Hands-On Lab
Lab Manual

DEV-HOL08: Data Access using ADO.NET (VB)

Please do not remove this manual from the lab
The lab manual will be available from CommNet

Release Date: June 2003
Information in this document is subject to change without notice. The example companies, organizations, products, people, and events depicted herein are fictitious. No association with any real company, organization, product, person or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarked, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

©2003 Microsoft Corporation. All rights reserved.


Other product and company names herein may be the trademarks of their respective owners.
Lab 8: Data Access using ADO.NET

This lab provides you with an overview of ADO.NET. The first exercise provides a brief introduction to ADO.NET and subsequent exercises delve into details on how to perform different database operations such as insert, update and delete of database records, performing database transaction. The lab also discusses programmatically creating DataSet, DataAdapter, DataReader, Command objects and work with the DataGrid control.

Lab Setup

The working solution and associated files for this lab are located at %SystemDrive%\Microsoft Hands-On-Lab\DEV-HOL08\Solution\Lab8-VB.

Lab Objective

Estimated time to complete this lab: 60 minutes

After completing this lab, you will be able to:

- Understand ADO.NET
- Work with the SqlCommand and SqlDataReader objects
- Work with the DataSet and SqlDataAdapter objects
- Insert, update and delete records using a DataSet
- Load a native ADO Recordset into a DataSet

Exercise 1 – Introduction to ADO.NET

ADO.NET provides consistent access to data sources such as Microsoft SQL Server, as well as data sources like OLE DB and XML. Data-driven applications can use ADO.NET to connect to data sources, retrieve and manipulate data.

Task 1 – Overview of ADO.NET Data Providers

Data providers are a core component of the ADO.NET architecture, which enable communication between an application and the data source. A data provider allows you to connect to a data source, retrieve and manipulate data, and update the data source. The data provider in the .NET Framework serves as a bridge between an application and a data source.

The ADO.NET data provider is designed to be lightweight, creating a minimal layer between the data source and your code, increasing performance without sacrificing functionality.

The four core objects that make up an ADO.NET data provider are:

- **Connection** – Establishes a connection to a specific data source.
- **Command** – Executes a command against a data source
- **DataReader** – Provides a fast, forward-only, read-only access to data.
**DataAdapter** – Populates a DataSet and resolves updates with the data source.

Out-of-the-box, the .NET Framework includes 4 data providers.

- **SQL Server data provider**
  
The SQL Server data provider uses the SQL Server-specific data transfer protocol (Tabular Data Stream) to communicate with SQL Server. The data provider is optimized to access SQL Server directly without adding an OLE DB or ODBC Connectivity layer. This provider is recommended for applications using Microsoft SQL Server 7.0 or later.

  To use this data provider include the following “using” statement in your code:

```csharp
Imports System.Data.SqlClient
```

- **OLEDB data provider**
  
The .NET Framework data provider for OLEDB uses native OLEDB through the COM interoperability layer to enable data access. The OLEDB data provider supports both manual and automatic transactions. This provider is recommended for middle-tier applications using Microsoft SQL Server 6.5 or earlier, or any OLE DB compliant provider.

  To use this data provider include the following “using” statement in your code:

```csharp
Imports System.Data.OleDb
```

- **ODBC data provider (New to Framework 1.1)**
  
The .NET Framework data provider for ODBC uses the native ODBC Driver Manager to enable access to ODBC-data sources. The ODBC data provider supports both local and distributed transactions. This provider is recommended for applications using ODBC data sources.

  To use this data provider include the following “using” statement in your code:

```csharp
Imports System.Data.Odbc
```

- **Oracle data provider (New to Framework 1.1)**
  
The .NET Framework data provider for Oracle enables data access to Oracle data sources using the Oracle client connectivity software. The data provider supports Oracle client software version 8.1.7 and later. This provider is recommended for applications using Oracle data sources.

  **Note:** To use this provider, you will need to import this namespace using the **Project | Add Reference...** menu command and select the **System.Data.OracleClient.dll** assembly.

  To use this data provider include the following “using” statement in your code:

```csharp
Imports System.Data.OracleClient
```
Task 2 – Connection String

The connection string in ADO.NET is similar to an OLEDB connection string. Applications need to specify a connection string for connecting to the underlying data source. The ConnectionString property of the Connection object can be set only when the connection is closed. The connection string is a series of key=value pairs delimited by semi-colons. To connect to a SQL Server running on the local machine, specify "(local)" for the server. The following elements constitute the ConnectionString property for a SqlConnection object:

<table>
<thead>
<tr>
<th>Data Source-or- server</th>
<th>The name or network address of the instance of SQL Server to which to connect.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Security-or-</td>
<td>When false, User ID and Password are specified in the connection. When true, the current windows account credentials are used for authentication</td>
</tr>
<tr>
<td>Trusted_Connection</td>
<td></td>
</tr>
<tr>
<td>User ID</td>
<td>The SQL Server login account</td>
</tr>
<tr>
<td>Password-or-Pwd</td>
<td>The password for the SQL Server account logging on</td>
</tr>
<tr>
<td>Initial Catalog-or-Database</td>
<td>The name of the database.</td>
</tr>
</tbody>
</table>

Exercise 2 – Working with the SqlCommand and SqlDataReader objects

In this exercise, you will accomplish common database tasks including connecting to a SQL server database using the SqlConnection class and inserting, updating and deleting data using the SqlCommand class. You will also use the SqlDataReader class to retrieve a read-only, forward-only stream of data from a database. Using the DataReader can increase application performance and reduce system overhead because only one row at a time is ever in memory. Finally you will understand how to perform database transactions.

Task 1 – Storing connection string in an application configuration file

It is very common to store the database connection string in a single place for ease of application maintenance. In this task you will create a project and then you will create an application configuration file, which will store the connection string.

- Click Start, select Programs | Microsoft Visual Studio .NET 2003 | Microsoft Visual Studio .NET 2003
- Select File | New | Project menu command
- In the Project Types pane select Visual Basic Projects
- In the Templates list select Windows Application
• Set the project name to “Lab8-VB”

• Click OK

• To add an application configuration file, select Project | Add New Item... menu command.

• In the Add New Item dialog box, select Application Configuration File from the Templates list and click Open.

• Write the following highlighted XML content inside the <configuration> tag in App.config file:

```xml
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <appSettings>
    <add key="ConnectionString" value="server=(local);Trusted_Connection=yes; database=Northwind"/>
  </appSettings>
</configuration>
```

Task 2 – Using SqlCommand to Insert, Update and Delete data

The SqlCommand object represents a Transact-SQL statement or a stored procedure, which will be executed against a SQL Server database. In this task, you will create a connection to SQL Server and then insert, update and delete data using the SqlCommand object.

• To open Solution Explorer, select View | Solution Explorer menu command.

• Right-click Form1.vb in the solution explorer and select View Code.

• Add the following highlighted code to the Imports block at the top of the code:

```vbnet
Imports System.Data.SqlClient
Public Class Form1
  Inherits System.Windows.Forms.Form
  ...
  ...

  Dim mySqlConnection As SqlConnection

  Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
  End Sub
```

• Declare a SqlConnection variable in the general declaration section of Form1 class.

• To view Form1 in Design mode, select View | Designer menu command.

• Double-click in design area of Form1, Visual Studio .NET 2003 will navigate to the source file of Form1 and will create an event handler for the form load event (Form1_Load).

• Assign a value to the mySqlConnection variable in the Form1_Load event handler:
• The above line of code, reads the value of the “ConnectionString” key from the Application configuration file (App.Config).

• To view Form1 in Design mode, select View | Designer menu command

• To open the Toolbox, select View | Toolbox menu command.

• Drag a Button from the toolbox and drop it on to the form

• Right-click the button and select the Properties menu command.

• Set the following properties of the button control using the Properties window:
  
  Name          - "btnInsert"
  Text          - "Insert Command"

• Expand the width of the button control to fit the text.

• Drag another Button from the toolbox and drop it on to the form

• Right-click the button and select the Properties menu command.

• Set the following properties of the button control using the Properties window:
  
  Name          - "btnUpdate"
  Text          - "Update Command"

• Expand the width of the button control to fit the text.

• Drag another Button from the toolbox and drop it on to the form

• Right-click the button and select the Properties menu command.

• Set the following properties of the button control using the Properties window:
  
  Name          - "btnDelete"
  Text          - "Delete Command"

• Expand the width of the button control to fit the text.

• Double-click the Insert Command button control. Visual Studio .NET will navigate to the code view of Form1 (Form1.vb) and create an event handler for the Click event of the Insert Command button.

• Add the following code in the btnInsert_Click method to insert data about a new customer using SqlCommand object:

```vbnet
Private Sub btnInsert_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCommand.Click

End Sub
```
Dim mySqlCommand As SqlCommand = New SqlCommand

'Open the connection
mySqlConnection.Open()

'Assign the connection property.
mySqlCommand.Connection = mySqlConnection

Try
    ' Insert new record for XYZ customer
    mySqlCommand.CommandText = "INSERT INTO Customers (CustomerId, CompanyName, ContactName, ContactTitle, Address) Values ('XYZ','XYZ Company', 'John Smith', 'Owner','One My Way')"
    mySqlCommand.ExecuteNonQuery()
    MessageBox.Show("New Customer data recorded!")
Catch ex As Exception
    MessageBox.Show("Error occurred, customer data could not be recorded: "+ ex.Message)
End Try

Finally
    'Close connection if it is open
    If (mySqlConnection.State = ConnectionState.Open) Then
        mySqlConnection.Close()
    End If
End Try

• The **State** property of the SqlConnection object indicates the state of the connection, whether it is open or closed. This property can be used to open the connection if it is closed to minimize the server resources consumed by the application.

• The **ExecuteNonQuery** method of the SqlCommand object is used when the query being executed, Insert in this case, is not expected to return any results.

• To compile and run your application select **Debug | Start Without Debugging** menu command or press **CTRL+F5**.

• Click the **Insert Command** button to perform the insert operation

• A message box appears saying “New Customer data recorded!”

• Click **OK** to close the message box.

• **Close Form1**

• To verify that the data has indeed been recorded successfully
  
  o Switch to Visual Studio .NET IDE
  
  o Select **View | Server Explorer** menu command
  
  o To add a new Data Connection, right-click **Data Connections** node and select **Add Connection...** menu command
In the Data Link Properties dialog box, specify the server name as "(local)"

- Select the radio button “Use Windows NT Integrated Security”
- In the “Select the database on the server” combo-box, select Northwind
- Click Test Connection. You will see a Microsoft Data Link message box saying “Test Connection succeeded”.
- Click OK to close the message box.
- Click OK to close the Data Link Properties dialog box.
- Select View | Server Explorer menu command and navigate to the Data Connections | <servername>.NorthWind.dbo | Tables | Customers node
Double-click the Customer table to view all data in the table.

Scroll to the bottom of the table to verify that a new customer with an ID of ‘XYZ’ has been recorded.

- To open Solution Explorer, select View | Solution Explorer menu command.
- To view Form1 in Design mode, double-click Form1.vb in the Solution Explorer.
- Double-click the Update Command button control. Visual Studio .NET will navigate to the code view of Form1 (Form1.vb) and will create an event handler for the Click event of the Update Command button.
- Add the following code in the btnUpdate_Click method to update the inserted data using SqlCommand object:

```vbnet
Private Sub btnUpdate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnUpdate.Click
    Dim mySqlCommand As SqlCommand = New SqlCommand
    'Open the connection
    mySqlConnection.Open()

    'Assign the connection property.
    mySqlCommand.Connection = mySqlConnection

    Try
        'Update XYZ customer
        mySqlCommand.CommandText = "UPDATE Customers SET ContactName='Ian Smith'
                                   WHERE CustomerId='XYZ'"
        mySqlCommand.ExecuteNonQuery()
        MessageBox.Show("Customer Data successfully updated.")
    Catch ex As Exception
        MessageBox.Show("Error occurred, customer data could not be updated: " + ex.Message)
    Finally
        'Close connection if it is open
        If (mySqlConnection.State = ConnectionState.Open) Then
            mySqlConnection.Close()
        End If
    End Try
End Sub
```

- To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.

- Click the Update Command button to update the inserted row using the SqlCommand object.
- A message box appears saying "Customer Data successfully updated."
- Click OK to close the message box.
- Close Form1
- To verify that the data has indeed been updated successfully.
o Switch to Visual Studio .NET IDE
o Select View | Server Explorer menu command
o Navigate to the Data Connections | <servername>.NorthWind.dbo | Tables | Customers node
o Double-click the Customer table to view all data in the table
o To refresh the data display, select Query | Run menu command
o Scroll to the bottom of the table to verify that the customer name has changed to “Ian Smith”.

- To open Solution Explorer, select View | Solution Explorer menu command.
- To view Form1 in Design mode, double-click Form1.vb in the Solution Explorer
- Double-click the Delete Command button control. Visual Studio .NET will navigate to the code view of Form1 (Form1.vb) and will create an event handler for the Click event of the Delete Command button.
- Add the following code in the btnDelete_Click method to delete the inserted data using the SqlCommand object:

```csharp
Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDelete.Click
    Dim mySqlCommand As SqlCommand = New SqlCommand
    'Open the connection
    mySqlConnection.Open()

    'Assign the connection property.
    mySqlCommand.Connection = mySqlConnection
    Try
        ' Delete XYZ customer
        mySqlCommand.CommandText = "DELETE FROM Customers WHERE CustomerId='XYZ'"
        mySqlCommand.ExecuteNonQuery()
        MessageBox.Show("Customer Data successfully deleted.")
    Catch ex As Exception
        MessageBox.Show("Error occurred, data could not be deleted: " + ex.Message)
    Finally
        'Close connection if it is open
        If (mySqlConnection.State = ConnectionState.Open) Then
            mySqlConnection.Close()
        End If
    End Try
End Sub
```
- To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.
- Click the Delete Command button to perform the Delete operation.
• A message box appears saying “Customer Data successfully deleted.”
• Click OK to close the message box.
• Close Form1
• To verify that the data has indeed been updated successfully
  o Switch to Visual Studio .NET IDE
  o Select View | Server Explorer menu command
  o Navigate to the Data Connections | <servername>.NorthWinddbo | Tables | Customers node
  o Double-click the Customer table to view all data in the table
  o To refresh the data display, select Query | Run menu command
  o Scroll to the bottom of the table to verify that the record with customer ID ‘XYZ’ is not present.

Task 3 – Working with the SqlTransaction object

A transaction is a unit of work in which a series of operations occur between the BEGIN TRANSACTION and END TRANSACTION statements of an application. A transaction executes exactly once and is atomic—all the work is done or none of it is performed. The SqlTransaction class represents a Transact-SQL transaction to be made in a SQL Server database. In this task, you will work with SqlTransaction object.

• The following code sample represents the structure of a transacted database operation:

```vbnet
Dim myTrans As SqlTransaction
myTrans = mySqlConnection.BeginTransaction()
SqlCommand.Transaction = myTrans
try
...
...
myTrans.Commit()
catch ex as Exception
  myTrans.Rollback()
... end try
```

• To open Solution Explorer, select View | Solution Explorer menu command.
• To view Form1 in Design mode, double-click Form1.vb in the Solution Explorer
• To open the Toolbox, select View | Toolbox menu command.
• Drag a Button from the toolbox and drop it on to the form
• Right-click the button and select the **Properties** menu command.

• Set the following properties of the button control using the **Properties** window:
  
  **Name** - “btnTransaction”
  
  **Text** - “Transaction”

• Expand the width of the button control to fit the text.

• Double-click the **Transaction** button control. Visual Studio .NET will navigate to the code view of **Form1** (Form1.vb) and will create an event handler for the **Click** event of the **Transaction** button.

• Add the following code in the **btnTransaction_Click** method to perform a transacted insert, update and delete operation using the SqlCommand object:

```vbnet
Private Sub btnTransaction_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnTransaction.Click
    Dim myTrans As SqlTransaction
    Dim mySqlCommand As SqlCommand = New SqlCommand

    'Open the connection
    mySqlConnection.Open()

    'Assign the connection property.
    mySqlCommand.Connection = mySqlConnection
    myTrans = mySqlConnection.BeginTransaction()
    mySqlCommand.Transaction = myTrans

    Try
        ' Insert new record for XYZ customer
        mySqlCommand.CommandText = "INSERT INTO Customers (CustomerId, CompanyName, ContactName, ContactTitle, Address) Values ('XYZ','XYZ Company', 'John Smith', 'Owner', 'One My Way')"
        mySqlCommand.ExecuteNonQuery()

        ' Update XYZ customer
        mySqlCommand.CommandText = "UPDATE Customers SET ContactName='Ian Smith' WHERE CustomerId='XYZ'"
        mySqlCommand.ExecuteNonQuery()

        ' Delete XYZ customer
        mySqlCommand.CommandText = "DELETE FROM Customers WHERE CustomerId='XYZ'"
        mySqlCommand.ExecuteNonQuery()

        MessageBox.Show("Committing transaction")
        myTrans.Commit()
        MessageBox.Show("Data successfully updated.")
    Catch ex As Exception
        myTrans.Rollback()
    End Try
End Sub
```
MessageBox.Show("Error occurred, data has not been successfully updated: " + ex.Message)
Finally
'Close connection if it is open
    If (mySqlConnection.State = ConnectionState.Open) Then
        mySqlConnection.Close()
    End If
End Try

• To compile and run your application select **Debug | Start Without Debugging** menu command or press **CTRL+F5**.

• Click the **Transaction** button to perform all the operations, either all the operations occur or none of them occur.

• A message box appears saying “Committing transaction”.

• Click **OK** to close the message box.

• A message box appears saying “Data successfully updated.”

• Click **OK** to close the message box.

• Close **Form1**.

• To verify that the transaction was successful
  o Switch to Visual Studio .NET IDE
  o Select **View | Server Explorer** menu command
  o Navigate to the **Data Connections | <servername>.NorthWind.dbo | Tables | Customers** node
  o Double-click the **Customer** table to view all data in the table
  o To refresh the data display, select **Query | Run** menu command
  o Scroll to the bottom of the table to verify that the record with customer ID ‘XYZ’ is not present.

**Task 4 – Using SqlDataReader**

SqlDataReader provides a mechanism for reading a forward-only stream of rows returned in a SQL Server database query. In this task, you will use SqlDataReader object. You will use CommandBehavior.CloseConnection parameter of DataReader so that the connection gets closed automatically when the command is executed.

• Select **View | Solution Explorer** menu command.

• To view **Form1** in Design mode, double-click Form1.vb in the Solution Explorer
• To open the Toolbox, select View | Toolbox menu command.

• Drag another Button from the toolbox and drop it on to the form as shown in Figure 8.3 below.

• Right-click the button and select Properties.

• Set the following properties of the button control using the Properties window:
  
  Name - “btnDataReader”
  Text - “Data Reader”

• Drag a ComboBox from the toolbox and drop it on to the form as shown in Figure 8.3 below.

• Right-click the comboBox and select Properties.

• Set the following properties of the comboBox control using the Properties window:
  
  DropDownStyle - “DropDownList” (select from the drop down list)

   ![Figure 8.3 Form1 Design view](image)

• Double-click the Data Reader button control. Visual Studio .NET will navigate to Form1 code view and will create an event handler for the Click event of the Data Reader button.

• Open a database connection and read the data in SqlDataReader using SqlCommand object in the btnDataReader_Click method as shown below:

  ```csharp
  Private Sub btnDataReader_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnDataReader.Click
    Dim myReader As SqlDataReader
    Dim mySqlCommand As SqlCommand = New SqlCommand("select ContactName from customers", mySqlConnection)
  ```
Try
    mySqlConnection.Open()
    myReader = mySqlCommand.ExecuteReader(CommandBehavior.CloseConnection)
    While (myReader.Read())
        ComboBox1.Items.Add(myReader("ContactName").ToString())
    End While
    MessageBox.Show("All names loaded successfully!")
    Catch ex As Exception
        MessageBox.Show("Error occurred: " + ex.Message)
    Finally
        If Not myReader Is Nothing Then
            myReader.Close()
        End If
    End Try
End Sub

- To compile and run your application select **Debug | Start Without Debugging** menu command or press **CTRL+F5**.
- Click the **Data Reader** button to load all customer contact names into the combo box.
- A message box appears saying “All names loaded successfully!”
- Click **OK** to close the message box. The combo box is populated with the data.
- Expand the combo box to verify that the contact names are listed
- Close **Form1**.

Note: Since you have used CommandBehavior.CloseConnection parameter in ExecuteReader method of SqlCommand object, the connection state is closed only when the DataReader is closed, you need not close the connection explicitly.

**Exercise 3 – Working with the DataSet object**

In this exercise you will populate a DataSet with data using the SqlDataAdapter object to retrieve data into the presentation layer. The ADO.NET DataSet is a memory-resident representation of data that provides a consistent relational programming model independent of the data source. The DataSet object represents a complete set of data including tables, constraints, and relationships among the tables. Because the DataSet is independent of the data source, a DataSet can include data local to the application, as well as data from multiple data sources. Interaction with existing data sources is controlled through the DataAdapter.

In this exercise you will:

- Populate a DataSet with data using the SqlDataAdapter object.
- Use SqlCommandBuilder object and the SqlDataAdapter object to insert, update and delete data.
- Call a stored procedure and persist DataSet into Xml.
• Create a DataRelation

Task 1 – Retrieving data with DataSet

In this task, you will retrieve data using a SqlDataAdapter object and fill a DataSet object. Finally you will bind the DataSet to a DataGrid control.

• To open the Solution Explorer, select View | Solution Explorer menu command.

• Right-click project (Lab8-VB) in the Solution Explorer and select Add | Add Windows Form...

• Click Open. Form2.vb is added to the project and is visible in solution explorer.

• Right-click Form2.vb and select View Code menu command.

• At the top section of your code, add the following highlighted code:

\begin{verbatim}
Imports System.Data
Imports System.Data.SqlClient
Public Class Form2
    Inherits System.Windows.Forms.Form

    Dim mySqlConnection As SqlConnection
    Dim myDataSet As DataSet
    Dim mySqlDataAdapter As SqlDataAdapter

    Public Sub Form2_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    End Sub
\end{verbatim}

• To view Form2 in Design mode, select View | Designer menu command

• Double-click in design area of Form2, Visual Studio .NET 2003 will take you to the source file of Form2 and will create form load method (Form2_Load).

• Set the ConnectionString property of the declared variable in Form2_Load event handler:

\begin{verbatim}
Private Sub Form2_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
End Sub
\end{verbatim}

• To view Form2 in Design mode, select View | Designer menu command

• To open the Toolbox, select View | Toolbox menu command.

• Drag a Button from the toolbox and drop it on to the Form2
• Right-click the button and select Properties.
• Set the following properties of the button control using the **Properties** window:

  **Name** - “btnGet”
  **Text** - “Get Data”

• To open the **Toolbox**, select **View | Toolbox** menu command.

• Drag a **DataGrid** control from the toolbox and drop it on to the form as shown in Figure 8.4 below. A DataGrid control is added with the name **dataGrid1**.

![Figure 8.4 Form2 Design view](image)

  **Figure 8.4 Form2 Design view**

• Double-click the **Get Data** button control.

• Add the following highlighted code in the **btnGet_Click** method

  ```vbnet
  Private Sub btnGet_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGet.Click
    mySqlDataAdapter = New SqlDataAdapter("select * from customers", mySqlConnection)
    Try
      myDataSet = New DataSet
      mySqlDataAdapter.Fill(myDataSet, "Customers")
      DataGrid1.DataSource = myDataSet.Tables("Customers").DefaultView
    Catch ex As Exception
      MessageBox.Show("Unable to retrieve Customer data: " + ex.Message)
    End Try
  End Sub
  ```

• To set **Form2** as Startup object
Select Project | Lab8-VB Properties menu command. Lab8-VB Property Pages dialog box appears.

Expand Common Properties and select General.

Set Form2 as the Startup object.

Figure 8.5 Setting Form2 as Startup object

Click OK to close the Lab8-VB Property Pages dialog box.

- To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.
- To load data into the DataGrid control, click Get Data button.
- Close Form2.

Task 2 – Creating Master – Detail relationship

A DataSet can contain multiple related or unrelated tables. Because data can be stored in relational or hierarchical databases, the DataSet object can handle both hierarchical relationships and primary key/foreign key relationships. Relationships in a DataSet can be enforced in different ways. By default, deletions and updates are cascaded: for example, if you delete a Customer row, the related Orders rows are also deleted; if you update the key of a Customer row, the associated foreign key values in the Orders table is also updated. It is possible to change the default behavior of cascading updates.

A DataSet object manages the relationship between tables in a Relations collection. You can add a
Relation to this collection using the column or columns (in a multicolumn key) of the related tables. In this task you will create a relation between Customers and Orders, and name the relation CustOrders. Finally, you will present the relational data in a DataGrid control.

- Declare a `SqlDataAdapter` variable in the general declaration section of Form2 class for managing updates to the `Orders` table. These adapters will be used to managing updates to the back-end SQL database tables:

```csharp
Public Class Form2
    Inherits System.Windows.Forms.Form
    Dim mySqlConnection As SqlConnection
    Dim myDataSet As DataSet
    Dim mySqlDataAdapter As SqlDataAdapter
    Dim mySqlDataAdapter2 As SqlDataAdapter
    ...
    ...
```

- Add the following highlighted code in `btnGet_Click` method: Will need to change to declare data adapters as module level variables and only perform a New in this method:

```csharp
Private Sub btnGet_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGet.Click
    mySqlDataAdapter = New SqlDataAdapter("select * from customers", mySqlConnection)
    mySqlDataAdapter2 = New SqlDataAdapter("select * from orders", mySqlConnection)
    Try
        myDataSet = New DataSet
        mySqlDataAdapter.Fill(myDataSet, "Customers")
        mySqlDataAdapter2.Fill(myDataSet, "Orders")
        ' ADD RELATION
        myDataSet.Relations.Add("CustOrders", myDataSet.Tables("Customers").Columns("CustomerId"), myDataSet.Tables("Orders").Columns("CustomerId"))
        DataGridView1.DataSource = myDataSet.Tables("Customers").DefaultView
    Catch ex As Exception
        MessageBox.Show("Unable to retrieve Customer data: " + ex.Message)
    End Try
End Sub
```

- This code loads data from the Orders table also in the Dataset and creates a DataRelation between the Customers and Orders table.

- To compile and run your application select `Debug | Start Without Debugging` menu command or press `CTRL+F5`.

- Click `Get Data`. Observe the relational data in the DataGrid control, binding the DataGrid to the Customers table, automatically retrieves the related data from the Orders table using the DataRelation between the two tables.

- Close `Form2`. 
Task 3 – Persisting DataSet in an XML document

In this task, you will save a DataSet as an XML document and load XML data into the DataSet.

- To open Form2 in Design mode, select View | Designer menu command
- To view the Toolbox, select View | Toolbox menu command
- Drag a Button from the toolbox and drop it on to the Form2.
- Right-click the button and select Properties.
- Set the following properties of the button control using the Properties window:
  - Name - “btnSave”
  - Text - “Save Xml”
- Drag a Button from the toolbox and drop it on to the Form2.
- Right-click the button and select Properties.
- Set the following properties of the button control using the Properties window:
  - Name - “btnLoad”
  - Text - “Load Xml”
To view Form2 in Code editing mode, select View | Code menu command.

Declare a String variable in the general declaration section of Form2 class. This string will hold the path of the Xml file where the Dataset is persisted.

```csharp
Public Class Form2
    Inherits System.Windows.Forms.Form
    Dim mySqlConnection As SqlConnection
    Dim myDataSet as DataSet
    Dim mySqlDataAdapter As SqlDataAdapter
    Dim mySqlDataAdapter2 As SqlDataAdapter
    Dim xmlFilename as String = "C:\Customers.xml"
    ...
```

To switch to the designer, select the View | Designer menu command.

Double-click the Save Xml button control.

Add the following highlighted code in btnSave_Click method.

```csharp
Private Sub btnSave_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSave.Click
    myDataSet.WriteXml(xmlFilename)
    MessageBox.Show("Customer details stored in: " + xmlFilename)
    myDataSet.Clear() 'this will clear the grid also
End Sub
```

To switch to the designer, select the View | Designer menu command.

Double-click the Load Xml button control.
• Add the following highlighted code in `btnLoad_Click` method

```vbnet
Private Sub btnLoad_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnLoad.Click
    MessageBox.Show("Reading data from XML file... ")
    myDataSet.ReadXml(xmlFilename)
    DataGrid1.DataSource = myDataSet.Tables("Customers").DefaultView
End Sub
```

• To compile and run your application select `Debug | Start Without Debugging` menu command or press `CTRL+F5`.

• Click `Get Data`. The application will load Customer and Order data from the Northwind database

• Click `Save Xml` button to write the Customer-Order data into the Xml file C:\Customers.xml. Click `OK` to close the “Customer details stored in C:\Customers.xml” message box.

• Click `Load Xml` button to read the Customer-Order data from the Xml file. Click `OK` to close the “Reading data from XML File” message box. The application will read the data from C:\Customers.xml file and will fill the data in the DataGrid control using the DataSet object.

• Close `Form2`.

• Open C:\Customers.xml in Internet Explorer and notice how the entire dataset data is persisted in this file.

Task 4 – Working with the Typed DataSets

In this task, you will use a Typed DataSet which allows you to access the data held in the DataSet through a "strongly typed" programming model. This allows the developer to access tables and columns that are part of the DataSet with strongly typed properties on the DataSet. In Task 2 you used the DataSet.Relations.Add method to create a relationship between the Customers and Orders table, in this task we will perform the same operation graphically using the XML Designer

• Select `View | Solution Explorer` menu command.

• Right-click project (Lab8-VB) in the Solution Explorer and select `Add | Add New Item…`

• In the `Add New Item` dialog box, select `Data Set` from the `Templates` list.

• Modify the `Name` to “Customer.xsd” and click `Open`.

Page 22
Figure 8.8: Creating Customer.xsd file

- Select View | Server Explorer menu command to open Server Explorer.
- Navigate to <server name>.Northwind.dbo | Tables | Customers node.
- Drag the Customers table from Server Explorer and drop it in the design area of Customer.xsd file.
- Drag the Orders table from Server Explorer and drop it in the design area of Customer.xsd file.
- Drag the CustomerID field from the Customers table in the design area to the CustomerID field in the Orders table. The Edit Relation dialog box is displayed, change the Name of the relation to “CustOrders”: 
To close the Edit Relation dialog, click **OK**. This creates a typed dataset as shown in Figure 8.10.
Figure 8.10: Customer.xsd in Design mode

- Select **File | Save All** menu command to save the solution.

- Select **View | Solution Explorer** menu command.

- Double-click Form2.vb in the Solution Explorer or click the Form2.vb [Design] tab.

- To view the Toolbox, select **View | Toolbox** menu command.

- Select the **Data** tab in the toolbox.

- Drag a **DataSet** object from the toolbox and drop it in the design area of **Form2**.

- In the **Add DataSet** dialog box, select the **Typed Dataset** radio button and accept the default value in the **Name** combo box as shown below:
Figure 8.11: Add Dataset dialog

- To close the Add Dataset dialog box, click OK. This adds the Customer1 Dataset to the form.
- Right-click the DataGrid control in the design area of Form2 and select Properties.
- Set the DataSource property of DataGrid control to "Customer1.Customers" (select from the drop down). The column headers will appear in the DataGrid control as shown below:

Figure 8.12 Form2 Design view

- To switch to the code editor, select the View | Code menu command
- Edit the declaration of myDataSet variable as shown below:
Public Class Form2  
   Inherits System.Windows.Forms.Form  
   Dim mySqlConnection As SqlConnection  
   'Dim myDataSet as DataSet  
   Dim myDataSet as Customer  
   ...  

• Add the following highlighted code in btnGet_Click method and comment the code which uses the old DataSet data type.

Private Sub btnGet_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGet.Click  
   mySqlDataAdapter = New SqlDataAdapter("select * from customers", mySqlConnection)  
   mySqlDataAdapter2 = New SqlDataAdapter("select * from orders", mySqlConnection)  
   Try  
      'myDataSet = New DataSet  
      myDataSet = New Customer  
      mySqlDataAdapter.Fill(myDataSet, "Customers")  
      mySqlDataAdapter2.Fill(myDataSet, "Orders")  
      'ADD RELATION  
      myDataSet.Relations.Add("CustOrders", myDataSet.Tables("Customers").Columns("CustomerId"), myDataSet.Tables("Orders").Columns("CustomerId"))  
      'DataGrid1.DataSource = myDataSet.Tables("Customers").DefaultView  
      DataGrid1.DataSource = myDataSet.Customers.DefaultView  
      Catch ex As Exception  
         MessageBox.Show("Unable to retrieve Customer data: " + ex.Message)  
      End Try  
   End Sub

• Delete the line of code which adds the DataRelation between Customers and Orders tables  

• Notice the Intellisense offered by the Visual Studio IDE, when typing "myDataSet." The Customers table is available as a property. Additionally, typing "myDataSet.Customers." will make available all columns of the Customers table as properties. This is one of the key benefits of Typed Datasets; the developer can access the database data as persisted objects.  

• To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.

• Click Get Data. Customer data will get populated in the DataGrid control from typed dataset.

• Expand the + icon next to any row and click the link to view Order details for the selected Customer

• Close Form2.

Task 5 – Inserting, Updating and Deleting data using SqlDataAdapter and SqlCommandBuilder

In this task, you will use the SqlDataAdapter object to insert, update and delete data.
• Declare a SqlCommandBuilder variable as shown below:

```csharp
Public Class Form2
    Inherits System.Windows.Forms.Form
    Dim mySqlConnection As SqlConnection
    'Dim myDataSet as DataSet
    Dim myDataSet as Customer
    ...
    Dim cbCustomers As SqlCommandBuilder
    ...
```

• To switch Form2 to the designer view, select the View | Designer menu command

• To open the Toolbox, select View | Toolbox menu command.

• Click Windows Forms tab of Toolbox.

• Drag a Button from the toolbox and drop it on to the form as shown in Figure 8.13 below.

• Right-click the button and select Properties.

• Set the following properties of the button control using the Properties window:
  
  Name - “btnSaveChanges”
  
  Text - “Save Changes”

• Expand the button to fit the text.

Figure 8.13 Form2 Design view

• Double-click the Get Data button control.

• Add the following highlighted code in the btnGetData_Click method

```csharp
Private Sub btnGet_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnGet.Click
```
To switch Form2 to the designer view, select the View | Designer menu command.

Double-click the Save Changes button control.

Add the following highlighted code in the btnSaveChanges_Click method:

```csharp
Private Sub btnSaveChanges_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSaveChanges.Click
    Try
        mySqlDataAdapter.Update(myDataSet, "Customers")
        MessageBox.Show("Changes Recorded Successfully into the Customers table.")
    Catch ex As Exception
        MessageBox.Show("Changes not saved: " + ex.Message)
    End Try
End Sub
```

This code sample saves all changes made to the DataSet back to the database table.

To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.

Click Get Data button to populate all Customer, Order data into the DataGrid control.

Scroll to the end of the DataGrid control. To add a new row, enter the following values in the row marked with a * specify Customer ID as “XXX”, Company Name as “XXX Company” and Contact Name as “John Doe”.

Add another new row, specify Customer ID as “YYY”, Company Name as “YYY Company” and Contact Name as “Jane Doe”.

Change the Company name for Customer ID “WOLZA” to “WolskEE Zajazd”.

To save the changed records to the Database click the Save Changes button. A message box appears saying “Changes Recorded Successfully into the Customers table”.

Click OK to close the message box.

To verify that the changes are saved, click Get Data button to re-populate all Customer, Order data into the DataGrid control.

To delete the Customer with the ID “YYY”, highlight the corresponding row and press the Delete key.

Change the Company name for Customer ID “WOLZA” to “Wolski Zajazd”.
To save the changed records to the Database click the **Save Changes** button. A message box appears saying "Changes Recorded Successfully into the Customers table"

Click **OK** to close the message box.

To verify that the changes are saved, click **Get Data** button to re-populate all Customer, Order data into the DataGrid control.

Close **Form2**.

**Task 6– Using Stored Procedures**

In this task, you will make a call to a stored procedure using a SqlDataAdapter object.

To switch **Form2** to designer mode, select the **View | Designer** menu command.

To open the **Toolbox**, select **View | Toolbox** menu command.

Drag a **Button** from the toolbox and drop it on to the Form2 as shown in Figure 8.14 below.

Right-click the button and select Properties.

Set the following properties of the button control using the **Properties** window:

- **Name** - "btnSP"
- **Text** - "Call SP"

To view the DataGrid control properties, right-click the DataGrid control and select **Properties** menu command.

Clear the DataSource property of DataGrid control
• Double-click the Call SP button control.

• Add the following highlighted code in the btnSP_Click method:

```vbnet
Private Sub btnSP_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnSP.Click
    Try
        Dim myDataAdapter As SqlDataAdapter = New SqlDataAdapter("CustOrderHist", mySqlConnection)
        myDataAdapter.SelectCommand.CommandType = CommandType.StoredProcedure
        myDataAdapter.SelectCommand.Parameters.Add(New SqlParameter("@CustomerID", SqlDbType.NVarChar, 5))
        myDataAdapter.SelectCommand.Parameters("@CustomerID").Value = "ALFKI"
        Dim ds As DataSet = New DataSet
        myDataAdapter.Fill(ds, "Products")
        DataGrid1.DataSource = ds.Tables("Products").DefaultView
    Catch ex As Exception
        MessageBox.Show("Error occurred: " + ex.Message)
    End Try
End Sub
```

• This code sample retrieves the order history for the specified customer ID by invoking the CustOrderHist stored procedure.

• To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.

• Click Call SP. Customer order history for the customer ID ‘ALFKI’ will be displayed in the DataGrid control.

• Close Form2.

**Exercise 4 – Loading native ADO Recordset into a DataSet**

In this exercise you will use the “classic” ADO Recordset object with the ADO.NET Dataset. This will be done using the OLEDB data provider’s overloaded OleDbDataAdapter.Fill method, which accepts an ADO Recordset object.

**Task 1 – Adding Reference to Microsoft ActiveX Data Objects**

• In order to use the ADO from a .NET application, a reference to the Microsoft ActiveX Data Object type library is required. To set this reference, open the Solution Explorer using the View | Solution Explorer menu command.

• Right-click the References node in Solution Explorer and select Add Reference. The Add Reference dialog box appears.

• Switch to the COM tab in the Add Reference dialog box.

• Select Microsoft ActiveX Data Objects 2.7 Library in the Component Name list.
• Click the Select button to add a reference to the selected type library.
• Click OK to close the Add Reference dialog box.

Task 2 – Loading native ADO Recordset into a DataSet

• To open the Solution Explorer, select View | Solution Explorer menu command.
• Right-click project (Lab8-VB) in the Solution Explorer and select Add | Add Windows Form…
• To add Form3.vb to the current project, click Open.
• Right-click Form3.vb in the Solution Explorer and select View Code.
• At the top section of your code, add the following highlighted code:

```vbnet
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.OleDb
Imports ADODB
Public Class Form3
  Inherits System.Windows.Forms.Form
```

• To switch Form3 to designer mode, select the View | Designer menu command.
• To open the Toolbox, select View | Toolbox menu command.
• Drag a DataGrid control from the toolbox and drop it on the design area of the Form3.
• Drag a Button from the toolbox and drop it on the design area of the Form3.
• Right-click the button and select Properties.
• Set the following properties of the button control using the Properties window:
  
  **Name**  - “btnLoadRS”
  **Text**  - “Load Recordset”

• Expand the width of the button control to fit the text.
Double-click the **Load Recordset** button control.

Add the following highlighted code in the **btnLoadRS_Click** method

```vbnet
Private Sub btnLoadRS_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnRS.Click
    Try
        Dim rsObj As RecordsetClass = New RecordsetClass
        Dim constr As String = "provider=sqloledb;server=localhost;database=Northwind;trusted_connection=yes"
        rsObj.Open("Customers", constr, ADODB.CursorTypeEnum.adOpenForwardOnly, ADODB.LockTypeEnum.adLockReadOnly, 2)
        Dim myDataSet As DataSet = New DataSet
        Dim adapter As OleDbDataAdapter = New OleDbDataAdapter
        adapter.Fill(myDataSet, rsObj, "Customers")
        DataGrid1.DataSource = myDataSet.Tables("Customers").DefaultView
    Catch ex As Exception
        MessageBox.Show("Error occurred while retrieving the data: " + ex.Message)
    End Try
End Sub
```

To set **Form3** as Startup object

- Select **Project | Lab8-VB Properties** menu command. **Lab8-VB Property Pages** dialog box appears.
- Expand **Common Properties** and select **General**.
- Set **Form3** as the Startup object.
Click OK to close the Lab8-VB Property Pages dialog box.

- To compile and run your application select Debug | Start Without Debugging menu command or press CTRL+F5.
- To load customer data in the DataGrid control, click the Load Recordset button.
- Close Form3.
- To close the Visual Studio solution, select File | Close Solution menu command.

Lab Summary
In this lab you have performed the following exercises:

- Understand ADO.NET
- Work with the DataSet and SqlDataAdapter objects
- Work with the SqlDataReader and SqlCommand objects
- Insert, update and delete records using a DataSet
- Load a native ADO Recordset into a DataSet