

Delphi to Kylix Conversion - the BUG Wizard

by Rob Bracken

We've heard a lot about how easy it is to port an application from Delphi to Kylix, so I was interested to try it for myself. I've described a wizard component – based on a TPageControl – so I decided to try and convert that.

First Steps

The Kylix Developer's Guide has a whole chapter (Chapter 10) on porting Windows applications to Linux. It was a big help with this conversion, and anyone planning to port an application should read it carefully before they start.

When I designed the wizard component, I had Kylix conversion in mind, so I made sure that Kylix had a TPageControl component (the base component for the wizard). This highlights one of the differences between Kylix and Delphi. Components on Delphi's Win 3.1 palette tab weren't ported to Kylix, so if I'd used a TNoteBook instead of a TPageControl, the conversion would have been much more complex, as TNoteBook is one of the Win 3.1 components which weren't ported.

Dangerous Game

Well, I think I've got a good chance of porting the wizard without much work, so I'll do a quick Samba and copy the source files across to my Kylix machine. Crank up Kylix and open the BUGWizard.dpk package. So far so good – the package editor appears, containing the two wizard source files. But what's this? The Requires section contains vcl50.dcp – the Delphi VCL package! As I'm sure you all know, Kylix doesn't have the Windows VCL – it has CLX. I remove it and try to find out what I should replace it with. I couldn't find clear instructions in Kylix help or the Developer's Guide, so I used Kylix to create a new package, to see what it did. It put baseclx.dcp and visualclx.dcp in its Requires section. A quick check in Chapter 10 confirmed that I would need them both in my wizard package, so I put them in (see Listing 1).

Next obvious thing to do was to make sure the uses clauses reference the Kylix equivalents of Delphi units. The BNWizard unit references Windows and Messages – neither of which exist in Kylix – and a number of other units whose names need to be changed. I put a `{$IFDEF WIN32}` round the existing uses clauses, and added new uses clauses, inside `{$IFDEF LINUX}` (see Listing 2). There is a list of Delphi units with their corresponding Kylix names in Chapter 10 of the Developer's Guide. Don't forget that Kylix unit names are case-sensitive.

One unit I couldn't find in Chapter 10's list was dsgnintf. In Delphi, it contains the TComponentEditor class, which I used to create a new right-click menu for use at design time. I checked Kylix help for the TComponentEditor class – it's in the DesignEditors unit. I replaced dsgnintf with DesignEditors and compiled the package. The compile failed, complaining that it couldn't find DesignEditors.dpu. (A .dpu file is created by the compiler when it generates position-independent code – see the section about Inline Assembler Code in Chapter 10). I couldn't find it either, but after

searching around the Kylix files, I discovered a couple of suspicious-looking files called designhooks.dcp and designide.dcp. I added them both to the Requires section (designhooks.dpu requires designide.dpu) - see Listing 1.

Build with Confidence

Compile again. Next problem – can't find the TCaption type, which I used for the extra captions in TBNWizNextFinishAction. Check the help again – I needed to add the QTypes unit to BNWizard's uses clause.

Compile again. Next problem – can't find the RegisterComponentEditor procedure. Check the help – it's in the DesignIntf unit. I add it to BNRRegWizard's uses clause.

Compile again. Final problem – I'd already put a `{$IFDEF WIN32}` clause into the component editor, with a `{$ELSE}` clause. This was to discriminate between 32 bit and 16 bit Windows (not sure why I did this, but I did). This is highlighted in Chapter 10 of the Developer's Guide – it recommends that you don't use a `{$ELSE}` clause in platform-sensitive `{$IFDEF}` clauses. You should put in a separate `{$IFDEF}` clause for each platform. I modified the `{$IFDEF}` clauses, so that it works correctly for 32 bit and 16 bit Windows, and Linux (see Listing 3).

The package now compiled and installed. I created a test form and put a wizard on it with four pages, four buttons and four of the new actions. It all works OK. There is a difference, however – the Delphi version expands the wizard pages to fill the page control, while the Kylix version leaves a space at the top, where the tabs would be if they were visible. Other than this, the wizard works as before.

And now – the acid test. I copied the new units back to the Delphi package, recompiled and installed it. It worked OK, exactly as before.

Different but the Same

It didn't take a lot of work to convert the wizard component so that it now compiles for both Delphi and Kylix. All I had to do was change the names of the units and make sure that the correct packages were included. Much of this is down to the hard work that Borland's developers put in to make sure that the two environments are as compatible as possible. Delphi 6 makes this task even easier by including the CLX components.

Of course, your conversion may not be as simple as this. Don't forget to read Chapter 10 of the Developer's Guide before you start to convert your Windows application.

References

Kylix Developer's Guide – especially Chapter 10. (This is the handbook that comes with the product.)

Listing 1 – the Kylix .dpk file

```
package BUGWizard;

{$R *.res}
{$ALIGN 8}
{$ASSERTIONS ON}
{$BOOLEVAL OFF}
{$DEBUGINFO ON}
{$EXTENDEDSTYNTAX ON}
{$IMPORTEDDATA ON}
{$IOCHECKS ON}
{$LOCALSYMBOLS ON}
{$LONGSTRINGS ON}
{$OPENSTRINGS ON}
{$OPTIMIZATION ON}
{$OVERFLOWCHECKS OFF}
{$RANGECHECKS OFF}
{$REFERENCEINFO ON}
{$SAFEDIVIDE OFF}
{$STACKFRAMES OFF}
{$TYPEDADDRESS OFF}
{$VARSTRINGCHECKS ON}
{$WRITEABLECONST ON}
{$MINENUMSIZE 1}
{$IMAGEBASE $40000}
{$DESCRIPTION 'Wizard components for BUG
                newsletter article'}

{$SOVERSION '1'}
{$IMPLICITBUILD OFF}
requires
    baseclx,
    visualclx,
    designhooks,
    designide;

contains
    BNWizardReg in 'BNWizardReg.pas',
    BNWizard in 'BNWizard.pas';

end
```

Listing 2 – Cross-platform “uses” clauses

```
(from unit BNWizard)

interface

{$IFDEF WIN32}
uses
    Windows, Messages, SysUtils, Classes,
        Graphics, Controls, Forms, Dialogs,
        ComCtrls, ActnList;
{$ENDIF}

{$IFDEF LINUX}
uses
    SysUtils, Classes, QGraphics, QControls,
        QForms, QDialogs,
        QComCtrls, QActnList, QTypes;
{$ENDIF}

(from unit BNRegWizard)

interface

{$IFDEF WIN32}
uses
    Classes, dsgnintf;
{$ENDIF}

{$IFDEF LINUX}
uses
    Classes, DesignEditors, DesignIntf;
{$ENDIF}

implementation

{$IFDEF WIN32}
uses
    BNWizard, ActnList, sysutils;
{$ENDIF}

{$IFDEF LINUX}
uses
    BNWizard, QActnList, SysUtils;
{$ENDIF}
```

Listing 3 – cross-platform with 16-bit Windows

```
{$IFDEF WIN32}
CompOwner := Designer.GetRoot;
{$ELSE}
    {$IFDEF LINUX}
CompOwner := Designer.GetRoot;
    {$ELSE}
        // Assume Win16
CompOwner := Designer.Form;
    {$ENDIF}
{$ENDIF}
```