**Использование DialogFragment**

*Этот пост является переводом руководства из [Android Developers Blog](http://android-developers.blogspot.ru/2012/05/using-dialogfragments.html" \t "_blank)*  
  
Начиная с **Honeycomb**в **Android**появились фрагменты (**Fragments**), которые позволили повторно использовать в приложении части его пользовательского интерфейса и логику. Методы **showDialog**и **dismissDialog**класса **Activity**были объявлены устаревшими. Им на замену пришли **DialogFragments**.  
  
В этом посте я продемострирую использование **DialogFragments**вместе с библиотекой совместимости **v4** (для обратной совместимости с устройствами, в которых установлены более ранние версии **Android**) на примере простого диалога, позволяющего вводить текст и возвращать результат вызвавшей его **Activity**с помощью интерфейса. [Здесь](http://developer.android.com/design/building-blocks/dialogs.html) вы можете посмотреть руководство по дизайну диалогов.

**Разметка (Layout)**

Ниже приведен **layout**диалога который находится в файле **fragment\_edit\_name.xml**.

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

   android:id="@+id/edit\_name"

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

   android:layout\_gravity="center" android:orientation="vertical"  **>**

**<TextView**

       android:id="@+id/lbl\_your\_name" android:text="Your name"

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

**<EditText**

       android:id="@+id/txt\_your\_name"

android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"

       android:inputType=”text”

       android:imeOptions="actionDone" **/>**

**</LinearLayout>**

Заметьте использование двух необезательных аттрибутов. Сочетание **android:inputType=”text”** и**android:imeOptions=”actionDone”** настраивает виртуальную клавиатуру на отображение кнопки "**Done**" вместо стандартной "**Enter**".

**Код диалога**

Класс диалога расширяет **DialogFragment**. Для обеспечения обратной совместимости мы будем импортировать его из библиотеки совместимости **v4**. (Для того чтобы добавить ее из Eclipse, кликните правой кнопкой на проект и выберите **Android Tools** | **Add Support Library...**).

**import** android.support.v4.app.DialogFragment;

*// ...*

**public** **class** EditNameDialog **extends** DialogFragment {

**private** EditText mEditText;

**public** EditNameDialog() {

*// Empty constructor required for DialogFragment*

    }

    @Override

**public** [View](http://www.google.com/search?hl=en&q=allinurl%3Aview+java.sun.com&btnI=I%27m%20Feeling%20Lucky) onCreateView(LayoutInflater inflater, ViewGroup container,

            Bundle savedInstanceState) {

[View](http://www.google.com/search?hl=en&q=allinurl%3Aview+java.sun.com&btnI=I%27m%20Feeling%20Lucky) view = inflater.inflate(R.layout.fragment\_edit\_name, container);

        mEditText = (EditText) view.findViewById(R.id.txt\_your\_name);

        getDialog().setTitle("Hello");

**return** view;

    }

}

Диалог расширяет **DialogFragment**и содержит обязательный пустой конструктор. Фрагменты имплементируют метод**onCreateView()** для загрузки вида с помощью предоставленного **LayoutInflater**.

**Отображение диалога**

Теперь мы должны добавить немного кода к нашей **Activity**для того чтобы отобразить диалог. Ниже приведен простой пример который показывает **EditNameDialog**, запрашивающий имя пользователя. В завершении он показывает **Toast**с введенным текстом.

**import** android.support.v4.app.FragmentActivity;

**import** android.support.v4.app.FragmentManager;

*// ...*

**public** **class** FragmentDialogDemo **extends** FragmentActivity **implements** EditNameDialogListener {

    @Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        showEditDialog();

    }

**private** **void** showEditDialog() {

        FragmentManager fm = getSupportFragmentManager();

        EditNameDialog editNameDialog = **new** EditNameDialog();

        editNameDialog.show(fm, "fragment\_edit\_name");

    }

    @Override

**public** **void** onFinishEditDialog([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) inputText) {

        Toast.makeText(**this**, "Hi, " + inputText, Toast.LENGTH\_SHORT).show();

    }

}

Следует обратить внимание на несколько вещей. Во-первых, наша **Activity**расширяет **FragmentActivity**, так как мы используем библиотеку поддержки для обратной совместимости с **Fragment API**. По этой причине мы вызываем метод**getSupportFragmentManager()** вместо **getFragmentManager()**.  
  
После загрузки вида **Activity**сразу показывает **EditNameDialog**вызывая его метод **show()**. Это позволяет **DialogFragment**согласовывать свое состояние с вызывающим его фрагментом. Нажатие кнопки "**back**" закроет диалог без использования какого-либо дополнительного кода.

**Использование диалога**

Следующим шагом давайте улучшим **EditNameDialog**, добавив возможность передавать введеную пользователем строку нашей **Activity**.

**import** android.support.v4.app.DialogFragment;

*// ...*

**public** **class** EditNameDialog **extends** DialogFragment **implements** OnEditorActionListener {

**public** **interface** EditNameDialogListener {

**void** onFinishEditDialog([String](http://www.google.com/search?hl=en&q=allinurl%3Astring+java.sun.com&btnI=I%27m%20Feeling%20Lucky) inputText);

    }

**private** EditText mEditText;

**public** EditNameDialog() {

*// Empty constructor required for DialogFragment*

    }

    @Override

**public** [View](http://www.google.com/search?hl=en&q=allinurl%3Aview+java.sun.com&btnI=I%27m%20Feeling%20Lucky) onCreateView(LayoutInflater inflater, ViewGroup container,

            Bundle savedInstanceState) {

[View](http://www.google.com/search?hl=en&q=allinurl%3Aview+java.sun.com&btnI=I%27m%20Feeling%20Lucky) view = inflater.inflate(R.layout.fragment\_edit\_name, container);

        mEditText = (EditText) view.findViewById(R.id.txt\_your\_name);

        getDialog().setTitle("Hello");

*// Show soft keyboard automatically*

        mEditText.requestFocus();

        getDialog().getWindow().setSoftInputMode(

                LayoutParams.SOFT\_INPUT\_STATE\_VISIBLE);

        mEditText.setOnEditorActionListener(**this**);

**return** view;

    }

    @Override

**public** **boolean** onEditorAction(TextView v, **int** actionId, [KeyEvent](http://www.google.com/search?hl=en&q=allinurl%3Akeyevent+java.sun.com&btnI=I%27m%20Feeling%20Lucky) event) {

**if** (EditorInfo.IME\_ACTION\_DONE == actionId) {

*// Return input text to activity*

EditNameDialogListener activity = (EditNameDialogListener) getActivity();

            activity.onFinishEditDialog(mEditText.getText().toString());

**this**.dismiss();

**return** **true**;

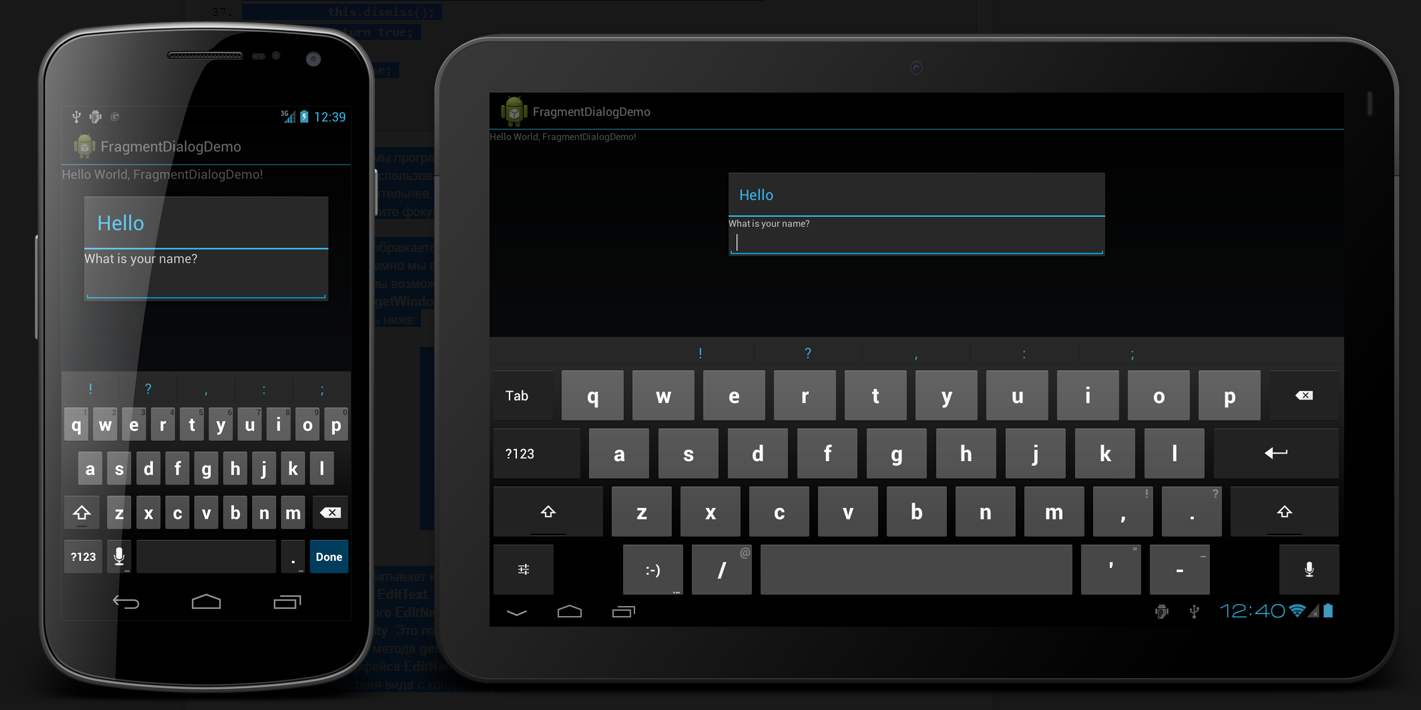
        }

**return** **false**;

    }

}

Для удобства пользователя мы программно вызываем фокус на **EditText**с помощью **mEditText.requestFocus()**. В качестве альтернативы мы могли бы использовать тег **<requestFocus/>** в XML **layout**-е. Однако в некоторых случаях запрашивать фокус программно предпочтительнее. К примеру **OnFocusChangeListener**добавленный нами в метод **onCreateView()** не будет вызван если вы запросите фокус в XML **layout**-е.  
  
Программная клавиатура отображается автоматически после того как пользователь делает фокус на **EditText**. Для того чтобы вызвать это событие программно мы вызываем **getDialog().getWindow().setSoftInputMode()**. Заметьте что многие из операций с классом **Window**которые вы возможно совершали в классе **Dialog**могут использоваться и с **DialogFragment**, но вам теперь нужно вызывать **getDialog().getWindow()** вместо **getWindow()**. То как будет выглядеть получившийся диалог на сматрфоне и планшете вы можете увидеть ниже:



Метод **onEditorAction()** срабатывает когда пользователь нажимает кнопку "**Done**". Он вызывается потому что мы указали**OnEditorActionListener**для **EditText**. Он в свою очередь вызывает метод нашей **Activity**для того чтобы послать введенный пользователем текст. Для этого **EditNameDialog**объявляет интерфейс **EditNameDialogListener**, который в свою очередь имплементирует наша **Activity**. Это позволяет использовать диалог во многих **Activity**. Мы можем получить **Activity**, которая вызвала диалог с помощью метода **getActivity()**. Теперь мы можем вызывать ее метод **onFinishEditDialog()**, предварительно закастив **Activity**до интерфейса **EditNameDialogListener**. В архитектуре **MVC**это является распространенным паттерном для обеспечения взаимодействия вида с контроллером.  
  
Мы можем закрыть диалог одним из двух путей. В этом примере мы вызываем метод **dismiss()** в самом диалоге. Он также может быть вызван из **Activity**так-же как метод **show()**.  
  
Надеюсь этот пост пролил немного света на фрагменты так как они связаны с диалогами. Вы можете найти исходный код этого поста [на Google Code](http://code.google.com/p/dialogfragmentdemo/).  
  
Ссылки для дальнейшего изучения фрагментов:

* [Fragments Dev Guide](http://developer.android.com/guide/topics/fundamentals/fragments.html)
* ["Basic Training" on Fragments](http://developer.android.com/training/basics/fragments/index.html)
* [Updating Applications for On-screen Input Methods](http://android-developers.blogspot.com/2009/04/updating-applications-for-on-screen.html)
* [OnEditorActionListener](http://developer.android.com/reference/android/widget/TextView.OnEditorActionListener.html)
* [EditorInfo and IME options](http://developer.android.com/reference/android/view/inputmethod/EditorInfo.html)