Utiliser l'ActiveX Windows Media Player dans votre Application

par Lelong Julien (jlelong.developpez.com)

Date de publication : 19 novembre 2008

Dernière mise à jour :

A travers cet article, nous découvrirons comment utiliser les fonctionnalités de bases de l'ActiveX WindowsMediaPlayer. Nous verrons donc comment créer sa propre playlist, activer ou non les différents mode de lecture (aléatoire, répéter).

I - Présentation

A travers cet article, nous découvrirons comment utiliser les fonctionnalités de bases de l'ActiveX WindowsMediaPlayer. Nous verrons donc comment créer sa propre playlist, activer ou non les différents mode de lecture (aléatoire, répéter).

II - Importation de l'ActiveX

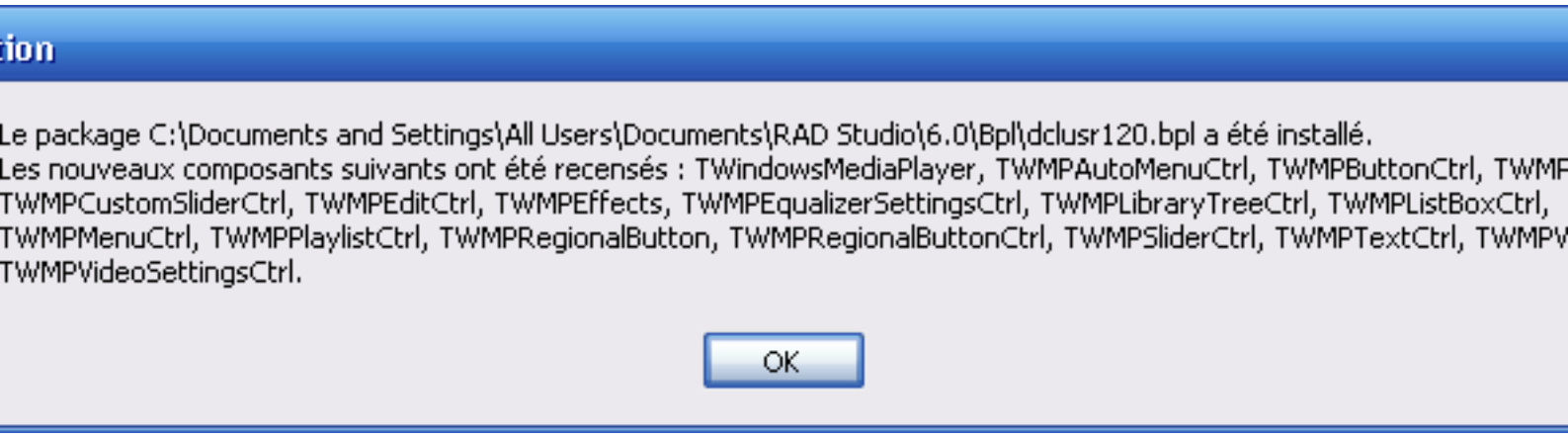
Lancez Delphi mais ne créez aucun projet. Si un projet est ouvert, faites :

- Fichier
- Tout fermer

De base, Delphi n'a pas l'ActiveX Windows Media Player d'importé. Il nous faut donc le faire.
Pour cela, suivez les étapes ci-dessous :

- Cliquez sur l'onglet *Composant*
- Sélectionnez *Importer un composant*
- Sélectionnez *Importer un contrôle ActiveX* puis faites *Suivant*
- Sélectionnez l'ActiveX *Windows Media Player* (situé vers la fin) puis faites *Suivant*
- Renseignez la palette en inscrivant "ActiveX", cliquez directement sur "Suivant"
- Sélectionnez *Installer dans un package existant* (façon la plus simple) puis faites *Suivant*
- Cliquez sur *Parcourir* et rechercher le package *dclusr.dpk* (Pour D2009, il se situe dans C:\Program Files \CodeGear\RAD Studio\6.0\lib)
- Enfin, cliquez sur *Terminer*

La compilation s'effectue est un message final nous averti que tout est installé.



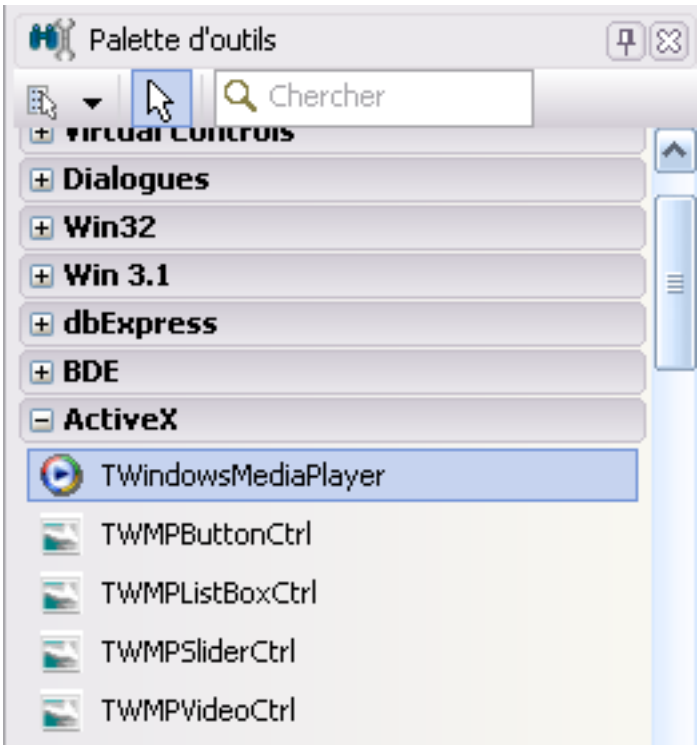
Installation ActiveX réussie

Maintenant que l'ActiveX est installé, nous pouvons créer notre application.
Nous allons donc avant toute chose tout fermer :

- Fichier
- Tout fermer

III - Utilisation de l'ActiveX WindowsMediaPlayer en conception (Design-Time)

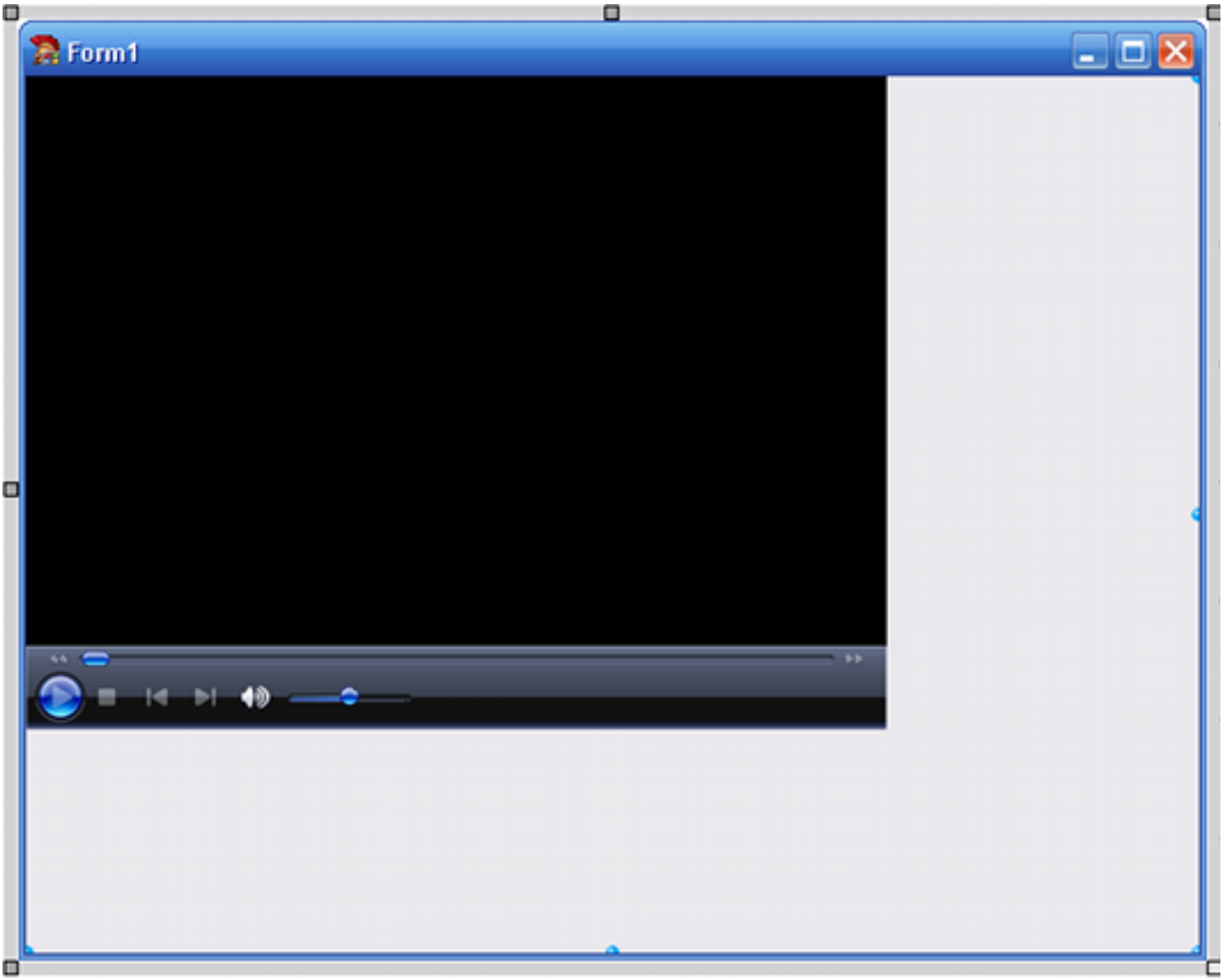
Tout d'abord, créons une nouvelle application. Ensuite déposons sur notre Form l'ActiveX TWindowsMediaPlayer présent dans la palette ActiveX.



ActiveX WindowsMediaPlayer

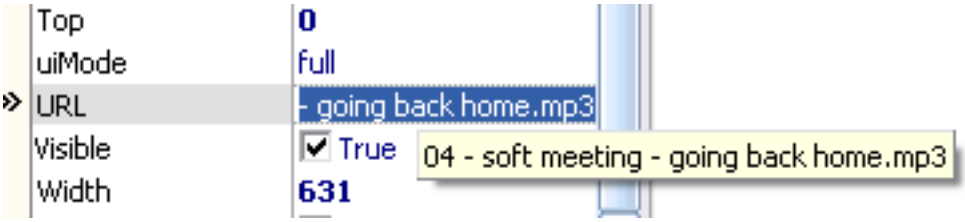
Nous renommons directement la propriété *Name* de notre ActiveX en WMP et mettons par la même occasion la propriété *align* à *AlClient*.

Vous devriez arriver à ce résultat:



Vous remarquerez que le lecteur ne prends pas toute la Form mais pourtant, sa taille est bien définie.
Pour vous en persuader, lancez l'application avec la touche F9.

Désormais, nous pouvons lire une musique ou une vidéo.
Pour ce faire, il faut renseigner la propriété *URL* de l'ActiveX en indiquant le chemin du fichier à lire.



L'URL du fichier à lire doit être le chemin complet si ce dernier ne se trouve pas dans le même répertoire que l'application.
Pareillement, rien ne vous empêche de lire un fichier avi!

Il ne vous reste plus qu'à contrôler le lecteur comme bon vous semble.

Notez également que lorsque vous faites un clic droit sur le TWindowsMediaPlayer, un menu déroulant apparaît vous permettant bon nombre de possibilités.

IV - Utilisation de l'ActiveX WindowsMediaPlayer par code

Nous créons un nouveau projet Delphi pour cette partie.
Tout sera géré par code. Dans cette partie, nous ne traiterons que de la lecture de fichiers musicaux.

Commencez par créer une interface ressemblant à ceci :

Interface Windows Media Player

IV-A - Les fonctionalités de bases : lecture, pause, stop

IV-A-1 - Création de l'ActiveX TWindowsMediaPlayer

Nous instancions l'ActiveX par code.
Il faut avant tout rajouter dans les uses ces trois unités : *WMPLib_TLB, OleServer, OleCtrls*.
Ensuite, nous déclarons une variable globale *WMP* et dans le OnCreate de la Form, nous l'instancions, ce qui donne ceci :

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
    BtnLecture: TButton;
    BtnPause: TButton;
    BtnStop: TButton;
    LabelPisteEnCours: TLabel;
    procedure FormCreate(Sender: TObject);
  private
    { Déclarations privées }
  public
    { Déclarations publiques }
    WMP:TWindowsMediaPlayer;
  end;
```

```
var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  WMP:=TWindowsMediaPlayer.Create(Form1);
  // Evite que la lecture d'un média démarre automatiquement
  WMP.settings.autoStart:=false;
end;

end.
```

Seule la ligne instanciant l'ActiveX est nécessaire! En effet, nous n'utiliserons pas l'interface que nous proprose l'ActiveX.
Remarquez également que j'ai mis la propriété autoStart à false. Ceci empêche la lecture automatique d'un média si il y en a déjà un.

IV-A-2 - Mode lecture

Dans l'événement OnClick du BtnLecture, on allons renseigner la propriété *URL* du TWindowsMediaPlayer en spécifiant le chemin du fichier à écouter.
De plus, nous afficherons le titre du média dans le label.

```
procedure TForm1.BtnLectureClick(Sender: TObject);
begin
  WMP.URL:='MonFichier.mp3';
  // Lance la lecture du fichier
  WMP.controls.play;
  // Récupère le titre du média lu actuellement
  LabelPisteEnCours.Caption:=WMP.currentMedia.name;
  // Désactive la Bouton de lecture
  BtnLecture.Enabled:=false;
end;
```

Vous pouvez tester le tout. Le titre du média apparaîtra dans le label.
Vous l'avez déjà sans doute compris, tout se passe par la méthode *controls*. C'est donc à partir de là que nous pouvons agir sur la lecture, la pause, l'arrêt, etc..., du média.

IV-A-3 - Mode pause

Nous codons maintenant la partie pour mettre en pause le média.
Ceci se passe dans l'événement OnClick du BtnPause.
Nous désactiverons le bouton pause et nous rajouterons un test sur le bouton Lecture pour la reprise afin de savoir si le morceau était déjà lancé ou non.
Le code pour la mise en pause est donc :

```
procedure TForm1.BtnPauseClick(Sender: TObject);
begin
  WMP.controls.pause;
  // Désactive le bouton Pause
  BtnPause.Enabled:=false;
  // Réactive le bouton Lecture
  BtnLecture.Enabled:=true;
end;
```

Le code pour le bouton de lecture quant à lui se trouve modifié. Nous utilisons la propriété **PlayState** qui nous retournera l'état du lecteur pour rejouer ou non le fichier depuis le début.

```
procedure TForm1.BtnLectureClick(Sender: TObject);
begin
  // On définit les états des boutons Lecture, Pause, Stop
  BtnLecture.Enabled:=false;
  BtnPause.Enabled:=true;
  BtnStop.Enabled:=true;
  // Selon le status de lecture, on effectue telle ou telle action
  case WMP.playState of
    wmppsUndefined,wmppsStopped : begin
                                    WMP.URL:='MonFichier.mp3';
                                    LabelPisteEnCours.Caption:=WMP.currentMedia.name;
                                    WMP.controls.play;
                                  end;
    wmppsPaused : begin
                    WMP.controls.play;
                  end;
  end;
end;
```

IV-A-4 - Mode stop

Pour terminer, il nous reste à coder le bouton BtnStop.
Rien de bien difficile, le plus dur étant déjà traité.

```
procedure TForm1.BtnStopClick(Sender: TObject);
begin
  WMP.controls.stop;
  // Désactive le bouton Stop et Pause
  BtnStop.Enabled:=false;
  BtnPause.Enabled:=false;
  // Réactive le bouton de lecture
  BtnLecture.Enabled:=true;
end;
```

IV-A-5 - Code complet

Voici le code complet pour la gestion de base d'un média:

```
unit Unit1;

interface

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

type
  TForm1 = class(TForm)
    BtnLecture: TButton;
    BtnPause: TButton;
    BtnStop: TButton;
    LabelPisteEnCours: TLabel;
    procedure FormCreate(Sender: TObject);
    procedure BtnLectureClick(Sender: TObject);
    procedure BtnPauseClick(Sender: TObject);
    procedure BtnStopClick(Sender: TObject);
  private
    { Déclarations privées }
```

```
  public
    { Déclarations publiques }
    WMP:TWindowsMediaPlayer;
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  WMP:=TWindowsMediaPlayer.Create(Form1);
  // Evite que la lecture d'un média démarre automatiquement
  WMP.settings.autoStart:=false;
end;

procedure TForm1.BtnLectureClick(Sender: TObject);
begin
  // On définit les états des boutons Lecture, Pause, Stop
  BtnLecture.Enabled:=false;
  BtnPause.Enabled:=true;
  BtnStop.Enabled:=true;
  // Selon le status de lecture, on effectue telle ou telle action
  case WMP.playState of
    wmppsUndefined,wmppsStopped : begin
                                    WMP.URL:='MonFichier.mp3';
                                    LabelPisteEnCours.Caption:=WMP.currentMedia.name;
                                    WMP.controls.play;
                                  end;
    wmppsPaused : begin
                    WMP.controls.play;
                  end;
  end;
end;

procedure TForm1.BtnPauseClick(Sender: TObject);
begin
  WMP.controls.pause;
  // Désactive le bouton Pause
  BtnPause.Enabled:=false;
  // Réactive le bouton Lecture
  BtnLecture.Enabled:=true;
end;

procedure TForm1.BtnStopClick(Sender: TObject);
begin
  WMP.controls.stop;
  // Désactive le bouton Stop et Pause
  BtnStop.Enabled:=false;
  BtnPause.Enabled:=false;
  // Réactive le bouton de lecture
  BtnLecture.Enabled:=true;
end;

end.
```

Il ne vous reste plus qu'à tester le tout.

IV-B - Un peu plus de fonctions : affichage de la durée, gestion du volume

IV-B-1 - Affichage de la durée

Pour afficher la durée du média, il faut passer par l'événement *OnStatusChange* qui, lorsque le statut du lecteur sera en lecture, rajoutera l'info temps dans le libellé.

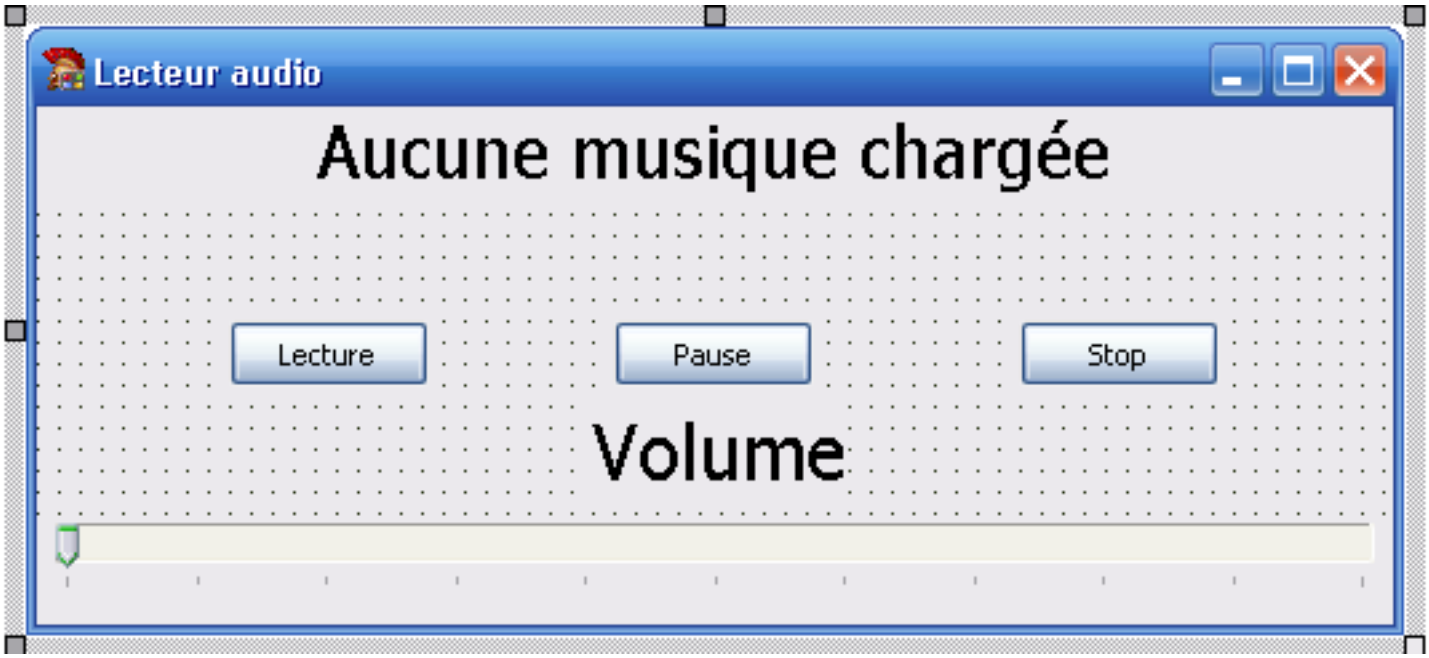
Voici le code en conséquence (seules les parties modifiées sont affichées):

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, WMPLib_TLB, OleServer, OleCtrls, ExtCtrls;

type
  TForm1 = class(TForm)
    BtnLecture: TButton;
    BtnPause: TButton;
    BtnStop: TButton;
    LabelPisteEnCours: TLabel;
    Timer1: TTimer;
    procedure FormCreate(Sender: TObject);
    procedure BtnLectureClick(Sender: TObject);
    procedure BtnPauseClick(Sender: TObject);
    procedure BtnStopClick(Sender: TObject);
  private
    { Déclarations privées }
    procedure StatusChange(ASender: TObject);
  public
    { Déclarations publiques }
    WMP:TWindowsMediaPlayer;
  end;

var
  Form1: TForm1;

implementation

{$R *.dfm}

procedure TForm1.FormCreate(Sender: TObject);
begin
  WMP:=TWindowsMediaPlayer.Create(Form1);
  // Evite que la lecture d'un média démarre automatiquement
  WMP.settings.autoStart:=false;
  WMP.OnStatusChange:=StatusChange;
end;

procedure TForm1.BtnLectureClick(Sender: TObject);
begin
  // On définit les états des boutons Lecture, Pause, Stop
  BtnLecture.Enabled:=false;
  BtnPause.Enabled:=true;
  BtnStop.Enabled:=true;
  // Selon le status de lecture, on effectue telle ou telle action
  case WMP.playState of
    wmppsUndefined,wmppsStopped : begin
                                    WMP.URL:='MonFichier.mp3';
                                    WMP.controls.play;
                                  end;
    wmppsPaused : begin
                    WMP.controls.play;
                  end;
  end;
end;

procedure TForm1.StatusChange(ASender: TObject);
begin
  case WMP.playState of
    wmppsPlaying :  begin
                      Timer.Enabled:=true;
                      LabelPisteEnCours.Caption:=WMP.currentMedia.name
+' ('+WMP.currentMedia.durationString+')';
                      JvTrackBar1.Max:=Trunc(WMP.currentMedia.duration);
```

```
                    JvTrackBar1.Frequency:=1;
                  end;
    wmppsStopped :  begin
                    BtnLecture.Enabled:=true;
                    BtnPause.Enabled:=false;
                    BtnStop.Enabled:=false;
                  end;
  end;
end;
```

IV-B-2 - Gestion du Volume

Nous rajoutons une TTrackBar (onglet Win32) pour gérer le son que nous renommerons TrackBarVolume et un TLabel que nous renommerons LabelVolume, pour ressembler à l'image ci-dessous.



Nous modifions certaines propriétés de la TrackBar :

- Max : 100 (volume maximum 100%)
- Frequency : 5 (pas d'affichage tous les 5%)

Nous modifions le code du OnCreate de la Form pour prendre en compte la modification du volume, et nous rajoutons du code dans l'événement OnChange de la TrackBar pour la gestion du volume.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  WMP:=TWindowsMediaPlayer.Create(Form1);
  // Evite que la lecture d'un média démarre automatiquement
  WMP.settings.autoStart:=false;
  WMP.OnStatusChange:=StatusChange;
  WMP.settings.volume:=50;
  TrackBarVolume.Position:=50;
end;

procedure TForm1.TrackBarVolumeChange(Sender: TObject);
begin
  WMP.settings.volume:=TrackBarVolume.Position;
end;
```

Pour mettre le son sur Mute (muet) il suffit d'utiliser la propriété mute (WMP.settings.mute) et de lui affecter true ou false selon le besoin.

IV-C - Gérer une PlayList

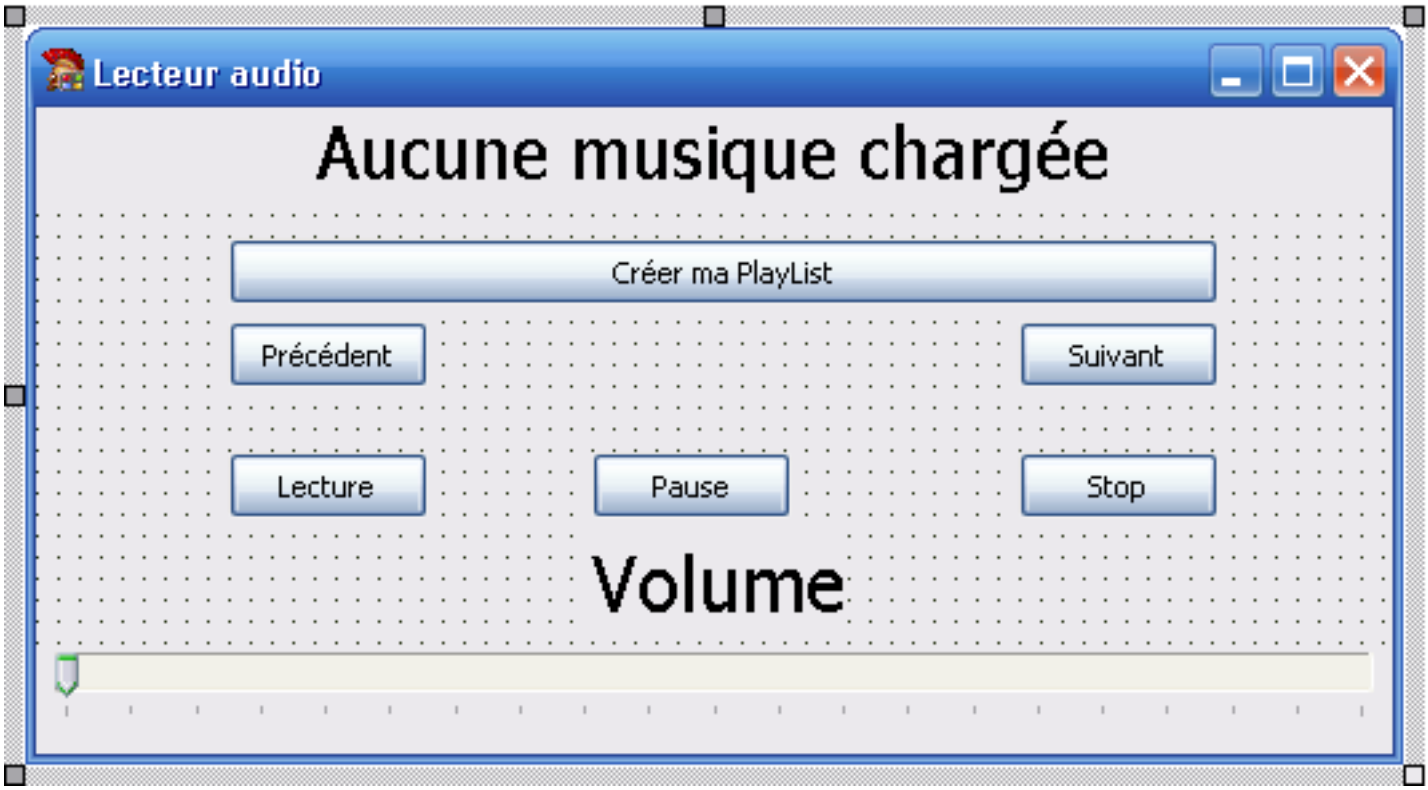
Dans cette partie, nous verrons comment manipuler une PlayList.
Nous verrons donc comment créer notre PlayList, ajouter des fichier à celle-ci et en supprimer.

IV-C-1 - Création de la PlayList

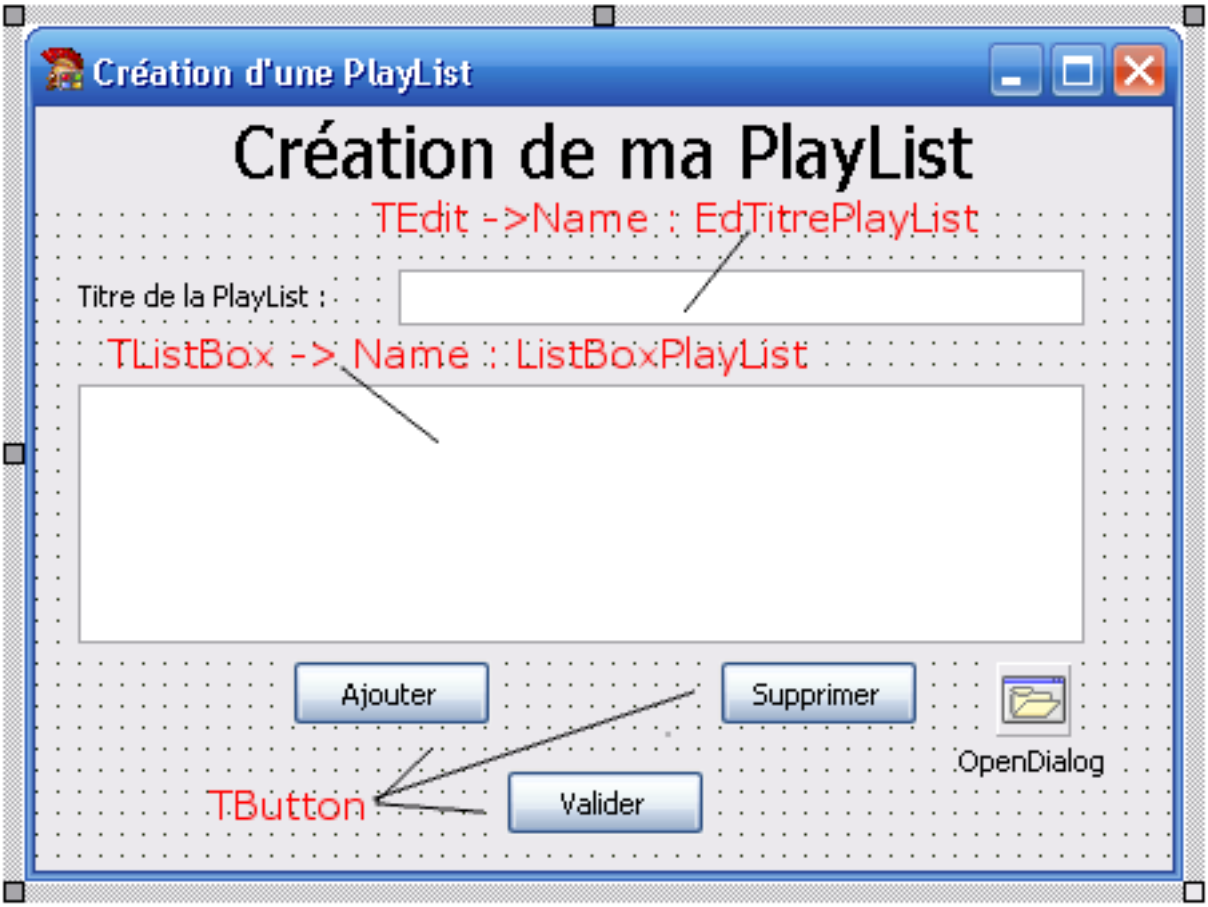
Pour créer notre PlayList, nous déclarons une variable dans la partie public de type IWMPPlaylist. De cette façon, nous pouvons avoir accès à la PlayList de la Form1 via la Form de création de notre PlayList.Nous rajoutons donc un bouton sur la Form1 nommé *BtnCreerPlayList* ainsi que deux autres boutons *BtnSuivant* et *BtnPrecedent*.
Par la même occasion, nous ajoutons déjà dans dans les uses de celle-ci l'unité *unit2* qui fait référence à la Form2 (création de la playlist).
Voici la déclaration :

```
unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StrUtils, StdCtrls, WMPLib_TLB, OleServer, OleCtrls, ExtCtrls, ComCtrls, Unit2;

type
  TForm1 = class(TForm)
    BtnLecture: TButton;
    BtnPause: TButton;
    BtnStop: TButton;
    LabelPisteEnCours: TLabel;
    TrackBarVolume: TTrackBar;
    LabelVolume: TLabel;
    BtnCreerPlayList: TButton;
    BtnPrecedent: TButton;
    BtnSuivant: TButton;
    procedure FormCreate(Sender: TObject);
    procedure BtnLectureClick(Sender: TObject);
    procedure BtnPauseClick(Sender: TObject);
    procedure BtnStopClick(Sender: TObject);
    procedure TrackBarVolumeChange(Sender: TObject);
    procedure BtnCreerPlayListClick(Sender: TObject);
  private
    { Déclarations privées }
    procedure StatusChange(ASender: TObject);
  public
    { Déclarations publiques }
    maPlayList:WMPLib_TLB.IWMPPlaylist;
    WMP:TWindowsMediaPlayer;
  end;
```

Voici la présentation de la Form1 :



Nous créons donc la nouvelle Form qui nous servira pour la création de la PlayList et nous rajoutons tout de suite dans les uses, la référence à la Form1 ainsi que l'unité *WMPLib_TLB*.



```
unit Unit2;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, WMPLib_TLB, Unit1;
```

La création des interfaces étant finie, nous pouvons désormais nous attarder sur la gestion de la PlayList.

IV-C-2 - Ajouter un fichier à la PlayList

Tout d'abord, nous filtrons le type de fichier à ajouter.
Pour cela, on définit la propriété *filter* du OpenDialog dans le OnCreate :

```
procedure TForm2.FormCreate(Sender: TObject);
begin
  OpenDialog.Filter:='*.mp3|*.mp3';
end;
```

Ensuite, nous codons la partie permettant l'ajout d'un fichier et par la même occasion, le bouton *Valider*.

```
procedure TForm2.BtnAjouterClick(Sender: TObject);
var
  // On déclare le média à ajouter à la playList
  chanson:WMPLib_TLB.IWMPMedia;
begin
  // Si le EdTitrePlayList est vide, on affiche un message d'erreur
  if(EdTitrePlayList.Text<>'')then
  begin
   // Si notre PlayList n'a pas encore été créée, alors on la créé, sinon on modifie son titre
    if(not(Assigned(Form1.maPlayList)))then
    begin
      Form1.maPlayList:=Form1.WMP.playlistCollection.newPlaylist(EdTitrePlayList.Text);
    end
    else
    begin
      Form1.maPlayList.name:=EdTitrePlayList.Text;
    end;

    if(OpenDialog.Execute())then
    begin
      // On crée le média et on l'ajoute dans la PlayList ainsi que dans le ListBoxPlayList
      chanson:=Form1.WMP.newMedia(OpenDialog.FileName);
      ListBoxPlayList.Items.Add(OpenDialog.FileName);
      Form1.maPlayList.insertItem(ListBoxPlayList.Items.Count-1,chanson);
    end;
  end
  else
  begin
    ShowMessage('Veuillez saisir le titre de la PlayList avant de continuer');
    EdTitrePlayList.SetFocus;
  end;
end;

procedure TForm2.BtnValiderClick(Sender: TObject);
begin
  Self.Close;
end;
```

Ici, on ajoute un fichier directement lorsque la sélection via l'OpenDialog est faite. Ce n'est juste pour l'exemple car le plus judicieux aurait été de faire une boucle qui ajouterait les fichiers une fois que l'on a validé entièrement le tout.

La partie d'ajout de fichiers est terminé, il nous reste à modifier le bouton de lecture situé sur la Form1 pour prendre en compte la PlayList.

```
procedure TForm1.BtnLectureClick(Sender: TObject);
begin
  // On définit les états des boutons Lecture, Pause, Stop
  BtnLecture.Enabled:=false;
  BtnPause.Enabled:=true;
  BtnStop.Enabled:=true;
  // Selon le status de lecture, on effectue telle ou telle action
  case WMP.playState of
    wmppsUndefined,wmppsStopped : begin
                                    if(not(Assigned(maPlayList)))then
                                    begin
                                      WMP.URL:='C:\MaMusique\04 - soft meeting - going back home.mp3';
                                    end
                                    else
                                    begin
                                      WMP.currentPlaylist:=maPlayList;
                                    end;
                                    WMP.controls.play;
                                  end;
    wmppsPaused : begin
                    WMP.controls.play;
                  end;
  end;
end;
```

Testez, et vous verrez, ça fonctionne!

IV-C-3 - Supprimer un fichier de la PlayList

Nous avons vu comment ajouter un fichier, maintenant, voyons comment en supprimer un. Nous repassons donc sur la Form2 et implémentons le code suivant dans le OnClick du *BtnSupprimer*.

```
procedure TForm2.BtnSupprimerClick(Sender: TObject);
var
  chanson:WMPLib_TLB.IWMPMedia;
begin
  // Si la liste n'est pas vide et qu'on a sélectionné un Item
  if((ListBoxPlayList.Items.Count>0) and (ListBoxPlayList.ItemIndex<>-1))then
  begin
    // On récupère l'item en question
    chanson:=Form1.maPlayList.Item[ListBoxPlayList.ItemIndex];
    // On le supprime
    Form1.maPlayList.removeItem(chanson);
    // On le retire de la ListBox
    ListBoxPlayList.Items.Delete(ListBoxPlayList.ItemIndex);
  end;
end;
```

IV-C-4 - Jouer le morceau suivant ou précédent

Il nous reste qu'à coder la partie correspondant au bouton suivant et précédent.
Je vous mets directement le code, rien de bien compliqué.

```
procedure TForm1.BtnPrecedentClick(Sender: TObject);
begin
  WMP.controls.previous;
end;

procedure TForm1.BtnSuivantClick(Sender: TObject);
begin
  WMP.controls.next;
end;
```

IV-C-5 - Mode alétoire, répéter

Ces modes permettent d'activer le mode *Shuffle* et *Loop* du TWindowsMediaPlayer.
Nous ajoutons deux TCheckBox (CheckBoxRandom,CheckBoxRepeat) pour la gestion de ces modes.
L'interface ressemble à celle-ci:

Pour activer ou non ces modes, il faut utiliser la propriété **SetMode** dans *WMP.Settings* et la propriété **GetMode** pour avoir l'état du mode.

```
procedure TForm1.CheckBoxRandomClick(Sender: TObject);
begin
  WMP.settings.setMode('shuffle',CheckBoxRandom.Checked);
end;

procedure TForm1.CheckBoxRepeatClick(Sender: TObject);
begin
  WMP.settings.setMode('loop',CheckBoxRepeat.Checked);
end;
```

V - Téléchargement

Le projet complet est disponible en téléchargement en cliquant **ici**.
Je rappelle tout de même que le projet a été créé sous Delphi 2009. La compatibilité avec les versions précédentes de Delphi devraient cependant être assurée.

VI - Conclusion

Ce tutoriel aborde les fonctionnalités de base du lecteur WindowsMediaPlayer. Je n'ai pas couvert toutes les possibilités mais vous avez déjà un aperçu de ce que l'on peut faire et également savoir dans quelle direction chercher pour utiliser telle ou telle propriété.
De même, ce tutoriel n'a pas couvert la lecture de fichiers vidéos puisque la gestion se fait quasiment de la même façon que pour un fichier audio.
Dans tous les cas, j'espère que ce tuto vous a plu.

VII - Remerciement

Je tiens à remercier **Aka Guymelef** pour m'avoir aidé à améliorer la qualité de ce tutoriel, ainsi que **Pedro** et **Laurent Dardenne** pour la relecture.