CHAPTER 13

# Referring to Objects

Up to this point in the book, you have learned how to design a database structure and how to create queries, forms, and reports to enter and retrieve data. But Access is more than a pretty interface; it is a full programming environment that enables you to respond to user input in dozens of different ways.

## What Are Objects?

**What** The objects referred to in this chapter are the forms and reports that you are already familiar with and the controls they contain. So far, you have used only the basic controls, in basic ways. In this chapter, you'll learn to manipulate these and other controls such as command buttons and option groups. In following chapters, you will discover that almost everything in Access is treated as an object, including table structures, query definitions, and the data itself. The techniques explained in this chapter are also essential for manipulating the less-visible programming objects.

These objects (forms, reports, and controls) have properties and events. Properties define how an object can be used; events define what happens when it is used. Creating an Access application is a process of creating objects, defining their properties, and programming responses for their events.

## Why Do I Need This?

**Why**

Learning how to refer to an object empowers you to manage the data it contains, and to control what the object can do. You need to know how to refer to the object before you can do the following:

- Pass a value from one control to another.
- Prevent the entry of data that doesn't make sense.
- Minimize the user's typing by assisting with data entry.
- Hide distracting detail, displaying it only when needed.
- Pop up another form to display or request additional information.
- Request confirmation before writing major changes.
- Add buttons that automate common processes.

These tasks are not difficult, but before you can perform them, you must know how to refer to the objects you want to use. If you have worked previously with object-oriented languages, the object hierarchy in Access will be straightforward. On the other hand, if you have no previous experience with objects and their properties, you might begin to wonder just how deep this system dives and why anyone would type half a line of text just to refer to a single item.

As an analogy, consider the way you address people via mail:

Name,
Number and street,
City,
Country

Because you write the most specific item first, the post office must read the address in reverse order. When the country has been identified, your letter goes to the appropriate city, and so on backwards through the address. It would be more logical to start with the most general name first. In Access, you begin with the name of the collection an object belongs to, and provide more specific data about its names and properties until you have described precisely what you want.

If you refer to an object incorrectly, Access displays its contents as

```
#Name?
```

In some cases, the problem is as simple as a spelling mistake, or a control that has been renamed so that the reference is no longer valid. In other cases, you might have assumed Access knows where to go to get the value you need—from another form for example—or you might have forgotten to include some of the syntax. To remedy the problem, open Properties and check the `ControlSource`, `RecordSource`, or `RowSource` property.

## How to Refer to Controls on Forms

**How**

Access keeps track of all open forms as a collection called (not surprisingly) Forms. Note that the Forms collection refers to the forms currently open, not to all the saved forms. For example, if you are using a form called frmClients, you can refer to it as

```
Forms![frmClients]
```

The bang (exclamation mark) separates the name of the object from the collection it belongs to. Square brackets are used around the object name, although they can be omitted if the name contains no spaces or other odd characters.

This form contains a control called Surname. Following the same principles, this control can be referred to as

```
Forms![frmClients]![Surname]
```

Use that reference, and Access looks up the control and returns the value it contains for the current record. For example, you can enter this reference in the criteria row of a query, and when you run your query, Access will limit the dynaset to people with the same surname as the record in the form.

In addition to reading a value from the object, it is also possible to write a value into it. Using the `SetValue` action in a macro (see Chapter 14, "Macros"), you can alter the contents of a control by setting it to a new value.

As you are aware from the Properties window, each object also has a number of properties. Although you cannot see the Properties window while the form is in use, you can still refer to a property such as `Visible` as follows:

```
Forms![frmClients]![Surname].Visible
```

By setting the `Visible` property to No, you can hide a control on a form. If you set

```
Forms![frmClients].Visible
```

to No, you hide the entire form. A form also has a `Caption` property, which defines the text to be displayed in the form's title bar. To change the caption while the form is in use, merely set a new value for the property:

```
Forms![frmClients].Caption
```

Several of the properties are two or more words in length. To refer to these properties, do not include the space. For example, forms have a `RecordSource` property (referring to the table or query providing records for the form), and text boxes have a `ControlSource` property (referring to the field from the table that provides data for the control). These properties are referenced as follows:

```
Forms![frmClients].RecordSource
```