Using the TMS form-wide or application-wide styler components

Over the past couple of years, appearances as these from Windows XP, Office 2003, Visual Studio .NET and now Office 2007/2010 have set standards for how an application should look. The techniques for defining the appearance of controls have also become more complex. Before Windows XP, controls were generally flat with single colors for normal state, hover state, checked or down state... With Windows XP, Office 2003, simple gradients for the different control states have become the norm. Now with Office 2007/2010, appearance control has become yet more complex. Controls can now have multiple radial/linear gradients and have subtle transitioning of appearance from state to state. To be able to have full control & flexibility over the appearance of controls, the number of properties has been steadily increasing over the years.

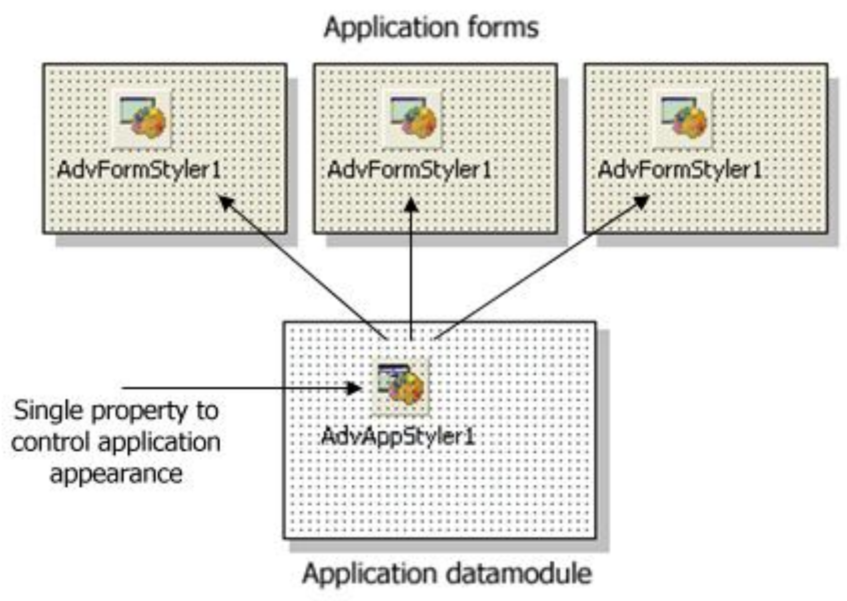
At TMS software we have always tried and keep trying to provide components with the latest look & feel as well as keeping backwards compatibility. We want to enable you to create applications with a brand new appearance and simultaneously do not want to break the look & feel of applications that were created perhaps years ago when installing new updates of the components. As the requirements for appearance control have evolved over the years, so have the properties to control look and feel of the TMS components. It has not always been easy to apply new appearance types to existing controls that were originally not designed for more complex appearances. This has caused that various components have sometimes different ways to control the latest and greatest appearance, let alone to synchronously change appearance of multiple controls to a new consistent style.

We have spent quite some effort to simplify form-wide and application-wide appearance control for TMS components and possibly also your custom controls. To do this, two new components have been designed:

TAdvFormStyler
TAdvAppStyler

A TAdvFormStyler is supposed to be dropped on a form. In a way, it will control just like TXPManifest the style of the TMS controls on the form. A TAdvFormStyler will only affect controls on the form itself. For application-wide appearance control, in addition to a TAdvFormStyler on a form, a TAdvAppStyler component can be dropped on a datamodule and is connected to the TAdvFormStyler components on the forms. By setting then a single property in TAdvAppStyler on the datamodule, the complete application appearance can change, both at design time but also dynamically at runtime.

Scenario



The component TAdvFormStyler has a property style. Setting this style property causes all components on the form that support the interface to set the style to change to the selected style. Similary, setting the style property for the TAdvAppStyler on a central datamodule invokes all TAdvFormStyler style changes and thus the style of all TMS controls on the form. Currently the TAdvFormStyler, TAdvAppStyler support following styles:

- tsOffice2003Blue : Office 2003 style on a blue XP theme
- tsOffice2003Silver : Office 2003 style on a silver XP theme
- tsOffice2003Olive : Office 2003 style on an olive XP theme
- tsOffice2003Classic : Office 2003 style on a non themed XP or pre XP operating system
- tsOffice2007Luna : Office 2007 Luna style
- tsOffice2007Obsidian : Office 2007 Obsidian style
- tsOffice2007Silver: Office 2007 Silver style
- tsWindowsXP : Windows XP / Office XP style
- tsWhidbey : Visual Studio 2005 style
- tsCustom : unforced style
- tsWindowsVista: Windows Vista style
- tsWindows7 : Windows 7 style
- tsTerminal : reduced color set for use with terminal servers
- tsOffice2010Blue : Office 2010 Blue style
- tsOffice2010Silver : Office 2010 Silver style
- tsOffice2010Black : Office 2010 Black style

Following TMS controls are currently TAdvFormStyler, TAdvAppStyler aware:

- TAdvAlertWindow
- TAdvCardList
- TAdvDockPanel
- TAdvInputTaskDialogEx
- TAdvMainMenu & TAdvPopupMenu (via TAdvMenuStyler)
- TAdvMemo

- TAdvNavBar
- TAdvOfficePager (via TAdvOfficePagerOfficeStyle)
- TAdvOutlookList
- TAdvPanel (via TAdvPanelStyler)
- TAdvSmoothButton
- TAdvSmoothCalculator
- TAdvSmoothCalendar
- TAdvSmoothComboBox
- TAdvSmoothDatePicker
- TAdvSmoothDock
- TAdvSmoothExpanderButtonPanel
- TAdvSmoothExpanderGroup
- TAdvSmoothGauge
- TAdvSmoothImageListBox
- TAdvSmoothListbox
- TAdvSmoothMegaMenu
- TAdvSmoothMenu
- TAdvSmoothMessageDialog
- TAdvSmoothPanel
- TAdvSmoothProgressBar
- TAdvSmoothScrollBar
- TAdvSmoothSlider
- TAdvSmoothSlideShow
- TAdvSmoothSpinner
- TAdvSmoothSplashScreen
- TAdvSmoothTabPager
- TAdvSmoothTimeLine
- TAdvSmoothToggleButton
- TAdvSmoothTrackBar
- TAdvSmoothTouchKeyboard
- TAdvStringGrid & descending controls
- TAdvTaskDialogEx
- TAdvToolBar (via TAdvToolBarStyler)
- TAdvToolBarPager (via TAdvToolBarStyler)
- TAdvToolButton & TAdvRepeatButton
- TAdvToolPanel, TAdvToolPanelTab
- TInspectorBar & descending controls
- TPlanner & descending controls
- TPlannerCalendar
- TPlannerDatePicker
- TPlannerMonthView
- TTodoList & descending controls

You can make your own controls also easily TAdvFormStyler, TAdvAppStyler aware so that your controls also automatically change their appearance when the application and/or form style changes. To do this, it is sufficient to add and implement the ITMSStyle interface to your control. This code snippet shows a sample custom control that was made TMS style aware:

```
interface

uses
  Classes, TAdvStyleIF;
type
  TMyCustomControl = class(TCustomControl, ITMSStyle)
  public
    procedure SetComponentStyle(AStyle: TTMSStyle);
  end;
```

```
{ TMyCustomControl }

procedure TMyCustomControl.SetComponentStyle(AStyle: TTMSStyle);
begin
  case AStyle of
  tsOffice2003Blue: // set properties correct here for the selected style
  tsOffice2003Silver:
  tsOffice2003Olive:
  tsOffice2003Classic:
  ...
  end;
end;
```

How to have 2 different types of styles in one form?

By default, TAdvFormStyler will assign the style configured to all components on the form automatically. You can however override this behaviour and implement AdvFormStyler.OnApplyStyle. This event is triggered for every component for which it will set the style. With setting the Allow parameter to false, the style will not be applied.

We hope these new style control components will simplify & accelerate the programmers job to create a consistent user interface and make it easier than ever to switch your complete application appearance by setting a single property.