[Диалоги AlertDialog Кастомизация](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/126-urok-65-dialogi-alertdialog-kastomizatsija.html" \o "Урок 65. Диалоги. AlertDialog. Кастомизация)

Кроме сообщения или списка мы можем помещать в диалог свои View-компоненты. Для этого доступно как основное тело диалога, так и заголовок. Для этого нам необходимо создать View и с помощью методов setCustomTitle или setView вставить этот View соответственно в заголовок или тело диалога. Я буду использовать только setView и работать с телом диалога. Использование метода setCustomTitle и работа с заголовком диалога полностью аналогична.

Нарисуем приложение, которое будет использовать свой View-компонент в теле диалога и рассмотрим, как можно модифицировать содержимое этого компонента. Будем добавлять и удалять TextView в теле диалога.

Создадим проект:

**Project name**: P0651\_AlertDialogCustom  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogCustom  
**Package name**: ru.startandroid.develop.p0651alertdialogcustom  
**Create Activity**: MainActivity

В **strings.xml** пропишем тексты:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="app\_name">AlertDialogCustom</string>      <string name="add">Добавить</string>      <string name="remove">Удалить</string>  </resources> |

**main.xml** – экран с двумя кнопками для удаления и добавления элементов в диалог

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="vertical">      <Button          android:id="@+id/btnAdd"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="@string/add"          android:onClick="onclick">      </Button>      <Button          android:id="@+id/btnRemove"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="@string/remove"          android:onClick="onclick">      </Button>  </LinearLayout> |

**dialog.xml** – наше будущее кастом-тело диалога.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:orientation="vertical">      <TextView          android:id="@+id/tvTime"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="">      </TextView>      <TextView          android:id="@+id/tvCount"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="">      </TextView>  </LinearLayout> |

В **tvTime**- будем отображать текущее время, а в **tvCount** – кол-во добавленных TextView.

**MainActivity.java**:

|  |
| --- |
| package ru.startandroid.develop.p0651alertdialogcustom;    import java.sql.Date;  import java.text.SimpleDateFormat;  import java.util.ArrayList;    import android.app.Activity;  import android.app.AlertDialog;  import android.app.Dialog;  import android.os.Bundle;  import android.view.View;  import android.view.ViewGroup.LayoutParams;  import android.widget.LinearLayout;  import android.widget.TextView;    public class MainActivity extends Activity {      final int DIALOG = 1;      int btn;    LinearLayout view;    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");    TextView tvCount;    ArrayList<TextView> textViews;      /\*\* Called when the activity is first created. \*/    @Override    public void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.main);      textViews = new ArrayList<TextView>(10);    }      public void onclick(View v) {      btn = v.getId();      showDialog(DIALOG);    }      @Override    protected Dialog onCreateDialog(int id) {      AlertDialog.Builder adb = new AlertDialog.Builder(this);      adb.setTitle("Custom dialog");      // создаем view из dialog.xml      view = (LinearLayout) getLayoutInflater()          .inflate(R.layout.dialog, null);      // устанавливаем ее, как содержимое тела диалога      adb.setView(view);      // находим TexView для отображения кол-ва      tvCount = (TextView) view.findViewById(R.id.tvCount);      return adb.create();    }      @Override    protected void onPrepareDialog(int id, Dialog dialog) {      super.onPrepareDialog(id, dialog);      if (id == DIALOG) {        // Находим TextView для отображения времени и показываем текущее        // время        TextView tvTime = (TextView) dialog.getWindow().findViewById(            R.id.tvTime);        tvTime.setText(sdf.format(new Date(System.currentTimeMillis())));        // если была нажата кнопка Добавить        if (btn == R.id.btnAdd) {          // создаем новое TextView, добавляем в диалог, указываем текст          TextView tv = new TextView(this);          view.addView(tv, new LayoutParams(LayoutParams.MATCH\_PARENT,              LayoutParams.WRAP\_CONTENT));          tv.setText("TextView " + (textViews.size() + 1));          // добавляем новое TextView в коллекцию          textViews.add(tv);          // иначе        } else {          // если коллекция созданных TextView непуста          if (textViews.size() > 0) {            // находим в коллекции последний TextView            TextView tv = textViews.get(textViews.size() - 1);            // удаляем из диалога            view.removeView(tv);            // удаляем из коллекции            textViews.remove(tv);          }        }        // обновляем счетчик        tvCount.setText("Кол-во TextView = " + textViews.size());      }    }  } |

Рассмотрим код. В методе **onCreate** выполняем стандартные процедуры и создаем коллекцию textViews для хранения добавляемых TextView.

Метод **onclick** – обработчик нажатий на кнопки, сохраняет в btn идентификатор нажатой кнопки и показывает диалог.

В методе создания диалога **onCreateDialog**мы задаем текст заголовка диалога, создаем **view** из layout-файла dialog.xml и с помощью метода [setView](http://developer.android.com/reference/android/app/AlertDialog.html" \l "setView(android.view.View)" \t "_blank) говорим диалогу, что надо использовать наше **view** в качестве тела диалога. И теперь, работая с **view**, мы будем формировать тело диалога. В только что созданном **view** сразу находим tvCount – для отображения кол-ва добавленных TextView.

Метод **onPrepareDialog** – здесь мы с помощью метода [getWindow](http://developer.android.com/reference/android/app/Dialog.html" \l "getWindow()" \t "_blank) получаем доступ к View-компонентам диалога, находим среди них tvTime и показываем в нем время. Далее определяем, какая кнопка была нажата. Если кнопка добавления, то создаем TextView и помещаем его в **view**(который определили в методе onCreateDialog) и добавляем в коллекцию textViews. Таким образом TextView добавится в тело диалога. Если же хотим удалить TextView, то находим в коллекции последний добавленный и удаляем его из компонента view и из коллекции textViews. В конце обновляем счетчик кол-ва добавленных TextView в диалоге.

Обратите внимание, что я использую два разных способа для получения доступа к tvCount и tvTime. tvCount я нашел сразу после создания **view**в методе onCreateDialog. А в случае с tvTime я показываю, как найти View-компонент в диалоге без использования объекта **view**. Какой вам удобнее по ситуации, тот и используйте.

Все сохраним и запустим приложение. Нажмем кнопку Добавить



Появился диалог. Отображает время, кол-во добавленных TextView и собственно сами добавленные TextView.

Закрываем диалог кнопкой Назад, жмем еще раз добавить



Добавилось еще одно TextView.

Закрываем диалог, жмем кнопку Удалить.



Последнее TextView удалилось.

Мы закрываем диалог нажатием на кнопку, на пункт списка или кнопкой Назад. Давайте рассмотрим, какие есть программные способы закрытия. Также узнаем, какие обработчики диалога можно использовать, чтобы отследить закрытие.

Создадим проект:

**Project name**: P0661\_AlertDialogOperations  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogOperations  
**Package name**: ru.startandroid.develop.p0661alertdialogoperations  
**Create Activity**: MainActivity

В **strings.xml** пропишем тексты:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="dialog">Диалог</string>      <string name="app\_name">P0661\_AlertDialogOperations</string>  </resources> |

**main.xml**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="vertical">      <Button          android:id="@+id/button1"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="@string/dialog"          android:onClick="onclick">      </Button>  </LinearLayout> |

**MainActivity.java**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 | package ru.startandroid.develop.p0661alertdialogoperations;    import android.app.Activity;  import android.app.AlertDialog;  import android.app.Dialog;  import android.content.DialogInterface;  import android.content.DialogInterface.OnCancelListener;  import android.content.DialogInterface.OnDismissListener;  import android.content.DialogInterface.OnShowListener;  import android.os.Bundle;  import android.os.Handler;  import android.util.Log;  import android.view.View;    public class MainActivity extends Activity {      final String LOG\_TAG = "myLogs";    final int DIALOG = 1;      Dialog dialog;      /\*\* Called when the activity is first created. \*/    public void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.main);    }      @Override    protected Dialog onCreateDialog(int id) {      if (id == DIALOG) {        Log.d(LOG\_TAG, "Create");        AlertDialog.Builder adb = new AlertDialog.Builder(this);        adb.setTitle("Title");        adb.setMessage("Message");        adb.setPositiveButton("OK", null);        dialog = adb.create();          // обработчик отображения        dialog.setOnShowListener(new OnShowListener() {          public void onShow(DialogInterface dialog) {            Log.d(LOG\_TAG, "Show");          }        });          // обработчик отмены        dialog.setOnCancelListener(new OnCancelListener() {          public void onCancel(DialogInterface dialog) {            Log.d(LOG\_TAG, "Cancel");          }        });          // обработчик закрытия        dialog.setOnDismissListener(new OnDismissListener() {          public void onDismiss(DialogInterface dialog) {            Log.d(LOG\_TAG, "Dismiss");          }        });        return dialog;      }      return super.onCreateDialog(id);    }      public void onclick(View v) {      showDialog(DIALOG);    }  } |

Код в основном должен быть понятен по прошлым урокам. Создаем диалог, настраиваем заголовок, сообщение и одну кнопку без обработчика (нам он сейчас не нужен). Далее для диалога указываем три обработчика: отображения, отмены и закрытия диалога. Все они пишут о себе в лог.

**onclick** – обработчик кнопки из main.xml. Здесь мы просто запускаем диалог.

Обработчики

Давайте смотреть, когда и какие обработчики событий диалога будут срабатывать. Все сохраним и запустим. Жмем кнопку Диалог, появляется диалог.



В логе видим:

*Create  
Show*

Диалог **создался**и сработал **обработчик отображения** диалога. Нажмем кнопку ОК. Диалог закрылся, а лог показал следующее:

*Dismiss*

Сработал **обработчик закрытия** диалога.

Теперь еще раз запустим диалог кнопкой Диалог. В логе видим:

*Show*

Метод onCreateDialog не отработал, т.к. диалог уже создан. Это мы подробно рассматривали в прошлых уроках. Сработал обработчик отображения.

Для закрытия диалога нажмем кнопку Back (Назад) на эмуляторе. В логе появились следующие строки:

*Cancel  
Dismiss*

Перед обработчиком закрытия (Dismiss) сработал **обработчик отмены** (Cancel), т.к. диалог был отменен.

Операции

Разберем программные методы управления диалогом. Для этого немного изменим код MainActivity.java. Добавим два пустых пока метода **method1**и **method2**, и перепишем **onclick**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | void method1() {  }    void method2() {  }    public void onclick(View v) {    showDialog(DIALOG);      Handler h = new Handler();      h.postDelayed(new Runnable() {      public void run() {        method1();      }    }, 2000);      h.postDelayed(new Runnable() {      public void run() {        method2();      }    }, 4000);  } |

Handler мы пока не проходили, его понимать необязательно. Сейчас просто надо принять, что вся эта конструкция в **onclick**покажет диалог, затем через 2 секунды выполнит метод method1 и еще через 2 секунды выполнит метод method2. Т.е. получится такая последовательность:

*отображение диалога  
2 сек  
выполнение method1  
2 сек  
выполнение method2*

dismiss

Мы будем работать с method1 и method2. Начнем с метода [dismiss](http://developer.android.com/reference/android/app/Dialog.html" \l "dismiss()" \t "_blank) – он закрывает диалог. Перепишем метод method1:

|  |  |
| --- | --- |
| 1  2  3 | void method1() {    dialog.dismiss();  } |

method2 пока не трогаем. Запустим приложение и нажмем кнопку Диалог. Диалог появился, повисел две секунды и закрылся. Это нам обеспечил **dismiss**, вызванный через 2 секунды после отображения диалога. Смотрим лог:

*Create  
Show  
Dismiss*

Все верно. Диалог создался, отобразился и закрылся. Обратите внимание на время записей в логе. Между *Show* и *Dismiss* должно быть примерно 2 секунды.

cancel

Теперь используем метод [cancel](http://developer.android.com/reference/android/app/Dialog.html" \l "cancel()" \t "_blank). Перепишем method1:

|  |  |
| --- | --- |
| 1  2  3 | void method1() {    dialog.cancel();  } |

Все сохраним, запустим приложение и вызовем диалог. Снова диалог появился и закрылся через две секунды. Сработал метод **cancel**. Логи:

*Create  
Show  
Cancel  
Dismiss*

Все так же, как при закрытии диалога кнопкой Back.

hide

Снова перепишем method1, используя метод [hide](http://developer.android.com/reference/android/app/Dialog.html" \l "hide()" \t "_blank):

|  |  |
| --- | --- |
| 1  2  3 | void method1() {    dialog.hide();  } |

Запустим приложение, вызовем диалог. Он отобразился и закрылся. Смотрим лог:

*Create  
Show*

На этот раз **обработчик закрытия** не сработал. Диалог просто скрылся. Зачем это нужно, я не знаю, но метод такой есть, поэтому я рассказал о нем.

Управление из Activity

Мы работали напрямую с объектом Dialog и вызывали его методы. Есть еще другой способ. Сначала немного теории о механизме взаимодействия Activity и диалога. Когда мы первый раз выполняем метод showDialog, мы передаем туда ID. Это ID далее передается в onCreateDialog.  В итоге onCreateDialog возвращает созданный диалог, и Activity для себя увязывает его с ID.  И если мы захотим обратиться к этому диалогу, нам нужен будет только ID, Activity сама по нему определит, какой диалог нам нужен.

Когда мы, например, следующие разы вызываем showDialog, мы передаем туда ID, но диалог не создается. Activity по ID находит ранее созданный диалог и показывает его. У Activity также есть методы по закрытию диалога – это [dismissDialog](http://developer.android.com/reference/android/app/Activity.html" \l "dismissDialog(int)" \t "_blank) и [removeDialog](http://developer.android.com/reference/android/app/Activity.html" \l "removeDialog(int)" \t "_blank). Первый просто закрывает диалог, а второй закрывает и заставляет Activity забыть про него. Т.е. когда мы в след.раз захотим показать этот диалог, Activity будет заново создавать его, а не брать уже готовый. Проверим это.

dismissDialog

Перепишем методы:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | void method1() {    dismissDialog(DIALOG);  }    void method2() {    showDialog(DIALOG);  } |

Все сохраним и запустим. Вызовем диалог и ждем. Диалог отобразился, через 2 секунды закрылся, и еще через 2 снова открылся. Смотрим лог:

*Create  
Show  
Dismiss  
Show*

Когда диалог отобразился второй раз, не сработал метод его создания, т.к. Activity использовало созданный при первом вызове объект.

removeDialog

Перепишем метод method1:

|  |  |
| --- | --- |
| 1  2  3 | void method1() {    removeDialog(DIALOG);  } |

Будем не только закрывать диалог, но и «забывать» его. method2 оставляем без изменений, он будет показывать диалог.

Запустим приложение, запустим диалог и ждем. Диалог открылся, закрылся и открылся снова. Смотрим лог:

*Create  
Show  
Dismiss  
Create  
Show*

Но на этот раз при втором показе он снова создавался, т.к. Activity его забыло благодаря методу removeDialog.

У объекта Dialog есть еще метод [show](http://developer.android.com/reference/android/app/Dialog.html" \l "show()" \t "_blank). Чем он отличается от метода Activity showDialog? show просто покажет созданный диалог, а showDialog, начинает проверять был ли уже создан диалог, создает его, если необходимо, и вызывает для него метод onPrepareDialog.

# [Диалоги. ProgressDialog](https://startandroid.ru/ru/uroki/vse-uroki-spiskom/128-urok-67-dialogi-progressdialog.html)

05 апреля 2012

В этом уроке:

- работаем с ProgressDialog

ProgressDialog позволяет показать пользователю, что идет какая-либо операция и надо подождать. Он бывает двух видов: просто вращающийся круг и полоса-индикатор, которая показывает процент выполнения. Сделаем приложение, которое будет показывать оба вида.

Создадим проект:

**Project name**: P0671\_ProgressDialog  
**Build Target**: Android 2.3.3  
**Application name**: ProgressDialog  
**Package name**: ru.startandroid.develop.p0671progressdialog  
**Create Activity**: MainActivity

В **strings.xml** пропишем тексты:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | <?xml version="1.0" encoding="utf-8"?>  <resources>      <string name="dflt">Обычный</string>      <string name="horiz">Горизонтальный</string>      <string name="app\_name">ProgressDialog</string>  </resources> |

**main.xml**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | <?xml version="1.0" encoding="utf-8"?>  <LinearLayout      xmlns:android="http://schemas.android.com/apk/res/android"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent"      android:orientation="vertical">      <Button          android:id="@+id/btnDefault"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="@string/dflt"          android:onClick="onclick">      </Button>      <Button          android:id="@+id/btnHoriz"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="@string/horiz"          android:onClick="onclick">      </Button>  </LinearLayout> |

Две кнопки: одна покажет диалог с крутящимся кругом (ProgressDialog по умолчанию), другая - с горизонтальной полосой индикатором

**MainActivity.java**:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70 | package ru.startandroid.develop.p0671progressdialog;    import android.app.Activity;  import android.app.Dialog;  import android.app.ProgressDialog;  import android.content.DialogInterface;  import android.content.DialogInterface.OnClickListener;  import android.os.Bundle;  import android.os.Handler;  import android.os.Message;  import android.view.View;    public class MainActivity extends Activity {      ProgressDialog pd;    Handler h;      /\*\* Called when the activity is first created. \*/    @Override    public void onCreate(Bundle savedInstanceState) {      super.onCreate(savedInstanceState);      setContentView(R.layout.main);      }      public void onclick(View v) {      switch (v.getId()) {      case R.id.btnDefault:        pd = new ProgressDialog(this);        pd.setTitle("Title");        pd.setMessage("Message");        // добавляем кнопку        pd.setButton(Dialog.BUTTON\_POSITIVE, "OK", new OnClickListener() {          public void onClick(DialogInterface dialog, int which) {          }        });        pd.show();        break;      case R.id.btnHoriz:        pd = new ProgressDialog(this);        pd.setTitle("Title");        pd.setMessage("Message");        // меняем стиль на индикатор        pd.setProgressStyle(ProgressDialog.STYLE\_HORIZONTAL);        // устанавливаем максимум        pd.setMax(2148);        // включаем анимацию ожидания        pd.setIndeterminate(true);        pd.show();        h = new Handler() {          public void handleMessage(Message msg) {            // выключаем анимацию ожидания            pd.setIndeterminate(false);            if (pd.getProgress() < pd.getMax()) {              // увеличиваем значения индикаторов              pd.incrementProgressBy(50);              pd.incrementSecondaryProgressBy(75);              h.sendEmptyMessageDelayed(0, 100);            } else {              pd.dismiss();            }          }        };        h.sendEmptyMessageDelayed(0, 2000);        break;      default:        break;      }    }  } |

Для первого диалога все почти как обычно. Сначала настраиваем заголовок и сообщение. Затем добавляем кнопку. Здесь немного по-другому, чем с AlertDialog.Builder. Мы используем метод [setButton](http://developer.android.com/reference/android/app/AlertDialog.html" \l "setButton(int,%20java.lang.CharSequence,%20android.os.Message)" \t "_blank). На вход передаем тип кнопки, текст и обработчик. Тип кнопки определяется константами: [BUTTON\_POSITIVE](http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_POSITIVE), [BUTTON\_NEGATIVE](http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_NEGATIVE), [BUTTON\_NEUTRAL](http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_NEUTRAL). В обработчике я ничего не пишу, мне он не нужен сейчас. Далее показываем диалог методом show.

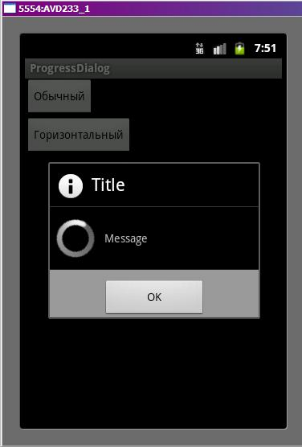
Для второго диалога указываем, что он будет индикатором. Это делается с помощью метода [setProgressStyle](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "setProgressStyle(int)" \t "_blank). Далее задаем заголовок и сообщение. Кнопку не добавляем, но если вдруг нужна будет, то это делается полностью  аналогично первому диалогу. Метод [setMax](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "setMax(int)" \t "_blank) устанавливает максимальное значение заполнения индикатора. Значение 2148 я выбрал случайно.  Метод [setIndeterminate](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "setIndeterminate(boolean)" \t "_blank) включает анимацию индикатора, имитирующую ожидание. Метод show показывает диалог.

Далее снова приходится использовать незнакомый нам Handler. Понимать его действие сейчас необязательно, позже мы его еще будем проходить. Сейчас я просто распишу на словах, что он делает. Handler ждет 2 секунды (просто имитация, например, подключения к серверу), выключает анимацию ожидания (setIndeterminate), затем каджые 100 миллисекунд увеличивает значение основного ([incrementProgressBy](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "incrementProgressBy(int)" \t "_blank)) и дополнительного ([incrementSecondaryProgressBy](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "incrementSecondaryProgressBy(int)" \t "_blank)) индикатора, пока основной индикатор ([getProgress](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "getProgress()" \t "_blank)) не достигнет максимума ([getMax](http://developer.android.com/reference/android/app/ProgressDialog.html" \l "getMax()" \t "_blank), в нашем случае = 2148). После этого диалог закрываем (dismiss).

Пример дополнительного индикатора вы могли видеть на YouTube. Когда смотрите видео, там снизу по мере воспроизведения заполняется основной индикатор, а правее его заполняется еще один, затемненный, который показывает, сколько видео закэшировалось. Разумеется, если он вам не нужен, вы просто его не заполняете. Будет отображаться только основной.

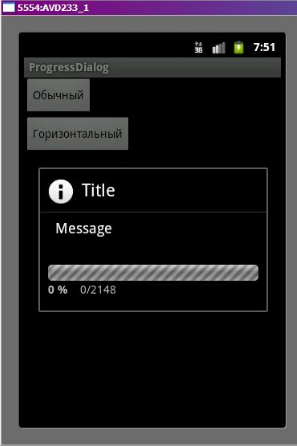
Все сохраним и запустим.

Вызовем первый диалог

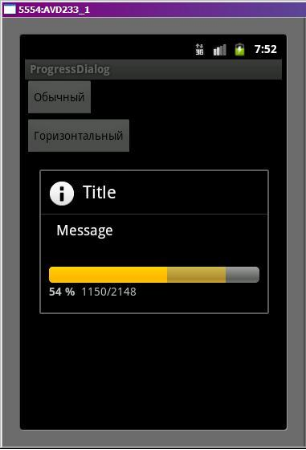


Бесконечно крутящийся круг показывает, что процесс идет. Но сколько осталось – непонятно. Не всегда можно спрогнозировать оставшееся время или показать процент выполнения задачи.

Закроем первый диалог и откроем второй. Он две секунды повисит с анимацией ожидания



А затем начнет заполнять индикатор (основной и дополнительный).



Когда основной индикатор заполнится,  диалог закроется.

Думаю, по диалогам тему можно закрывать. Мы рассмотрели их достаточно подробно. То, что пока непонятен Handler – это ничего страшного, через несколько уроков возьмемся и за него. Если вдруг чего осталось непонятным, велкам на [форум](http://forum.startandroid.ru/), будем разбираться )

## Передача данных в диалоговое окно

Передача данных в диаговое окно, как и в любой фрагмент, осуществляется с помощью объекта Bundle.

Так, определим в файле **activity\_main.xml** список ListView:

|  |
| --- |
| <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      android:orientation="vertical">      <ListView          android:id="@+id/phonesList"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent"  />  </LinearLayout> |

В классе **MainActivity** определим для этого списка данные:

|  |
| --- |
| package com.example.eugene.dialogsapp;    import android.support.v7.app.AppCompatActivity;  import android.os.Bundle;  import android.view.View;  import android.widget.AdapterView;  import android.widget.ArrayAdapter;  import android.widget.ListView;    import java.util.ArrayList;    public class MainActivity extends AppCompatActivity {        @Override      public void onCreate(Bundle savedInstance) {          super.onCreate(savedInstance);          setContentView(R.layout.activity\_main);            ListView phonesList = (ListView) findViewById(R.id.phonesList);          ArrayList<String> phones = new ArrayList<>();          phones.add("Google Pixel");          phones.add("Huawei P9");          phones.add("LG G5");          phones.add("Samsung Galaxy S8");            final ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple\_list\_item\_1, phones);          phonesList.setAdapter(adapter);            phonesList.setOnItemClickListener(new AdapterView.OnItemClickListener() {              @Override              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {                    String selectedPhone = adapter.getItem(position);                  CustomDialogFragment dialog = new CustomDialogFragment();                  Bundle args = new Bundle();                  args.putString("phone", selectedPhone);                  dialog.setArguments(args);                  dialog.show(getSupportFragmentManager(), "custom");              }          });      }  } |

В обработчике нажатия на элемент в списке получаем выбранный элемент и добавляем его в объект Bundle. Далее через метод dialog.setArguments() передаем данные из Bundle во фрагмент.

Теперь определим следующий класс фрагмента **CustomDialogFragment**:

|  |
| --- |
| package com.example.eugene.dialogsapp;    import android.app.AlertDialog;  import android.app.Dialog;  import android.os.Bundle;  import android.support.annotation.NonNull;  import android.support.v4.app.DialogFragment;    public class CustomDialogFragment extends DialogFragment  {        @NonNull      public Dialog onCreateDialog(Bundle savedInstanceState) {            String phone = getArguments().getString("phone");          AlertDialog.Builder builder=new AlertDialog.Builder(getActivity());          return builder                  .setTitle("Диалоговое окно")                  .setIcon(android.R.drawable.ic\_dialog\_alert)                  .setMessage("Вы хотите удалить " + phone + "?")                  .setPositiveButton("OK", null)                  .setNegativeButton("Отмена", null)                  .create();      }  } |

С помощью метода getArguments() получаем переданный в MainActivity объект Bundle. И так как была передана строка, то для ее извлечения применяется метод getString().

И при нажатии элемент списка будет передаваться в диалоговое окно:

