Home x Database Functions x Definitions x Error handling x Events handling x Methods x
Find x GetChunk x Objects x Module x Properties x On (control event) x
On (database event) x On (dialog event) x On (form event) x ProcBodyLine x ProcCountLines x
ProcOfLine x ProcStartLine x Shortcuts x Snippets of Code x Module example x
On control event example x On dialog event example x On form event example x Special controls x
Tutorial x DLookupSamples x Object Model x Standalone Forms x User's Guide x

User's Guide

tags: Menu

The reader is assumed to have already a reasonable knowledge of *LibO/AOO Basic* and of the *LibO/AOO Basic* IDE. The knowledge of the *LibreOffice/OpenOffice UNO API* is not required. A basic knowledge of the *MSAccess* object model is an advantage.

DEFINITIONS

What are objects ?

AOO/LibO Basic is a rudimentary object-oriented language.
To stay close to the syntax of the *MSAccess* object model the implementation of Access2Base has nevertheless been based on BASIC object classes. Their main characteristics are :

- Use of classes of types **Form**, **Subform** or **Control**, etc.
- In BASIC code using these classes, variables should be declared as being of type `Object`.
- **Property** and **method** names are identical in *MSAccess* and in *Access2Base*. Note however that *Access2Base* implements only a limited subset of the object model of *MSAccess*. Note also that their semantics might differ from the original *MSAccess* one. Read the current documentation carefully.
- Support of indirection and introspection for properties.
- 2 ROOT objects: **Application** and **DoCmd**.
- The **CurrentDb** method returns a **Database** object. It might be (and most often is ...) the database related to the only opened Base document (".odb") or any of the databases related to one of the **standalone forms** stored in a non-Base (Writer, Calc, ...) document.

What are collections ?

- Their **Count** and **Item** properties
- Their **Add**, **Delete**, **Remove** and **RemoveAll** methods

> **Here is the complete Object Model**

FORMS, DIALOGS and CONTROLS

Introduction

- The **AllForms** collection - all forms
- The **Forms** collection - all active forms
- The **AllDialogs** collection - all available dialogs
- The **Form** object - one single active form
- The **Dialog** object - one single active dialog
- The **Controls** collection - all controls of an active form or dialog
- The **Control** object - one single control on an active form or dialog

Properties

- Form properties
  - **Name**
  - **AllowAdditions**, **AllowDeletions**, **AllowEdits** - is a form updatable ?
  - **RecordSource**, **Filter**, **FilterOn**, **OrderBy**, **OrderByOn** - which data are queried and in which sequence ?
  - **Recordset** - to query the same data programmatically
  - **IsLoaded** - is the form active ?
  - **Caption**, **Height**, **Width**, **Visible** - how is the form formatted ?
  - **Bookmark**, **CurrentRecord** - how to identify the current record and go back to it afterwards, how to identify the current record number ?

    - **OpenArgs**
    - **On ... form events** - which routine is triggered when an event occurs ?
- Dialog properties
    - **Name**
    - **IsLoaded** - is the dialog active ?
    - **Caption**, **Height**, **Width**, **Visible** - how is the dialog formatted ?
- Control properties
    - **Name**
    - **ControlType**, **SubType** - control typology ?
    - **BackColor**, **BorderColor**, **BorderStyle**, **ControlTipText**
      FontBold, **FontItalic**, **FontName**, **FontSize**, **FontUnderline**, **FontWeight**, **ForeColor**
      Format, **TextAlign**, **Visible**, **TripleState**
      Cancel, **Caption**, **Default**
      - how is the control's look & feel ?
    - **Page** - define the set of controls of a dialog displayed page by page
    - **Enabled**, **Locked**, **Required** - is the control read-only ... ?
    - **SelStart**, **SelLength** and **SelText** - what are the characters selected by the user, where is the insertion point in a textbox control ?
    - **DefaultValue**, **Tag**, **Text**, **Value**, **Picture** - what is the content of the control ?
    - **TabIndex**, **TabStop** - what is the tab sequence of the control ?
    - **ControlSource** - which data is linked to the control ?
    - **Parent** - which object contains the control ?
    - **On ... control events** - which routine is triggered when an event occurs ?

Methods

- **Move** - move and resize a form or dialog
- **Refresh**, **Requery** - requery the underlying data of a form or a control
- **SetFocus** - set the cursor somewhere
- Manage dialogs with the **Start**, **Execute**, **EndExecute** and **Terminate** methods

Special controls

Subforms

- What is a **subform** ?
- The **form** property of a control
- Subform properties
    - **Name**
    - **AllowAdditions**, **AllowDeletions**, **AllowEdits** - is a subform updatable ?
    - **RecordSource**, **Filter**, **FilterOn**, **OrderBy**, **OrderByOn** - which data are queried and in which sequence ?
    - **Recordset** - to query the same data programmatically
    - **LinkChildFields**, **LinkMasterFields** - how is the subform related to its parent form ?
    - **Parent** - which (other sub)form contains the subform ?

Gridcontrols

- Tabular display of data via a **gridcontrol**
- Use of the **Controls** collection to find the columns of a gridcontrol

List- and Comboboxes

- What is a **ListBox** ? What is a **ComboBox** ?
- List- and combobox properties
    - **ItemData**, **RowSource**, **RowSourceType** - which data in the box and where does it come from ?
    - **ListCount**, **ListIndex** - how long is the list and which item is currently selected ?
    - **MultiSelect**, **Selected** - how to manage multi-select listboxes ?
- Listbox methods
    - **AddItem**, **RemoveItem** - manage its content

OptionGroup and RadioButton controls

- How are **OptionGroups** and **RadioButtons** related ?
- How many radio buttons in an options group (**Count**)?
- The **OptionGroup** method of a (sub)form
- The **OptionValue** property of a RadioButton

DATABASE ACCESS

Info about the database

- The **Name** provides the file name of the Base document ("`xxx.odb`").
- The **Connect** property returns the connection string between the Base document and the effective database.
- The **Version** property describes the database system supporting the actual database and its version.

Database tables

- Use of the **TableDefs** collection to access individual tables
- Each **TableDef** object represents a table
- When applied to the *TableDefs* collection the **CreateTableDef** and **Add** methods create a new table in the database
- The **Fields** collection applied to a TableDef object gives access to each individual field of the concerned table
- The **OpenRecordset** method gives access to the individual records of the table

Database queries

- Use of the **QueryDefs** collection to access individual queries
- Each **QueryDef** object represents a query
- The **Fields** collection applied to the QueryDef object gives access to each individual field of the concerned query
- The **OpenRecordset** method gives access to the individual records of the query
- The **SQL** property of the querydef allows getting and setting the SQL statement related to the query

Access to the records in a table, a query or an arbitrary SQL statement

- Such a set of records is represented by a **Recordset** object
- A recordset object is created via the **OpenRecordset** method. A recordset can be explored forward or backward starting from its *current position*
- The individual fields of a recordset are accessed from its **Fields** collection
- Properties of a recordset
    - **RecordCount** - know the total number of records in the recordset
    - **BOF and EOF** - identify if the current position is *before the first* or *after the last* record
    - **AbsolutePosition** - know or set the position of the current record
    - **Bookmarkable** and **Bookmark** to remember the current record to prepare a later return to it
    - **Filter** helps preparing to build a new recordset which will be a subset of the current one
- Methods of a recordset
    - **Move**, **MoveFirst**, **MoveLast**, **MoveNext**, **MovePrevious** - navigate thru the recordset and set its new current record
    - **OpenRecordset** - open a new recordset based on a preset **Filter**
    - **AddNew**, **Edit**, **Update** and **CancelUpdate** - add new records or update the current record. With the **EditMode** property, know the current editing status
    - **GetRows** - Read a set of records at once into an array.
    - **Close** - closing a recordset after use is **mandatory** !

Access to the individual fields of a table, a query or any recordset

- Such a set of fields is represented by a **Fields** collection of indivudual **Field** objects
- New fields can be added to a table thru the **CreateField** method.
- Properties of a field
    - **Name** - the name of the field as given by the source SQL statement
    - **Description** - to know the help text associated with a field when the table is listed in design view
    - **DataType**, **DbType**, **TypeName** and **Size** - know the type of the field in the database and its size
    - **SourceTable** and **SourceField** - identify the names of the underlying table and field
    - **DataUpdatable** - know if that specific field may be updated
    - **DefaultValue** - get or set the value that will be preset in new records
    - **Value** - get the current value of the field or modify it

- Methods of a field (only for text and binary fields)
    - **WriteAllText** and **WriteAllBytes** - export the content of a (text or binary) database field into a file identified by its name
    - **ReadAllText** and **ReadAllBytes** - import the content of a (text or binary) file identified by its name directly into a database field
    - **GetChunk** and **AppendChunk** - move the content of a binary field by chunks and store it by chunk into another binary field

TEMPORARY VARIABLES

Temporary variables are variables that can be created or removed at any time by any macro. They help passing values through macros or LibO/AOO applications sharing the same Access2Base API.
They are gathered in the **TempVars** collection.
Their value can be get or set with the **Value** property.

INTROSPECTION

All objects have an **ObjectType** property
Specific methods are available to manage property *indirection* and property *introspection*.

- **hasProperty** determines if an object has a given property
- **getProperty** and **setProperty** help managing the values of properties

See also the **Property** object and the **Properties** collection.

SHORTCUTS

A **shortcut** is a character string designating unambiguously forms and controls. Next functions help managing them:

- **getObject** returns the corresponding object
- **getValue** and **setValue** get and set their properties

ACTIONS

- **OpenForm**, **OpenTable**, **OpenQuery**, **OpenReport**, **Close** - open or close *OpenOffice/LibreOffice Base* objects
- **CopyObject** - copy tables and/or queries
- **RunSQL** - run action SQL statements
- **OpenSQL** - display the data related to a SELECT SQL statement
- **FindRecord**, **FindNext** - search strings or values in gridcontrols
- **GoToRecord** - move back-and forward in form records
- **ApplyFilter**, **SetOrderBy**, **ShowAllRecords** - set or remove filters and sorting sequences
- **GoToControl** - move the focus to a control simply by its name
- **SelectObject**, **SetHiddenAttribute**, **GetHiddenAttribute** - browse thru open windows, hide them dynamically
- **Maximize**, **Minimize**, **MoveSize** - move and resize windows
- **RunCommand** - run *AOO/LibO* menu commands
- **RunApp** - run an external application
- **OutputTo** - export data to a Calc spreadsheet, a text/csv (comma-separated-value) file or to a formatted html page
- **SendObject** - output a form in another format (PDF, ...) and send it as attachment of a mail
- **Quit** - quit the application

DATABASE FUNCTIONS

- Search a single value with **DLookup**
- Make totals or similar computations with **DSum**, **DAvg**, **DCount**, **DMin and DMax**
- Make use of statistical functions with the **DStDev, DStDevP**, **DVar and DVarP** functions

ERROR HANDLING

- The **Introduction about error handling** - to read first
- **TraceLevel** sets the minimal level for error messages to be traced
- The **TraceError** function may be used by one's own Basic code
- **TraceLog** is for user messages or internal debugging information
- Use **DebugPrint** for debugging

- The logged information can be displayed by mean of **TraceConsole**.

EVENTS HANDLING

- The **Introduction about event handling** - to read first
- The **Events** collection of ...
- ... the **Event** objects with their specific properties

Bookmark this page » » User's Guide