Unit 5. Database access using ADO.NET

5.1. Visual Database Tools
5.2. ADO .NET Object Model
5.3. ADO .NET Programming

5.1 Visual Database Tools

- It helps us to us to create and maintain databases and data-driven applications.
- It is possible to access Microsoft Visual Database Tools from the IDE (Integrated Development Environment).

Visual database tools provide 3 designers to create these objects:

i. Database diagram
ii. Table designer
iii. Query and View designer

Database Diagram

- Database diagram tool allows user to design and visualize a database to which we are connected.
- This diagram contains the tables along with the relationship and primary key.

Table designer

- It provides an entire window to design the individual table.
- It also allow user to change an existing database by changing, adding, duplicating and deleting tables.
- It is used to add or remove columns, specify its data type, sine, description, keys, constraints and relationships.

Query and View designer

- It helps user to create and maintain the data retrieval and data manipulation.
- When user designs any query, view, functions, or stored procedure, the designer is made up of panes.

Evolution Of ADO.NET

- DAO (Data Access Object) was the first data access model created for local database. It consists of built-in Jet engine which had performance and functionality issues.
- Then RDO (Remote Data Object) and ADO (Active Data Object) came. They were designed for Client Server architectures but soon ADO took over RDO.
- ADO was a good architecture, but all the data stored in a recordset object which creates a problems when implemented on the network. ADO was a connected data access, which means that when a connection to the database is established the connection remains open until the application is closed. So, if the connection is open for the lifetime of application, it raises issues like database security, network traffic and productivity.
- To handle the problems above. ADO.NET came into existance

ADO.NET

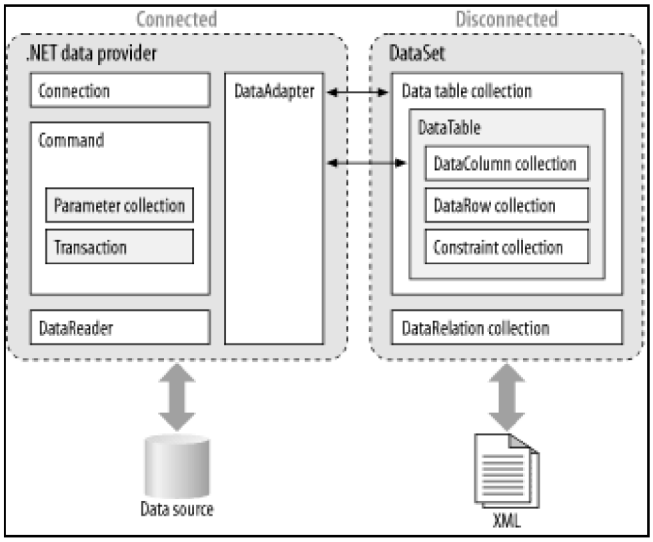
- ADO.NET(ActiveX Data Objects) is a database technology of .NET Framework used to connect **application system (Frontend Application) and database server (Backend Application).**
- It is a part of the .NET Framework.
- It consists of a **set of classes used to handle data access**.
- It uses **XML** to store and transfer data among applications, which is not only an industry standard but also provide fast access of data for desktop and distributed applications.
- It works with **disconnected architecture.**

5.2 ADO .NET Object Model

Connected & Disconnected Data (Architecture)

- The data access with ADO.NET consists of two parts;
  (1) Data Provider
  (2) DataSet

The following figure shows the connected disconnected architecture of ADO.NET.



(1) Data Provider

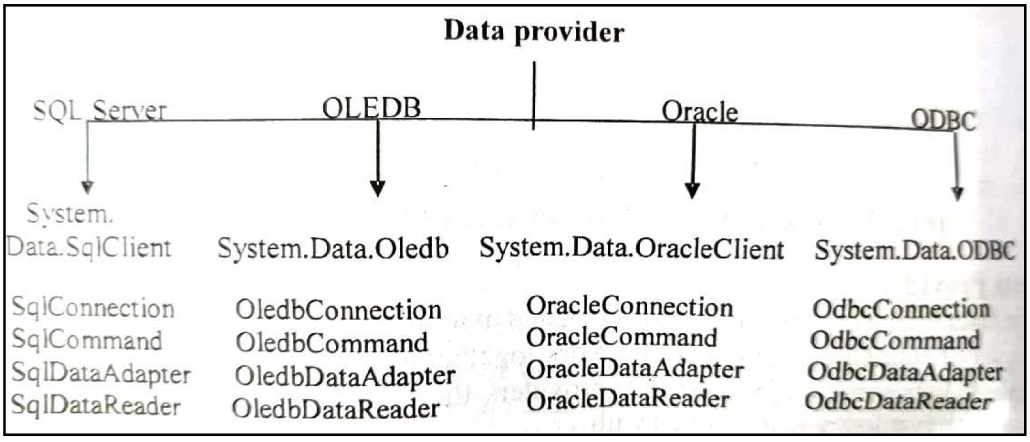
- The Data Provider is used for providing and maintaining the connection to the database.
- Data provider is used to connect to the database, execute commands and retrieve the record.
- We can use following data provider in ADO.Net.

Data Provider	DataSource Description
SQL Server	It is interacting with Microsoft SQL Server. It provides data access for Microsoft SQL Server. It requires the **System.Data.SqlClient** namespace.
OLE DB	Data Sources that expose an OleDb interface, i.e. Access or Excel. It is used to connect with OLE DB. It requires the **System.Data.OleDb** namespace.
ODBC	Data Sources with an ODBC interface. Normally older databases. It is used to connect to data sources by using ODBC. It requires the **System.Data.Odbc** namespace.
Oracle	It is used for Oracle Data Bases. It is used for Oracle data sources. It uses the **System.Data.OracleClient** namespace.

Following are the core object of Data Providers.

Object	Description
Connection	It is used to establish a connection to a specific data source.
Command	It is used to execute queries to perform database operations.
DataReader	It is used to read data from data source.
DataAdapter	It populates a DataSet and resolves updates with the data source. It is interface between dataset and database.

Data Providers & Namespaces available in ADO.NET



(1) Connection Object

- To establish connection with a database, you must have a connection object.
- The connection object helps to identify the database server, the database name, user name, password, and other parameters that are required for connecting to the database.

Connection String: A string that specifies information about a data source and the means of connecting to it is called Connection String.

It is passed in code to an underlying driver or provider in order to initiate the connection.

Parameters of Connection String:

✓ AttachDBFileName /Initial FileName

✓ ConnectTimeOut/Connection TimeOut

✓ DataSource/Server/Address/Addr/Network address

✓ Initial catalog/database

✓ Trusted connection/integrated security

✓ Persist security info.

Note: In this Unit all examples are with respect to SQLEXPRESS(In-Built Database of Visual Studio) as a Database.

Example(SQLEXPRESS):

```
Data Source=.\SQLEXPRESS;
AttachDbFilename="H:\ 4th sem\VB.NET\Projects\DBteacher.mdf";
Integrated Security=True;
Connect Timeout=30;
User Instance=True;
```

Example:

Server= myServerAddress; Database=myDataBase; User ID=myUsername: Password= myPassword; Trusted Connection=False;

Example

Data Source=myServerAddress; Initial Catalog=myDataBase; User Id=myUsername: Password=myPassword;

To create SqlConnection Object:

```
Dim con As New SqlConnection
Con.ConnectionString="Connection String"
```

Properties

ConnectionString	It stores the connection string that is passed to the connection object at the time of creating its object
Database	It stores the name of the database to which you need to connect
State	It returns the state of the connection i.e **IsClose or IsOpen**
ConnectionTimeOut	Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.

Methods

Open	It opens the connection
Close	It closes the connection
BeginTransaction	It creates the transaction object

(2) Command Object

- It is used to retrieve a subset of data.
- It uses the connection object to **execute SQL queries(**insert, Update and Delete**)**.
- The queries can be in the Form of **Inline text, Stored Procedures or direct Table access.**
- If a select query is issued, the result set it returns is usually stored in either a **DataSet or a DataReader** object.

Properties

Property	**Description**
Connection	To set connection object
CommandText	It specifies SQL Statement or the name of the Stored Procedure.
CommandType	This property indicates how the CommandText property should be interpreted. The possible values are: • Text (T-SQL Statement) • StoredProcedure (Stored Procedure Name) • TableDirect
ConnectionTimeOut	Gets the time to wait while trying to establish a connection before terminating the attempt and generating an error.

Methods

Method	**Description**
ExecuteNonQuery	Used to execute an SQL statement and **returns the no. of rows affected** by the given statement. Return type of this method is **int**
ExecuteScalar	Used to execute an SQL statement and **return a single value**.
ExecuteReader	Used to execute a select a statement and return the rows returned by the select statement as a DataReader.

Example

```
Dim conn As New SqlConnection("Connection String")
Dim cmd As New SqlCommand
cmd.Connection=conn
cmd.CommandText="Select Count(*) from Emp"
Dim EmpCnt As Integer
conn.Open()
EmpCnt=Convert.ToInt32(cmd.ExecuteScalar())
conn.Close()
MessageBox.Show("Total Employee=" + EmpCnt.ToString())
```

(3) DataReader Object

- A SqlDataReader is used to read data in the most efficient manner.
- You cannot use it for writing data.
- You can read forward-only and in sequential manner from SqlDataReader.
- Once you have read some data, you must save it because you will not be able to go back and read it again.
- A single row of data is in the memory at a time.
- It is an alternative to the Dataset and DataAdapter combination.

Properties

Property	Description
FieldCount	It is used to get the number of columns in the current row.
HasRows	It is used to get a value that indicates whether the SqlDataReader contains one or more rows.
IsClosed	It is used to retrieve a boolean value that indicates whether the specified SqlDataReader instance has been closed
Item	It is used to get the value of the specified column
RecordsAffected	It is used to get the number of rows changed, inserted or deleted by execution of the Transact-SQL statement

Methods

Method	Description
Read()	It is used to read record from the SQL Server database
Close()	It is used to closes the SqlDataReader object
GetName(index)	It is used to get the name of the specified column
GetSchemaTable()	It is used to get a DataTable that describes the columnmetadata of the SqlDataReader
GetValue(Index)	It is used to get the value of the specified column
GetValues	It is used to populate an array of objects
NextResult()	It is used to get the next result, when reading theresults of SQL statements

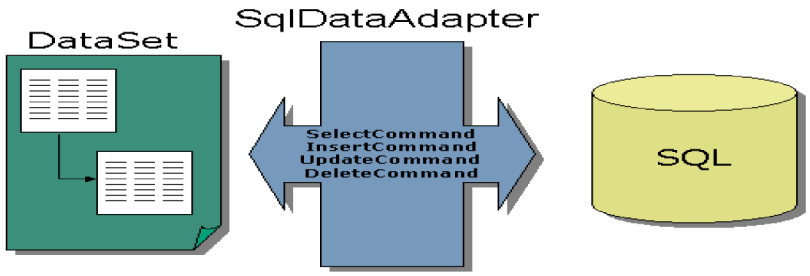
Example

```
Imports System
Imports System.Data
Imports System.Data.SqlClient
Dim conn As New SqlConnection("Connection String”)
Dim cmd As New SqlCommand
Dim dr As SqlDataReader
cmd.Connection=conn
cmd.CommandText="Select Eid, Ename from Emp"
conn.Open()
dr=cmd.ExecuteReader()
while (dr.Read())
      MessageBox.Show(dr.GetInt32("Eid").ToString +""+ dr.GetString("Ename"))
Wend
dr.Close()
conn.Close()
```

Note: Here note that the object of DataReader is not created directly with New keyword. The Object is created with the ExecuteReader method of the Command Object.

(4) DataAdapter Object

- It acts as a bridge between Database and the Dataset.
- It is associated Command and Connection objects.



Properties

Property	Description
SelectCommand	Used to hold a command that retrieve data from the data source.
InsertCommand	Used to hold a command that insert data into the data source.
UpdateCommand	Used to hold a command that updates data to the data source.
DeleteCommand	Used to hold a command that delete data from the data source.
CommandType	This property indicates how the CommandText property should be interpreted. The possible values are: • Text (T-SQL Statement) • StoredProcedure (Stored Procedure Name) • TableDirect

Methods

Method	Description
Fill	It is used to populate a dataset object with the data that the DataAdapter object retrieve from the data and store using its SelectCommand.
Update	It is used to update the database according to the changes that are made in the Dataset

(2) DataSet

- It is **disconnected architecture** that means there is no need to active connections during work with dataset.
- It is a collection of DataTables and Relation between tables.
- It can be considered as a local copy of the portions of the database.
- The DataSet is persisted in memory and it can be manipulated and updated independent of the database.
- When the use of this DataSet is finished, changes can be made back to the central database for updating.
- The DataSet class resides in the System.Data namespace.

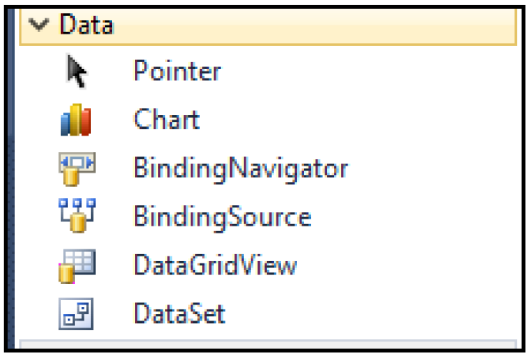
Example

Button_Click()

```
Dim conn As New SqlConnection("Connection String")
Dim da As New SqlDataAdapter
Dim cmd As New SqlCommand
cmd.CommandText = "Select from Emp"
cmd.Connection =conn
da.SelectCommand = cmd
Dim ds As New DataSet
da.Fill(ds, "Emp")
End Sub
```

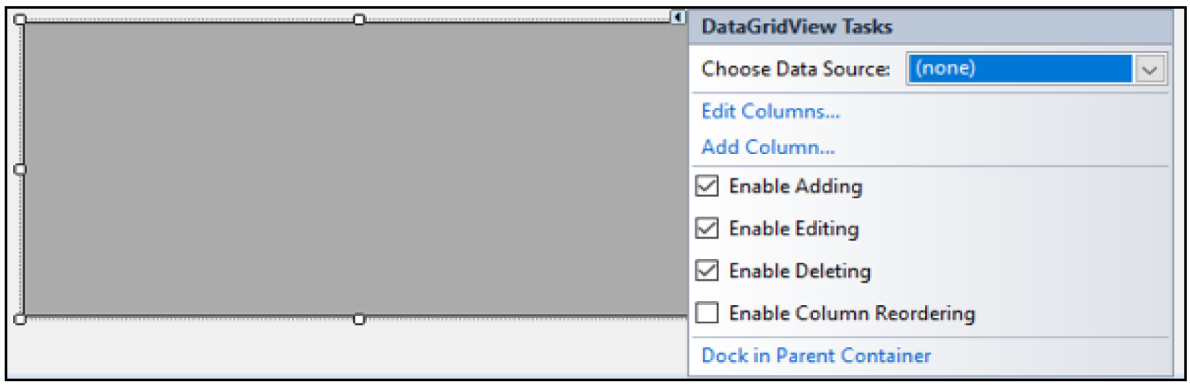
DataSet	DataReader
It is disconnected object and can provide access to data even when connection toDatabase was closed.	It is connected object and can not provide access to data when connection to databaseWas closed.
It can store data from multiple tables.	It can store data from only one table.
It allows insert, update and delete on data.	It is read only and it doesn't allow insert,Update and delete on data.
It allows navigation between record eitherForward or backward.	It allows only forward navigation that alsoOnly to immediate next record.
It can contain multiple records.	It can contain only one record at a time.
All the data of a dataset will be on client	All the data of a DataReader will be onserver and one record at a time is retrieved and stored in DataReader when you call theRead() method of datareader.

Data Controls



❖ DataGridView

- It displays data in a customizable grid.
- The DataGridView class allows customization of cells, rows, columns, and borders through the use of properties.
- You can use a DataGridView control to display data with or without an underlying datasource.
- Without specifying a data source, you can create columns and rows that contain data and add them directly to the DataGridView.
- As an alternative to populating the control manually, you can set the DataSource and DataMember properties to bind the DataGridView to a data source and automatically populate it with data.



Properties

Property	Description
Allow User ToAddRows	It gets or sets a value indicating whether the option to add rows is displayed to the user
Allow User ToDelete Rows	It gets or sets a value indicating whether the user is allowed to delete rows from the DataGridView.
Allow User ToOrderColumns	It gets or sets a value indicating whether manual column repositioning is enabled.
Allow User ToResizeColumns	It gets or sets a value indicating whether users can resize columns.
Allow User ToResizeRows	It gets or sets a value indicating whether users can resize rows.
ColumnCount	It gets or sets the number of columns displayed in the DataGridView.
Columns	It gets a collection that contains all the columns in the control.
Controls	It gets the collection of controls.
CurrentCell	It gets or sets the currently active cell.
CurrentRow	It gets the row containing the current cell.
DataBindings	It gets the data bindings for the control.
DataMember	It gets or sets the name of the list or table in the data source for which the DataGridView is displaying data.
DataSource	It gets or sets the data source that the DataGridView is displaying data.
Item	It gets or sets the cell located at the intersection of the column and row with the specified indexes.
MultiSelect	It gets or sets a value indicating whether the user is allowed to select more that one cell, row or column of the DataGridView at a time.
RowCount	It gets or sets the number of rows displayed in the DataGridView.
Rows	It gets a collection that contains all the rows in the DataGridView control.

Methods:

Property	Description
ClearSelection	It clears the current selection by unselecting all selected cells
Sort	It sorts the contents of the DataGridView control in ascending or descending order based on the contents of the specified column.
Update	It causes the control to redraw the invalidated regions within its client area.
UpdateCellErrorText	It forces the cell at the specified location to update its error text.
UpdateCellValue	It forces the control to update its display of the cell at the specified location based on its new value, applying any automatic sizing modes currently in effect.
GetCellCount	It gets the number of cells that satisfy the provided filter.
Show	It displays the control to the user.
DisplayedRowCount	It returns the number of rows displayed to the user.
Select	It activates the control.
SelectAll	It selects all the cells in the DataGridview.

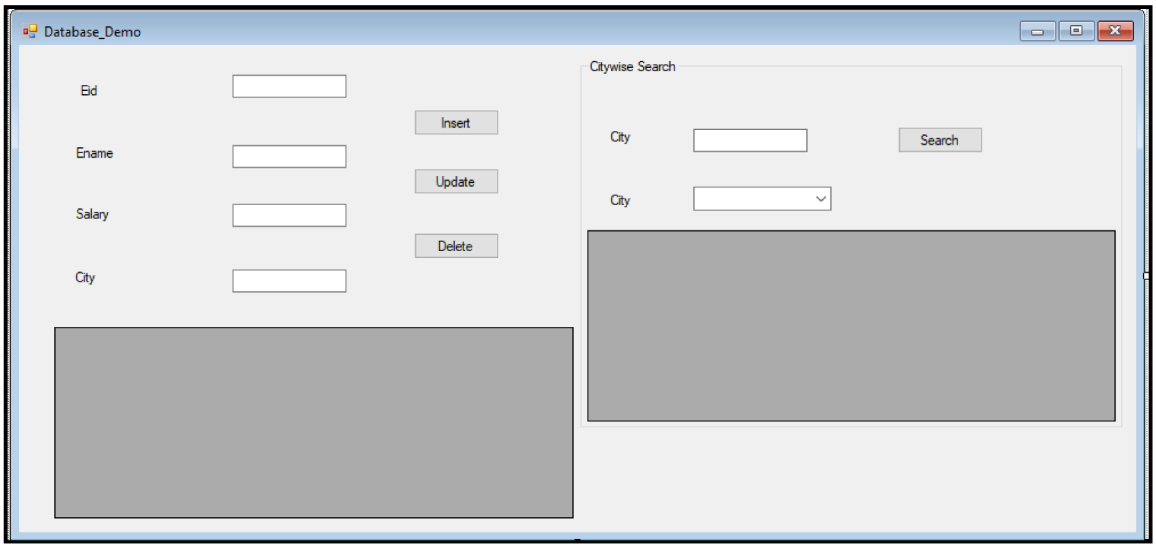
✓ Default Event: CellContentClick()

Database Connection Example(Functionality: Insert, Update, Delete, Show Data in DataGridView, Search):

Steps:

1. Design Form As Per Requirement.
2. Add Database
   - ➔ Right click on Project Name From **Solution Explorer** → Add → Add New Item→Select ServiceBased Database(Database1.mdf) → If you want Change Name of Database than can Change→click on Add Button
3. Create Table From Server Explorer.
   - ➔ Click on Database Name (Which you want Create as above) from **Server Explorer**→Right Click on **Tables option→ Add New Table**→Add Column Names With Data types as per requirement(Eid,Ename,Salary,City)→Give Primarykey to Eid(Right click on field Eid→ click on set primary key option) Save it(Ctrl+S)→Give Proper Table Name(Emp)→OK
4. **Connection String:**
   - ➔ Right click on Database Name from Server Explorer→ Click on Properties→Property window will be open(Right side of the window)→Select Value of ConnectionString Property carefully → Copy it(Ctrl+C) → Used this connection when needed.

Form Design:



Coding:

```
Imports System.Data.SqlClient

    Dim con As New SqlConnection
    Dim cmd As New SqlCommand
```

Load Event of Form:

```
Try
        con.ConnectionString = "Connection String will be paste(step mention as above)"
        cmd.Connection = con
Catch ex As Exception
      MsgBox(ex.Message)
End Try
      display_record()
End Sub
```

```
Sub display_record()
       Try
          cmd.CommandText = "select * from Emp"
          Dim da As New SqlDataAdapter(cmd)
          Dim dt As New DataTable
          da.Fill(dt)
          DataGridView1.DataSource = dt
       Catch ex As Exception
          MsgBox(ex.Message)
       End Try
      Fill_combo()
End Sub
```

```
Sub clear_control()
       TextBox1.Text = ""
       TextBox2.Text = ""
       TextBox3.Text = ""
       TextBox4.Text = ""
End Sub
```

Insert_Button() Click Event:

```
Try
  con.Open()
   cmd.CommandText = "insert into Emp values('" & TextBox1.Text & "','" & TextBox2.Text & "'," &
   TextBox3.Text & ",'" & TextBox4.Text & "')"
  cmd.ExecuteNonQuery()
  con.Close()
  MsgBox("inserted Successfully")
Catch ex As Exception
  MsgBox(ex.Message)
End Try
  display_record()
  clear_control()
```

Update_Button() Click Event:

```
Try
  con.Open()
  cmd.CommandText = "update Emp set Ename='" & TextBox2.Text & "',Salary=" & TextBox3.Text &
   ",City='" & TextBox4.Text & "' where  Eid='" & TextBox1.Text & "'"
  cmd.ExecuteNonQuery()
  con.Close()
  MsgBox("Update Successfully")
 Catch ex As Exception
  MsgBox(ex.Message)
 End Try
  display_record()
  clear_control()
```

Delete_Button() Click Event:

```
Try
  con.Open()
  cmd.CommandText = "delete from Emp where Eid='" & TextBox1.Text & "'"
  cmd.ExecuteNonQuery()
  con.Close()
  MsgBox("Deleted Successfully")
Catch ex As Exception
  MsgBox(ex.Message)
End Try
  display_record()
  clear_control()
```

DataGridView1_CellClick Event:

```
TextBox1.Text = DataGridView1.Rows(e.RowIndex).Cells(0).Value
TextBox2.Text = DataGridView1.Rows(e.RowIndex).Cells(1).Value
TextBox3.Text = DataGridView1.Rows(e.RowIndex).Cells(2).Value
TextBox4.Text = DataGridView1.Rows(e.RowIndex).Cells(3).Value
```

Search_Button_Click() Event:

```
Try
        cmd.CommandText = "select * from Emp where City='" + TextBox5.Text + "'"
        Dim dt As New DataTable
        Dim da As New SqlDataAdapter(cmd)
        da.Fill(dt)
        DataGridView2.DataSource = dt
Catch ex As Exception
        MsgBox(ex)
End Try
```

Combobox_SelectedIndexChanged Event:

```
Try
        cmd.CommandText = "select * from Emp where City='" + ComboBox1.SelectedItem + "'"
        Dim dt As New DataTable
        Dim da As New SqlDataAdapter(cmd)
        da.Fill(dt)
        DataGridView2.DataSource = dt
Catch ex As Exception
        MsgBox(ex)
End Try
```

```
Sub Fill_combo()
      Try
        cmd.CommandText = "select * from Emp"
        Dim dt As New DataTable
        Dim da As New SqlDataAdapter(cmd)
        da.Fill(dt)
        DataGridView1.DataSource = dt
        For Each a As DataRow In dt.Rows
          Dim s As String = a(3).ToString
          If Not ComboBox1.Items.Contains(s) Then
            ComboBox1.Items.Add(s)
          End If
        Next
      Catch ex As Exception
        MsgBox(ex)
      End Try
 End Sub
```
