

補足資料

●キーイベント処理サンプル

```
package jp.co.keyevent;

import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.Toast;

public class KeyEventSampleActivity extends Activity {
    /** 起動時初期処理 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    // これが、最も汎用的な、キーイベント処理コールバックメソッドです
    public boolean dispatchKeyEvent(KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_UP) { // キーが離された時
            switch (event.getKeyCode()) {
                case KeyEvent.KEYCODE_DPAD_CENTER: // 十字中央キー
                    Toast.makeText(this, "十字中央キー", Toast.LENGTH_SHORT).show();
                    return true;

                case KeyEvent.KEYCODE_DPAD_UP: // 十字上キー
                    Toast.makeText(this, "十字上キー", Toast.LENGTH_SHORT).show();
                    return true;

                default:
                    }
            }

        return super.dispatchKeyEvent(event);
    } // dispatchKeyEvent()
```

```

// 長押し系キーイベントは、キーダウンイベントと併用はできない。
// API レベル 5 以上で使用可能
public boolean onKeyLongPress(int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_DPAD_RIGHT: // 十字右キー
            Toast.makeText(this, "長押しで、十字右キー", Toast.LENGTH_SHORT).show();
            break;

        default:
            break;
    }
    return super.onKeyLongPress(keyCode, event);
} // onKeyLongPress()

public boolean onKeyDown(int keyCode, KeyEvent event) {
    //キーの状態を追跡する（長押しか否かを判別する）
    event.startTracking();
    return true;
} // onKeyDown()

// キーを押して離れたときに動きます
public boolean onKeyUp(int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_DPAD_DOWN: // 十字下キー
            Toast.makeText(this, "十字下キー", Toast.LENGTH_SHORT).show();
            break;
        case KeyEvent.KEYCODE_DPAD_CENTER:
            break;
        default:
            break;
    }
    return super.onKeyDown(keyCode, event);
} // onKeyUp()
} // class

```

●タッチイベント

Android には入力モードとして通常のキー入力モードの他にタッチモードが存在します。ユーザーが画面に一度タッチをすると入力モードはタッチモードに切り替わり、何らかのキー入力が発生したタイミングで、再び通常のキー入力モードに切り替わります。

タッチイベントとはタッチモードにおいてウィジェットがタッチされたタイミングで発生するイベントです。

Android のビュークラスは、onTouchEvent というコールバックメソッドを持っており、従ってビューから派生したクラス(独自のビューなど)に関しても、タッチイベントを取得することが可能です。

```
public boolean onTouchEvent(MotionEvent event);
```

また、自作のビューなどでは、ビューの設計時に implements OnTouchListener という記載をつけて、OnTouchListener というインターフェースを実装すると、このインターフェースの持つ onTouch というコールバックメソッドも使用することが可能になります。

```
public boolean onTouch(View v, MotionEvent event);
```

MotionEvent.getAction() で取得できる動作

この中では ACTION_OUTSIDE が見慣れませんが、View の範囲外をタッチした場合に呼ばれます。

定数名	説明
ACTION_DOWN	0x02 タッチ押下
ACTION_MOVE	0x00 指を持ち上げずにスライドさせた場合
ACTION_UP	0x01 指を持ち上げた場合
ACTION_CANCEL	0x03 UP+DOWN の同時発生(=キャンセル)の場合
ACTION_OUTSIDE	0x04 ターゲットとする UI の範囲外を押下

キャンセルの場合は、定数値 0×03 です。これはビットの OR 演算で見ると、 $(0 \times 02 \mid 0 \times 01)$ となり、DOWN と UP が同時に発生しています。

ユーザーは短期間のうちに指を離したことから、操作しようとしてやめています。DOWN/UP アクションは行わず、キャンセルで良いでしょう。

壁紙を替えるには WallpaperManager を使用します

替え方は

まず AndroidManifest でパーミッションを設定する必要があります

AndroidManifest.xml

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.     ... >
4.     .
5.     .
6.     .
7.     <uses-permission
8.         android:name="android.permission.SET_WALLPAPER" />
9. </manifest>
```

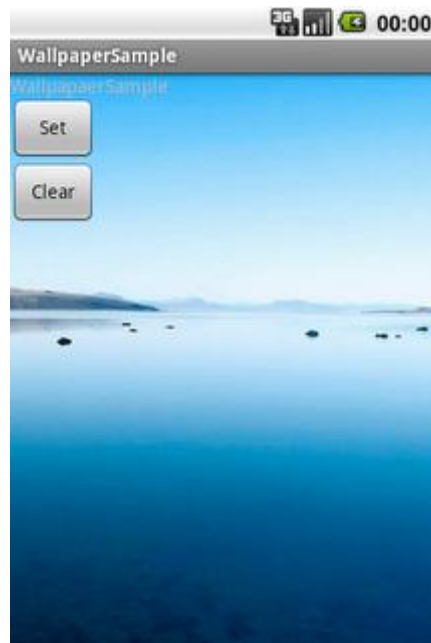
サンプルコード

```
1. public class WallpaperSample extends Activity implements OnClickListener {
2.     WallpaperManager mWM;
3.
4.     @Override
5.     public void onCreate(Bundle savedInstanceState) {
6.         super.onCreate(savedInstanceState);
7.         setContentView(R.layout.wallpaper_sample);
8.     }
```

```
9.     Button button;
10.     button = (Button) findViewById(R.id.Button_Set);
11.     button.setOnClickListener(this);
12.     button = (Button) findViewById(R.id.Button_Clear);
13.     button.setOnClickListener(this);
14.
15.     // WindowManager の取得
16.     mWM = WallpaperManager.getInstance(this);
17.
18.     // 壁紙の最小の幅, 最小の高さの取得
19.     int width = mWM.getDesiredMinimumWidth();
20.     int height = mWM.getDesiredMinimumHeight();
21. }
22.
23. @Override
24. public void onClick(View v) {
25.     switch(v.getId()) {
26.     case R.id.Button_Set:
27.         try {
28.             // 壁紙をリソースから設定
29.             mWM.setResource(R.drawable.icon);
30.         } catch (IOException e) {
31.             e.printStackTrace();
32.         }
33.         break;
34.     case R.id.Button_Clear:
35.         try {
36.             // 壁紙をデフォルトに戻す
37.             mWM.clear();
38.         } catch (IOException e) {
39.             e.printStackTrace();
40.         }
41.         break;
42.     }
43. }
44. }
```

プログラムを実行すると...

変更前



変更後



Activity の背景を壁紙にしたいときは

1. `<activity`

```
2.    android:name=".activity.WallpaperSample"
3.    android:theme="@android:style/Theme.Wallpaper">
```

とテーマを指定すれば OK です。

ちなみに

API 2.0 以前ではできません。

Android で直接ネット接続 (HTTP 通信)

サンプルコード)

```
import android.app.Activity;
import android.widget.TextView;
import android.os.Bundle;
import java.net.*;
import java.io.*;

public class HttpAndroidTest extends Activity {

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this setContentView(R.layout.main);

        TextView tv = (TextView) this.findViewById(R.id.textView01);

        try {
            URL url = new URL("http://www.google.co.jp");
            //指定URLにhttp通信接続の準備をする
            HttpURLConnection http = (HttpURLConnection) url.openConnection();

            //データの獲得要求を設定
            http.setRequestMethod("GET");
            //WEBサーバーに接続
```

```

        http.connect();
        //WEBデータを読み込む準備
        InputStream in = http.getInputStream();
        //データを入れる配列を用意
        byte b[] = new byte[1024];
        //配列にデータを流し込む
        in.read(b);
        //読み込み終了
        in.close();
        //通信を切断
        http.disconnect();

        //画面にデータを文字化して表示
        tv.setText(new String(b));
    } catch (Exception e) {
        tv.setText(e.toString());
    }

} // onCreate()
} // class

```

AndroidManifest.xml に `<uses-permission . . . >` を追記する

```

. . .
</application>
<uses-sdk android:minSdkVersion="3" />

```

```

<uses-permission android:name="android.permission.INTERNET" />

```

```

</manifest>

```

AndroidManifest.xml に ↑ を記述していなかったためにエラーで下記のログが出力されました。

```

"java.net.SocketException: Permission denied (maybe missing INTERNET
permission)"

```


アプリがインターネットを利用する場合はこの記述が必要ですのでお気をつけ下さい。