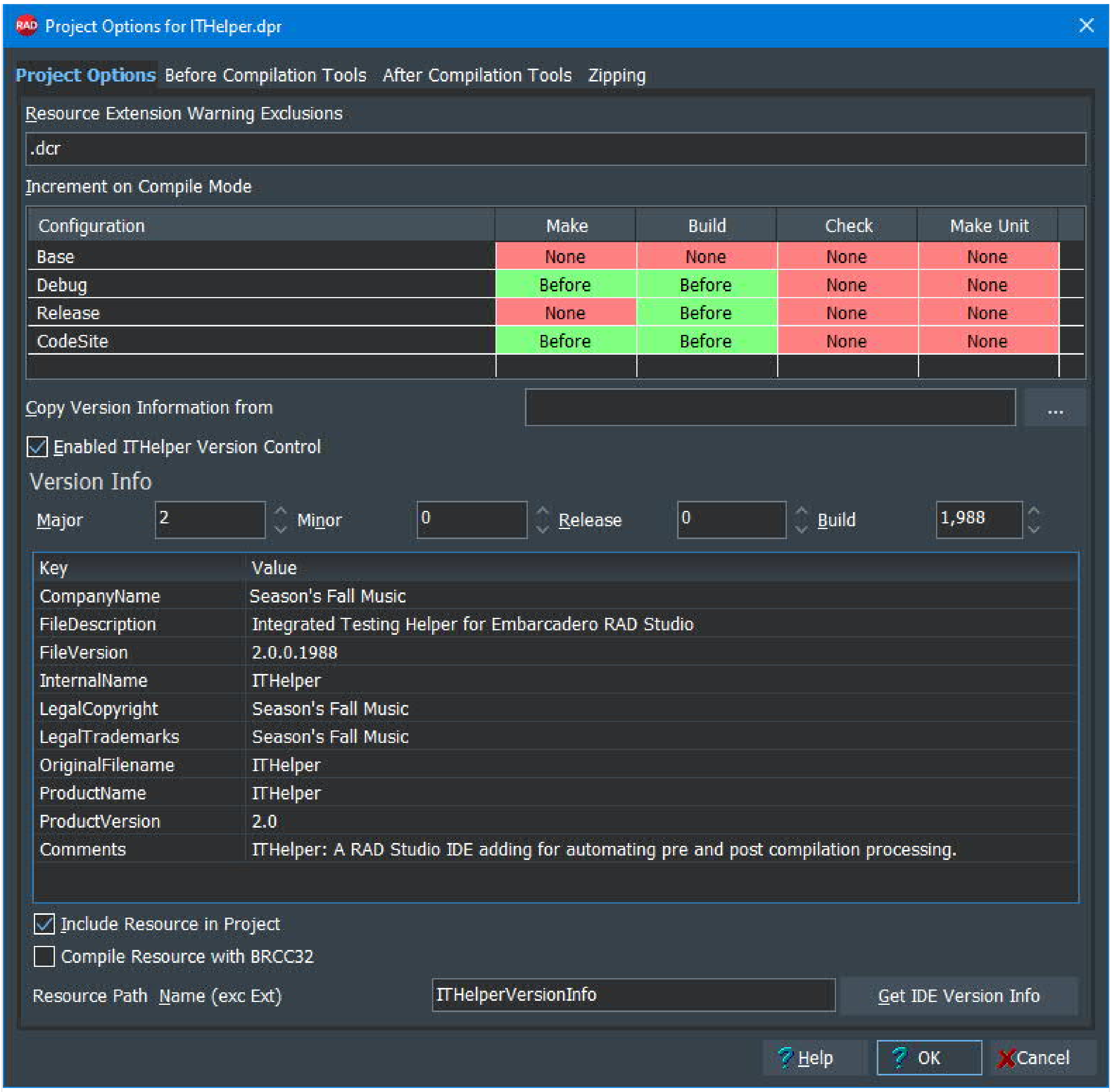
Dave's Development Blog

Software Development using Borland / Codegear / Embarcadero RAD Studio

Theming OTA Forms

By David | September 1, 2020 0 Comment

When I first started to theme my plug-ins in RAD Studio 10.2.1 (I think this was when the dark theme came along), I found it problematic theming the forms. Later on in 10.3.x I revisited theming of forms and came up with the below code which has now been updated for 10.4.x.


Project Options for ITHelper.dpr
Project Options Before Compilation Tools After Compilation Tools Zipping

Resource Extension Warning Exclusions: .dcr

Increment on Compile Mode

Configuration	Make	Build	Check	Make Unit
Base	None	None	None	None
Debug	Before	Before	None	None
Release	None	Before	None	None
CodeSite	Before	Before	None	None

Copy Version Information from

✓ Enabled ITHelper Version Control

Version Info

Major 2 Minor 0 Release 0 Build 1,988

Key	Value
CompanyName	Season's Fall Music
FileDescription	Integrated Testing Helper for Embarcadero RAD Studio
FileVersion	2.0.0.1988
InternalName	ITHelper
LegalCopyright	Season's Fall Music
LegalTrademarks	Season's Fall Music
OriginalFilename	ITHelper
ProductName	ITHelper
ProductVersion	2.0
Comments	ITHelper: A RAD Studio IDE adding for automating pre and post compilation processing.

✓ Include Resource in Project

Compile Resource with BRCC32

Resource Path Name (exc Ext): ITHelperVersionInfo

Now, this is only part of the theming saga and relates to modal forms, modeless forms and dockable forms. Frames that appear in the IDE's options dialogue do not need to use this technique as they will be themed by the IDE however you do need to check the theming of the components you use as there are a number of controls that do not theme in the IDE even though they theming in a standalone application.

First, I'll present the code I've come up with and then explain what it's doing. The code is contained in a record as a static method as below:

```
Class Procedure TBADIToolsAPIFunctions.RegisterFormClassForTheming(Const AFormClass
: TCustomFormClass;
  Const Component : TComponent = Nil);

{$IFDEF DXE102}
Var
  {$IFDEF DXE104} // Breaking change to the Open Tools API - They fixed the wrongly
defined interface
```

```
  ITS : IOTAIDEThemingServices;
  {$ELSE}
  ITS : IOTAIDEThemingServices250;
  {$ENDIF DXE104}
{$ENDIF DXE102}

Begin
  {$IFDEF DXE102}
  {$IFDEF DXE104}
  If Supports(BorlandIDEServices, IOTAIDEThemingServices, ITS) Then
  {$ELSE}
  If Supports(BorlandIDEServices, IOTAIDEThemingServices250, ITS) Then
  {$ENDIF DXE104}
    If ITS.IDEThemingEnabled Then
      Begin
        ITS.RegisterFormClass(AFormClass);
        If Assigned(Component) Then
          ITS.ApplyTheme(Component);
      End;
  {$ENDIF DXE102}
End;
```

The first thing is the use of compiler defines. Theming only appeared in RAD Studio 10.2.1 Tokyo. If I remember correctly 10.2 was not themed and the Open Tools API (`ToolsAPI.pas`) did not contain the new interfaces for theming. I think it was 10.2.1 in which Embarcadero enabled theming in the IDE and added the interfaces in the `ToolsAPI.pas` file. Note: I assume anyone running RAD Studio 10.2.x Tokyo is running at least 10.2.1. From these defines this function is set-up to be empty in IDEs before 10.2 Tokyo.

The second set of compiler defines are to do with a mistake in the original implementation of the theming interfaces in the IDE. When you look at the `ToolsAPI.pas` file, all the latest interfaces (`IOTASomething`) do not end in numbers. If an interface was extended, the existing non-numbered interface was updated to have the previous version number (`IOTASomething123`) and the interface without the number used for the new version. Unfortunately, Embarcadero didn't do this when they created these interfaces but they did fix this issue in 10.4.x Sydney, hence the defines.

Now to the code.

First, we check that the interface is supported. I haven't tested this with 10.2.0, as I do not have virtual machines to check this, but I believe that this check will handle that version missing the interfaces.

If the interface is supported, we check that theming is enabled and if so we register with the IDE the class for the form we want to theme. Now at this point we should not need to do anything else however I've found that not all the components get themed properly (`TLabel` springs to mind), so if an instance of the form is also passed, we apply the theme to the form which themes the components.

As referred to earlier, there are some components that simply do not theme in the IDE, although they do theme in applications. Two that I recall are `TBitBtn` and `TValueListEditor`. The first needs to be replaced with `TButtons` (if you want the icons you need to use a `TImageList`) and the second needs to be manually themed by settings the colours using the `IOTAIDEThemingXxx.StyleServices`.

If your form is modeless or dockable, you will need to register an `IOTAIDEThemingServicesNotifier` so that when the IDE changes themes, you can run this code again to update your form's theme.

Category: Delphi Open Tools API RAD Studio Tags: IOTAIDEThemingServices , IOTAIDEThemingServices250