Диалог выбора на Android или контекстное меню с картинками.

Задача выбора элемента из списка возникает в Android-приложениях регулярно. В принципе, для выбора элемента можно воспользоваться [стандартным контекстным меню](http://developer.android.com/guide/topics/ui/menus.html). Но у контекстного меню есть ряд ограничений:

* В контекстном меню функции вызова (onCreateContextMenu) и обработки результатов(onContextItemSelected) реализуются в Activity.
* Нельзя выбрать несколько элементов.
* В контекстном меню показывается только название элемента. Картинки не отображаются.

Наиболее напрягает необходимость реализации функций в Activity. Типичная ситуация - вам нужна вспомогательная функция, которая предлагает пользователю что-нибудь выбрать из списка и затем выполняет действие над выбранным элементом. Такая функция может вызываться из **множества** разных Activity. И что, мне в каждой Activity реализовывать функции onCreateContextMenu иonContextItemSelected? Так не пойдет.   
  
Лучше сделать диалог. А поскольку подобные диалоги нужны сплошь и рядом, нужно сделать более-менее универсальный диалог, который подойдет для выбора элементов в большинстве случаев. О реализации такого диалога и пойдет речь.

**Постановка задачи**

Итак, задача - создать гибкий диалог выбора элементов из списка. Диалог, который идеально подходит для случаев, когда есть фиксированный список произвольных элементов (не обязательно строк) и нужно дать возможность пользователю выбрать один или несколько элементов. Диалог должен "уметь" показывать название, картинку и описание элемента, причем показ картинки и описания должен быть опциональным (просто потом, что картинка и описание требуются далеко не всегда).   
  
В сложных случаях подобного диалога, естественно, не хватит. Например, при выборе списка контактов потребуется возможность фильтрации списка, фоновая подгрузка картинок, возможность вызова меню и т.д. Но в простейших случаях его будет за глаза.   
  
Еще одним требованием к диалогу является возможность передать в него функцию обработчик полученного результата. Таким образом, появится возможность реализовывать функцию обработчик в анонимном классе прямо в точке вызова диалога.   
  
Ну и последнее требование - простота использования.

**Реализация UniDialog**

Исходные коды получившегося диалога (я его назвал UniDialog) можно [посмотреть здесь](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/). Реализация включает три файла: [UniDialogMultySelector](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/src/com/kbitubit/android/demo/UniDialogMultySelector.java) - диалог для выбора нескольких элементов, [UniDialogSingleSelector](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/src/com/kbitubit/android/demo/UniDialogSingleSelector.java) - диалог для выбора одного элемента, [UniDialogUtils](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/src/com/kbitubit/android/demo/UniDialogUtils.java) - вспомогательные классы. Кроме того, для работы диалог требует файлы ресурсов: [unidialog\_selector\_row.xml](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/res/layout/unidialog_selector_row.xml), [unidialog\_selector.xml](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/res/layout/unidialog_selector.xml),[colors.xml](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/res/values/colors.xml) и [unidialog\_listview\_background.xml](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/res/drawable/unidialog_listview_background.xml).   
  
Далее, опуская детали реализации, рассмотрим несколько примеров вызова диалога.

**Пример 1. Список строк**

/\*\* Список строк - элементов списка \*/

private static String[] items = {"Item 1"

, "Item 2"

, "Item 3"

, "Item 4"};

...

//подготавливаем данные для диалога

DialogConfig dc = new DialogConfig("Main title"

, false //картинки не поддерживаются

, false //описания элементов отсутствуют

);

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

for (String s: items) {

list.add(s);

}

//вызываем диалог

UniDialogSingleSelector.showDialog(this

, dc //конфигурация диалога

, list //список элементов диалога

, new UniDialogSingleSelector.SingleItemReceiver<String>() {

@Override public void onSelectItem(String selectedItem) {

if (selectedItem == null) {

//пользователь отменил выбор

} else {

//пользователь выбрал selectedItem

}

}

});

В примере выводится список строк, из которого пользователь должен выбрать одну строку. Код делится на две части: подготовка данных для диалога и вызов диалога.   
  
Подготовка данных включает создание объекта DialogConfig, задающего конфигурацию диалога: название и два флага, указывающих требуется ли показывать картинки и описания элементов. В нашем случае элементы - это обычные строки, у них нет картинок и описаний, поэтому флаги выставлены в false.  
  
Диалог вызывается статической функцией UniDialogSingleSelector.showDialog, в которую передается исходная Activity, конфигурация диалога, список элементов и обработчик результатов. Обработчик задается в виде анонимного класса, реализующего функцию onSelectItem. Если пользователь отменил выбор, то в onSelectItem будет передат null. Если пользователь выбрал элемент - то в onSelectItem будет передан выбранный элемент.   
  
Функция UniDialogSingleSelector.showDialog является параметризированной. Параметром является тип элементов списка.

**Пример 2. Список элементов с картинками и описанием**

Более сложный пример. Введем класс для хранения элементов списка:

class CustomDataItem implements UniDialogUtils.ItemImageProvider, ItemDescriptionProvider {

public final String Title;

private final Bitmap Image;

public final String Description;

public CustomDataItem(String title, Bitmap image, String description) {

this.Title = title;

this.DrawableId = drawableId;

this.Description = description;

}

@Override public Bitmap getImage() {

return this.Image;

}

@Override public String toString() {

return this.Title;

}

@Override public String getDescription() {

return this.Description;

}

}

Класс реализует два вспомогательных интерфейса - ItemImageProvider иItemDescriptionProvider. Через эти интерфейсы диалога получает от элементов списка картинки и описания.

UniDialogUtils.DialogConfig dc = new UniDialogUtils.DialogConfig(items[2]

, true //CustomDataItem supports images

, true //CustomDataItem supports desctiptions

);

//предполагаем, что массивы ITEMS, IMAGES и DESCRIPTIONS

//содержат необходимые названия, картинки и описания

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

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

list.add(new CustomDataItem(ITEMS[i], IMAGES[i], DESCRIPTIONS[i]));

}

UniDialogMultySelector.showDialog(this

, dc

, list

, new UniDialogMultySelector.MultyItemsReceiver<CustomDataItem>() {

@Override public void OnSelectItems(List<CustomDataItem> selectedItems) {

if (selectedItems == null) {

//выбор отменен

} else {

//пользователь выбрал элементы

}

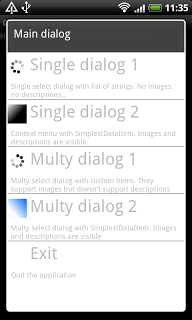
}

}

, true //все элементы выделены по умолчанию

);

Здесь мы используем диалог мульти-выбора - UniDialogMultySelector. Он отличается от UniDialogSingleSelector наличием дополнительного (последнего) параметра в showDialog, задающего набор выбранных по умолчанию элементов. Кроме того, используется другой класс для обработки результатов - MultyItemsReceiver. Этот класс реализует функцию OnSelectItems, принимающую список выбранных элементов или null, если выбор отменен.  
  
Вместо CustomDataItem можно использовать "стандартный" класс UniDialogUtils.SimplestDataItem, который дополнительно позволяет сохранять для элементов теги. А можно использовать сторонние классы, применяемые в других местах приложения - иногда это оказывается очень удобно. Единственное ограничение - если требуется показывать картинки и/или описания, то классы должны реализовывать один-два дополнительных интерфейса.   
  
В showDialog последним параметром вместо true/false можно явно передать список элементов, которые должны быть по умолчанию выделены.  
  
На картинке показан вид получившегося диалога.

[](http://3.bp.blogspot.com/-US9wbmNE7Ps/TuwPCebPAAI/AAAAAAAAAE4/JcIbQuX5a1M/s1600/dialog_screen.png)

Редактируя файлы ресурсов, можно подогнать вид диалога под "стандартный" для приложения вид.

**Заключение**

Выше я привел два примера использования диалога. В демонстрационном приложении можно посмотреть и другие примеры.   
  
Я активно использую данный диалог в своих приложения - удобно. Надеюсь, пригодится кому-нибудь еще. Пожелания и предложения, а так же замечания об ошибках - приветствуются :)  
  
[Скачать](http://code.google.com/p/dvsrc/downloads/detail?name=20111217DemoSelectDialog.7z&can=2&q=) демонстрационный проект и apk.   
  
[Просмотреть](http://code.google.com/p/dvsrc/source/browse/trunk/Android/DemoSelectDialogs/) исходные коды приложения DemoSelectDialog.