**Диалоги. TimePickerDialog**

В этом уроке:

- используем TimePickerDialog

Начнем рассматривать новую тему – **диалоги**. Диалог – небольшое всплывающее окно, которое чаще всего используется для подтверждения каких-либо операций или ввода небольшого количества данных.

Мы рассмотрим стандартные диалоги, предоставляемые системой, и научимся делать свои. На этом уроке рассмотрим [**TimePickerDialog**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/TimePickerDialog.html). Это диалог, который позволяет указать время.

Сделаем простое приложение. На экране будет TextView, на него можно будет нажать и появится диалог для выбора времени. После выбора оно отобразится в TextView.

Создадим проект:

**Project name**: P0581\_TimePickerDialog  
**Build Target**: Android 2.3.3  
**Application name**: TimePickerDialog  
**Package name**: ru.startandroid.develop.p0581timepickerdialog  
**Create Activity**: MainActivity

Экран **main.xml**:

<?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"*>  
<TextView  
 android:id=*"@+id/tvTime"*  
 android:layout\_width=*"fill\_parent"*  
 android:layout\_height=*"wrap\_content"*  
 android:clickable=*"true"*  
 android:onClick=*"onclick"*  
 android:text=*"@string/hello"*  
 android:textSize=*"22sp"*>  
</TextView>  
</LinearLayout>

На экране **TextView**, для которого включим **clickable** и назначим метод обработки **onclick**.

Код **MainActivity.java**:

**package** ru.startandroid.develop.p0581timepickerdialog;  
  
**import** android.app.Activity;  
**import** android.app.Dialog;  
**import** android.app.TimePickerDialog;  
**import** android.app.TimePickerDialog.OnTimeSetListener;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.TextView;  
**import** android.widget.TimePicker;  
  
**public class** MainActivity **extends** Activity {  
    
  **int** DIALOG\_TIME = 1;  
  **int** myHour = 14;  
  **int** myMinute = 35;  
  TextView tvTime;  
    
    /\*\* Called when the activity is first created. \*/  
    **public void** onCreate(Bundle savedInstanceState) {  
        **super**.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        tvTime = (TextView) findViewById(R.id.tvTime);  
    }  
      
    **public void** onclick(View view) {  
      showDialog(DIALOG\_TIME);  
    }  
      
    **protected** Dialog onCreateDialog(**int** id) {  
      **if** (id == DIALOG\_TIME) {  
        TimePickerDialog tpd = **new** TimePickerDialog(this, myCallBack, myHour, myMinute, **true**);  
        **return** tpd;  
      }  
      **return super**.onCreateDialog(id);  
    }  
      
    OnTimeSetListener myCallBack = **new** OnTimeSetListener() {  
    **public void** onTimeSet(TimePicker view, **int** hourOfDay, **int** minute) {  
      myHour = hourOfDay;  
      myMinute = minute;   
      tvTime.setText("Time is " + myHour + " hours " + myMinute + " minutes");  
    }  
  };  
}

В **onCreate** находим TextView.

В **onClick** вызываем метод showDialog и передаем ему ID диалога. Этот метод (showDialog) создает диалог с помощью отдельного метода и показывает его. ID используется для указания, какой именно диалог создавать и показывать.

Метод **onCreateDialog** – это и есть отдельный метод, который вызывается в showDialog для создания диалога. В этом методе мы смотрим, какой ID пришел на вход, создаем соответствующий диалог и возвращаем его.

В нашем случае мы создаем TimePickerDialog, используя конструктор:

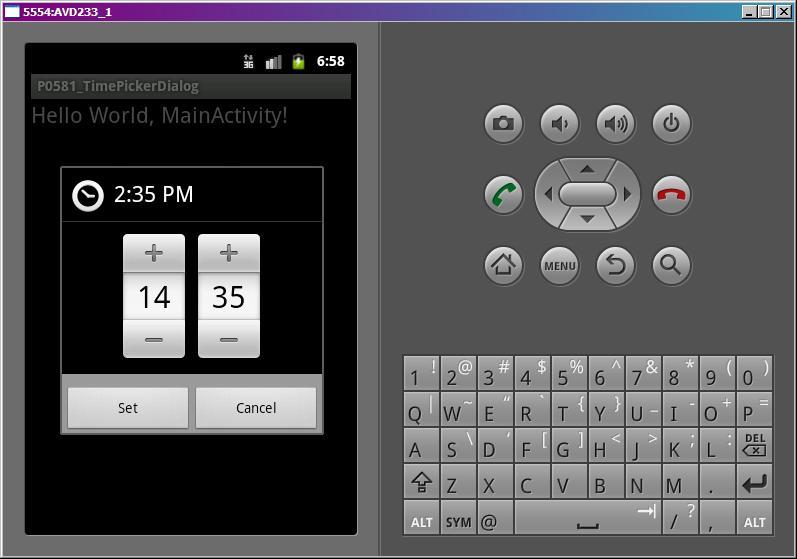
[**TimePickerDialog (Context context, TimePickerDialog.OnTimeSetListener callBack, int hourOfDay, int minute, boolean is24HourView)**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/TimePickerDialog.html#TimePickerDialog(android.content.Context, android.app.TimePickerDialog.OnTimeSetListener, int, int, boolean)), где

*context* – контекст  
*callBack* – это обработчик с интерфйесом [**TimePickerDialog.OnTimeSetListener**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/TimePickerDialog.OnTimeSetListener.html), метод которого срабатывает при нажатии кнопки ОК на диалоге  
*hourOfDay* – час, который покажет диалог  
*minute* – минута, которую покажет диалог  
*is24HourView* – формат времени 24 часа (иначе AM/PM)

**myCallBack** – объект, реализующий интерфейс TimePickerDialog.OnTimeSetListener. У него только один метод – onTimeSet, который предоставляет нам [**TimePicker**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/widget/TimePicker.html) из диалога, и час и минуту, которые он показывает. Т.е. то, что мы ввели в диалоге.

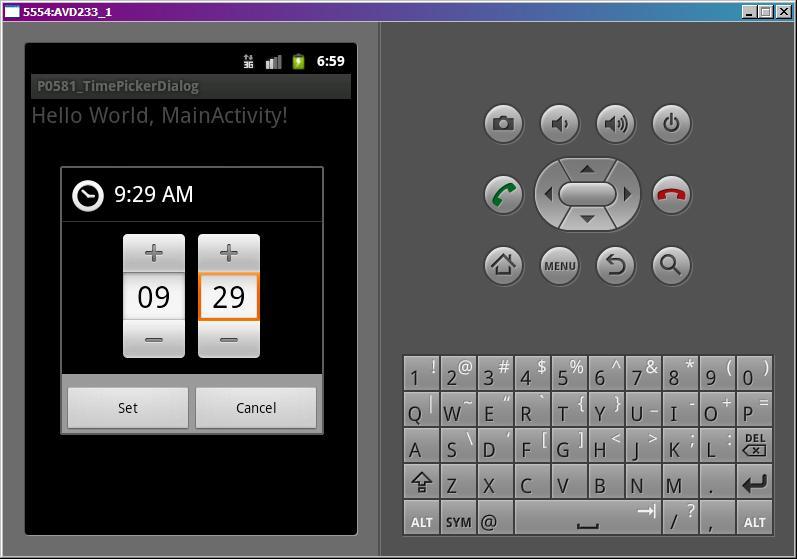
Эти данные мы пишем в tvTime.

Все сохраним и запустим. Нажмем на TextView. Появился диалог для ввода времени.

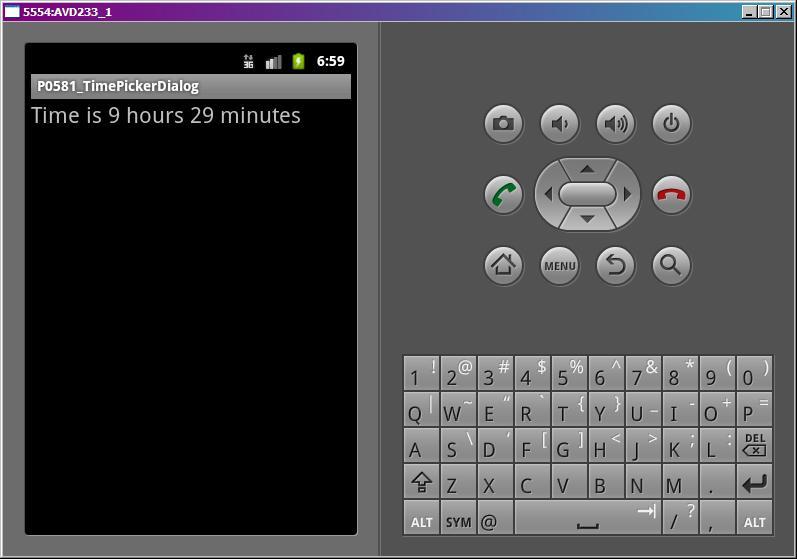


Сейчас он показывает 14:35, т.к. мы при создании передали ему значения myHour и myMinute.

Поменяем значения



и нажмем Set.



Текст показал новые значения.

Диалог также можно закрыть кнопкой **Cancel**или кнопкой **Back** (назад)на эмуляторе. В этом случае метод обработчика не сработает и текст не обновится.

Это самый простейший диалог. И тут, разумеется, рассмотрены далеко не все возможности. Здесь пока важно понять схему взаимодействия showDialog и onCreateDialog, и наличие обработчика результатов диалога.

Также здесь использовался новый для нас компонент TimePicker. Он несложен и просто позволяет удобно задавать или отображать время.

**Диалоги. DatePickerDialog**

В этом уроке:

- используем DatePickerDialog

Урок будет аналогичен прошлому, но теперь в диалоге мы будем задавать дату, а не время. Такой диалог называется [**DatePickerDialog**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/DatePickerDialog.html).

Сделаем простое приложение. На экране будет TextView, на него можно будет нажать и появится диалог для выбора даты. После выбора она отобразится в TextView.

Создадим проект:

**Project name**: P0591\_DatePickerDialog  
**Build Target**: Android 2.3.3  
**Application name**: DatePickerDialog  
**Package name**: ru.startandroid.develop.p0591datepickerdialog  
**Create Activity**: MainActivity

Экран **main.xml**:

<?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"*>  
<TextView  
 android:id=*"@+id/tvDate"*  
 android:layout\_width=*"fill\_parent"*  
 android:layout\_height=*"wrap\_content"*  
 android:clickable=*"true"*  
 android:onClick=*"onclick"*  
 android:text=*"@string/hello"*  
 android:textSize=*"22sp"*>  
</TextView>  
</LinearLayout>

Код **MainActivity.java**:

**package** ru.startandroid.develop.p0591datepickerdialog;  
  
**import** android.app.Activity;  
**import** android.app.DatePickerDialog;  
**import** android.app.DatePickerDialog.OnDateSetListener;  
**import** android.app.Dialog;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.DatePicker;  
**import** android.widget.TextView;  
  
**public class** MainActivity **extends** Activity {  
    
  **int** DIALOG\_DATE = 1;  
  **int** myYear = 2011;  
  **int** myMonth = 02;  
  **int** myDay = 03;  
  TextView tvDate;  
    
    /\*\* Called when the activity is first created. \*/  
  
    **public void** onCreate(Bundle savedInstanceState) {  
        **super**.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        tvDate = (TextView) findViewById(R.id.tvDate);  
    }  
       
    **public void** onclick(View view) {  
      showDialog(DIALOG\_DATE);  
    }  
      
  
    **protected** Dialog onCreateDialog(**int** id) {  
      **if** (id == DIALOG\_DATE) {  
        DatePickerDialog tpd = **new** DatePickerDialog(this, myCallBack, myYear, myMonth, myDay);  
        **return** tpd;  
      }  
      **return super**.onCreateDialog(id);  
    }  
      
    OnDateSetListener myCallBack = **new** OnDateSetListener() {  
  
    **public void** onDateSet(DatePicker view, **int** year, **int** monthOfYear,  
        **int** dayOfMonth) {  
      myYear = year;  
      myMonth = monthOfYear;  
      myDay = dayOfMonth;  
      tvDate.setText("Today is " + myDay + "/" + myMonth + "/" + myYear);  
    }  
    };  
}

В **onCreate**находим TextView.

В **onClick**вызываем метод showDialog и передаем ему ID диалога. Этот метод (showDialog) создает диалог с помощью отдельного метода и показывает его. ID используется для указания, какой именно диалог создавать и показывать.

Метод **onCreateDialog**– это и есть отдельный метод, который вызывается в showDialog для создания диалога. В этом методе мы смотрим, какой ID пришел на вход, создаем соответствующий диалог и возвращаем его.

В нашем случае мы создаем DatePickerDialog, используя конструктор:

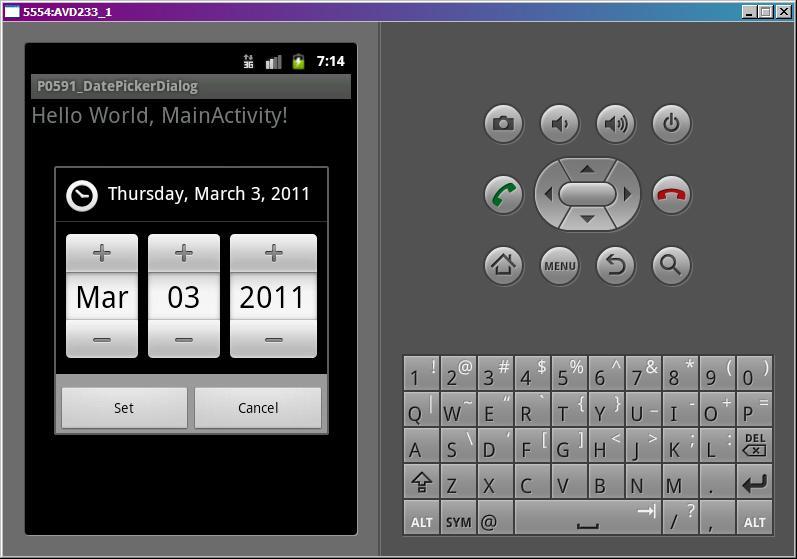
[**DatePickerDialog (Context context, DatePickerDialog.OnDateSetListener callBack, int year, int monthOfYear, int dayOfMonth)**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/DatePickerDialog.html#DatePickerDialog(android.content.Context, android.app.DatePickerDialog.OnDateSetListener, int, int, int)), где

context – контекст  
callBack – это обработчик с интерфейсом [**DatePickerDialog.OnDateSetListener**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/DatePickerDialog.OnDateSetListener.html), метод которого срабатывает при нажатии кнопки ОК на диалоге  
year – год, который покажет диалог  
monthOfYear – месяц, который покажет диалог  
dayOfMonth – день, который покажет диалог

**myCallBack** – объект, реализующий интерфейс DatePickerDialog.OnDateSetListener. У него только один метод – onDateSet, который предоставляет нам [**DatePicker**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/widget/DatePicker.html) из диалога, и год, месяц и день, которые он показывает. Т.е. то, что мы ввели в диалоге.

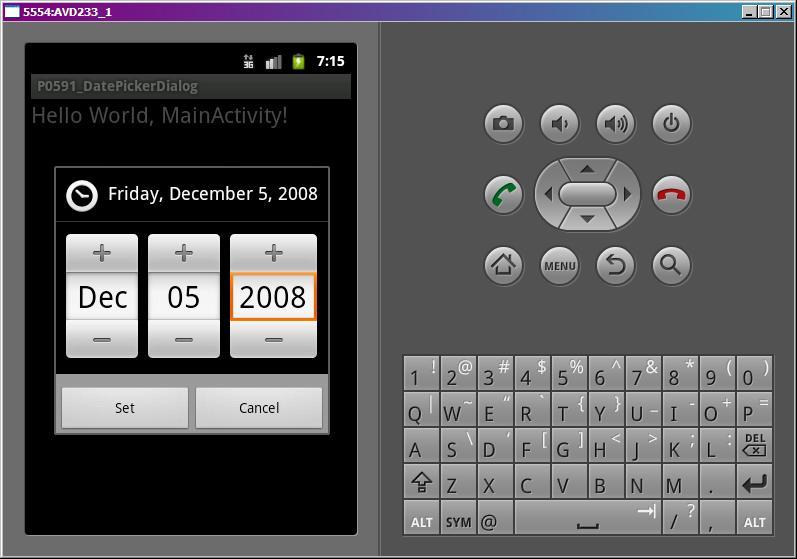
Эти данные мы пишем в tvDate.

Все сохраним и запустим. Нажмем на TextView. Появился диалог для ввода даты.

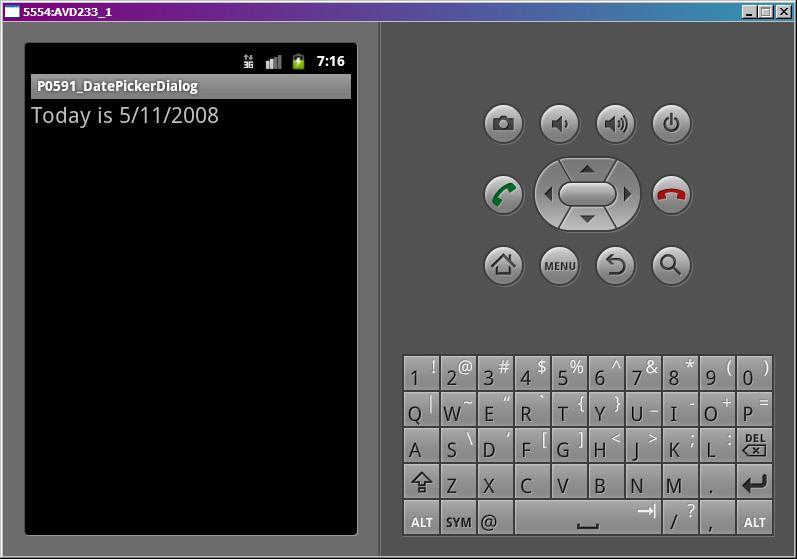


Сейчас он показывает 3 марта 2011, т.к. мы при создании передали ему значения myYear, myMonth и myDay. Месяцы он считает, начиная с нуля.

Поменяем значения



и нажмем Set.



Текст показал новые значения. Обратите внимание, что месяц декабрь он выдал как число 11. Месяцы нумеруются с нуля.

Диалог также можно закрыть кнопкой Cancel или кнопкой Back на эмуляторе. В этом случае обработчик не сработает и текст не обновится.

**Диалоги. AlertDialog: Title, Message, Icon, Buttons**

В этом уроке:

- создаем AlertDialog  
- настраиваем заголовок, сообщение, картинку и кнопки

Начнем знакомство с [**AlertDialog**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.html). Этот **диалог**используется, если вы хотите сообщить о чем-то пользователю или попросить его сделать выбор типа Да/Нет/Отмена.

Напишем приложение, которое при закрытии будет вызывать диалог о сохранении данных, аналогичный диалогу из программ MS Office . Если мы ответим Да, то данные сохранятся, если Нет – то не сохранятся, если Отмена – приложение не закроется.

Создадим проект:

**Project name**: P0601\_AlertDialogSimple  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogSimple  
**Package name**: ru.startandroid.develop.p0601alertdialogsimple  
**Create Activity**: MainActivity

Добавим в **res/values/strings.xml** строки c текстами:

<?xml version=*"1.0"* encoding=*"utf-8"*?>  
<resources>  
<string name=*"app\_name"*>AlertDialogSimple</string>  
<string name=*"exit"*>Выход</string>  
<string name=*"save\_data"*>Сохранить данные?</string>  
<string name=*"yes"*>Да</string>  
<string name=*"no"*>Нет</string>  
<string name=*"cancel"*>Отмена</string>  
<string name=*"saved"*>Сохранено</string>  
</resources>

Экран **main.xml**:

<?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/btnExit"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/exit"*  
 android:onClick=*"onclick"*>  
</Button>  
</LinearLayout>

Код **MainActivity.java**:

**package** ru.startandroid.develop.p0601alertdialogsimple;  
  
**import** android.app.Activity;  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.content.DialogInterface;  
**import** android.content.DialogInterface.OnClickListener;  
**import** android.os.Bundle;  
**import** android.view.View;  
**import** android.widget.Toast;  
  
**public class** MainActivity **extends** Activity {  
    
  **final int** DIALOG\_EXIT = 1;  
    
    /\*\* Called when the activity is first created. \*/  
    **public void** onCreate(Bundle savedInstanceState) {  
        **super**.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
      
    **public void** onclick(View v) {  
      // вызываем диалог  
      showDialog(DIALOG\_EXIT);  
    }  
      
    **protected** Dialog onCreateDialog(**int** id) {  
      **if** (id == DIALOG\_EXIT) {  
        AlertDialog.Builder adb = **new** AlertDialog.Builder(**this**);  
        // заголовок  
        adb.setTitle(R.string.exit);  
        // сообщение  
        adb.setMessage(R.string.save\_data);  
        // иконка  
        adb.setIcon(android.R.drawable.ic\_dialog\_info);  
        // кнопка положительного ответа  
        adb.setPositiveButton(R.string.yes, myClickListener);  
        // кнопка отрицательного ответа  
        adb.setNegativeButton(R.string.no, myClickListener);  
        // кнопка нейтрального ответа  
        adb.setNeutralButton(R.string.cancel, myClickListener);  
        // создаем диалог  
        **return** adb.create();  
      }  
      **return super**.onCreateDialog(id);  
    }  
      
    OnClickListener myClickListener = **new** OnClickListener() {  
    **public void** onClick(DialogInterface dialog, **int** which) {  
      **switch** (which) {  
      // положительная кнопка  
      **case** Dialog.BUTTON\_POSITIVE:  
        saveData();  
        finish();  
        **break**;  
      // негаитвная кнопка  
      **case** Dialog.BUTTON\_NEGATIVE:  
        finish();  
        **break**;  
      // нейтральная кнопка    
      **case** Dialog.BUTTON\_NEUTRAL:  
        **break**;  
      }  
    }  
  };  
    
  **void** saveData() {  
    Toast.makeText(this, R.string.saved, Toast.LENGTH\_SHORT).show();  
  }  
}

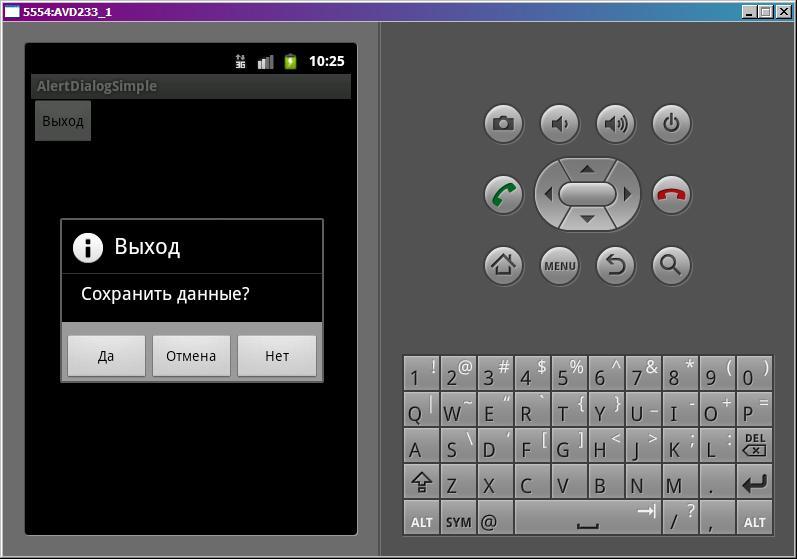
В обработчике кнопки **onclick** вызываем диалог.

В **onCreateDialog** мы создаем диалог. Для этого используется класс [**AlertDialog.Builder**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html). Мы указываем заголовок, текст сообщения, иконку и кнопки. Диалог может содержать максимум три кнопки ответа: положительная, отрицательная и нейтральная. Для каждой указываем текст и обработчик. Метод [**create**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#create())создает диалог и мы его возвращаем (return).

Обработчик кнопок **myClickListener**реализует интерфейс [**DialogInterface.OnClickListener**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.OnClickListener.html) и в нашем случае является общим для всех кнопок. В нем мы проверяем, какая кнопка была нажата:  
если положительная ([**BUTTON\_POSITIVE**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_POSITIVE)), то сохраняем данные и закрываем приложение  
если отрицательная ([**BUTTON\_NEGATIVE**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_NEGATIVE)), то закрываем приложение без сохранения  
если нейтральная ([**BUTTON\_NEUTRAL**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_NEUTRAL)), то не делаем ничего

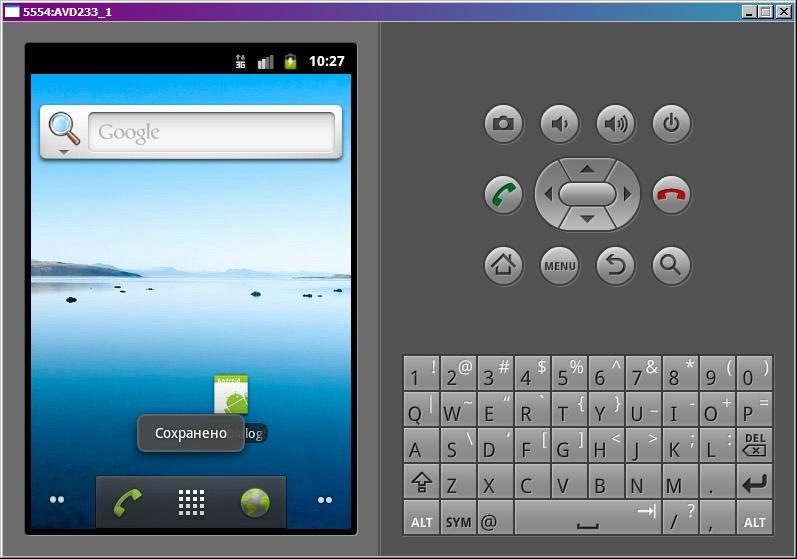
В своем методе saveData выводим текст, что данные как-будто сохранены. Просто, чтобы убедиться, что метод выполняется.

Все сохраним и запустим приложение. Нажмем кнопку Выход:



если жмем **Да**,

то приложение закроется и метод**saveData будет выполнен**.



Если жмем **Отмена**, то диалог закроется и с приложением ничего не произойдет.  
А если жмем **Нет**, то приложение закроется**без вызова** нашего метода **saveData**.

Вот так несложно и недолго создать диалог для взаимодействия с пользователем. Использованные нами атрибуты диалога **не являются обязательными**. Вы можете, например, не указывать заголовок, или сделать только одну кнопку, а не три.

Для указания заголовка, сообщения и текстов кнопок необязательно использовать переменные **R.string**.  Есть аналогичные реализации методов, которые принимают на вход просто текст.

И еще пара советов.

1) Чтобы диалог вызывался не только по кнопке выход, но и при нажатии на кнопку **Назад**в приложении, добавьте вызов диалога в реализацию метода [**onBackPressed**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Activity.html#onBackPressed())

**public void** onBackPressed() {  
      // вызываем диалог  
      showDialog(DIALOG\_EXIT);  
    }

2) А если хотите, чтобы вызванный диалог не закрывался по нажатию кнопки Назад, то используйте метод [**setCancelable**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Dialog.html#setCancelable(boolean)):

adb.setCancelable(**false**);

**Диалоги. AlertDialog.Метод onPrepareDialog**

В этом уроке:

- используем метод подготовки диалога

Когда мы создаем диалог в методе [**onCreateDialog**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Activity.html#onCreateDialog(int)), **Activity**складывает его в кучу созданных диалогов. И когда надо отобразить, достает его и показывает. Т.е. метод **onCreateDialog**выполняется только один раз для диалога. И если вам надо перед отображением что-то изменить, надо использовать метод[**onPrepareDialog**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Activity.html#onPrepareDialog(int, android.app.Dialog)). Этот метод вызывается каждый раз перед показом диалога.

Напишем приложение, в нем будет AlertDialog, который будет показывать текущее время.

**Создадим проект:**

**Project name**: P0611\_AlertDialogPrepare  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogPrepare  
**Package name**: ru.startandroid.develop.p0611alertdialogprepare  
**Create Activity**: MainActivity

На экране **main.xml** только кнопка:

<?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:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/hello"*  
 android:onClick=*"onclick"*>  
</Button>  
</LinearLayout>

**MainActivity.java:**

**package** ru.startandroid.develop.p0611alertdialogprepare;  
  
**import** java.sql.Date;  
**import** java.text.SimpleDateFormat;  
  
**import** android.app.Activity;  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
  
**public class** MainActivity **extends** Activity {  
    
  **final static** String LOG\_TAG = "myLogs";  
  **final int** DIALOG = 1;  
  SimpleDateFormat sdf = **new** SimpleDateFormat("HH:mm:ss");  
    
    /\*\* 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) {  
      showDialog(DIALOG);  
    }  
      
    **protected** Dialog onCreateDialog(**int** id) {  
      Log.d(LOG\_TAG, "onCreateDialog");  
      **if** (id == DIALOG) {  
        AlertDialog.Builder adb = **new** AlertDialog.Builder(**this**);  
        adb.setTitle("Текущее время");  
        adb.setMessage(sdf.format(**new** Date(System.currentTimeMillis())));  
        **return** adb.create();  
      }  
      **return super**.onCreateDialog(id);  
    }  
      
    **protected void** onPrepareDialog(**int** id, Dialog dialog) {  
      **super**.onPrepareDialog(id, dialog);  
      Log.d(LOG\_TAG, "onPrepareDialog");  
      **if** (id == DIALOG) {  
        ((AlertDialog)dialog).setMessage(sdf.format(**new** Date(System.currentTimeMillis())));  
      }  
    }  
}

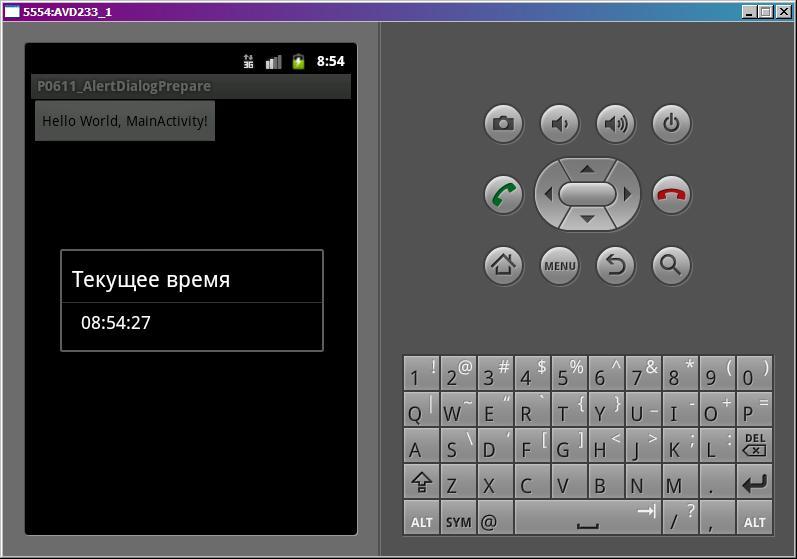
[**SimpleDateFormat**](http://elsof.ru/go.php?url=http://developer.android.com/reference/java/text/SimpleDateFormat.html) – это класс, позволяющий выводить дату-время в нужном формате. Задаем ему формат **HH:mm:ss** и он покажет время в классическом виде **часы:минуты:секунды**.

В **onCreateDialog** создаем диалог и устанавливаем заголовок и текст. Кнопки не добавляем, они не нужны. Диалог можно будет закрыть кнопкой Назад.

В **onPrepareDialog** мы на вход получаем ID вызываемого диалога и сам диалог (Dialog). Мы преобразуем его к AlertDialog и пишем в текст диалога текущее время.

В методах создания и подготовки диалога мы пишем лог, чтобы убедиться, что создание происходит один раз, а подготовка выполняется перед каждым показом.

Все сохраним и запустим приложение. Нажмем кнопку, появился диалог:

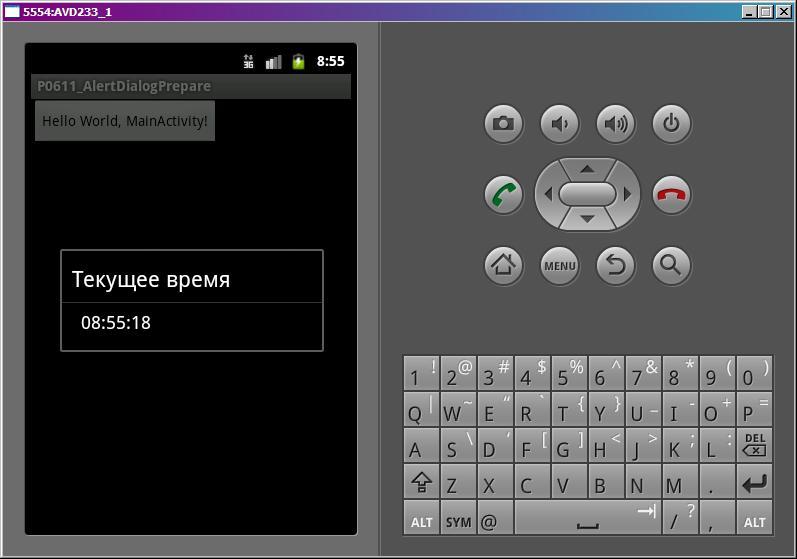


Показывает время. Смотрим лог:

*onCreateDialog  
onPrepareDialog*

Выполнились оба метода – создание и подготовка.

Закроем диалог (но не программу) и вызовем снова



Время обновилось, а в логах добавилась запись.

*onPrepareDialog*

На этот раз Activity просто достало созданный ранее диалог и выполнило метод его подготовки.

Т.е. когда мы показываем диалог **первый**раз, он проходит через методы **создания**и **подготовки**. Далее мы его закрываем, но при этом объект не уничтожается, а **Activity сохраняет** его у себя. И когда мы **снова**хотим **отобразить**диалог, Activity **достает**его, прогоняет через **метод подготовки** и показывает.

В этом механизме есть небольшой изъян. Диалог, как и все экраны, состоит из набора View-компонентов. В зависимости от используемых при создании параметров диалог задает видимость этих View. Т.е. если вы при создании не задали, например, Message, то в созданном диалоге будет скрыта View (*setVisibility(View.GONE)*), которая отвечает за отображение текста Message. И если в методе подготовки диалога вы решите-таки Message указать, то диалог его просто не отобразит, т.к. структура задается при создании.

Я не нашел способ, как можно заставить диалог заново сформировать себя без удаления и создания заново. Если кто знает – пишите, добавлю в урок. А о том, как удалить диалог, чтобы при показе снова вызвался метод onCreateDialog, мы еще поговорим в следующих уроках.

**Урок 62. Диалоги. AlertDialog. Список**

В этом уроке:

- формируем список в диалоге

В диалог можно выводить не только **текст**, но и **список**значений. Диалоговый список может быть **трех**видов:

- без выбора элементов  
- с одиночным выбором  
- с множественным выбором

В этом уроке рассмотрим первый вид.

Научимся создавать диалог со списком, используя **массив**данных, **адаптер**или **курсор**. Кроме создания попробуем менять данные и обновлять список перед каждым показом. Для этого введем счетчик показов и будем показывать его в последней строке списка. Каждый вызов списка будет увеличивать счетчик на единицу и это должно отразиться в списке. Так мы убедимся, что он обновляется.

**Создадим проект:**

**Project name**: P0621\_AlertDialogItems  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogItems  
**Package name**: ru.startandroid.develop.p0621alertdialogitems  
**Create Activity**: MainActivity

В **strings.xml** пропишем тексты:

<?xml version=*"1.0"* encoding=*"utf-8"*?>  
<resources>  
<string name=*"app\_name"*>AlertDialogItems</string>  
<string name=*"items"*>Items</string>  
<string name=*"adapter"*>Adapter</string>  
<string name=*"cursor"*>Cursor</string>  
</resources>

На экране **main.xml** три кнопки:

<?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=*"horizontal"*>  
<Button  
 android:id=*"@+id/btnItems"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/items"*  
 android:onClick=*"onclick"*>  
</Button>  
<Button  
 android:id=*"@+id/btnAdapter"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/adapter"*  
 android:onClick=*"onclick"*>  
</Button>  
<Button  
 android:id=*"@+id/btnCursor"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/cursor"*  
 android:onClick=*"onclick"*>  
</Button>  
</LinearLayout>

Для работы с БД выделим отдельный класс **DB.java**:

**package** ru.startandroid.develop.p0621alertdialogitems;  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteDatabase.CursorFactory;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**public class** DB {  
    
  **private static final** String DB\_NAME = "mydb";  
  **private static final int** DB\_VERSION = 1;  
  **private static final** String DB\_TABLE = "mytab";  
    
  **public static final** String COLUMN\_ID = "\_id";  
  **public static final** String COLUMN\_TXT = "txt";  
    
  **private static final** String DB\_CREATE =   
    "create table " + DB\_TABLE + "(" +  
      COLUMN\_ID + " integer primary key, " +  
      COLUMN\_TXT + " text" +  
    ");";  
    
  **private final** Context mCtx;  
    
    
  **private** DBHelper mDBHelper;  
  **private** SQLiteDatabase mDB;  
    
  **public** DB(Context ctx) {  
    mCtx = ctx;  
  }  
    
  // открыть подключение  
  **public void** open() {  
    mDBHelper = **new** DBHelper(mCtx, DB\_NAME, null, DB\_VERSION);  
    mDB = mDBHelper.getWritableDatabase();  
  }  
    
  // закрыть подключение  
  **public void** close() {  
    **if** (mDBHelper!=**null**) mDBHelper.close();  
  }  
    
  // получить все данные из таблицы DB\_TABLE  
  **public** Cursor getAllData() {  
    **return** mDB.query(DB\_TABLE, null, null, null, null, null, **null**);  
  }  
    
  // изменить запись в DB\_TABLE  
  **public void** changeRec(**int** id, String txt) {  
    ContentValues cv = **new** ContentValues();  
    cv.put(COLUMN\_TXT, txt);  
    mDB.update(DB\_TABLE, cv, COLUMN\_ID + " = " + id, **null**);  
  }  
  
  // класс по созданию и управлению БД  
  **private class** DBHelper **extends** SQLiteOpenHelper {  
  
    **public** DBHelper(Context context, String name, CursorFactory factory,  
        **int** version) {  
      **super**(context, name, factory, version);  
    }  
  
    // создаем и заполняем БД  
    @Override  
    **public void** onCreate(SQLiteDatabase db) {  
      db.execSQL(DB\_CREATE);  
        
      ContentValues cv = **new** ContentValues();  
      **for** (**int** i = 1; i < 5; i++) {  
        cv.put(COLUMN\_ID, i);  
        cv.put(COLUMN\_TXT, "sometext " + i);  
        db.insert(DB\_TABLE, null, cv);  
      }  
    }  
  
    @Override  
    **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
    }  
  }  
}

Тут все как обычно. Методы для открытия и закрытия подключения, получение курсора с данными, и изменение записи по ID. Таблица всего из двух полей – \_id и txt. При создании вставляем в таблицу 4 записи.

**MainActivity.java**:

**package** ru.startandroid.develop.p0621alertdialogitems;  
  
**import** android.app.Activity;  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.content.DialogInterface;  
**import** android.content.DialogInterface.OnClickListener;  
**import** android.database.Cursor;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.BaseAdapter;  
**import** android.widget.CursorAdapter;  
**import** android.widget.ListAdapter;  
  
**public class** MainActivity **extends** Activity {  
  
  **final** String LOG\_TAG = "myLogs";  
    
  **final int** DIALOG\_ITEMS = 1;  
  **final int** DIALOG\_ADAPTER = 2;  
  **final int** DIALOG\_CURSOR = 3;  
  **int** cnt = 0;  
  DB db;  
  Cursor cursor;  
  
  String data[] = { "one", "two", "three", "four" };  
  
  /\*\* Called when the activity is first created. \*/  
  **public void** onCreate(Bundle savedInstanceState) {  
    **super**.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
  
    // открываем подключение к БД  
    db = **new** DB(**this**);  
    db.open();  
    cursor = db.getAllData();  
    startManagingCursor(cursor);  
  }  
  
  **public void** onclick(View v) {  
    changeCount();  
    **switch** (v.getId()) {  
    **case** R.id.btnItems:  
      showDialog(DIALOG\_ITEMS);  
      **break**;  
    **case** R.id.btnAdapter:  
      showDialog(DIALOG\_ADAPTER);  
      **break**;  
    **case** R.id.btnCursor:  
      showDialog(DIALOG\_CURSOR);  
      **break**;  
    **default**:  
      **break**;  
    }  
  }  
  
  **protected** Dialog onCreateDialog(**int** id) {  
    AlertDialog.Builder adb = **new** AlertDialog.Builder(**this**);  
    **switch** (id) {  
    // массив  
    **case** DIALOG\_ITEMS:  
      adb.setTitle(R.string.items);  
      adb.setItems(data, myClickListener);  
      **break**;  
    // адаптер  
    **case** DIALOG\_ADAPTER:  
      adb.setTitle(R.string.adapter);  
      ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(this,  
          android.R.layout.select\_dialog\_item, data);  
      adb.setAdapter(adapter, myClickListener);  
      **break**;  
    // курсор  
    **case** DIALOG\_CURSOR:  
      adb.setTitle(R.string.cursor);  
      adb.setCursor(cursor, myClickListener, DB.COLUMN\_TXT);  
      **break**;  
    }  
    **return** adb.create();  
  }  
  
  **protected void** onPrepareDialog(**int** id, Dialog dialog) {  
    // получаем доступ к адаптеру списка диалога  
    AlertDialog aDialog = (AlertDialog) dialog;  
    ListAdapter lAdapter = aDialog.getListView().getAdapter();  
   
    **switch** (id) {  
    **case** DIALOG\_ITEMS:  
    **case** DIALOG\_ADAPTER:  
      // проверка возможности преобразования  
      **if** (lAdapter **instanceof** BaseAdapter) {  
        // преобразование и вызов метода-уведомления о новых данных  
        BaseAdapter bAdapter = (BaseAdapter) lAdapter;  
        bAdapter.notifyDataSetChanged();  
      }  
      **break**;  
    **case** DIALOG\_CURSOR:  
      **break**;  
    **default**:  
      **break**;  
    }  
  };  
  
  // обработчик нажатия на пункт списка диалога  
  OnClickListener myClickListener = **new** OnClickListener() {  
    **public void** onClick(DialogInterface dialog, **int** which) {  
      // выводим в лог позицию нажатого элемента  
      Log.d(LOG\_TAG, "which = " + which);  
    }  
  };  
  
  // меняем значение счетчика   
  **void** changeCount() {  
    cnt++;  
    // обновляем массив  
    data[3] = String.valueOf(cnt);  
    // обновляем БД  
    db.changeRec(4, String.valueOf(cnt));  
    cursor.requery();  
  }  
  
  @Override  
  **protected void** onDestroy() {  
    **super**.onDestroy();  
    db.close();  
  }  
}

**data** – массив из 4 элементов. В таблице из **DB.java** у нас тоже 4 записи. Мы решили, что будем менять последний (четвертый) элемент/запись в данных и помещать туда кол-во показов.

В **onCreate** подключаемся к базе.

**onclick** – меняем значение счетчика и корректируем данные массива и БД,  и в зависимости от нажатой кнопки вызываем соответствующий диалог.

**onCreateDialog** – создаем вызываемый диалог, используя AlertDialog.Builder. Диалог может построить**список**, используя один из следующих объектов:

1) **Массив**строк. Используется метод [**setItems**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setItems(int, android.content.DialogInterface.OnClickListener)). На вход подается массив и обработчик нажатия.

2) **Адаптер**. Мы создаем ArrayAdapter, используя массив **data**и стандартный **layout select\_dialog\_item**, и передаем его в метод [**setAdapter**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setAdapter(android.widget.ListAdapter, android.content.DialogInterface.OnClickListener)). Также передаем туда и обработчик.

3) **Курсор**БД. Вызываем метод [**setCursor**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setCursor(android.database.Cursor, android.content.DialogInterface.OnClickListener, java.lang.String)). Передаем туда курсор, обработчик нажатия и имя поля, значение которого будет показано в списке.

Кроме списка указываем только заголовок. Кнопки и иконку не добавляем. В конце создаем и возвращаем Dialog.

Метод создания диалога (onCreateDialog) выполняется один раз, чтобы создать диалог. При последующих показах диалога выполняется метод **onPrepareDialog**. В нем мы будем обновлять данные списка. С помощью преобразований и методов [**getListView**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.html#getListView()) и [**getAdapter**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/widget/ListView.html#getAdapter()) получим список из диалога, а потом адаптер из списка.

Далее для диалогов, использующих массив и адаптер мы выполняем преобразование до BaseAdapter, чтобы иметь возможность вызвать метод [**notifyDataSetChanged**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/widget/BaseAdapter.html#notifyDataSetChanged()). Этот метод обновит список в соответствии с новыми данными.

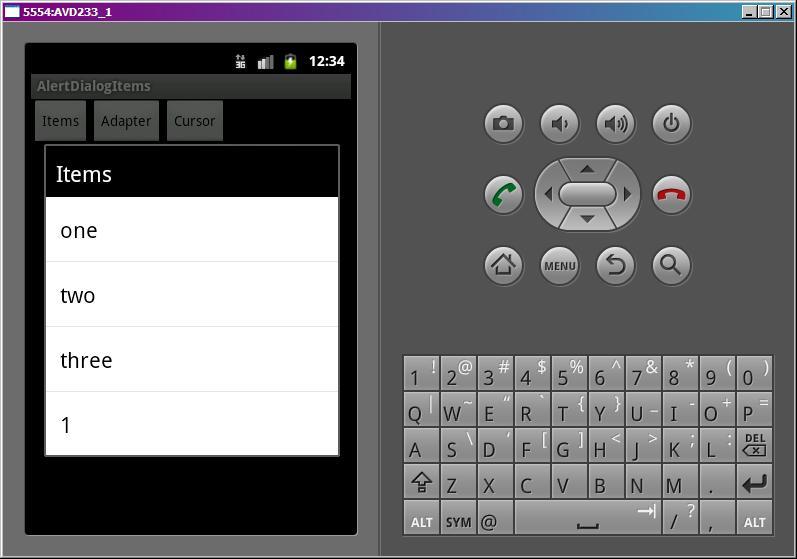
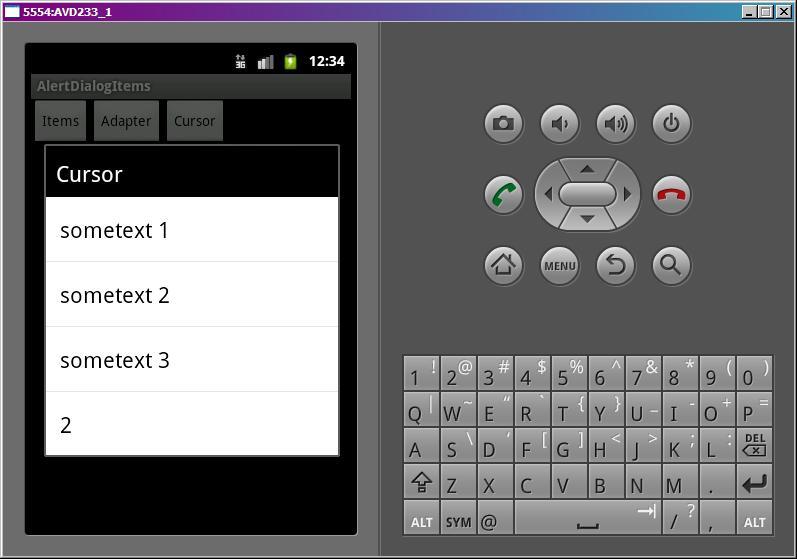
Для диалога с курсором нет необходимости уведомлять адаптер о новых данных. Курсор это делает сам.

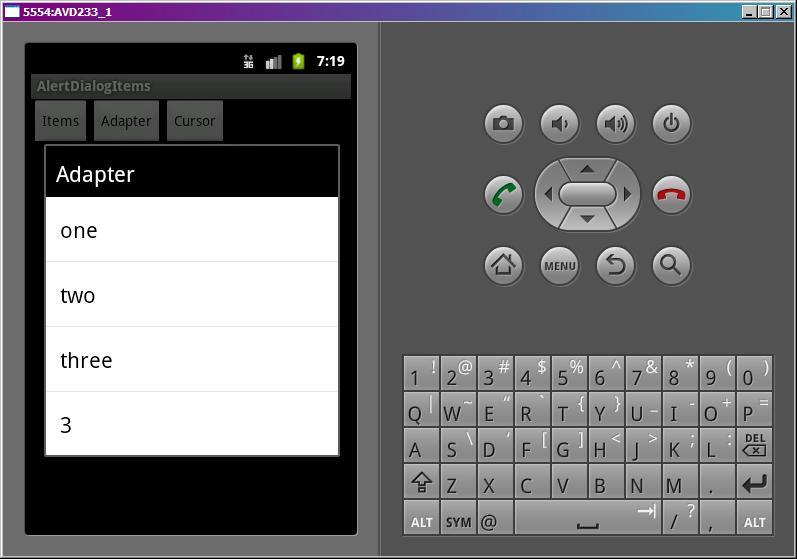
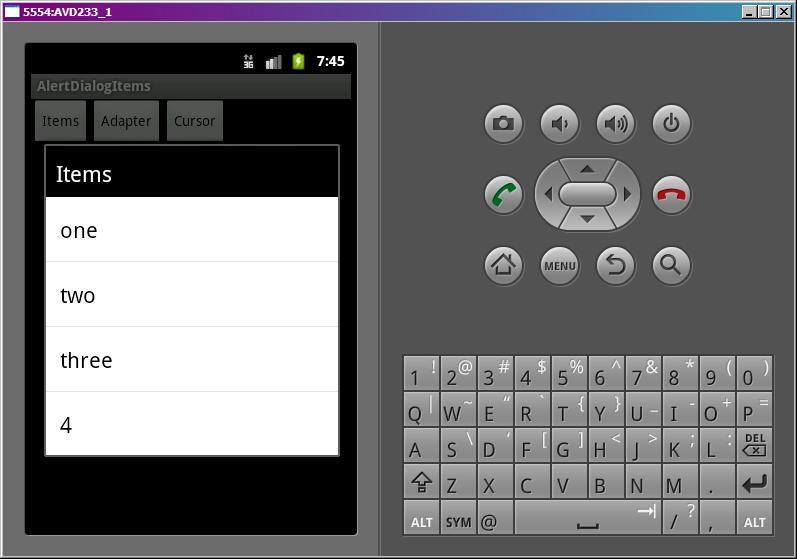
**myClickListener** – обработчик нажатия на пункты списка. Он у нас общий для всех  диалогов и просто выводит в лог позицию нажатого пункта.

В **changeCount** увеличиваем счетчик на единицу и пишем это значение в четвертый элемент массива (нумерация с нуля) и в строку с \_id = 4 в БД. Обновляем курсор.

В **onDestroy** закрываем подключение к БД.

Все сохраним и запустим. Повызываем диалоги и убедимся, что данные в них обновляются при каждом показе.

Клик на пункте списка задействует обработчик, выведет в лог его позицию и закроет диалог.

Если бы мы не реализовали метод **onPrepareDialog**, то списки диалогов (кроме курсорного) не обновлялись и выводили в последней строке те значения счетчика, при которых они создавались в методе **onCreateDialog**. Попробуйте закоментить содержимое onPrepareDialog и убедиться в этом.

UPD от 11.07.2012. Протестил этот урок на Android 3.2 - данные обновляются из без onPrepareDialog.

**Урок 63. Диалоги. AlertDialog. Список с одиночным выбором**

В этом уроке:

- формируем список с одиночным выбором в диалоге

Урок будет аналогичен прошлому, только теперь мы не будем обновлять данные списка диалога.

Мы будем формировать в диалоге список с одиночным выбором и определять какой элемент был выбран (чекнут).

Создадим проект:

**Project name**: P0631\_AlertDialogItemsSingle  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogItemsSingle  
**Package name**: ru.startandroid.develop.p0631alertdialogitemssingle  
**Create Activity**: MainActivity

В **strings.xml** пропишем тексты:

<?xml version=*"1.0"* encoding=*"utf-8"*?>  
<resources>  
<string name=*"app\_name"*>AlertDialogItemsSingle</string>  
<string name=*"items"*>Items</string>  
<string name=*"adapter"*>Adapter</string>  
<string name=*"cursor"*>Cursor</string>  
<string name=*"ok"*>OK</string>  
</resources>

**main.xml**, три кнопки:

<?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=*"horizontal"*>  
<Button  
 android:id=*"@+id/btnItems"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/items"*  
 android:onClick=*"onclick"*>  
</Button>  
<Button  
 android:id=*"@+id/btnAdapter"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/adapter"*  
 android:onClick=*"onclick"*>  
</Button>  
<Button  
 android:id=*"@+id/btnCursor"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/cursor"*  
 android:onClick=*"onclick"*>  
</Button>  
</LinearLayout>

**DB.java** аналогичен с прошлого урока, только уберем метод изменения записей

**package** ru.startandroid.develop.p0631alertdialogitemssingle;  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteDatabase.CursorFactory;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**public class** DB {  
    
  **private static final** String DB\_NAME = "mydb";  
  **private static final int** DB\_VERSION = 1;  
  **private static final** String DB\_TABLE = "mytab";  
    
  **public static final** String COLUMN\_ID = "\_id";  
  **public static final** String COLUMN\_TXT = "txt";  
    
  **private static final** String DB\_CREATE =   
    "create table " + DB\_TABLE + "(" +  
      COLUMN\_ID + " integer primary key, " +  
      COLUMN\_TXT + " text" +  
    ");";  
    
  **private final** Context mCtx;  
    
    
  **private** DBHelper mDBHelper;  
  **private** SQLiteDatabase mDB;  
    
  **public** DB(Context ctx) {  
    mCtx = ctx;  
  }  
    
  // открыть подключение  
  **public void** open() {  
    mDBHelper = **new** DBHelper(mCtx, DB\_NAME, null, DB\_VERSION);  
    mDB = mDBHelper.getWritableDatabase();  
  }  
    
  // закрыть подключение  
  **public void** close() {  
    **if** (mDBHelper!=**null**) mDBHelper.close();  
  }  
    
  // получить все данные из таблицы DB\_TABLE  
  **public** Cursor getAllData() {  
    **return** mDB.query(DB\_TABLE, null, null, null, null, null, **null**);  
  }  
    
  // класс по созданию и управлению БД  
  **private class** DBHelper **extends** SQLiteOpenHelper {  
  
    **public** DBHelper(Context context, String name, CursorFactory factory,  
        **int** version) {  
      **super**(context, name, factory, version);  
    }  
  
    // создаем и заполняем БД  
    @Override  
    **public void** onCreate(SQLiteDatabase db) {  
      db.execSQL(DB\_CREATE);  
        
      ContentValues cv = **new** ContentValues();  
      **for** (**int** i = 1; i < 5; i++) {  
        cv.put(COLUMN\_ID, i);  
        cv.put(COLUMN\_TXT, "sometext " + i);  
        db.insert(DB\_TABLE, null, cv);  
      }  
    }  
  
    @Override  
    **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
    }  
  }  
}

**MainActivity.java** немного изменился:

**package** ru.startandroid.develop.p0631alertdialogitemssingle;  
  
**import** android.app.Activity;  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.content.DialogInterface;  
**import** android.content.DialogInterface.OnClickListener;  
**import** android.database.Cursor;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.view.View;  
**import** android.widget.ArrayAdapter;  
**import** android.widget.ListView;  
  
**public class** MainActivity **extends** Activity {  
  
  **final** String LOG\_TAG = "myLogs";  
  
  **final int** DIALOG\_ITEMS = 1;  
  **final int** DIALOG\_ADAPTER = 2;  
  **final int** DIALOG\_CURSOR = 3;  
  DB db;  
  Cursor cursor;  
  
  String data[] = { "one", "two", "three", "four" };  
  
  /\*\* Called when the activity is first created. \*/  
  **public void** onCreate(Bundle savedInstanceState) {  
    **super**.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
  
    // открываем подключение к БД  
    db = **new** DB(**this**);  
    db.open();  
    cursor = db.getAllData();  
    startManagingCursor(cursor);  
  }  
  
  **public void** onclick(View v) {  
    **switch** (v.getId()) {  
    **case** R.id.btnItems:  
      showDialog(DIALOG\_ITEMS);  
      **break**;  
    **case** R.id.btnAdapter:  
      showDialog(DIALOG\_ADAPTER);  
      **break**;  
    **case** R.id.btnCursor:  
      showDialog(DIALOG\_CURSOR);  
      **break**;  
    **default**:  
      **break**;  
    }  
  }  
  
  **protected** Dialog onCreateDialog(**int** id) {  
    AlertDialog.Builder adb = **new** AlertDialog.Builder(**this**);  
    **switch** (id) {  
    // массив  
    **case** DIALOG\_ITEMS:  
      adb.setTitle(R.string.items);  
      adb.setSingleChoiceItems(data, -1, myClickListener);  
      **break**;  
    // адаптер  
    **case** DIALOG\_ADAPTER:  
      adb.setTitle(R.string.adapter);  
      ArrayAdapter<String> adapter = **new** ArrayAdapter<String>(this,  
          android.R.layout.select\_dialog\_singlechoice, data);  
      adb.setSingleChoiceItems(adapter, -1, myClickListener);  
      **break**;  
    // курсор  
    **case** DIALOG\_CURSOR:  
      adb.setTitle(R.string.cursor);  
      adb.setSingleChoiceItems(cursor, -1, DB.COLUMN\_TXT, myClickListener);  
      **break**;  
    }  
    adb.setPositiveButton(R.string.ok, myClickListener);  
    **return** adb.create();  
  }  
  
  // обработчик нажатия на пункт списка диалога или кнопку  
  OnClickListener myClickListener = **new** OnClickListener() {  
    **public void** onClick(DialogInterface dialog, **int** which) {  
      ListView lv = ((AlertDialog) dialog).getListView();  
      **if** (which == Dialog.BUTTON\_POSITIVE)  
        // выводим в лог позицию выбранного элемента  
        Log.d(LOG\_TAG, "pos = " + lv.getCheckedItemPosition());  
      **else** // выводим в лог позицию нажатого элемента  
        Log.d(LOG\_TAG, "which = " + which);  
    }  
  };  
  
  @Override  
  **protected void** onDestroy() {  
    **super**.onDestroy();  
    db.close();  
  }  
  
}

В **onCreate**подключаемся к базе.

**onclick**– в зависимости от нажатой кнопки вызываем соответствующий диалог.

**onCreateDialog**– создаем вызываемый диалог, используя AlertDialog.Builder. Диалог может построить список, используя один из следующих объектов:

1) **Массив**строк. Используется метод [**setSingleChoiceItems**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setSingleChoiceItems(java.lang.CharSequence[], int, android.content.DialogInterface.OnClickListener)). На вход подается массив, позиция выбранного элемента и обработчик нажатия.  Если мы в значение выбранного элемента передаем -1, то в списке не будет выбранного элемента.

2) **Адаптер**. Мы создаем ArrayAdapter, используя массив data и стандартный layout select\_dialog\_singlechoice, и передаем его в метод [**setSingleChoiceItems**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setSingleChoiceItems(android.widget.ListAdapter, int, android.content.DialogInterface.OnClickListener)). Также передаем туда позицию выбранного элемента и обработчик.

3) **Курсор**БД. Вызываем метод [**setSingleChoiceItems**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setSingleChoiceItems(android.database.Cursor, int, java.lang.String, android.content.DialogInterface.OnClickListener)). Передаем туда курсор, позицию выбранного элемента, имя поля (значение которого будет показано в списке) и обработчик нажатия.

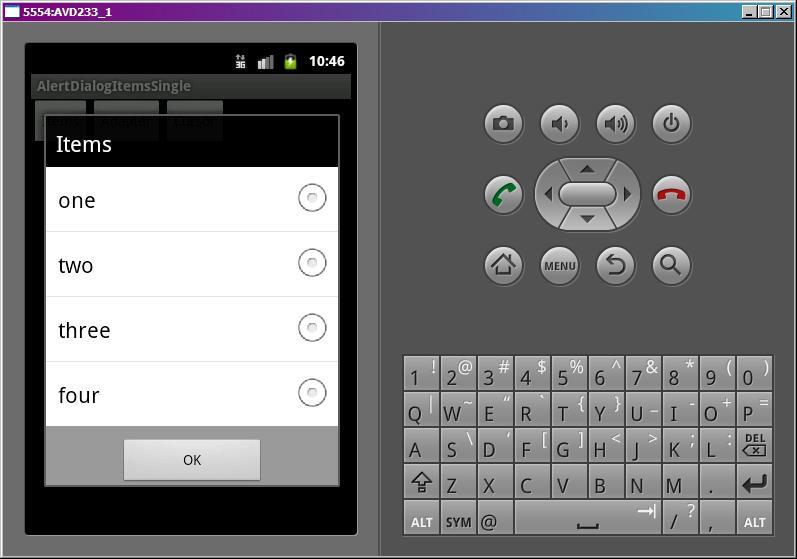
Кроме списка указываем только заголовок. В конце добавим кнопку ОК, создаем и возвращаем Dialog.

Как видим для всех трех способов создания используются методы с одинаковым названием setSingleChoiceItems, но с разными аргументами на вход.

**myClickListener**– обработчик нажатия на пункты списка и на кнопку. Если было нажатие на кнопку (а значит - закрытие диалога), то в лог выводим информацию о выбранном элементе. Иначе (нажатие на пункт списка) – выводим позицию нажатого элемента. Если диалог содержит список с **одиночным**или**множественным**выбором, то нажатие на пункт списка **не приводит к закрытию**диалога. Но это можно без проблем реализовать программно.

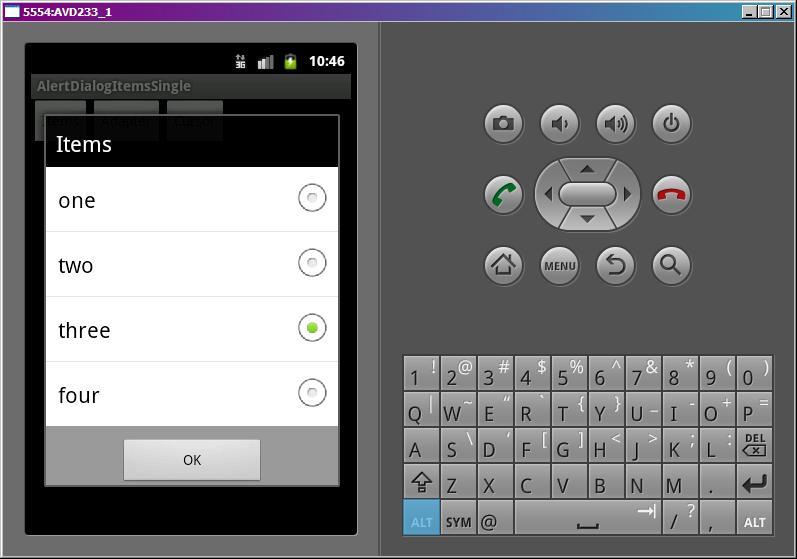
В **onDestroy** закрываем подключение к БД.

Все сохраним и запустим. Вызовем какой-нибудь диалог.



Ничего не выбрано, т.к. мы указали -1 в методе setSingleChoiceItems.

Нажмем на пункт, он выделился



А в логе появился текст

*which = 2*

Если нажмем на ОК, то диалог закроется. А лог отобразит позицию выбранного элемента.

*pos = 2*

Если теперь снова откроете диалог, то выбранный пункт так и останется выбранным. Т.к. диалог не уничтожается при закрытии. Если хотите перед каждым вызовом менять выбранный элемент, реализуйте метод **onPrepareDialog**. В нем надо добраться до списка и вызвать метод **setItemChecked**.

Пример кода, в котором выбирается третий элемент (нумерация с нуля):

**protected void** onPrepareDialog(**int** id, Dialog dialog) {  
    ((AlertDialog) dialog).getListView().setItemChecked(2, **true**);  
  };

**Урок 64. Диалоги. AlertDialog. Список с множественным выбором**

В этом уроке:

- формируем список с множественным выбором в диалоге

Урок будет аналогичен прошлому. Мы будем формировать в диалоге список с множественным выбором и определять какие элементы были выбраны (чекнуты). На этот раз мы сможем работать только с массивом и курсором. Дать диалогу адаптер не получится, нет соответствующего метода. Это особенности реализации.

Создадим проект:

**Project name**: P0641\_AlertDialogItemsMulti  
**Build Target**: Android 2.3.3  
**Application name**: AlertDialogItemsMulti  
**Package name**: ru.startandroid.develop.p0641alertdialogitemsmulti  
**Create Activity**: MainActivity

В **strings.xml** пропишем тексты:

<?xml version=*"1.0"* encoding=*"utf-8"*?>  
<resources>  
<string name=*"app\_name"*>AlertDialogItemsMulti</string>  
<string name=*"items"*>Items</string>  
<string name=*"cursor"*>Cursor</string>  
<string name=*"ok"*>OK</string>  
</resources>

**main.xml**:

<?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=*"horizontal"*>  
<Button  
 android:id=*"@+id/btnItems"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/items"*  
 android:onClick=*"onclick"*>  
</Button>  
<Button  
 android:id=*"@+id/btnCursor"*  
 android:layout\_width=*"wrap\_content"*  
 android:layout\_height=*"wrap\_content"*  
 android:text=*"@string/cursor"*  
 android:onClick=*"onclick"*>  
</Button>  
</LinearLayout>

**DB.java**:

**package** ru.startandroid.develop.p0641alertdialogitemsmulti;  
  
**import** android.content.ContentValues;  
**import** android.content.Context;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.database.sqlite.SQLiteDatabase.CursorFactory;  
**import** android.database.sqlite.SQLiteOpenHelper;  
  
**public class** DB {  
    
  **private static final** String DB\_NAME = "mydb";  
  **private static final int** DB\_VERSION = 1;  
  **private static final** String DB\_TABLE = "mytab";  
    
  **public static final** String COLUMN\_ID = "\_id";  
  **public static final** String COLUMN\_CHK = "checked";  
  **public static final** String COLUMN\_TXT = "txt";  
    
  **private static final** String DB\_CREATE =   
    "create table " + DB\_TABLE + "(" +  
      COLUMN\_ID + " integer primary key, " +  
      COLUMN\_CHK + " integer, " +  
      COLUMN\_TXT + " text" +  
    ");";  
    
  **private final** Context mCtx;  
    
    
  **private** DBHelper mDBHelper;  
  **private** SQLiteDatabase mDB;  
    
  **public** DB(Context ctx) {  
    mCtx = ctx;  
  }  
    
  // открыть подключение  
  **public void** open() {  
    mDBHelper = **new** DBHelper(mCtx, DB\_NAME, null, DB\_VERSION);  
    mDB = mDBHelper.getWritableDatabase();  
  }  
    
  // закрыть подключение  
  **public void** close() {  
    **if** (mDBHelper!=**null**) mDBHelper.close();  
  }  
    
  // получить все данные из таблицы DB\_TABLE  
  **public** Cursor getAllData() {  
    **return** mDB.query(DB\_TABLE, null, null, null, null, null, **null**);  
  }  
    
  //изменить запись в DB\_TABLE  
  **public void** changeRec(**int** pos, **boolean** isChecked) {  
      ContentValues cv = **new** ContentValues();  
      cv.put(COLUMN\_CHK, (isChecked) ? 1 : 0);  
      mDB.update(DB\_TABLE, cv, COLUMN\_ID + " = " + (pos + 1), **null**);  
  }  
    
  // класс по созданию и управлению БД  
  **private class** DBHelper **extends** SQLiteOpenHelper {  
  
    **public** DBHelper(Context context, String name, CursorFactory factory,  
        **int** version) {  
      **super**(context, name, factory, version);  
    }  
  
    // создаем и заполняем БД  
    @Override  
    **public void** onCreate(SQLiteDatabase db) {  
      db.execSQL(DB\_CREATE);  
        
      ContentValues cv = **new** ContentValues();  
      **for** (**int** i = 1; i < 5; i++) {  
      cv.put(COLUMN\_ID, i);  
      cv.put(COLUMN\_TXT, "sometext " + i);  
      cv.put(COLUMN\_CHK, 0);  
      db.insert(DB\_TABLE, null, cv);  
      }  
    }  
  
    @Override  
    **public void** onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion) {  
    }  
  }  
}

Мы в таблице создаем числовое поле **checked**. По этому полю список диалога будет определять выделен элемент (значение = 1) или нет (0). Метод **changeRec**берет на вход позицию элемента в списке и boolean-значение, выделен элемент или нет, и меняет соответствующую запись в таблице.

**MainActivity.java**:

**package** ru.startandroid.develop.p0641alertdialogitemsmulti;  
  
**import** android.app.Activity;  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.content.DialogInterface;  
**import** android.content.DialogInterface.OnClickListener;  
**import** android.content.DialogInterface.OnMultiChoiceClickListener;  
**import** android.database.Cursor;  
**import** android.os.Bundle;  
**import** android.util.Log;  
**import** android.util.SparseBooleanArray;  
**import** android.view.View;  
**import** android.widget.CursorAdapter;  
**import** android.widget.ListView;  
  
**public class** MainActivity **extends** Activity {  
  
  **final** String LOG\_TAG = "myLogs";  
    
  **final int** DIALOG\_ITEMS = 1;  
  **final int** DIALOG\_CURSOR = 3;  
  DB db;  
  Cursor cursor;  
  
  String data[] = { "one", "two", "three", "four" };  
  **boolean** chkd[] = { false, true, true, **false** };  
  
  /\*\* Called when the activity is first created. \*/  
  **public void** onCreate(Bundle savedInstanceState) {  
    **super**.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
  
    // открываем подключение к БД  
    db = **new** DB(**this**);  
    db.open();  
    stopManagingCursor(cursor);  
    cursor = db.getAllData();  
  }  
  
  **public void** onclick(View v) {  
    **switch** (v.getId()) {  
    **case** R.id.btnItems:  
      showDialog(DIALOG\_ITEMS);  
      **break**;  
    **case** R.id.btnCursor:  
      showDialog(DIALOG\_CURSOR);  
      **break**;  
    **default**:  
      **break**;  
    }  
  }  
  
  **protected** Dialog onCreateDialog(**int** id) {  
    AlertDialog.Builder adb = **new** AlertDialog.Builder(**this**);  
    **switch** (id) {  
    // массив  
    **case** DIALOG\_ITEMS:  
      adb.setTitle(R.string.items);  
      adb.setMultiChoiceItems(data, chkd, myItemsMultiClickListener);  
      **break**;  
    // курсор  
    **case** DIALOG\_CURSOR:  
      adb.setTitle(R.string.cursor);  
      adb.setMultiChoiceItems(cursor, DB.COLUMN\_CHK, DB.COLUMN\_TXT, myCursorMultiClickListener);  
      **break**;  
    }  
    adb.setPositiveButton(R.string.ok, myBtnClickListener);  
    **return** adb.create();  
  }  
  
  // обработчик для списка массива  
  OnMultiChoiceClickListener myItemsMultiClickListener = **new** OnMultiChoiceClickListener() {  
    @Override  
    **public void** onClick(DialogInterface dialog, **int** which, **boolean** isChecked) {  
      Log.d(LOG\_TAG, "which = " + which + ", isChecked = " + isChecked);  
    }  
  };    
    
  // обработчик для списка курсора  
  OnMultiChoiceClickListener myCursorMultiClickListener = **new** OnMultiChoiceClickListener() {  
    @Override  
    **public void** onClick(DialogInterface dialog, **int** which, **boolean** isChecked) {  
      ListView lv = ((AlertDialog) dialog).getListView();  
      Log.d(LOG\_TAG, "which = " + which + ", isChecked = " + isChecked);  
      db.changeRec(which, isChecked);  
      cursor.requery();  
    }  
  };  
    
  // обработчик нажатия на кнопку  
  OnClickListener myBtnClickListener = **new** OnClickListener() {  
    **public void** onClick(DialogInterface dialog, **int** which) {  
        SparseBooleanArray sbArray = ((AlertDialog)dialog).getListView().getCheckedItemPositions();  
        **for** (**int** i = 0; i < sbArray.size(); i++) {  
          **int** key = sbArray.keyAt(i);  
          **if** (sbArray.get(key))  
            Log.d("qwe", "checked: " + key);  
        }      
    }  
  };  
  
  **protected void** onDestroy() {  
    **super**.onDestroy();  
    db.close();  
  }  
  
}

Разбираем код. Кроме массива строк **data**, нам нужен массив boolean[] **chkd**, который укажет диалогу, какие элементы списка надо будет сразу сделать выделенными. Будем выделять второй и третий.

В **onCreate** подключаемся к базе.

**onclick** – в зависимости от нажатой кнопки вызываем соответствующий диалог.

**onCreateDialog** – создаем вызываемый диалог, используя AlertDialog.Builder. Диалог может построить список, используя один из следующих объектов:

1) **Массив**строк. Используется метод [**setMultiChoiceItems**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setMultiChoiceItems(java.lang.CharSequence[], boolean[], android.content.DialogInterface.OnMultiChoiceClickListener)). На вход подается массив строк, boolean-массив, определяющий выделенные элементов, и обработчик нажатия.

2) **Курсор**БД. Вызываем метод [**setMultiChoiceItems**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.Builder.html#setMultiChoiceItems(android.database.Cursor, java.lang.String, java.lang.String, android.content.DialogInterface.OnMultiChoiceClickListener)). Передаем туда курсор, имя поля выделения (данные о выделении элементов списка), имя поля с текстом (текст, который будет отображен в списке) и обработчик нажатия.

Кроме списка указываем только заголовок. В конце добавим кнопку ОК, создаем и возвращаем Dialog.

Для обоих способов создания используется методы с одинаковым названием setMultiChoiceItems, но с разными аргументами на вход.

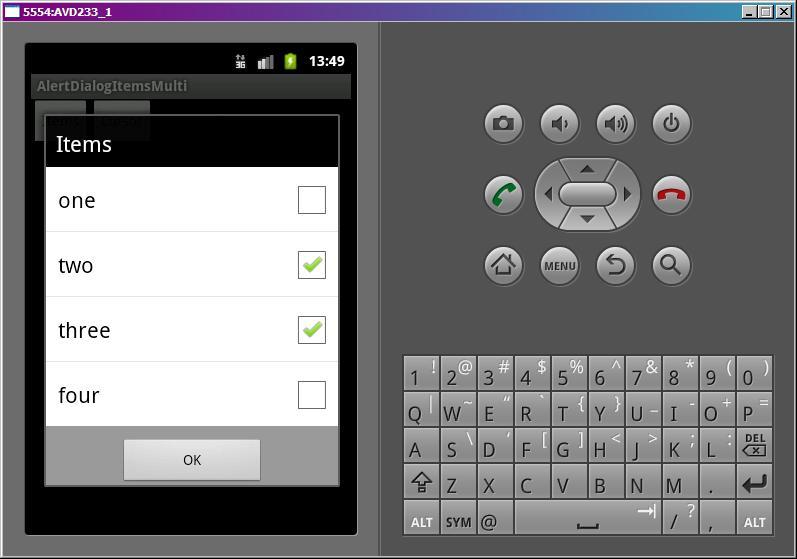
**myItemsMultiClickListener** – обработчик нажатий на список, построенный из массива. Выводит в лог какой элемент был нажат и стал он выделенным или не выделенным. Реализует интерфейс [**OnMultiChoiceClickListener**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.OnMultiChoiceClickListener.html).

**myCursorMultiClickListener**- обработчик нажатий на список, построенный из курсора. Выводит в лог какой элемент был нажат и стал он выделенным или не выделенным. Также он соответствующим образом меняет данные в БД и обновляет курсор списка диалога. Т.к. если какой-то элемент выделили, мы должны в обработчике сбегать в БД, обновить соответствующую запись (поле checked) и обновить курсор. Ради интереса закомментируйте код этого обработчика – вы увидите, что галки в списке просто не ставятся.

**myBtnClickListener**– обработчик нажатия на кнопку. Получает из списка информацию о выделенных элементах и выводит ее в лог.

В **onDestroy**закрываем подключение к БД.

Все сохраним и запустим. Открываем диалог Items.

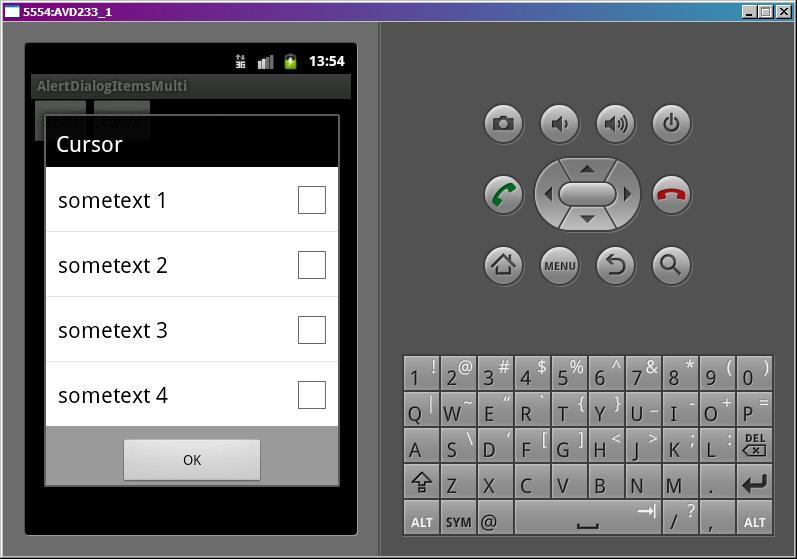


Видим, что галки проставились так, как мы указывали в массиве chkd. Если понажимать на пункты списка, лог показывает, какие изменения происходят.

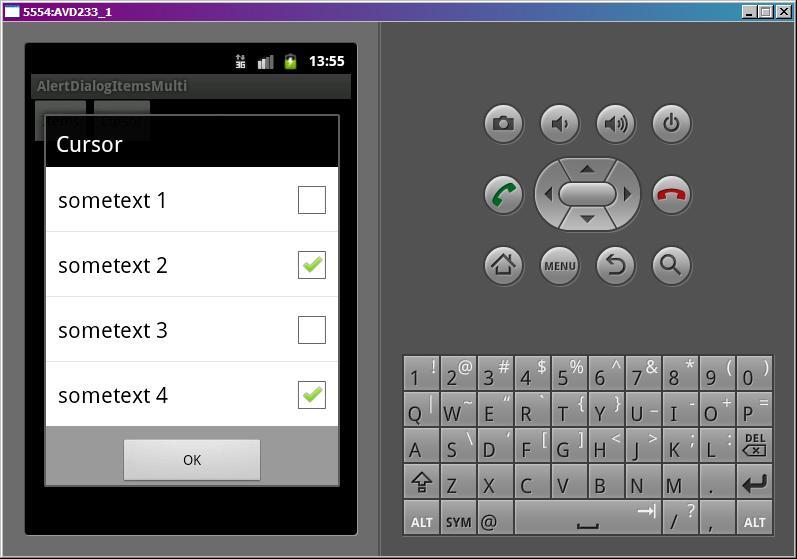
*which = 1, isChecked = false  
which = 0, isChecked = true  
which = 3, isChecked = true*

Если есть необходимость, можно добавить в обработчик код, который обновлял бы массив chkd в соответствии с нажатиями. Индекс элемента и значение у нас есть.

Откроем диалог Cursor. Здесь ничего не выделено, т.к. в поле checked мы поместили нули.



Проставим галки,



лог это отобразит

*which = 1, isChecked = true  
which = 3, isChecked = true*

Это работает только потому, что мы вручную обновляем БД и курсор. Повторюсь, попробуйте закоментить код обработчика myCursorMultiClickListener и элементы перестанут выделяться.

Я не стал реализовывать метод **onPrepareDialog**. В случае Items, там все просто, аналогично прошлому уроку используйте такой код:

*((AlertDialog)dialog).getListView().setItemChecked(2,****true****);*

В этом примере: 2 – это позиция элемента списка, а true - означает, что галка должна стоять. Если хотите снять галку, передавайте false.

В случае с курсором код будет аналогичен коду из обработчика **myCursorMultiClickListener**. Меняете запись в БД методом **db.changeRec**, обновляете курсор и передаете его адаптеру.

Нетривиальная такая получилась штука, не знаю пригодится кому или нет, но раз уж взялся за диалоги - решил расписать и это.

Также наверно имеет смысл сказать, что у методов setItems, setSingleChoiceItems, setMultiChoiceItems есть также реализация, использующая не напрямую массив, а ID массива строк из файла ресурсов.

**Урок 65. Диалоги. AlertDialog. Кастомизация**

В этом уроке:

- используем свои View для построения диалога

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.

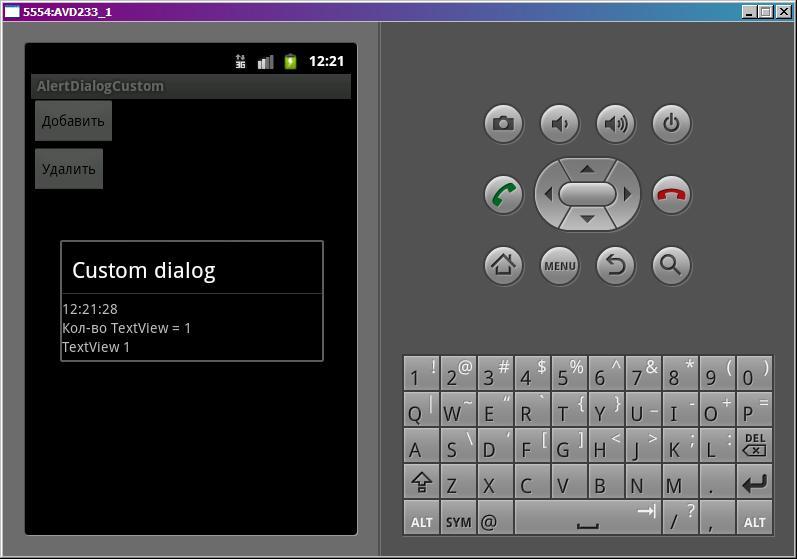
Метод **onclick** – обработчик нажатий на кнопки, сохраняет в btn идентификатор нажатой кнопки и показывает диалог.

В методе создания диалога **onCreateDialog**мы задаем текст заголовка диалога, создаем **view** из layout-файла dialog.xml и с помощью метода [**setView**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.html#setView(android.view.View)) 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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Dialog.html#getWindow()) 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 им 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.

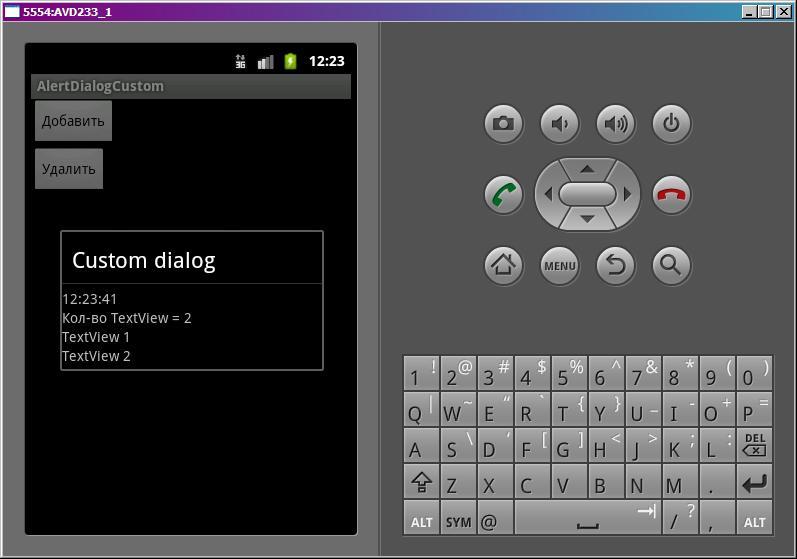
Обратите внимание, что я использую два разных способа для получения доступа к tvCount и tvTime. tvCount я нашел сразу после создания **view**в методе onCreateDialog. А в случае с tvTime я показываю, как найти View-компонент в диалоге без использования объекта **view**. Какой вам удобнее по ситуации,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

Все сохраним и запустим приложение. Нажмем кнопку Добавить



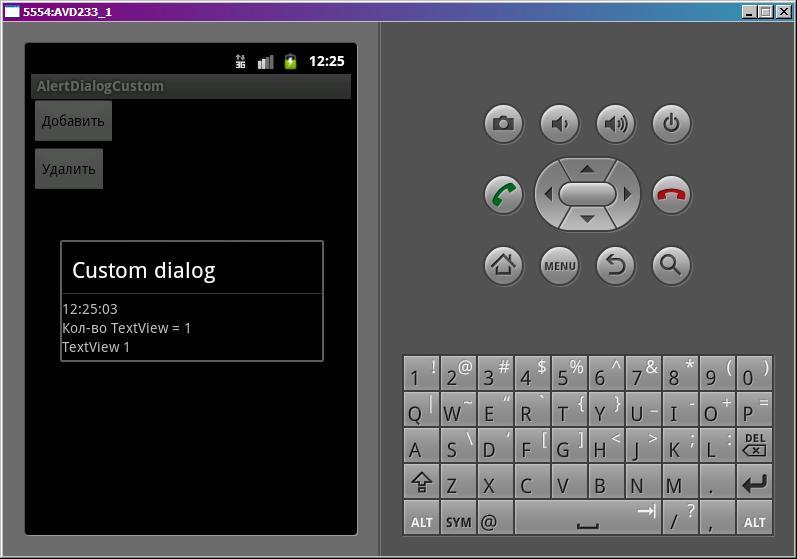
Появился диалог. Отображает время, кол-во добавленных TextView и собственно сами добавленные TextView.

Закрываем диалог кнопкой Назад, жмем еще раз добавить



Добавилось еще одно TextView.

Закрываем диалог, жмем кнопку Удалить.



Последнее TextView удалилось.

**Урок 66. Диалоги. Обработчики и операции**

В этом уроке:

- рассматриваем обработчики событий диалога  
- программно закрываем и показываем диалог

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:

<?xml version=*"1.0"* encoding=*"utf-8"*?>  
<resources>  
<string name=*"dialog"*>Диалог</string>  
<string name=*"app\_name"*>P0661\_AlertDialogOperations</string>  
</resources>

**main.xml**:

<?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**:

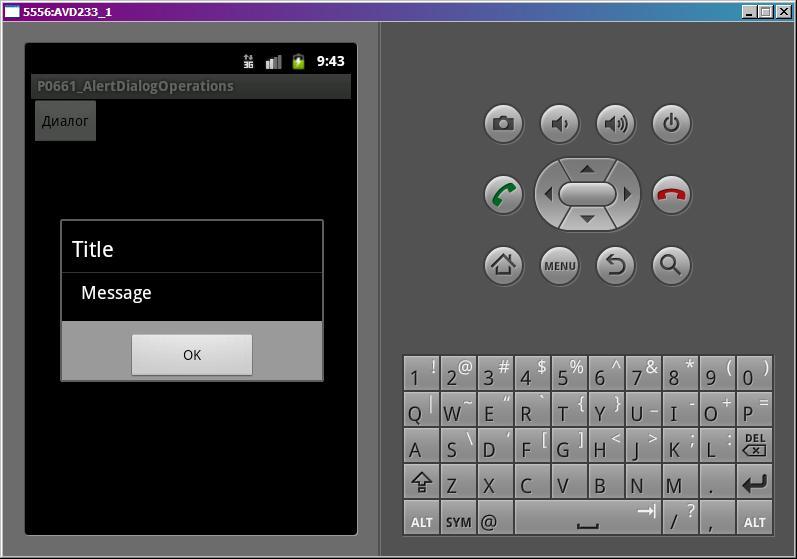
**package** ru.startandroid.develop.p0661alertdialogoperations;  
  
**import** android.app.Activity;  
**import** android.app.AlertDialog;  
**import** android.app.Dialog;  
**import** android.content.DialogInterface;  
**import** android.content.DialogInterface.OnCancelListener;  
**import** android.content.DialogInterface.OnDismissListener;  
**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**:

**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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Dialog.html#dismiss()) – он закрывает диалог. Перепишем метод method1:

**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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Dialog.html#cancel()). Перепишем method1:

**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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Dialog.html#hide()):

**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 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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Activity.html#dismissDialog(int)) и [**removeDialog**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Activity.html#removeDialog(int)). 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:

**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:

**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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/Dialog.html#show()). 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.

**Урок 67. Диалоги. ProgressDialog**

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:

<?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**:

<?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**:

**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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/AlertDialog.html#setButton(int, java.lang.CharSequence, android.os.Message)). На вход передаем тип кнопки, текст и обработчик. Тип кнопки определяется константами:[**BUTTON\_POSITIVE**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_POSITIVE), [**BUTTON\_NEGATIVE**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/content/DialogInterface.html#BUTTON_NEGATIVE), [**BUTTON\_NEUTRAL**](http://elsof.ru/go.php?url=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.

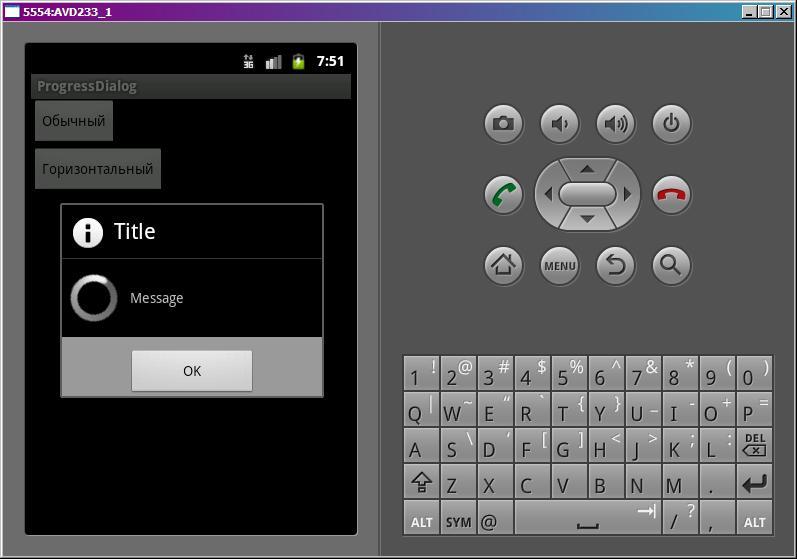
Для второго диалога указываем, что он будет индикатором. Это делается с помощью метода[**setProgressStyle**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#setProgressStyle(int)). Далее задаем заголовок и сообщение. Кнопку не добавляем, но если вдруг нужна будет, то это делается полностью  аналогично первому диалогу. Метод [**setMax**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#setMax(int)) устанавливает максимальное значение заполнения индикатора. Значение 2148 я выбрал случайно.  Метод[**setIndeterminate**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#setIndeterminate(boolean)) 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://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#incrementProgressBy(int))) и дополнительного ([**incrementSecondaryProgressBy**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#incrementSecondaryProgressBy(int))) индикатора, пока основной индикатор ([**getProgress**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#getProgress())) не достигнет максимума ([**getMax**](http://elsof.ru/go.php?url=http://developer.android.com/reference/android/app/ProgressDialog.html#getMax()), 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.

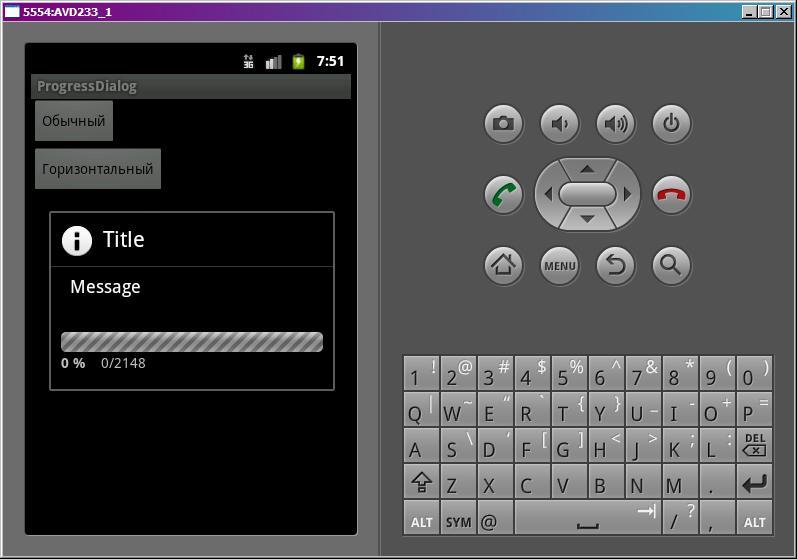
Все сохраним и запустим.

Вызовем первый диалог

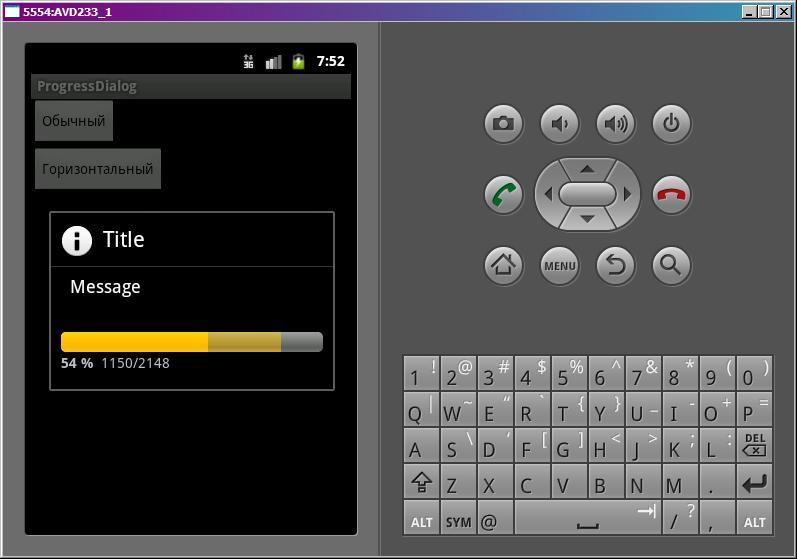


Бесконечно крутящийся круг показывает, что процесс идет. Но сколько осталось – непонятно. Не всегда можно спрогнозировать оставшееся время или показать процент выполнения задачи.

Закроме 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, будем разбираться )