



В этой статье мы создадим компонент **TFindFile**, в котором инкапсулирован код с функциями **FindFirst**, **FindNext** и **FindClose**

. Идея состоит в том, чтобы создать невизуальный компонент Delphi, который заполняет объект

TListString

списком файлов (полный путь + имя файла) по некоторой существующей маске.

Естественно, мы должны получить наш компонент из **TComponent**.

Свойства TFindFile

Вот свойства компонента **TFindFile**

- **FileAttr** - инкапсулированный параметр **Attr**, который используется в функции **FindFirst**

. Например, чтобы искать и скрытые файлы и только для чтения, установите и *ffaReadOnly*

и *ffaHidden*

в **True**

- **FileMask** - используется, чтобы установить маску имени файла, включая подстановочные символы.
- **InSubFolders** - **True**, если мы хотим искать файлы рекурсивно во всех подкаталогах данного пути, **False** - иначе.
- **Path** - определяет каталог, в котором мы хотим искать файлы.
- **Name** и **Tag** - получены от **TComponent**.

Методы TFindFile

Единственная работа компонента **TFindFile** - это заполнить некоторый объект **TStringList**

списком файлов, который соответствует некоторым критериям, поэтому он имеет единственный метод

SearchForFiles

. Выполнение его подобно:

```
function TFindFile.SearchForFiles: TStringList; begin  s.Clear;  try  FileSearch(Path) ;  
finally  Result := s;  end; end;
```

Обратите внимание: переменная **s** (частная для компонента) имеет тип **TStringList**.

Процедура

SearchForFiles

вызывает процедуру

FileSearch

(тоже частная). Процедура

FileSearch

заполняет (рекурсивно) переменную

s

списком всех найденных файлов.

TFindFile в действии

Чтобы увидеть компонент в действии, запустите новый проект Delphi, поместите один

компонент **TListBox**, один **TButton** и конечно один **TFindFile** на форму. При нажатии на кнопку заполняется свойство

Items

у

TListBox

а всеми файлами, которые найдены компонентом

TFindFile

. Только одна строка кода, вот и все, что нужно:

```
ListBox1.Items := FindFile1.SearchForFiles;
```

Естественно, все свойства компонента **TFindFile** первоначально были установлены во время проектирования. Когда Вы опускаете компонент

TFindFile

на форму, свойство

Path

указывает на текущий каталог, маска файла равна

.

, а свойство

InSubFolders

установлено в

False

по умолчанию.

Обратите внимание: функция **SearchForFiles** заполняет объект **StringList**, в дополнение к

ListBox

мы можем использовать

Memo

(свойство

Lines

) или

TComboBox

(свойство

Items

).

Создание и выполнение TFindFile во время выполнения

Как и любой другой компонент Delphi, компонент **TFindFile** может быть создан,

использоваться и разрушен во время выполнения. Чтобы создать компонент **TFindFile**

во время выполнения, можно использовать следующий код (если предположить, что мы имеем

TMemo

по имени

Memo1

):

```
uses FindFile; { ... } procedure TfrMain.Button2Click(Sender: TObject) ; var FFile : TFindFile; begin FFile := TFindFile.Create(nil) ; try FFile.FileAttr := [ffaAnyFile]; FFile.InSubFolders := True; FFile.Path := ExtractFilePath(ParamStr(0)) ; FFile.FileMask := '*.pas'; Memo1.Lines := FFile.SearchForFiles; finally FFile.Free; end; end;
```

Эта часть кода создает компонент **TFindFile** во время выполнения и заполняет компонент **Memo** всеми файлами ***.pas**, которые найдены во всех подкаталогах запущенного приложения.

Примечание 1: Если Вы посмотрите на код **TFindFile**, Вы заметите, что каждый раз, когда Вы пытаетесь изменить свойство

Path

(процедура

SetPath

): если указанный каталог не существует, свойство

Path

восстанавливается.

Примечание 2: Если Вы хотите определить к какому каталогу принадлежит данная папка, установите **'.'** в свойстве **Path**, если Вы установите **'..'** в свойстве **Path**

, это свойство укажет на родительскую папку.

Исходный код компонента TFindFile

```
unit FindFile; interface uses Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs, FileCtrl; type TFileAttrKind = (ffaReadOnly, ffaHidden, ffaSysFile, ffaVolumeID, ffaDirectory, ffaArchive, ffaAnyFile); TFileAttr = set of TFileAttrKind; TFindFile = class(TComponent) private s : TStringList; fSubFolder : boolean; fAttr: TFileAttr;
```

```
fPath : string;   fFileMask : string;   procedure SetPath(Value: string);   procedure
FileSearch(const inPath : string);   public   constructor Create(AOwner: TComponent);
override;   destructor Destroy;   override;   function SearchForFiles: TStringList;   published
property FileAttr: TFileAttr read fAttr write fAttr;   property InSubFolders : boolean read
fSubFolder write fSubFolder;   property Path : string read fPath write SetPath;   property
FileMask : string read fFileMask write fFileMask ;   end;   procedure Register;   implementation
procedure Register;   begin   RegisterComponents('Samples', [TFindFile]);   end;   constructor
TFindFile.Create(AOwner: TComponent);   begin   inherited Create(AOwner);   Path :=
IncludeTrailingBackslash(GetCurrentDir);   FileMask := '*.*';   FileAttr := [ffaAnyFile];   s :=
TStringList.Create;   end;   destructor TFindFile.Destroy;   begin   s.Free;   inherited Destroy;
end;   procedure TFindFile.SetPath(Value: string);   begin   if fPath Value then   begin   if
Value " then   if DirectoryExists(Value) then   fPath := IncludeTrailingBackslash(Value);
end;   end;   function TFindFile.SearchForFiles: TStringList;   begin   s.Clear;   try
FileSearch(Path);   finally   Result := s;   end;   end;   procedure TFindFile.FileSearch(const
InPath : string);   var   Rec : TSearchRec;   Attr : integer;   begin   Attr := 0;   if ffaReadOnly in
FileAttr then Attr := Attr + faReadOnly;   if ffaHidden in FileAttr then Attr := Attr + faHidden;   if
ffaSysFile in FileAttr then Attr := Attr + faSysFile;   if ffaVolumeID in FileAttr then Attr := Attr +
faVolumeID;   if ffaDirectory in FileAttr then Attr := Attr + faDirectory;   if ffaArchive in FileAttr
then Attr := Attr + faArchive;   if ffaAnyFile in FileAttr then Attr := Attr + faAnyFile;   if
SysUtils.FindFirst(inPath + FileMask, Attr, Rec) = 0 then   try   repeat   s.Add(inPath +
Rec.Name);   until SysUtils.FindNext(Rec) 0;   finally   SysUtils.FindClose(Rec);   end;   If
not InSubFolders then Exit;   if SysUtils.FindFirst(inPath + '*.*', faDirectory, Rec) = 0 then   try
repeat   if ((Rec.Attr and faDirectory) 0) and (Rec.Name'.') and (Rec.Name'..') then   begin
FileSearch(IncludeTrailingBackslash(inPath + Rec.Name));   end;   until
SysUtils.FindNext(Rec) 0;   finally   SysUtils.FindClose(Rec);   end;   end;   end.   { Zarko
Gajic About.com Guide to Delphi Programming http://delphi.about.com free newsletter:
http://delphi.about.com/library/blnewsletter.htm forum: http://forums.about.com/ab-delphi/start/
}   Исходный код
```

[Скачать](#)

*.*em (свойство)