

「ケビン ホイトのコンポーネント」

本ドキュメントは、Adobeのプラットフォームエバンジェリストであるケビン ホイト(Kevin Hoyt)の[ブログ](#)で公開されている記事「[Some Flash Android Components](#)」のコンポーネントの使い方をヒム・カンパニー 永井勝則が自主的に記述したものです。

<http://www.himco.jp/>

[knagai@himco.jp](mailto:knagai@himco.jp)

(2010/10)

## 「ケビン ホイトのコンポーネント」

本記事執筆現在Flash CS5 ではAIR for Androidで作成するAndroidアプリケーションに適したコンポーネントが提供されていません。Adobeでは現在、FlexのモバイルフレームワークとしてHeroが策定中ですが、正式版のリリースは 2011 年とされており、これがFlash CS5 で使えるかどうかは不明です。



この状況下において、Adobe のプラットフォームエバンジェリストであるケビン ホイトは Flash CS5 の AIR for Android で使用できるコンポーネントを作成し、公開しています。これは簡易的なものではありませんが、Android アプリケーションを Android OS 風の外観で作成したいときに役立ちます。しかし公開ページではコンポーネントのごく簡単な作成コードが示されているだけなので、本記事ではコンポーネントごとの具体的な使い方を見ていきます。

### ○準備

1. コンポーネントを下記ページからダウンロードし、展開します。

<http://blog.kevinhoyt.org/?p=548>

すると components という名前のフォルダが現れます。中にはコンポーネント用の AS ファイルのほか、コンポーネントで使用するムービークリップなどを含む android.fla ファイルやコンポーネントをまとめた android.swc ファイルなどがあります。この android.swc を利用すると、Android アプリケーションでコンポーネントが使用できるようになりますが、android.swc では使用するフォントが”Droid Sans”に固定されているので、日本語が表示できません。そこで次の手順で、android.fla に含まれているダイナミックテキストフィールドのフォント指定をすべて、”Droid Sans”から”\_sans”に変更します。

2. android.fla ファイルを android\_JA.fla に別名保存します。

3. android\_JA.fla を Flash で開き、ライブラリの[comps]フォルダに含まれているムービークリップをそれぞれダブルクリックしてシンボル編集モードに移り、ダイナミックテキストフィールドのフォント指定をすべて、”Droid Sans”から”\_sans”に変更します。これで android\_JA.fla のライブラリから埋め込みフォントの”Droid Bold”と”Droid Regular”は削除できます。

ケビン ホイトのコンポーネントでは TweenMax というアニメーションライブラリが使われているので、これをコンピュータにダウンロードします。

4. [TweenMaxのページ](#)からTweenMaxのAS3.0 版をダウンロードします。

5. 展開すると greensock-as3 というフォルダが現れるので、適当な場所に保存します。使用するのはフォルダ内の greensock.swc です。

ケビン ホイトのコンポーネントの AS ファイルでは、TweenMax のインポートで、gs.TweenMax というパスが使われているので、これを com.greensock.TweenMax に変更します。

6. コンポーネントの components フォルダにある DatePicker.as、ListPicker.as、Menu.as、TimePicker.as を開き、TweenMax の import ステートメントの、

```
import gs.TweenMax;
```

を次のように変更します。

```
import com.greensock.TweenMax;
```

7. android\_JA.fla の [ActionScript 3.0 の詳細設定] の [ライブラリパス] タブで、greensock.swc をライブラリパスに追加します。

8. android\_JA.fla の [パブリッシュ設定] では [SWC 書き出し] が選択されているので、android\_JA.fla をパブリッシュすると、android\_JA.swc というファイルが作成されます。これから作成するコンポーネントのサンプル Flash ファイルでは、ライブラリパスにこの android\_JA.swc を指定します。準備は以上です。

○各コンポーネント

1) Button

コンストラクタメソッド:	
public function Button( label:String = null, width:Number = 193, style:String = "regular" )	ボタンに表示する文字 幅 文字の位置

label はボタンに表示する文字で、指定しない場合には表示されません。文字は 28 ピクセル、黒で固定されます。width はボタンの幅で、指定しない場合には 193 ピクセルになります。ボタンの幅は setWidth() メソッドで変更できます。style は文字の位置で、Button クラスの TRIGGER(左寄せ)か REGULAR(センター)定数を指定します。

プロパティ	
label:String	ボタンの文字の取得と設定ができます。

メソッド	
setWidth( value:Number ):void	ボタンの幅を設定します。

コードの例)

```
package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;

    public class ButtonTest extends Sprite {

        public function ButtonTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("ボタンテスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );

            var tapBtn:Button = new Button(
```

```
        "タップ",
        193,
        Button.TRIGGER
    );

    tapBtn.x = 30;
    tapBtn.y = ttl.y + 50;
    tapBtn.addEventListener( MouseEvent.CLICK, onTap );
    addChild( tapBtn );

    var clearBtn:Button = new Button(
        "クリア",
        193,
        Button.REGULAR
    );

    clearBtn.x = tapBtn.x + tapBtn.width + 30;
    clearBtn.y = ttl.y + 50;
    clearBtn.addEventListener( MouseEvent.CLICK, onClear );
    addChild( clearBtn );
}

private function onTap(evt:MouseEvent):void {
    info_txt.text = "タップされた¥n";
    info_txt.appendText("stageX : " + evt.stageX + "¥n");
    info_txt.appendText("localY : " + evt.localY + "¥n");
}

private function onClear(evt:MouseEvent):void {
    info_txt.text = "";
}
}
}
```



図1: 多くの場面で使用できる Button コンポーネント

なお compoInit()メソッドの最初では、画面のタイトルバーとして Title コンポーネントを使っています。Title は表示する文字を第1引数として指定し、第2引数で幅を指定します。指定しないときは 480 ピクセルが使用されます。

```
public function Title( label:String = null, width:Number = 480 )
```

## 2) CheckBox

コンストラクタメソッド:	
<pre>public function CheckBox(     label:String = "All day",     selected:Boolean = false )</pre>	チェックボックスの左に表示する文字 選択されているかどうか

label はチェックボックスの左に表示する文字で、指定しない場合には”All day”が表示されます。null を指定するとチェックボックスだけにできます。文字は 28ピクセル、黒で固定されます。selected はチェックボックスが選択されているかいないかを示す Boolean 値で、省略するとチェックされない状態で表示されます。

プロパティ	
label:String	チェックボックスの文字の取得と設定ができます。
selected:Boolean	選択されているかどうかを Boolean 値で示します。true を設定するとチェックされている状態が表せます。

## コードの例)

```
package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;

    public class CheckBoxTest extends Sprite {

        public function CheckBoxTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("チェックボックステスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );

            var chk:CheckBox = new CheckBox("チェック!",false);
            chk.x = 230;
            chk.y = 80;
            chk.addEventListener( MouseEvent.CLICK, onCheckTap );
            addChild( chk );
        }

        private function onCheckTap(evt:MouseEvent):void {
            var chk:CheckBox = evt.target.parent;
            if (chk.selected) {
                info_txt.text = "選択されている";
            }
        }
    }
}
```

```

        } else {
            info_txt.text = "選択されていない";
        }
    }
}
}
}

```



図2: チェックした状態の CheckBox コンポーネント

### 3) ComboBox

コンストラクタメソッド:	
<pre> public function ComboBox(     label:String = null,     width:Number = 451 ) </pre>	コンボボックスに表示する文字 幅

label はコンボボックスに表示する文字で、null を指定すると文字が表示されません。width はコンボボックスの幅で、指定しないときには 451 ピクセルが使用されます。

プロパティ	
label:String	コンボボックスの文字の取得と設定ができます。

#### コードの例)

```

package {

    import flash.display.Sprite;

```

```

import flash.display.StageScaleMode;
import flash.display.StageAlign;
import flash.events.MouseEvent;

public class ComboBoxTest extends Sprite {

    private var cmb:ComboBox;
    private var listing:ListPicker;

    public function ComboBoxTest() {

        stage.scaleMode = StageScaleMode.NO_SCALE;
        stage.align = StageAlign.TOP_LEFT;

        compoInit();
    }

    private function compoInit():void {

        var ttl:Title = new Title("コンボボックステスト");
        ttl.x = stage.stageWidth / 2 - ttl.width / 2;
        addChild( ttl );

        cmb = new ComboBox("サッカー");
        cmb.x = 13;
        cmb.y = ttl.y + ttl.height + 20;
        cmb.addEventListener( MouseEvent.CLICK, onComboTap );
        addChild( cmb );

        listing = new ListPicker();
        listing.addEventListener(
            ListEvent.CLICK, doListingClick );
        addChild( listing );
    }

    private function onComboTap(evt:MouseEvent):void {

```

```

        listing.show( ["磐田", "鹿島", "浦和", "清水"], "チーム選択");
    }

    private function doListingClick( evt:ListEvent ):void {
        cmb.label = evt.label;
        info_txt.text = cmb.label + "を選択。";
    }
}
}

```



図3: ComboBox コンポーネントで複数の項目から1つを選択した

#### 4) Footer

コンストラクタメソッド: <pre> public function Footer(     ok:String = "Ok",     cancel:String = "Cancel",     option:String = null ) </pre>	OK ボタンの文字 キャンセルボタンの文字 オプションボタンの文字
---	---

Footer はその名前の通り、画面下部に置くコンポーネントで、その画面の最終的な是認か否認を下すものようです。ボタンは OK(是認)、キャンセル(否認)ボタンがつけられ、オプションでもう1つつけられます。

Footer クラスは内部に Button コンポーネントの ok と cancel、btnoption を持っており、各ボタンのクリックでそれぞれ、OK、CANCEL、OPTION タイプの FooterEvent オブジェクトを送出します。

#### コードの例)

```

package {

```

```

import flash.display.Sprite;
import flash.display.StageScaleMode;
import flash.display.StageAlign;
import flash.events.MouseEvent;

public class FooterTest extends Sprite {

    public function FooterTest() {

        stage.scaleMode = StageScaleMode.NO_SCALE;
        stage.align = StageAlign.TOP_LEFT;

        compoInit();
    }

    private function compoInit():void {

        var ttl:Title = new Title("フッターテスト");
        ttl.x = stage.stageWidth / 2 - ttl.width / 2;
        addChild( ttl );

        var ftr:Footer = new Footer("OK","キャンセル","オプション");
        ftr.y = stage.stageHeight - ftr.height;
        ftr.addEventListener(FooterEvent.OK, onFooterOK);
        ftr.addEventListener(
            FooterEvent.CANCEL, onFooterCANCEL);
        ftr.addEventListener(
            FooterEvent.OPTION, onFooterOPTION);
        addChild( ftr );

        var lbl:Label = new Label(
            "Footer にはボタンが2つか3つつけられる");
        addChild(lbl);
        lbl.y = ftr.y - lbl.height - 10;
    }
}

```

```

private function onFooterOK(evt:FooterEvent):void {
    info_txt.text = "OK ボタンがタップされた";
}

private function onFooterCANCEL(evt:FooterEvent):void {
    info_txt.text = "キャンセルボタンがタップされた";
}

private function onFooterOPTION(evt:FooterEvent):void {
    info_txt.text = "オプションボタンがタップされた";
}
}
}
}

```



図4: Footer コンポーネントにはボタンが2つか3つつけられる

なお Footer の上には、Label コンポーネントを使ってテキストを表示しています。Label コンポーネントのインスタンスでは text プロパティを使って、テキストの取得と設定が行えます。

```
public function Label( label:String = null )
```

## 5) TextInput

コンストラクタメソッド:	
<pre>public function TextInput(     label:String = null,     width:Number = 460 )</pre>	<p>テキスト入力に表示する文字 幅</p>

label はテキスト入力フィールドに最初に表示する文字で、null を指定すると文字が表示されません。  
width はテキスト入力フィールドの幅で、指定しないときには 460 ピクセルが使用されます。

コードの例)

```
package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;
    import flash.events.Event;
    import flash.text.TextField;
    import flash.events.TextEvent

    public class TextInputTest extends Sprite {

        public function TextInputTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("テキスト入力テスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );
        }
    }
}
```

```
        var txtInput1:TextInput = new TextInput("CHANGE イベント");
        txtInput1.y = ttl.y + txtInput1.height + 10;
        addChild(txtInput1);
        txtInput1.field.addEventListener(
            Event.CHANGE, onTextChange);

        var txtInput2:TextInput = new TextInput(
            "TEXT_INPUT イベント");
        txtInput2.y = txtInput1.y + txtInput2.height + 10;
        addChild(txtInput2);
        txtInput2.field.addEventListener(
            TextEvent.TEXT_INPUT, onTextInput);

        info_txt.y = txtInput2.y + txtInput2.height + 50;
        info_txt.x = txtInput2.x + 4
    }

    private function onTextChange(evt:Event):void {
        var tf:TextField = evt.target as TextField;
        info_txt.text = tf.text;
    }

    private function onTextInput(evt:TextEvent):void {
        info_txt.text = evt.text;
    }
}
}
```



図5: TextInput に音声入力でテキストを入力した

## 6) Menu

コンストラクタメソッド:	
public function Menu( items:Array )	メニューで表示する項目の配列

Menu コンポーネントでは、Android デバイスのメニューキーを押すと、画面下部からメニュー項目を表示したメニューが出ます。メニュー項目は項目のタップで選択できます。項目は6個まで作成できます。配列の各要素は、path と label をプロパティ名とした Object オブジェクトで指定します。path には項目で表示する画像へのパス、label には項目で表示する文字を指定します。画像はコンパイル時、APK ファイルに含める必要があります。

```
[{path: "image1.png", label: "lbl1"},{path: " image1.png ", label: " lbl1"}]
```

Note: ただしコンポーネントの SWC ファイルはメニューキーのタッチを感知しないようだったので、元のコードを書き換えることにしました。具体的には、Menu.as の 265 行め当たりの、

```
if( event.keyCode == 95 )
```

を

```
if(event.keyCode == Keyboard.MENU)
```

に変えました (flash.ui.Keyboard のインポートも必要です)。そして Menu コンポーネントに必要な AS ファイル (Menu.as、MenuItem.as、MenuEvent.as) をプロジェクトフォルダにコピーしました。なお下の例では Title を使っているので SWC ファイルのライブラリパスは生かしています。

またメニューには search.png という虫めがねの画像が使用されるので、コンパイル時これを APK ファイル

に含める必要があります。

コードの例)

```
package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;

    public class MenuTest extends Sprite {

        public function MenuTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("メニューテスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );

            var mnu:Menu = new Menu( [
                {path: "search.png", label: "天丼"},
                {path: "search.png", label: "親子丼"},
                {path: "search.png", label: "玉子丼"},
                {path: "search.png", label: "鉄火丼"},
                {path: "search.png", label: "うな丼"},
                {path: "search.png", label: "? 丼"}
            ] );
            mnu.addEventListener( MenuEvent.CLICK, onMenuClick );
            addChild( mnu );
        }
    }
}
```

```

    }

    private function onClickMenu(evt:MenuEvent):void {
        info_txt.text = evt.label + "を選んだ"
    }
}
}
}

```



図6:メニューキーを押してメニューを表示させ、“玉子丼”をタップし、玉子丼を選んだ

## 7) TimePicker

TimePicker は時刻を扱うことができます。TimePicker は内部に TimeDialog インスタンスを持っています。

### [TimePicker]

プロパティ	
time:Date	時刻の取得と設定が行えます。

メソッド	
show( date:Date = null ):void	指定した時刻を表示します。

### [TimeDialog]

プロパティ	
time:Date	時刻の取得と設定が行えます。

メソッド	
formatShort( value:Date ):String	指定した時刻を短い形式(12時間)の文字で返します。静的メソッド。

#### コードの例)

```
package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;

    public class TimePickerTest extends Sprite {

        var timeDialog:TimeDialog;

        public function TimePickerTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("タイムピッカーテスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );

            var today:Date = new Date();
            timeDialog = new TimeDialog();
            timeDialog.time = today;
            var now:String = TimeDialog.formatShort(today);

            var lbl:Label = new Label("起動時の時刻 : " + now);
```

```

        lbl.x = ttl.x;
        lbl.y = ttl.y + lbl.height;
        addChild( lbl );

        var timeBtn:Button = new Button(
            "時刻の設定",282,Button.REGULAR);
        timeBtn.x = lbl.x;
        timeBtn.y = lbl.y + timeBtn.height + 10;
        timeBtn.addEventListener(MouseEvent.CLICK, onSetTime);
        addChild( timeBtn );
    }

    private function onSetTime(evt:MouseEvent):void {
        var timePicker:TimePicker = new TimePicker();
        addChild(timePicker);
        timePicker.show(timeDialog.time);
        timePicker.addEventListener( DialogEvent.OK, onTimeOk );
        timePicker.addEventListener(
            DialogEvent.CANCEL, onDialogCancel );
    }

    private function onTimeOk(evt:DialogEvent):void {
        var tPicker:TimePicker = evt.target.parent as TimePicker;
        if (tPicker != null) {
            timeDialog.time = tPicker.time;
            info_txt.text =
                TimeDialog.formatShort(tPicker.time);
            tPicker.removeEventListener(
                DialogEvent.OK, onTimeOk );

            tPicker.removeEventListener( DialogEvent.CANCEL, onDialogCancel );
            tPicker = null;
        }
    }

    private function onDialogCancel(evt:DialogEvent):void {

```



プロパティ	
date:Date	日付けの取得と設定が行えます。

メソッド	
formatShort( value:Date ):String	指定した日付けを短い形式(曜日、月日、年)の文字で返します。静的メソッド。

コードの例)

```
package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;

    public class DatePickerTest extends Sprite {

        var dateDialog:DateDialog;

        public function DatePickerTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("デイトピッカーテスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );

            var today:Date = new Date();
            dateDialog = new DateDialog();
            dateDialog.date = today;
        }
    }
}
```

```

var now:String = DateDialog.formatShort(today);

var lbl:Label = new Label("起動時の日付 : " + now);
lbl.x = ttl.x;
lbl.y = ttl.y + lbl.height;
addChild( lbl );

var timeBtn:Button = new Button(
    "日付の設定",282,Button.REGULAR);
timeBtn.x = lbl.x;
timeBtn.y = lbl.y + timeBtn.height + 10;
timeBtn.addEventListener(MouseEvent.CLICK, onSetDate);
addChild( timeBtn );
}

private function onSetDate(evt:MouseEvent):void {
    var datePicker:DatePicker = new DatePicker();
    addChild(datePicker);
    datePicker.show(dateDialog.date);
    datePicker.addEventListener( DialogEvent.OK, onDateOk );
    datePicker.addEventListener(
        DialogEvent.CANCEL, onDialogCancel );
}

private function onDateOk(evt:DialogEvent):void {
    var dPicker:DatePicker = evt.target.parent as DatePicker;
    if (dPicker != null) {
        dateDialog.date = dPicker.date;
        info_txt.text =
            DateDialog.formatShort(dPicker.date);
        dPicker.removeEventListener(
            DialogEvent.OK, onDateOk );
        dPicker.removeEventListener(
            DialogEvent.CANCEL, onDialogCancel );
        dPicker = null;
    }
}

```

```

    }

    private function onDialogCancel(evt:DialogEvent):void {
        var dPicker:DatePicker = evt.target.parent as DatePicker;
        if (dPicker != null) {
            info_txt.text = "日付の設定をキャンセルした";
            dPicker.removeEventListener(
                DialogEvent.OK, onDataOk);
            dPicker.removeEventListener(
                DialogEvent.CANCEL, onDialogCancel);
            dPicker = null;
        }
    }
}

```



図8: DatePicker を使って日付けを設定した

## 9) NumericStepper

これはもうおなじみのコンポーネントです。

コンストラクタメソッド: <pre> public function NumericStepper(     min:Number = 0,     max:Number = 0,     current:Number = 0 ) </pre>	指定範囲の最小値 指定範囲の最大値 現在の値
---	------------------------------

プロパティ	
current	現在の値の取得と設定ができます。
maximum	指定範囲の最大値の取得と設定ができます。
minimum	指定範囲の最小値の取得と設定ができます。

コードの例)

```

package {

    import flash.display.Sprite;
    import flash.display.StageScaleMode;
    import flash.display.StageAlign;
    import flash.events.MouseEvent;

    public class NumericStepperTest extends Sprite {

        public function NumericStepperTest() {

            stage.scaleMode = StageScaleMode.NO_SCALE;
            stage.align = StageAlign.TOP_LEFT;

            compoInit();
        }

        private function compoInit():void {

            var ttl:Title = new Title("数値ステッパータスト");
            ttl.x = stage.stageWidth / 2 - ttl.width / 2;
            addChild( ttl );

            var numericStepper:NumericStepper =
                new NumericStepper();
            numericStepper.minimum = 1;
            numericStepper.maximum = 20;
        }
    }
}

```

```

        numericStepper.current = 10;
        numericStepper.x =
            stage.stageWidth / 2 - numericStepper.width / 2;
        numericStepper.y = ttl.y + ttl.height + 50;
        addChild(numericStepper);
        info_txt.text = "今の値 : " + int(numericStepper.current);
        numericStepper.addEventListener(
            StepperEvent.CHANGE, onChange);
    }

    private function onChange(evt:StepperEvent):void {
        var nStepper:NumericStepper =
            evt.target as NumericStepper;
        info_txt.text = "今の値 : " + int(nStepper.current);
    }
}
}
}

```



図9: 数値を1つずつ増減する NumericStepper