Accessing Database Information Using Visual Basic:

Data Access Objects and Remote Data Objects

Amanda Reynolds
Y398--Internship
June 23, 1999

Abstract

Data Access Objects

Declaring a Data Bound Control

Connecting a Data Bound Control with a Database

Manipulating Data Records with a Data Bound Control

Remote Data Objects

Description of Remote Data Objects

The Remote Data Object Programming Model
Declaring a Remote Data Object

Connecting a Remote Data Object with a Database

Manipulating Data Records with a Remote Data Object

Conclusion

Abstract

Much of the programming done today requires data access. With large amounts of data being maintained in databases, access mechanisms that allow manipulation of this data are vital. Visual Basic is a popular programming package because of its user-friendly interface and data access features. Two popular data access mechanisms are data bound controls with data access objects, and remote data controls with remote data objects.

The main difference between these methods are that data bound controls are usually used to connect to desktop databases, and remote data objects are used to connect to client/server data (for client database manipulation of a remote database).

Data Access Objects

Declaring a Data Bound Control

A data bound control connects a data control in Visual Basic to a database table or query. Data controls are visual objects that are said to be data-aware. Data controls may include check boxes, images, labels, picture boxes, and text boxes (see figure A). Visual Basic's data controls allow users to access stored database records. The data control establishes a link between the database and other controls in the interface, in a process called binding. When a control is bound, it displays database field contents when the Data control is present.
Required property settings for the Data control include the `DatabaseName` property which specifies the complete path to the database (see figure B), and the `RecordSource` property, which indicates the table to use from the database (see figure C). The `DataSource` property (specifies the name of the Data control to which it is bound) and the `DataField` property (specifies the name of a field in the database to which the control is linked) should be set for all form controls that access database information (Cashman, VB 4).

**Connecting a Data Bound Control with a Database**

The properties list for a data control (see figures B and C) includes a "DatabaseName" to specify the database the control is to connect to and a "RecordSource" to specify the table in the database, or the SQL statement, for the data. SQL stands for Structured Query Language—a high-level language used to manipulate and/or define databases.

Data controls rely on Data Access Objects (DAO) to perform processing. The DAO provide a more comprehensive interface to data using Visual Basic—they allow the user to run queries, update database table values and create the structure of databases (McManus, Database Access with Visual Basic 6.0).

In general, four parameters must be defined for data access:

1. Database Name--`(DatabaseName)` This is the full path to the database to which you would like to connect.
2. Table Name--`(RecordSource)` This is the name of the table in 1) to which you would like to connect.
3. Field Name--(DataField) This is the field to which your control should be connected.
4. Data Source--(DataSource) This is the name of the data control to be specified in linked components (such as text boxes). This is necessary because it is possible to have two data controls on one form.

**Manipulating Data Records with a Data Bound Control**

Users may add records to the database with data controls in two ways:

- Using the **EOFAction** property, which indicates the action to take when the user tries to move past the last record in a table. Parameters for the EOFAction property are: 0 for Move Last (move the last record in the database), 1 for EOF (default), or 2 for Add New (adds a new record to the database).
- Using "AddNew" with the connected data control's record set.

(ex. DataCustomer.Recordset.AddNew )

Records may be updated using "Update" with the connected data control's record set in the same way "AddNew" is used.

Data binding features may be set for a control in the Procedure Attributes dialog box (see figure D). The Procedure Attributes box is active after a procedure has been added. A procedure may be added to a module by choosing Add Procedure from the Tools menu. After adding the procedure, the full procedure attributes box is available under Tools, Procedure Attributes, then Advanced. The "Name" property must be set to the name of the data control being bound, and the "Property is Data Bound" checkbox must be checked.
The checkbox labeled "This property binds to DataField" specifies that the property should appear alone under the DataField Property which is used when a control has a single bound property. The checkbox labeled "Show in DataBindings collection at design time" determines whether the bound property appears in the Properties window under the DataBindings property. The checkbox labeled "Property will call CanPropertyChange before changing" specifies that the control will check with the container to see if the property can be changed (Cashman, VB 4).

Remote Data Objects

Description of Remote Data Objects

Remote Data Objects (RDO) exist as a thin layer over Open Database Connectivity (ODBC)--this technology allows a database client application to connect to a remote database. The application using ODBC does not need to know the specific type of relational database it is connecting to, it simply uses uniform SQL statements to manipulate the data of the remote database (see figure E). RDOs are sets of programmable objects that provide properties, events and methods access to ODBC data (see figure F). The Remote Data Control is similar to the Data Control. It can be dropped on a form and provides a reduced set of properties, events and methods for accessing ODBC data (Mezick, 161).
The Remote Data Object programming model is similar to the Data Access Objects (DAO) programming model in many respects. However, far more emphasis is focused on handling stored procedures and their result sets, and less emphasis is placed on data access retrieval methods used solely by ISAM programming models. The following table (figure G) describes each of the objects.

**The Remote Data Object Programming Model**

The Remote Data Object programming model is similar to the Data Access Objects (DAO) programming model in many respects. However, far more emphasis is focused on handling stored procedures and their result sets, and less emphasis is placed on data access retrieval methods used solely by ISAM programming models. The following table describes each of the objects.
<table>
<thead>
<tr>
<th>RDO object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RdoEngine</td>
<td>The base object. Created automatically when you first access RDO in your application.</td>
</tr>
<tr>
<td>rdoError</td>
<td>Used to handle all ODBC errors and messages generated by RDO. Created automatically.</td>
</tr>
<tr>
<td>RdoEnvironment</td>
<td>Defines a logical set of connections and transaction scope for a particular user name. Contains both open and allocated (but unopened) connections, provides mechanisms for simultaneous transactions, and provides a security context for data manipulation language (DML) operations on the database. rdoEnvironments(0) created automatically.</td>
</tr>
<tr>
<td>RdoConnection</td>
<td>Represents an open connection to a remote data source and a specific database on that data source, or an allocated but as yet unconnected object, which can be used to subsequently establish a connection.</td>
</tr>
<tr>
<td>RdoTable</td>
<td>Represents the stored definition of a base table or an SQL view.</td>
</tr>
<tr>
<td>RdoResultset</td>
<td>Represents the rows that result from running a query.</td>
</tr>
<tr>
<td>RdoColumn</td>
<td>Represents a column of data with a common data type and a common set of properties.</td>
</tr>
<tr>
<td>RdoQuery</td>
<td>An SQL query definition that can include zero or more parameters.</td>
</tr>
<tr>
<td>RdoParameter</td>
<td>Represents a parameter associated with an rdoQuery object. Query parameters can be input, output, or both.</td>
</tr>
</tbody>
</table>

Figure G: RDO Object Model Description

**Declaring a Remote Data Object**

To use an RDO, one must first reference the RDO object model using the Reference dialog (see figure H). The Microsoft Remote Data Object 2.0 box should be checked.
Connecting a Remote Data Object with a Database

Next, a variable must be declared as an rdoConnection. The connect string and cursor driver must be included in the connection process. The object must then be instantiated and methods and properties may be called on the connection. Queries may be assigned to recordsets which may be opened with OpenResultSet (see Figure I).
Rows may be added, manipulated, or deleted from an existing record set using AddNew, Edit, Update, and Delete methods. The AddNew method allows you to add an initialized row to a record set that you may edit and apply to the record set and the table (ex. CustRecSet.AddNew adds a new record space to the customer record set which is open for editing). The Edit method allows you to
open an existing row in your record set for editing (ex. CustRecSet.Edit allows you to edit information in the record that is currently active in the customer record set). After you have entered information into a new row or edited an existing row, you must use the Update method on the record set to commit your work to the appropriate table (ex. CustRecSet.Update saves any changes or additions you have made to the current customer record). If you do not wish to commit the changes you have made, you can use the CancelUpdate method to change the EditMode of the record set to non-editing (see Figure J) (ex. CustRecSet.CancelUpdate erases the buffer holding the additions or changes to the current customer record). The Delete method removes the current row open for editing (ex. CustRecSet.Edit <newline> CustRecSet.Delete removes the current record from the database).

Figure J: EditMode

### Return Values

The EditMode property returns an integer or constant as described in the following table:

<table>
<thead>
<tr>
<th>Constant</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdEditNone</td>
<td>0</td>
<td>No editing operation is in progress.</td>
</tr>
<tr>
<td>rdEditInProgress</td>
<td>1</td>
<td>The Edit method has been invoked, and the current row is in the copy buffer.</td>
</tr>
<tr>
<td>rdEditAdd</td>
<td>2</td>
<td>The AddNew method has been invoked, and the current row in the copy buffer is a new row that hasn't been saved in the database.</td>
</tr>
</tbody>
</table>

### Remarks

The EditMode property is most useful when you want to depart from the default functionality of a RemoteData control. You can check the value of the EditMode property and the value of the action parameter in the Validate event procedure to determine whether to invoke the UpdateRow method.

### Conclusion

This paper gives a brief view of the differences of data bound controls and remote data objects. Data bound controls are a fast and simple way to retrieve table information. While remote data control definition is more involved (code must be written to manipulate or add records) than that of data bound controls, RDOs make it easier to retrieve queried information on one or more tables and they are superior to DAOs for client/server applications where a remote database must be accessed.

### WORKS CITED


McManus, Jeffrey P. *Database Access with Visual Basic 6*. Sams, IN, 1999

All screen captures are from MSDN Library Visual Studio 6.0, Subject: RDO