Displaying associated file icons in Delphi

Michaël Van Canneyt

December 27, 2021

Abstract

Showing a list of files in some directory is something one often needs to do. Showing the associated file icon and descriptive text next to the filename can be a little harder. In this article we show how to do this.

1 Introduction

Often, you need to display a list of files to the user. These filenames need not actually exist on disk, they can be names of files inside a .zip archive, or a list of filenames stored in a database. To make it more pleasant and recognisable for the user, it helps if the associated file icon or file description (as displayed in the Explorer) is shown next to the filename.

The operating system has this information available. On Windows, the ShellAPI offers a call to retrieve this information: SHGetFileInfo. In this article, a component is presented that uses this call to fetch 3 kinds of information based on a file extension:

1. Associated File icon
2. File description
3. Mime type

The mime type is useful for instance when you want to serve a file in a webserver or send it by mail.

2 The TFileInfoCollector component

This component will load the necessary information on demand and keeps it in memory in a collection. If the same extension is queried a second time, the information will be retrieved from the collection.

The component can also be used to add the necessary images to an image list: by setting the `ImageList` property to a `TImageList` instance, the component will add any icons it finds to the image list. It will store the index of the image in the list in the collection.

The image list can then be used to show a file type image for instance in a listview or a treeview.

The component is defined as follows:

```
TFileInfoCollector = Class(TComponent)
  Function IndexOfExtension(AExtension : String;
                            CachedOnly : Boolean = False) : Integer;
```

```
  Function FindDescription(AExtension : String;
                           CachedOnly : Boolean = False) : String;
  Function FindExtensionInfo(AExtension : String;
                             CachedOnly : Boolean = False) : TExtensionInfo;

  Property Extensions[AIndex : Integer] : String;
  Property Descriptions[AIndex : Integer] : String;
  Property MimeTypes[AIndex : Integer] : String;
  Property IconHandles[AIndex : Integer] : Thandle;
  Property ImageIndex[AIndex : Integer] : Integer;
  Property InfoCount : Integer;
Published
  Property ImageList : TImageList;
  Property SmallIcons : Boolean;
  Property FreeIconHandles : Boolean;
end;
```

The component exposes 3 methods to fetch file information, based on an extension `AExtension`:

IndexOfExtension Returns the index of the relevant item in the collection.

FindDescription Returns the description of the relevant item in the collection.

FindExtensionInfo Returns the relevant item in the collection.

The search is performed case insensitively. If `CachedOnly` is `True`, the search is only performed in the in-memory collection. If it is `false` (the default) then it will first search the in-memory collection. If it doesn't find the necessary information there, then it will query the OS for the relevant information.

Armed with an index, the necessary information can be retrieved through the various Array properties:

Extensions The file extension.

Descriptions The textual description of the file type.

MimeTypes The MIME type of the file, if available.

IconHandles A handle to an icon (if FreeIconHandles is not `True`).

ImageIndex If ImageList is assigned, the index of the icon in the image list.

InfoCount The number of items in the collection.

The three published properties control the behaviour of the component:

ImageList If set, the component will add any found icons to this imagelist. The ImageIndex array contains the indexes in the image list.

SmallIcons If set, the component will retrieve small icons from the Operating System. The default is to fetch large icons.

FreeIconHandles If set to `True`, the Icon handles returned by the OS will be returned at once. It can be safely set to `True` if the image list is used: the icon is immediatly copied to the image list.

3 Querying the OS

The search methods will search through the collection with file information. If no info is found, and `CachedOnly` is `False`, then the component will query the OS for the relevant info, in the `FetchExtensionInfo` call:

```
function TFileInfoCollector.FetchExtensionInfo(
  AExtension: String): TextensionInfo;

Const
  IconOptions : Array[Boolean] of DWORD
              = (SHGFI_LARGEICON,SHGFI_SMALLICON);

Var
  FileInfo : SHFILEINFO;
  Attr :  DWORD;
  Info : TextensionInfo;
  AnIcon : TIcon;

begin
  Result:=Nil;
  Attr:=SHGFI_ICON or SHGFI_TYPENAME
        or SHGFI_USEFILEATTRIBUTES or IconOptions[SmallIcons];
  if (SHGetFileInfo(PChar('*'+AExtension), FILE_ATTRIBUTE_NORMAL,
                    FileInfo,SizeOf(FileInfo),Attr)<>0) then
    begin
    Info:=FExtensions.Add as TextensionInfo;
    Info.Extension:=AExtension;
    Info.Description:=FileInfo.szTypeName;
    Info.hIcon:=FileInfo.hIcon;
    Result:=Info;
    if Assigned(ImageList) then
      begin
      AnIcon:=TIcon.Create;
      try
        AnIcon.Handle:=Info.hIcon;
        Info.ImageIndex:=ImageList.AddIcon(anIcon);
      finally
        if FreeIconHandles then
          Info.hIcon:=0
        else
          AnIcon.Handle:=0;
        AnIcon.Free;
      end;
      end
    else
      begin
      Info.ImageIndex:=-1;
      if FreeIconHandles then
        begin
        DestroyIcon(Info.hIcon);
        Info.hIcon:=0;
        end;
      end;
```

```
    if FRegistry.OpenKeyReadOnly(AExtension) then
      Info.MimeType:=Fregistry.ReadString('Content Type');
    end;
end;
```

The code is pretty straightforward. If the `SHGetFileInfo` call succeeds, a new item is added to the collection, and the icon is copied to the image list. Depending on the `FreeIconHandles` property, the icon handle is freed. The last thing that is done is read the mime type from the registry: known extensions are present as keys below `HKEY_CLASSES_ROOT`, and the actual Mime Type (if present) is in the string `Content Type` below the extension key.

4 Using the component

To demonstrate the component, a small application can be created, which allows the user to select a directory, and at the push of a button, the files in the directory are listed, with their associated icon, mime type and description.

In order to avoid having to install a package in order to run the demo, the `TFileInfoCollector` instance is created in the `OnCreate` method of the form:

```
procedure TMainForm.FormCreate(Sender: TObject);
begin
  FIL:=TFileInfoCollector.Create(Self);
  Fil.ImageList:=ImageList1;
  Fil.SmallIcons:=True;
  BEDir.Text:=ExtractFilePath(ParamStr(0));
  FetchFiles;
end;
```

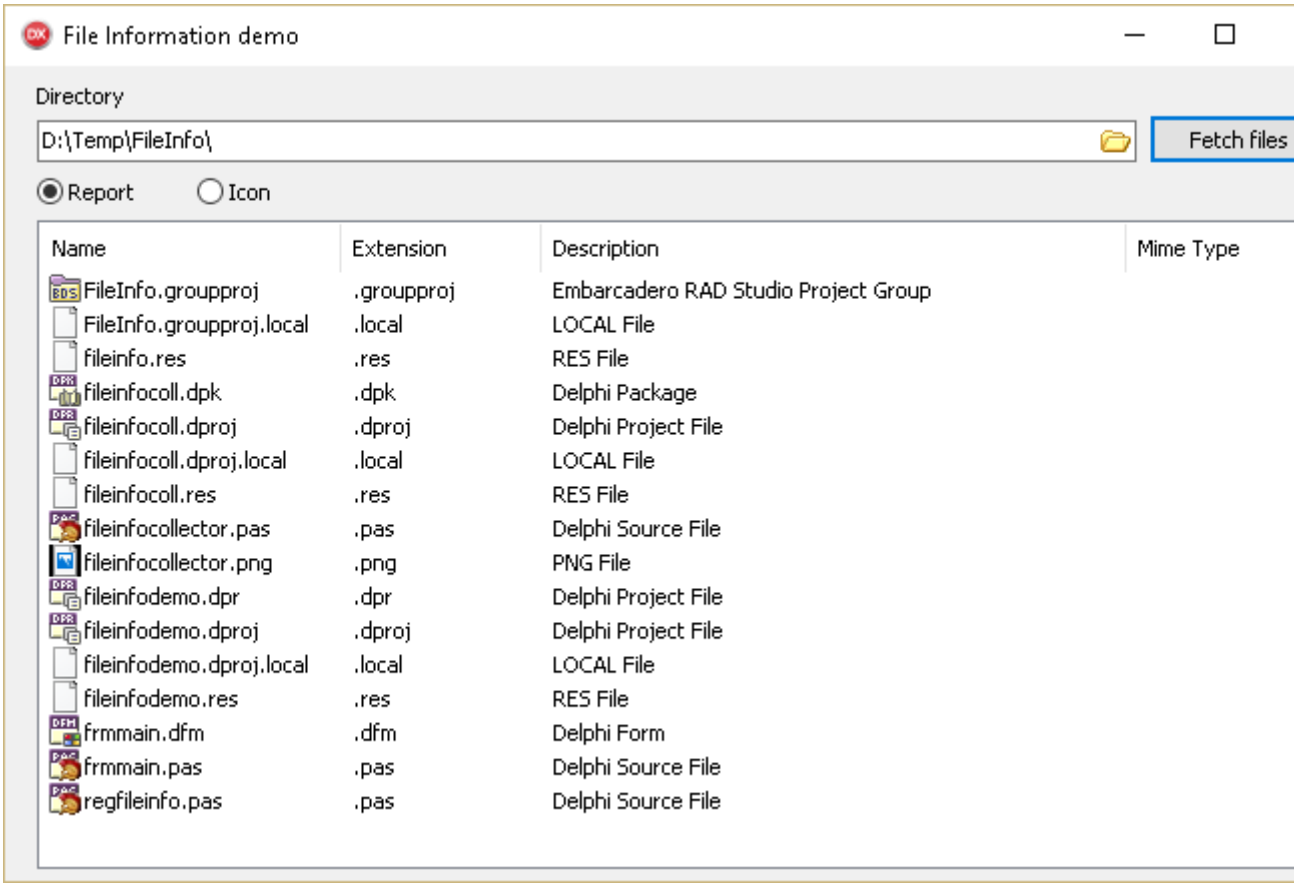
After setting the necessary properties for the component (ImageList and SmallIcons), the contents of the directory in which the binary lives, are displayed using the `FetchFiles` method, which is a simple `FindFirst/FindNext` loop:

```
procedure TMainForm.FetchFiles;

var
  Info : TSearchRec;
  anItem : TListItem;
  Ext : String;
  I : integer;

begin
  LVDir.Items.Clear;
  If FindFirst(BEDir.Text+PathDelim+'*.*',0,Info)=0 then
    try
      Repeat
      Ext:=ExtractFileExt(Info.Name);
      AnItem:=LVDir.Items.Add;
      AnItem.Caption:=Info.Name;
      I:=FIL.IndexOfExtension(ext,false);
      if (I<>-1) then
         begin
```

Figure 1: The file information collector in action




```
          AnItem.ImageIndex:=Fil.ImageIndex[I];
          AnItem.SubItems.Add(Ext);
          AnItem.SubItems.Add(Fil.Descriptions[i]);
          AnItem.SubItems.Add(Fil.MimeTypes[i]);
          end;
      until (FindNext(Info)<>0);
    finally
      FindClose(Info);
    end;
end;
```

All files are added to the listview. If `IndexOfExtension` returns a valid index, then the additional information is copied to the subitems of the list item, so they can be displayed.

The result of all this can be seen in figure 1 on page 5.

5 conclusion

Using the icons which the operating system shows when displaying files is not hard, as can be seen in the small code snippets displayed here. To make things work a bit more optimal,

a component has been presented which caches the results of querying the OS; For optimal convenience an imagelist can be filled to make displaying the icon in controls that use an image list (such as a listview or treeview). The component can probably be improved by having 2 image lists: one for small and one for large images. This improvement is left as an exercise to the reader.