Oracle Forms 10g – Forms Look and Feel

An Oracle Forms Community White Paper

François Degrelle f.degrelle@free.fr April 2007



Introduction	4
The look and feel tool	4
Buttons	7
Multi-selection Tlists:	7
Pre-defined Colour Schemes	8
The provided sample dialogs	9
How define Look and Feel in A Forms module	10
When-New-Form-Instance	10
When-Timer-Expired	11
When-New-Record-Instance	12
Post-Query	12
Documentation	13
The DrawLAF public properties:	13
ADD_IMAGE	13
ADD_LINE	13
ADD_RECT	14
ADD_TEXT	15
CLEAR	16
GET_SCHEME	16
GET_TEXT_HEIGHT	16
GET_TEXT_SIZE	16
GET_TEXT_WIDTH	16
SET_ENHANCED_POPLISTS	17
SET_GRADIENT_COLORS	17
SET_GRADIENT_DIRECTION	17
SET_GRADIENT_TEXT	17
SET_H_CYCLE	18
SET_LIST_ORIENTATION	18
SET_LOG	
SET_MULTI_SELECTION	19
SET_SCHEME	
SET_SORTED_LIST	
SET_TEXT	20
SET_TIME_KEY_SELECT	
SET_V_CYCLE	
SET_IMAGE	22
SET_IMAGE_OFF	23

SET_IMAGE_ON	23
SET_IMAGE_POSITION	
SET_SEPARATOR	24
SET_SHADOW_COLOR	24
SET_TEXT_POSITION	24
SET_DEBUG	25
SET_OPAQUE	26
SET_OPAQUE_ALL	26
SET_DEBUG	27
SET_OPAQUE	28
SET_OPAQUE_ALL	28
SET_DEBUG	29
GET_MAX_ROWS	30
GET_TIME_KEY_SELECT	30
SET_ENHANCED	30
SET_ENHANCED_ALL	
SET_MAX_ROWS	31
SET_TIME_KEY_SELECT	31
SET_DEBUG	32
CLEAR_LIST_SELECTION	33
GET_LIST_ORIENTATION	33
GET_LIST_SELECTION	34
GET_MULTI_SELECTION	35
GET_SORTED_LIST	35
SET_ENHANCED	35
SET_ENHANCED_ALL	35
SET_LIST_ORIENTATION	36
SET_MULTI_SELECTION	36
SET_SORTED_LIST	36
SET_DEBUG	37
The CSS file description	38
Introduction	38
Tags used to draw on the canvas	39
Tags for the block title	40
Tags for the block table header	42
Tags for the block table body	44
The sample dialog to update the CSS file	
Summary	46
About the author:	47

Oracle Forms 10g – Forms Look and Feel

INTRODUCTION

Oracle Forms application have traditionally behaved as, and looked like, desktop applications. Even when Oracle Forms applications were Web deployed, they still gave the user the feel of the desktop. This has been one of the strong plus points for businesses looking to evolve their applications seamlessly to the Web. However, with more and more users becoming familiar with more traditional Web UIs (like HTML), there is a requirement from some customers to change the look of their Forms application: to "freshen" it or align it with non-Forms Web applications.

This paper explains how you can separate the look of your Forms application from the functional design. All the decoration elements are read from an external CSS file, as currently done with HTML pages and so both HTML and Forms applications can share a common look and feel.

THE LOOK AND FEEL TOOL

This paper discusses the use a number of look and feel tools, namely a PL/SQL library (laf.pll) and a set of Java Beans and PJCs grouped in a JAR file (laf.jar).

The PL/SQL library contains a package of functions and procedures needed to decorate both canvas and block tables, while the JAR file contains the beans needed to paint over the canvas and overload the standards Forms widgets (buttons, check-boxes, radio groups and lists).

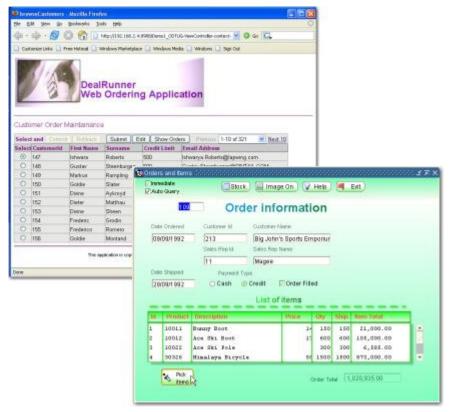


Figure 1 – Jdeveloper/ADF and Forms screens

Because the graphical information is read from a given CSS file, it is easy to change the look and feel of the application without modifying the Forms modules. Also, by using this tool your can "outsource" the look of the Forms application by separating the functional implementation from the presentation.

As well as influencing basic features like the colour scheme, you can add graphical elements such as images, lines, rectangles, texts and gradients. Figure 2 show an example of an Oracle Forms table block decorated like an HTML table, background colour gradients and images on canvases.

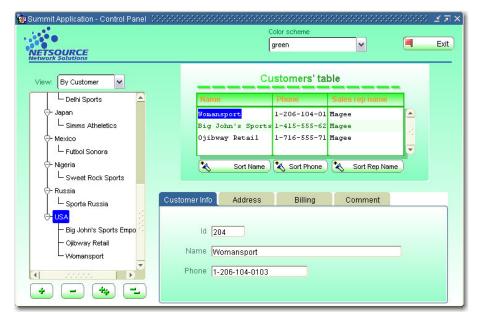


Figure 2 – Canvases and blocks Look and Feel

The power of this approach is that there is almost no work to be done at design time on the block (just set the item bevel to none). As shown in Figure 3, the look and feel of the Oracle Forms components is done at runtime by reading the CSS.

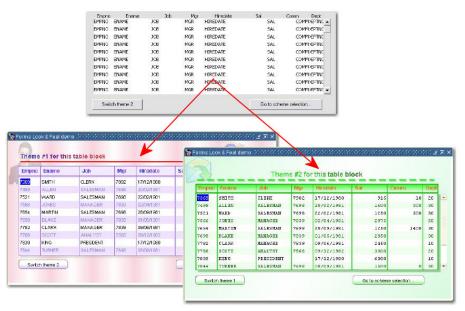


Figure 3 – "no work" design time

Buttons

As well as changing the appearance of standard Forms items, these widgets, like buttons, can be overloaded by Swing equivalents:

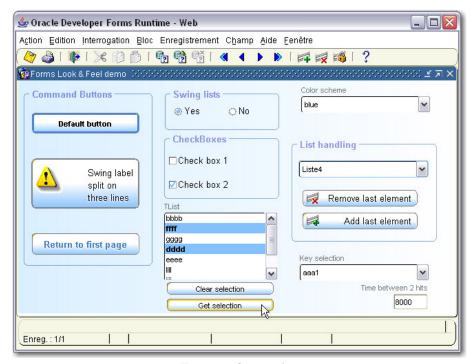


Figure 4 – Swing widgets

Figure 4 shows, amongst other things, a multi line button with icon image and graduated background.

Multi-selection Tlists:

Other common Forms control such as PopLists and ComboBoxes can also be changed to both look, and behave differently.

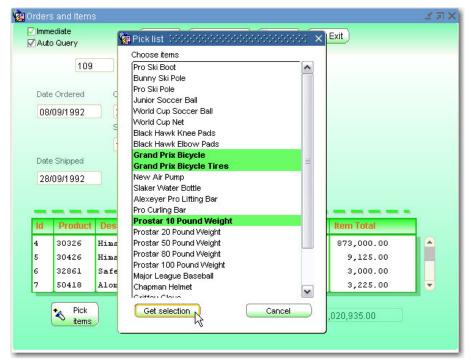


Figure 5 – Mutli-selection Tlists

Figure 5 shows an example of how a Poplists can pre-select values by typing more than one character, and Tlists can be multi-select as well as being sorted.

Pre-defined Colour Schemes

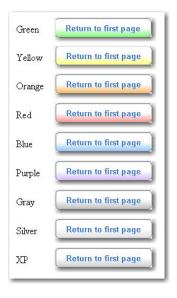


Figure 6 – Pre-defined Colour Schemes

The tool supports a number of pre-defined colour schemes. Figure 6 shows examples of the colour schemes applied to buttons.

THE PROVIDED SAMPLE DIALOGS

The two sample dialogs provided with this tool are adapted dialogs from the Oracle Forms "Summit" demo application.

customer_laf.fmb is the modified version of customer.fmb

orders laf.fmb is the modified version of orders.fmb

You can download the original application on OTN by following this link:

http://www.oracle.com/technology/products/forms/files/summit.zip

Follow the installation guide provided to configure this sample application, and then add the Look and Feel elements to it.

- laf.pll
- customer_laf.fmb
- orders_laf.fmb

Compile these three modules then move the executable equivalents (.plx, .fmx) to one of the directories pointed by your FORMS_PATH environment variable.

Copy the laf.jar file to your <DEVSUITE_HOME>/forms/java directory

Update your <DEVSUITE_HOME>/forms/server/formsweb.cfg file to add the jar file to the archive tag:

archive=frmall.jar,...,laf.jar

This tool uses Java gradients available from the JRE 1.4, so it cannot run with the JInitiator.

You have to configure a section of the formsweb.cfg file to run with the Sun Java plugin,

HOW DEFINE LOOK AND FEEL IN A FORMS MODULE

If you start a new module from scratch, the simplest method is to create the module from the **LAF_TEMPLATE.fmb** module. The new module will contain all the necessary objects:

- laf.pll library attachment
- form-level triggers
- control block with a bean area
- parameters
- visual attributes

For every canvas you want to decorate, you need a bean area which has its Implementation Class set to: oracle.forms.fd.DrawLAF

(You can drop this bean from the laf.olb object library).

Some default values needed to decorate canvases and table blocks are stored in the Form's parameters:

PM\$CSS_FILENAME	contains the default CSS file name
PM\$CANVAS	contains the default CSS file section name that stores canvas tags
PM\$TITLE	contains the default CSS file section name that stores block's title tags
PM\$HEADER	contains the default CSS file section name that stores block's header tags
PM\$BODY	contains the default CSS file section name that stores block's body tags
PM\$VA	contains the default visual attribute used to decorate the block

The goal of these parameters is to allow a maximum of flexibility (no need to recompile the module if you want to change the look). The template contains four form-level triggers:

When-New-Form-Instance

Used only to create a non-repeating timer:

```
Declare
   timer_id Timer ;
```

```
Begin
  timer_id := Create_Timer( 'laf_timer', 100, NO_REPEAT ) ;
End ;
```

When-Timer-Expired

Used to instantiate the painting process

Let's study the code of the sample module:

```
Begin
  If lower(Get_Application_Property( TIMER_NAME )) = 'laf_timer' Then
  -- form main initializations --
  Set_Custom_Property( 'CTRL.BEAN', 1, 'SET_SCHEME', 'purple' ); oldsymbol{1}
  If PKG_Look_And_Feel.Open_Css(:PARAMETER.PM$CSS_FILENAME) Then
    -- paint the canevases --
    PKG_LOOK_AND_FEEL.Paint_Canevas(:PARAMETER.PM$CANVAS, 'CTRL.BEAN'
); 3
    -- paint the blocks --
    PKG_LOOK_AND_FEEL.Paint_Block
     PC$Block
                   => 'EMP'
    ,PC$BeanName => 'CTRL.BEAN'
    ,PC$VA_Name => :PARAMETER.PM$VA
,PC$HeadClass => :PARAMETER.PM$HEADER
    ,PC$BodyClass => :PARAMETER.PM$BODY
    ,PC$TitleClass => :PARAMETER.PM$TITLE
     ,PC$Title => 'Theme #1 for this table block'
,PB$ScrollBar => True
    ,PC$Title
    Go_Block('EMP');
    -- populate the block --
    P_Execute_Query ;
    Synchronize ;
  End if ;
  -- read the global GUI properties --
  PKG_LOOK_AND_FEEL.Set_GUI_Properties( '.GUIProperties1', 'CTRL.LAF'
): 6
  -- hidden canvases that supports PJCs must be displayed once
  -- to initialize the bean areas and PJCs implementation classes
 Show_View('CV2'); 7
  -- set some individual properties --
  PKG_LOOK_AND_FEEL.Paint_Canevas('.canvas3', 'CTRL.LAF');
  Set_Custom_Property( 'CTRL.LAF', 1, 'SET_SCHEME', 'purple');
                                      1, 'SET_TEXT_POSITION', 'right'
  Set_Custom_Property( 'BL.BT2',
  Set_Custom_Property( 'BL.BT2',
                                     1, 'SET_IMAGE_POSITION', 'LM' );
```

Explanation

- 1. Setting the global scheme
- 2. Opening the CSS file
- 3. Painting the canvas corresponding to the :CTRL.BEAN bean area (first canvas)
- 4. Painting the EMP block
- 5. Populating the EMP block
- 6. Reading the common GUIs' properties
- 7. Displaying the second canvas in order to set/get some custom properties
- 8. Setting some individual item properties

When-New-Record-Instance

The EMP block in the sample dialog uses the odd/even decoration behaviour.

```
-- colorize each row --
PKG_LOOK_AND_FEEL.Fill_Table ;
```

Post-Query

Idem as When-New-Record-Instance, but when records are fetched from the database.

DOCUMENTATION

The DrawLAF public properties:

These properties can be set or read from any bean area whose Implementation Class property is set to: **oracle.forms.fd.DrawLAF**

The methods are split into two categories:

- Drawing onto the current canvas (the canvas that supports the bean area)
- Set/get some GUI global properties (applied on every concerned PJC of the current module)

ADD_IMAGE

ADD_xxx methods take an integer number as the first argument (indice). This integer is not

taken into account at this moment, but will

probably used in the future.

Add an image on the current canvas.

property value:

indice,full_image_name,X pos,Ypos[, transparancy_level [,width, height]]

indice is an incremental number greater than 0

image can be read:

- from the jar file : /image.gif
- from the client machine : c:/image.jpg
- from an internet url: http://...

transparancy_level must be a real value between 0 and 1

width and height can be provided to enforce the image to the given size.

ADD_LINE

Add a line on the current canvas

property value:

indice,x1,y1,x2,y2,width[,color[,transparency_level[,cap,join,dash]]]

indice is an incremental number greater than 0

if **color** is not specified, the default is white (r255g255b255).

if you want to specify a transparency level but not the color, put – (minus) to the color

transparency_level must be a real value between 0 and 1

You can have a dashed line by providing the 3 last parameters:

cap is the end line style

Allowed values are:

- BUTT
- ROUND
- SQUARE

join is the join line style

Allowed values are:

- BEVEL
- ROUND
- MITER

dash is 2 integers separated by a comma.

The first integer gives the length in pixel of the visible segment The second integer gives the length in pixel of the invisible segment (gap)

```
-- add a plain line --
Set_Custom_Property( ..., 'ADD_LINE', '1,10,10,300,10,2,-,.5 );
-- add a dashed line --
Set_Custom_Property( ..., 'ADD_LINE',
'1,10,10,300,10,2,r125g50b200,.5,square,miter,14,4' );
```

ADD_RECT

Add a rectangle on the current canvas

property value:

indice,x1,y1,x2,y2,width,frame_color,inside_color_transparency,inside_color,shade _color,gradient_start,gradient_end,gradient_direction,rounded_factor

default values are:

```
width: 1
```

frame_color : black
 other colors are : white

inside_color_transparency : 1
gradient_direction : LeftToRight

Other gradient direction can be: UpToDown

If you want to specify a transparency level but not the color, put – (minus) to the color

indice is an incremental number greater than 0

```
Set_Custom_Property( ..., 'ADD_RECT', '1,10,10,100,100,2,-,.5');
```

ADD_TEXT

Add a string to the current canvas.

property value is:

indice,text,x1,y1,font_name,wight,size,gradient_flag/color,transparency

default values are:

weight : plain
size : 8
color : black
transparency : 1

If **gradient_flag/color** = 'G' then the text use the gradient colors previously set by the SET_GRADIENT_TEXT method.

Else, the color defined (format rxgxbx) is use to paint the string.

indice is an incremental number greater than 0

```
-- set a text with defined color and a half transparency level
Set_Custom_Property( ..., 'ADD_TEXT',
'1,Hello,10,10,Arial,plain,12,r100g25b255,.5' );
-- set a text with a gradient pre-defined color and no
transparency level --
Set_Custom_Property( ..., 'ADD_TEXT',
'1,Hello,10,10,Arial,plain,12,G,1' );
```

CLEAR

Remove all objects created on the current canvas

```
Set_Custom_Property( ..., 'CLEAR', '' );
```

GET_SCHEME

Returns the current scheme.

If the scheme is not one of the predefined schemes, the method returns 'Custom'

```
Declare
  LC$Scheme Varchar2(50);
Begin
  LC$Scheme := Get_Custom_Property( ..., 'GET_SCHEME' );
End;
```

GET_TEXT_HEIGHT

Returns the height in pixel corresponding to the given text (with SET_TEXT)

```
Set_Custom_Property( ..., 'SET_TEXT', 'Hello,Arial,bold,12' );
height := Get_Custom_Property( ..., 'GET_TEXT_HEIGHT' );
```

GET TEXT SIZE

Returns the couple width, height in pixel corresponding to the given text (with SET_TEXT)

values are separated by a comma.

```
Set_Custom_Property( ..., 'SET_TEXT', 'Hello,Arial,bold,12');
SiZe := Get_Custom_Property( ..., 'GET_TEXT_SIZE' ) ;
```

size would have the following content: width, height (e.g.: 100,20)

GET_TEXT_WIDTH

Get the width in pixel corresponding to the given text (with SET_TEXT)

```
Set_Custom_Property( ..., 'SET_TEXT', 'Hello,Arial,bold,12' );
width := Get_Custom_Property( ..., 'GET_TEXT_WIDTH' ) ;
```

SET_ENHANCED_POPLISTS

Transform all lists (PopList, Tlist and CombBox) of the form module to enhanced (Swing) or standard

```
property value is 'true' or 'false'

Set_Custom_Property( ..., 'SET_ENHANCED_POPLISTS', 'true' );
```

SET_GRADIENT_COLORS

Set the two colors needed to paint a gradient.

The two colors are of type RxGxBx and separated by a comma. The first color is the start color, the second is the end color.

```
Set_Custom_Property(,'SET_GRADIENT_COLORS','r0g0b0,r80g80b80');
```

SET_GRADIENT_DIRECTION

Indicates the drawing gradient direction.

Allowed values are:

- LeftToRight (default)
- UpToDown
- LeftUpToRightDown
- LeftDownToRightUp

SET_GRADIENT_TEXT

Indicates the text gradient direction.

Identical that SET_GRADIENT_COLORS but used to set the gradient for texts. These gradient colors are set to be used by a following ADD_TEXT() call.

SET_H_CYCLE

In case a cycling gradient, allow to set the cycle value.

Default is 0

The value is given in pixels.

If the value starts with / the value is considered as a divisor factor.

```
-- gradient cycles 4 times horizontally in the current canvas -- Set_Custom_Property( ..., 'SET_H_CYCLE', '/4' );
```

SET_LIST_ORIENTATION

Set every current form enhanced(Swing) Poplist orientation.

Allowed values are:

- HORIZONTAL_WRAP
- VERTICAL_WRAP
- VERTICAL

```
Set_Custom_Property(..., 'SET_LIST_ORIENTATION', 'VERTICAL_WRAP');
```

SET_LOG

Turn ON/OFF the logging messages.

By default the all logs are OFF.

Allowed values are: 'true' or 'false'.

```
Set_Custom_Property( ..., 'SET_LOG', 'true' );
```

SET_MULTI_SELECTION

Set every current form enhanced (Swing) Tlists multi-selection true/false.

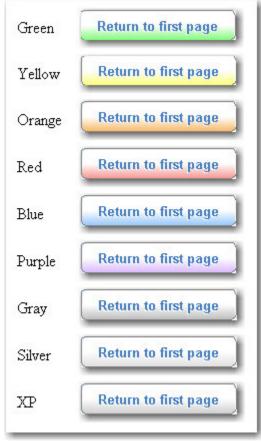
At creation time, each Tlist default value is 'false'.

```
Set_Custom_Property( ..., 'SET_MULTI_SELECTION', 'true' );
```

SET_SCHEME

Set the current scheme.

Available values are:



'SET_SCHEME', 'purple');

Set_Custom_Property(...,

SET_SORTED_LIST

Set every current form enhanced (Swing) Tlists sorted flag.

```
Available values are: 'true' or 'false'

Default creation value is 'false'

Set_Custom_Property( ..., 'SET_SORTED_LIST', 'true' );
```

SET_TEXT

Send a text to the bean to calculate its dimensions.

```
property value is:
```

text,font_name[,weight,size]

```
default size is : 8
default weight is : plain
```

Available weight are:

- plain
- bold
- italic
- bolditalic

```
Set_Custom_Property( ..., 'SET_TEXT', 'Hello, Arial, bold, 12' );
```

The calculated dimensions can, then, be read with the GET_TEXT_xx methods.

SET_TIME_KEY_SELECT

Set the number of millisecond between each keyboard hit in all Poplists.

A high number allows entering several characters added to each other to pre-

select the attempted value.

For instance, with a value of 5000, the end user has 5 seconds to cumulate keys to find the correct list entry ("abdominal" by entering abd, for instance).

With a lower value (for instance 100), the end user has only 1/10th of second to concatenate characters.

In the reality, each character hit set the focus on the list entry that starts with the key just hit.

```
Set_Custom_Property( ..., 'SET_TIME_KEY_SELECT', '5000' );
```

SET_V_CYCLE

Identical to SET_H_CYCLE but for the vertical gradient cycling.

```
-- gradient cycles every 20 pixels vertically in the current canvas -- Set_Custom_Property( ..., 'SET_V_CYCLE', '20');
```

LAF XP Button.java - public properties

These properties can be set or read from any Push button whose Implementation Class property is set to: **oracle.forms.fd.LAF_XP_Button**



SET_IMAGE

Set the current image of the button.

property value:

full_image_name[,image_position]

full_image_name can be one of the following:

- image read from the jar file: /image.gif
- image read from the client machine : c:/image.jpg
- image read from an internet url: http://...

image_position can be one of the following:

- LT Left Top
- **CT** Centre Top
- RT Right Top
- **LM** Left Middle
- **CM** Centre Middle
- RM Right Middle

- LB Left Bottom
- **CB** Centre Bottom
- RB Right Bottom

```
-- set an image stored in the jar file at Left Middle position Set_Custom_Property( '...', 1, 'SET_IMAGE', '/del.gif,LM' ) ;
```

SET_IMAGE_OFF

Set the image when the mouse exits the button

property value:

See SET_IMAGE

SET_IMAGE_ON

Set the image when the mouse enters the button

property value:

See SET_IMAGE

SET_IMAGE_POSITION

To set the new image position

Position can be one of the following:

- LT Left Top
- **CT** Centre Top
- RT Right Top
- LM Left Middle
- **CM** Centre Middle

- **RM** Right Middle
- LB Left Bottom
- **CB** Centre Bottom
- **RB** Right Bottom

```
-- set image to Centre Middle position --
Set_Custom_Property( '...', 1, 'SET_IMAGE_POSITION', 'CM' );
```

SET_SEPARATOR

To change the default character used to split the label into multiple lines

The default character is | (Alt-124)

You can enforce multi-line button's label by adding this separator in the label property.

e.g.: Label | split on | three lines

```
Set_Custom_Property( '...', 1, 'SET_SEPARATOR', '^' );
```

SET_SHADOW_COLOR

To set the color used for the label's shadow

The default color is white

```
Set_Custom_Property('...', 1, 'SET_SHADOW_COLOR', '200, 200, 200');
```

SET_TEXT_POSITION

To set the text's position in the button

Available values are:

- Left
- Centre

• Right

```
Set_Custom_Property( '...', 1, 'SET_TEXT_POSITION', 'right' );
```

SET_DEBUG

Turn ON/OFF the debug messages.

By default, all logs are OFF.

Allowed values are: 'true' or 'false'.

```
Set_Custom_Property( '...', 1, 'SET_DEBUG', 'true' );
```

LAF XP CBox.java - public properties

These properties can be set or read from any Check Box item whose Implementation Class property is set to: oracle.forms.fd.LAF_XP_CBox



SET_OPAQUE

Draw the check box background or not.

property value:

true|false

The standard Forms items use always a background color to draw the GUI component onto the canvas.

If you use a gradient canvas, the gradient is hidden by this background and the result is not nice.

In this case, set the opaque property to false.

SET_OPAQUE_ALL

Idem as SET_OPAQUE but for all the check boxes of the current module.

property value:

See SET_OPAQUE

SET_DEBUG

Turn ON/OFF the debug messages.

By default, all logs are OFF.

Allowed values are: 'true' or 'false'.

Set_Custom_Property('...', 1, 'SET_DEBUG', 'true');

LAF XP RadioButton.java - public properties

These properties can be set or read from any Radio Group item whose Implementation Class property is set to: oracle.forms.fd.LAF_XP_RadioButton



SET_OPAQUE

Draw the radio button background or not.

property value:

true|false

The standard Forms items use always a background to draw the GUI component onto the canvas.

If you use a gradient canvas, the gradient is hidden by this background and the result is not nice.

In this case, set the opaque property to false.

SET_OPAQUE_ALL

Idem as SET_OPAQUE but for all the radio buttons of the current module.

property value:

See SET_OPAQUE

SET_DEBUG

Turn ON/OFF the debug messages.

By default, all logs are OFF.

Allowed values are: 'true' or 'false'.

Set_Custom_Property('...', 1, 'SET_DEBUG', 'true');

LAF XP PopList.java - public properties

These properties can be set or read from any List Item (Poplist) whose Implementation Class property is set to: **oracle.forms.fd.LAF_XP_PopList**



GET_MAX_ROWS

Returns the maximum number of rows displayed in the list.

property value:

```
number := Get_Custom_Property( '...', 1, 'GET_MAX_ROWS' ) ;
```

GET_TIME_KEY_SELECT

Returns the amount of milliseconds allowed to enter more than one character property value:

```
number := Get_Custom_Property('...', 1, 'GET_TIME_KEY_SELECT') ;
```

SET_ENHANCED

Set/unset the enhanced (Swing) Poplist

property value: true | false.

Set the value to 'false' to get the native Forms Poplist widget.

Set it to 'true' to overload the native Forms Poplist with a Swing list component.

```
Set_Custom_Property( '...', 1, 'SET_ENHANCED', 'true' );
```

SET_ENHANCED_ALL

Idem as SET_ENHANCED but for every Poplist of the current module.

SET_MAX_ROWS

Set the maximum number of rows visible when the list is deployed.

property value:

Any integer greater than 0

Without indication, the Poplist is created with a default value of 6

```
Set_Custom_Property( '...', 1, 'SET_MAX_ROWS', '10' );
```

SET_TIME_KEY_SELECT

Set the number of milliseconds allowed to enter more than one character to preselect the list value.

By default, when you hit a key, the list displays the first occurrence found that starts with the letter entered.

With a low value, the end user won't have the possibility to pre-select for instance the value "abdominal" by hitting abd on the keyboard. The list will display the first value that starts with the letter d. With a higher level, the list value is pre-selected with the concatenation of the several keys hit within the corresponding laps of time. So with a value of 5000, the end user has 5 seconds to concatenate the 3 keys hit then he can pre-select the value "abdominal" by hitting abd.

property value: Any integer greater than 0

Without indication, the Poplist is created with a default value of 100

```
Set_Custom_Property( '...', 1, 'SET_TIME_KEY_SELECT', '3000' );
```

SET_DEBUG

Turn ON/OFF the debug messages.

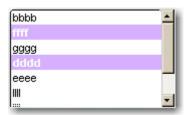
By default, all logs are OFF.

Allowed values are: 'true' or 'false'.

```
Set_Custom_Property( '...', 1, 'SET_DEBUG', 'true' );
```

LAF XP TList.java - public properties

These properties can be set or read from any List Item (Tlist) whose Implementation Class property is set to: **oracle.forms.fd.LAF_XP_TList**



CLEAR_LIST_SELECTION

Clear the selected values.

property value: none

```
Set_Custom_Property( '...', 1, 'CLEAR_LIST_SELECTION', '' );
```

GET_LIST_ORIENTATION

Returns the list orientation (only with enhanced Tlist)

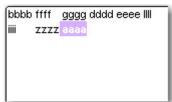
```
char := Get_Custom_Property('...', 1, 'GET_LIST_ORIENTATION');
```

The possible returned values can be:





VERTICAL_WRAP



HORIZONTAL_WRAP

GET_LIST_SELECTION

Returns the comma delimited list of selected indexes

```
char := Get_Custom_Property( '...', 1, 'GET_LIST_SELECTION' ) ;
```

If the user has selected the 1st, 3rd and 5th value the method will return: 1,3,5

The **laf.pll** PL/SQL library contains a function that takes a delimited string and returns a collection of strings:

```
Declare
  LT$Coll   PKG_LOOK_AND_FEEL.TYP_TAB_STRINGS;
  LC$Sel   Varchar2(256);

Begin
  -- get the multi-selection values --
  LC$Sel := Get_Custom_Property('BL.LIST', 1, 'GET_LIST_SELECTION');
  PKG_LOOK_AND_FEEL.To_String_Collection( LC$Sel, LT$Coll );
  -- read the elements --
  For i In LT$Coll.First .. LT$Coll.Last Loop
       LC$Sel := Get_List_Element_Label( 'BL.LIST', LT$Coll(i)+1 );
  End loop;
End;
```

GET_MULTI_SELECTION

Returns a Varchar2 that indicates if the multi-selection is allowed for the List Item Returned value can be 'true' or 'false'

```
char := Get_Custom_Property( '...', 1, 'GET_MULTI_SELECTION' ) ;
```

GET_SORTED_LIST

Returns a Varchar2 that indicates if the enhanced list is sorted

Returned value can be 'true' or 'false'

```
char := Get_Custom_Property( '...', 1, 'GET_SORTED_LIST' );
```

SET_ENHANCED

Set/unset the enhanced (Swing) Tlist

```
property value : true | false.
```

Set the value to 'false' to get the native Forms Tlist widget.

Set it to 'true' to overload the native Forms Tlist with a Swing list component.

```
Set_Custom_Property( '...', 1, 'SET_ENHANCED', 'true' );
```

SET_ENHANCED_ALL

Idem as SET_ENHANCED but for every Tlist of the current module.

SET_LIST_ORIENTATION

Set the enhanced Tlist orientation.

Allowed values:

- VERTICAL
- VERTICAL_WRAP
- HORIZONTAL_WRAP

Without indication, the Tlist is created with a default value of VERTICAL

```
Set_Custom_Property( '...', 1, 'SET_LIST_ORIENTATION',
'VERTICAL_WRAP' ) ;
```

SET_MULTI_SELECTION

Set the multi-selection flag.

Allowed values:

'true': you can select more than one value in the list

'false': you can select only one value

Without indication, the Tlist is created with a default value of 'false'

```
Set_Custom_Property( '...', 1, 'SET_MULTI_SELECTION', 'true' );
```

If the value is set to 'true', you can get the list of selected indexes with the GET_LIST_SELECTION() method.

SET_SORTED_LIST

Set the sort flag (on enhanced Tlist only).

Allowed values:

```
'true': the Tlist is sorted
```

'false': the Tlist is not sorted

Without indication, the Tlist is created with a default value of 'false'

```
Set_Custom_Property( '...', 1, 'SET_SORTED_LIST', 'true' );
```

SET_DEBUG

Turn ON/OFF the debug messages.

By default, all logs are OFF.

Allowed values are 'true' or 'false'.

```
Set_Custom_Property( '...', 1, 'SET_DEBUG', 'true' );
```

THE CSS FILE DESCRIPTION

Introduction

Every block decoration's property is read from an external CSS file. The name of this file is read from the **PM\$CSS_FILENAME** form parameter.

The default value is: c:\forms.css

Note: the tags used by this tool are Forms specific.

The file is opened with the *PKG_Look_And_Feel.Open_Css()* library's function that returns TRUE is the file has been correctly read.

```
If PKG_Look_And_Feel.Open_Css( 'c:\forms1.css' ) Then
   /* file open, we can continue */
End if;
```

There are four sections used to decorate the form module: One for the current canvas and three for the block itself (title, header and body). The default values can be read from the form parameters:

Canvas section: PM\$CANVAS

Block title section: **PM\$TITLE**

Block header section: **PM\$HEADER**

Block body section: **PM\$BODY**

The Visual Attribute used to decorates the block table's lines is read from the **PM\$VA** parameter.

The current canvas decoration is done with the *PKG_LOOK_AND_FEEL_Paint_Canevas()* library's procedure.

```
PKG_LOOK_AND_FEEL.Paint_Canevas('.canvas1', 'CTRL.BEAN' );
```

The first parameter is the CSS section that contains the tags. The second parameter indicates the bean area which the Implementation Class property is set to: oracle.forms.fd.DrawLAF

Note: The canvas to decorate is the one attached to the given bean area.

The block decoration is done through the *PKG_LOOK_AND_FEEL.Paint_Block()* **laf.pll** library's procedure.

```
Procedure Paint_Block

(

PC$Block IN Varchar2, -- the block name to decorate

PC$BeanName IN Varchar2, -- the associated bean area

PC$VA_Name IN Varchar2, -- the associated visual attribute

PC$HeadClass IN Varchar2, -- the table header CSS class name

PC$BodyClass IN Varchar2, -- the table body CSS class name

PC$TitleClass IN Varchar2 Default Null, -- the table title CSS class name

PC$Title IN Varchar2 Default Null, -- the block title PB$ScrollBar IN Boolean Default True -- scrollbar exists on block true/false
```

Tags used to draw on the canvas

```
.canvas1 {
    type:canvas
    gradient-colors: r170g250b190,r255g255b255
    gradient-vcycle: /2
    gradient-hcycle: /2
    image1[2[n]]: /env-64.gif,1,1,.3
}
```

gradient-colors

Can contain the 2 colors used to construct a gradient background. The colors are of type RxGxBx and are separated by a comma. The first color is the gradient start color. The second color is the gradient end color.

gradient-vcycle

Could contain an integer value expressed in pixel (e.g. 20) or a quotient (e.g. /4) to set a vertical cycle to the gradient. If the value equals 0 (zero) or is not provided, then gradient has no cycle (repetition).

gradient-hcycle

Could contain an integer value expressed in pixel (e.g. 20) or a quotient (e.g. /4) to set a horizontal cycle to the gradient. If the value equals 0 (zero) or is not provided, then gradient has no cycle (repetition).

image

This tag allows drawing images onto the canvas. You can have as many as you want. Each image is coded with the keyword: *image* followed by an incremented

number. The value is composed by an image name, an X and Y position and finally a transparency level.

```
Image_name, X pos, Y pos, Transparency [, width, height]
```

Image_name can be one of the following:

- Stored in the jar file: /image.gif
- Stored on the local machine drive : c:/image.jpg
- Stored on an internet URL: http://..../image.jpg

X pos and **Y** pos are expressed in pixel coordinates. Transparency level must be a real value between 0 and 1. You can set the **width** and the **height** to enforce the image to fit into the desired size.

Examples:

```
Image1: /image1.gif,1,1,.5
Image2: c:/images/image2.jpg,100,1,.5
Image3: /image4.jpg,100,1,.5,200,2
...
```

Tags for the block title

```
.tableTitle {
      type:title
      font-family:Arial
      font-size:16
      font-weight:bold
      line-color: r0g255b0
      line-dash: 16,8
      line-cap: square
      line-join: miter
      line-transparency:.8
      shadow-line-color: r200g200b200
      shadow-line-transparency:.8
      line-width:4
      text-transparency:.8
      text-color:none
      text-gradient: r0g255b0,r0g100b0
      text-align: center
      text-line-Hoffset: 0
      text-line-Voffset: 8
      title-head-offset: -15
```

font-family

Indicates the font name used to draw the title

font-size

Indicates the font size in points

font-weight

Can take one of the following values:

- Plain
- Bold
- Italic
- Bolditalic

line-color

Indicates the RGB color of the main separation line

line-cap

Indicates the kind of end of line used

- Square (default)
- Round
- Butt

line_join

Indicates the kind of line join used:

- Bevel
- Round
- Miter (default)

line-dash

Allows indicating the dashed values. A comma separates the two integer values. The first integer indicates the number of visible pixels. The second integer indicates the number of invisible pixels (gap).

For example, **10,10** will draw the following dashed line: -- -- -- -- -- -- **20,10** will draw : --- --- ----

line-transparency

Indicates the transparency level (real value between 0 and 1)

line-width

Indicates the width in pixel of the line

text-transparency

Indicates the title's transparency level (real value between 0 and 1)

text-color

Indicates the title's RGB value (none or missing if you use a gradient title)

text-gradient

Gives the two colors of the text gradient

text-align

Can take one of the following values:

- Left
- Center
- Right

Tags for the block table header

```
.tableHeader1 {
      type:header
      font-family: Arial;
      font-size:12
      font-weight:bold
      frame-color:r135q55b255
      inside-color:r243g236b255
      font-color:r0g0b255
      font-gradient: r0g0b255,r155g50b255
      frame-gradient: r255g0b0,r255g255b255
      frame-gradient-direction: UpToDown
      shade-color:r160g160b160
      frame-width:1
      frame-rounded-border: 10
      transparency:.8
      text-align:left
      text-align-offset:5
```

font-family

Indicates the font name used to draw the title

font-size

Indicates the font size in points

font-weight

Can take one of the following values:

- Plain
- Bold
- Italic
- Bolditalic

frame-color

Indicates the color of the frame (draw)

inside-color

Indicates the color inside the frame (fill)

font-color

Indicates the header title font color

font-gradient

Indicates the header title font gradient colors

frame-gradient

Indicates the frame inside gradient colors

frame-gradient-direction

Indicates the gradient direction:

- LeftToRight (default)
- UpToDown

shade-color

Indicates the color of the shade behind the frame

frame-width

Indicates the frame width in pixels

frame-rounded-border

Allows to have rounded frames. The value must be between 0 and 25.

transparency

Indicates the frame inside color transparency

text-align can be:

- Left
- Centre
- Right

text-align-offset

Indicates the offset between the cell border and the text

Tags for the block table body

```
.tableBody1 {
    type:body
    font-family:Arial
    font-size:8
    font-weight:bold
    frame-color:r135g55b255
    inside-color:r243g236b255
    shade-color:r160g160b160
    frame-width:1
    frame-rounded-border: 10
    transparency:.8
    odd-foreground-color:r128g128b255
    even-foreground-color:r240g225b255
    odd-background-color:r240g225b255
    even-background-color:r255g255b255
}
```

font-family

Indicates the font name used to draw the title

font-size

Indicates the font size in points

font-weight

Can take one of the following values:

- Plain
- Bold
- Italic
- Bolditalic

frame-color

Indicates the color of the frame (draw)

inside-color

Indicates the color inside the frame (fill)

frame-width

Indicate the frame width in pixels

frame-rounded-border

Allows to have rounded frames. The value must be between 0 and 25.

transparency

Indicates the frame inside color transparency

odd-foreground-color

Indicates the foreground color for odd lines

even-foreground-color

Indicates the foreground color for even lines

odd-background-color

Indicates the background color for odd lines

even-background-color

Indicates the background color for even lines

THE SAMPLE DIALOG TO UPDATE THE CSS FILE

A sample dialog is also provided for you to open any Forms dedicated CSS file, set the decoration setting and save the changes to disk.

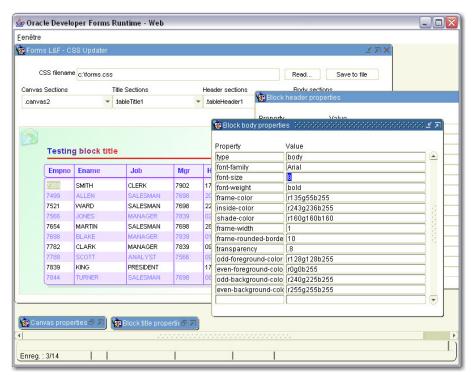


Figure 7 – the css_updater.fmb dialog

Since a CSS file was opened, you can select any section to modify their properties and see immediately the result.

As seen in the previous section, every section has a type tag that determines the kind of Forms object used (canvas, title, header, body).

This tags are read from the **css_updater.fmb** dialog to populate the four properties windows.

SUMMARY

The solution explained in this paper can be downloaded from the Oracle Forms section on OTN. It enables Oracle Forms developers to separate the look of the form dialogs from the functional design. All the decoration elements are read from an external CSS file.

ABOUT THE AUTHOR:

François Degrelle works as a consultant for a French SSII company and is an Oracle specialist (PL/ SQL, Developer, Designer) who likes to share his Forms expertise, writing technical papers about Oracle DB, PLSQL and Forms.

For questions regarding the sample code, please contact François at f.degrelle@free.fr.



Oracle Forms 10*g* – Dynamic Application Color Customization Seotember 2005 Contributing Authors: Grant Ronald (content review)

Oracle Corporation

Copyright © 2005, Oracle. All rights reserved.

Redwood Shores, CA 94065

World Headquarters 500 Oracle Parkway

U.S.A.

Worldwide Inquiries: Phone: +1.650.506.7000 Fax: +1.650.506.7200 www.oracle.com

This document is provided for information purposes only and the contents hereof are subject to change without notice.

This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or

transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its

affiliates. Other names may be trademarks of their respective owners.