CHAPTER 10

Building Reports with the Microsoft Data Report Designer

In this chapter

- The Data Report Designer
- Adding the Data Report to your program
- Previewing a report
- Printing a report
- Exporting reports
- Presenting the data report object model

In this chapter, I'm going to talk about the Microsoft Data Report and show you how to incorporate it into your application.

Introducing the Microsoft Data Report

Contrary to popular belief, paper reports are far from dead. Your applications need the ability to produce reports from their database. Microsoft Data Report is the answer. Microsoft Data Report is a full-featured report generator whose features include:

- **Controls** – includes the familiar `Label`, `Shape`, `Image`, `TextBox`, and `Line` controls used in creating Visual Basic forms, plus the `Function` control that allows you to compute the sum, average, minimum or maximum value for a database field.
- **Drag and drop design** – works just like forms in Visual Basic. You select controls from a toolbox and drop them on your report. The controls support data binding and work with the Data Environment Designer to make it easy to define your data source.

- **Print preview** – allows your users to see what their reports will look like before they print them.
- **File export** – means that you can save the generated report as a text file or as an HTML file that can be displayed on your Web site.
- **Asynchronous operation** – permits your reports to run in the background while your users continue to use their application for other tasks.

Using the Data Report Designer

To create a data r eport, you use the Data Report Designer in Visual Basic. You start by creating a data source and an instance of the Data Report Designer. Typically, you will want to use the Data Environment Designer discussed in Chapter 9 to create the data source.

Getting your data

In order to create a report, you must first add a Data Environment Designer to your application. Then you need to create a `Command` object that will retrieve the data you want to include in your report. The `Command` object can contain child commands to create a hierarchical recordset.

Building a data report's structure

To add a data report to your application, choose Project ÿ Add Data Report. Then the Data Report Window will appear on your screen (see Figure 10-1). This window represents a virtual view of your report. It is broken into a series of sections that contains controls that will display the data from your data source. Note that I included each of the section types in Figure 10-1. The group header and footer sections must be manually added to the report.

- **Report header** – is printed only once at the very beginning of the report.
- **Page header** – is printed at the top of each page of the report. This section is generally used for things like page title.
- **Group header** – is printed at the beginning of a new group of data. A group section corresponds to a `Command` object in a hierarchical recordset, or may include the `RptFunction` control to summarize data from a column in another `Command` object.
- **Detail** – is printed for each row that is retrieved from the lowest level recordset in the hierarchical recordset.

- **Group footer** – is printed at the end of a group of data. It corresponds to the group header section.
- **Page footer** – is printed at the bottom of each page of the report.
- **Report footer** – is printed only once at the end of the report.

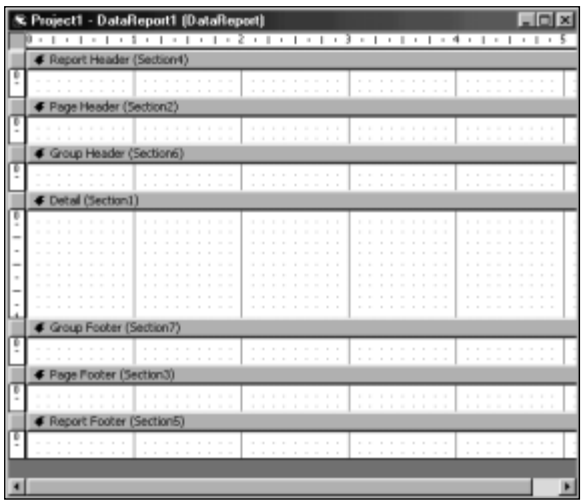


Figure 10-1: The Data Report Window

Growing groups: By default, no groups are included in your report. If you right click on the report and choose Insert Group Header/Footer, you can add a new group to the report.

Binding the data report to the data source

The very first thing you should do after creating a data report is to bind the data report to your data source. You do this by setting the values for the `DataSource` and `DataMember` properties in the normal Visual Basic Properties window. The `DataSource` property should be bound to the `DataEnvironment` object and the `DataMember` property should be bound to the `Command` object that retrieves the information you want to display in the report. All of the controls you use in your report will automatically use these values to retrieve the data for the report.

Grouping data: You should use a hierarchical `Command` object if you wish to define groups in your reports. Each level above the lowest level in the hierarchical command will correspond to a group section in the report.

Adjusting the structure to fit your data

Once you define the `DataSource` for your report, you can right click on the report and choose Retrieve Structure from the pop-up menu. This will add group sections to your report to accommodate the hierarchical recordset from your data source. Note that retrieving the structure will erase any controls you have on the report. You don't need to use the Retrieve Structure function when you're using a regular recordset, because the default report doesn't include any data.

You can also manually add or delete sections to your report by right clicking on the report and selecting or deselecting the section from the pop-up menu. This is useful when you don't want to use a section, such as the Report Header and Footer.

Where did that space come from?: If you only need a header or footer section but not both, you should set the height of the one you don't want to zero. This means that it won't take up space in your report.

You can use the Insert Group Header/Footer on the pop-up menu to add a group section to your report. If you already have a group on your report, a dialog box will be displayed, asking you where the new groups should be placed (see Figure 10-2). Simply press the up or down arrows to move the new group up or down in the group hierarchy. If the group can't be moved up or down in the hierarchy, the appropriate arrow will be disabled.

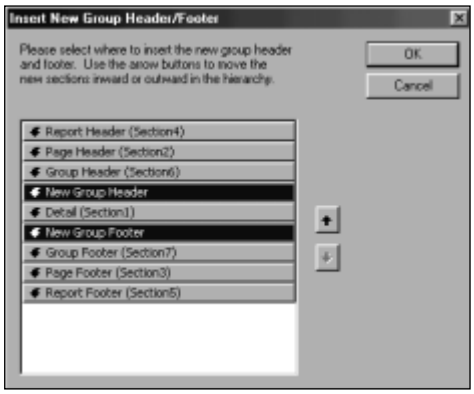


Figure 10-2: Placing a new group in your report

Placing controls on your report

There are two main ways to add controls to your report. The easiest method is to add them automatically by dragging `Command` objects and/or fields from the Data Environment Designer onto the report in the appropriate section and then move

them to where you would like the information to appear. The other way is to add controls manually to your report from the Toolbox and then bind them to the data source.

Automatically adding controls

Once you have bound your report to a `Command` object in the Data Environment Designer, you can drag the object over to a section in your report and the designer will automatically add all of the fields in the section. If you had the Drag and Drop Field Captions check box marked in the Data Environment Designer Options window, labels for each field will also be created. The controls will be arranged in a single column, and the height of the section will automatically be adjusted to accommodate the controls if there isn't sufficient space (see Figure 10-3).

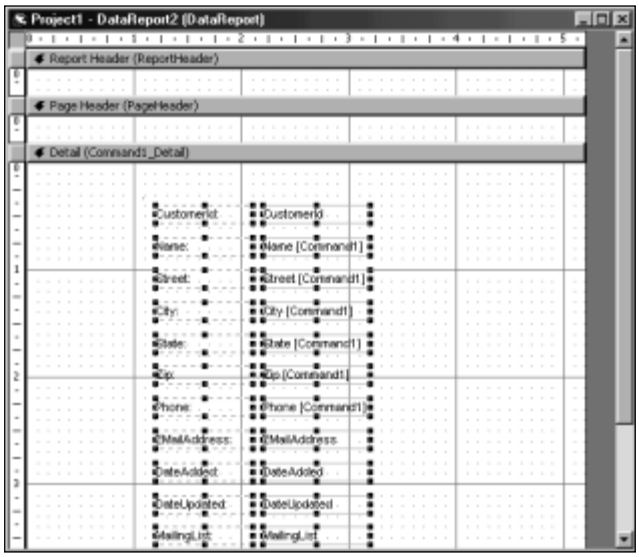


Figure 10-3: Dropping controls on a report

Once the controls are on the form, you will probably want to rearrange them to better use the space available. Since most reports tend to be columnar-based, you may want to try to arrange the controls to fit into a single line, or into a format that doesn't waste as much space. You can drag the columns around the screen. If you want to change their size, simply move the mouse pointer over the appropriate handle and press the left mouse button. Then you can just drag the mouse pointer around until the field is the size you want it to be.

Creating column headers: By default, dragging a field from the `Command` object will also drag along a `RptLabel` field with the name of the field. The `RptLabel` control is similar to a `Label` control in that it will always display a constant text value. In many cases, you will want to move the `RptLabel` control from the section in which you dropped the controls to a section higher in the report hierarchy. For instance, you might want to move the labels from the Detail section to the Page Header section or to a group section. See the next section, "Manually Adding Controls," for an explanation of the `RptLabel` control.

If you have a hierarchical recordset, you must move the `Command` objects one at a time into the appropriate sections. The designer won't let you move a `Command` object containing child commands to a Details section. It must be moved to either a group header or footer section.

You may also drag information from the Data Environment Designer one field at a time onto the report. I find this approach more useful than dragging all of the fields over at once, because it is easier to position each control where I want, and I don't have all of the other controls cluttering the work area.

Manually adding controls

In addition to dragging information from the Data Environment Designer, you can also manually add controls from the standard Visual Basic Toolbox. These controls are kept in a new section on the toolbox known as the DataReport section (see Figure 10-4). The controls available in the toolbox include:

- **RptLabel** – displays a label on your report with a fixed block of text.
- **RptTextBox** – displays a value from your database.
- **RptImage** – displays an image on your report. Note that this field can't be bound to a field in your database.
- **RptLine** – displays a line on your report. This is useful when you want to separate one area of your report from another.
- **RptShape** – displays a shape (rectangle, square, circle, or oval) on your report. This can be used to draw attention to a particular area on your report.
- **RptFunction** – computes the sum, average, minimum value, or maximum value of a column in your database. This control must be used only in a group footer or a report footer section.

These controls are discussed in their respective sections toward the end of this chapter.

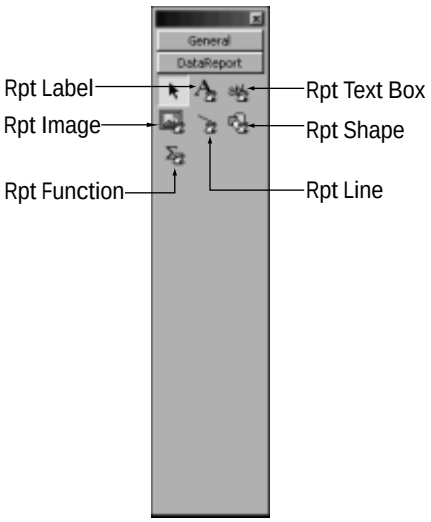


Figure 10-4: Viewing the DataReport toolbox

Setting control properties

Each of the Data Report controls has properties just like traditional controls. The properties for each control are displayed in the Properties window, just as they would be for any other control. You should review the properties of any control you add manually to make sure that their values are appropriate for how you plan to use them.

They're different too: While the names of the controls in the Data Report are similar to the names of the controls you use in a Visual Basic form, they are significantly different. The Data Report controls are often missing various properties or have slightly different definitions for others. For instance, while the `TextBox` control has a `Text` property, the `RptTextBox` control doesn't. Don't assume that the controls are the same. Refer to "The Data Report Object Model" later in this chapter for all of the details about each control.

Binding relationships: The `RptTextBox` and `RptFunction` controls are useless unless they are bound to a data source. Since the `DataReport` object itself is already bound to a data source, you need to bind the `RptTextBox` and `RptFunction` control's `DataField` property to the appropriate field from your data source. Also, you need to select the function that the `RptFunction` control will perform on the bound data.

Adding shortcut controls

In addition to the controls listed above, the Data Report Designer includes some useful shortcuts to predefined `RptLabel` controls. These shortcuts are not available in the toolbox, but can be accessed by moving your mouse pointer over the section of the report you wish to change and right clicking to display a popup menu. Then you can choose the shortcut you want. The following shortcuts may be available:

- **Current Page Number** – displays the current page number (%p).
- **Total Number of Pages** – displays the total number of pages in the report (%P).
- **Current Date (Short Format)** – displays the current date using the Windows short date format (%d).
- **Current Date (Long Format)** – displays the current date using the Windows long date format (%D).
- **Current Time (Short Format)** – displays the current time using the Windows short time format (%t).
- **Current Time (Long Format)** – displays the current time using the Windows long time format (%T).
- **Report Title** – displays the value in the `DataReport`'s `Title` property (%i).

Build your own: You can also add any of the format codes (%p, %P, and so on) listed above to your own `RptLabel` control.

Percent complete: In order to display a percent sign (%) in a `RptLabel` control, you need to enter it twice (%%).

Programming Your Report

Most of the work of building a report is done using the Data Report Designer. Very little code is required to use this facility.

Previewing a report

Displaying a report in a preview window requires nothing more than a call to the `Show` method, as shown below:

```
DataReport1.Show
```

This call is easily inserted as a menu item in your program or in the `Click` event of a command button. Running the `Show` method displays the first page of the report in the preview window, as shown in Figure 10-5. In addition to the scrollbars, users can press buttons to print the report, export the report to a disk file, control the zoom level, or select which page they want to view.

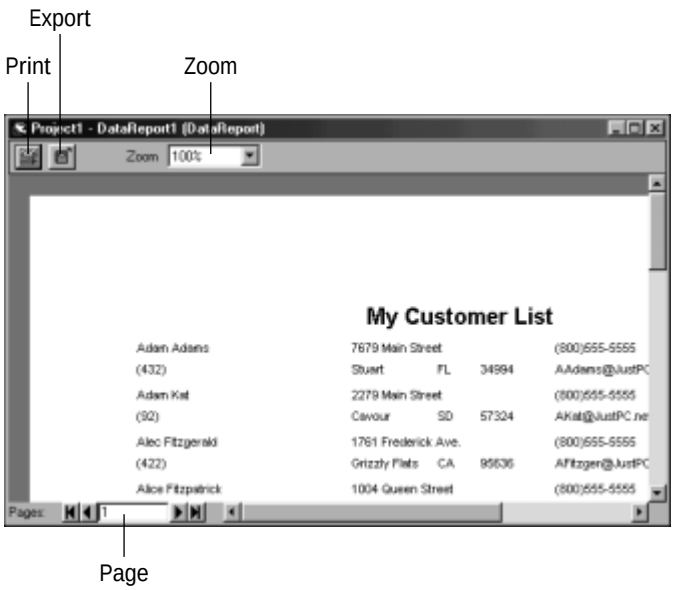


Figure 10-5: Showing the preview of your report

Window sizing: While this may seem obvious, the default size of the Data Report Preview window is the same size as the window used by the Data Report Designer. Also, remember to allow for the width of the margins for your printer when choosing the initial size of the Print Preview window.

Printing a Report

While the user can press the Print button on the Preview window to send the report to the printer, you can do the same thing in your own code by using the `PrintReport` method. The following code will show the Print dialog box displayed in Figure 10-6. If you don't display the Print dialog box (in other words, if the `ShowDialog` parameter is set to `False`), the report will be sent directly to the printer.

```
Cookie = DataReport1.PrintReport (True, rptRangeAllPages)
```

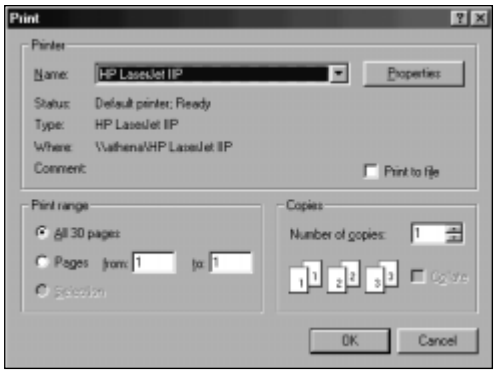


Figure 10-6: Showing the Print dialog box to the user

The value returned by the `PrintReport` method is known as a `Cookie`. It allows you to track this particular report task using the various events that occur while the report is being processed. This can be important, since it is possible to run several reports concurrently. Without the `Cookie` value, you wouldn't be able to identify which report triggered an event.

You can also specify which pages you want to print when you use the `PrintReport` method. If you display the Print dialog box by setting the `ShowDialog` parameter to `True`, the user can override any of the default parameters for the printer and any other values you specified as parameters.

Exporting reports

In addition to sending your reports to the printer, you can also export them to a disk file. You can create either a regular text file or an HTML file. Both types of files can be created with regular characters or with Unicode characters.

You can export a report by calling the `ExportReport` method, using code like this:

```
Cookie = DataReport1.ExportReport (rptKeyHTML, _
         "d:\vb6db\Chapter10\MSReport\report1.HTM", _
         True, False, rptRangeFromTo, 1, 3)
```

This will create a file called `report1.HTM` without showing the Export dialog box. If the file exists, it will be overwritten and only pages one through three will be exported. To understand the meaning of the various parameters in the ExportReport method, see the section on `DataReport` methods later in this chapter.

Tracking asynchronous activity

Recall that *asynchronous operation* permits your reports to run in the background while your users continue to use their application for other tasks. Also keep in mind that the Data Report facility operates independently of your application program. This means that your user can continue to work with your application while the report is being generated and printed. You can track the progress of the report in your application by using the `AsyncProgress` event, as shown in Listing 10-1. Note that one of the values passed to the event is the same `Cookie` value that was returned by the method that created the report or export task.

Listing 10-1: The DataReport_AsyncProgress event

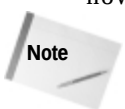
```
Private Sub DataReport_AsyncProgress _
   (ByVal JobType As MSDataReportLib.AsyncTypeConstants, _
    ByVal Cookie As Long, ByVal PageCompleted As Long, _
    ByVal TotalPages As Long)

Form1.ProgressBar1.Min = 0
Form1.ProgressBar1.Value = PageCompleted
Form1.ProgressBar1.Max = TotalPages

End Sub
```

The Data Report object model

The Data Report Designer is based around an object called `DataReport`. This object contains a number of different properties, methods, and events that control how your report is generated.



Forming an object: The `DataReport` object is based on the `Form` object and it inherits all of the same properties, methods, and events.

DataReport properties

The `DataReport` object contains many different properties. I've listed the key ones for this object in Table 10-1.

Table 10-1
Properties of the DataReport Object

Key Properties of the DataReport ObjectProperty	*Description*
`AsyncCount`	A `Long` value containing the number of asynchronous operations still executing.
`BottomMargin`	A `Long` value containing the size of the bottom margin in twips.
`Caption`	A `String` value containing the value that will be displayed in the DataReport's title bar.
`DataMember`	A `Variant` value containing the data member of the `DataSource` that will be used to generate the report.
`DataSource`	A `Variant` containing the data source for the report.
`ExportFormats`	An object reference to an `ExportFormats` collection that contains the type of export formats available.
`Font`	An object reference to a `Font` object that will be used as the default font for the report.
`LeftMargin`	A `Long` value containing the size of the left margin in twips.
`ReportWidth`	A `Long` value containing the total width of the report in twips.
`RightMargin`	A `Long` value containing the size of the right margin in twips.
`Sections`	An object reference to a `Sections` collection containing information about the various sections of the report.
`Title`	A `String` value containing the title of the report.
`TopMargin`	A `Long` value containing the size of the top margin in twips.

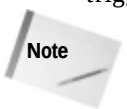
Twips are for kids: Visual Basic uses the *twip* as the fundamental unit of measurement when placing controls on a form or report. A twip is 1/20 of a point or 1/1440 of an inch. A pixel is the smallest element that can be displayed on a screen or a printer. Pixels depend on the characteristics of the device they are being displayed on, while a twip is independent of any physical device.

DataReport methods

The DataReport object has a number of methods. Listed below are the ones that are the most important when using Data Report in your application.

Function ExportReport ([FormatIndexOrKey], [FileName], [Overwrite As Boolean = True], [ShowDialog As Boolean = False], [Range As PageRangeConstants], [PageFrom], [PageTo]) As Long

The ExportReport method writes the text part of a report to a disk file. The return value is known as a Cookie, which is used to identify the specific data report process. This value will be used in various events to allow you to identify which report triggered the event.



Not saved: Any images or shapes that you include in your printed report will not be saved in your exported report.

FormatIndexOrKey is a Variant containing either a reference for an ExportFormat object or a String that is one of the following: rptKeyHTML ("key_def_HTML"), export the report in standard HTML format; rptKeyText ("key_def_Text"), export the report as a regular text file; or rptKeyUnicodeHTML_UTF8 ("key_def_UnicodeHTML_UTF8). If this value is not specified, the Export dialog box will be displayed.

FileName is a String containing the name of the file to be displayed. If this value isn't specified, the Export dialog box will be shown.

Overwrite is a Boolean value when TRUE means that if the file specified by FileName exists, it will automatically be overwritten.

ShowDialog is a Boolean value when TRUE means that the Export dialog box will be displayed.

Range is an enumerated data type. A value of rptRangeAllPages (0) will print all of the pages in the report (default), while a value of rptRangeFromTo (1) will print only the pages specified in the PageFrom and PageTo parameters.PageFrom is a Long value containing the starting page number used when Range is set to rptRangeFromTo (1).

PageTo is a Long value containing the ending page number used when Range is set to rptRangeFromTo (1).

Function PrintReport ([ShowDialog As Boolean = False], [Range As PageRangeConstants], [PageFrom], [PageTo]) As Long

The `PrintReport` method is used to send a copy of the report to the default system printer. The return value is known as a `Cookie`, which is used to identify the specific data report process. This value will be used in various events to allow you to identify which report triggered the event.

`ShowDialog` is a `Boolean` value when `TRUE` means that the Print dialog box will be displayed.

`Range` is an enumerated data type. A value of `rptRangeAllPages` (0) will print all of the pages in the report (default), while a value of `rptRangeFromTo` (1) will print only the pages specified in the `PageFrom` and `PageTo` parameters.`PageFrom` is a `Long` value containing the starting page number used when `Range` is set to `rptRangeFromTo` (1).

`PageTo` is a `Long` value containing the ending page number used when `Range` is set to `rptRangeFromTo` (1).

Sub Refresh ()

The `Refresh` method is used to reprocess the report and repaint the display.

Sub Show ([Modal], [OwnerForm])

The `Show` method displays the Data Report form as the report's Print Preview window.

`Modal` is an `Integer` that determines how the Preview window is displayed. A value of `vbModal` (1) will freeze all activity in the application until the Preview window is closed, while `vbNormal` (0) allows the application to continue running while the Preview window is displayed. If not specified, `vbNormal` is assumed.

`OwnerForm` is a `String` containing the name of the owner form. The default value for this argument is `"Me"`.

Key DataReport events

The `DataReport` object contains a few events that allow you to monitor its progress as your report is being processed.

Event AsynchProgress (JobType as AsynchTypeConstants, Cookie As Long, PageCompleted As Long, TotalPages As Long)

The `AsynchProgress` event is called when the Data Report process has finished generating a page of the report.

Where am I?: You may want to use this event to update a progress bar that indicates how much of the report has been processed.

`JobType` is an enumerated data type whose value is `rptAsyncPreview` (0) when a preview operation is being processed, `rptAsyncPrint` (1) when a print report operation is being processed, or `rptAsyncReport` (2) when an export operation is being processed.

`Cookie` is a `Long` value that identifies the specific report job.

`PageCompleted` is a `Variant` containing the number of pages that have been completed.

`TotalPages` is a `Variant` containing the total number of pages to be processed.

Event Error (JobType as AsynchTypeConstants, Cookie As Long, ErrObj As RptError, ShowError As Boolean)

The `Error` event is called when the Data Report program encounters an error.

`JobType` is an enumerated data type whose value is `rptAsyncPreview` (0) when a preview operation is being processed, `rptAsyncPrint` (1) when a print report operation is being processed, or `rptAsyncReport` (2) when an export operation is being processed.

`Cookie` is a `Long` that identifies the specific report job.

`ErrObj` is an `RptError` object containing the details of the error that triggered this event.

`ShowError` is a `Boolean`, which when set to `TRUE` will display the error information to the user in a dialog box.

Event ProcessingTimeout (Seconds As Long, Cancel As Boolean, JobType as AsynchTypeConstants, Cookie As Long)

The `ProcessingTimeout` event is fired periodically while the report is being generated. You can examine the amount of time that the job has been running and choose whether or not to cancel the report process.

`Seconds` is a `Long` value containing the number of seconds since the report process was started.

`Cancel` is a `Boolean`, which when set to `TRUE` will cancel the report process.

`JobType` is an enumerated data type whose value is `rptAsyncPreview` (0) when a preview operation is being processed, `rptAsyncPrint` (1) when a print report operation is being processed, or `rptAsyncReport` (2) when an export operation is being processed.

`Cookie` is a `Long` value that identifies the specific report job.

Controls collection properties

The `Controls` collection is an object that contains object references to the controls stored in a particular Section object. Table 10-2 lists the properties of the `Controls` collection. Note that controls can't be added or deleted at runtime.

Table 10-2
Properties of the Controls Collection

Property	*Description*
`Count`	A `Long` value containing the number of controls in the collection.
`Item`	An object reference to one of the control objects (`RptFunction`, `RptImage`, `RptLabel`, `RptLine`, `RptShape`, and `RptTextBox`).

ExportFormat object properties

The `ExportFormat` object contains information about a single export format. Table 10-3 lists the various properties associated with the `ExportFormat` object.

ExportFormats collection properties

The `ExportFormats` object contains information about a single export format. Table 10-4 lists the various properties associated with the `ExportFormat` object.

Table 10-3
Properties of the ExportFormat Object

Property	*Description*
`FileFilter`	A `String` value containing the list of file extensions that will be used in the file filter. If more than one file extension is used, they must be separated by semicolons (;). For example: `"*.HTM"` is a valid value.
`FileFormatString`	A `String` value containing the text value displayed in the Export dialog box's Save As type box that corresponds to the list extensions listed in `FileFilter`. For example: `"Web Report (*.HTM)"` is a valid value for this property.
`FormatType`	An enumerated data type containing the type of export format. Legal values are: `rptFmtHTML` (0), export the report using the regular HTML format; `rptFmtText` (1), export the report as a text file; `rptFmtUnicodeText` (2), export the report as a Unicode text file; `rptFmtUnicodeHTML_UTF8` (3), export the report using an HTML format with the UTF – 8 Unicode character set.
`Key`	A `String` that uniquely identifies the export format.
`String`	A `String` containing the template that will be used to export the report.

Table 10-4
Properties of the ExportFormat Object

Property	*Description*
`Count`	A `Long` value containing the number of objects in the collection.
`Item (key)`	An object reference to an `ExportFormat` object containing information about a particular export format, where `key` contains a reference to the `ExportFormat` object.

ExportFormats collection methods

The `ExportFormats` collection has no default methods. However, each time you add a command in the Data Environment Designer, a shortcut method will be added to the `DataEnvironment` object. This makes it easier to code your application.

Function Add (Key As String, FormatType As ExportFormatTypeConstants, FileFormatString As String, FileFilter As String, [Template]) As ExportFormat

The `Add` method creates a new `ExportFormat` object and inserts it into the `ExportFormats` collection.

`Key` is the key value that will be used to locate the new `ExportFormat` object.

`FormatType` is an enumerated data type describing the type of format stored in the object: `rptFmtHTML` (0), export the report using the regular HTML format; `rptFmtText` (1), export the report as a text file; `rptFmtUnicodeText` (2), export the report as a Unicode text file; `rptFmtUnicodeHTML_UTF8` (3), export the report using an HTML format with the UTF – 8 Unicode character set. If this value is not specified, the Export dialog box will be displayed.

`FileFormatStrong` is a `String` value containing the text value displayed in the Export dialog box's Save As type box that corresponds to the list extensions listed in `FileFilter`. For example: `"Web Report (*.HTM)"` would be a valid value for this argument.

`FileFilter` is a `String` value containing the list of file extensions that are associated with the export format. If more than one file extension is used, they must be separated by semicolons (;) and the first extension is the default extension. For example: `"*.HTM"` is a valid value.

`Template` is a `String` value containing the template that will be used when someone uses this export format. If not specified, an empty string will be assumed.

Sub Clear ()

The `Clear` method removes all of the members of the collection.

Sub Insert (Format As ExportFormat)

The `Insert` method adds an `ExportFormat` object to the collection. `Format` is an `ExportFormat` object containing information to be added to the collection.

Sub Remove (Key)

The `Remove` method deletes a member identified by `Key` from the collection. `Key` is the index or key value of the `ExportFormat` object to be removed from the collection.

RptError object properties

The RptError object contains error information about an error that occurred while processing a report. Table 10-5 lists the various properties associated with the RptError object.

Table 10-5
Properties of the RptError Object

Property	*Description*
Description	A String value containing a textual description of the error.
ErrorNumber	A Long value containing error number value.
HelpContext	A Long containing the help context for the error, which will contain more information about the error.
HelpFile	A String containing the name of the help file associated with HelpContext.
Source	An object reference to the object that caused the error.

RptFunction control properties

The RptFunction control is used to compute a value based on a series of values retrieved from the database. Table 10-6 lists the various properties associated with the RptFunction object.

Critical properties: DataField should be set to the appropriate field from your data source and FunctionType should be set to the calculation you would like to apply to the data prior to running a report.

Table 10-6
Properties of the RptFunction Control

Property	*Description*
Alignment	An enumerated value describing how the contents of the control will be displayed. Legal values are: rptJustifyLeft (0), rptJustifyRight (1), and rptJustifyCenter (2).
BackColor	A Long value containing the color value for the background of the control.

Continued

Table 10-6 *(continued)*

Property	*Description*
BackStyle	An enumerated data type containing the style of the background. Legal values are: rptTransparent (0) and rptOpaque (1).
BorderColor	A Long value containing the color value for the border surrounding the control.
BorderStyle	An enumerated data type describing the style of the border surrounding the control. Legal values are: rptBSTransparent (0), rptBSSolid (1), rptBSDashes (2), rptBSDots (3), rptBSDashDot (4), rptBSDashDotDot (5).
CanGrow	A Boolean value, when TRUE means that the height of the control will automatically be increased if the size of the text exceeds the default height of the control.
DataField	A String value containing the name of the database field which the function will be applied to.
DataFormat	An object reference to a StdDataFormat object containing the information on how to format the data displayed by the control.
DataMember	A String value containing the name of the data member of the data source that will be used as the source of data for the control.
Font	An object reference to a Font object that describes the font information used to display the value of the control.
ForeColor	A Long containing the color that the text will be displayed in.
FunctionType	An Integer value identifying the function used with the field specified by DataField to compute the value that will be displayed by this control. Legal values are: rptFuncSum (0), computes the sum of the values in DataField; rptFuncAve (1), computes the average of the values in DataField; rptFuncMin (2), finds the minimum value of the values in DataField; rptFuncMax (3), finds the maximum value of the values in DataField; rptFuncRCbt (4), counts the number of rows retrieved in this section; rptFuncVCnt (5), counts the number of rows retrieved in this section whose values are not **Null**; rptFuncSDEV (6), computes the standard deviation of the values in DataField; rptFuncSERR (7), computes the standard error of the values in DataField.
Height	A Long containing the height of the control in twips.
Left	A Long containing the distance between the left edge of the report and the left edge of the control.

Property	Description
Name	A String containing the name of the control.
RightToLeft	A Boolean, when TRUE means that the text will be displayed right to left in a bi-directional version of Windows.
Top	A Long containing the distance between the top edge of the section and the top edge of the control.
Visible	A Boolean, when TRUE means that the control will be displayed at runtime.
Width	A Long containing the width of the control in twips.

RptImage control properties

The RptImage control is used to display a static image on your report. Table 10-7 lists the various properties associated with the RptImage object.

Critical properties: You should set the Picture property to the image you want to see at design time. Then you should adjust the SizeMode and Picture Alignment if needed to make sure that the picture is displayed properly.

Table 10-7
Properties of the RptImage Control

Property	Description
BackColor	A Long value containing the color value for the background of the control.
BackStyle	An enumerated data type containing the style of the background. Legal values are: rptTransparent (0) and rptOpaque (1).
BorderColor	A Long value containing the color value for the border surrounding the control.
BorderStyle	An enumerated data type describing the style of the border surrounding the control. Legal values are: rptBSTransparent (0); rptBSSolid (1); rptBSDashes (2); rptBSDots (3); rptBSDashDot (4); rptBSDashDotDot (5).
Height	A Long containing the height of the control in twips.
Left	A Long containing the distance between the left edge of the report and the left edge of the control.

Continued

Table 10-7 *(continued)*

Property	*Description*
`Name`	A `String` containing the name of the control.
`Picture`	An object reference to an `IPictureDisp` object.
`PictureAlignment`	An enumerated data type describing how the picture will be positioned in the control. Legal values are: `rptPATopLeft` (0), position the image at the top left corner; `rptPATop` (1), position the image at the top center edge; `rptPATopRight` (2), position the image at the top right corner; `rptPARight` (3), position the image at the right center corner; `rptPABottomRight` (4), position the image at the bottom right corner; `rptPABottom` (5), position the image at the bottom center edge; `rptPABottomLeft` (6), position the image at the bottom left corner; `rptPALeft` (7), position the image at the left center edge; `rptPACenter` (8), center the image in the control.
`SizeMode`	An enumerated type describing how the image will be displayed in the control. Legal values are: `rptSizeClip` (0), the image will be clipped to fit inside the control; `rptSizeStretch` (1), the image will be stretched to fill the control; `rptSizeZoom` (2), the image will be stretched to fill the control, but the image's height to width proportion will remain unchanged.
`Top`	A `Long` containing the distance between the top edge of the section and the top edge of the control.
`Visible`	A `Boolean`, when `TRUE` means that the control will be displayed at runtime.
`Width`	A `Long` containing the width of the control in twips.

RptLabel control properties

The `RptLabel` control is used to display a constant text value on your report. Table 10-8 lists the various properties associated with the `RptLabel` object.

Critical properties: Enter the text you want to display in `Caption`. Next, set the `Alignment` property to position text properly within the label. If there is too much text for the label, then either resize the control on the Report Designer window or set the `CanGrow` property to `TRUE` to display all of the text.

Table 10-8
Properties of the RptLabel Control

Property	*Description*
`Alignment`	An enumerated value describing how the contents of the control will be displayed. Legal values are: `rptJustifyLeft` (0), `rptJustifyRight` (1), and `rptJustifyCenter` (2).
`BackColor`	A `Long` value containing the color value for the background of the control.
`BackStyle`	An enumerated data type containing the style of the background. Legal values are: `rptTransparent` (0) and `rptOpaque` (1).
`BorderColor`	A `Long` value containing the color value for the border surrounding the control.
`BorderStyle`	An enumerated data type describing the style of the border surrounding the control. Legal values are: `rptBSTransparent` (0), `rptBSSolid` (1), `rptBSDashes` (2), `rptBSDots` (3), `rptBSDashDot` (4), `rptBSDashDotDot` (5).
`CanGrow`	A `Boolean` value, when `TRUE` means that the height of the control will automatically be increased if the size of the text exceeds the default height of the control.
`Caption`	A `String` value containing the text that will be displayed by this control.
`Font`	An object reference to a `Font` object that describes the font information used to display the value of the control.
`ForeColor`	A `Long` containing the color that the text will be displayed in.
`Height`	A `Long` containing the height of the control in twips.
`Left`	A `Long` containing the distance between the left edge of the report and the left edge of the control.
`Name`	A `String` containing the name of the control.
`RightToLeft`	A `Boolean`, when `TRUE` means that the text will be displayed right to left in a bi-directional version of Windows.
`Top`	A `Long` containing the distance between the top edge of the section and the top edge of the control.
`Visible`	A `Boolean`, when `TRUE` means that the control will be displayed at runtime.
`Width`	A `Long` containing the width of the control in twips.

Fortune favors the bold: In most cases, you should display the text in a label in bold to help set apart column headers and other text from the data in the detail section.

RptLine control properties

The `RptLine` control is used to draw a line on your report. You define a rectangle using the `Height` and `Width` properties and the line will be drawn from one corner to the diagonal. Which corners are used depends on the `LineSlant` property. Table 10-9 lists the various properties associated with the `RptLine` control.

Critical properties: Adjust the `LineSlant` to change the direction of the line. Note that a control whose width is only one twip tall will be displayed as a horizontal line without a slant.

Table 10-9
Properties of the RptLine Control

Property	*Description*
`BorderColor`	A `Long` value containing the color value of the line.
`BorderStyle`	An enumerated data type describing how the line will be drawn. Legal values are: `rptBSTransparent` (0), `rptBSSolid` (1), `rptBSDashes` (2), `rptBSDots` (3), `rptBSDashDot` (4), `rptBSDashDotDot` (5).
`Height`	A `Long` containing the height of the control in twips.
`Left`	A `Long` containing the distance between the left edge of the report and the left edge of the control.
`LineSlant`	An enumerated type describing which corners will be used to draw the line. Legal values are: `rptSlantNWSE`, the line will be drawn from the top left to the bottom right; `rptSlantNESW`, the line will be drawn from the top right to the bottom left.
`Name`	A `String` containing the name of the control.
`Top`	A `Long` containing the distance between the top edge of the section and the top edge of the control.
`Visible`	A `Boolean`, when `TRUE` means that the control will be displayed at runtime.
`Width`	A `Long` containing the width of the control in twips.

RptShape control properties

The RptShape control is used to draw a shape on your report. You define a rectangle using the Height and Width properties and then choose the shape that will fill the rectangle using the Shape property. If you choose to draw a circle or a square, then the shape will start in the upper-left corner of the rectangle you drew and its size will be dictated by the width or height, whichever is smaller. Table 10-10 lists the various properties associated with the RptShape control.

Critical properties: Shape should be set to the shape you want to display.

Table 10-10
Properties of the RptShape Control

Property	*Description*
BackColor	A Long value containing the color value for the background of the control.
BackStyle	An enumerated data type containing the style of the background. Legal values are: rptTransparent (0) and rptOpaque (1).
BorderColor	A Long value containing the color value of the line.
BorderStyle	An enumerated data type describing how the shape will be drawn. Legal values are: rptBSTransparent (0), rptBSSolid (1), rptBSDashes (2), rptBSDots (3), rptBSDashDot (4), rptBSDashDotDot (5).
Height	A Long containing the height of the control in twips.
Left	A Long containing the distance between the left edge of the report and the left edge of the control.
Name	A String value containing the name of the control.
Shape	An enumerated data type that identifies the shape that will be drawn on the report. Legal values are: rptShpRectangle (0), rptShpSquare (1), rptShpOval (2), rptCircle (3), rptShpRoundedRectangle (4), rptShpRoundedSquare (5).
Top	A Long containing the distance between the top edge of the section and the top edge of the control.
Visible	A Boolean, when TRUE means that the control will be displayed at runtime.
Width	A Long containing the width of the control in twips.

RptTextBox control properties

The RptTextBox control is used to display a value from the database on your report. Table 10-11 lists the various properties associated with the RptTextBox object.

Critical properties: The DataField property should be set to the field from your data source that will be displayed in the text box.

Table 10-11
Properties of the RptLabel Control

Property	*Description*
Alignment	An enumerated value describing how the contents of the control will be displayed. Legal values are: rptJustifyLeft (0), rptJustifyRight (1), and rptJustifyCenter (2).
BackColor	A Long value containing the color value for the background of the control.
BackStyle	An enumerated data type containing the style of the background. Legal values are: rptTransparent (0) and rptOpaque (1).
BorderColor	A Long value containing the color value for the border surrounding the control.
BorderStyle	An enumerated data type describing the style of the border surrounding the control. Legal values are: rptBSTransparent (0), rptBSSolid (1), rptBSDashes (2), rptBSDots (3), rptBSDashDot (4), rptBSDashDotDot (5).
CanGrow	A Boolean value, when TRUE means that the height of the control will automatically be increased if the size of the text exceeds the default height of the control.
Font	An object reference to a Font object that describes the font information used to display the value of the control.
ForeColor	A Long containing the color that the text will be displayed in.
Height	A Long containing the height of the control in twips.
Left	A Long containing the distance between the left edge of the report and the left edge of the control.
Name	A String value containing the name of the control.
RightToLeft	A Boolean value, when TRUE, means that the text will be displayed right to left in a bi-directional version of Windows.

Property	Description
Top	A Long containing the distance between the top edge of the section and the top edge of the control.
Visible	A Boolean, when TRUE means that the control will be displayed at runtime.
Width	A Long containing the width of the control in twips.

Section object properties

The Section object contains information about how a section is displayed on the report. See Table 10-12 for the properties available in this object.

Table 10-12
Properties of the Section Object

Property	Description
Controls	An object reference to a Controls collection containing information about the set of controls displayed in the section.
ForcePageBreak	An enumerated type describing how page breaks will be taken for the section. Values are: rptPageBreakNone (0), no page breaks are taken; rptPageBreakBefore (1), the page break occurs before the section; rptPageBreakAfter (2), the page break occurs after the section; rptPageBreakBeforeAndAfter (3), the page break occurs before and after the section.
Height	A Long value containing the height of the section in twips.
KeepTogether	A Boolean value, which when set to TRUE forces the entire section to fit on the same page. If there isn't sufficient room, a new page will be started.
Name	A String value containing the name of the section.
Visible	A Boolean, which when set to TRUE means the report section is visible at runtime.

Sections collection properties

The Sections collection is an object that contains object references to the sections stored in your report. Table 10-13 lists the properties of the Sections collection. Note that sections can't be added or deleted at runtime.

Table 10-13
Properties of the Controls Collection

Property	*Description*
`Count`	A `Long` value containing the number of `Section` objects in the collection.
`Item`	An object reference to the `Section` object.

Thoughts on the Data Report Feature

A tool like Microsoft Data Report is important to many application developers. With the emphasis today on building graphical applications that can provide information on demand that eliminates the need for paper reports, programmers often forget that sometimes having a paper report is critical.

Some types of reports deserve to be archived on paper, even if they are never looked at. For example, consider an audit log. Having a paper audit trail may be the only way to prove that someone unauthorized changed information in your database. Also, for some high profile applications, having a paper report available may allow the user to continue operating in the event of a catastrophic system failure. Failure to plan for this type of situation is the sign of a poor job analyzing the user's true needs.

Of course, there are many cases where actual reports are critical. Printing receipts from a point of sale system, as well as printing packing lists for the warehouse and invoices for accounts receivable, are all cases where a well-designed paper report is important. While it is possible to do this by using the `Print` object in Visual Basic and manually building the code to do the job, using Data Report is a much more efficient use of a programmer's time.

While I'm not a big fan of the Data Environment Designer for designing forms, I've found that the Data Report Designer works best when you use the Data Environment Designer as a data source. This is especially true if you want to use hierarchical recordsets as input. This is because I find the drag and drop report design using the field objects from the Data Environment Designer much easier than manually binding the controls after I've created them.

Summary

In this chapter you learned the following:

- ✦ You can use the Data Report Designer to design a report graphically just like you design a form for a Visual Basic program.
- ✦ You can extract the information for the report using the Data Environment Designer.
- ✦ You can define various sections on the report to help you summarize your data.
- ✦ You can place fields on the report by dragging them from the Data Environment Designer.
- ✦ You can allow the user to preview the report before sending it to the printer.
- ✦ You can also export your report to a text file or an HTML file.

✦ ✦ ✦