Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio

Theming OTA Custom Messages

By David | November 7, 2018 0 Comment

Overview

A number of my RAD Studio plug-ins use custom message to output information and most of them work with a themed IDE however one did not and I was wondering why. This article is about what I found and how I fixed things (Note: At this moment in time none of them had been specifically coded for IDE theming).

Debugging Stupid Assumptions

I must admit that in the initial instance I through that the IDE was broken as I've had quite a few issues with theming my plug-in BUT in the end it was a stupid assumption in my code.

What I found was the following code in the custom messages `Draw()` method:

```
Procedure TITHCustomMessage.Draw(Canvas: TCanvas; Const Rect: TRect; Wrap: Boolean);
//FI:O804

Var
  R: TRect;
  strMsg: String;

Begin
  If Canvas.Brush.Color = clWindow Then
    Begin
      Canvas.Font.Color := FForeColour;
      Canvas.Brush.Color := FBackColour;
      Canvas.FillRect(Rect);
    End;
  R := Rect;
  strMsg := FMsg;
  Canvas.Font.Name := FFontName;
  Canvas.Font.Style := FStyle;
```

```
  Canvas.TextRect(R, strMsg, [tfLeft, tfVerticalCenter, tfEndEllipsis]);
End;
```

The problem is on the first line as it assumes that the background colour of the message view is clWindow which theming will change to the window colour of the theme. God only know why I wrote this in the first place but its worked for many years until now 😕

Implementing Theming

So I decided to start again with the code to ensure it understood theming and the first thing I did was make the custom message aware of theming by declaring a new field for the style services as follows:

```
Type
  (** This class defined a custom message for the IDE. **)
  TITHCustomMessage = Class(TInterfacedObject, IUnknown, IOTACustomMessage,
INTACustomDrawMessage,
    IITHCustomMessage)
  Strict Private
    ...
    {$IFDEF DXE102}
    FStyleServices : TCustomStyleServices;
    {$ENDIF}
  Strict Protected
    ...
  Public
    ...
  End;
```

I've not shown the entire code, just the additional lines. If you're not familiar with custom messages you might want to read Chapter 6: Open Tools API Custom Messages first. I've `IFDEF` ed this as we do not want this in earlier versions of the IDE. You will also have to add `[VCL.]Themes` to your `Uses` clause.

The next thing to do was update the constructor to initialise this new variable if theming is enabled and available. So the constructor has the following additional code:

```
Constructor TITHCustomMessage.Create(Const strMsg: String; Const FontName: String;
  Const ForeColour: TColor = clNone; Const Style: TFontStyles = [];
  Const BackColour: TColor = clNone);

...

Begin
  ...
  {$IFDEF DXE102}
  FStyleServices := Nil;
  If Supports(BorlandIDEServices, IOTAIDEThemingServices, ITS) Then
    If ITS.IDEThemingEnabled Then
```

```
      FStyleServices := ITS.StyleServices;
  {$ENDIF}
End;
```

Again this is `IFDEF` ed for earlier versions of the IDE. I also initialise the style services field to `Nil` as we will use this as the test later on as to whether theming is enabled – it will save us calling the `IOTAIDEThemingServices` interface repeatedly which is important for custom drawing code performance.

Now the new drawing code which only resembles the original slightly.

The first thing to note about custom messages is that the OTA interfaces for custom messages (`IOTACustomMessage` and `INTACustomDrawMessage`) do not provide any information on whether the message is selected in the message view or not so we need to determine that first as follows:

```
Procedure TITHCustomMessage.Draw(Canvas: TCanvas; Const Rect: TRect; Wrap: Boolean);
//FI:O804

Var
  ...
  iHighlightColour: TColor;
  boolIsSelected: Boolean;

Begin
  // Determine if the message is selected
  iHighlightColour := clHighlight;
  {$IFDEF DXE102}
  If Assigned(FStyleServices) Then
    iHighlightColour := FStyleServices.GetSystemColor(clHighlight);
  {$ENDIF}
  boolIsSelected := Canvas.Brush.Color = iHighlightColour;
  ...
End;
```

This tests the current canvas brush colour to see if its is the **Highlight** background colour and sets `boolIsSelected` accordingly.

Now we know whether the message is selected we can choose to update the background and font colours. In the following code I do not change the fore or background colours of a selected message else a custom font colour might not show up on a themed selected background.

```
Procedure TITHCustomMessage.Draw(Canvas: TCanvas; Const Rect: TRect; Wrap: Boolean);
//FI:O804

Var
  R: TRect;
  strMsg: String;
  ...
```

```
Begin
  ...
  // Draw background
  If Not boolIsSelected Then
    Begin
      Canvas.Brush.Color := FBackColour;
      If Canvas.Brush.Color = clNone Then
        Canvas.Brush.Color := clWindow;
      {$IFDEF DXE102}
      If Assigned(FStyleServices) Then
        Canvas.Brush.Color := FStyleServices.GetSystemColor(Canvas.Brush.Color);
      {$ENDIF}
    End;
  Canvas.FillRect(Rect);
  // Draw text
  If Not boolIsSelected Then
    Begin
      Canvas.Font.Color := FForeColour;
      If Canvas.Font.Color = clNone Then
        Canvas.Font.Color := clWindowText;
      {$IFDEF DXE102}
      If Assigned(FStyleServices) Then
        Canvas.Font.Color := FStyleServices.GetSystemColor(Canvas.Font.Color);
      {$ENDIF}
    End;
  R := Rect;
  strMsg := FMsg;
  Canvas.Font.Name := FFontName;
  Canvas.Font.Style := FStyle;
  Canvas.TextRect(R, strMsg, [tfLeft, tfVerticalCenter, tfEndEllipsis]);
End;
```

Again this code uses `IFDEF` ed selection for the theming support but it also checks to see if the fore and background colours are set to `clNone`. The constructor of the message has default colour parameters so you don't have to specify them if you do not want to. If `clNone` is found the code sets the fore and background colours of the message to `clWindowText` and `clWindow` respectively.

After all of this I have my custom messages back. All I need to do now is implement similar code in the other IDE plug-ins.

After Thoughts

This code is part of my Integrated Testing Helper project but only existing on my GitHub site (https://github.com/DGH2112/Integrated-Testing-Helper) at the moment as there is still lots to do to fix this plug-in, not as a consequence of theming but because its old and needs updating.

regards

Dave

Category: Delphi Integrated Testing helper Open Tools API RAD Studio Tags: INTACustomDrawMessage, IOTACustomMessage