Basic4Android Form Generator

Contents table

Disclaimer

This SOFTWARE PRODUCT is provided by El Condor "as is" and "with all faults." El Condor makes no representations or warranties of any kind concerning the safety, suitability, lack of viruses, inaccuracies, typographical errors, or other harmful components of this SOFTWARE PRODUCT. There are inherent dangers in the use of any software, and you are solely responsible for determining whether this SOFTWARE PRODUCT is compatible with your equipment and other software installed on your equipment. You are also solely responsible for the protection of your equipment and backup of your data, and El Condor will not be liable for any damages you may suffer in connection with using, modifying, or distributing this SOFTWARE PRODUCT.

You can use this **SOFTWARE PRODUCT freely, if you would you can credit me in program comment:**

El Condor – CONDOR INFORMATIQUE – Turin

Comments , suggestions and criticisms are welcomed: mail to rossati@libero.it

Conventions

Commands syntax, instructions in programming language and examples are with font `COURIER NEW`. The optional parties of syntactic explanation are contained between `[square parentheses]`, alternatives are separated by `|` and the variable parties are in *`italics`*.

1 Form generator

Form generator, briefly *FormGen*, is a class module for Basic4Android which allows to build and handle forms data; it is sufficiently generalised for a wide use.

1.1 Using the form generator

Before use FormGen the class `fgen` must be instantiate:

```
Sub Globals
...
      Dim fg As fgen      ' instantiate Form Generator class
...
End Sub
...
      fg.Initialize
```

The form is generated on calling the `fg` method:

instantiatedName`.fg(`*activity*`,`*dataDescription*`,`*subHandleAnswer*`,`*subHandleEvents*`)`

where *activity* is the activity which will contains the form, *dataDescription* is a character string containing the form components description, *subHandleAnswer* and *subHandleEvents* are characters string containing the name of the sub for handle data when the form is closed and the possible sub for handle events or an empty string if you wouldn't handle events; they are in the form:

moduleName`.`*functionName*.

Example:

```
...
Dim parms As String = "Slide,,5,S,3,10 -10;Rdb,Sexe,10,R,,M:Male|F:Female;"
...
fg.fg(Activity,Parms,"Main.handleAnswerTest","Main.handleEvents")
...
fg.fg(Activity,Parms,"Main.handleAnswerTest","")
...
```

1.2 Data description

Every view (or widget) is characterised by a list of attributes (comma separated) in this order: Field Name, Field Label, Length, Type, Default Value and Extra. Views are separated by semicolon.

1.2.1 Field Name

Is the name of the field, which is returned, when the form is closed, with the value associated; the name is case-sensitive and is used to access data and possibly handle the views.

1.2.2 Field Label

Label of Field or caption of button, if omitted it is used the Field Name.

1.2.3 Length

The length, in characters, of the view.

If the length is omitted is assumed the maximum from 20 and the length of the possibly default value.

For SeekBar is the length of the (unmodifiable) text which shows the value.

For the Radiobuttons is the length of the single view with his text.

1.2.4 View Type

- Buttons:

    - **B** button;
    - **BC** CallBack button;
    - **R** radio button, a set of Radio buttons.
- **CKB** heck box, values are 1 for selected check boxes, 0 otherwise;
- Spinners or Combos:
    - **CMB** spinner;
    - **CMT** is a spinner with Text associated for insert values not in spinner;
    - **F** file and directory;
- Text fields:
    - **N** numeric field;
    - **DN** decimal numeric field;
    - **S** seekbar or slider is an extension of the standard control: it works also on float number range and inverted direction i.e. the start value can be greater than the end value;
    - **P** password field, the data entered are masked;
    - **T** text field is the default if the Type is omitted;
    - **U** not modifiable field i.e. a protected field.

An **M** after the Type means that the field is mandatory, this is ignored for **B, CKB** and **U** type; if a mandatory field is omitted the form can't be submitted, and the label of omitted field is changed to red.

The Types are accepted also in lower case.

1.2.5 Default value

Is the value proposed in form.

For check box (**CKB**) if the default value is `check[ed]` or `1` then the check box appears checked.

For type **B** and **BC** buttons the default value `dis[able]` disable the button.

For **F** type is the initial Folder (and possibly file name) to start search, if it is omitted or if it is not a folder, the `File.DirRootExternal` is assumed.

The form is restored with the defaults values when the `Reset` button is clicked.

1.2.6 Extra

For text type view (**T**, **P**, **N**, **DN**), if present, is the hint that will appear if the EditText is empty.

For type CKB check box is a possibly description at right of check box.

In the CallBack Button is a name of the function called in the form: *`moduleName.functionName`*.

For `spinners,` and `radiobuttons` is an item list separated by `|`. For get a key instead the description, the item can have the form: `key:value`.

For type **S** Slider *Extra field* contains the start and end values in the form `start end`, e.g. `-5 5`; if omitted the range is `0 100`, if only one value is present, the default value for the second is `100`; the result can have decimals depending on the difference from `start` and `end` value, see table at right.

abs(start - end)	n. decimals
> 99	0
<100 and > 10	1
<10 and > 1	2
<1 and > 0.1	3
...	...

1.2.7 Handling Buttons

Form Generator inserts the `Ok` button, the `Cancel` button and the `Reset` button in function of the views contained in the form:

- the C`ancel` button is always present,
- the `Reset` button is present if there are data fields (e.g. Type **F**, **DN**, **N**, **T**, **R**, **CKB, CMB, CKT),**
- the `Ok` button is not present if there are types **B** Buttons, i.e. this is useful for show different captions or take different actions on closing form.

1.2.8 Pseudo fields

Pseudo fields are flavours for show form; they have a type.

1.2.8.1 Title

The *default* field is used as form title, if omitted the *label* field is used.

```
Title,Send mail parameters,20,TITLE;
```

1.2.8.2 Tip (Tool tip)

The *default* field is used for tooltip the *extra* field must contain the name of the type text view for which show the tool tip. *Label, name* and *length* are meaningless.

```
Dim form As String = "MailTo,Mail to,20,TM,elcondor@libero.it;" _
      & "Subject,,40,T;" _
      & "Message,,200,T,,Message here;" _
      & "Send,,10,B,,;" _
      & "Parms,,10,B;" _
      & "tip,,,tip,Insert subject,Subject;" _
      & "Title,Send mail,20,TITLE;"
```

1.2.9 Parameters summary by type

Type	Length	Default	Extra
B	Ignored	possibly `dis[able]`	
BC	Ignored	Value returned	Name of CallBack function
CKB		`1` or `check[ed]` = checked	Possible Description at right of check box
CMB		*`value`*	An item list separated by `|` : `[`*`key`*`:]`*`value`*
F		Initial folder	
S		Initial value	Start and end value, default is `0 100`
T,N,DN,P		Initial value	hint
U		Unmodifiable text	

1.3 Returned Values

The data are accessible in the Sub indicated as third parameter of the call, which is called when the buttons `Ok` or `Cancel` or the possible type B button are pushed.

The sub has a parameter which is the map which contains the data; the key is the field name, besides there is the element with key `fh_button` which contains the name of the button pushed.

1.4 CallBack

FormGen works on CallBack not only at the end of form, but also, possibly, for handle events (the fourth parameters of the call), or by a sub associated at **BC** buttons.

1.4.1 Handle Events

This function can be used to personalize the form e.g. modify the state of the view, perform controls end so on.

A functions receive an array which contains view name and event, besides, for views, contain the view handle, value end possibly extra field, see below.

Parameters		
View type	**Event**	**Note**
`Event`	`Start`	view name = `fh_start`, no view handle and value.
`Event`	`End`	view name = `fh_end`, no view handle and value.
`Event`	`Reset`	view name = `fh_reset`
`Button`	`CLICK`	
`EditText`	`FOCUS, LFOCUS`	Focus and lost focus.
`EditText`	`VREND`	At end of Voice recognition (if is active)
`Check box`	`CHK`	Value = 1 if checked, else 0
`Slider`	`SLIDE`	Extra is the handle of text containing the value.
`Spinner text`	`CLICK`	

The button `CLICK` event precedes the closing of the form, however, it is possible to inhibit the closing by setting the property `fh_yesToExit` to `False`.

```
Sub handleEvents(parm() As Object)      ' widget events handler
  Dim value As String = ""
  If parm.Length > 2 Then
       If parm.Length > 3 Then value = parm(3)
  End If
  Log("Handle event " & parm(0) & " event: " & parm(1) & " Value: " & value)
       Select parm(0)
         Case "fh_start","fh_reset"
         Case "Message"
              If parm(1) = "VREND" Then
                Dim txt As String = parm(3)
                parm(3) = txt.SubString2(0,1).ToUpperCase & txt.substring(1) & "."
              End If
         Case "Parms"
              fg.fg(Activity,"Prova,,,t","Main.handleAnswer","Main.handleEvents")
         Case "Send"
              sendMail(parm(0))
              fg.fh_yesToExit = False
       End Select
End Sub
```

Example of sub for handle events

1.4.1.1 Work with views

We can modify the view properties getting the view by the function `getHandle`; therefore for some properties there are specific functions:

- enable view: **`enable`**`(`*`viewName`*`)`
- disable view: **`disable`**`(`*`viewName`*`)`
- get the handle of the view: **`getHandle`**`(`*`viewName`*`)` In case of radiobutton is the handle of radiobutton checked.
- set some properties: **`setWidget`**`(`*`viewName`*`,`*`property`*`)`

where `property` can be: `enable`, `disable`, `visible`, `hidden`.

- Change the value: **`setValue`**(*`viewName`*,*`value`*)
- Get the view value: **`valueOf`**(*`viewName`*)

Examples:

```
fg.disable("btnGo")
Dim lbl As Label = fg.getHandle("title")
lbl.TextColor = Colors.Green
If fg.valueOf("Consent") = 1 Then fg.enable("btnGo")
fg.SetValue("Slider",0.2)
fg.setValue("Number",400)
fg.setValue("Spinner","Delta")
fg.setValue("UnMod","**********")
fg.setValue("Rdb","Married")
```

1.4.2 Handle CallBack

The function is called when a type **BC** button is clicked; the function receive the name of the button.
Note: you can also manage the callBack in event management function, in this case it is not necessary to specify a dedicated function.

1.5 Remarks

1.5.1 Data presentation

The data are presented in the order they appear in the parameters list, except for the Type **B** and **BC** buttons that appear together to buttons inserted by Form generator at the bottom of the form.

For view of Type Text if the length exceed the maximum characters allowed for the line, the view is multi lined; this maximum characters for line depends from the labels width.

The value of check box is a string containing 0 or 1 i.e. they must be compared like string:

```
cmd.Append(fg.iIF(fh_Data.Get("Mandatory") = "1","M",""))
```

1.5.2 Voice recognition

It is possible to use voice recognition for to enter text spoken into text field; this is done by the function `startVoiceRec`.

Voice recognition is deactivated, for enable uncomment:

```
'Dim VR As VoiceRecognition
'VR.Initialize("VR")
Sub startVoiceRec()
 'VR.Listen 'calls the voice recognition external activity
End Sub
```

In the event `VREND` one can access to the text and modify this (see the above sample).

```
Sub handleEvents(parm() As Object)' widget events handler
     Dim value As String = ""
     If parm.Length > 2 Then
              If parm.Length > 3 Then value = parm(3)
     End If
     Select parm(0)
        ...
        Case "Message"
           If parm(1) = "VREND" Then
             Dim txt As String = parm(3)
             parm(3) = txt.SubString2(0,1).ToUpperCase & txt.substring(1) & "."
           End If
        ...
```

```
      End Select

End Sub
```

1.6 Others functions and utilities

FormGen module contains some, may be, useful functions.

1.6.1 iIf

`iIf` function return an object depending on a test:

```
Dim Number As Int = fg.valueOf("Number")
Log(Number & " is " & fg.iIf((Number Mod 2) = 0,"even","odd"))
```

1.6.2 Right and Left string functions

Right and Left functions returns a pieces of string stripped by some extent:

```
Log(fg.Left(fg.Right("Condor Informatique",12),6))   ' Inform
```

1.6.3 Get filename

`stripFile` return a file without the directory components.

1.6.4 Extend Log function

`toText` is a function for speedy logs; it has two parameters, the first is a string containing some text and a formatting command (`%` or `%n` or `%n.d`), the second is an array of object. Every `%` is replaced by an element of the second parameter. The possible *n* is a length of output, the possible *d* is the number of decimals to show.

```
aaa = Array As Object(3,3.14,"vintun",False,True, 18723)
Dim t As String = "% %6.3 %11 %1 aaaa %3 %12.2"
Log(fg.toText(t,aaa))
```

The risult is:

```
3  3.140 vintun       f aaaa tru    18,723.00
```

1.6.5 Implode

`implode` function returns a string containing all elements of an array with separator between each element:

```
Log(fg.implode("<br>",aaa))
```

The risult is:

```
set<br>3<br>3.14<br>vintun<br>false<br>true<br>18723
```

1.6.6 Object type

`typeOf` is a function which return a type of object, stripping `java.lang` from what is returned by `GetType` Basic4Android function.

```
Select TypeOf(values(j))
   Case "Integer","Double"
      ...
```

1.6.7 Mask comma and semicolon

If label or default or extra contains commas or semicolons, the function `mask(data)` must be used:

```
instantiatedName.mask(data_to_be_masked)
cmd.Initialize
cmd.Append(fh_Data.Get("Name") & ",")
cmd.Append(fg.mask(fh_Data.Get("Label")) & ",")
cmd.Append(fh_Data.Get("Length") & ",")
cmd.Append(fh_Data.Get("Type"))
cmd.Append(fg.iIF(fh_Data.Get("Mandatory") = 1,"M",""))
```

```
cmd.Append("," & fg.mask(fh_Data.Get("Default")) & ",")
cmd.Append(fg.mask(fh_Data.Get("Extra")))
```

1.6.8 Sand box

The Sand box is a demo of *FormGen* and a tool for testing the forms. The initial form, not created by *FormGen,* has a text box and some buttons:

- Add button generate a form for create a view.
- Test button generate a form starting from what is contained in the text box.
- Sample button generate a sample form with some views and a callback button.

2 Differences from previous version

2.1 Buttons

Default field of type **B** and **BC** buttons can contain the disable command, instead of the value returned when the button is clicked; the value returned is the button name.

2.2 Check box

For checking Check boxes in addition to the value `1` is also accepted `check[ed]`.

3 Technical notes

3.1 Library

Form generator need Core library and Phone library (if use Voice recognition).

3.2 Maps

- `Elem` data fields key = *`fieldname`*, *value* = *`array field description`*
- `mapValues` combo/radio key = *`fieldname`* & *`visualizedValue`*, value = *`key`*
- `mapPanel` key = `fieldname`,value = *`view panel`*, contains handle to panel with label and view or for handle to buttons.
- `tipsMap` key = `fieldname`,value = tips
- `widgetRef` contains a reference to all view and TITLE pseudo field,
  key = *`fieldname`*, value = `array(`*`Id, widgetType, label, extraField`*`)`
  `extraField` is:
  - for button type **BC** a sub for callback,
  - for seekbar (type **S**) is the ID of text containing the value,
  - for file (type **F**) is the file path,
  - for spinner text (type CMT) is the ID of the spinner associated.

Name	Key	Value	Note
`Elem`	`Field name`	`Property field array`	
`widgetRef`	`Field name`	`Id,widgetType, label, extraField1, extraField2`	
`limits`	*`fieldName`*`Min`, *`fieldName`*`Max`	`value`	Slider limits
`mapValues`	`Field name & value`	`key`	For Combo box (spinner)
`fh_Data`	`Field name`	`value`	

3.3 Lists and arrays

- `rdbList` the data array contains normalized RadioButtons data description
- btns (finalButtons) : label, name. The name of Ok button is `Ok`.

4 May be in future

- Add Hidden Fields,
- handle CR and LF for Type **U**,
- field controls (e.g. range numeric, date, mail, etc.),
- date fields.

5 Alphabetical Index