Ecrire et utiliser un objet dans une DLL avec Delphi

Auteur : Michaël Moreno
Site : http://michael.moreno.free.fr/

De la même façon qu'il ne viendrait pas à l'idée à un programmeur d'écrire les milliers de lignes de son programme dans un seul fichier, il ne viendrait pas à l'idée du programmeur professionnel de livrer son programme en un seul fichier. Certains considèrent que de ne livrer qu'un seul fichier est un avantage pour la maintenance et la livraison du produit. C'est en effet le cas pour les petits produits créés par des débutants/amateurs, mais sûrement pas pour le professionnel ayant un parc de logiciels à maintenir et à faire évoluer éventuellement à l'aide de mise à jour par internet.

Découper l'exécutable comme l'on découpe le code source est l'une des clés de la réussite de la maintenance et du développement de progiciels. Il existe plusieurs techniques de démembrement de l'exécutable. L'une d'entre elle repose sur l'utilisation de DLL (Dynamic Linked Library).

Les DLL permettent de regrouper un sein d'un fichier plusieurs méthodes (fonctions ou procédures) que l'on utilise depuis un programme extérieur. Plusieurs programmes pourront utiliser cette même DLL ce qui permet de gagner du temps de développement de logiciels. La mise à jour de la DLL permet de mettre à jour tous les programmes l'utilisant sans avoir à les recompiler. Le gain en temps de maintenance est souvent significatif.

Les DLL peuvent être chargées statiquement ou dynamiquement. Le chargement dynamique plus difficile à mettre en œuvre possède un avantage considérable, celui de permettre la programmation des fameux *plug-ins*.

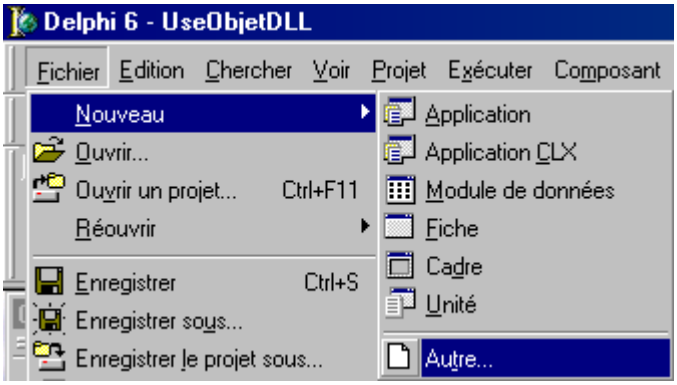
De nos jours, même si la technologie COM (Component Object Model) s'impose de plus en plus il faut reconnaître que les DLL ont encore de belles journées devant elles. La raison est simple. Le développement et la maintenance d'objet COM reste assez difficile pour le programmeur. En revanche, quel plaisir de développer l'application cliente !

Dans ce bref article, la programmation des objets au sein des DLL à l'aide de Delphi 6 est présentée de manière concise. Il est supposé que le programmeur connaît déjà le langage Pascal Objet. L'ensemble des étapes essentielles est détaillé à l'aide de captures partielles d'écran. Le programmeur non habitué au développement de DLL est en mesure de réaliser et mettre en oeuvre sa première DLL avec cet exemple sans aucune difficulté. Les sources de l'exemple sont disponibles.

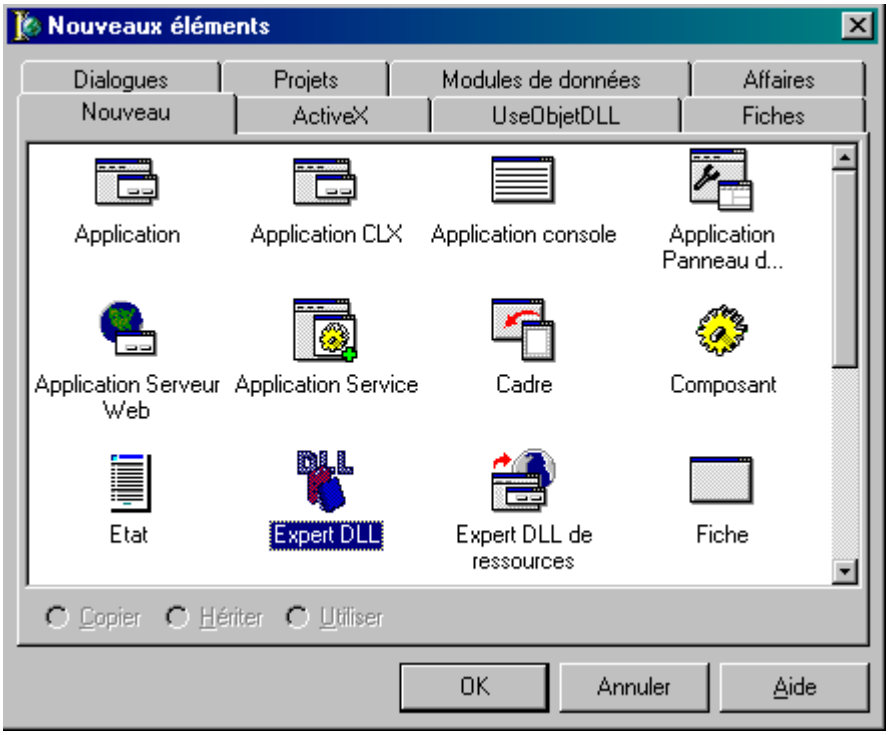
I. Ecriture de la DLL

A. Création du projet de la DLL

Dans le menu **Fichier**, cliquer sur **Nouveau** puis sur Autre…



Dans la boîte de dialogues qui apparaît, sélectionner **Expert DLL** puis cliquer sur OK.



Vous venez de créer le projet contenant la DLL. Lisez attentivement le commentaire. Il vous importe de choisir si oui ou non vous décider de faire passer ou de renvoyer des string de votre DLL. Il me semble qu'il ne vaut mieux jamais utiliser cette possibilité. Il est préférable de n'utiliser que les PChar pour des raisons de compatibilités et d'évolutivité de votre programme sur les différentes plateformes et versions de Microsoft Windows ©.

Finalement, enregistrer votre Projet DLL sous le nom : « DLLObjet.dpr ».

B. Ecriture de l'objet

Créez une nouvelle unité qui contiendra l'objet et enregistrez-là sous le nom de « DefObjet.pas ».

Puis définissez l'objet suivant (l'unité entière est recopiée ci-dessous afin de faciliter la compréhension) :

---

unit DefObjet;

```
interface
uses SysUtils;

type
  TObjetExemple = class
  private
    FX : Double;
  public
    procedure WX(const Value: Double); virtual; export; // Ecriture de X
    function ReadX : Double; virtual; export; // Lecture de X

    function Carre : Double; virtual; export; // Calcul du Carre de X
  end;

function GetObjetExemple : TObjetExemple;
// Cette fonction permet de créer et de renvoyer une instance de l'objet TobjetExemple.

implementation

{ TObjetExemple }

function GetObjetExemple : TObjetExemple; export;
begin
  result := TObjetExemple.Create;
end;
//-------------------------------------------------------------------------------------
function TObjetExemple.Carre: Double; export;
begin
  result := sqr(FX);
end;
//-------------------------------------------------------------------------------------
function TObjetExemple.ReadX: Double; export;
begin
  result := FX;
end;
//-------------------------------------------------------------------------------------
procedure TObjetExemple.WX(const Value: Double); export;
begin
  if Value > 0 then
    FX := Value
  else
    raise exception.Create('X must be  0');
end;

end.
```

Remarquez la présence de la clause **export** à la fin de la déclaration des fonctions.

Ecriture du source du projet de la DLL

Il faut à présent terminer l'exportation des fonctions de la DLL vers les autres programmes.
Pour cela retourner à votre source du projet (fichier DLLObjet.dpr) et déclarez dans la clause exports la fonction GetObjetExemple.
Votre source du projet doit ressembler à ceci :

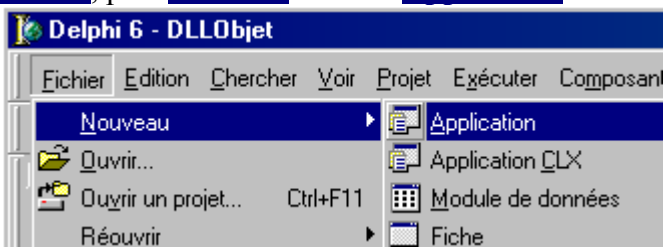
```
library DLLObjet;

uses
  SysUtils,
  Classes,
  DefObjet in 'DefObjet.pas';

{$R *.res}

exports
  GetObjetExemple;

begin
end.
```

Finalement, compilez votre DLL. Dans le répertoire où vous avez enregistré votre projet se trouve le fichier compilé « DLLObjet.DLL » que nous allons utiliser depuis une application externe.

II. Ecriture de l'application utilisant la DLL

A. Création d'un nouveau projet

Cliquez sur le menu **Fichier**, puis **Nouveau** et enfin **Application** :



Enregistrez votre projet sous le nom « UseObjetDDL.dpr » et l'unit1 sous le nom de « WinMain.pas ».
Créez une nouvelle unité et enregistrez-là sous le nom de « ImportObjetDLL.pas ».
Cette unité va servir à l'importation de la DLL.

B. Ecriture de l'unité d'importation de l'objet

Copiez-collez la déclaration de l'objet écrite dans la partie interface de l'unité DefObjet.pas de la DLL dans la nouvelle unité ImportObjetDLL.pas. Remplacez les mots clés **export** par **abstract** ce qui permet de ne pas avoir à réécrire l'implémentation de l'objet.

Enfin, il reste à déclarer comme fonction externe la fonction GetObjetExemple de l'unité DefObjet.pas de la manière qui suit :
function GetObjetExemple : TObjetExemple; external 'DLLObjet.dll';
dans la partie interface de ImportObjetDLL.

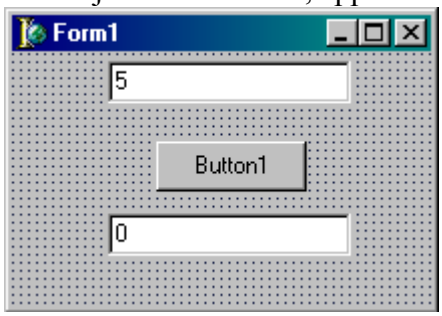
L'unité ImportObjetDLL.pas doit donc ressembler à ceci :

```
unit ImportObjetDLL;

interface
type
  TObjetExemple = class
  private
    FX : Double;
  public
    procedure WX(const Value: Double); virtual; abstract;
    function ReadX : Double; virtual; abstract;

    function Carre : Double; virtual; abstract;
  end;

function GetObjetExemple : TObjetExemple; external 'DLLObjet.dll';


implementation

{ TObjetExemple }

end.
```

« External 'DLLObjet.dll' » signifie que le liage de la DLL est statique. La DLL doit se trouver soit dans le répertoire du programme qui l'utilise soit dans l'un des répertoires clés de votre ordinateur (Windows ou Windows\System par exemple).

C. Utilisation de l'objet

Il ne nous reste plus qu'à utiliser l'objet. Sur la form1, apposez 2 TEdits et un TButton :



Dans la partie implémentation de l'unité WinMain, déclarez dans les **uses** l'unité ImportObjetDLL.

Puis sur l'évenement OnClic du Button utiliser l'objet en le créant à l'aide de la méthode de la DLL GetObjetExemble. Au final, l'unité WinMain ressemble à :

```
unit WinMain;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TForm1 = class(TForm)
    Button1: TButton;
    Edit1: TEdit;
    Edit2: TEdit;
    procedure Button1Click(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
  end;

var
  Form1: TForm1;

implementation
uses ImportObjetDLL;

{$R *.dfm}

procedure TForm1.Button1Click(Sender: TObject);
var Ob : TObjetExemple;
begin
  Ob := GetObjetExemple; // Création de l'objet
  try
    Ob.WX(StrToFloat(Edit1.Text));
    Edit2.Text := FloatToStr(OB.Carre);
  finally
    Ob.Free; // Libération
  end;
end;

end.
```

C’est à vous de tester votre premier programme liant un objet dans une DLL externe.

III.Conclusion

Ce bref article a été développé dans un soucis pédagogique avec un objet très simple. La construction d’objets plus complexes posera sans nul doute des difficultés, notamment lorsque l’objet est visuel tel qu’une TForm auquel cas il faudra récupérer le Handle de la Form.
Le développeur débutant pourra se référer à d’autres articles du site www.developpez.com et trouvera de l’aide auprès de la communauté française sur le forum nzn.fr.delphi que l’on trouve facilement en naviguant sur le site http://www.vienneinfo.org/.
Enfin, avant de tout écrire au sein des DLL, essayez de vous documenter sur les objets COM. Cette technologie et ses dérivées constituent une véritable révolution de l’informatique même si, malheureusement, son apprentissage et sa maîtrise sont longues et difficiles.