Which COM objects to use?

In many situations, it is necessary to enumerate the COM objects that can be used in a certain context. Furthermore we may need information about all COM objects that were specially developed to solve one particular problem.

For example, we are in the middle of developing some COM based plug-in framework for our application. Surely this question will soon come up: "What is the standard way to determine what plug-ins are available for a given host?" Everytime the host app runs, it needs to find out its available (compatible) plug-ins.

In Windows 98/2000 this is where Component Categories come to the rescue. In short, Component Categories are windows way to group several COM objects that shared some common functionality into one category. That way client applications can easily determine which COM objects are available to use. Just like interfaces, coclasses and other COM related stuff, each category is uniquely identified by a GUID (CATID - Category ID - to be precise.

Windows defines two interfaces *ICatRegister* and *ICatInformation* to help developers use the Component Categories service. Both *ICatRegister* and *ICatInformation* are both implemented by a COM coclass defined by *CLSID_StdComponentCategoryMgr*. We use *ICatRegister.RegisterCategories* to register one or more categories. *RegisterCategories* receives two parameters. The first determines how many category will be registered and the second is the array (pointer) to *TCategoryInfo*. *TCategoryInfo* is defined as follows:

```
TCategoryInfo = record
  catid: TGUID; // category ID
  lcid: UINT;   // locale ID, for multi-language support
  szDescription: array[0..127] of WideChar; //category desc
end;
```

To register one specified COM object (class) to one category we use *ICatRegister RegisterClassImplCategories*. *RegisterClassImplCategories* uses three parameters.

1. The CLSID of the coclass that implement the category,
2. The number of categories to be listed and
3. the array of categories (*TCategoryInfo*) itself.

Lastly, for the host application to scan all available COM objects that fit into one category, we use *ICatInformation.EnumClassesOfCategories*. *EnumClassesOfCategories* has five parameters, but we only use three of them. The first is to indicate which category we are interested in The second is the array of categories and the last is the CLSID/GUID enumerator of matching coclasses. The host application will use *GetImplementedClasses* to retrieve the CLSIDs of all COM objects that fit into the category provided. Notice that the code uses *TImplementedClasses* as a container for all CLSIDs retrieved. *TImplementedClasses* is defined simply as array of 256 CLSIDs. I Think that should be enough even for all extreme cases. *RegisterClassImplementation* with parameter *bRegister* set to True is used in the ActiveX DLL *DLLRegisterServer* function. This will register the category and the coclass that implements it in the Component Categories. *RegisterClassImplementation* with parameter *bRegister* set to False is used in the ActiveX DLL *DLLUnregisterServer* function will unregister the coclass that implements the given category.

```
unit uhdshake;

interface

uses
  Windows, ActiveX, ComObj;

type
  TImplementedClasses = array [0..255] of TCLSID;

function GetImplementedClasses(CategoryInfo: TCategoryInfo;
  var ImplementedClasses: TImplementedClasses): integer;

procedure RegisterClassImplementation(
  const CATID, CLSID: TCLSID; const sDescription: string;
  bRegister: boolean);
```

```
implementation

function GetImplementedClasses(CategoryInfo: TCategoryInfo;
  var ImplementedClasses: TImplementedClasses): integer;
var
  CatInfo: ICatInformation;
  Enum: IEnumGuid;
  Fetched: UINT;
begin
  Result := 0;
  CatInfo:= CreateComObject(CLSID_StdComponentCategoryMgr)
    as ICatInformation;
  OleCheck(
    CatInfo.EnumClassesOfCategories(1, @CategoryInfo, 0, nil, Enum)
  );
  if (Enum <> nil) then
  begin
    OleCheck(Enum.Reset);
    OleCheck(
      Enum.Next(High(ImplementedClasses), ImplementedClasses [1], Fetched)
    );
    Result:= Fetched;
  end;
end;

procedure RegisterClassImplementation(const CATID, CLSID: TCLSID;
  const sDescription: string; bRegister: boolean);
var
  CatReg: ICatRegister;
  CategoryInfo: TCategoryInfo;
begin
  CoInitialize(nil);
  CategoryInfo.CATID:= CATID;
  CategoryInfo.LCID := LOCALE_SYSTEM_DEFAULT;  //dummy
  StringToWideChar(
    sDescription, CategoryInfo.szDescription, Length(sDescription) + 1
  );
  CatReg := CreateComObject (CLSID_StdComponentCategoryMgr)
    as ICatRegister;
  if (bRegister) then
  begin
    OleCheck(CatReg.RegisterCategories (1, @CategoryInfo));
    OleCheck(
      CatReg.RegisterClassImplCategories(CLSID, 1, @CategoryInfo)
    );
  end
  else
  begin
    OleCheck(
      CatReg.UnregisterClassImplCategories(CLSID, 1, @CategoryInfo)
    );
    DeleteRegKey(
      'CLSID\' + GuidToString (CLSID) + '\' + 'Implemented Categories'
    );
  end;
  CatReg:= nil;
  CoUninitialize;
end;

end.
```

Author:	Shlomo Abuisak
Contributor:	Shlomo Abuisak
Added:	2009-11-05
Last updated:	2009-11-05