Клавиатура и аппаратные кнопки

Аппаратные и клавиатурные клавиши

Обработка аппаратных клавиш и клавиатуры имеет следующие методы

* **onKeyDown()** — вызывается при нажатии любой аппаратной клавиши;
* **onKeyUp()** — вызывается при отпускании любой аппаратной клавиши;

Кроме клавиш, есть еще другие методы обработки пользовательского ввода (здесь не рассматриваются):

* **onTrackballEvent()** — срабатывает при движениях трекбола;
* **onTouchEvent()** — обработчик событий сенсорного экрана, срабатывает при касании, убирания пальца и при перетаскивании.

Чтобы ваши компоненты и активности реагировали на нажатия клавиш, переопределите обработчики событий **onKeyUp()** и **onKeyDown()**:

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

// Обработайте нажатие, верните true, если обработка выполнена

return false;

}

@Override

public boolean onKeyUp(int keyCode, KeyEvent event) {

// Обработайте отпускание клавиши, верните true, если обработка выполнена

return false;

}

Параметр **keyCode** содержит код клавиши, которая была нажата; сравнивайте его со статическими кодами клавиш, хранящимися в классе **KeyEvent**, чтобы выполнять соответствующую обработку.

Параметр **KeyEvent** также включает в себя несколько методов: **isAltPressed()**, **isShiftPressed()** и **isSymPressed()**, определяющих, были ли нажаты функциональные клавиши, такие как Alt, Shift или Sym. Статический метод **isModifierKey()** принимает **keyCode** и определяет, является ли нажатая клавиша модификатором.

Кнопка Back: Вы уверены, что хотите выйти из программы?

Кнопка Back (Назад) закрывает приложение, точнее текущую активность, но если приложение состоит из одной активности, то это равносильно закрытию всего приложения. В большинстве случаев вам нет никакого дела до неуклюжего пользователя, который по ошибке нажал на кнопку Back вместо кнопки **Подарить разработчику миллион**. Но, если ваша программа, будучи запущенной на телефоне пользователя, потихоньку списывает деньги клиента в счет Фонда голодных котов, то нужно дать ему шанс задуматься и вывести диалоговое окно с вопросом: "А действительно ли вы хотите выйти из программы?"

Чтобы реализовать такую задачу, нужно переопределить поведение кнопки Back через метод **onBackPressed()**следующим образом:

@Override

public void onBackPressed() {

// super.onBackPressed();

openQuitDialog();

}

private void openQuitDialog() {

AlertDialog.Builder quitDialog = new AlertDialog.Builder(

CustomViewDemoActivity.this);

quitDialog.setTitle("Выход: Вы уверены?");

quitDialog.setPositiveButton("Таки да!", new OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// TODO Auto-generated method stub

finish();

}

});

quitDialog.setNegativeButton("Нет", new OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

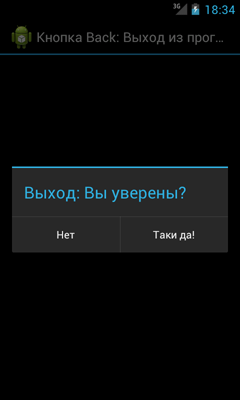
// TODO Auto-generated method stub

}

});

quitDialog.show();

}



Данный метод появился в Android 2.0. Для более ранних версий использовался стандартный код обработки **onKeyDown()**:

@Override

public boolean onKeyDown(int keyCode, KeyEvent event)

{

//replaces the default 'Back' button action

if(keyCode == KeyEvent.KEYCODE\_BACK)

{

// ваш код

}

return true;

}

Двойное нажатие на кнопку Back

Другой вариант - выход из приложения при двойном нажатии на кнопку Back. Удобно в тех случаях, когда считаете, что пользователь может случайно нажать на кнопку, например, во время активной игры. Приложение закроется, если пользователь дважды нажмёт на кнопку в течение двух секунд.

private static long back\_pressed;

@Override

public void onBackPressed() {

if (back\_pressed + 2000 > System.currentTimeMillis())

super.onBackPressed();

else

Toast.makeText(getBaseContext(), "Press once again to exit!",

Toast.LENGTH\_SHORT).show();

back\_pressed = System.currentTimeMillis();

}

Кнопка Home

Можно отследить нажатие кнопки **Home** через метод активности **onUserLeaveHint()**:

@Override

protected void onUserLeaveHint() {

Toast toast = Toast.makeText(getApplicationContext(), "Нажата кнопка HOME", Toast.LENGTH\_SHORT);

toast.show();

super.onUserLeaveHint();

}

Обработка кнопки Menu

У телефона, кроме кнопки Back, есть еще кнопка Menu для вызова команд меню (на некоторых устройствах). Если необходимо обрабатывать нажатия этой кнопки (например, управление в игре), то используйте следующий код (обычное и долгое нажатие):

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_MENU) {

event.startTracking();

editText.setText("Key Down"); //вывожу текст в текстовом поле

return true;

}

return super.onKeyDown(keyCode, event);

}

@Override

public boolean onKeyLongPress(int keyCode, KeyEvent event) {

if (keyCode == KeyEvent.KEYCODE\_MENU) {

editText.setText("Long Press"); //вывожу текст в текстовом поле

return true;

}

return super.onKeyLongPress(keyCode, event);

}

Должен заметить, что длинное нажатие трудно уловить, так как обычное нажатие постоянно подавляет это событие.

Другие кнопки

Ну на самом деле можно отслеживать не только нажатие кнопки Меню, но и кнопку Поиска и кнопки увеличения громкости.

@Override

public boolean onKeyDown(int keyCode, KeyEvent event) {

switch (keyCode) {

case KeyEvent.KEYCODE\_MENU:

Toast.makeText(this, "Нажата кнопка Меню", Toast.LENGTH\_SHORT)

.show();

return true;

case KeyEvent.KEYCODE\_SEARCH:

Toast.makeText(this, "Нажата кнопка Поиск", Toast.LENGTH\_SHORT)

.show();

return true;

case KeyEvent.KEYCODE\_BACK:

onBackPressed();

return true;

case KeyEvent.KEYCODE\_VOLUME\_UP:

event.startTracking();

return true;

case KeyEvent.KEYCODE\_VOLUME\_DOWN:

Toast.makeText(this, "Нажата кнопка громкости", Toast.LENGTH\_SHORT)

.show();

return false;

}

return super.onKeyDown(keyCode, event);

}

Обратите внимание, что для кнопки громкости возвращаем **false**, т.е. мы не переопределяем поведение кнопки, а оставляем её на усмотрение системы.

Пример работы с кнопками громкости можно посмотреть в статье [Рингтоны. Управление громкостью](http://developer.alexanderklimov.ru/android/ringtone.php#volume)

По такому же принципу работает метод **onKeyUp()**. Метод **onKeyLongPress()** можно использовать, если в методе **onKeyDown()** был задействован метод **event.startTracking()**, отслеживающий поведение кнопки. В нашем примере мы отслеживали кнопку **Volume\_Up**.

Прячем клавиатуру

Бывает так, что при запуске активности сразу выскакивает клавиатура. Если такое поведение не нравится, то пропишите в манифесте нужное значение у атрибута **android:windowSoftInputMode** ([см. ниже](http://developer.alexanderklimov.ru/android/keyboard.php#moveactivity)).

В некоторых случаях хочется убрать клавиатуру с экрана, не нажимая кнопку Back, а программно. В одном моём приложении, где было много текстовых полей, я воспользовался следующим кодом при щелчке кнопки:

// прячем клавиатуру. butCalculate - это кнопка

InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT\_METHOD\_SERVICE);

imm.hideSoftInputFromWindow(butCalculate.getWindowToken(),

InputMethodManager.HIDE\_NOT\_ALWAYS);

Код так выглядит, если писать его в **Activity**. Если расположить его в другом классе, экземпляр **Activity** нужно передать туда как параметр и вызывать методы как **activity.getApplicationContext()**, где **activity** - экземпляр **Activity**.

Можно избавить компонент от фокуса:

android:focusable="false"

Чтобы принудительно показать клавиатуру, используйте следующий код:

InputMethodManager imm = (InputMethodManager) getSystemService(Context.INPUT\_METHOD\_SERVICE);

imm.toggleSoftInput(InputMethodManager.SHOW\_FORCED, 0);

Кстати, повторный вызов метода закроет клавиатуру. Указанный способ не требует наличия элементов **View**.

Если продолжить тему показа клавиатуры, то может возникнуть следующая ситуация. Допустим у вас есть **DialogFragment** с **EditText**. При выводе диалогового окна вам нужно установить фокус на **EditText** и показать клавиатуру:

public class EditNameDialog extends DialogFragment {

private EditText editText;

public EditNameDialog() {

// Empty constructor required for DialogFragment

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.fragment\_edit\_name, container);

editText = (EditText) view.findViewById(R.id.txt\_yourName);

// Request focus and show soft keyboard automatically

editText.requestFocus();

getDialog().getWindow().setSoftInputMode(LayoutParams.SOFT\_INPUT\_STATE\_VISIBLE);

return view;

}

}

Либо используйте тег **<requestFocus />** для нужного EditText.

Изменить вид клавиатуры для данного EditText

Когда элемент **EditText** получает фокус, то появляется клавиатура. Можно установить нужный вид клавиатуры через [атрибут InputType](http://developer.alexanderklimov.ru/android/views/edittext.php#keyboard) или программно через метод **setInputType()**:

EditText ipt = new EditText(this);

ipt.setInputType(InputType.TYPE\_CLASS\_PHONE); //установит клавиатуру для ввода номера телефона

Другие варианты:

TYPE\_CLASS\_DATETIME - дата и время  
TYPE\_CLASS\_NUMBER - цифры  
TYPE\_CLASS\_TEXT - буквы

Переопределяем кнопку Enter

Кроме атрибута **InputType** можно также использовать атрибут **android:imeOptions** в компоненте **EditText**, который позволяет заменить кнопку **Enter** на клавиатуре на другие кнопки, например, **Next**, **Go**, **Search** и др. Возможны следующие значения:

* **actionUnspecified**: Используется по умолчанию. Система сама выбирает нужный вид кнопки (IME\_NULL)
* **actionGo**: Выводит надпись **Go**. Действует как клавиша Enter при наборе адреса в адресной строке браузера (IME\_ACTION\_GO)
* **actionSearch**: Выводит значок поиска (IME\_ACTION\_SEARCH)
* **actionSend**: Выводит надпись **Send** (IME\_ACTION\_SEND)
* **actionNext**: Выводит надпись **Next** (IME\_ACTION\_NEXT)
* **actionDone**: Выводи надпись **Done** (IME\_ACTION\_DONE)

Чтобы увидеть все варианты воочию, можете создать несколько текстовых полей и переключаться между ними:

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical" >

<EditText

android:id="@+id/editSearch"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:imeOptions="actionSearch"

android:singleLine="true" />

<EditText

android:id="@+id/editGo"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:imeOptions="actionGo"

android:singleLine="true" />

<EditText

android:id="@+id/editSend"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:imeOptions="actionSend"

android:singleLine="true" />

<EditText

android:id="@+id/editNext"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:imeOptions="actionNext"

android:singleLine="true" />

<EditText

android:id="@+id/editDone"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:imeOptions="actionDone"

android:singleLine="true" />

</LinearLayout>

Чтобы реагировать на нажатия разных состояний кнопки Enter, необходимо реализовать интерфейс **TextView.OnEditorActionListener**. Небольшой пример:

package ru.alexanderklimov.test;

import ...

public class TestActivity extends Activity implements OnEditorActionListener {

EditText editSearch;

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

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_test);

// Добавляем слушателя к компонентам

editSearch = (EditText) findViewById(R.id.editSearch);

editSearch.setOnEditorActionListener(this);

EditText editGo = (EditText) findViewById(R.id.editGo);

editGo.setOnEditorActionListener(this);

// и так далее

}

@Override

public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {

if (actionId == EditorInfo.IME\_ACTION\_SEARCH) {

// обрабатываем нажатие кнопки поиска

if (!editSearch.getText().toString().equals("cat")) {

Toast.makeText(this, "Не буду ничего искать!", Toast.LENGTH\_LONG).show();

}

return true;

}

if (actionId == EditorInfo.IME\_ACTION\_GO) {

// обрабатываем нажатие кнопки GO

return true;

}

return false;

}

}

В нашем примере если пользователь ищет что-то, не связанное с котом, то кнопка поиска не будет выполнять желание владельца устройства.

Также можно поменять текст на кнопке с помощью атрибута **android:imeActionLabel**:

<EditText

android:id="@+id/editDone"

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:imeOptions="actionDone"

android:imeActionId="@+id/action\_sign\_in"

android:imeActionLabel="Meow"

android:singleLine="true" />

Текст на кнопке поменялся, но вот обработка Enter из предыдущего примера у меня перестала работать. Мой неработающий код на память.

@Override

public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {

if (actionId == EditorInfo.IME\_ACTION\_DONE || actionId == R.id.action\_sign\_in) {

// обрабатываем нажатие кнопки

if(mEditText.getText().toString() != "кот"){

Toast.makeText(this, "Не буду ничего искать!", Toast.LENGTH\_LONG).show();

}

return true;

}

return false;

}

Upd: Читатель Максим Г. предложил следующее решение проблемы. Убираем атрибуты **imeOptions**, **imeActionId**, **imeActionLabel** и установим их программно.

mEditText = (EditText) findViewById(R.id.editDone);

mEditText.setOnEditorActionListener(this);

// вместо imeActionLabel и imeOptions

**mEditText.setImeActionLabel("Мяу", EditorInfo.IME\_ACTION\_DONE);**

@Override

public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {

boolean handled = false;

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

// обрабатываем нажатие кнопки

if(mEditText.getText().toString() != "кот"){

Toast.makeText(this, "Не буду ничего искать!", Toast.LENGTH\_LONG).show();

}

return handled;

}

return handled;

}

По желанию можете отслеживать только у нужного поля. Поставьте дополнительное условие после первого блока **if**:

// только у данного поля

if (v.getId == R.id.editDone){ }

Интерфейс OnKeyListener

Чтобы среагировать на нажатие клавиши внутри существующего представления из активности, реализуйте интерфейс **OnKeyListener** и назначьте его для объекта **View**, используя метод **setOnKeyListener()**. Вместо того, чтобы реализовывать отдельные методы для событий нажатия и отпускания клавиш, **OnKeyListener** использует единое событие **onKey()**.

myView.setOnKeyListener(new OnKeyListener() {

public boolean onKey(View v, int keyCode, KeyEvent event)

{

// TODO Обработайте нажатие клавиши, верните true, если

// обработка выполнена

return false;

}

});

Используйте параметр **keyCode** для получения клавиши, которая была нажата. Параметр **KeyEvent** нужен для распознавания типа события (нажатие представлено константой **ACTION\_DOWN**, а отпускание — **ACTION\_UP**).

Сдвигаем активность

Чтобы всплывающая клавиатура не заслоняла элемент интерфейса, который получил фокус, а сдвигала активность вверх, можно в манифесте для нужной активности прописать атрибут **android:windowSoftInputMode** с параметром **adjustPan**:

<activity

android:name=".CatsActivity"

android:label="@string/app\_name"

android:windowSoftInputMode="adjustPan" >

</activity>

Также доступны и другие параметры:

* **stateUnspecified** - настройка по умолчанию. Система сама выбирает подходящее поведение клавиатуры.
* **stateUnchanged** - клавиатура сохраняет своё последнее состояние (видимое или невидимое), когда активность с текстовым полем получает фокус.
* **stateHidden** - клавиатура скрыта, когда открывается активность. Клавиатура появится при наборе текста. Если пользователь переключится на другую активность, то клавиатура будут скрыта, но при возвращении назад клавиатура останется на экране, если она была видима при закрытии активности.
* **stateAlwaysHidden** - клавиатура всегда скрывается, если активность получает фокус.
* **stateVisible** - клавиатура видима.
* **stateAlwaysVisible** - клавиатура становится видимой, когда пользователь открывает активность.
* **adjustResize** - размеры компонентов в окне активности могут изменяться, чтобы освободить место для экранной клавиатуры.
* **adjustPan** - окно активности и его компоненты не изменяются, а сдвигаются таким образом, чтобы текстовое поле с фокусом не было закрыто клавиатурой.
* **adjustUnspecified** - настройка по умолчанию. Система сама выбирает нужный режим.

Параметры с префиксом **state** можно комбинировать с настройками с префиксом **adjust**:

Например, чтобы показать клавиатуру при старте активности, используйте **stateVisible**.

<activity android:windowSoftInputMode="stateVisible | adjustResize" />

Данные настройки доступны и программно. Например, код для **adjustResize**:

activity.getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT\_INPUT\_ADJUST\_RESIZE);

Кстати, этот код не сработает в полноэкранном режиме (флаг **FLAG\_FULLSCREEN**). Сверяйтесь с документацией.

Узнать выбранный язык на клавиатуре

Для определения текущего языка на клавиатуре можно использовать следующий код.

public void onClick(View view) {

InputMethodManager imm = (InputMethodManager)getSystemService(Context.INPUT\_METHOD\_SERVICE);

InputMethodSubtype ims = imm.getCurrentInputMethodSubtype();

String localeString = ims.getLocale();

Locale locale = new Locale(localeString);

String currentLanguage = locale.getDisplayLanguage();

EditText languageEditText = (EditText)findViewById(R.id.etNewItem);

Toast.makeText(getApplicationContext(), currentLanguage, Toast.LENGTH\_SHORT).show();

}

Следует быть осторожным с примером. На эмуляторе с Android 6.0 пример работал корректно. На реальных устройствах у меня корректно определялся русский язык, но при переключении на английский язык выдавал пустую строку или значение "zz". В этом случае можно прибегнуть к условиям **if** и проверять ожидаемое значение.

Запустить окно настроек клавиатур через намерение

Откроем окно настроек клавиатур.

Intent intent = new Intent(Settings.ACTION\_INPUT\_METHOD\_SETTINGS); // виртуальные клавиатуры

Intent intent = new Intent(Settings.ACTION\_HARD\_KEYBOARD\_SETTINGS); // API 24: реальные клавиатуры

if (intent.resolveActivity(getPackageManager()) != null) {

startActivity(intent);

}