Borland®

Delphi 프로젝트의 Kylix로의 마이그레이션

Bob Swart (a.k.a. Dr. Bob - www.drbob42.com)

목차

서론

이 글은 애플리케이션을 윈도우용 볼랜드 Delphi.5에서 리눅스용 Kylix로 마이그레이션하기 위한 개요와 상세한 설명을 제공합니다. 우리는 우선 윈도우와 리눅스 양쪽 모두를 위한 애플리케이션을 개발할 수 있다는 것이 (그리고 기존의 애플리케이션을 윈도우에서 리눅스로 쉽게 옮기는 것이) 왜 장점인지를 설명하기로 합니다. 그런 다음, 이 툴들과 프로젝트들 사이의 공통된 아키텍처 및 기술을 개략적으로 살펴볼 것입니다. 여기에는 VCL(Visual Component Library)과 CLX(Component Library for Cross-platform development) 사이의 차이도 포함됩니다. 이 글에서는 또한 개발자들이 윈도우에서 리눅스로 코드를 옮기고 두 플랫폼 사이에서 단일 소스 크로스플랫폼 애플리케이션을 유지하기 위해 개발자들이 새겨두어야 할 주제들과 지켜야 할 가장 좋은 습관들에 대하여 다룹니다.

Kylix™

white paper

Delphi 프로젝트를 Kylix로 마이그레이션

이 화이트페이퍼는 Delphi 5와 Kylix 사이에서 공유되는 크로스플랫폼 코드에 대한 것이 아니라 기존의 Delphi 5 (VCL) 프로젝트, 컴포넌트 및 소스 코드를 리눅스 상의 Kylix (CLX)로 옮기는 것에 관한 글입니다. 이것은 크로스 플랫폼 코드를 작성하기 위한 중요한 첫 단계입니다. (Kylix를 위해 수정된 프로젝트 소스는 더 이상 Delphi 5에서 작동하지 않겠지만, Delphi 6에서는 완벽하게 작동할 것입니다.)

윈도우를 위한 Delphi (VCL)와 리눅스를 위한 Kylix (CLX) 모두를 사용하여 컴파일할 수 있는 간단한 단일 소스 애플리케이션을 작성하는 방법을 살펴보기로 합니다. (가급적 IFDEF는 사용하지 않을 것입니다) 또한, Delphi 5에는 있지만 Kylix에서는 지원되지 않는 기능들을 알아보고, 그 대안을 제시해보도록 하겠습니다.

왜 마이그레이션하는가?

Kylix를 사용하여 Delphi 5 프로젝트를 윈도우에서 리눅스로 옮기는 이점을 생각해 봅시다. 컴포넌트들과 소스 코드 부분들을 공유할 수 있는 것 이외에 가장 큰 이점은, 기존의 코드를 가지고 완전히 새로운 전체 시장을 뚫을 수 있다는 것입니다. 바퀴를 새로 발명할 필요가 없는 것입니다. (타이어에 바람은 새로 넣어야 할 지는 몰라도.)

예를 들어, 필자는 기존의 많은 웹 서버 애플리케이션들을 (Delphi 5 WebBroker 기술을 사용하여) 윈도우에서 실행되는 마이크로소프트 IIS (Internet Information Server)로부터 리눅스에서 실행되는 Apache Shared Object (DSO)로 각각 한 시간 이내에 옮길 수 있었습니다. 이것은 더 이상 내가 윈도우 NT나 윈도우 2000을 구동하는 웹 서버에 한정되지 않는다는 것을 의미합니다. 나는 (리눅스가 특히 강한 영역인) 리눅스 웹 서버에 배포할 수 있습니다. 물론 CGI 웹 서버 애플리케이션에 대해서도 마찬가지입니다. 나중에 알게 되겠지만, 사실 이것이 더 쉽습니다.

프로젝트 마이그레이션

Delphi 5 프로젝트를 Kylix로 옮길 때 첫째 목표는 Delphi 5 프로젝트를 윈도우 머신으로부터 리눅스 머신으로 물리적으로 이동시키는 것입니다. 이것은 리눅스가 파일 이름에서 대소문자를 구분하는 반면 윈도우는 그렇지 않기 때문에 반드시 이루어져야 합니다. 윈도우에서는 Project1.dpr과 PROJECT1.DPR이 모두 같은 파일을 가리키지만, 리눅스에서는 별개의 파일입니다. 프로젝트와 유닛의 이름들도 (프로젝트 및 유닛 소스 파일의 첫 라인에서 참조되는 것과) 실제 파일 이름과 같아야 합니다. 그리고, 각 유닛의 uses 절에 있는 유닛 목록을 고려하면 대소문자 구분은 더욱 중요합니다. 만일 유닛의 이름이 "Myunit"이고 디스크에 Myunit.pas로 저장되어 있는 경우, uses 절에서도 MyUnit이 아닌 Myunit으로 지정해야 합니다. 왜냐하면 MyUnit.pas 파일에서 MyUnit이라는 이름의 유닛은 찾을 수 없기 때문입니다. 문제를 피하는 방법으로서 메인 프로젝트 소스 파일의 uses 절에서 유닛이 들어 있는 파일 이름을 각 유닛에 명시적으로 추가할 수 있습니다. 예를 들면 다음과 같습니다.

```
uses
  MyUnit in "Myunit.pas",
  unit1 in "Unit1.pas";
```

"in" 절은 메인 프로젝트 소스 파일에만 나타나고 유닛 소스 파일에는 나타나지 않는 것에 주의하십시오. 이것은 사소한 것으로 보일 수도 있으나, 적어도 컴파일러가 프로젝트 유닛을 모두 발견할 수 있도록 모든 것의 이름을 올바로 다시 정하는 데에는 시간이 걸릴 수 있습니다.

각각에 대해 말하자면, Delphi 5 프로젝트는 .dpr (프로젝트 소스 코드) 파일, .res (아이콘 리소스) 파일, .opt (프로젝트 옵션) 파일, .cfg (커맨드 라인 옵션) 파일 및 일단의 .pas 파일들과 이에 대응하는 .dfm 파일들로 구성되어 있습니다. Kylix 프로젝트는 .dpr (프로젝트 소스 코드) 파일, .res 파일, .conf (환경설정) 파일, .kof (Kylix 옵션 파일) 및 일단의 .pas 파일들과 이에 대응하는 .xfm 파일들로 구성되어 있습니다. Delphi 5 .cfg 파일은 Kylix .conf 파일에 해당하며, Delphi .opt 파일은 Kylix .kof 파일에 해당합니다. 내부적으로 이러한 파일들은 조금 다르며 (일부 컴파일러 옵션은 Kylix 또는 윈도우에서 의미가 없고, 디렉토리는 리눅스의 경우 드라이브 문자 대신에 슬래시로 지정되고 윈도우의 경우에는 백슬래시로 지정), 그래서 새로운 파일 확장자가 생긴 것입니다. 수작업으로 파일을 변환하거나 새로운 빈 프로젝트를 시작하여 유닛과 기타 소스 코드를 옮기는 것이 가장 좋습니다. 필자는 후자를 선호합니다 (특히 WebBroker 프로젝트에 있어서).

.xfm 파일은 단순히 .dfm 파일입니다. .dfm 파일을 .xfm 파일로 변환하기 위한 가장 쉽고도 믿을 수 있는 방법은 .dfm 파일이 텍스트로 저장되도록 하는 것입니다. Delphi에서 텍스트로 저장하거나, Delphi와 함께 제공되는 convert.exe 유틸리티를 사용하여 텍스트로 변환합니다. 그런 후 이름을 .xfm 파일로 바꿉니다. 따라서, .dfm 파일이 이미 이진 파일이 아닌 텍스트로 저장되어 있다면 .dfm에서 .xfm로 이름만 바꾸면 됩니다.

단일 소스

Delphi 및 Kylix와 컴파일되는 프로젝트의 단일 소스를 유지하려면 최소한 Delphi가 .dfm 파일을 사용하고 Kylix가 .xfm 파일을 사용하는 것을 확인해야 합니다. 이것은 Delphi의 {$R *.DFM} 문장과 Kylix의 {$R *.xfm} 문장이 처리합니다. 단일 소스의 코드는 IFDEF를 이용하여 작성할 수 있는데, 윈도우는 컨디셔널 심볼 MSWINDOWS로 식별되고 리눅스는 LINUX로 식별됩니다.

```
{$IFDEF MSWINDOWS}
  {$R *.DFM}
{$ENDIF}
{$IFDEF LINUX}
  {$R *.xfm}
{$ENDIF}
```

절대로 {$ELSE} 디렉티브를 사용하거나, 윈도우가 아니라면 리눅스라거나 그 반대로 생각해서는 안됩니다. 미래에 출시될 Delphi나 Kylix 버전이 전혀 새로운 운영 시스템에서 실행되는 것이 불가능한 것도 아니고, 그 때에는 {$ELSE}가 잘못된 결과를 가져올 수 있습니다. 또한, WIN32 대신에 MSWINDOWS를 사용하는 것을 잊지 말아야 합니다. 그렇지 않으면, 64비트 버전의 윈도우가 출시되었을 때 코드가 깨질 수 있습니다.

Kylix에 대하여 정의된 또 하나의 유용한 조건 기호는 VER140이며, 이것은 Kylix가 Pascal 컴파일러의 버전 14인 것을 가리킵니다. (이와 비교해서, Turbo Pascal은 버전 1이고, Delphi 1.0x 는 VER80이 정의된 버전 8이며, Delphi 5는 VER130이 정의된 버전 13입니다.)

이식 불가능한 프로젝트

모든 Delphi 5 프로젝트를 다 Kylix로 옮길 수 있는 것은 아닙니다. 특히, ActiveX 라이브러리는 리눅스로 이식할 수 없습니다 (COM 객체, ActiveX 컨트롤, ADO, ActiveForm, Active Server Pages). 놀랄 일은 아니지만, 단일 소스 크로스 플랫폼 솔루션에 ActiveForm을 이용하는 데 시간과 금전을 투자하기 전에 이에 대해서 알아 두는 것이 좋습니다.

위에서 언급한 마이그레이션을 생각할 때 일반적인 윈도우 GUI 애플리케이션은 리눅스로 이식하기 위한 좋은 대상입니다.

WebBroker 대상

Delphi 5 WebBroker 구조는 CGI, WinCGI 및 ISAPI/NSAPI 웹 서버 애플리케이션을 지원합니다. NSAPI 지원은 ISAPI DLL과 동일한 코드에 기반을 둔 WebBridge에 의해서 처리됩니다. 리눅스는 ISAPI(WebBridge를 통한 NSAPI도 포함)도 WinCGI도 지원하지 않습니다. 그렇다고 해서 CGI WebBroker 애플리케이션이 아닌 것은 모두 리눅스로 옮길 수 없다는 것은 아닙니다. WebBroker의 좋은 점 가운데 하나는 웹 서버 애플리케이션의 핵심 기능이 웹 모듈내에 포함되어 있다는 사실입니다. 웹 모듈은 애플리케이션의 대상 플랫폼과 관계가 없습니다. 실제로, 표준 CGI WebBroker 애플리케이션에 있어서도 웹 모듈만을 리눅스로 이식하고 Kylix를 사용하여 새로운 WebBroker 애플리케이션을 생성하고자 할 수도 있습니다. (CGI 또는 Apache 공유 객체 중에서 선택할 수 있습니다) 그런 후에 웹 모듈을 리눅스 상에서 새로운 원시 WebBroker 애플리케이션에 추가할 수 있습니다. 단 몇 분 간의 작업만으로, WebBroker 애플리케이션의 골격이 작동하게 될 것입니다.

DB 데이터를 리눅스로 마이그레이션

Delphi 5는 여러 가지 데이터베이스와 테이블 포맷을 지원합니다. BDE (dBASE, Paradox) InterBase (네이티브 및 BDE를 통한), ODBC/ADO (예: Access, FoxPro 등) 및 Oracle, DB2, Informix와 SQL Server 등입니다. Kylix는 BDE를 지원하지 않지만, 네이티브 리눅스 데이터베이스 엔진에 대한 개방적인 액세스를 위하여 dbExpress로 바꾸었습니다. Kylix Desktop Developer는 MyBase (로컬 XML), MySQL 및 InterBase를 지원하며, Kylix Server Developer는 Oracle와 DB2에 대한 지원이 추가되어 있습니다.

따라서, Delphi 5에서 사용할 수 있는 모든 데이터베이스 포맷이 Kylix에서 사용할 수 있는 것은 아닙니다. 특히, Borland Database Engine (dBase, Paradox 포맷)은 리눅스로 이식되지 않습니다. 이것은 dBase와 Paradox 테이블을 Kylix에서 사용할 수 없다는 것을 의미합니다.

다행히, Delphi 5를 사용하여 dBase와 Paradox 테이블을 InterBase 또는 로컬 ClientDataSet 포맷(CDS 또는 XML)으로 옮길 수 있습니다. XML 포맷의 데이터는 리눅스를 위한 MyBase 개인 XML 데이터베이스로 사용할 수 있습니다.

Delphi 5 Enterprise 이후 버전을 사용하면 테이블을 DataSetProvider에 연결하여 ClientDataSet 컴포넌트에서 데이터를 .cds나 .XML 포맷으로 저장할 수 있습니다. Delphi 5 Enterprise 이후의 버전가 없는 경우(ClientDataSet 컴포넌트가 없는 경우)에는 데이터셋을 XML로 변환하는 작업을 위해 아래의 소스 코드(iTec의 *The Delphi Magazine*에 실린 필자의 글에 기초함, www.TheDelphiMagazine.com)를 사용할 수 있습니다.

```
unit TableXML;
// Routine DataSetXML converts a DataSet to XML
format
// Author: Bob Swart (a.k.a. Dr. Bob -
www.drbob42.com)
// Available as freeware, but use at your own
risk!
//
interface
uses
  DB;

function DataSetXML(DataSet: TDataSet; const
FileName: String): Integer;

implementation
uses
  SysUtils, TypInfo;

function DataSetXML(DataSet: TDataSet; const
FileName: String): Integer;
var
  F: System.Text;
  i: Integer;

  function Print(Str: String): String;
  { Convert a fieldname to a printable name }
  var
    i: Integer;
  begin
    for i:=Length(Str) downto 1 do
      if not (UpCase(Str[i]) in ['A'..'Z','1'..'9'])then
        Str[i] := '_';
    Result := Str
  end {Print};

  function EnCode(Str: String): String;
  { Convert memo contents to single line XML }
  var
    i: Integer;
  begin
    for i:=Length(Str) downto 1 do
    begin
      if (Ord(Str[i]) < 32) or (Str[i] = '"') then
      begin
        Insert('&#'+IntToStr(Ord(Str[i]))+';',Str,i+1);
        Delete(Str,i,1)
      end
    end;
    Result := Str
  end {EnCode};

begin
  Result := -1;
  ShortDateFormat := 'YYYYMMDD';
  System.Assign(F,FileName);
  try
    System.Rewrite(F);
    writeln(F,'<?xml version="1.0" standalone="yes"?>');
    writeln(F,'<DATAPACKET Version="2.0">');
    with DataSet do
    begin
      writeln(F,'<METADATA>');
      writeln(F,'<FIELDS>');
      if not Active then
        FieldDefs.Update { get info without opening the
        database };
      for i:=0 to Red(FieldDefs.Count) do
      begin
        write(F,'<FIELD ');
        if Print(FieldDefs[i].Name) <> FieldDefs[i].Name
        then { fieldname }
          write(F,'fieldname="',FieldDefs[i].Name,'" ');
        write(F,'attrname="',Print(FieldDefs[i].Name),
              '" fieldtype="');
        case FieldDefs[i].DataType of
          ftString,
       ftFixedChar,
      ftWideString: write(F,'string');
         ftBoolean: write('boolean');
        ftSmallint: write(F,'i2');
         ftInteger,
            ftWord: write(F,'i4');
        ftAutoInc: write(F,
          'i4" readonly="true" SUBTYPE="Autoinc');
          ftFloat: write(F,'r8');
        ftCurrency: write(F, 'r8" SUBTYPE="Money');
             ftBCD: write(F,'fixed');
            ftDate: write(F,'date');
            ftTime: write(F,'time');
        ftDateTime: write(F,'datetime');
           ftBytes: write(F,'bin.hex');
        ftVarBytes,
            ftBlob: write(F,'bin.hex" SUBTYPE="Binary');
            ftMemo: write(F,'bin.hex" SUBTYPE="Text');
```

```
        ftGraphic,
      ftTypedBinary: write(F,
          'bin.hex" SUBTYPE="Graphics');
        ftFmtMemo: write(F,
          'bin.hex" SUBTYPE="Formatted');
       ftParadoxOle,
        ftDBaseOle: write(F,'bin.hex" SUBTYPE="Ole')
        end;
        if FieldDefs[i].Required then
          write(F,'" required="true');
        if FieldDefs[i].Size > 0 then
          write(F,'" WIDTH="',FieldDefs[i].Size);
        writeln(F,'"/>')
      end;
      writeln(F,'</FIELDS>');
      writeln(F,'</METADATA>');
      if not Active then Open;
      writeln(F,'<ROWDATA>');
      Result := 0;
      while not Eof do
      begin
        Result := Result + 1;
        write(F,'<ROW ');
        for i:=0 to Pred(Fields.Count) do
          if (Fields[i].AsString <> '') and
          ((Fields[i].DisplayText = Fields[i].AsString)
          or (Fields[i].DisplayText = '(MEMO)'))
          then
            write(F,Print(Fields[i].FieldName),'="',
            EnCode(Fields[i].AsString),'"');
        writeln(F,'/>');
        Next
      end;
      writeln(F,'</ROWDATA>')
    end;
    writeln(F,'</DATAPACKET>')
  finally
  System.Close(F)
  end
end;
```

테이블을 변환한 후에는 이를 윈도우에서 Kylix로 이동하고 MyBase를 사용하여 XML 파일을 로딩합니다. 필요하면 MyBase를 사용하여 데이터를 더 옮길 수 있습니다 (예를 들면 Oracle로). 물론 언제나 사용자의 데이터베이스를 리눅스에서 실행되는 InterBase 데이터베이스로 이동할 수 있습니다.

마지막으로, Kylix는 ODBC/ADO (예: Access 및 FoxPro) 또는 SQL Server 및 Informix를 지원하지 않습니다. 이러한 데이터베이스 포맷의 경우에는 InterBase 또는 Oracle로의 마이그레이션을 고려하는 것이 좋습니다.

써드 파티 ODBC 지원

Easysoft (www.easysoft.com/products/kylix/main.phtml)의 dbExpress Gateway for ODBC를 사용하면 Kylix 애플리케이션이 ODBC 데이터 소스를 액세스할 수 있습니다. 이것은 데이터베이스 드라이버(InterBase, Postgres, Oracle 등)와 Easysoft ODBC 브리지 소프트웨어를 통한 원격 데이터베이스를 가진 리눅스 데이터베이스와 작동합니다. 이것은 Kylix 애플리케이션이 Access와 SQL Server 뿐 아니라 윈도우 ODBC 드라이버가 있는 어떠한 데이터베이스와도 작동이 된다는 의미입니다. 인터페이스는 리눅스를 위한 공개 소스 ODBC 드라이버 관리자인 unixODBC와 호환 가능합니다.

dbExpress

Kylix는 크로스플랫폼 데이터 액세스 기술로 dbExpress를 사용합니다. dbExpress가 데이터베이스와 테이블에 액세스하는 방법은 여러분이 BDE, ADO 및 기타 Delphi 데이터베이스 메커니즘을 사용하던 방법과는 차이가 있습니다. 우선, dbExpress는 테이블에 대하여 단방향 커서만을 제공합니다. 이것은 앞으로는 검색할 수 있어도 뒤로는 할 수 없다는 의미입니다. 이것은 효율성을 위해서 고안된 것이지만, 익숙한 방법과는 많이 다릅니다.

dbExpress 데이터 소스를 ClientDataSet에 연결하면 데이터를 앞뒤로 검색할 수 있습니다. 이후 이 소스는 로컬의 서류가방처럼 사용됩니다. dbExpress 드라이버는 InterBase, Oracle, DB2 및 MySQL에 대하여 사용 가능하지만, 직접 dbExpress 드라이버를 작성할 수도 있습니다 (자세한 내용은 볼랜드에 문의).

표 1은 BDE, InterBase Express 및 ADO Express 컴포넌트와 dbExpress 컴포넌트의 대응 관계를 보여줍니다. 대부분의 Delphi 데이터베이스 애플리케이션은 편집과 내비게이션을 하므로, 새로 이식하는 각 dbExpress 컴포넌트에 대하여 DataSetProvider와 ClientDataSet 컴포넌트를 추가할 생각을 해야 합니다. 그러면 애플리케이션을 TDBGrid 또는 TDBEdit와 같은 데이터 인식 컴포넌트에 연결할 수 있습니다. 당분간 이 과정을 도와줄 자동 절차나 도구는 없습니다.

데이터베이스 애플리케이션을 옮기는 마지막 단계는

테이블에서의 수정 내용이 (ClientDataSet 컴포넌트를 사용하여) 원래의 dbExpress 데이터세트로 다시 보내졌는지 확인하는 것입니다. 이 작업은 ClientDataSet의 OnAfterPost 이벤트에서 명시적으로 ClientDataSet. ApplyUpdates 메소드를 호출하여 실행합니다.

```
Procedure TForm1.ClientDataSet1AfterPost(DataSet:
  TDataSet);
begin
  if DataSet IS TDataSet then
  TClientDataSet(DataSet).ApplyUpdates(-1)
end;
```

이 작업은 ClientDataSet에 있는 모든 Post에 대해서 ApplyUpdates를 호출하기 때문에 다소 느릴 수 있습니다. 다른 방법은 사용자의 Form에 "ApplyUpdates" 버튼을 만드는 것입니다 (또는 사용자가 Form을 종료하거나 특정한 타이머 이벤트가 작동할 때 ApplyUpdates를 실행하는 것입니다). 가능성은 여러 가지입니다. 가장 느리기는 해도 가장 안전한 방법은 위에 설명한 것입니다.

ClientDataSet에 관한 추가 정보는 볼랜드 커뮤니티 웹 사이트 http://bdn.borland.com 나 필자의 사이트 www.drbob42.com에 있는 MIDAS 문서를 참조하십시오.

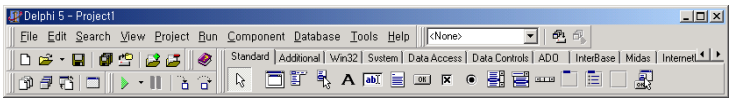
VCL에서 CLX로

Delphi 비주얼 컴포넌트 라이브러리(VCL)에는 비주얼 컴포넌트 이상의 것들이 포함되어 있습니다. 필자가 볼 때 이 수수께끼는 크로스 플랫폼 개발을 위한 컴포넌트 라이브러리(CLX)로 풀 수 있는데, 이것은 네 부분 (BaseCLX,. VisualCLX, DataCLX, NetCLX)으로 나뉘어 있으며 그 중 하나만이 비주얼합니다.

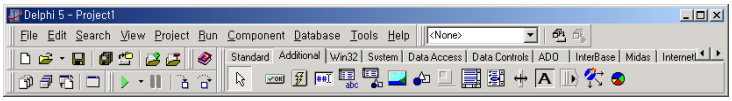
Kylix CLX는 Delphi VCL의 구조와 계층을 유지하고 있는데, Delphi 개발자와 애플리케이션에게는 모양과 느낌이 거의 같습니다. 이면에서 CLX는 Qt 라이브러리로 호출을 하는데, 이것은 윈도우와 리눅스에 모두 사용 가능한 라이브러리입니다. (그래서 Kylix와 Delphi에서 CLX의 크로스 플랫폼 특성을 보장합니다). 그러나, 우선 기존의 Delphi 5 VCL 프로젝트, 특히 Delphi 5 컴포넌트 팔레트에서 사용 가능한 컴포넌트들을 먼저 살펴보고, 어느 것이 새로운 CLX 구조에서 사용 가능할 것인지 (그리고 더 중요하게는 어느 것이 사용할 수 없는지) 확인해야 합니다. 어떤 컴포넌트가 Kylix에서 사용할 수 없거나 지원되지 않는다는 것을 미리 안다면 리눅스로 이동해서 모든 것을 다시 작성할지도 모르는 수고 대신에 Delphi 5에서 미리 대비하는 것이 훨씬 비용이 적게 드는 경우가 많습니다.

사용 가능한 컴포넌트

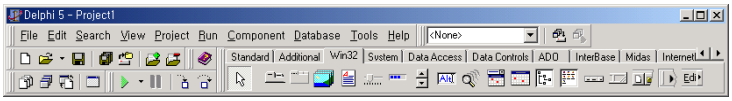
이제 Kylix (Server Developer)에서 사용 가능한 Delphi 5 (Enterprise)의 컴포넌트들을 살펴보기로 합니다. Kylix에서 사용할 수 없는 Delphi 5의 컴포넌트들도 열거함으로써 대안으로서의 써드 파티 컴포넌트나 이러한 컴포넌트를 사용하지 않을 수 있는 방법을 제시합니다. (Delphi 6 이상에서는 CLX 애플리케이션을 열면 오브젝트 인스펙터에 CLX 컴포넌트만 나타나므로 이런 수고를 덜 수 있습니다.)



Delphi 5 컴포넌트 팔레트의 Standard 페이지는 Kylix에도 있는 컴포넌트들만 포함하고 있으므로 모든 것을 이식할 수 있습니다. 각각의 컴포넌트에 있어서는 차이가 있는 속성이나 메소드가 일부 있을 수 있으나 이러한 차이는 보통 쉽게 고칠 수 있습니다.



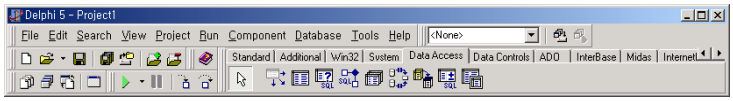
Delphi 5 컴포넌트 팔레트의 Additional 페이지에는 Kylix에는 없는 세 개의 컴포넌트가 포함되어 있는데, 그것은 TStaticText, TApplicationEvents 및 TChart입니다. TStaticText에 대해서는 항상 TLabel을 사용할 수 있습니다 (StaticText과 Label 사이의 차이점은, StaticText은 TWinControl로부터 상속받아 윈도우 핸들을 가지고 있다는 점입니다).



Delphi 5 컴포넌트 팔레트의 Win32 페이지는 Kylix에는 없습니다. 그 대신, Kylix의 컴포넌트 팔레트에는 Common Controls 페이지가 있는데, 여기에는 전부는 아니지만 Win32 탭에 있는 컴포넌트의 대부분이 있습니다. Kylix의 Common Controls 탭에 없는 컴포넌트들은 TRichEdit, TUpDown

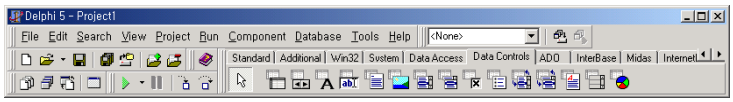
(Delphi에서는 Samples 탭에 있는 TSpinEdit가 있습니다), THotKey, TAnimate, TDateTimePicker, TCalendar, TCoolBar 및 TPageScroller입니다.

Delphi 5 컴포넌트 팔레트의 System 페이지도 역시 Kylix에는 없으며, 그 페이지에 있는 대부분의 컨트롤도 없습니다. 그 중 TTimer와 TPaintBox는 Kylix 컴포넌트 팔레트의 Additional 페이지에 있습니다. 그 밖의 컴포넌트인 TMediaPlayer, TOleContainer, TDdeClientConv, TDdeClientItem, TDdeServerConv 및 TDdeServerItem는 Kylix에 없습니다.

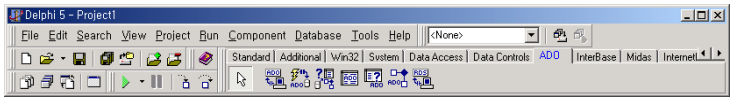


Delphi 5 컴포넌트 팔레트의 Data Access 페이지, 혹은 다른 이름으로 BDE 페이지는 거의 완전히 Kylix에서 사라졌습니다. 오직 TDataSource만이 유일한 컴포넌트로 이 탭에 남아 있습니다. Kylix에 있는 데이터 액세스 페이지에는 TClientDataSet와 TDataSetProvider 컴포넌트도 있습니다 (모두 세 개).

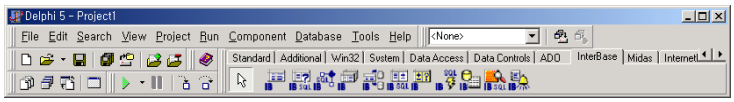
BDE로부터 MyBase 또는 InterBase로의 데이터베이스 데이터의 마이그레이션에 관한 더 자세한 정보는 "DB 데이터를 리눅스로 마이그레이션"을 참조하십시오.



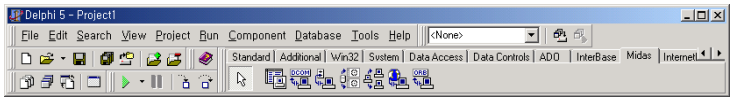
Delphi 5 컴포넌트 팔레트의 Data Control 페이지에는 Kylix의 Data Control 탭에 없는 세 개의 컴포넌트, 즉 TDBRichEdit, TDBCtrlGrid, TDBChart이 있습니다.



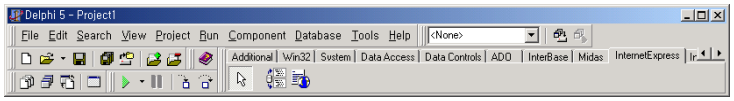
Delphi 5 컴포넌트 팔레트의 ADO 탭은 Kylix에서 완전히 사라져서 전혀 나타나지 않습니다.



Delphi 5 컴포넌트 팔레트의 InterBase 탭 또한 Kylix에 없습니다. Kylix의 dbExpress 탭에 있는 컴포넌트들이 Data Access (BDE), ADO 및 InterBase 탭에 있는 컴포넌트들의 기능을 대신합니다.



Delphi 5 컴포넌트 팔레트의 MIDAS 탭은 Kylix에 없지만 TClientDataSet와 TDataSetprovider 컴포넌트는 Kylix의 데이터 액세스 탭에 있습니다. 다른 컴포넌트들인 TDCOMConnection, TSocketConnection, TSimpleObjectBroker, TwebConnection 및 TCorbaConnection은 Kylix에 없습니다. TDCOMConnection은 Kylix에 절대로 나타나지 않겠지만, 다른 것들은 Kylix의 2 이상 버전의 Enterprise 에디션에는 포함됩니다.

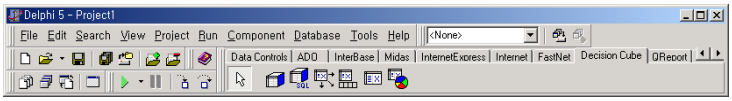


Delphi 5 컴포넌트 팔레트의 InternetExpress 탭은 Kylix에 없는데, 그 주된 이유는 InternetExpress와 MIDAS가 Kylix Server Developer 에디션에서 지원되지 않기 때문입니다. 그러나, 이 탭들은 Kylix 2 이상 버전의 Enterprise 판에는 포함될 예정입니다. MIDAS와 InternetExpress가 없다는 것은 Kylix로 분산 애플리케이션을 작성할 수 없다는 것을 의미합니다.

Delphi 5 컴포넌트 팔레트의 Internet 탭에는 Kylix의 Internet 탭에서 볼 수 없는 컴포넌트를 많이 포함하고 있습니다. 즉, TClientSocket와 TServerSocket (이 두개는 TTcpClient, TTcpServer 및 TUdpSocket로 대체된 것 같습니다.) 및 TWebBrower (이것은 단순히 Internet Explorer ActiveX 컨트롤을 둘러싸는 래퍼입니다) 등입니다. 이 세개의 컴포넌트 외에 Delphi에 포함되어 있는 TQueryTableProducer는 Kylix에서 TSQLQueryTableProducer로

이름이 바뀌었습니다.

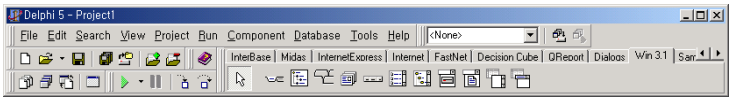
Delphi 5 컴포넌트 팔레트의 FastNet 탭은 세 개의 Indy (Internet direct) 탭, 즉 Indy Clients, Indy Servers 및 Indy Misc로 바뀌었습니다. FastNet을 사용하는 프로젝트에 있어서 전부는 아니더라도 대부분의 코드는 Indy 컴포넌트를 사용하여 다시 작성되어야 할 것입니다.



Delphi 5 컴포넌트 팔레트의 Decision Cube 탭도 Kylix에 없습니다.

Delphi 5 컴포넌트 팔레트의 QReport 탭은 더 이상 Kylix에 없습니다. 리포트 작성을 위해서는 Nevrona의 Rave Reports 등과 같은 써드 파티 지원을 찾아보는 것이 좋습니다.

Delphi 5 컴포넌트 팔레트의 Dialogs 탭에는 Kylix에는 없는 네 개의 컴포넌트, 즉 TOpenPictureDialog, TSavePictureDialog, TprintDialog 및 TprinterSetupDialog가 포함되어 있습니다.



Delphi 5 컴포넌트 팔레트의 Win 3.1 탭에 있는 컴퍼넌트들 중에는 Kylix에 있는 컴포넌트가 하나도 없습니다. 이것은 TTabSet, TOutline, TTabbedNotebook, TNotebook, THeader, TFileListBox, TDirectoryListbox, TDriveComboBox, TFilterComboBox가 없다는 것을 의미합니다.

마지막으로, Delphi 5 컴포넌트 팔레트의 Samples 페이지는 Kylix에 없지만, TSpinEdit는 Kylix의 Common Controls 탭에 있습니다.

TGauge, TColorGrid, TSpinButton, TDirectoryOutline, TCalendar, TIBEventAlerter도 없습니다.

대안 제안

아래의 표는 Kylix에 없는 컴포넌트들(때로는 전체 탭)과 이에 상응하는 Kylix에 포함된 추천 컴포넌트 또는 필요한 기능을 구현하기 위한 대안 방법의 리스트를 보여주고 있습니다.

*표 1

Delphi 5 Component	**Kylix Component**	**설명**
Additional 페이지		
TStaticText	TLabel	
TApplicationEvents		
TChart		TChart
Win32 페이지		
TRichEdit		Win32에 한정된 컴포넌트
TUpDown	TSpinEdit	Win32에 한정된 컴포넌트
THotKey		Win32에 한정된 컴포넌트
TAnimate		Win32에 한정된 컴포넌트
TDateTimePicker		Win32에 한정된 컴포넌트
TCalendar		Win32에 한정된 컴포넌트
TCoolBar		Win32에 한정된 컴포넌트
TPageScroller		Win32에 한정된 컴포넌트
System 페이지		
TMediaPlayer		
TOleContainer		Win32에 한정된 컴포넌트
TDdeClientConv		Win32에 한정된 컴포넌트

TDdeClientItem		Win32에 한정된 컴포넌트
TDdeServerConv		Win32에 한정된 컴포넌트
TDdeServerItem		Win32에 한정된 컴포넌트
Data Access 페이지	dbExpress	
TTable	TSQLTable	DataSetProvider와 ClientDataSet 추가
TQuery	TSQLQuery	DataSetProvider와 ClientDataSet 추가
TStoredProc	TSQLStoredProc	DataSetProvider와 ClientDataSet 추가
TDatabase	TSQLConnection	
TSession		BDE에 한정된 컴포넌트
TBatchMove		
TUpdateSQL	TSQLDataSet	DataSetProvider와 ClientDataSet 추가
TNestedTable		
Data Controls		
TDbRichEdit		Win32에 한정된 컴포넌트
TDbCtrlGrid		
TDbChart		
ADO 페이지	dbExpress	
TADO Connection	TSQL Connection	
TADO Command	TSQLDataSet	DataSetProvider와 ClientDataSet 추가
TADO DataSet	TSQLDataSet	DataSetProvider와 ClientDataSet 추가
TADO Table	TSQLTable	DataSetProvider와 ClientDataSet 추가
TADO Query	TSQLQuery	DataSetProvider와 ClientDataSet 추가
TADO StoredProc	TSQLStored Proc	DataSetProvider와 ClientDataSet 추가
TRDSConnection		
	TSQLMonitor	
	TSQLClient DataSet	
InterBase 페이지	dbExpress	
TIBTable	TSQLTable	DataSetProvider와 ClientDataSet 추가
TIBQuery	TSQLQuery	DataSetProvider와 ClientDataSet 추가
TIBStoredProc	TSQLStored Proc	DataSetProvider와 ClientDataSet 추가
TIBDatabase	TSQL Connection	
TIBTransaction		
TIBUpdateSQL	TSQLDataSet	DataSetProvider와 ClientDataSet 추가
TIBDataSet	TSQLDataSet	DataSetProvider와 ClientDataSet 추가
TIBSQL	TSQLDataSet	DataSetProvider와 ClientDataSet 추가
TIBDatabaseInfo		
TIBSQLMonitor	TSQLMonitor	
TIBEvents		
MIDAS 페이지		
TDCOMConnection		Win32에 한정된 컴포넌트
TSocketConnection		Kylix 2에 추가될 예정
TsimpleObject Broker		Kylix 2에 추가될 예정
TWebConnection		Kylix 2에 추가될 예정
TCorbaConnection		Kylix 2에 추가될 예정
InternetExpress 페이지		
TXMLBroker		Kylix 2에 추가될 예정
TMidasPageProducer		Kylix 2에 추가될 예정
Internet 페이지		
TClientSocket	TTcpClient, TUdpSocket	
TServerSocket	TTcpServer, TUdpSocket	
TWebBrowser		Win32에 한정된 컴포넌트
FastNet 페이지	Indy Servers Indy Clients Indy Misc	
Decision Cube 페이지		
QReport 페이지		Nevrona의 Rave Reports 추천
Dialogs 페이지		
TOpenPictureDialog		
TSavePictureDialog		
TPrintDialog		
TPrinterSetupDialog		
Win 3.1		

팁: Delphi 5와 Kylix를 가지고 크로스 플랫폼 애플리케이션 구축을 시작하려는 경우에는 Kylix에 없는 컴포넌트는

보이지 않도록 Delphi 5 컴포넌트 팔레트를 설정해야 합니다. 이렇게 하기 위한 방법은 간단합니다. 즉, 컴포넌트 팔레트 위에서 마우스 오른쪽 버튼을 클릭한 후에 "properties"를 선택하고, 리스트에서 모든 컴포넌트를 숨기면 됩니다.

컴포넌트와 유닛

CLX와 VCL의 비교 논의는 CLX 유닛의 네이밍 컨벤션(예: QGraphics와 Graphics)으로 계속됩니다. 이를 통해 Delphi 6가 두 애플리케이션 프레임워크 (VCL와 CLX)가 디자인타임 지원이 통합되는 첫번째 개발 환경이 될 것이라는 점을 예상할 수 있습니다.

아래의 표는 Kylix(Source\VCL 디렉토리)에서 이름이 바뀐 Delphi 5(Source\VCL 디렉토리)의 유닛들을 나열하고 있습니다. 대부분의 변경에서 Q가 앞에 붙은 것을 주목하십시오.

Delphi 5 유닛	**Kylix 유닛**	**설명**
ActnList	QActnList	Win32 한정됨 Kylix에는 없음
ADODB		Win32 한정됨 Kylix에는 없음
ADOInt		Win32 한정됨 Kylix에는 없음
AppEvnts		Kylix에는 없음
AxCtrls		Win32 한정됨 Kylix에는 없음
BdeConst		Kylix에는 없음
BdeMts		Kylix에는 없음
Buttons	QButtons	
CheckLst	QCheckLst	
Classes	Classes	
ClipBrd	QClipBrd	
ComCtrls	QComCtrls	
ComStrs		Kylix에는 없음
Consts	Consts	Qconsts도 참조
Contnrs	Contnrs	
Controls	QControls	
CORBACon		Kylix 2에 추가될 예정
CORBARdm		Kylix 2에 추가될 예정
CORBAStd		Kylix 2에 추가될 예정
CORBAVcl		Kylix 2에 추가될 예정
CtlPanel		Kylix에는 없음
DataBkr		Kylix에는 없음
DB	DB	
DBActns	QDBActns	
DBCGrids		Kylix에는 없음
DBClient	DBClient	
DBCommon	DBCommon	
DBConsts	DBConsts	
DBCtrls	QDBCtrls	
DBExcept		Kylix에는 없음
DBGrids	QDBGrids	
DBInpReq		Kylix에는 없음
DBLookup		Kylix에는 없음
DBOleCtl		Win32 한정됨 Kylix에는 없음
DBPWDlg	DBLogDlg	
DBTables		Kylix에는 없음
DdeMan		Win32 한정됨 Kylix에는 없음
Dialogs	QDialogs	
DRTable		Kylix에는 없음
DSIntf	DSIntf	
ExtCtrls	QExtCtrls	
ExtDlgs		Kylix에는 없음
FileCtrl		Kylix에는 없음
Forms	QForms	
Graphics	QGraphics	
Grids	QGrids	
ImgList	QImgList	
IniFiles	IniFiles	
Mask	Qmask	
Masks	Masks	MaskUtil도 참조
MConnect		Kylix 2에 추가될 예정
Menus	Qmenus	
Midas	Midas	
MidasCon		Kylix에는 없음
MidConst	MidConst	MIDAS는 DataSnap으로 이름이 변경됨
MPlayer		Win32 한정됨 Kylix에는 없음
Mtsobj		Win32 한정됨 Kylix에는 없음
MtsRdm		Win32 한정됨 Kylix에는 없음
Mtx		Win32 한정됨 Kylix에는 없음
mxConsts		Decision Cube - Kylix에는 없음
ObjBrKr		Kylix 2에 추가될 예정
OleAuto		Win32 한정됨 Kylix에는 없음
OleConst		Win32 한정됨 Kylix에는 없음
OleCtnrs		Win32 한정됨

		Kylix에는 없음
OleCtrls		Win32 한정됨 Kylix에는 없음
OLEDB		Win32 한정됨 Kylix에는 없음
OleServer		Win32 한정됨 Kylix에는 없음
Outline		Kylix에는 없음
Printers	QPrinters	
Provider	Provider	
Registry		Kylix에는 없음 (IniFiles를 사용)
ScktCnst		Kylix에는 없음
ScktComp		Kylix에는 없음
Sconnect		Kylix에는 없음
SMIntf		Kylix에는 없음
StdActns	QStdActns	
StdCtrls	QStdCtrls	
StdVcl		Win32 한정됨 Kylix에는 없음
SvcMgr		WinNT에 한정됨 Kylix에는 없음
SyncObjs	SyncObjs	
Tabnotbk		Kylix에는 없음
Tabs		Kylix에는 없음
Toolwin		Kylix에서는 도킹 지원이 없음
TypInfo	TypInfo	
VCLCom		Win32 한정됨 Kylix에는 없음
WebConst		Win32 한정됨 Kylix에는 없음
Windows		Win32 한정됨 Kylix에는 없음

Delphi 6에서의 CLX 구현은 Kylix에서 현재 사용되는 유사한 파일 이름을 사용할 것으로 예상됩니다.

RTL 사용 코드의 마이그레이션

여기에서는 주로 Delphi 5와 Kylix 사이의 런타임 라이브러리(RTL)와 Object Pascal 언어의 차이에 초점을 두기로 합니다. Delphi 런타임 라이브러리(System 및 SysUtils 유닛)는 Kylix에서 거의 동일합니다. 거의라고 한 것은 전형적인 Win32 항목 몇 개(예: RaiseLastWin32Error와 Win32Check)가 제거되었기 때문입니다.

인터페이스

IInterface는 IUnknown를 대신하지만, 하위 호환성을 위해 IUnknown는 IInterface로 정의됩니다. IDispatch는 COM을 필요로 하므로 Kylix에는 없습니다. Kylix는 인터페이스 RTTI와 이식 가능한 variant를 도입했습니다.

이식 가능한 Variants

새로운 Variant 구현은 Portable Variants 라고 부릅니다. (새로운 Variants 유닛에 구현되어 있습니다) 이것은 Win32와 리눅스 사이에서 바이너리 호환이 가능한 variant이며, Win32 고유의 API를 사용하는 대신 이제는 Delphi Object Pascal에 네이티브하게 구현되어 있습니다. 새로운 variant는

- variant 배열과 안전한 배열 지원을 포함하고
- 사용자정의 variant 데이터 형식의 등록을 지원하고
- data coercion, 데이터 복사, 연산자 오버로딩 및 method invocation을 허용합니다.

디스크, 디렉토리, 파일

리눅스의 파일 시스템은 대소문자를 구분하므로 리눅스에서의 텍스트 파일 작업과 일반적인 파일 작업은 윈도우와는 다르게 작동합니다. “TABLE.DB”와 “table.db”가 같은 테이블을 가리키는 것으로 간주되는 애플리케이션의 영역들을 찾아서 수정해야 합니다.

리눅스에서는 디렉토리에서 윈도우가 백슬래시(\)를 사용하는 것과는 달리 슬래시(/)를 사용한다는 사실도 중요합니다. 또한, 리눅스에는 C: 드라이브가 없습니다! 이러한 사항들은 SysUtils 유닛에서 정의된 PathDelim 상수를 항상 잊지 않고 사용하면 (그리고 Kylix에서는 /로 설정하고 Delphi에서는 \로 설정하면) 쉽게 처리할 수 있습니다. 한가지 더 알려주자면, 다음과 같이 정의되는 DriveDelim과 PathSep 상수를 사용하십시오.

```
const
  PathDelim = {$IFDEF MS윈도우} '\';
    {$ELSE} '/'; {$ENDIF}
  DriveDelim = {$IFDEF MS윈도우} ':';
    {$ELSE} ''; {$ENDIF}
  PathSep = {$IFDEF MS윈도우} ';';
```

```
{$ELSE} ':'; {$ENDIF}
```

마지막으로, ExtractShortFileName 기능은 DiskSize 및 DiskFree와 마찬가지로 Kylix에 없습니다.

파일 I/O

관련된 주제로서, 윈도우가 텍스트 파일에서 라인의 끝을 표시할 때 CR+LF (carriage return과 line feed)을 사용하지만, 리눅스는 LF (line feed)만을 사용합니다. System 유닛에는 리눅스에서 tlbsLF로 설정된 DefaultTextLineBreakStyle라는 전역변수가 있는데, line break가 단순히 LF라는 의미입니다.

아래와 같이 SetLineBreakStyle 루틴을 호출하면 file-by-file 기반에서 DefaultTextLineBreakStyle의 값을 변경할 수 있습니다.

```
SetLineBreakStyle(logfile, tlbsCRLF);
```

이것은 CR+LF를 end-of-line 문자로 사용하여 logfile을 읽거나 씁니다. 이미 열려 있는 파일에 대해서는 이 루틴을 호출할 수 없습니다!

API 마이그레이션

애플리케이션을 하나의 플랫폼에서 다른 플랫폼으로 마이그레이션/이식할 때 API 레벨(윈도우 유닛과 Libc 유닛의 비교)에서 작동하는 코드와 부딪히지 않을 수 없습니다. 몇 가지 예를 살펴보겠지만, 이런 종류의 이식 문제는 피하는 것이 상책입니다.

API 호출

윈도우 API 호출은 Windows 유닛과 Delphi5\Source\RTL\Win 디렉토리(리눅스에는 명백히 없음)에 있는 관련 유닛들에 있습니다. 네이티브 리눅스 API는 Kylix에 대한 Libc, Xlib 및 Xpm 유닛과 Kylix\Source\RTL\Linux 디렉토리(Delphi 5에는 명백히 없음)에 래핑되어 있습니다.

윈도우 메시지

직접적인 윈도우 API는 제쳐놓더라도, 애플리케이션에서 보내는 윈도우 메시지의 수를 최소화하도록 노력해야 합니다. PostMessage와 SendMessage는 리눅스에서 동작하지 않으며, 이벤트 핸들러 또는 노티파이어에 대한 호출로 대체되어야 합니다.

IniFiles

윈도우 IniFile은 리눅스로 이식됩니다. 그러나, 윈도우 Registry는 물론 리눅스에 없습니다. 따라서, TIniFile은 제대로 동작하는 반면 TRegIniFile는 작동하지 않으며 TIniFile로 대체해야 합니다.

파일 시간

또 하나의 플랫폼 고유의 문제는 날짜와 시간이 저장되는 내부 포맷입니다. 여기에는 UNIX 포맷으로 저장된 파일 날짜가 포함됩니다. 이것은 파일 날짜를 파일 날짜나 Now 또는 Date와 비교하는 것과 같이 날짜를 비교할 때에는 문제가 되지 않습니다. 사실, Date와 Now는 Delphi와 Kylix 모두에서 1899년 12월 31일에 대하여 아직도 오프셋 1을 사용합니다. TDateTime 표현은 Kylix와 Delphi에서 동일하고, 단지 차이는 파일 날짜 스탬프입니다.

결론

기존의 윈도우용 Delphi 애플리케이션을 리눅스용 Kylix로 옮기는 기능은 대단히 강력합니다. 이 기능은 Delphi 6에서 CLX를 사용하는 진정한 크로스플랫폼 개발로 확장되고 개선될 것입니다.

사용했던 컴포넌트가 Kylix에서 사용 가능한지 점검한 후에 마이그레이션을 시작하여 프로젝트 소스 파일과 데이터베이스 테이블을 변환하고 Object Pascal 소스 코드를 깊이 있게 수정해야 합니다. 대부분의 프로젝트는 거의 변경 없이 이식할 수 있지만, 어떤 것은 전혀 이식되지 않습니다(예: ActiveForms). WebBroker 프로젝트는 거의 변경없이 이식되며, 일반적으로 마이그레이션을 시작한 후 몇 분에서 몇 시간 이내에 작동합니다.

Bob Swart (a.k.a. Dr.Bob - drbob@chello.nl)는 Delphi 컨설턴트이자 트레이너이며 Dr.Bob's Delphi Clinic (http://www.drbob42.com)의 웹마스터입니다. 또한, *The Delphi Magazine, UK-BUG Developer's Magazine, Delphi Developer, Blaise, SDGN Magazine, DevX, TechRepublic*의 프리랜서 기술 기고가이며, *Delphi 4 Unleashed, C++Builder 5 Developer's Guide* 및 곧 출간될 *Kylix Developer's Guide*와 *Delphi 6 Developer's Guide*와 같은 프로그래밍 서적에 많은 글을 썼습니다. 저자는 1993년부터 연례 볼랜드 컨퍼런스에서 정규 연사로 활동하고 있고, "Dr.Bob's Delphi 6 Clinic" 교육 과정을 위한 고유의 트레이닝 교재를 저술했습니다.

Borland®

100 Enterprise Way
Scotts Valley, CA 95066-3249
www.borland.com | 831-431-1000

respective owners. 11814