О применении библиотеки FastScript в своих проектах. Часть-4 «Парсинг раздела «Uses» FS-скрипта»

Оглавление

1 Введение

Следует отметить, что:

1. Соглашения, сокращения, термины (и их определения) приведены в документе [2];
2. Автор излагает свой собственный подход к применению библиотеки FastScript, совершенно не претендуя на «истину в последней инстанции»;
3. К статье прилагается иллюстрирующий пример.

Библиотека FastScript предоставляет отличную возможность − структурировать функционал скриптов, размещая соответствующие программные объекты (библиотечные функции, переменные, константы) в отдельных файлах, а затем − вызывать их из других FS-скриптов (также, как это реализовано в Delphi).

Для этого (по аналогии с Delphi) используется директива «**uses**».

Пример использования приведен в Руководстве [1].

На рисунке ниже – скрин со страницы Руководства.

Использование директивы "uses"

Вы можете разбивать большой скрипт на модули, подобно тому, как это делается в Object Pascal. Для использования модуля служит директива "uses". Вот пример ее применения:

Файл unit1.pas:

```
uses 'unit2.pas';

begin
  Unit2Proc('Hello!');
end.
```

Файл unit2.pas:

```
procedure Unit2Proc(s: String);
begin
  ShowMessage(s);
end;

begin
  ShowMessage('initialization of unit2...');
end.
```

Рисунок 1 – Пример использования директивы «**uses**»

Но проблема в том, что в текущей реализации FastScript в директиве «**uses**» можно использовать только полные имена файлов, что создает ощутимые неудобства при практическом применении библиотеки FastSript.

Причем, эта проблема существенно «мешает» не только при изменении местоположения FS-скриптов (в файловой системе ОС),
но и **при выгрузке их из БД** для выполнения.

В разделах ниже рассмотрен вариант «купирования» этой проблемы на конкретных, иллюстрирующих примерах.

2 Практическое применение библиотеки FastScript

Как вариант решения обозначенной в разделе выше проблемы, можно использовать следующий подход.

В рамках обвязки (см. здесь [2]) реализовать (по аналогии с Delphi) механизм «переменных окружения» (далее − **EV**).

Определение «переменной окружения» приведено здесь [2] следующим образом:

Переменная окружения *− внешняя (по отношению к FS-скрипту) текстовая переменная (идентифицируемая определенным образом), значение которой устанавливается обвязкой. Перед компиляцией FS-скрипта, обвязка производит парсинг текста скрипта и заменяет в тексте скрипта идентификатор внешней переменной на ее значение.*

В качестве «контейнера» для EV (если без фанатизма) вполне подходит Delphi-объект **`TStrings`** (используется, как список значений).

Например, так:

```
App_Dir=D:\spFSI\
Scr_Dir=FS\
Lib_Dir=lib\
TMP_Dir=TMP\
```

В рамках FS-скрипта EV может (как вариант) идентифицироваться следующим образом:

$(Идентификатор)

где

символы $( ) - обрамляют идентификатор внешней переменной

ВАЖНО! Следует отметить, что механизм «переменных окружения» (применительно к FastScript) далеко НЕ ограничен парсингом только лишь директивы (раздела) **`uses`** FS-скрипта.

2.1 «Отвязка» от «абсолютного пути» в именах библиотечных файлов

Ниже приведен текст функции обвязки **fsiBase_Main_Script_EnvVars_Prepare()**, которая выполняет парсинг текста скрипта (в контексте рассматриваемой темы).

```
function fsiBase_Main_Script_EnvVars_Prepare(
                                            scrText:TStrings;
                                            List_EnvVars:TStrings
                                           ):boolean;
//Переменные окружения. Парсинг скрипта
//  scrText - текст скрипта (TStrings)
//  List_EnvVars - список значений переменных окружения
begin
  Result:=false;
  if Assigned(scrText) then begin
     if scrText.Count>0 then begin
        Result:=true;
        if Assigned(List_EnvVars) then begin
           if List_EnvVars.Count>0 then begin
              scrText.Text := fsiBase_Main_Script_EnvVars_Prepare(
                                                                  scrText.Text,
                                                                  List_EnvVars
                                                                 );
           end;
        end;
     end;
  end;
end;

function fsiBase_Main_Script_EnvVars_Prepare(
                                            sScript:string;
                                            List_EnvVars:TStrings
                                           ):string;
//Переменные окружения. Парсинг скрипта
//  sScript - текст скрипта
//  List_EnvVars - список значений переменных окружения
Var
  vn,vv:string;
  i:integer;
begin
  Result:='';
  if Assigned(List_EnvVars) then begin
     if List_EnvVars.Count>0 then begin
        Result:=sScript;
        i:=-1;
        while i<(List_EnvVars.Count-1) do
         begin
           i:=i+1;
           vn:=trim(List_EnvVars.Names[i]);
           if length(vn)>0 then begin
              vv:=trim(List_EnvVars.Values[vn]);
              if length(vv)>0 then begin
                 Result:=Replace_In_String(Result, '$('+vn+')', vv, true);
              end;
           end;
        end;
     end;
  end;
end;

function Replace_In_String(const S, Srch, Replace: string; CaseIgnore:boolean=true): string;
//замена подстроки в строке
var
  N:Integer;
  Source:string;
begin
  Source:= S;
  Result:= '';
```

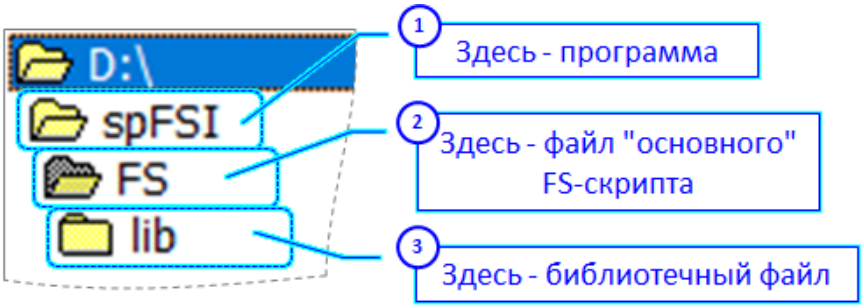
```
  repeat
    if CaseIgnore then begin
       N:=Pos(AnsiUpperCase(Srch), AnsiUpperCase(Source));
    end
    else begin
       N:=Pos(Srch, Source);
    end;
    if N>0 then begin
       Result:=Result+Copy(Source,1,N-1)+Replace;
       Source:=Copy(Source,N+Length(Srch),MaxInt);
    end
    else begin
      Result:=Result+Source;
    end;
  until N<=0;
end;
```

Пример вызова функции **fsiBase_Main_Script_EnvVars_Prepare()**:

```
fsScript_Script_EnvVars_Prepare(
                                fsScript1.Lines,
                                List_EnvVars
                               );
if fsScript1.Compile then begin
   …
   …
end
else begin
   …
   …
end;
```

2.1.1 Иллюстрирующий пример

На рисунке ниже приведена структура папок (каталогов), где размещены FS-Скрипты:



Полное имя файла библиотечного FS-скрипта:
D:\spFSI\FS\lib\FS_Библиотека_01.pas

Рисунок 2 – Структура папок (каталогов), где размещены FS-Скрипты

На рисунке ниже приведен исходный текст библиотечного FS-скрипта **FS_Библиотека_01.pas.**

```
#language PascalScript

//*****************************
//Библиотечная функция
function Текст_Колво_Слов_Подсчитать(Текст:TStrings):integer;
Var
  S:string;
  i:integer;
begin
  Result:=0;
  if Текст<>nil then begin
     if Текст.Count>0 then begin
        i:=-1;
        while i<(Текст.Count-1) do
         begin
          i:=i+1;
          S:=trim(Текст[i]);
          Result:=Result+Строка_Слов_Колво(S,#32+'",.:`!@#$%^&*()-+=[]{}\|/?.,<>');
        end;
     end;
  end;
end;
//*****************************

//*****************************
BEGIN
  //В библиотечном файле этот блок есть смысл использовать
  //для инициализации каких-то программных (библиотечных) объектов
END.
//*****************************
```

"Встроенная" в FSI функция

Рисунок 3 – Исходный текст библиотечного FS-скрипта **FS_Библиотека_01.pas**

Полное имя библиотечного файла:

D:\spFSI\FS\lib\FS_Библиотека_01.pas

На рисунке ниже приведен пример «штатного» использования директивы **uses**.

```
#language PascalScript

Uses
 'D:\spFSI\FS\lib\FS_Библиотека_01.pas';

Var
  Текст: TStrings;

BEGIN
  Текст:= TStringList.Create;
  TRY
    Текст.Add('Парсинг раздела "Uses" FS-скрипта.');
    Текст.Add('Цель:');
    Текст.Add('  "Отвязаться" от "абсолютного пути" в именах библиотечных файлов;');

    ShowMessage(IntToStr(Текст_Колво_Слов_Подсчитать(Текст)));

  FINALLY
    FreeAndNil(Текст);
  END;
END.
```

"Штатное" использование директивы **uses**

Рисунок 4 – «Штатное» использование директивы **uses**

На рисунке ниже приведен пример директивы **uses** с применением «переменных окружения».

```
#language PascalScript

Uses
 '$(App_Dir)$(Scr_Dir)$(Lib_Dir)FS_Библиотека_01.pas';

Var
  Текст: TStrings;

BEGIN
  Текст:= TStringList.Create;
  TRY
    Текст.Add('Парсинг раздела "Uses" FS-скрипта.');
    Текст.Add('Цель:');
    Текст.Add('  "Отвязаться" от "абсолютного пути" в именах библиотечных файлов;');

    ShowMessage(IntToStr(Текст_Колво_Слов_Подсчитать(Текст)));

  FINALLY
    FreeAndNil(Текст);
  END;
END.
```

Директива **uses** с применением «переменных окружения»

Рисунок 5 – Директива **uses** с применением «переменных окружения»

Значения переменных окружения заданы следующим образом:

```
App_Dir=D:\spFSI\
Scr_Dir=FS\
Lib_Dir=lib\
```

При применении функции **fsiBase_Main_Script_EnvVars_Prepare()** для парсинга текста скрипта, оба (указанные на рисунках выше) варианта будут «работать» корректно.

Более предметно «все это» можно посмотреть, скачав исходные тексты иллюстрирующего примера, прилагаемого к этой статье (на roamer55.ru).

Скрин главной формы Приложения (иллюстрирующий пример) – см. на рисунке ниже.

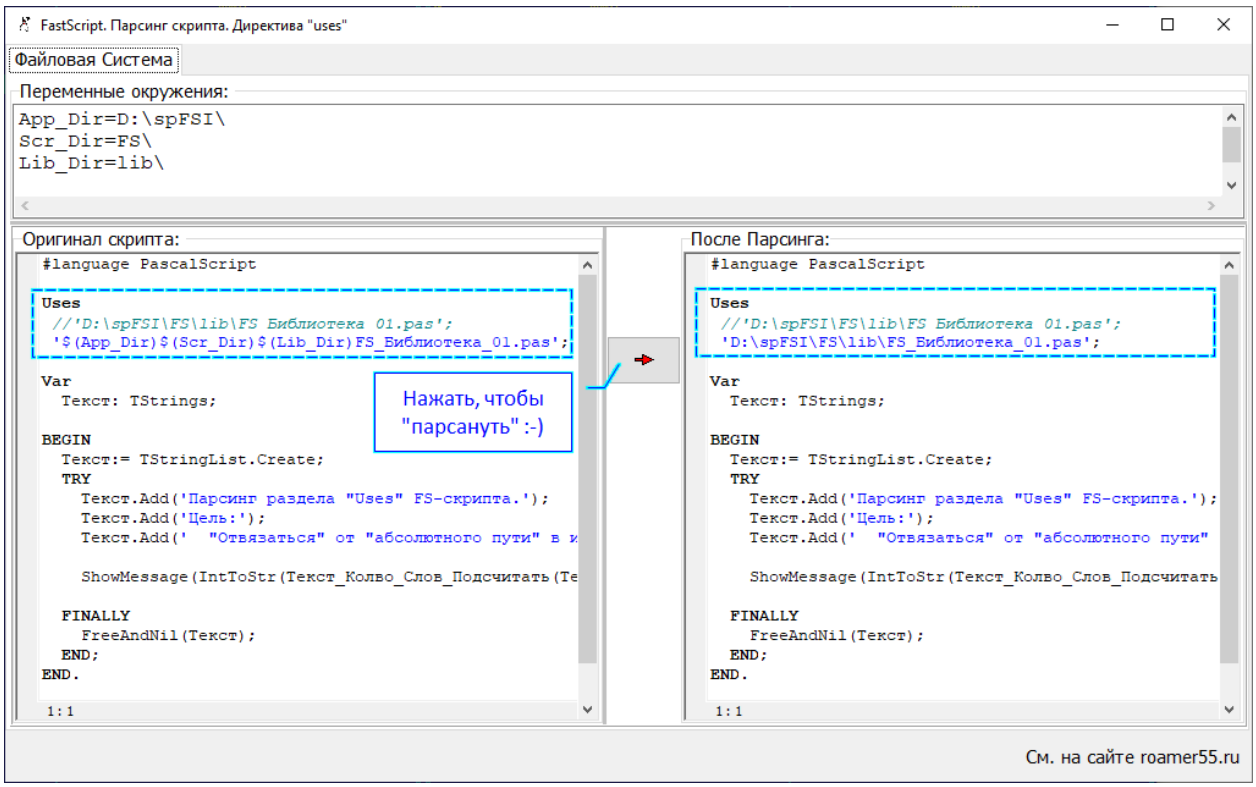


Рисунок 6 – Иллюстрирующий пример.
Результат применения функции **`fsiBase_Main_Script_EnvVars_Prepare()`**

2.2 Корректная выгрузка библиотечных FS-скриптов, хранящихся в БД, и «подключение» их к «вызывающему» FS-скрипту

В тех случаях, когда библиотечные FS-скрипты (подключаемые к другим FS-скриптам с использованием директивы «uses») хранятся в таблицах БД, то дополнительно (к «отвязке» от «абсолютного пути», см. раздел выше) необходимо предусматривать какой-то механизм, который бы позволял в рамках директивы «uses» корректно и однозначно обозначать (идентифицировать) библиотечный (подключаемый) FS-скрипт, хранящийся в БД.

Обвязка должна «уметь» распознавать это «обозначение», чтобы:

1. Найти соответствующий FS-Скрипт в БД;
2. Выгрузить его в соответствующую папку (файловой системы);
3. Изменить соответствующим образом строку в «uses».

Автор, для этих целей, использует такой же подход, как и указанный в разделе на странице 3 («переменные окружения»).

${Идентификатор}

где

символы **${ }** - обрамляют идентификатор FS-скрипта, хранящегося в БД

В качестве идентификатора FS-скрипта, хранящегося в БД, может быть использовано значение ключевого поля (primary key) в соответствующей таблице БД.

Например, так:

```
Uses
 '$(App_Dir)$(Scr_Dir)$(Lib_Dir)${128}',
 '$(App_Dir)$(Scr_Dir)$(Lib_Dir)${5AF2805D586D4CDCA137751382E45236}';
```

Автор предпочитает для этих целей использовать МнемоКоды:

```
Uses
 '$(App_Dir)$(Scr_Dir)$(Lib_Dir)${Скрипт_Библиотечный-1}';
```

Мнемокод FS-скрипта

Это (по личному мнению Автора) – «читабельнее» для Пользователя.

На рисунке ниже представлено содержание таблицы **fs_scripts** БД (см., также, раздел 4.1).

Рисунок 7 – Содержание таблицы **fs_scripts** БД

Автор использует следующий алгоритм при парсинге раздела «**uses**» FS-скрипта (см., также, рисунок 9).

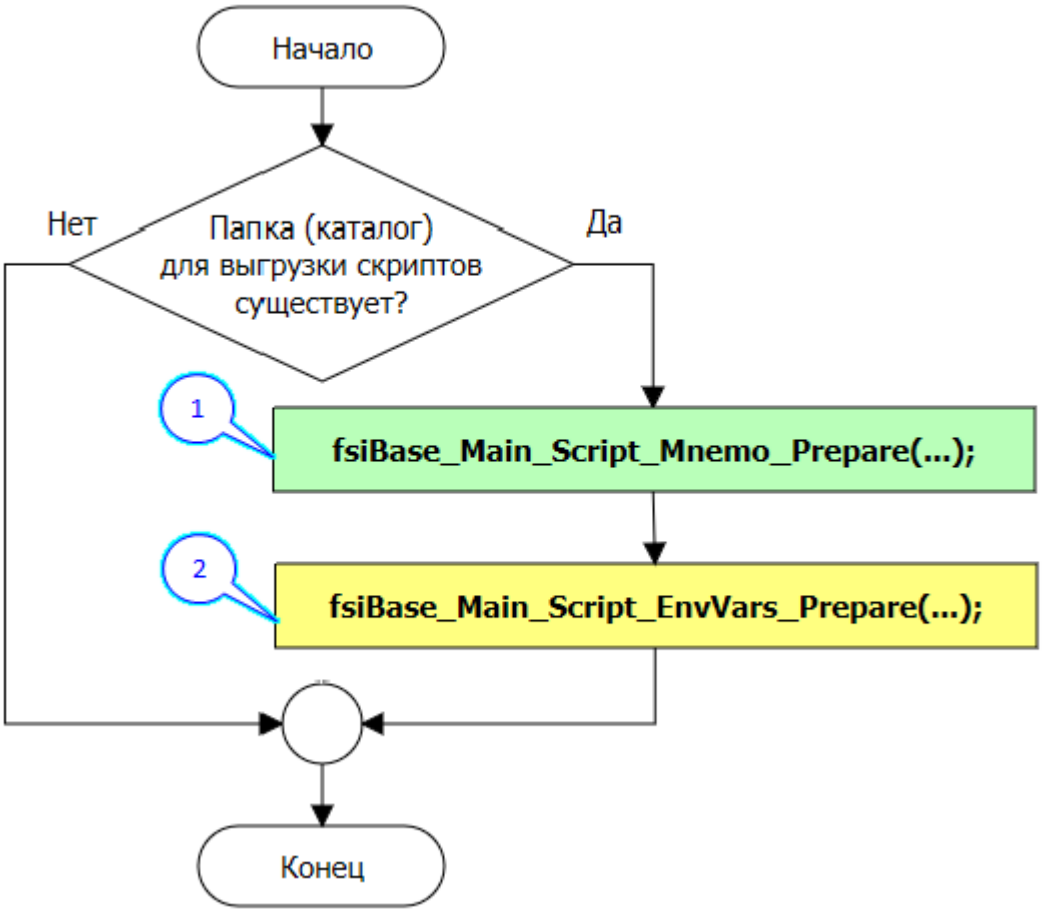


Рисунок 8 – Обобщенный алгоритм парсинга раздела «**uses**» FS-скрипта

Выноски на рисунке выше:

выноска-1 – функция обвязки **fsiBase_Main_Script_Mnemo_Prepare()**, см. листинг ниже (в этом разделе);

выноска-2 – функция обвязки **fsiBase_Main_Script_EnvVars_Prepare()**, см. листинг выше (раздел «2.1 «Отвязка» от «абсолютного пути» в именах библиотечных файлов»).

```
function fsiBase_Main_Script_Mnemo_Prepare(Q:TFDQuery;
                                           scrText:TStrings;
                                           List_EnvVars:TStrings
                                           ):boolean;
//Выгрузка библиотечных скриптов из БД. Парсинг скрипта
Var
  ListScr:TStrings;
```

```
  sDir, sMnemo:string;
  Sx:string;
  InUses:boolean;
  N,i:integer;
begin
  Result:=false;
  if Assigned(Q) then begin
     if Assigned(List_EnvVars) then begin
        if List_EnvVars.Count>0 then begin
           //Получить имя папки для сохранения библиотечных файлов из БД
           sDir:=DirName_from_List_EnvVars(List_EnvVars);
           if DirectoryExists(sDir) then begin
              //удалить все PAS-файлы из папки (если требуется)
              Files_Delete(sDir+'*.pas');
              if Assigned(scrText) then begin
                 if scrText.Count>0 then begin
                    Result:=true;
                 end;
              end;
           end;
        end;
     end;
  end;
  if Result then begin
     //=========================================
     //Поиск раздела USES и его парсинг
     ListScr:=TStringList.Create;
     TRY
       InUses:=false;
       i:=-1;
       while i<(scrText.Count-1) do
        begin
         i:=i+1;
         Sx:=trim(scrText[i]);
         sMnemo:='';
         //---------------------------------------------------
         if InUses then begin
            if Sx<>'' then begin
               //.............................
               //Выделить мнемокод из строки FS-скрипта
               //если он там присутствует
               sMnemo:=Mnemo_Extract(Sx);
               //.............................
               if sMnemo<>'' then begin
                  //.............................
                  //Загрузить скрипт из БД
                  ListScr.Text:=FS_LoadFromDB(Q,
                                              sMnemo,
                                              true);
                  //.............................
                  if ListScr.Count>0 then begin
                     //Сохранить срипт в папку
                     ListScr.SaveToFile(sDir+sMnemo+'.pas');
                  end;
                  ListScr.Text:='';
                  //.............................
                  //Заменить строку в скрипте
                  Sx:=Replace_In_String(scrText[i], '${'+sMnemo+'}', sMnemo+'.pas', true);
                  scrText[i]:=Sx;
                  //.............................
               end;
            end;
         end;
         //---------------------------------------------------
         //.............................
         //Определение начала раздела Uses FS-Скрипта
         if AnsiUpperCase(Sx) = 'USES' then InUses:=true;
         //.............................
         if InUses then begin
            //.............................
            //Оценка: это конец раздела USES?
```

```
            if Sx<>'' then begin
               if Sx[length(Sx)]=';' then begin
                  InUses:=false;
                  i:=(scrText.Count+1); //Выход из парсинга
               end;
            end;
            //.............................
         end;
       end;
     FINALLY
       FreeAndNil(ListScr);
     END;
     //=======================================
  end;
end;
```

Исходные тексты дополнительных функций, используемых в **fsiBase_Main_Script_Mnemo_Prepare()**, см. в разделе 4.2.

Пример вызова функций обвязки **fsiBase_Main_Script_Mnemo_Prepare()** и **fsiBase_Main_Script_EnvVars_Prepare()** приведен на рисунке ниже.

```
procedure TfMain.sbFS_Parsing_from_DBClick(Sender: TObject);
begin
  Application.ProcessMessages;
  fsSyntaxMemo4.Lines.Clear;
  if DirectoryExists(DirName_from_List_EnvVars(Memo_EV.Lines)) then begin
     Files_Delete(DirName_from_List_EnvVars(Memo_EV.Lines)+'*.pas');
     if qScrid.AsInteger>0 then begin
        fsSyntaxMemo4.Lines.Text:=fsSyntaxMemo3.Lines.Text;
        //.........................................
        //Парсинг раздела Uses скрипта и выгрузка
        //соответствующих библиотечных скриптов из БД
        //в файлы
        fsiBase_Main_Script_Mnemo_Prepare(qTMP,
        ------------------------------ fsSyntaxMemo4.Lines,
                                       Memo_EV.Lines
                                      );
        //.........................................
        //.........................................
        //Парсинг скрипта. Переменные окружения
        fsiBase_Main_Script_EnvVars_Prepare(
        ------------------------------  fsSyntaxMemo4.Lines,
                                        Memo_EV.Lines
                                       );
        //.........................................
     end;
  end
  else begin
     ShowMessage('Ошибка! Не найдена папка для выгрузки библиотечных скриптов из БД');
  end;
end;
```

Рисунок 9 – Пример вызова функций обвязки **fsiBase_Main_Script_Mnemo_Prepare()** и **fsiBase_Main_Script_EnvVars_Prepare()**

2.2.1 Иллюстрирующий пример

На рисунке 10 представлен результат выгрузки заданного в разделе «`uses`» FS-скрипта.

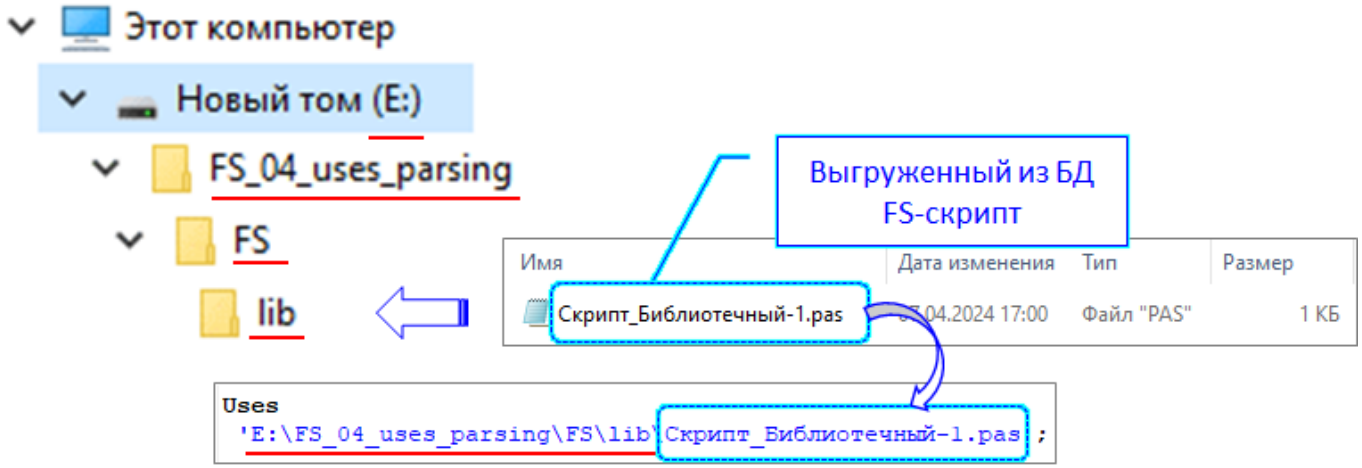


Рисунок 10 – результат выгрузки заданного в разделе «`uses`» FS-скрипта.

На рисунке 11 представлен скрин главной формы приложения **`FS_04_uses_parsing.exe`** (иллюстрирующий пример), открытой на вкладке «База данных (выгрузка)».

Более детально – см. исходники иллюстрирующего примера.

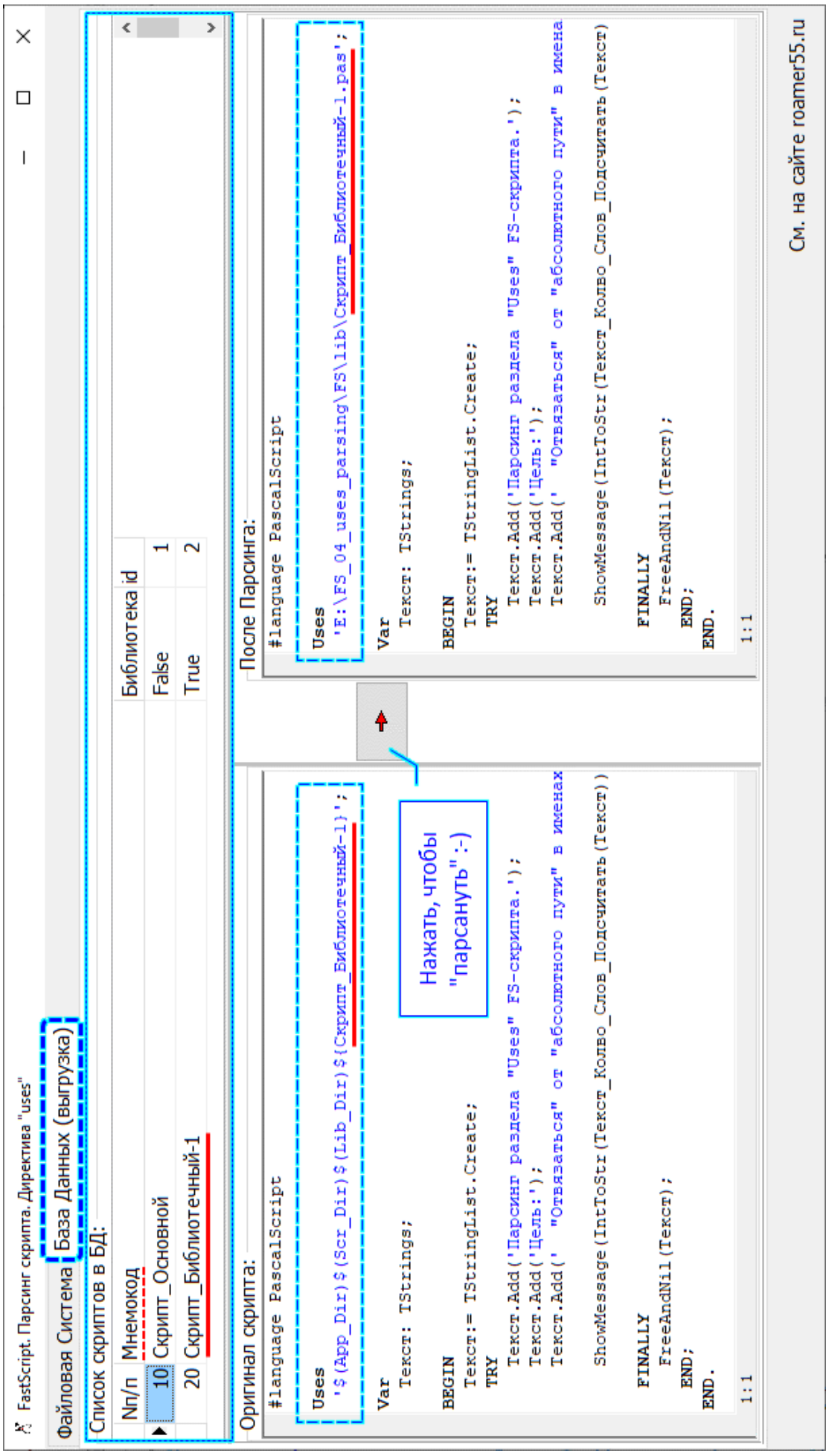


Рисунок 11 – Скрин главной формы приложения **`FS_04_uses_parsing.exe`**, открытой на вкладке «База данных (выгрузка)»

3 Использованные источники

1. «FastScript. Библиотека скриптов. Руководство разработчика» (опубликовано на сайте Разработчика)
2. О применении библиотеки FastScript в своих проектах. Часть-1 «Расширение функционала» (опубликована на сайте roamer55.ru, имя файла: `01_FS_exp_func.pdf`)

4 Приложения

4.1 Приложение-1. Структура таблицы БД для размещения FS-скриптов

Тексты FS-скриптов хранятся в таблице БД **SQLite** следующей структуры (см. рисунок ниже):

```
CREATE TABLE fs_scripts -- список FS-скриптов
(
id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, -- уникальный ID
npp INTEGER,                                   -- порядковый номер в списке
mnemocode VARCHAR(100),                        -- мнемокод скрипта
its_lib BOOLEAN,                               -- флаг: если =TRUE,
                                               --       то библиотечный скрипт
mnemocode_lc VARCHAR(100),                     -- мнемокод скрипта
                                               --      (LoCase по значению mnemocode)
fs_script TEXT                                 -- текст FS-скрипта
);

CREATE INDEX npp_fsscripts ON fs_scripts(npp ASC);
CREATE UNIQUE INDEX mnemo_fsscripts ON fs_scripts(mnemocode);
CREATE INDEX mnemolc_fsscripts ON fs_scripts(mnemocode_lc);
```

Рисунок 12 – Таблица **fs_scripts** БД SQLite

Имя БД: **FS_04_uses_parsing.db**.

Файл БД должен находиться в той же папке, что и программа **FS_04_uses_parsing.exe**.

4.2 Приложение-2. Дополнительные функции к разделу «Корректная выгрузка библиотечных FS-скриптов, хранящихся в БД, и «подключение» их к «вызывающему» FS-скрипту»

Исходные тексты дополнительных (существенных) функций, используемых в `fsiBase_Main_Script_Mnemo_Prepare(...)`.

```
function DirName_from_List_EnvVars(List_EnvVars:TStrings):string;
//Получить полное имя папки для библиотечных файлов
begin
  Result:='';
  if Assigned(List_EnvVars) then begin
     if List_EnvVars.Count>0 then begin
        Result:=trim(List_EnvVars.Values['App_Dir']);
        Result:=Result+trim(List_EnvVars.Values['Scr_Dir']);
        Result:=Result+trim(List_EnvVars.Values['Lib_Dir']);
     end;
  end;
end;


function FS_LoadFromDB(Q:TFDQuery;
                       ScrMnemo:string;
                       ItsLibOnly:boolean=true
                      ):string;
//Загрузить FS-скрипт по МнемоКоду
Var
  sFilter:string;
begin
  Result:='';
  ScrMnemo:=trim(ScrMnemo);
  if ScrMnemo<>'' then begin
     sFilter:='(mnemocode_lc='+#39+AnsiLowerCase(ScrMnemo)+#39+')';
     if ItsLibOnly then begin
        sFilter:=sFilter+' and (its_lib=1)';
     end;
     Result:=DB_Table_Field_AsString(Q,
                                     'fs_scripts',
                                     'fs_script',
                                     sFilter
                                     );
  end;
end;


function Delete_start_String(Sx : string; Index:integer) : string;
//Удалить начало строки, начиная с 1-й позиции и до заданной (включая заданную)
begin
  Result:=Sx;
  if length(Sx)>0 then begin
     if Index>0 then begin
        System.Delete(Result, 1, Index);
     end;
  end;
end;


function Delete_end_String(Sx : string; Index:integer) : string;
//Удалить окончание строки, начиная с заданой позиции
begin
  Result:=Sx;
  if length(Sx)>0 then begin
     if Index>0 then begin
        System.Delete(Result, Index, length(Result)+1);
     end;
  end;
end;
```

```
function DB_Table_Field_AsString(Q:TFDQuery;
                                 tndb:string;
                                 fntn:string;
                                 sFilter:string;
                                 sValDef:string='';
                                 sOrderBy:string='';
                                 YesTrim:boolean=true):string;
//Получить значение заданного поля (как строка) заданной таблицы Б.Д. по заданному фильтру
Var
  Yes:boolean;
begin
  Result:=sValDef;
  tndb:=trim(tndb);
  fntn:=trim(fntn);
  if (length(tndb)>0) and (length(fntn)>0) then begin
      if Assigned(Q) then begin
         sFilter:=trim(sFilter);
         sOrderBy:=trim(sOrderBy);
         Yes:=false;
         Q.Close;
         Q.SQL.Clear;
         TRY
           Q.SQL.Add('select');
           Q.SQL.Add(fntn);
           Q.SQL.Add('from');
           Q.SQL.Add(tndb);
           if length(sFilter)>0 then begin
              Q.SQL.Add('where');
              Q.SQL.Add(sFilter);
           end;
           if length(sOrderBy)>0 then begin
              Q.SQL.Add('order by');
              Q.SQL.Add(sOrderBy);
           end;
           //Q.SQL.Add('limit 1');
           Q.Open;
           if Q.RecordCount>0 then begin
              if not Q.Fields[0].IsNull then begin
                 Result:=Q.Fields[0].AsString;
              end;
           end;
           Yes:=true;
         FINALLY
           Q.Close;
           if not Yes then begin
              ShowMessage(Q.SQL.Text);
           end;
           Q.SQL.Clear;
         END;
      end;
  end;
  if YesTrim then Result:=trim(Result);
end;


function Replace_In_String(const S, Srch, Replace: string; CaseIgnore:boolean=true): string;
//замена подстроки в строке
var
  N:Integer;
  Source:string;
begin
  Source:= S;
  Result:= '';
  repeat
    if CaseIgnore then begin
       N:=Pos(AnsiUpperCase(Srch), AnsiUpperCase(Source));
    end
    else begin
       N:=Pos(Srch, Source);
    end;
    if N>0 then begin
```

```
       Result:=Result+Copy(Source,1,N-1)+Replace;
       Source:=Copy(Source,N+Length(Srch),MaxInt);
    end
    else begin
      Result:=Result+Source;
    end;
  until N<=0;
end;


function Files_Delete(NameMask: String): integer;
//Удалить файлы по маске
Var
  DirInfo: TSearchRec;
  Err: integer;
begin
  Result := 0;
  Err := 0;
  Err := FindFirst(NameMask, faArchive, DirInfo);
  while Err = 0 do
   begin
    if DeleteFile(ExtractFilePath(NameMask) + DirInfo.Name) then begin
       Result := Result + 1;
    end;
    Err := FindNext(DirInfo);
  end;
  FindClose(DirInfo);
end;
```