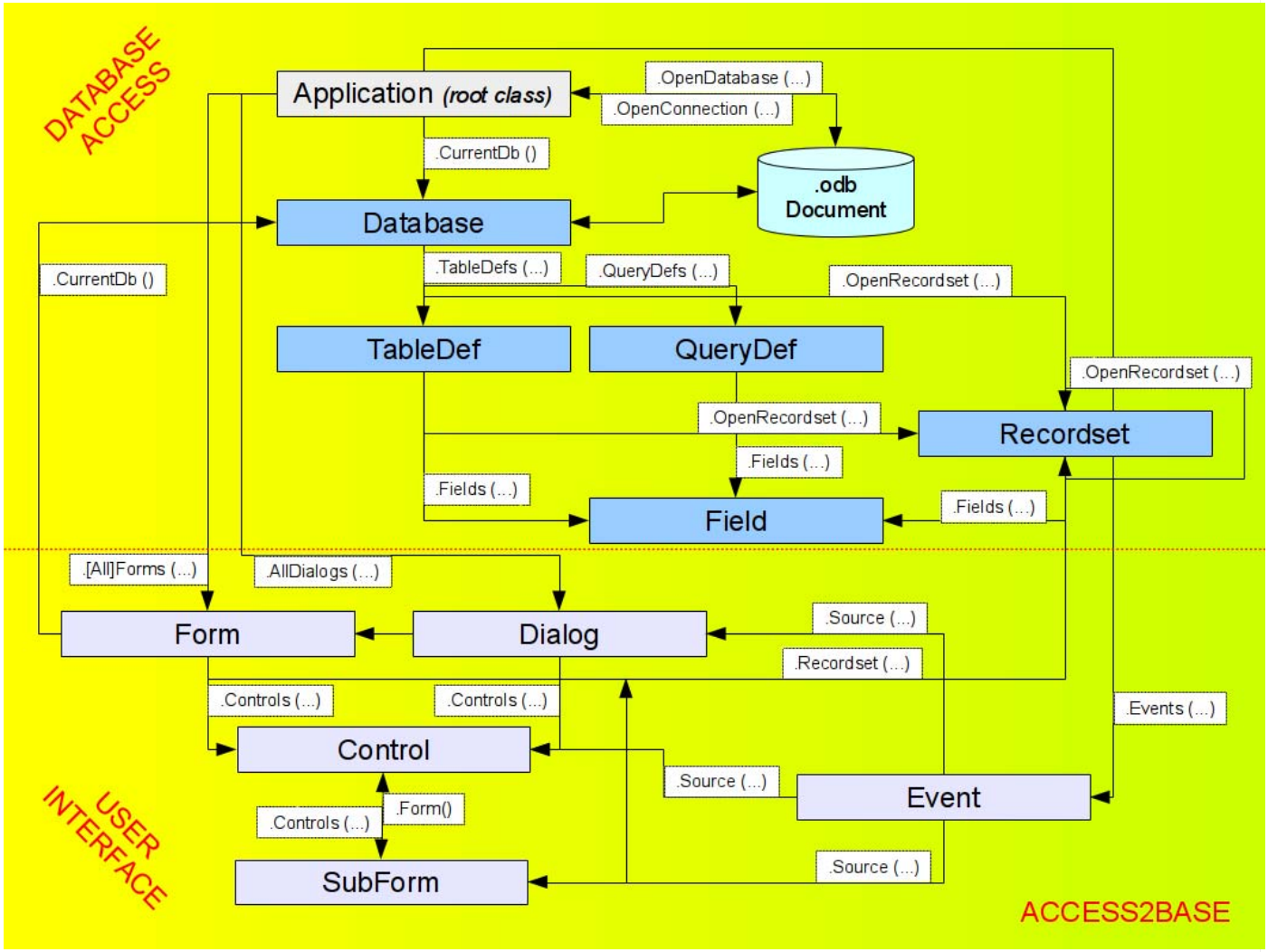
Object Model

AOO/LibO Basic is a simple but still an object-oriented language. To know more about object-oriented programming in Basic, you can have a look at here
To stay close to the syntax of the *MSAccess* object model the implementation of Access2Base has been based on BASIC object classes.

tags: Definitions

- Specific functions, sometimes called "Collections" will create instances of classes of type **Form**, **Subform**, **Control** or **Database**, etc.
- To use these instances in your own code, your local/global/... variables should be declared as being of type `Object`
- The names of the **properties** and **methods** to be applied on those variables are identical in *MSAccess* and in *Access2Base*. Their arguments are identical as well. 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.
- **Indirection**, i.e. accessing a property by its name given as a `string` argument, and **introspection**, i.e. knowing which properties are available for any object, are supported for propertiesl.

The Object Model



Root classes

Class	Description	Collections	Properties	Methods
Application	The root class	AllDialogs AllForms CommandBars Events Forms TempVars	CurrentUser ProductCode Version	CurrentDb DAvg DCount DLookup DMin, DMax DStDev, DStDevP DSum DVar, DVarP OpenConnection OpenDatabase
DoCmd	A secondary root class from which a number of commands can be run		GetHiddenAttribute	ApplyFilter Close CopyObject FindNext FindRecord GoToControl GoToRecord Maximize Minimize MoveSize OpenForm OpenQuery

				OpenReport OpenSQL OpenTable OutputTo Quit RunApp RunCommand RunSQL SelectObject SendObject SetHiddenAttribute ShowAllRecords SysCmd
Collection	An array of objects accessible via their index or their name		Count Item ObjectType	Add Delete Remove RemoveAll

Forms, dialogs, command bars and controls

Class	Description	Collections	Properties	Methods
Form	The representation of an *OpenOffice/LibreOffice* database form	Controls OptionGroup Properties	AllowAdditions AllowDeletions AllowEdits Bookmark Caption CurrentRecord Filter FilterOn Height IsLoaded Name ObjectType OpenArgs OrderBy OrderByOn Recordset RecordSource Visible Width	Close CurrentDb Move Refresh Requery SetFocus
Dialog	The representation of a Basic dialog	Controls OptionGroup Properties	Caption Height IsLoaded Name ObjectType Visible Width	Execute Move Start Terminate
CommandBar	Identifies a toolbar, a menubar or the statusbar.	CommandBarControls Controls Properties	BuiltIn Name ObjectType Visible	Reset
Control	The representation of a control within a Form, a Dialog, a SubForm or an OptionGroup	Controls Properties	BackColor BorderColor BorderStyle Cancel Caption ControlSource ControlTipText ControlType Count Default DefaultValue Enabled FontBold FontItalic FontName FontSize FontUnderline FontWeight ForeColor Form Format ItemData ListCount ListIndex Locked MultiSelect Name ObjectType OptionValue Parent Required RowSource RowSourceType Selected SelLength SelStart SelText SubType TabIndex	AddItem RemoveItem Requery SetFocus

			TabStop Tag Text TextAlign TripleState Value Visible	
SubForm	Identifies a specific control which is a subform of a database form or another subform	Controls OptionGroup Properties	AllowAdditions AllowDeletions AllowEdits Filter FilterOn LinkChildFields LinkMasterFields Name ObjectType OrderBy OrderByOn Parent Recordset RecordSource	Refresh Requery
OptionGroup	Identifies a group of specific controls, i.e. radio buttons.	Controls Properties	Count Name ObjectType Value	
CommandBarControl	Identifies a control within a CommandBar.	Execute Properties	BeginGroup BuiltIn Caption Index ObjectType OnAction Parent TooltipText Type Visible	

Database access

Class	Description	Collections	Properties	Methods
Database	One of the databases to which the application is connected	Properties TableDefs QueryDefs Recordsets	ObjectType	Close CloseAllRecordsets CreateQueryDef DAvg DCount DLookup DMin, DMax DStDev, DStDevP DSum DVar, DVarP OpenRecordset
TableDef	The representation of a Table.	Fields Properties	Name ObjectType	OpenRecordset
QueryDef	The representation of a Query.	Fields Properties	Name ObjectType SQL Type	Execute OpenRecordset
Recordset	The representation of a set of records from a table, a query or a SQL statement.	Fields Properties	AbsolutePosition BOF Bookmark Bookmarkable EditMode EOF Filter Name ObjectType RecordCount	AddNew CancelUpdate Clone Close Delete Edit GetRows Move MoveFirst MoveLast MoveNext MovePrevious OpenRecordset Update
Field	The representation of a field of a table, a query or a recordset.	Properties	DataType DataUpdatable DbType DefaultValue Description FieldSize Name ObjectType Size SourceField SourceTable TypeName Value	ReadAllBytes ReadAllText WriteAllBytes WriteAllText

Other

Class	Description	Collections	Properties	Methods

Property	A name-value pair allowing objects introspection (see below)		**Name** **ObjectType** **Value**	
TempVar	The representation of a temporary variable		**Name** **ObjectType** **Value**	
Event	A description of an occurred form, dialog or control event		ButtonLeft ButtonMiddle ButtonRight ClickCount ContextShortcut EventName EventSource EventType FocusChangeTemporary KeyAlt KeyChar KeyCode KeyCtrl KeyFunction KeyShift **ObjectType** Recommendation RowChangeAction Source SubComponentName SubComponentType XPos YPos	

The object class is returned for all objects as a string by the **ObjectType** property.

Remark about the *Application* and *DoCmd* classes

The *Application* and *DoCmd* classes may be instantiated only once. Their instance MUST be called `Application` and `DoCmd` respectively. In fact they are implemented as module names.
As an example, it is equivalent to write next statements:

```
        DoCmd.OpenForm "anyform"
```

or

```
        OpenForm "anyform"
```

This complies with the *MSAccess* object model. Additionally this freedom solves potential homonymy issues between concurrent Basic libraries.

Collections - Functions returning objects

Objects are created by the invocation of specific functions included in the API. In the calling procedure an object is defined as a variable of type *Object*.
Example:

```
Dim ofForm As Object                    ' Variant would also be correct
        Set ofForm = AllForms("myForm") ' The Set verb is mandatory in MSAccess but optional in AOO/LibO Basic
```

`ofForm` contains after execution the instance of an object of class Form.
`AllForms` is called a **collection**. Individual members of a collection are reachable either by their index or by their name.

Properties

Property types

Within an object class one can distinguish in the current documentation next 2 property types:

Type of property	Description
UNO	The property refers to a UNO object that can be used from user macros to invoke directly alternative properties and methods to those supported by the *Access2Base* API.
Normal	Where it is really about in the *Access2Base* software. Such a property complies with or is close to the *MSAccess* object model.

Property names

The names of the properties are identical to their equivalent in *MSAccess*.
However

- the semantics of the property might differ more or less,
- their arguments might differ,
- the returned values might be different.

Read the documentation about each individual property for more info.

Get properties

To get the value of a normal property named "MyProperty", write :

```
    vValue = myObject.MyProperty
```

Depending on the context an error message can be generated stopping the execution of the macro. The error message can be issued either by AOO/LibO Basic or by the API.

If the property is an array use :

```
    vValue = myObject.MyProperty(i)
```

Set properties

To set the value of a normal property named "MyProperty", use next syntax :

```
    myObject.MyProperty[(i)] = vValue
```

the optional argument being the index if the property returns an array.

The macro will be stopped if the property setting failed.

Indirection

To get the value of a normal property of an object, one can write also:

```
vValue = myObject.getProperty("MyProperty"[, i])
```

To set its value:

```
myObject.setProperty("MyProperty", vValue[, i])
```

Introspection

Finally it is possible to get the value of ALL normal properties of an object with the *Properties* function as in next example:

```
Dim ofForm As Object, i As Integer, oProperty As Object
        Set ofForm = AllForms("myForm")
        For i = 0 To ofForm.Properties().Count - 1
                Set oProperty = ofForm.Properties(i)
                Print oProperty.Name & " = " & oProperty.Value & " / ",
        Next i
        Print
```

Additionally all objects have a `hasProperty` method indicating if the given property argument is applicable or not, like in:

```
Dim ofForm As Object
        Set ofForm = AllForms("myForm")
        If ofForm.hasProperty("AllowEdits") Then
                ...
        End If
```

Methods

Method names

The names of the methods are identical to their equivalent in *MSAccess*.
However

- when the name is a **reserved word** in AOO/LibO Basic the name is preceded by a "m" (like in method). E.g. *mClose* replaces *Close*,
- the semantics of the property might differ more or less,
- the arguments might be different in number or in admitted values or value types.

Read the documentation about each individual method for more info.

Syntax

Example:

```
Dim ofForm As Object
        Set ofForm = AllForms("myForm")
        ofForm.Move(100, 200)
```

Compatibility

Existing programs written with *Access2Base* in versions older than 0.9.0 keep running normally. The invocation of properties and methods like in

> `getPROPERTY(OBJECT)`
> `setPROPERTY(OBJECT, `*value*`)`
> `METHOD(OBJECT, `*arg1*`, ...)`}}

is still supported.
It might not be supported anymore in a later release.

Bookmark this page » » Object Model