

Les Streams en Delphi

Introduction

On trouve relativement peu d'explications concernant les Streams, aussi bien dans les ouvrages que sur les sites internet consacrés à Delphi. Pourtant les Streams représentent une manière élégante, simple et efficace de travailler avec des flux de données. Au travers de quelques exemples nous allons aborder ce sujet intéressant.

Qu'est-ce qu'un Stream (flux) ?

Comme le mot Stream est fréquemment utilisé en rapport avec des fichiers, on pourrait croire qu'un Stream n'est rien d'autre qu'un fichier. Le Stream est toutefois une notion plus générale. Il permet en fait d'écrire ou de lire des informations les unes après les autres.

On peut, bien entendu, faire appel à des procédés de lecture et d'écriture dans des fichiers, mais l'avantage des Streams est qu'ils sont quasiment indépendants du support (mémoire centrale, fichiers ou autres). Il suffit d'écrire une seule fois le code permettant de lire et d'écrire les données dans un Stream. Par la suite on peut canaliser ce flux sur différents supports.

Comment le Stream stocke-t-il des données ?

Un Stream écrit et lit simplement un octet après l'autre, il n'a aucune connaissance de la structure de l'information sous-jacente. Par exemple le stockage d'un Longint dans un Stream correspond à l'écriture de 4 octets, sans préciser au Stream qu'il s'agit d'un nombre entier. Il n'y a donc aucune vérification de type. Le programmeur a toute liberté de relire les informations dans un autre type de structure réceptrice.

Cette liberté a un prix: il est du ressort du programmeur de savoir dans quel ordre il a écrit les données, afin de pouvoir les lire de manière cohérente, surtout si des informations de types différents cohabitent dans le même Stream.

Comment peut-on utiliser les Streams ?

Les utilisations des Streams sont variées:

- lecture et écriture de données dans des fichiers
- échange de données provenant de champs BLOB d'une base de données
- échange de données entre deux applications par WM_COPYDATA
- etc

Enregistrement de données dans un fichier (FileStream)

Dans le premier exemple, nous allons stocker des données dans un fichier à l'aide de la classe TFileStream que la VCL nous met à disposition. Pour éviter que l'exemple ne soit trop trivial, nous allons enregistrer une suite de données formées d'un nombre entier et d'une chaîne de caractères. En fait, nous allons travailler avec un tableau d'enregistrements.

Voici la structure de données utilisée:

```
type TData = record
  no      : integer;
  texte  : AnsiString;
end;
TDonnees = array of TData;
```

A l'aide des déclarations suivantes:

```
var Donnees: TDonnees;
    Stream: TStream;
```

Nous pouvons initialiser les données et créer le Stream, en fait un FileStream:

```
SetLength(Donnees, 2);
Donnees[0].no := 1;
Donnees[0].texte := 'Premier';
Donnees[1].no := 2;
Donnees[1].texte := 'Deuxième';
Stream := TFileStream.Create(GetCurrentDir() + '\'+
                             'Fichier.data', fmCreate);
```

'Fichier.data' est le nom du fichier créé dans le répertoire courant. Le paramètre fmCreate permet de créer le fichier. Dans le cas où le fichier existe, il sera simplement écrasé.

La phase suivante consiste à écrire le nombre d'enregistrements dans le flux, puis, dans une boucle, les données proprement dites:

```
try
  Len := Length(Donnees);
  Stream.Write(Len, SizeOf(Len)); // écriture de la taille des données
  for I := 0 to Length(Donnees) - 1 do begin
    Stream.Write(Donnees[I].no, SizeOf(Donnees[I].no)); // 1er champ
    Len := Length(Donnees[I].texte); // 2ème champ
    Stream.Write(Len, SizeOf(Len));
    Stream.Write(Donnees[I].texte[1], Len);
  end;
finally
  Stream.Free; // libération du flux
end;
```

Le premier paramètre de la méthode write est un buffer (variable tampon) sans type, alors que le second paramètre représente le nombre d'octets à écrire dans le flux.

L'écriture du contenu d'une variable entière ne pose pas de problème, mais il est moins évident d'écrire le contenu d'une chaîne de caractères. Un AnsiString est en fait un pointeur vers la zone mémoire dans laquelle se trouvent les données (caractères). Nous devons donc:

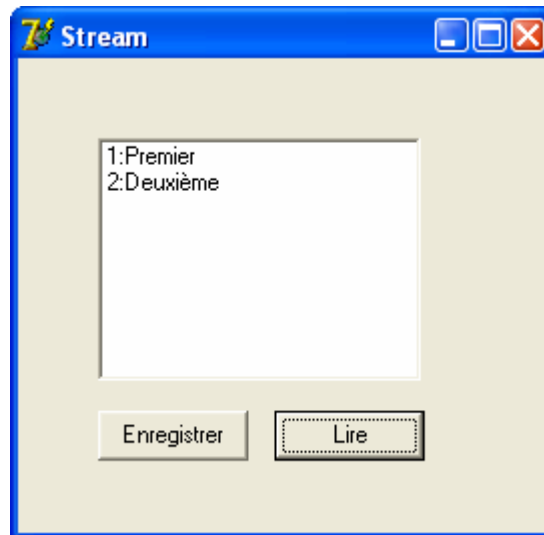
- écrire la longueur de la chaîne de caractères Length(Donnees[I].texte)
- puis écrire les caractères composant la chaîne

Grâce au déréférencement PChar(Donnees[I].texte)^ de la chaîne de caractères nous pouvons effectivement écrire la suite de caractères dans le flux.

A la fin de l'écriture, il faut penser à libérer la mémoire allouée au flux.

Lecture des données depuis un fichier

La lecture des données s'effectue de manière analogue. L'exemple qui suit reprend l'écriture des données dans le fichier. Il contient, en plus, la partie qui relit les données depuis le fichier. Un ListBox permet de visualiser les données lues.



Voici le programme au complet:

```
unit Unit1;

interface

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

type TData = record
  no      : integer;
  texte  : AnsiString;
end;
TDonnees = array of TData;
type
  TForm1 = class(TForm)
    BtnRecord: TButton;
    Btnlire: TButton;
    ListBox1: TListBox;
    procedure BtnRecordClick(Sender: TObject);
    procedure BtnlireClick(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
    procedure MontrerDonnees (Donnees: TDonnees);
  end;

var Form1: TForm1;

implementation

{$R *.dfm}
```

```

procedure TForm1.BtnRecordClick(Sender: TObject);
var Donnees: TDonnees;
    Stream: TStream;
    I: integer;
    Len: Longint;
begin
    SetLength(Donnees, 2);
    Donnees[0].no := 1;
    Donnees[0].texte := 'Premier';
    Donnees[1].no := 2;
    Donnees[1].texte := 'Deuxième';
    Stream := TFileStream.Create(getcurrentdir() + '\ ' + 'Fichier.data', fmCreate);
    try
        Len := Length(Donnees);
        Stream.Write(Len, SizeOf(Len)); // écriture de la taille des données
        for I := 0 to Length(Donnees) - 1 do begin
            Stream.Write(Donnees[I].no, SizeOf(Donnees[I].no)); // 1er champ
            Len := Length(Donnees[I].texte); // 2ème champ
            Stream.Write(Len, SizeOf(Len));
            Stream.Write(Donnees[I].texte[1], Len);
        end;
    finally
        Stream.Free; // libération du flux
    end;
end;

procedure TForm1.BtnlireClick(Sender: TObject);
var Donnees: TDonnees;
    Stream: TStream;
    I: integer;
    Len: Longint;
begin
    Stream := TFileStream.Create(GetCurrentDir() + '\ ' + 'Fichier.data', fmOpenRead);
    try
        Stream.Read(Len, sizeof(Len));
        SetLength(Donnees, Len);
        for i := 0 to Len - 1 do begin
            Stream.Read(Donnees[i].no, SizeOf(Donnees[i].no));
            Stream.Read(Len, SizeOf(Len));
            SetLength(Donnees[i].Texte, Len);
            Stream.Read(Donnees[i].Texte[1], Len);
        end;
    finally
        Stream.Free;
    end;
    MontrerDonnees(Donnees);
end;

procedure TForm1.MontrerDonnees (Donnees: TDonnees);
var Len : longint;
    i : longint;
begin
// affichage des données lues, dans un listbox
listbox1.Items.Clear;
Len := length(Donnees);
for i := 0 to Len - 1 do begin
    listbox1.items.Add(inttostr(Donnees[i].no) + ':' + Donnees[i].texte)
end;
end;

end.

```

Travail avec un flux en mémoire (MemoryStream)

Le traitement est identique, à part les points suivants:

- les données sont en mémoire et on n'indique donc pas un nom de fichier
- la variable Stream est un champ de l'objet Form1
- le flux (MemoryStream) est créé lors de la création de la fiche (lancement du programme) et libéré lors de la destruction de la fiche (fin du programme)
- avant de lire les données du flux, il faut se repositionner sur la première donnée (Stream.position := 0;)

Les deux derniers points se sont pas une modification de structure, mais sont plus liés au déroulement du programme.

Voici le programme complet avec mise en évidence des modifications apportées:

```
unit Unit1;

interface

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

type TData = record
  no      : integer;
  texte  : ansistring;
end;
TDonnees = array of TData;

type
  TForm1 = class(TForm)
    BtnRecord: TButton;
    Btnlire: TButton;
    ListBox1: TListBox;
    procedure BtnRecordClick(Sender: TObject);
    procedure BtnlireClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormDestroy(Sender: TObject);
  private
    { Déclarations privées }
    Stream: TStream;
  public
    { Déclarations publiques }
    procedure MontrerDonnees (Donnees: TDonnees);
  end;

var Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.BtnRecordClick(Sender: TObject);
var Donnees: TDonnees;
    I: integer;
    Len: Longint;
begin
  SetLength(Donnees, 2);
  Donnees[0].no := 1;
  Donnees[0].texte := 'Premier';
  Donnees[1].no := 2;
  Donnees[1].texte := 'Deuxième';
  Len := Length(Donnees);
  Stream.Write(Len, SizeOf(Len)); // écriture de la taille des données
```

```

for I := 0 to Length(Donnees) - 1 do begin
  Stream.Write(Donnees[I].no, SizeOf(Donnees[I].no)); // 1er champ
  Len := Length(Donnees[I].texte); // 2ème champ
  Stream.Write(Len, SizeOf(Len));
  Stream.Write(Donnees[I].texte[1], Len);
end;
end;

procedure TForm1.BtnlireClick(Sender: TObject);
var Donnees: TDonnees;
    I: integer;
    Len: Longint;
begin
  if not assigned (stream) then begin
    showmessage ('Le flux n''existe pas');
    exit;
  end;
  Stream.position := 0; // avant de lire il faut se replacer au début du flux!
  Stream.Read(Len, sizeof(Len));
  SetLength(Donnees, Len);
  for i := 0 to Len - 1 do begin
    Stream.Read(Donnees[i].no, SizeOf(Donnees[i].no));
    Stream.Read(Len, SizeOf(Len));
    SetLength(Donnees[i].Texte, Len);
    Stream.Read(Donnees[i].Texte[1], Len);
  end;
  MontrerDonnees(Donnees);
end;

procedure TForm1.MontrerDonnees (Donnees: TDonnees);
var Len : longint;
    i : longint;
begin
  // affichage des données lues, dans un listbox
  listbox1.Items.Clear;
  Len := length(Donnees);
  for i := 0 to Len - 1 do begin
    listbox1.items.Add(inttostr(Donnees[i].no) + ':' + Donnees[i].texte)
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Stream := TMemoryStream.Create;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
  if assigned (stream) then
    stream.Free;
end;

end.

```

TStream

Comme on peut le constater au vu des exemples précédents, les MemoryStreams et les FileStreams (Delphi propose d'autres types de Streams) sont très ressemblants. En fait, ils dérivent d'un type commun TStream. Ils en héritent donc les méthodes et les propriétés. Nous avons vu précédemment les méthodes Create, Read, Write et Free et la propriété Position. Il existe également la méthode Seek et la propriété Size.

Voici une brève description des ces principales méthodes et propriétés:

Create:

Ce constructeur permet de créer le Stream

Read:

Cette fonction lit des données en provenance du Stream, à partir de la position courante. Les données sont placées dans un buffer d'une taille déterminée:

Function Read (var Buffer; Count: Longint): Longint;

Cette fonction retourne le nombre d'octets réellement lus.

Write:

Cette fonction écrit des données dans un Stream, à partir de la position courante. Les données à écrire se trouvent dans un buffer d'une taille déterminée.

Function Write (const Buffer; Count: Longint): Longint;

Cette fonction retourne le nombre d'octets écrits.

Seek:

Cette fonction permet de déplacer la position courante à l'intérieur du Stream. La position courante représente l'emplacement à partir duquel les données vont être lues ou écrites lors de la prochaine opération de lecture ou d'écriture.

Function Seek (Offset: Longint; Origin: Word): Longint;

Offset représente le positionnement désirés en octets. Origin indique par rapport à quel emplacement se fait le positionnement (soFromBeginning depuis le début, soFromEnd depuis la fin et soFromCurrent depuis la position courante). Par exemple:

STream.Seek (-10, soFromEnd) permet de se positionner sur le 10^{ème} octet avant la fin du Stream.

CopyFrom:

Cette fonction permet de copier une portion de Stream dans un second Stream.

Function CopyFrom (Source: TStream; Count: Longint): Longint;

Le nombre d'octets copiés est spécifié par Count. Si Count vaut 0, tout le Stream source sera copié. La copie s'effectue depuis la position courante de la source vers et à partir de la position courante de la destination.

Cette fonction retourne le nombre d'octets copiés.

Position:

Cette propriété correspond à la position courante (en octets). En affectant une valeur à cette propriété on déplace la position courante. En lisant la valeur de cette propriété on peut connaître la position courante.

Size:

Cette propriété indique la taille actuelle du Stream, en octets.

Quelques exemples

Afin d'illustrer l'utilisation des Streams, voici quelques programmes.

Exemple 1

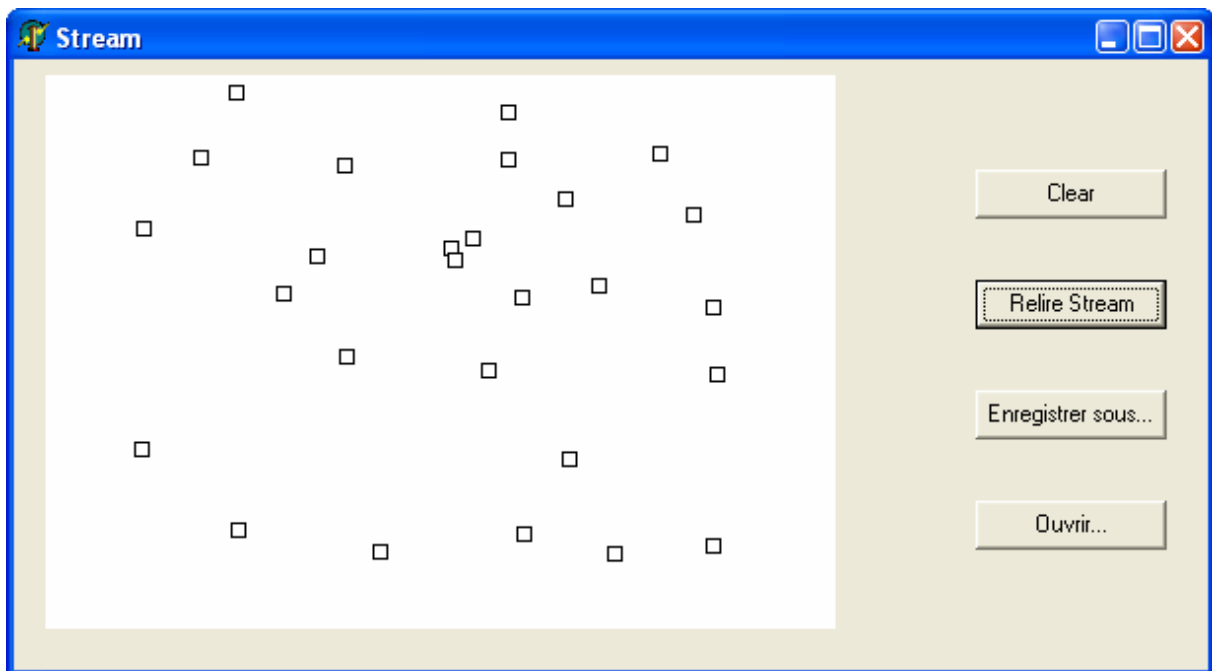
Dans ce programme l'utilisateur peut cliquer dans une zone (de type TImage) et marquer l'emplacement à l'aide d'un petit carré. La position (x,y) de chaque clic est enregistrée au fur et à mesure dans un MemoryStream.

Le bouton **Clear** permet d'effacer la zone de dessin.

Le bouton **Relire Stream** permet de relire les données du MemoryStream et de redessiner les petits carrés (dans un but de vérification, par exemple).

Le bouton **Enregistrer sous...** permet d'enregistrer les données du MemoryStream dans un FileStream (fichier sur disque)

Le bouton **Ouvrir...** permet de relire les données du FileStream et de redessiner les carrés correspondants.



Voici les différentes parties de ce programme. Tout d'abord la déclaration (dans la partie privée de la fiche):

```
mstream : TMemoryStream; // stream mémoire
```

Création de la fiche:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    mstream := TMemoryStream.Create; // création du stream mémoire
end;
```

Destruction de la fiche:

```
procedure TForm1.FormDestroy(Sender: TObject);
begin
    if assigned (mstream) then
```

```

        mstream.free;                // libération du stream
    end;

```

La gestion des clics de la souris:

```

procedure TForm1.ImagelMouseDown(Sender: TObject; Button: TMouseButton;
                                Shift: TShiftState; X, Y: Integer);
begin
    imagel.canvas.Rectangle (x-4, y-4, x+4, y+4); // dessin rectangle
    mstream.Write (x, sizeof(integer));           // enregistrer la position
    mstream.Write (y, sizeof(integer));           // dans le stream mémoire
end;

```

Bouton de relecture du Stream:

```

procedure TForm1.BtnreadClick(Sender: TObject);
var i, x, y : integer;
begin
    // affichage des points stockés dans le stream mémoire
    mstream.position := 0;
    for i := 1 to mstream.size div (2*sizeof(integer)) do begin
        mstream.read (x, sizeof(integer));
        mstream.read (y, sizeof(integer));
        imagel.canvas.Rectangle(x-4, y-4, x+4, y+4);
    end;
end;

```

Bouton d'effacement du dessin:

```

procedure TForm1.BtnclearClick(Sender: TObject);
begin
    // effacement de l'image
    imagel.picture := nil;
    imagel.canvas.moveto (0,0);
end;

```

Bouton d'enregistrement du Stream. SD est un TSaveDialog:

```

procedure TForm1.BtnsaveClick(Sender: TObject);
begin
    if SD.execute then begin
        mstream.SaveToFile (SD.filename); // enregistrer dans un fichier
        imagel.picture := nil;           // effacement de l'image
        imagel.canvas.moveto (0,0);
    end;
end;

```

Bouton de lecture du FileStream. OD est un TOpenDialog:

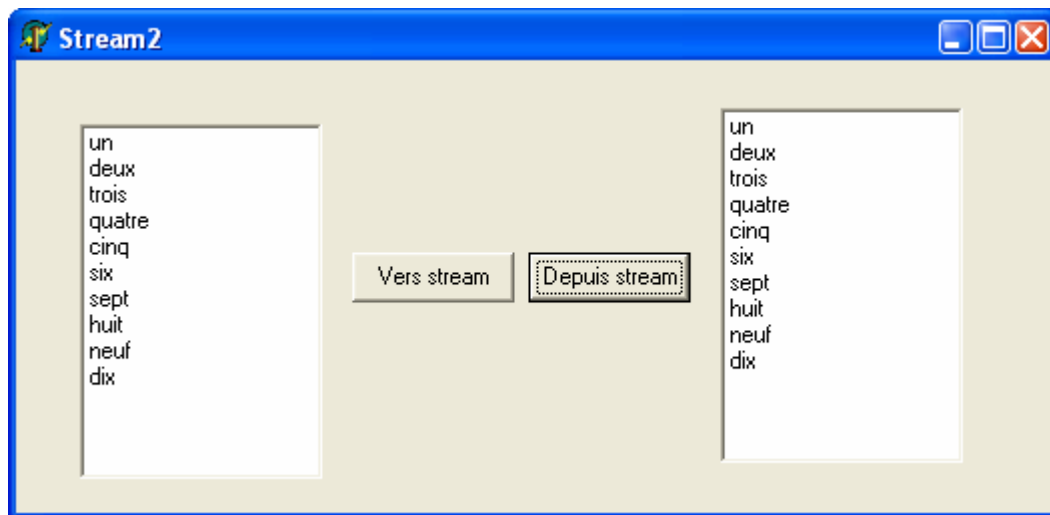
```

procedure TForm1.BtnopenClick(Sender: TObject);
var i, x, y : integer;
begin
    if OD.execute then begin
        if assigned (mstream) then
            mstream.free;                // libération du stream
        mstream := TMemoryStream.Create; // création du stream
        mstream.LoadFromFile (OD.filename); // chargement des données
        for i := 1 to mstream.size div (2*sizeof(integer)) do begin
            mstream.read (x, sizeof(integer)); // lecture des données
            mstream.read (y, sizeof(integer));
            imagel.canvas.Rectangle(x-4, y-4, x+4, y+4); // affichage rectangle
        end;
    end;
end;

```

Exemple 2

Ce programme met en évidence l'utilisation des méthodes LoadFromStream et SaveToStream disponibles pour plusieurs composants. Dans notre cas, le programme copie le contenu d'un ListBox dans un second ListBox en passant par un MemoryStream intermédiaire.



Pour l'utiliser, il faut d'abord cliquer sur le bouton **Vers stream**, puis sur le bouton **Depuis stream**.

La variable suivante est déclarée dans la partie privée de la fiche:

```
flux : TMemoryStream;
```

Le reste du programme est le suivant:

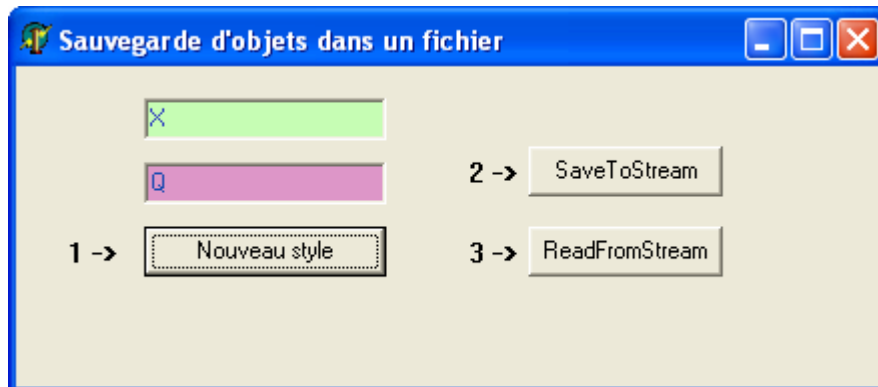
```
procedure TForm1.BtnversClick(Sender: TObject);
begin
    if not assigned (flux) then
        flux := TMemoryStream.Create;
    Listel.Items.SaveToStream (flux);
end;

procedure TForm1.BtndepuisClick(Sender: TObject);
begin
    if assigned (flux) then begin
        flux.position := 0;
        liste2.items.LoadFromStream (flux);
    end;
end;

procedure TForm1.FormDestroy(Sender: TObject);
begin
    if assigned (flux) then
        flux.free;
end;
```

Exemple 3

Cet exemple montre le streaming des composants et se présente sous la forme suivante:



Voici une manière d'utiliser ce programme:

- cliquer une ou plusieurs fois sur le bouton **Nouveau style**
- cliquer sur le bouton **SaveToStream** et se rappeler des couleurs des deux zones de texte
- cliquer une ou plusieurs fois sur le bouton **Nouveau style**
- cliquer sur le bouton **ReadFromStream** et vérifier que le style sauvegardé est bien restitué

Bien entendu ce qui est fait ici avec un MemoryStream peut également l'être avec un FileStream. On dira que les propriétés des composants sont "streamables".

Exemple 4

L'exemple suivant consiste à reprendre l'exemple 3, enregistrer les propriétés des composants dans un fichier, enfin de relire le fichier afin de restituer ces propriétés.

Indications

- les programmes exécutables ainsi que les sources des exemples ci-dessus sont en principe disponibles sur le site où se trouve le présent document. Si ce n'est pas le cas, il convient de les chercher sur jca.developpez.com ou sur www.unvrai.com
- n'oubliez pas de consulter la section Delphi de developpez.com (delphi.developpez.com) pour tout ce qui concerne Delphi
- n'hésitez pas à consulter le forum consacré à Delphi (accessible depuis delphi.developpez.com)