чтобы создать дополнительный ресурс для конкретного региона или языка, необходимо использовать спецификатор, который определяет комбинацию региона и языка. Пример спецификаторов приведен в табл.

#### Спецификаторы локализованных ресурсов

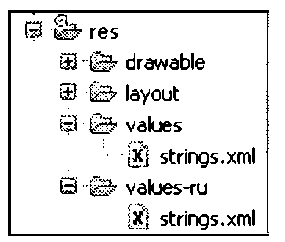
|  |  |  |
| --- | --- | --- |
| Л**окальный код** | **Язык / Страна** | **Размещение файла strings.xml** |
| Default | English / United Kingdom | res/values/ |
| rurRU | Russian / Russia | res/valuesru/ |
| derDE | German / Germany | res/valuesde/ |
| jarJP | Japanese/Japan | res/valuesj a/ |
| frrFR | French / France | res/valuesfr/ |
| frrCA | French / Canada | res/valuesfr/ |
| enrCA | English / Canada | res/values/ |
| enrUS | English / United States | res/values/ |

Как видно из таблицы, английский язык является заданным по умолчанию и для него нет смысла определять отдельный ресурс, создавая каталог **res/valuesen/**.

Давайте создадим локализованное приложение, переведя на русский язык интерфейс приложения для работы с базой данных контактов из предыдущей главы. При создании проекта мастер создает под каталогом **res/** папки по умолчанию.

В каталоге **res/** создадим дополнительную папку **valuesru/**— каталог для русской локализованной версии. В этот каталог мы поместим локализованную версию файла строковых ресурсов, как показано на рис.

На основе английского файла создайте файл для русской версии приложения, переведя все строки, находящиеся в нем, на русский язык, как показано в листинге. Сохраните файл в каталоге **res/valuesru/**.



Каталоги с локализованными строковыми ресурсами

**Файл строковых ресурсов strings.xml для русского языка**

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

<resources>

<string name="app\_name">Загрузка контактов из базы данных</string>

<string name="btn\_ok">Сохранить</string>

<string name="btn\_cancel">Отмена</string>

<string name="field\_name">Имя:</string>

<string name="field\_phone">Тел.:</string>

<string name="title\_add">Добавить новый контакт</string>

<string name="title\_edit">Изменить контакт</string>

<string name="title\_delete">Удалить контакт?</string>

<string name="menu\_add">Добавить</string>

<string name="menu\_edit">Изменить</string>

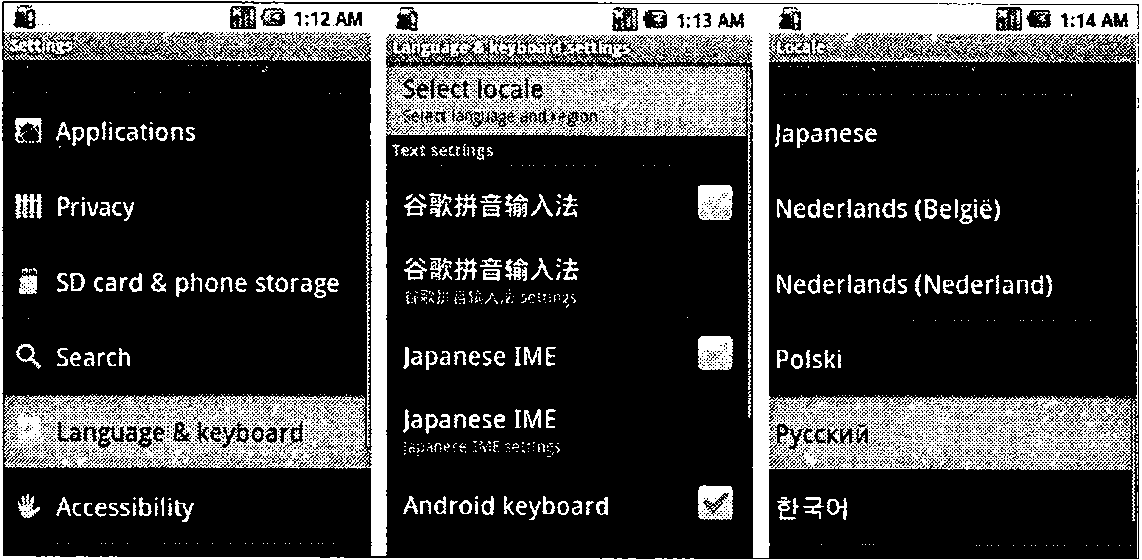
<string name="menu\_delete">Удалить</string>

<string name="toast\_notify">Выберите контакт!</string>

</resources>

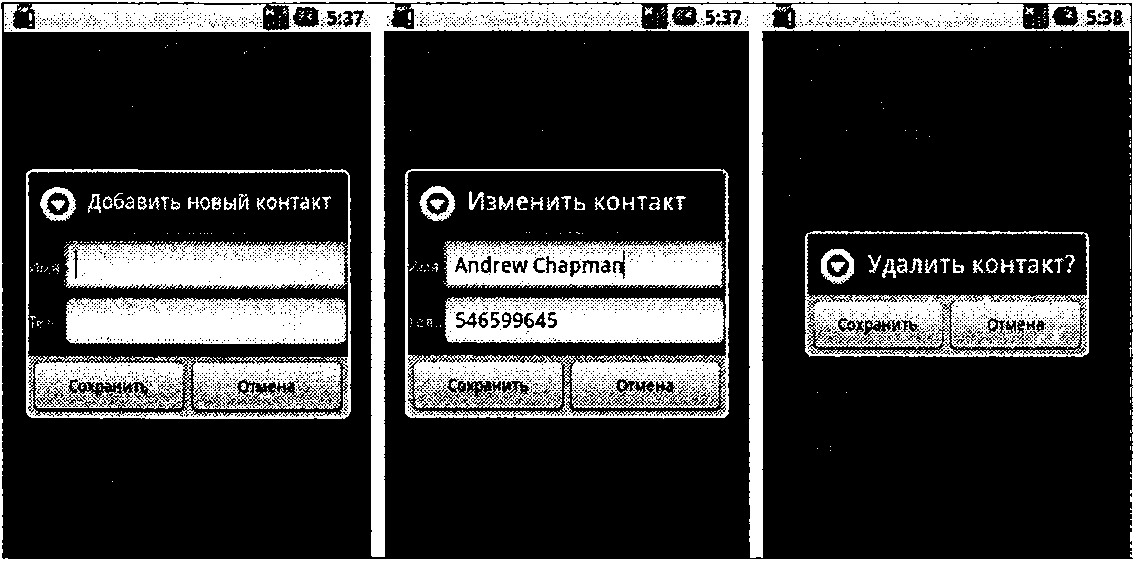
Создав необходимые файлы, выполните компиляцию проекта. Если у вас в эмуляторе мобильного устройства остались настройки по умолчанию, отобразится английская версия приложения.

Теперь поменяйте язык. Для этого откройте панель Application Launcher и выберите последовательно **Settings | Language & keyboard | Select locale | Русский**, как показано на рис.



Установка языковых настроек в эмуляторе

Снова зайдите в **Application Launcher**. Теперь наше **приложение в Application Launcher**называется Загрузка контактов из базы данных. Запустите его. Приложение и все его диалоговые окна будут на русском языке. Внешний вид локализованного приложения с русским языком интерфейса, запущенного в эмуляторе мобильного устройств:



Локализованное приложение для редактирования контактов

**Создание ресурсов**

Как уже было сказано , все ресурсы создаются и сохраняются в предопределенных подкаталогах в каталоге **res/** проекта. **Android SDK** имеет утилиту***aapt***для компиляции ресурсов. Далее приводится список каталогов и вложенных файлов для каждого типа ресурса.

* **res/anim/** — файлы анимации.
* **res/drawable/**—графика.
* **res/layout/** — XMLфайлы для схем разметки.
* **res/values/**— XMLфайлы, которые могут быть откомпилированы во многие виды ресурса. В отличие от других каталогов, res/ может содержать любое число файлов. При создании файлов ресурсов следует соблюдать соглашение об именовании файлов ресурсов по типу элементов, определенных в них:

1. *arrays.xml* — массивы;
2. *colors.xml* — значения цвета для графики и строк текста;
3. *dimens.xml* —размерности;
4. *strings.xml* — строковые значения;
5. *styles.xml* —стили.

* **res/xml/**— произвольные XMLфайлы, которые могут загружаться во время выполнения.
* **res/raw/**— произвольные файлы, предназначенные для копирования непосредственно на устройство.

# Ссылки на ресурсы

Значение атрибута XMLэлемента (или ресурса) может также быть ссылкой на ресурс. Это часто используется в файлах разметки для строк и изображений (которые находятся в другом файле), хотя ссылкой может быть любой тип ресурса, например, цветовая кодировка или целочисленные значения.

Например, если мы имеем цветовые ресурсы, мы можем записать файл разметки, который устанавливает значение цвета для текста, находящегося в одном из этих ресурсов:

android:textColor="@color/opaque\_red"

Обратите внимание здесь на использование префикса @ для того, чтобы ввести ссылку ресурса — текст после этого префикса — имя ресурса. В этом случае мы не должны были указывать пакет, потому что мы ссылаемся на ресурс в нашем собственном пакете.