Working with Objects and Collections
Understanding Objects and Collections

Възможно е да създадете едно приложение в Access без въобще да програмирате. Но чрез Visual Basic код, получавате много по прецизен контрол върху това, какво прави приложението. Когато програмирате във Visual Basic, в действителност работите с обекти (*objects*), които кореспондират с различни аспекти в базата от данни. Групите (*collections*
) са комплекс от обекти от един тип.

Обектите достъпни за Вас в Access са от четири различни източника:

- Access предоставя обекти, използвани за показване на данните, такива като обект Form, обект Report и обект Control.
- Microsoft DAO предоставя Data Access Objects (DAO), такива като TableDef и QueryDef обекти, които определят структурата на базата от данни и с които може да се манипулират данните в базата от данни.
- Visual Basic предоставя обекти, които Ви дават гъвкавост при програмирането, такива като Debug и Err обектите.
- Microsoft Office предоставя обекти които можете да използвате за управление на интерфейса в приложението, такива като CommandBar и FileSearch обектите.

Много от обектите, с които се работи във Visual Basic кореспондират с части от базата от данни. Например, Access Form и Report обектите кореспондират с Вашите форми и отчети, докато DAO TableDef и QueryDef обектите кореспондират с Вашите таблици и запитвания. Microsoft Office CommandBar обекта кореспондира с менютата от Access.

Други обекти с които работи Visual Basic са в значителна степен абстрактни. Например Err обекта съдържа информация, свързана с грешки открити при изпълнението на код, само в известен смисъл кореспондират с базата от данни.

Organization of Objects and Collections

Всяко приложение поддържа библиотека на обекти (*objects library*). Библиотеката на обектите съдържа информация за обектите от приложението и техните характеристики и методи. Access включва четири вградени библиотеки на обекти,

разглеждани в следващите раздели.

Access и DAO обектите са организирани в йерархия на обектите (object hierarchies). В тази йерархия, едни обекти съдържат други обекти и групи (*collections*). Групата е специален тип обект, който е набор от обекти от даден тип. Например, Application обекта съдържа групата Forms, включваща отделните индивидуални форми. От своя страна обекта съдържа групата Controls, съставена от отделните контроли. Следващата илюстрация показва тези отношения.

Можете да възприемате групата като масив от вече декларирани обекти. Групите, подобно на масивите имат елементи, и Вие се обръщате към обектите от групата чрез техните имена или на базата на местоположението им в групата.

Microsoft Access Objects

Вие вече познавате някои от обектите в Microsoft Access 8.0 библиотеката. Обектите на Access Form, Report и Control кореспондират с Вашите форми, отчети и контроли. Тези обекти се ползват за контролиране на начина за показване на данните от базата с данни. Обектите Application и Reference предоставят възможност да се работи с обекти от други приложения. Обекта Module осигурява контрола върху модулите в базата от данни. Обекта DoCmd се използва за включване на действия чрез макрос в Visual Basic кода. Обекта Screen се ползва за обръщане към активния върху екрана обект. Следващата таблица описва обектите от Microsoft Access 8.0 библиотеката.

Object	Description
Application	

Active Microsoft Access application

Control

Control on a form or report

DoCmd

Macro actions used in Visual Basic code

Form

Open form, including subforms

Module

Open standard module or class module

Reference

Reference to an application’s object library

Report

Open report, including subreports

Screen

Screen that currently has the focus

See Also For information on an individual object, search the Help index for the name of the object.

От обектите представени в таблицата, обектите Control, Form, Module, Reference и Report са елементи от групи. Обекта Application е на най-горното ниво в йерархията. Илюстрацията на следващата страница представя йерархичната организация на Access обектите и групите.

DAO Objects

Microsoft DAO 3.5 библиотеката на обектите предоставя DAO обекти, които се ползват за работа с Microsoft Jet database engine. Някои DAO обекти представят структурата на базата от данни и връзките между данните в нея. Тези обекти са Database, TableDef, QueryDef, Field, Index, Parameter, Property и Relation. Други обекти са свързани със сигурността на базата от данни и това са обектите Container, Document, User, Group и Workspace. Обекта Recordset Ви предоставя директен достъп до данните в базата от данни. Обекта DBEngine предоставя възможността за контрол над самия него. Обекта

Connection представя мрежова връзка към Open Database Connectivity (ODBC) базата от данни и е достъпен само при работа с ODBCDirect Workspace обект.

See Also For more information on the Connection object and ODBCDirect, search the Help index for “Connection object.”

Следващата таблица описва DAO обектите от Microsoft DAO 3.5 библиотеката.

Object	Description
Connection	Network connection to an ODBC database
Container	Security and other information for various types of objects in the database

Database

Open database

DBEngine

Microsoft Jet database engine itself

Document

Security and other information for individual objects in the database

Error

Data access error information

Field

Field in a table, query, recordset, index, or relation

Group

Group account in Microsoft Jet's current workgroup

Index

Table index

Parameter

Query parameter

Property

Property of an object

QueryDef

Saved query in a database

Recordset

Set of records defined by a table or query

Relation

Relationship between two table or query fields

TableDef

Saved table in a database

User

User account in Microsoft Jet’s current workgroup

Workspace

Active Microsoft Jet session

Всеки обект в Microsoft DAO 3.5 принадлежи на група (без DBEngine. обекта). Този обект е на най-високото ниво и предоставя достъп до всички останали обекти в групата, по подобие на обекта Application от Access йерархията. Следващата илюстрация представя йерархичната организация на DAO обектите, като за опростяване се представя само DBEngine обекта и групите за всички останали обекти в йерархията на обектите.

Всеки DAO обект има група Properties. Тази група съдържа Property обекти, които представляват характеристиките, специфични за всеки обект от DAO обектната йерархия.

See Also For more information on the Properties collection, see “The Properties Collection” later in this chapter.

Visual Basic Objects

Visual Basic for Applications обектната библиотека предоставя три обекта на Access, но те не са организирани в йерархия на обекти. Нито един от тях не принадлежи на група. Следващата таблица представя обектите от Visual Basic for Applications библиотеката на обектите.

Object

Description

Collection

User-defined collection

Debug

Debug window

Err

Information about Visual Basic errors

Microsoft Office Objects

Microsoft Office 8.0 библиотеката на обекти, предоставя обекти, които могат да се ползват за управление на това, което се появява в едно приложение. Например, може да се създаде потребителска ивица с бутони и потребителско меню чрез обекта CommandBar. Също така, може да се осъществи търсене на потребителски файл посредством обекта FileSearch. Възможно е да се управлява и Office Assistant в съответствие с действията на потребителя.

Note In order to use objects in the Microsoft Office 8.0 object library from Visual Basic, you

must first set a reference to the object library. When you set a reference to an object library, you notify Visual Basic that you may want to use the objects in that library. To set a reference to the Microsoft Office 8.0 object library, open a module and click References on the Tools menu. Then select the Microsoft Office 8.0 Object Library check box in the Available References box.

Не всички обекти от Microsoft Office 8.0 библиотеката на обектите се използват в Access. Следващата таблица описва някои от обектите в тази библиотека, които се ползват от Access.

Object

Description

Assistant

The Office Assistant

Balloon

Balloon associated with the Office Assistant

BalloonCheckBox

Check box control for the balloon

BalloonLabel

Label control for the balloon

CommandBar

Toolbar, menu bar, menu, or shortcut menu

CommandBarButton

Button on a CommandBar object

CommandBarComboBox

Combo box control on a CommandBar object

CommandBarControl

Any control on a CommandBar object

CommandBarPopup

Pop-up control on a CommandBar object

FileSearch

Microsoft Office file searching

FoundFiles

Files found through file search operation

Note The Office Assistant is not available in Microsoft Access run-time applications.

Working with Objects and Collections

Сега вече, след запознаването с обектите достъпни в Access, може да се премине към работата с тях във Visual Basic.

Referring to Objects

За да се използва един обект във Visual Basic , той трябва да бъде посочен. Обектите са два основни типа - едните са индивидуални обекти, а другите принадлежат на група.

Индивидуалните обекти (тези, които не принадлежат на група) могат да бъдат посочвани директно. Например, можете да се позовете на обекта Application във Visual Basic по следният начин:

Application

Други обекти принадлежат на група, и трябва да се посочи с кой обект от групата желаете да работите, а също така - и коя група съдържа този обект.

Възможни са три подхода за обръщане към обект от група. Най-краткият е да се посочи името на групата, последвано от името на обекта който Ви е необходим, както е посочено в следващите примери:

Forms!Employees

QueryDefs![Current Product List]

Операторът “!” се ползва за разделител между името на групата и името на обект от нея. Ако името на обекта съдържа интервали, то трябва да се постави в (средни) скоби. Трябва да се помни, че групата Form включва само текущо отворените форми, т.е. ако формата Employees не е отворена и се направи опит за изпълнение на кода от предходния пример, ще се генерира съобщение за грешка. Същото важи и за групата Reports.

В много от случаите, няма да знаете името на обекта, който трябва да ползвате и няма да може да ползвате този синтаксис. В този случай може да се ползва стрингова променлива в която да се съдържа името на обекта. В следващите примери strFormName и strQueryDefName са стрингови променливи, съдържащи името на обект Form и на обект QueryDef.

Forms(strFormName)

QueryDefs(strQueryDefName)

Ако на strFormName стойността е "Employees" и на strQueryDefName стойността е "Current Product Name", тогава предходният пример е еквивалентен на следващия:

Forms("Employees")

QueryDefs("Current Product Name")

Може да се обръщате към обект от група по неговия индекс. Индекса е номера на обекта в групата. В следващите примери се използва индекс за обръщане към отделен обект в група.

Forms(0)

QueryDefs(1)

Повечето групи са индексирани, като се започва от нула. Така първият обект от групата има индекс 0, вторият 1 и т.н. В първият от предходните примери има обръщане към първият отворен Form обект в групата Forms. В повечето случаи индексите на Form обектите в групата съответстват на поредността на отварянето им.

Note Some collections, such as the Microsoft Office CommandBars collection, are indexed beginning with 1 rather than 0. To determine how a particular collection is indexed, search the

Help index for the name of that collection.

Вторият от предходните примери се обръща към вторият обект QueryDef в групата QueryDefs. Тази група включва всички съхранени запитвания в базата от данни, независимо дали те са отворени. В повечето случай QueryDef обектите и други обекти се индексират в зависимост от поредността на създаването им в базата от данни.

Когато се обръщате към обект по някои от посочените начини, Visual Basic връща мястото за връзка с обекта (*object reference*). Мястото за връзка с обекта е мястото, което той заема в оперативната памет и където той съществува. Когато се работи с обект във Visual Basic, то в действителност се работи с мястото за връзка на обекта в паметта.

Referring to Objects in a Default Collection

Много обекти в Access съдържат една или няколко групи. Една от тях се приема за група по подразбиране. Например, групата Controls е групата по подразбиране за обекта Form, то може да се ползва обръщане към един контрол без да се посочва изрично името на групата.

Следващият код връща мястото за връзка с контролът LastName във формата Employees, като се ползва подразбиращата се група.

```
Forms!Employees!LastName
```

Може да се ползва и явно указване на групата както е в следващият код:

```
Forms!Employees.Controls!LastName
```

Следващата таблица описва групите по подразбиране за някои обекти.

Object library

Object

Default collection

Microsoft Access

Form

Controls

Report

Controls

DAO

Container

Documents

Database

TableDefs

DBEngine

Workspaces

Group

Users

Index

Fields

QueryDef

Parameters

Recordset

Fields

Relation

Fields

TableDef

Fields

User

Groups

Workspace

Databases

Declaring and Assigning Object Variables

Променливата на един обект (*object variable*) е променлива, която представя съответния обект във Visual Basic. Използването й, води до значително съкращаване на програмния код.

Променлива за обект се обявява по сходен начин с обявяването на променливите, чрез операторите Dim, ReDim, Static, Private и Public. Следващият примерен код декларира променлива на обект от тип Form.

```
Dim frm As Form
```

See Also For information on declaring variables, see Chapter 4, “Working with Variables, Data Types, and Constants.”

След като е декларирана една такава променлива, тя може да се използва като място за връзка (object reference). Когато се нуждаете да се обръщате към един обект многократно, най-удачно е да се създаде променлива за обекта.

За присвояването на място за връзка в паметта (object reference) към променливата на обекта се ползва оператор Set. Следващият код присваява място за връзка на обекта Employees към променливата на обект, декларирана в предходния пример.

```
Set frm = Forms!Employees
```

В това се състой и съществената разлика при използването на Set оператор за

присвояване на стойност на другите променливи, например от тип String или Integer. Обикновено променливите съхраняват стойности, докато променлива за обект сочи към място за връзка в паметта, а не към обект. В действителност винаги се работи с мястото за връзка с обекта в програмният код, а не със самият обект

The Nothing Keyword

Чрез ключовата дума Nothing може да се освободи системната памет, ангажирана до конкретен момент от променливата на обект. Тази ключова дума се използва заедно с оператор Set. Например, ако няма нужда повече от променливата на обекта frm, то може да се ползва следният код:

```
Set frm = Nothing       ' Where frm is a Form object variable.
```

След изпълнение на този код, променливата продължава да съществува и на нея може да се присвой друг обект, ако това е необходимо.

Using Objects and Collections in Code

След като разбрахме как да се обръщаме към обектите във Visual Basic и как се създава и ползва променлива на обект, може да се разгледа използването им в код.

Navigating the Object Hierarchy

Когато един обект принадлежи на група, то трябва при обръщане към него да се посочва и името на групата. Ако групата в йерархично отношение принадлежи на друга група, то трябва да се посочва и нейното име и т.н.

Когато се създава променлива за обект и й се присвоява обект, то информацията за позицията на този обект се съхранява в променливата.

Следващият пример показва как се ползва йерархията на обектите за достъп до тях. Процедурата връща мястото за връзка към обекта Employees, който е член на групата Forms. След това кода връща мястото за връзка на обекта LastName, който е член на групата Controls. Накрая, се проверява характеристиката ControlType , за да се

определи вида на контрола. Ако този контрол е текстова кутия, то процедурата установява характеристиката Locked на стойност True.

```
Sub LockControl()

Dim frm As Form, ctl As Control                ' Declare object variables.

Set frm = Forms!Employees                                ' Return reference to Form
object.

Set ctl = frm!LastName                                   ' Return reference
to Control object.

If ctl.ControlType = acTextBox Then            ' Check ControlType property.

ctl.Locked = True                                        ' Lock control if it's a text box.

End If

Set frm = Nothing

End Sub
```

При работа с обекти и групи от DAO йерархията се ползва същият подход. Следващият пример Ви сочи това.

```
Sub ListTableFields()

' Declare object variables.

Dim dbs As Database, tdf As TableDef, fld As Field

' Return reference to current database.

Set dbs = CurrentDb

' Return reference to Employees table.

Set tdf = dbs.TableDefs!Employees

' Print out all fields in the table.

For Each fld In tdf.Fields

Debug.Print fld.Name

Next fld
```

```
Set dbs = Nothing


End Sub
```

Enumerating the Objects in a Collection

В предходния пример, процедурата трябва да провери (*enumerate*) всички Field обекти в групата Fields на обекта TableDef. Това се постига чрез оператора For Each...Next.

За да се използва този оператор първо трябва да се установи кои обекти ще се проверяват. След това се обявява променлива съответстваща на обекта. В предходния пример това е променливата fld от тип Field. Вътре в оператора For Each...Next, тази променлива е връзката към всеки обект в групата Fields. Чрез използването на променливата, може да се изпълняват методи или променят характеристики на всеки обект от групата, без да е предварително ясно колко са обектите от групата.

See Also For more information on the For Each...Next statement, search the Help index for “For Each...Next statement.”

Adding New DAO Objects to a Collection

Както вече е известно, някои DAO обекти представят структурата на базата от данни, а други служат за работа с данните, разположени в нея. Обектите, представящи структурата на базата от данни се съхраняват като нейни компоненти. Обектите, служещи за работа с данните в базата от данни не се съхраняват с нея, а се създават всеки път, когато има нужда от тях.

Когато се създава нов DAO обект, който ще се съхрани с нея, то той трябва да се добави в групата на съхранените обекти. Следващият пример създава нов TableDef обект с име ArchivedInvoices с нов Field обект с име OrderID.. Той добавя новият Field обект към групата Fields на новия TableDef обект, и добавя TableDef обекта към групата TableDefs на Database обекта, представляващ текущата база от данни. След изпълнението на този код, в таб-а Tables на Database прозореца се появява нова

таблица

```
Sub AddTable()

' Declare object variables.

Dim dbs As Database, tdf As TableDef, fld As Field

' Assign the current database to the database variable.

Set dbs = CurrentDb

' Create new table and field, and assign to table and field variables.

Set tdf = dbs.CreateTableDef("ArchivedInvoices")

Set fld = tdf.CreateField("OrderID", dbLong)

' Add field to table's Fields collection.

tdf.Fields.Append fld

' Add table to database's TableDefs collection.
```

```
dbs.TableDefs.Append tdf

' Refresh TableDefs collection.

dbs.TableDefs.Refresh

Set dbs = Nothing

End Sub
```

Note The preceding example uses the CurrentDb function to return a reference to the current database, and assigns this reference to an object variable of type Database. Anytime you're writing code to work with the database that's currently open, you should use CurrentDb to return a reference to the current database.

The Properties Collection

DAO обектите и обектите на Microsoft Access Form, Report и Control притежават група Properties. Съдържащото се в тази група кореспондира с характеристиките на обекта.

Например, следващата процедура извежда имената на всички характеристики в Debug прозореца.

```
Sub DisplayProperties()

' Declare variables.
```

```
Dim dbs As Database, frm As Form, prp As Property

' Return reference to current database.

Set dbs = CurrentDb

Debug.Print "Current Database Properties"

' Enumerate Properties collection.

For Each prp In dbs.Properties

Debug.Print prp.Name

Next prp

' Print blank line.

Debug.Print

Debug.Print "Employees Form Properties"
```

```
' Open Employees form in Form view.

DoCmd.OpenForm "Employees", acWindowNormal

' Return reference to Employees form.

Set frm = Forms!Employees

' Enumerate Properties collection.

For Each prp In frm.Properties

Debug.Print prp.Name

Next prp

Set frm = Nothing

Set dbs = Nothing

End Sub
```

Note If you’re looping through the Properties collection of a table or query, some properties aren’t displayed because they’re added to the collection only when they have a value.

See Also For more information on the Properties collection, search the Help index for “Properties collection.”

Working with CommandBar Objects

Създаването на нов CommandBar обект е различно от създаването на други нови обекти в Access. За целта се използва Add метод на CommandBar групата. Следващият код създава нов CommandBar обект и добавя бутон в него:

```
Sub CreateNewCommandBar()

Dim cmb As CommandBar, cbc As CommandBarControl

' Create new CommandBar object and return reference to it.

Set cmb = CommandBars.Add("NewCommandBar", msoBarFloating)

' Create new CommandBarControl object and return reference to it.

Set cbc = cmb.Controls.Add(msoControlButton)

' Set properties of new command bar control.

With cbc
```

```
.Caption = "Button1"

.DescriptionText = "First button in NewCommandBar"

.OnAction = "Button1Function()"    ' Run this function when button is pressed.

.Visible = True

End With

' Make command bar visible.

cmb.Visible = True

Set cmb = Nothing

End Sub
```

Note In order to use objects in the Microsoft Office 8.0 object library from Visual Basic, you must first set a reference to the object library. When you set a reference to an object library, you notify Visual Basic that you may want to use the objects in that library. To set a reference to the Microsoft Office 8.0 object library, open a module and click References on the Tools menu. Then select the Microsoft Office 8.0 Object Library check box in the Available References box.

Creating New Objects with Class Modules

Всеки обект в Access представен от уникалната дефиниция на обекта. Тя включва името

му, специфичните му характеристики, неговите методи и събития. Дефиницията на обекта се именува Class.

Във вашите възможности е да дефинирате нов потребителски обект в class module, свързан с обектните библиотеки. Class module е модул, който може да съдържа дефиниция за нов обект.

To create a definition for a new object in a class module

1 1 Дефинирайте целта на съществуването на новия обект. Мислете за него с термините на методите и характеристиките, който трябва да притежава. Например, извикване на функции от dynamic link library (DLL) често е твърде сложно. Можете да създадете обект, който притежава методите, които ще извиквате. След това, когато желаете да извикате някоя функция, можете просто да извикате метода който да я съдържа.

2 2 Създава се нов class модул чрез чукване на Class Module от Insert командата. Изберете име за този class и съхранете модула с посоченото име.

3 3 Добавете процедури в новия модул. Всяка Sub или Function процедура която дефинирате в class модула става потребителски метод на новия обект. Всяка Property Get, Property Let или Property Set процедура която дефинирате става потребителска процедура на новия обект.

4 4 Ако желаете част от кода да се ползва когато само част от class модула е създаден, добавете този код към Initialize процедурата на class модула. В противен случай го добавете към Terminate процедурата.

See Also For more information on the Initialize and Terminate events, search the Help index for “Initialize event” or “Terminate event.”

5 5 Тествайте новия модул, като създадете нов екземпляр (instance). За целта ползвайте ключовата дума New, създаваща нов екземпляр. Например, ако class е именуван NewClass, може да се създаде нов екземпляр както е показано в следващият пример:

```
Dim obj As New NewClass
```

Ако сте дефинирали метод извикващ ListNames от class модула, то може да се запише следното:

```
obj.ListNames
```

See Also For more information on the New keyword, search the Help index for “New keyword.”

Можете да видите новия class и неговите променливи, методи и характеристики чрез Object Browser, който е достъпен чрез View командата в Module прозореца. В Project/Library кутията чукнете върху името на проекта, а след това върху името на class в Classes кутията. Името на проекта се задава в кутията Project Name на таб-а Advanced на Options диалоговата кутия (Tools команда).

See Also For more information on the Object Browser, see “Using the Object Browser” later in this chapter. For more information on programming with class modules, search the Help index for “class modules.”

Creating Multiple Instances of Forms and Reports

Модулите на форма или отчет са също class модули. Когато създавате нов екземпляр, той има същите характеристики и методи като оригиналната форма или отчет. Допълнително, всяка процедура, включена в class модула на формата или отчета става като метод или характеристика на новия екземпляр.

За създаване на нов екземпляр class на форма или отчет, се декларира нова променлива на обекта, чрез ключовата дума New и името на class модула на формата или отчета. Името на class модула се появява в заглавната ивица на модула. То показва, че class е свързан с формата или отчета. Например, името на class за Orders формата е Form Orders. Следващият код създава нов екземпляр на Orders формата.

```
Dim frmInstance As New Form_Orders
```

Когато се създава екземпляр на class на формата чрез ключовата дума New, той е “скрит”. За показването му трябва да се определи Visible характеристиката като True.

Декларирането на променливата представяща новия екземпляр в class на формата трябва да се прави на модулно ниво.

Note When you create a new form or report in Microsoft Access, the form or report doesn’t automatically have an associated module. Forms and reports without associated modules load more quickly. If you’re working in form or report Design view, Microsoft Access automatically creates the form or report module when you click Code on the View menu. Once you enter code in the module, Microsoft Access saves the module with the form or report.

Whether or not the form or report module exists is determined by the setting of the HasModule property. When a form or report is created, the HasModule property is automatically set to False. When you create a form or report module by clicking Code on the View menu, Microsoft Access sets the HasModule property to True. If you refer to the Module property of a form or report, the HasModule property is also automatically set to True. For more information on the HasModule property, search the Help index for “HasModule property.”

Working with Properties and Methods

За описването на специфичното за обектите, се използват техните характеристики. Контрола върху поведението на обекта се осъществява с методите му.

Тъй като групата е също обект, всяка група в Access има характеристики и методи.

Манипулирането с тях е същото, както за обектите.

See Also For information on the properties and methods an object supports, search the Help index for the name of the object. You can also search the Help index for the name of a property or method.

Setting and Reading Properties

Visual Basic ползва стандартен синтаксис за задаване и получаване на стойностите за характеристиките. Когато задавате характеристика, Вие и давате просто нова стойност. Синтаксиса е следният:

object.property = setting

Следващият код задава Caption характеристика за формата Employees.

Forms!Employees.Caption = "Employees Form"

Когато се чете характеристиката, Вие получавате нейната текуща стойност. В следващият пример се присвоява стойността на Caption характеристиката на променлива, и след това се извежда стойността на тази характеристика в диалогова кутия.

Dim strCaption As String

strCaption = Forms!Employees.Caption

MsgBox strCaption

Properties That Return Objects

Понякога ще желаете да следите какво се случва с обектите по време на изпълнението на кода. Например, ще пожелаете може да пожелаете да скриете един контрол, веднага след напускането му от фокуса.

Аналогично на следенето на отделни характеристики на обектите, някои от характеристиките представят връзките на обект с другите обекти. Тези характеристики връщат мястото за връзка с обекта, с което може да се работи директно. Следващата таблица е списък на някои от характеристиките, които връщат обектите.

Property	Applies to	Returns a reference to
ActiveControl	Screen, Form, or Report object	The Control object that has the focus.

ActiveForm

Screen object

The Form object that has the focus or that contains the control with the focus.

ActiveReport

Screen object

The Report object that has the focus or that contains the control with the focus.

Application

Numerous objects

The active Microsoft Access Application object.

DBEngine

Application object

The current DBEngine object.

Form

Subform Control object

The Form object associated with the subform control.

Me

Form or Report object

The Form or Report object in which code is currently running.

Module

Form or Report object

The Module object associated with a Form or Report object.

Parent

Numerous objects

The object or collection that contains an object.

PreviousControl

Screen object

The Control object that had the focus immediately before the currently active control.

RecordsetClone

Form object

A clone of the form’s underlying recordset.

Report

Subreport Control object

The Report object associated with the subreport control.

Section

Form, Report, or Control object

A section on a form or report.

See Also For information on properties that return objects, search the Help index for the specific property name.

The Section Property

Характеристиката Section връща мястото за връзка с раздела на формата. Например, може да се използва Section характеристиката за връщането на мястото за връзка на detail раздела на формата. След като го получите, можете да работите с характеристиките на раздела. Следващият пример използва Section характеристика за задаване на характеристика на detail раздела на формата Employees.

```
Forms!Employees.Section(acDetail).Visible = False
```

See Also For more information on setting properties of sections, search the Help index for “Section property.”

The Me Property

Характеристиката Me връща мястото за връзка с обекта Form, в който се изпълнява текущият код. Чрез тази характеристика може да се обръщате към формата, без да се

интересувате от нейното име.

Например, следващият код използва Me характеристиката за връщане мястото за връзка с формата Employees в която се изпълнява този код. След това се предава мястото за връзка на процедурата ChangeDetailColor в отговор на Current event. То използва също Me характеристиката за връщане на мястото за връзка на формата Employees и връщане на стойности за контролите FirstName и LastName във формата. Обърнете внимание, че е използван оператор “.” за характеристиката и оператор “!” за обръщане към контрол от формата.

```
' Place this procedure in a standard module.

Sub ChangeDetailColor(frm As Form)

frm.Section(acDetail).BackColor = RGB(Rnd * 256, Rnd * 256, Rnd * 256)

End Sub

' Place this procedure in the form module associated with the Employees form.

Private Sub Form_Current()

ChangeDetailColor Me

Me.Caption = Me!FirstName.Value & " " & Me!LastName.Value
```

End Sub

В повечето случаи характеристиката Me е самата форма представяна от ActiveForm характеристиката на обекта Screen. Но разликата съществува - ActiveForm е активната в момента форма, докато Me винаги сочи формата от която е изпълняваният код.

Using Methods

Методите са вградени операции които може да се задават над обект. За задаване на метод към обект се ползва следния синтаксис:

object.method [[(] arg1, arg2...[)]]

Повечето методи приемат един или няколко аргумента. Аргумента предоставя на метода допълнителна информация за операцията. Ако метода връща стойност или обект, трябва списъка от аргументи да се загради в скоби.

Следващият пример обяснява разликите. Методът OpenRecordset създава нов Recordset обект и връща мястото за връзка на новия обект. Може да се присвои това място за връзка на променлива на обект чрез Set оператор. Тъй като OpenRecordset методът връща стойност, то неговите аргументи трябва да са в скоби. Същевременно FindFirst методът, който не връща стойност и неговите аргументи не трябва да се поставят в скоби. Същото се отнася и за методите Print и Close.

Sub FindEmployee()

Dim dbs As Database, rst As Recordset

Set dbs = CurrentDb

```
' Requires parentheses.

Set rst = dbs.OpenRecordset("Employees", dbOpenDynaset)

rst.FindFirst "[HireDate] >= #1-1-93#"                    ' No parentheses needed.

Debug.Print rst!LastName                                                    ' No
parentheses needed.

rst.Close                                                                   '
Doesn't take arguments.

Set dbs = Nothing

End Sub
```

Performing Multiple Actions on an Object

Често ще Ви се налага да задавате радлични действия с един и същи обект. Вместо да задавате това да се изпълнява с отделни оператори, използвайте специално създадения за такива случаи оператор With...End With. В следващият пример чрез него се задават нови стойности за характеристики на командния бутон HelpButton в Load събитието на формата.

```
Private Sub Form_Load()
```

With Me!HelpButton

.Caption = "Help"

.Visible = True

.Top = 200

.Left = 5000

.Enabled = True

End With

End Sub

Using the Object Browser

Object Browser е инструмент, предоставящ информация за всички обекти от едно приложение.

Note Some objects show up in the Object Browser automatically when you start Microsoft Access. Others, such as the Microsoft Office objects, show up only after you have set a reference to the object library that contains them.

To use the Object Browser

1 1 Отворете модул.

2 2 От View командата, изберете Object Browser.

3 3 В кутията Project/Library, изберете библиотеката на обекта, която желаете да видите или изберете за да видите всички библиотеки заедно. В кутията Classes са всички обекти от избраната библиотека.

4 4 В кутията Classes изберете един обект. В кутията Members Of е списъка на методите, характеристиките, събитията и константите свързани с избрания обект.

Например, ако зададете Access в Project/Library кутията, то всички обекти от библиотеката на обектите от Access ще се появят в кутията Classes. Ако изберете обект от тази кутия, то ще видите всички методи, характеристики, събития и константи свързани с този обект. Ако например избе-рете Control в кутията Classes, то ще видите методите и характеристиките на този обект в кутията Members Of.

От средата на Object Browser може да получите помощна информация за всеки обект, метод, характеристика или събитие. За целта трябва да се избере това за което се интересувате и да натиснете Help бутона. Може да се копира всеки обект в Clipboard, чрез избирането му и последващо натискане на Copy бутона, след което съдържанието на Clipboard може да се вмъкне в нов код.

See Also For more information on the Object Browser, search the Help index for “Object Browser.”