【B6】Delphiテクニカルセッション

EMBARCADERO TECHNOLOGIES®

DEVELOPER CAMP

Delphi 2010 で探る！ Windows 7 新機能

株式会社シリアルゲームズ　取締役
細川　淳

アジェンダ

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Windows 7 概要
- Windows Vista での変更点
  - UAC
  - フォルダパス・環境変数
- Windows 7 の新機能
  - タスクバー
  - ジャンプリスト
  - ジェスチャー・マルチタッチ
  - Direct2D
  - リボン
  - ライブラリ
  - センサーAPI

青字は、VCL でサポート済み。
このセッションでは VCL でサポートされない
赤字の Win32 API を紹介します。

Windows 7 概要

Windows 7 とは？

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Windows 7 とは？
  - Windows Vista のマイナーバージョンアップ版
    - Vista のバージョン番号は 6.0
    - 7 のバージョン番号は **6.1**
      - SysUtils.CheckWin32Version(6, 1); // でバージョンチェック可能
  - カーネルなどのコア部分は、Windows Vista とほぼ同じ
    - Longhorn カーネル
  - UI 部を修正し Vista に比べ、ストレスフリーになっている
    - 起動・終了時間の短縮
      - サービスの段階的ロードなど
    - デスクトップ・ウィンドウ・マネージャ(DWU)
      - DirectX 10.1 の使用
      - Window 管理手法の変更

Windows 7 と Vista の違い

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- ユーザーの観点
  - UACが、あまり「うざく」無くなった。
  - Aero がちょっとだけリッチになった。
  - 操作性が上がった？
    - 新しいタスクバー
    - エアロスナップ
- アプリケーションの観点
  - Vista 対応のアプリであれば、そのまま動いてしまうほど互換性が高く、違いはほとんどない。
  - 違いの多くは、新しい要素
    - 新しいタスクバー
    - ジェスチャー・マルチタッチ
    - Direct 系 API の追加

Windows Vista での変更点

UAC

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Vista から「ユーザーアカウント制御」が加わりました。
  - Xp では
    - Administrators グループに所属するユーザー ＝ Administrator
  - Vista / 7 では
    - Administrators グループに所属するユーザー ≠ Administrator

- UAC に対応する
  - manifest ファイルを変更する
    - アプリケーションが必要なレベルを指定する
  - UAC による制限を明示する
    - ボタンに「盾」アイコンを付与する
  - ShellExecute の runas を使用する
    - UAC に関係するコードを別アプリとして起動する

- 詳しくは過去のセッション資料を参照
  - http://edn.embarcadero.com/jp/article/images/34159/devcamp04_g4.pdf

UAC に対応するマニフェスト

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

Delphi 2010 が自動的に付与する manifest は対応済み

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity
    type="win32"
    name="CodeGear RAD Studio"
    version="14.0.3539.24502"
    processorArchitecture="*"/>
  <dependency>
    <dependentAssembly>
      <assemblyIdentity
        type="win32"
        name="Microsoft.Windows.Common-Controls"
        version="6.0.0.0"
        publicKeyToken="6595b64144ccf1df"
        language="*"
        processorArchitecture="*"/>
    </dependentAssembly>
  </dependency>
```

本来は自分の組織名・アプリケーション名を入れる

```
  <trustInfo xmlns="urn:schemas-microsoft-com:asm.v3">
    <security>
      <requestedPrivileges>
        <requestedExecutionLevel
          level="asInvoker"
          uiAccess="false"/>
      </requestedPrivileges>
    </security>
  </trustInfo>
</assembly>
```

ここでアプリケーションに
必要なレベルを入れる

UAC で管理者権限を求めるようにする

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

1.管理者権限を要求する manifest を作る

manifest.xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
  <assemblyIdentity
    type="win32"
    name="会社名やアプリケーション名"
    version="1.0.0.0"
    processorArchitecture="X86"/>

  <description>アプリケーションの説明文</description>

  <dependency>
    <dependentAssembly>
      <assemblyIdentity
        type="win32"
        name="Microsoft.Windows.Common-Controls"
        version="6.0.0.0"
        publicKeyToken="6595b64144ccf1df"
        language="*"
        processorArchitecture="*"/>
    </dependentAssembly>
  </dependency>

   <trustInfo xmlns="urn:schemas-microsoft-com:asm.v2">
      <security>
        <requestedPrivileges>
          <requestedExecutionLevel level="requireAdministrator" uiAccess="false" />
        </requestedPrivileges>
      </security>
   </trustInfo>
</assembly>
```

level を requireAdministrator にする

UAC で管理者権限を求めるようにする

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

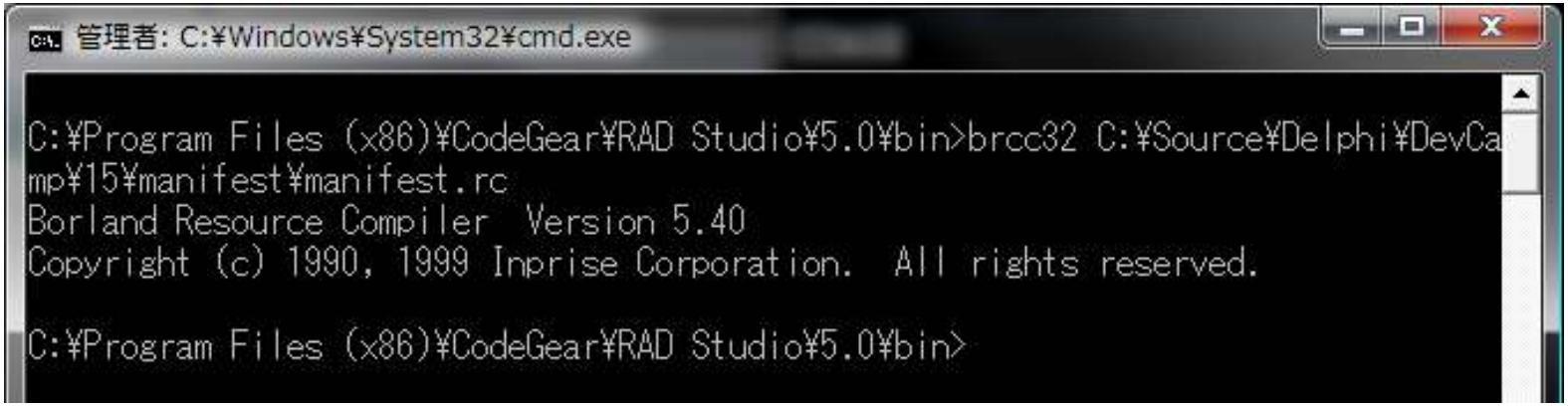
2.RC ファイルを作る

manifest.rc

```
#define CREATEPROCESS_MANIFEST_RESOURCE_ID 1
#define RT_MANIFEST 24

CREATEPROCESS_MANIFEST_RESOURCE_ID RT_MANIFEST "manifest.xml"
```

3.RC ファイルをコンパイルして RES ファイルを作る

- %RADStudio%¥5.0¥bin¥brcc32.exe の引数に RC ファイルを渡す



UAC で管理者権限を求めるようにする

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

4.RAD Studio 2010 のプロジェクトオプションから
「ランタイムテーマを有効にする」を外す

5.できあがった RES ファイルをアプリケーションに組み込む

```
implementation

{$R *.dfm}
{$R manifest.res}
```

6.ビルド

アイコンに盾が付く

フォルダパス

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Xp と、Vista/7 ではフォルダパスが異なります。
  - 例えば

	Xp	Vista / 7
ユーザープロファイル	C:¥Documents and Settings	C:¥Users
アプリケーションデータ (Roaming あり)	C:¥Documents and Settings¥ユーザー名 ¥Application Data	C:¥Users¥ユーザー名¥AppData¥Roaming

- フォルダパスは決めうちしない
  - SHGetSpecialFolderLocation API を使う
  - 環境変数を使う

SHGetSpecialFolder・環境変数

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

SHGetSpecialFolder

```
uses
  ShlObj, ActiveX;

function GetSpecialFolder(const iFolder: DWORD): String;
var
  IDL: PItemIDList;
begin
  Result := '';

  SHGetSpecialFolderLocation(
    Application.Handle,
    iFolder,
    IDL);
  try
    Result := StringOfChar(#0, MAX_PATH);

    if (IDL <> nil) and (SHGetPathFromIDList(IDL, PChar(Result))) then
      Result := Trim(Result);
  finally
    CoTaskMemFree(IDL);
  end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  Label1.Caption := GetSpecialFolder(CSIDL_APPDATA);
end;
```

環境変数

```
function GetWindowsDir: String
begin
  SetLength(Result, MAX_PATH);

  Len :=
    ExpandEnvironmentStrings(
      PChar('%windir%'),
      PChar(Result),
      Length(Result));

  if (Len > 0) then
    Result := Copy(Result, 1, Len)
  else
    Result := '';
end;
```

環境変数を指定する

CSIDL_APPDATA は
アプリケーションデータを表す

Windows 7 の新機能

タスクバー

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- Windows 7 ではタスクバーに大幅な改良が加えられました
  - 具体的には以下の要素が追加されています
    - タスクバー・ボタン
    - タスクバー・オーバーレイアイコン
    - タスクバー・プログレスバー
    - ジャンプリスト
  - アプリケーションは起動時に、アプリケーション固有の値を設定することで、タスクバーのグループ化を制御できるようになりました

タスクバーのグループ化

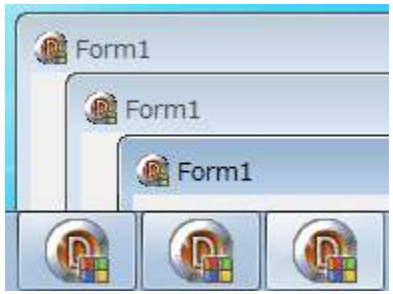
EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
ShlObj.SetCurrentProcessExplicitAppUserModelID(PChar('アプリケーション固有の値'));
```

SetCurrentProcessExplicitAppUserModelID API を使うことで、アプリケーションのグループ化を制御出来るようになりました。
同じ ID を指定することで下の図のようにタスクバーがグループ化されます。


```
procedure TForm1.FormCreate(Sender: TObject);
begin
  SetCurrentProcessExplicitAppUserModelID('SERIALGAMES.DevCamp.AppID.1');
end;
```

逆に、同じアプリケーションでも、違うIDを指定すれば、グループ化はされません。




```
procedure TForm1.FormCreate(Sender: TObject);
begin
  Randomize;
  SetCurrentProcessExplicitAppUserModelID(PChar(IntToStr(Random(100))));
end;
```

タスクバーのグループ化

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- AppUserModelID（以下、AppID）は、次のように設定することが推奨されています。
  - CompanyName.ProductName.SubProduct.VersionInformation
    - 最大 128 byte

- AppID を指定しない場合は、Windows がアプリケーションの名前などから、自動的に設定します。
  - パフォーマンスの観点から、AppUserModelID を定めることが強く求められています。

- AppID は、タスクバーのグループ化だけではなく、後述のジャンプリストなど、タスクバー関連の API と深く関わってきます。

ITaskbarList3 インターフェース

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

※画像は MSDN より引用

タスクバー・ボタン

タスクバー・オーバーレイアイコン

タスクバー・プログレス

Windows 7 では、上図のように、タスクバーに表示されているアプリケーションのボタンに様々な効果を加えることができます。

これらの機能は

ITaskbarList3 インターフェース

によって提供されています。

タスクバーボタンを登録する

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
procedure TfrmTaskbarSample.SetTaskbarButton;
var
  TL: ITaskbarList3;
  Buttons: array [0.. 1] of TThumbButton;
begin
  if
    (
      Succeeded(
        CoCreateInstance(
          CLSID_TaskbarList,
          nil,
          CLSCTX_ALL,
          IID_ITaskbarList3,
          TL)
      )
    )
  then begin
    // アイコンを表示するボタン
    with Buttons[0] do begin
      dwMask := THB_ICON or THB_TOOLTIP or THB_FLAGS;
      iId := 0;
      szTip := 'アイコンボタン';
      dwFlags := THBF_ENABLED;

      hIcon := LoadSysIcon(IDI_INFORMATION);
end;
```

```
    // ビットマップを表示するボタン
    TL.ThumbBarSetImageList(Handle, ImageList1.Handle);

    with Buttons[1] do begin
      dwMask := THB_BITMAP or THB_TOOLTIP or THB_FLAGS;
      iId := 1;
      szTip := 'ビットマップボタン';
      dwFlags := THBF_ENABLED;

      iBitmap := 0;
    end;

    // 実行
    TL.ThumbBarAddButtons(Handle, Length(Buttons), @Buttons);
  end;
end;
```

※LoadSysIcon

```
function TfrmTaskbarSample.LoadSysIcon(const iKind: PChar): HICON;
begin
  Result :=
    LoadImage(0, iKind, IMAGE_ICON, 0, 0, LR_DEFAULTSIZE or LR_SHARED);
end;
```

タスクバーボタンのクリックを受け取る

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

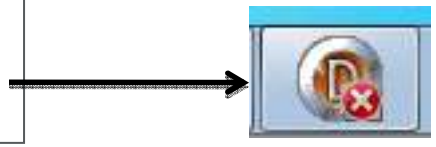
```
procedure TfrmTaskbarSample.WMCommand(var iMsg:
TWMCommand);
begin
  inherited;

  if (iMsg.NotifyCode = THBN_CLICKED) then
    case iMsg.ItemID of
      0:
        ShowMessage('アイコンボタンがクリックされた！');

      1:
        ShowMessage('ビットマップボタンがクリックされた！
');
    end;
end;
```

タスクバーオーバーレイアイコン

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

アイコンを設定

```
procedure TfrmTaskbarSample.SetOverlayIcon;
var
  TL: ITaskbarList3;
begin
  if
    (
      Succeeded(
        CoCreateInstance(
          CLSID_TaskbarList,
          nil,
          CLSCTX_ALL,
          IID_ITaskbarList3,
          TL)
      )
    )
  then
    TL.SetOverlayIcon(Handle, LoadSysIcon(IDI_ERROR), 'ERROR アイコン！');
end;
```



アイコンを削除

```
TL.SetOverlayIcon(Handle, 0, nil);
```

タスクバープログレス

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

初期化

```
procedure TfrmTaskbarSample.InitProgress;
begin
  CoCreateInstance(
    CLSID_TaskbarList,
    nil,
    CLSCTX_ALL,
    IID_ITaskbarList3,
    FTaskbarList);
end;
```

設定出来るステータス

TBPF_NOPROGRESS	プログレス無し
TBPF_INDETERMINATE	周期的に動くバー
TBPF_NORMAL	通常のプログレス
TBPF_ERROR	エラー状態
TBPF_PAUSED	一時停止状態

進行状況の設定

```
procedure TfrmTaskbarSample.SetProgress(const iCompleted: Integer);
begin
  FTaskbarList.SetProgressState(Handle, TBPF_NORMAL);
  FTaskbarList.SetProgressValue(Handle, iCompleted, 100);
end;
```

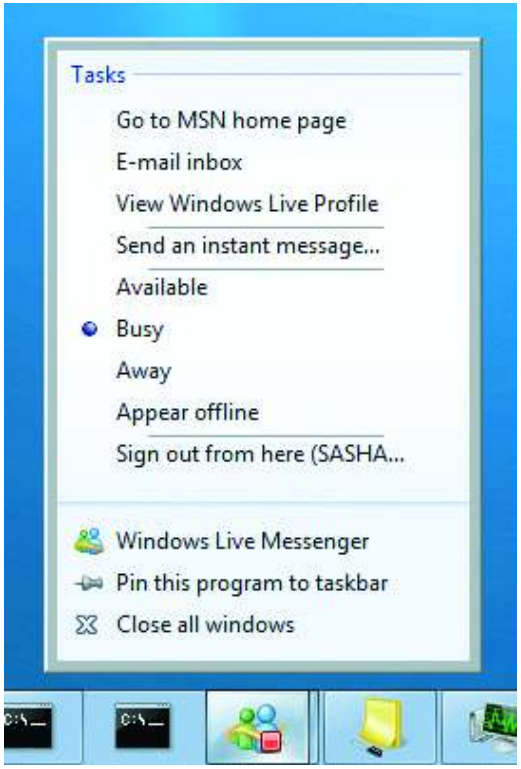
一時停止

```
procedure TfrmTaskbarSample.PauseProgress;
begin
  FTaskbarList.SetProgressState(Handle, TBPF_PAUSED);
end;
```

エラー表示

```
procedure TfrmTaskbarSample.ErrorProgress;
begin
  FTaskbarList.SetProgressState(Handle, TBPF_ERROR);
end;
```

ジャンプリスト



※画像は MSDN より引用

Windows 7 では、タスクバーに表示されているアプリケーションを右クリックすると「ジャンプリスト」というショートカット群を定義できるようになりました。

ジャンプリストには大きく分けて

・タスク
・カテゴリ

の2つがあります。

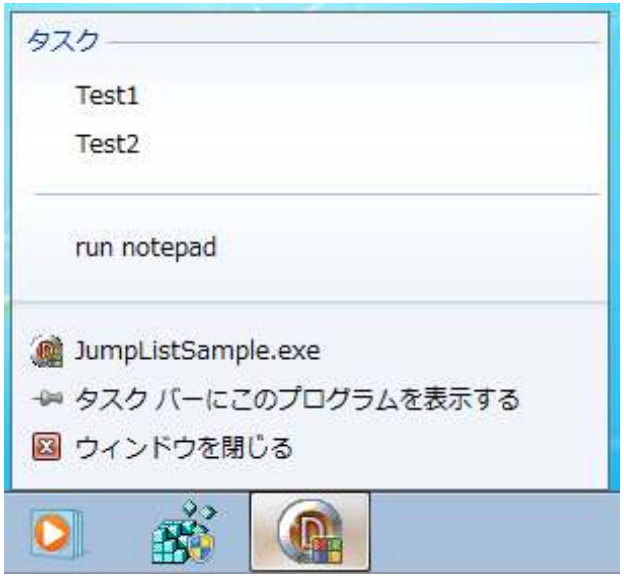
タスクは、デフォルトの「タスク」というカテゴリに表示される特別なカテゴリです。

カテゴリは、アプリケーションで自由に設定出来る項目で、カテゴリ名と、そこに表示するファイルなどを指定します。

タスクの追加

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- 今回のセッションでは、タスクに項目を追加する方法を紹介します。



作成するアプリケーション

ジャンプリストは、Win32 API から扱うと非常に複雑です。
順を追って説明します。

ちなみに.NET Framework 4.0 では、ジャンプリストを簡単に扱うクラスが提供されています。
VCL でもサポートされるといいのに……。

なお、このセッションで書いたコードは MSDN からの参照が多分に含まれています。

下準備:必要な API のロード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

ジャンプリストを操作するために下記の3つの API が必要です。

- InitPropVariantFromString
- InitPropVariantFromBoolean
- PropVariantClear

これらの API を使うために注意するべき事があります。

MSDN では InitPropVariantFrom* は、propsys.dll 6.0 以降で提供されていると書いてありますが、Windows 7 に搭載されている propsys.dll 7.0 では、下記の2つの API が廃止されています。

- InitPropVariantFromString
- InitPropVariantFromBoolean

下準備：必要な API のロード

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

では、どうすればよいのかと言うと

- InitPropVariantFromString は InitPropVariantFromStringVector で代用
- InitPropVariantFromBoolean は InitPropVariantFromBooleanVector で代用

と、ぞれぞれ Vector 系 API で代用します。

これらの処理をするユニット uPropSysForJumplist.pas を次ページから見ていきます。

uPropSysForJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
unit uPropSysForJumpList;

interface

uses
  Windows, ActiveX;

type
  PPWideChar = ^PWideChar;
  PBOOL = ^BOOL;

  TInitPropVariantFromStringVector =
    function(
      prgsz: PPWideChar;
      cElems: ULONG;
      ppropvar: PPropVariant): HRESULT; stdcall;

  TInitPropVariantFromString =
    function(
      psz: PWideChar;
      ppropvar: PPropVariant): HRESULT; stdcall;

  TInitPropVariantFromBooleanVector =
    function(
      prgf: PBOOL;
      cElems: ULONG;
      ppropvar: PPropVariant): HRESULT; stdcall;

  TInitPropVariantFromBoolean =
    function(
      fVal: BOOL;
      ppropvar: PPropVariant): HRESULT; stdcall;

  TPropVariantClear = function(pVar: PPropVariant): HRESULT;
stdcall;
```

各関数の型を宣言しています。
このユニットでは、万一 InitPropVariantFrom* 系が使えた場合に備えて InitPropVariantFrom* 系の API の型も宣言しています。

uPropSysForJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
var
  InitPropVariantFromStringVector: TInitPropVariantFromStringVector = nil;
  InitPropVariantFromString: TInitPropVariantFromString = nil;
  InitPropVariantFromBooleanVector: TInitPropVariantFromBooleanVector = nil;
  InitPropVariantFromBoolean: TInitPropVariantFromBoolean = nil;
  PropVariantClear: TPropVariantClear = nil;

function PropSysLoaded: Boolean;

implementation

const
  CPropSysLib = 'propsys.dll';
  COle32Lib = 'Ole32.dll';

var
  GPropSysLib: HMODULE = 0;
  GOle32Lib: HMODULE = 0;
```

InitPropVariant 系は propsys.dll から
PropVariantClear だけは ole32.dll から
取り出します

uPropSysForJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
function _InitPropVariantFromString(
  psz: PWideChar;
  ppropvar: PPropVariant): HRESULT; stdcall;
begin
  Result := ERROR_INVALID_FUNCTION;

  if (Assigned(InitPropVariantFromStringVector)) then
    Result := InitPropVariantFromStringVector(@psz, 1, ppropvar);
end;

function _InitPropVariantFromBoolean(
  fVal: BOOL;
  ppropvar: PPropVariant): HRESULT; stdcall;
begin
  Result := ERROR_INVALID_FUNCTION;

  if (Assigned(InitPropVariantFromBooleanVector)) then
    Result := InitPropVariantFromBooleanVector(@fVal, 1, ppropvar);
end;

function PropSysLoaded: Boolean;
begin
  Result :=
    Assigned(InitPropVariantFromString) and
    Assigned(InitPropVariantFromBoolean) and
    Assigned(PropVariantClear);
end;
```

この2つの関数は、
InitPropVariantFrom* を
エミュレートします。

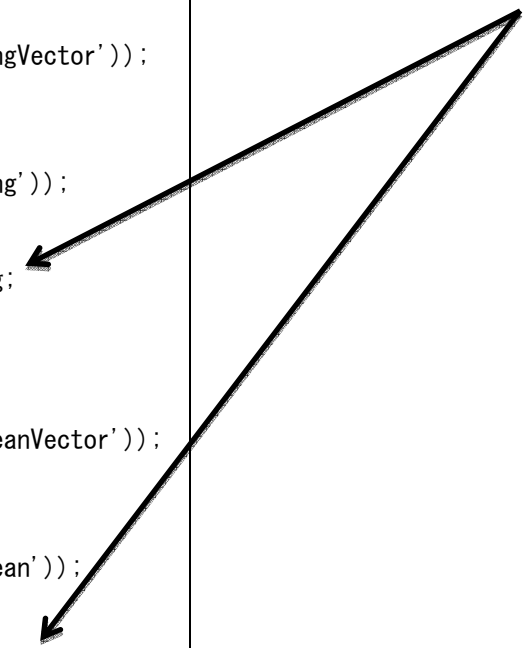
uPropSysForJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
procedure LoadPropSys;
begin
  GPropSysLib := LoadLibrary(CPropSysLib);

  if (GPropSysLib <> 0) then begin
    // InitPropVariantFromString
    InitPropVariantFromStringVector :=
      TInitPropVariantFromStringVector(
        GetProcAddress(GPropSysLib, 'InitPropVariantFromStringVector'));

    InitPropVariantFromString :=
      TInitPropVariantFromString(
        GetProcAddress(GPropSysLib, 'InitPropVariantFromString'));

    if (not Assigned(InitPropVariantFromString)) then
      InitPropVariantFromString := _InitPropVariantFromString;

    // InitPropVariantFromBoolean
    InitPropVariantFromBooleanVector :=
      TInitPropVariantFromBooleanVector(
        GetProcAddress(GPropSysLib, 'InitPropVariantFromBooleanVector'));

    InitPropVariantFromBoolean :=
      TInitPropVariantFromBoolean(
        GetProcAddress(GPropSysLib, 'InitPropVariantFromBoolean'));

    if (not Assigned(InitPropVariantFromBoolean)) then
      InitPropVariantFromBoolean := _InitPropVariantFromBoolean;
  end;
```

実際にロードします。

InitPropVariantFrom* が無かったときは
先ほどの関数で代用するようにします。



uPropSysForJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
  GOle32Lib := LoadLibrary(COle32Lib);

  if (GOle32Lib <> 0) then begin
    // PropVariantClear
    PropVariantClear :=
      TPropVariantClear(GetProcAddress(GOle32Lib, 'PropVariantClear'));
  end;
end;

procedure UnloadPropSys;
begin
  if (GPropSysLib <> 0) then
    FreeLibrary(GPropSysLib);

  if (GOle32Lib <> 0) then
    FreeLibrary(GOle32Lib);
end;

initialization
begin
  LoadPropSys;
end;

finalization
begin
  UnloadPropSys;
end;
```

PropVariantClear のロードです

ジャンプリストタスク

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

ジャンプリストにタスクを追加するには大体以下のような流れになります。

- PropSys 系関数のロード
- IDestionationList の作成
- AppID の設定
- IDestionationList の更新スタート
- 必要な情報を設定した IShellLink の作成
- IObjectCollection に作成した IShellLink を追加
- IDestionationList コミット

この流れに沿ってジャンプリストを作成するクラスを作成します。
作成するクラスを、どのように使うかを先にお見せします

TJumpList の使い方

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
procedure TForm1.Button1Click(Sender: TObject);
var
  ExeName: String;
begin
  ExeName := Application.ExeName;

  with TJumplist.Create(ExeName) do
    try
      AddTask(ExeName, 'Test1', '-test1');
      AddTask(ExeName, 'Test2', '-test2');
      AddTask('', '', '');
      AddTask('notepad.exe', 'run notepad', '');

      CreateJumplist;
    finally
      Free;
    end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
  SetCurrentProcessExplicitAppUserModelID(PChar(Application.ExeName));

  if (ParamStr(1) = '-test1') then
    ShowMessage('-test1 で起動された！');

  if (ParamStr(1) = '-test2') then
    ShowMessage('-test2 で起動された！');
end;
```

AddTask でタスクを追加し、最後に CreateJumplist を呼び出します。

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
unit uJumpList;

interface

uses
  Classes, ShlObj;

type
  TJumplist = class
  private
    // Variables
    FAppID: String;
    FParams: TStringList;
    FCategories: TStringList;
    // Methods
    function AddTasksToList(const iCDL: ICustomDestinationList): Boolean;
    function CreateShellLink(const iExe, iTitle, iArg: String): IShellLink;
    function CreateSeparatorLink: IShellLink;
  public
    // Constructor & Destructor
    constructor Create(const iAppID: String); reintroduce;
    destructor Destroy; override;
    // Task
    function AddTask(const iExe, iTitle, iArg: String): Integer;
    procedure DeleteTask(const iIndex: Integer);
    procedure ClearTask;
    procedure GetParams(const iIndex: integer; var oExe, oTitle, oArg: String);
    // Jumplist
    function CreateJumplist: Boolean;
    function DestroyJumplist: Boolean;
  end;
```

TJumplist の宣言部です。

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
constructor TJumplist.Create(const iAppID: String);
begin
  inherited Create;

  FAppID := iAppID;

  FCategories := TStringList.Create;
  FParams := TStringList.Create;
end;
```

コンストラクタで AppID を貰います

なお、前ページに出てきた ShlObj ユニット以外に赤字で示したユニットが必要です。

```
uses
  Windows, SysUtils, Ole2, ShellAPI, ActiveX, ObjectArray, PropSys, PropKey,  uPropSysForJumpList;
```

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
function TJumplist.CreateShellLink(
  const iExe, iTitle, iArg: String): IShellLink;
var
  HR: HRESULT;
  SL: IShellLink;
  PS: IPropertyStore;
  PropVar: TPropVariant;
begin
  Result := nil;

  HR :=
    CoCreateInstance(
      CLSID_ShellLink,
      nil,
      CLSCTX_INPROC_SERVER,
      IID_IShellLink,
      SL);

  if (Failed(HR)) then
    Exit;

  HR := SL.SetPath(PChar(iExe));

  if (Failed(HR)) then
    Exit;

  HR := SL.SetArguments(PChar(iArg));

  if (Failed(HR)) then
    Exit;
```

```
  HR := SL.QueryInterface(IID_IPropertyStore, PS);

  if (Failed(HR)) then
    Exit;

  HR := InitPropVariantFromString(PChar(iTitle), @PropVar);
  try
    if (Failed(HR)) then
      Exit;

    HR := PS.SetValue(PKEY_Title, PropVar);

    if (Failed(HR)) then
      Exit;

    HR := PS.Commit;

    if (Failed(HR)) then
      Exit;

    HR := SL.QueryInterface(IID_IShellLink, Result);

    if (Failed(HR)) then
      Result := nil;
  finally
    PropVariantClear(@PropVar);
  end;
end;
```

実行ファイルのパス、表示するタイトル、引数を渡して、それらの値を持った IShellLink を作って返します

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
function TJumplist.CreateSeparatorLink: IShellLink;
var
  PS: IPropertyStore;
  HR: HRESULT;
  PropVar: TPropVariant;
begin
  Result := nil;

  HR :=
    CoCreateInstance(
      CLSID_ShellLink,
      nil,
      CLSCTX_INPROC_SERVER,
      IID_IPropertyStore,
      PS);

  if (Failed(HR)) then
    Exit;

  HR := InitPropVariantFromBoolean(True, @PropVar);

  if (Failed(HR)) then
    Exit;

  HR := PS.SetValue(PKEY_AppUserModel_IsDestListSeparator, PropVar);

  if (Failed(HR)) then
    Exit;

  HR := PS.Commit;

  if (Failed(HR)) then
    Exit;

  HR := PS.QueryInterface(IID_IShellLink, Result);

  if (Failed(HR)) then
    Result := nil;
end;
```

こちらはセパレータの IShellLink を返します。

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
function TJumplist.AddTasksToList(
  const iCDL: ICustomDestinationList): Boolean;
var
  HR: HRESULT;
  OC: IObjectCollection;
  OA: IObjectArray;
  SL: IShellLink;
  i: Integer;
  Exe, Title, Arg: String;
begin
  Result := False;

  HR :=
    CoCreateInstance(
      CLSID_EnumerableObjectCollection,
      nil,
      CLSCTX_INPROC,
      IID_IObjectCollection,
      OC);

  if (Failed(HR)) then
    Exit;

  for i := 0 to FParams.Count - 1 do begin
    GetParams(i, Exe, Title, Arg);

    if (Exe <> '') then
      SL := CreateShellLink(Exe, Title, Arg)
    else
      SL := CreateSeparatorLink;

    if (SL = nil) then
      Continue;

    HR := OC.AddObject(SL);

    if (Failed(HR)) then
      Continue;
  end;

  HR := OC.QueryInterface(IID_IObjectArray, OA);

  if (Failed(HR)) then
    Exit;

  Result := Succeeded(iCDL.AddUserTasks(OA));
end;
```

ICustomDestinationList に必要な数だけ IShellLink を追加します。

なお、GetParams は、FParams から Exe, Title, Arg を取得するメソッドです。
Add メソッドで FParams 値を追加します。

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
function TJumplist.CreateJumplist: Boolean;
var
  HR: HRESULT;
  CDL: ICustomDestinationList;
  OARemoved: IObjectArray;
  MinSlots: Cardinal;
begin
  Result := False;

  if (not PropSysLoaded) then
    Exit;

  HR :=
    CoCreateInstance(
      CLSID_DestinationList,
      nil,
      CLSCTX_INPROC_SERVER,
      IID_ICustomDestinationList,
      CDL);

  if (Failed(HR)) then
    Exit;

  HR := CDL.SetAppID(PChar(FAppID));

  if (Failed(HR)) then
    Exit;

  HR := CDL.BeginList(MinSlots, IID_IObjectArray, OARemoved);

  if (Failed(HR)) then
    Exit;

  if (AddTasksToList(CDL)) then
    Result := Succeeded(CDL.CommitList)
  else
    CDL.AbortList;
end;
```

実際にジャンプリストタスクを作るコードです。

uJumpList

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

```
function TJumplist.DestroyJumplist: Boolean;
var
  HR: HRESULT;
  CDL: ICustomDestinationList;
begin
  Result := False;

  if (not PropSysLoaded) then
    Exit;

  HR :=
    CoCreateInstance(
      CLSID_DestinationList,
      nil,
      CLSCTX_INPROC_SERVER,
      IID_ICustomDestinationList,
      CDL);

  if (Failed(HR)) then
    Exit;

  Result := Succeeded(CDL.DeleteList(nil));
end;
```

ジャンプリストタスクを削除するコードです。

参考文献

EMBARCADERO TECHNOLOGIES
DEVELOPER CAMP

- MSDN
  - Windows7対応アプリケーション開発を支援します
    - http://www.microsoft.com/japan/powerpro/developer/default.mspx
  - Introducing The Taskbar APIs
    - http://msdn.microsoft.com/en-us/magazine/dd942846.aspx
  - MSDN Sample Code
    - CustomJumpListSample
      - http://msdn.microsoft.com/en-us/library/dd940352(VS.85).aspx

ご静聴ありがとうございました