Chapter 10
Accessing Database Files

Programming In
Visual Basic .NET
Visual Basic and Databases

- VB projects can display and update the data from database files
- VB.NET uses ADO.NET for database access
- ADO.NET is the next generation of database technology, based on Microsoft's previous version ActiveX Data Objects (ADO)
ADO.NET

• Data is stored and transferred in Extensible Markup Language (XML)
• Allows access to database data in many formats
• Two types of Connections
  – SQLClient for SQL Server
  – OLEDB for all other database formats
Database Terminology

- Databases
- Tables
  - Records
    - Fields
    - Primary Key
    - Foreign Key
- Relationships
- SQL: Industry-standard non-procedural data access language
Relational Databases

- Database includes
  - Tables => collection of related data
  - Queries => SQL designed to select data from tables

- Table
  - Record => row of related data for one instance
  - Field => column of specific data element
  - Keys
    - Primary => uniquely identifies a record
    - Foreign => links record to related record in related table
XML Data

- Industry-standard format for storing and transferring data
- Specifications at http://www.w3.org/XML, World Wide Web Consortium (W3C)
- The needed XML for accessing databases will be automatically generated for you in Visual Basic
XML

• Data stored in XML is all text
• XML files can be edited by any text editor, such as Notepad
• Tags
  – Used for identification, similar to HTML
  – Not predefined like HTML tags
  – Can identify database fields by name
XML Files

- **XML Data File**
  - Contains actual data

- **XML Schema File**
  - Contains descriptions for
    - Fields
    - Data Types
    - Constraints, such as required fields
  - .xsd file in Solution Explorer
Using ADO.NET and VB

• Can display data from a database on
  – Windows Form
  – Web Form

• Add controls to form and bind data to them
  – Normal Controls like label, textbox
  – Special controls designed just for data like DataGrid, DataList
Initial Steps for Data Access

• Set up a **Connection**
  – Establishes a link to a data source, a specific file or server

• Set up a **Data Adapter**
  – Handles retrieving and updating the data; creates a **Dataset**

• Add controls to form and set properties to bind them to fields in the **dataset**

• Write code to fill the **dataset**
Steps for Data Access (cont.)

Data Source

Connection

Data Adapter

Dataset

Web Form

Windows Form

Specific file

OleDbConnection Object

Handles data transfer and provides data for dataset; uses SQL

Actual data, can contain multiple tables

Bound controls display data
Connection

- Used to establish link from specific file or database to program

- Types
  - SqlConnection
    - Microsoft SQL Server databases only
  - OleDbConnection
    - All other databases
Setting Up a Connection

• Can be accomplished from
  – Toolbox
  – Server Explorer window
  – Within Data Adapter Configuration Wizard using New Connection button

• No matter which method above is used to begin the connection, the Data Link Properties dialog appears for you to complete the task
Creating a Connection from Toolbox

- Click Data in Toolbox
- Drag one of the connection objects from toolbox to form
- Properties Window, ConnectionString, New Connection
Creating a Connection from Server Explorer

- Click Connect to Database button

  **OR**

- Right-click Data Connections
Lab 1

- Select the Server Explorer toolbar
- Select the Connect to database button to create a new connection
Data Link Properties Dialog

- Used to define Connection's
  - OLE DB Provider
  - Actual DB Filename
  - Logon UserName and Password (optional)

- Used to Test Connection
Lab 1

• In the connection wizard select the Provider tab then select the Jet 4.0 driver
• Click Next
Lab 1

- On the Connection tab enter the location of the database.
- Click the Test Connection button. If the connection was successful you will see the following message box.
- Click OK.
Connections in Server Explorer

- Shows all created Connections
- Same Connection can be used for multiple projects
- Expand Connection node to view tables and fields
Data Adapter

- Does all the work of passing data back and forth between a data source and a program
- Does not have to be a database
- Can be text file, object, or even an array
- Types, must match Connection type
  - OleDbDataAdapter
  - SqlDataAdapter
Setting Up Data Adapter

• Can be accomplished from
  – Server Explorer
  – Toolbox

• Use db prefix
Setting Up a Data Adapter from Server Explorer

- Drag Table name to form
- OR
- CTRL-click Field names and drag to form
Setting Up a Data Adapter from Toolbox

- Click Data in Toolbox
- Drag one of the data adapter objects from toolbox to form
  - OleDbDataAdapter
  - SqlDataAdapter
- Data Adapter Configuration Wizard opens
Lab 1

- Expand the nodes of the connection so you can see the table and its fields.
- Next click and drag the table Emp to the form, this will create a Connection object and a Data Adapter object.
Lab 1

- Change the name of the connection to cnDemo
- Change the name of the Data Adapter to daDemo
- Right click the Data Adapter and select Configure Data Adapter
Data Adapter Configuration Wizard

- Walks you through the steps for selecting fields in a database
- Only used if you create/add a Data Adapter from the Toolbox
- Can be used to modify existing Data Adapters also
Data Adapter Configuration Wizard (cont.)

• Defines
  – Connection to use
    • Existing Connection
    • Or new connection can be created
  – Query Type
    • SQL for selecting data from database; Query Builder available for generating SQL
    • Or existing/new stored procedure if using SQL Server
Data Adapter/Query Builder-SQL Example

The Data Adapter/Query Builder is a tool used to create SQL statements for database operations. In this example, we are creating an SQL query to select specific columns from the 'Books' table:

```
SELECT Author, ISBN, Title
FROM Books
```

The Query Builder is used to graphically design the query, and the Data Adapter Configuration Wizard helps generate the SQL statements. The Select statement will be used to create the Insert, Update, and Delete statements.

Type in your SQL Select statement or use the Query Builder to graphically design the query.
Lab 1

- Take the defaults for the first 3 steps.
- Type in the SQL select statement shown if yours does not match.
- Click Next
Lab 1

- If you entered all of the information correct your screen should look like this.
- Click Finish
Dataset

• Temporary set of data stored in memory
• In ADO.NET datasets are disconnected; the copy of data kept in memory does not keep an active connection to the data source
• Dataset may contain multiple tables and relationships (But in this Introductory text you will only use 1 table per dataset, multiple tables are covered in the associated Advanced text)
Defining a Dataset

- Drag **Dataset** control from the toolbox to form
- Select **Data** menu, **Generate Dataset**
Defining a Dataset (cont.)

- Name dataset with **ds prefix** in Generate Dataset dialog
Lab 1

- Next right click the Data Adapter and select Generate Dataset
Lab 1

- Give the Dataset a new name: dsEmp
- Click OK
Initial Code Requirements

• Write code to fill the dataset (which will open the connection to the data source and retrieve the data into the dataset)
• Any controls bound to the dataset will automatically fill with data
Data Passing from Data Source to Dataset to User Interface

Presentation Tier
- Windows Forms
- Web Forms

Business Tier
- Data Object (Class)
- Dataset
- Data Adapter

Data Tier
- Data Source
Fill Method

• Method of the Data Adapter executed at run time to fill the dataset
• Usually coded in Form_Load (or Page_Load for Web Projects)
• General Form
  \[\text{DataAdapterName}\text{.Fill (DataSetName)}\]
• Example
  \[\text{dbRnR. Fill (dsBooks)}\]
Option Explicit On
Option Strict On
...
Private Sub Form1_Load(...) Handles MyBase.Load
daDemo.Fill(DsDemo1)
End Sub
Lab 1

- Alter the (DataBindings – Text) property of each textbox and set it to its appropriate field in the Dataset.
- Run the program and you should see the first record display on the form.
Binding Data to Controls

• Windows Form
  – Set properties of the controls
  – Call Fill method

• Web Form
  – Use Fill and BindData methods in code to actually bind controls
  – Since Web Forms are static data they do not retain data
  – Each time a page is displayed the dataset is recreated
DataAdapter Preview (p 422)

- Allows you to see what your data looks like while you are designing the user interface
- Right-click DataAdapter control, select Preview Data
- In the resulting dialog box verify that the correct DataAdapter is in the dropdown
- Click Fill Dataset, resize columns and scroll as necessary
Types of Data Binding

• Simple
  – Connects 1 control to 1 data element
  – Use for TextBoxes and Labels
  – Set DataBindings property

• Complex
  – Binds more than 1 data element to a single control
  – Used for DataGrids and ListBoxes
  – Set DataSource and DataMember properties (at least)
Navigating Through DataSets

• Refer to properties of the Windows Form's BindingContext

• Position property
  – Stores current record position
  – Zero Based
  – Add 1 to move to next record
  – Subtract 1 to move to previous record
  – To use as record number, add 1
Navigation Examples

' Move to next record
Me.**BindingContext**(dsBooks1, "Books").**Position** += 1

' Move to previous record
Me.**BindingContext**(dsBooks1, "Books").**Position** -= 1

' Obtain current record number
intRecordPosition=Me.**BindingContext**(dsBooks1, "Books").**Position** + 1

**“Books” is the name of the table**
Retrieving Record Count

- Specify
  - DataSet
  - Table Name
  - Rows Collection
  - Count Property

- Example

  ```csharp
  intRecordCount=dsBooks1.Tables("Books").Rows.Count
  ```
Lab 1

![Form 1 with input fields for Name, Address, City, State, and ZIP, and buttons for btnFirst, btnPrev, btnNext, btnLast]
Lab 1 Code

Private Sub btnPrev_Click(...) Handles btnPrev.Click
    Me.BindingContext(DsEmp1, "Emp").Position -= 1
End Sub

Private Sub btnNext_Click(...) Handles btnNext.Click
    Me.BindingContext(DsEmp1, "Emp").Position += 1
End Sub

Private Sub btnFirst_Click(...) Handles btnFirst.Click
    Me.BindingContext(DsEmp1, "Emp").Position = 0
End Sub

Private Sub btnLast_Click(...) Handles btnLast.Click
    Me.BindingContext(DsEmp1, "Emp").Position = _
        Me.BindingContext(DsEmp1, "Emp").Count - 1
End Sub
Updating a Database File

• Types of updating
  – Modify existing records
  – Add records
  – Delete records

• Must display the dataset in bound controls that can be used for input (no labels)

• You must write code to actually update the database
Typical Form for Updating

Add caption changes to Cancel during Add or Edit

Save Button is enabled only when an Add or Edit is in progress

All Navigation Buttons are disabled during Add or Edit
Update Method

• Method of DataAdapter used to write changes to database (data source)

• Can be executed
  – After every change, perform data validation, then execute Update (*Save Record button*)
  – Once, just before program termination for all changes execute Update (*Form's Closing event*)
Update Method (cont.)

- Automatically loops through the records in a table to determine what type of update is needed and then performs the required SQL:
  - Update
  - Insert
  - Delete
- Execute during Form's Closing event if running once to apply all changes
Update Method (cont.)

• Return Values
  – Number of records affected by Update
  – Zero, if operation fails

• General Form

  $\text{DataAdapterName}.\text{Update} (\text{DataSet, } "\text{Table}\")$

• Example

  $\text{dbRnR. Update} (\text{dsBooks1, } "\text{Books}\")$
Example - Update Form's Closing Event

' mblnIsDirty was declared in Declarations section
'Save changes, Update Database
If mblnIsDirty Then
    If MessageBox.Show("Save Changes?", "Books" _
        MessageBoxButtons.YesNo, _
        MessageBoxIcon.Question ) = DialogResult.Yes Then
        Try
            dbRnR.Update(dsBooks1, "Books")
        Catch
            MessageBox.Show("Error saving file", "Books")
        End If
    End If
End If
Lab 1

• Add the following buttons to support
  – Add
  – Edit
  – Delete
  – Save
  – Cancel
Lab 1 Code

Private Sub Form1_Closing(...) Handles MyBase.Closing
daDemo.Update(DsDemo1, "Emp")
End Sub

** Note **
If the above line of code generates an error it is possibly due to a bug in the data adapter. Delete the data set and the data adapter and recreate them. In the Lab 1 example change the SQL to read Select * from Emp;
Logic of Update Programs

- Enclose all statements that access the database in Try/Catch blocks
- Display data in bound input controls
  - Initially set ReadOnly property = True
- When user clicks Edit button
  - Set ReadOnly = False
  - Disable Navigation buttons
  - Provide Cancel and Save buttons for user
Saving Changes Logic - User Clicks Save Button

- Perform Data Validation
  - If valid data then
    • Set ReadOnly = True
    • Set mblnDirty = True
    • Enable Navigation buttons
    • Disable Save button
  - If invalid data then
    • Notify user with MessageBox
    • Set focus to control containing invalid data
Canceling Changes Logic -
User Click Cancel Button

• Replace changed values in controls with original values
  – See RejectChanges procedure (p 441)
  – Set ReadOnly = True
  – Enable Navigation buttons
  – Disable Save button
Deleting Records

• Execute Delete Method of the table's Rows collection

• Example

```vbnet
Dim intCurrRecNum As Integer
intCurrRecNum = Me.BindingContext(dsBooks1, _
    "Books").Position
mlnIsDirty = True
```
Adding Records

- If user clicks Add button
  - Clear bound input controls
  - Set ReadOnly = False
  - Set mblnAdding = True
  - Disable Navigation buttons
  - Provide Cancel and Save buttons for user

- Logic for Save and Cancel are same as on previous slide except for resetting mblnAdding to False
Example - Saving an Added Record

Dim newRow As DataRow = dsBooks1.Books.NewRow

newRow("Author") = txtAuthor.Text
newRow("ISBN") = txtISBN.Text
newRow("Title") = txtTitle.Text

Logic for Save Button

• Must determine type of operation in progress since used for saving changes made as
  – Edits
  – Adds
• Use mblnAdding variable to identify Add operations in progress
Lab 1 Code

Private Sub LockStatus(ByVal blnState As Boolean)
    txtName.ReadOnly = blnState
    txtAddress.ReadOnly = blnState
    txtCity.ReadOnly = blnState
    txtState.ReadOnly = blnState
    txtZIP.ReadOnly = blnState
End Sub
Lab 1 Code

Private Sub btnAdd_Click(...) Handles btnAdd.Click
    mblnAdd = True
    txtName.Clear()
    txtAddress.Clear()
    txtCity.Clear()
    txtState.Clear()
    txtZIP.Clear()
    LockStatus(False)
End Sub

Private Sub btnEdit_Click(...) Handles btnEdit.Click
    mblnAdd = False
    LockStatus(False)
End Sub
Private Sub btnSave_Click(...) Handles btnSave.Click
    If mblnAdd Then
        Dim newRow As DataRow = DsDemo1.Emp.NewRow
        newRow("Name") = txtName.Text
        newRow("Address") = txtAddress.Text
        newRow("City") = txtCity.Text
        newRow("State") = txtState.Text
        newRow("ZIP") = txtZIP.Text
        DsDemo1.Emp.Rows.Add(newRow)
        daDemo.Update(DsDemo1, "Emp")
        DsDemo1.Clear()
        daDemo.Fill(DsDemo1)
    Else
        btnLast_Click(sender, e)
        daDemo.Update(DsDemo1, "Emp")
        DsDemo1.Clear()
        daDemo.Fill(DsDemo1)
    End If
End Sub
Private Sub btnCancel_Click(...) Handles btnCancel.Click
    Dim intPos As Integer = Me.BindingContext(DsDemo1, "Emp").Position
    Dim curRow As DataRow = DsDemo1.Emp.Rows(intPos)
    txtName.Text = Convert.ToString(curRow("Name", _
                                   DataRowVersion.Original))
    txtAddress.Text = Convert.ToString(curRow("Address", _
                                   DataRowVersion.Original))
    txtCity.Text = Convert.ToString(curRow("City", _
                                   DataRowVersion.Original))
    txtState.Text = Convert.ToString(curRow("State", _
                                   DataRowVersion.Original))
    txtZIP.Text = Convert.ToString(curRow("ZIP", _
                                   DataRowVersion.Original))
    LockStatus(True)
End Sub
Private Sub btnDelete_Click(...) Handles btnDelete.Click
    Dim intRec As Integer = Me.BindingContext(DsDemo1, _
        "Emp").Position
    DsDemo1.Emp.Rows(intRec).Delete()
    daDemo.Update(DsDemo1, "Emp")
    DsDemo1.Clear()
    daDemo.Fill(DsDemo1)
    btnFirst_Click(sender, e)
End Sub
Creating Bound DataGrid Control (p 416-421)

• New control in VB.NET
• Displays database records in a grid with records as rows and fields as columns
• Steps
  – Set up/add Connection, DataAdapter, DataSet
  – Add DataGrid control from toolbox (dbg prefix)
Creating Bound DataGrid Control (Steps cont.)

- Set properties of DataGrid to bind to data
  - **DataSource** to DataSet name
  - **DataMember** to Table name
  - **DataKeyField** to Table's Primary Key field

- Add code to populate control with data
  - Windows Form, Form_Load ==> 
    `DataAdapterName.Fill (DataSetName)`
  - Web Form, Page_Load ==> 
    `DataAdapterName.Fill (DataSetName)`
    `DataGridName.DataBind()`