Appendix 1

Library for Work with Object Space

In this appendix the demo example full source for program realization of library for work with object space (OS) when using Bold for Delphi is presented.

Purpose

The purpose of this library is to provide realization of the minimal necessary set of most often used operations with OS objects, i.e. addition, deletion, editing of objects, search of objects, program linkage of objects (for cases of single and multiple associations between classes). In other words, this library allows to carry out in program way all operations, which support autoforms, or any other means of Bold for Delphi graphic interface.

Objectives and Reasons for Library Creation

In practice when developing applications in Bold for Delphi one serious problem often occurs. It consists in the following. Starting database applications creation with use of this new technology, the developer faces with necessity of the information import, already gathered by this moment in some database existing earlier. This "old" database has been created by traditional ways. How to provide such import without using manual data input? In fact as we know Bold for Delphi generates new DB structure. At that use of GUI (autoforms, grids, etc.) is standard method of the information input into object space. From this point of view the program library presented below can be considered as a basis for creation of similar import tool. The second purpose of this library demonstration is to show principles of work with OS objects, examined partially in this book.

Description

Main Procedures and Functions

In Table A3.1 the structure of library basic procedures and functions, their purpose and description of input parameters and output values are presented.

Table A3.1. Library Procedures and Functions Description

#	Type	Name	Purpose	Parameters
1	Function	Attr	Returns object attribute value	L– object; attrname– attribute name Returns option value
2	Function	ObjectLocate	Locates object by attribute value	ClassName– object class name; Attr– attribute name; Searchvalue– attribute value Returns located object or NIL
3	Procedure	AddObject	Adds object into OS	ClassName– object class name; AttrName– attributes names array AttrValue– attributes values array
4	Procedure	AddObjectWithBlobs	The same, but with BLOB-attributes	The same plus BlobAttrName– BLOB-attributes names array Fname– files names array, which data will be placed into BLOB
5	Function	NewObject	Creates new object in OS	ClassName– object class name AttrName– attributes names array AttrValue– attributes values array Returns new object
6	Function	NewObjectWithBlobs	The same, but with BLOB-attributes	The same plus BlobAttrName– BLOB-attributes names

				array Fname– files names array, which data will be placed into BLOB
7	Procedure	EditObject	Changes object attributes	ClassName– object class name AttrName– attributes names array AttrValue– array of attributes new values
8	Procedure	EditObjectWithBlobs	The same, including BLOB-attributes	The same plus BlobAttrName– BLOB-attributes names array Fname– files names array, which data will be placed into BLOB
9	Procedure	SetObjectToLink	Links two objects (single association)	S– object Rolename– name of role on the association opposite end LinkObject– class object on the association opposite end
10	Procedure	AddObjectToMultiLink	Adds object to the link (multiple association)	O– object Rolename– name of role on the association opposite end AdOb– added class object on the association opposite end
11	Procedure	LocateAndSetLinkToObject	Locates object and links the specified object to it (single association)	S– object Rolename– name of role on the association opposite end attrlink– name of required object attribute searchvalue– attribute value, by which the search will be fulfilled
12	Procedure	LocateAndSetMultiLinkToObject	The same, for multiple association	S– object Role– name of role on the association opposite end attr– name of required object attribute LinkObject– name of required object class searchvalue– attribute values array, by which the search will be fulfilled

Auxiliary Procedures and Functions

For convenience of work with national-language names of classes, attributes and roles directly in the program, in structure of the given library TL transliteration function is included. There is no necessity to call this function directly, as it is used in the program code of the described procedures and functions. Besides for use of library it is necessary in section of the main program initialization (or somewhere else, before the first reference to the given library) to call the following procedure

```
RegisterObjectSpace(bsystem:TBoldSystemHandle)
```

Where it is necessary to set OS system handle as bsystem.

Examples of Use

Here some examples of using functions and procedures from the given library will be considered for visual presentation. Fragment of model considered in this book is accepted as UML-model.

OS Objects Addition and Change

Operator for creating (adding) one author:

```
NewObject('Author',['name','name'],['Pushkin','Alexander']);
```

Operator for adding author and his photo

```
NewObjectWithBlobs('Author',['name'],['Pushkin'],['photo'],['C:\Photo\push.jpg']);
```

Operator for changing country name

```
EditObject('Country',['name'],['USA']);
```

Object Search and Link

For book search by attribute it is enough to write

```
var Book:TBoldObject;
    st :string;
begin
  st:='Book Title';
  Book:=ObjectLocate('Book','name',st);
```

or

```
  Book:=ObjectLocate('Book','name', 'Book Title');
```

Book variable will contain the found object (if search is successful) or NIL value.

We can link the certain author with the country of residing in several ways. The first way is when both link objects are located beforehand.

```
var Author,Country:TBoldObject;
begin
  Country:=ObjectLocate('Country','name','USA');
  Author:=ObjectLocate('Author','name','Pushkin');
  SetObjectToLink(author,'country',country);
end;
```

The second way is when we have Author object-variable and it is known that this author lives in the USA, but the object on the opposite end of association (country) does not exist yet.

```
  LocateAndSetLinktoObject(Author,'country','name', 'USA');
```

And, at last, work with multiple associations. By means of one operator we shall "inform" object of country type that many authors should be linked to it.

```
LocateAndSetMultilinkToObject(Country,'authors','Author','name',['Author1','Author2',
'Author3','Author4'])
```

Or we shall link several authors to one book.

```
Book:=ObjectLocate('Book','name', 'Book Title');
LocateAndSetMultilinkToObject(Book,'writtedby','Author','name',['Author1','Author2'])
```

Thus, at use of procedures of the given library it is possible in rather simple way to add new objects into the OS, to edit them if necessary, and also to connect with each other by means of associations.

Warnings

It is necessary to consider the library program text (see Listing A3.1) as demo example, which is not intended for commercial implementation. The author intentionally did not optimize a code for the sake of simplicity and visual information presentation, besides in some cases it is necessary before using the presented code to provide protection of blocks type – «try.. finally», at work with BLOB-streams and files. Also for use of library without code changes, it is necessary to take into account that graphic files formats are "JPG" by default. The attentive developer can independently “polish” the presented example of library realization up to the necessary level of reliability and processing speed and universality.

ATTENTION

Author is not responsible for possible violations in operability of program or hardware environment, which is directly or indirectly connected with use of the program code presented in Listing A3.1.

Listing A3.1. A demonstration example of program realization of library for work with Bold for Delphi object space

```
unit ObjectSpace;

                                INTERFACE
uses
  SysUtils,
  Variants,
  Boldattributes,
  BoldSystem,
  BoldSystemHandle,
  BoldElements,
  BoldMeta;
var
    BSys : TBoldSystem;
    CurObj,ObjSave,ObjLocate,NewObj,LocObj,AddObj : TBoldObject;
    ObjList : TBoldObjectList;

//---------------------------------------------------------------------------
Function TL(mylangstring:string) : String;
//---------------------------------------------------------------------------
Procedure RegisterObjectSpace(bsystem:TBoldSystemHandle);
//---------------------------------------------------------------------------
Function Attr(
         L:TBoldObject;
         attrname:string)  :variant;
//---------------------------------------------------------------------------
Function ObjectLocate(
                      ClassName:string;
                      attr:string;
                      searchvalue:variant)  :TBoldObject;
//---------------------------------------------------------------------------
Procedure AddObject(
                    ClassName:string;
                    AttrName: array of string;
                    AttrValue:array of variant);
//---------------------------------------------------------------------------
Procedure AddObjectWithBlobs(
                             ClassName:string;
                             AttrName: array of string;
                             AttrValue:array of variant;
                             BlobAttrName:array of string;
                             FName:array of string);
//---------------------------------------------------------------------------
Function NewObject(
                   ClassName:string;
                   AttrName: array of string;
                   AttrValue:array of variant):    TBoldObject;
//---------------------------------------------------------------------------
Function NewObjectWithBlobs(
                            ClassName:string;
                            AttrName: array of string;
                            AttrValue:array of variant;
                            BlobAttrName:array of string;
                            FName:array of string):      TBoldObject;
//---------------------------------------------------------------------------
```

```
Procedure EditObject(
                    L:TBoldObject;
                    AttrName: array of string;
                    AttrValue:array of variant);
//-----------------------------------------------------------------------------
Procedure EditObjectWithBlobs(
                            L:TBoldObject;
                            AttrName: array of string;
                            AttrValue:array of variant;
                            BlobAttrName:array of string;
                            FName:array of string);
//-----------------------------------------------------------------------------
Procedure SetObjectToLink(
                         S:TBoldObject;
                         rolename:string;
                         LinkObject:TBoldObject);
//-----------------------------------------------------------------------------
Procedure AddObjectToMultiLink(
                             O:TBoldObject;
                             rolename:string;
                             AdOb:TBoldObject);
//-----------------------------------------------------------------------------
Procedure LocateAndSetLinkToObject(
                                 S:TBoldObject;
                                 rolename:string;
                                 attrlink:string;
                                 searchvalue:variant);
//-----------------------------------------------------------------------------
Procedure LocateAndSetMultiLinkToObject(
                                      S:TBoldObject;
                                      role:string;
                                      LinkObject:string;
                                      attr:string;
                                      searchvalue:Array of string);
//-----------------------------------------------------------------------------
//*****************************************************************************
                            IMPLEMENTATION

var InnerObj : TBoldObject;

//=============================================================================

Procedure RegisterObjectSpace(bsystem:TBoldSystemHandle);
begin
  BSys:=bsystem.System;
end;
//=============================================================================

Function Attr(
         L:TBoldObject;
         attrname:string)  :variant;
begin
  result:=L.BoldMemberByExpressionName[TL(attrname)].AsString;
end;
//=============================================================================

Function ObjectLocate(
                     ClassName:string;
                     attr:string;
                     searchvalue:variant)  :TBoldObject;
var expr:string;
```

```
    SearchObj:TBoldObject;
    L:TBoldObjectList;
begin
     if searchvalue=null then begin Result:=nil; exit;end;
     L:=BSYS.ClassByExpressionName[TL(ClassName)];
     expr:='self->select('+TL(attr)+'='+QuotedStr(searchvalue)+')->first';
     SearchObj:=L.EvaluateExpressionAsDirectElement(expr) as TBoldObject;
     if Assigned(SearchObj) then Result:=SearchObj else Result:=nil;
end;
//=============================================================================

Procedure AddObject(
                   ClassName:string;
                   AttrName: array of string;
                   AttrValue:array of variant);
var
    L  : TBoldObjectList;
    i  : word;
    nattr  :integer;
begin
   L:=BSys.ClassByExpressionName[TL(ClassName)];
   InnerObj:=(L.AddNew as TBoldObject);
   nattr:=High(AttrName);
   if nattr<0 then exit;
   for i:=0 to High(Attrname)  do
   if string(AttrValue[i])<>'' then
   InnerObj.BoldMemberByExpressionName[TL(AttrName[i])]
   .SetAsVariant(AttrValue[i]);
   AddObj:=InnerObj;
end;
//=============================================================================

Procedure AddObjectWithBlobs(
                            ClassName:string;
                            AttrName: array of string;
                            AttrValue:array of variant;
                            BlobAttrName:array of string;
                            FName:array of string);
var
    L  : TBoldObjectList;
    bs : TBoldBlobStream;
    sm : TBoldBlobStreamMode;
    i : word;
    blobnum:integer;
    filext : string;
begin
   L:=BSys.ClassByExpressionName[TL(ClassName)];
   InnerObj:=(L.AddNew as TBoldObject);
   for i:=0 to High(Attrname)  do
   if string(AttrValue[i])<>'' then
      InnerObj.BoldMemberByExpressionName[TL(AttrName[i])]
      .SetAsVariant(AttrValue[i]);
   blobnum:=Length(BlobAttrName);
   sm:=bmReadWrite;
   if (blobnum<>0) then  for i:=0 to blobnum-1 do
   if FileExists(FName[i]) then
   begin          // here it is necessary to use the block try..finally !
     filext:=ExtractFileExt(FName[i]);
     bs:=(InnerObj.BoldMemberByExpressionName[TL(BlobAttrName[i])] as TBABlob)
     .CreateBlobStream(sm);
     bs.LoadFromFile(FName[i]);
```

```
     if (filext='.jpg') then
     begin
       (innerobj.BoldMemberByExpressionName[TL(BlobAttrName[i])] as TBATypedBlob)
       .ContentType:='image/jpeg';
       // here we can add contexts types for other graphic formats (BMP etc)
       //
     end;
     bs.Free;
   end;
   AddObj:=InnerObj;
end;
//=============================================================================

Function NewObject(
                  ClassName:string;
                  AttrName: array of string;
                  AttrValue:array of variant):    TBoldObject;
begin
  AddObject(ClassName,AttrName,AttrValue);
  Result:=InnerObj;
  CurObj:=InnerObj;
  NewObj:=InnerObj;
end;
//=============================================================================

Function NewObjectWithBlobs(
                           ClassName:string;
                           AttrName: array of string;
                           AttrValue:array of variant;
                           BlobAttrName:array of string;
                           FName:array of string):      TBoldObject;
begin
  AddObjectWithBlobs(ClassName,AttrName,AttrValue,BlobAttrName,FName);
  Result:=InnerObj;
  CurObj:=InnerObj;
  NewObj:=InnerObj;
end;
//=============================================================================

Procedure EditObject(
                     L:TBoldObject;
                     AttrName: array of string;
                     AttrValue:array of variant);
var
 i: word;
begin
   if L=nil then Exit;
   if Length(AttrName)<>0 then
   for i:=0 to High(Attrname) do
   L.BoldMemberByExpressionName[TL(AttrName[i])].SetAsVariant(AttrValue[i]);
end;
//=============================================================================

Procedure EditObjectWithBlobs(
                             L:TBoldObject;
                             AttrName: array of string;
                             AttrValue:array of variant;
                             BlobAttrName:array of string;
                             FName:array of string);

var bs : TBoldBlobStream;
```

```
    sm : TBoldBlobStreamMode;
    i  : word;
    blobnum:integer;
    filext:string;
begin
   if L=nil then exit;
   if Length(AttrName)<>0 then
   for i:=0 to High(Attrname)  do
   L.BoldMemberByExpressionName[TL(AttrName[i])].SetAsVariant(AttrValue[i]);
   sm:=bmReadWrite;
   blobnum:=Length(BlobAttrName);
   if (blobnum<>0) then  for i:=0 to blobnum-1 do  if FileExists(FName[i]) then
   begin  // here it is necessary to use the block try..finally !
     filext:=ExtractFileExt(FName[i]);
     bs:=(L.BoldMemberByExpressionName[TL(BlobAttrName[i])] as TBABlob)
          .CreateBlobStream(sm);
     bs.LoadFromFile(FName[i]);
     if (filext='.jpg') then
     begin
       (L.BoldMemberByExpressionName[TL(BlobAttrName[i])] as TBATypedBlob)
       .ContentType:='image/jpeg';
     end;
     bs.Free;
   end;
end;
//=============================================================================

Procedure SetObjectToLink(
                          S:TBoldObject;
                          rolename:string;
                          LinkObject:TBoldObject);
begin
   S.BoldMemberByExpressionName[TL(rolename)].assign(LinkObject);
end;
//=============================================================================

Procedure LocateAndSetMultiLinkToObject(
                                       S:TBoldObject;
                                       role:string;
                                       LinkObject:string;
                                       attr:string;
                                       searchvalue:Array of string);

var expr:string;
    LinkObj:TBoldObject;
    LinkList:TBoldObjectList;
    i,multi:word;
begin
   LinkList:=bsys.ClassByExpressionName[TL(LinkObject)];
   multi:=Length(searchvalue);
   for i:=0 to multi-1  do
   begin
     expr:='self->select('+TL(attr)+'='+QuotedStr(searchvalue[i])+')->first';
     LinkObj:=LinkList.EvaluateExpressionAsDirectElement(expr) as TBoldObject;
     if Assigned(LinkObj) then
     (S.BoldMemberByExpressionName[TL(role)] as TBoldObjectList).Add(LinkObj);
   end;
   LocObj:=LinkObj;
end;
//=============================================================================
```

```
Procedure AddObjectToMultiLink(
                               O:TBoldObject;
                               rolename:string;
                               AdOb:TBoldObject);
begin
 (O.BoldMemberByExpressionName[TL(rolename)] as TBoldObjectList).Add(AdOb);
end;
//=============================================================================

Procedure LocateAndSetLinkToObject(
                                   S:TBoldObject;
                                   rolename:string;
                                   attrlink:string;
                                   searchvalue:variant);
var expr:string;
    LinkObj:TBoldObject;
    L:TBoldObjectList;
    LinkClass :string;
begin
   if searchvalue=NULL then exit;
   LinkClass:=S.BoldMemberByExpressionName[TL(rolename)].BoldType.ModelName;
   L:=BSys.ClassByExpressionName[TL(LinkClass)];
   expr:='self->select('+TL(attrlink)+'.asstring='
          +QuotedStr(searchvalue)+')->first'; // to find object by attribute
   LinkObj:=L.EvaluateExpressionAsDirectElement(expr) as TBoldObject;
   if Assigned(LinkObj) then
   S.BoldMemberByExpressionName[TL(rolename)].assign(LinkObj);
   LocObj:=LinkObj;
end;
//*****************************************************************************

end.
```