Resource Strings And Internationalisation

by Patrik Wang

It can be important to design your application to be international aware. Delphi 5 will (as far as we know when we go to press) include a degree of internationalisation support, but there are simpler techniques already available to us in Delphi that will be appropriate for some applications. Since version 100 of the Borland Pascal compiler (Delphi 3 and C++Builder 3) we can make use of the built-in resource string compiler constant, which will automatically generate code that uses `LoadString` from the resources in your application.

The VCL for Delphi 3 onwards uses the resource string concept. Here's an example unit:

```
unit UserConsts;
interface
resourcestring
  SHelloWorld = 'Hello world';
  SHelloName = 'Hello %s';
implementation
end.
```

Here is some application code demonstrating its use:

```
ShowMessage(Format(SHelloName),
  [Application.Title]);
ShowMessage(Format(SHelloName),
  [Application->Title]);
```

It is worth noting that the compiler will automatically assign unique resource identifiers backwards from 65,535, when you compile your application.

One great advantage is that you can override previous declarations in any project which uses this unit, so that it is possible to re-declare new strings in your own source code, then the compiler will take care of duplicates and link previous declarations to the new ones. Because of this great feature you are, for example, able to translate only parts of Consts.pas file if you wish. To give a practical example, consider that you are only interested in translating the button messages such as `OK`, `Cancel`, `Yes` and `No`, but not the rest of the VCL library Consts.pas file. This is done by creating a resource string unit such as the one above and then only including the resource identifiers necessary, such as:

```
SOKButton = 'You Bet';
SCancelButton = 'Go Away';
SYesButton = '&Oui';
SNoButton = '&Non';
```

You can call resource string constants using the unit name (eg `Consts.SReadError`) or not (eg `SReadError`). If you do *not* call a resourcestring with its unit name, you can also re-declare the constant in any other unit (the unit does *not* need to be called Consts.pas). This is why it is possible to re-declare all the VCL library strings in your own application in your own language, even though the VCL library has already been compiled.

If you are a component developer it is also very important to use resourcestring constants since your customers can then re-declare them in their own language without the source.

For more information about resource string usage see the Delphi or C++Builder help.

Patrik Wang works for Lingscape Ltd, who produce globalisation tools for developers. Email him at Support@lingscape.com or visit www.lingscape.com