How to subclass the default Delphi Dock Manager

After a lot of trial and error, I finally figured out how to properly subclass the default Delphi Dock Manager so that you can take control of how the docked controls / forms look. Here are the details.

First, create your own dock manager implementation, inheriting from the Delphi default, for e.g.:

```
uses
  Windows, Graphics, Controls;

type
  TMyDockTree = class(TDockTree)
  protected
    procedure PaintDockFrame(Canvas: TCanvas; Control: TControl;
      const ARect: TRect); override;
  end;
```

In this case, I want to override the drawing behavior, so I will supply my own *PaintDockFrame* routine. But you can override any of the default methods that you want.

OK, now, here is the trick. The control that you want to dock into (like a *TPanel*), needs to have the *UseDockManager* property in the Object Inspector set to *False*. Then, in the *FormCreate* routine, you assign your custom dock manager to the control and *only then* turn the *UseDockManager* property on:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Panel1.DockManager := TMyDockTree.Create( Panel1);
  Panel1.UseDockManager := true;
end;
```

In this case, I'm using a *TPanel* as my dock site. The *DockSite* property of the panel is turned on in the object inspector.

That's it! The trick was turning off *UseDockManager* to avoid Delphi assigning the default dock manager, and turning it on manually in the form creator.

Original resource:	The Delphi Pool
Author:	Mike Potter
Added:	2012-07-08
Last updated:	2012-07-08