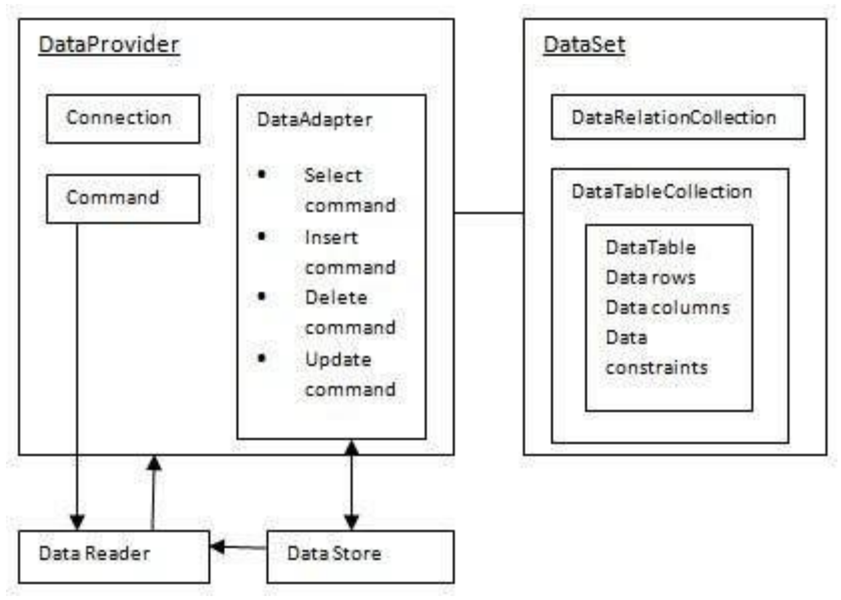
❖ ADO.NET

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

The following figure shows the ADO.NET objects at a glance:



The DataSet Class

- The dataset represents a subset of the database. It does not have a continuous connection to the database.
- To update the database a reconnection is required. The DataSet contains DataTable objects and DataRelation objects.
- The DataRelation objects represent the relationship between two tables.

Data Providers

- Data provider is used to connect to the database, execute commands and retrieve the record.
- It is lightweight component with better performance. It also allows us to place the data into DataSet to use it further in our application.

The .NET Framework provides the following data providers that we can use in our application.

.NET Framework data provider	Description
.NET Framework Data Provider for SQL Server	It provides data access for Microsoft SQL Server. It requires the **System.Data.SqlClient** namespace.
.NET Framework Data Provider for OLE DB	It is used to connect with OLE DB. It requires the **System.Data.OleDb** namespace.
.NET Framework Data Provider for ODBC	It is used to connect to data sources by using ODBC. It requires the **System.Data.Odbc** namespace.
.NET Framework Data Provider for Oracle	It is used for Oracle data sources. It uses the **System.Data.OracleClient** namespace.
.NET Framework Data Provider for SQL Server Compact 4.0.	It provides data access for Microsoft SQL Server Compact 4.0. It requires the **System.Data.SqlServerCe** namespace.

Data Providers Objects

- 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. The DbDataReader is a base class for all DataReader objects.
DataAdapter	The base class for all DataAdapter objects is the DbDataAdapter It populates a DataSet and resolves updates with the data source class.

Data Provider for SQL Server

Data provider for SQL Server is a lightweight component. It provides better performance because it directly access SQL Server without any middle connectivity layer. In early versions, it interacts with ODBC layer before connecting to the SQL Server that created performance issues.

The .NET Framework Data Provider for SQL Server classes is located in the **System.Data.SqlClient** namespace. We can include this namespace in our C# application by using the following syntax.

1. using System.Data.SqlClient;

This namespace contains the following important classes.

Class	Description
SqlConnection	It is used to create SQL Server connection. This class cannot be inherited.
SqlCommand	It is used to execute database queries. This class cannot be inherited.
SqlDataAdapter	It represents a set of data commands and a database connection that are used to fill the DataSet. This class cannot be inherited.
SqlDataReader	It is used to read rows from a SQL Server database. This class cannot be inherited.
SqlException	This class is used to throw SQL exceptions. It throws an exception when an error is occurred. This class cannot be inherited.

The DataAdapter Object

The DataAdapter object acts as a mediator between the DataSet object and the database. This helps the Dataset to contain data from multiple databases or other data source.

The DataReader Object

The DataReader object is an alternative to the DataSet and DataAdapter combination. This object provides a connection oriented access to the data records in the database. These objects are suitable for read-only access, such as populating a list and then breaking the connection.

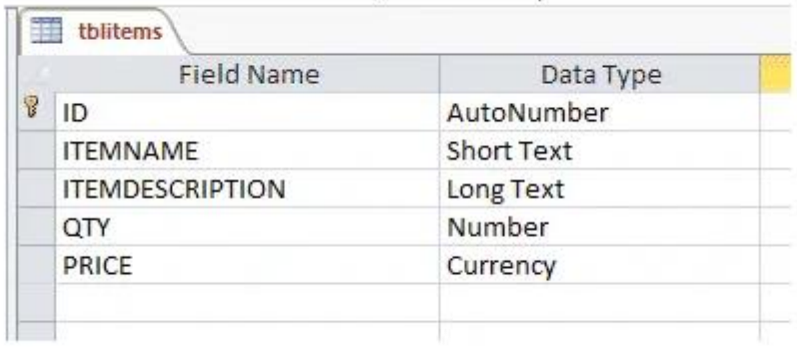
❖ Steps How to Connect Access Database in VB.Net

1. **Step 1: Create an MS Access Database.**

   Open an **MS Access Database** in your Computer and Create a **Blank Database** and Save it as ***"inventorydb.accdb"***.

2. **Step 2: Create a Database Table.**

   To create a table, follow the image below and save it as ***"tbliitems"***.

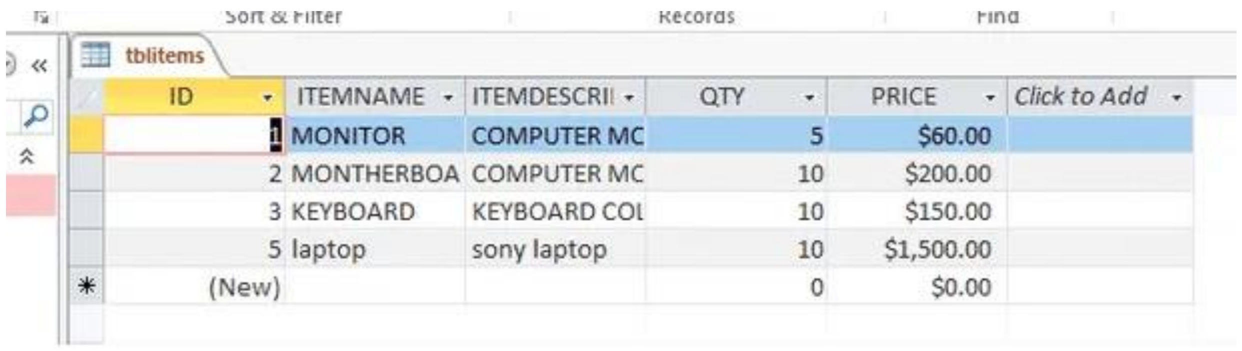


tblitems

Field Name	Data Type
ID	AutoNumber
ITEMNAME	Short Text
ITEMDESCRIPTION	Long Text
QTY	Number
PRICE	Currency

3. **Step 3: Populate the table.**

   Add sample records in the table. follow the sample records in the image below.



tblitems

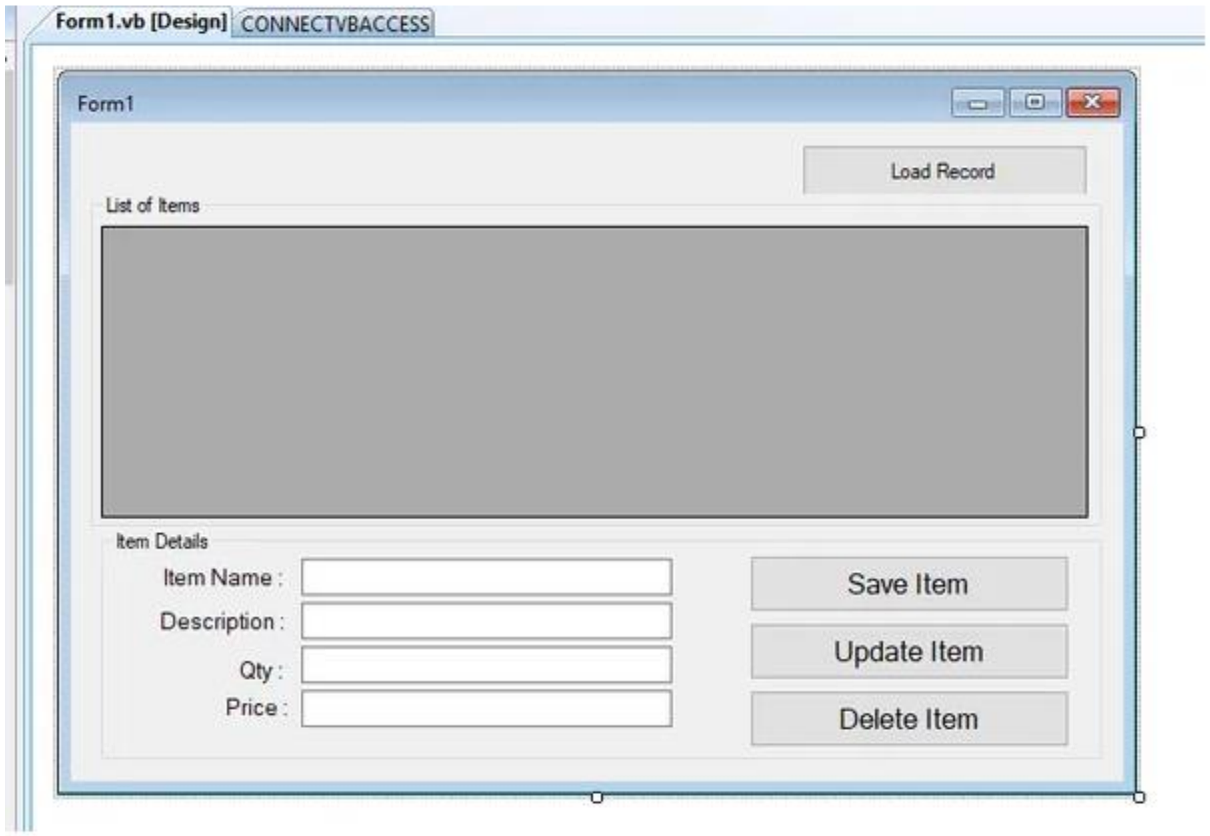
ID	ITEMNAME	ITEMDESCRII	QTY	PRICE	Click to Add
1	MONITOR	COMPUTER MC	5	$60.00	
2	MONTHERBOA	COMPUTER MC	10	$200.00	
3	KEYBOARD	KEYBOARD COI	10	$150.00	
5	laptop	sony laptop	10	$1,500.00	
(New)			0	$0.00	

4. **Step 4: Create a VB.Net Application.**

   Open Visual Studio and Create a Visual Basic Application project and Save it as ***"connectvbaccess"***.

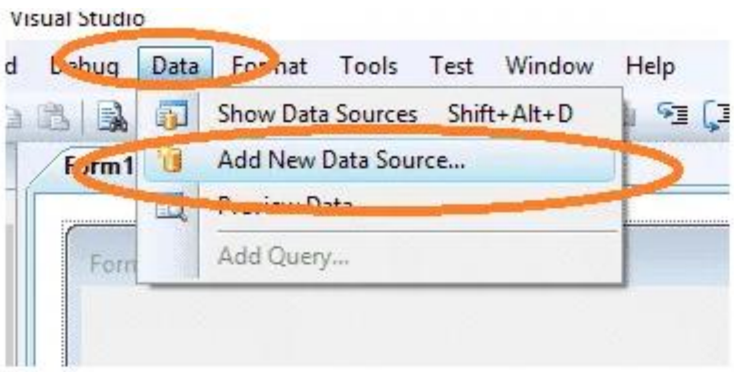
5. **Step 5: Design the user interface.**

   To design the form, you need to follow the image below.



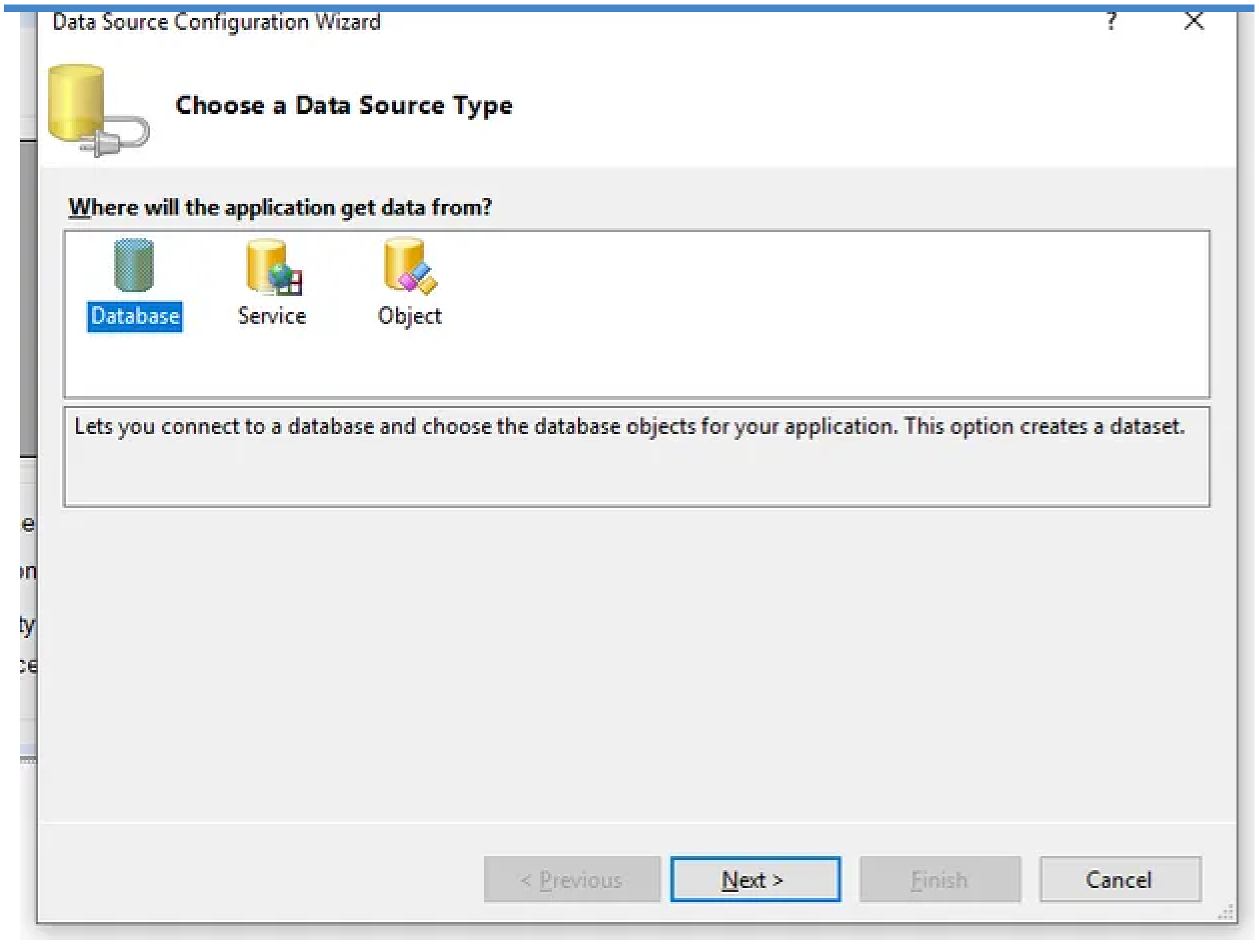
6. **Step 6: Add New Data Source.**

   On the menu, click data and select **"Add new Data Source.."**



7. **Step 7: Choose a Data Source Type**

Select **Database** and **click Next.**


Data Source Configuration Wizard

Choose a Data Source Type

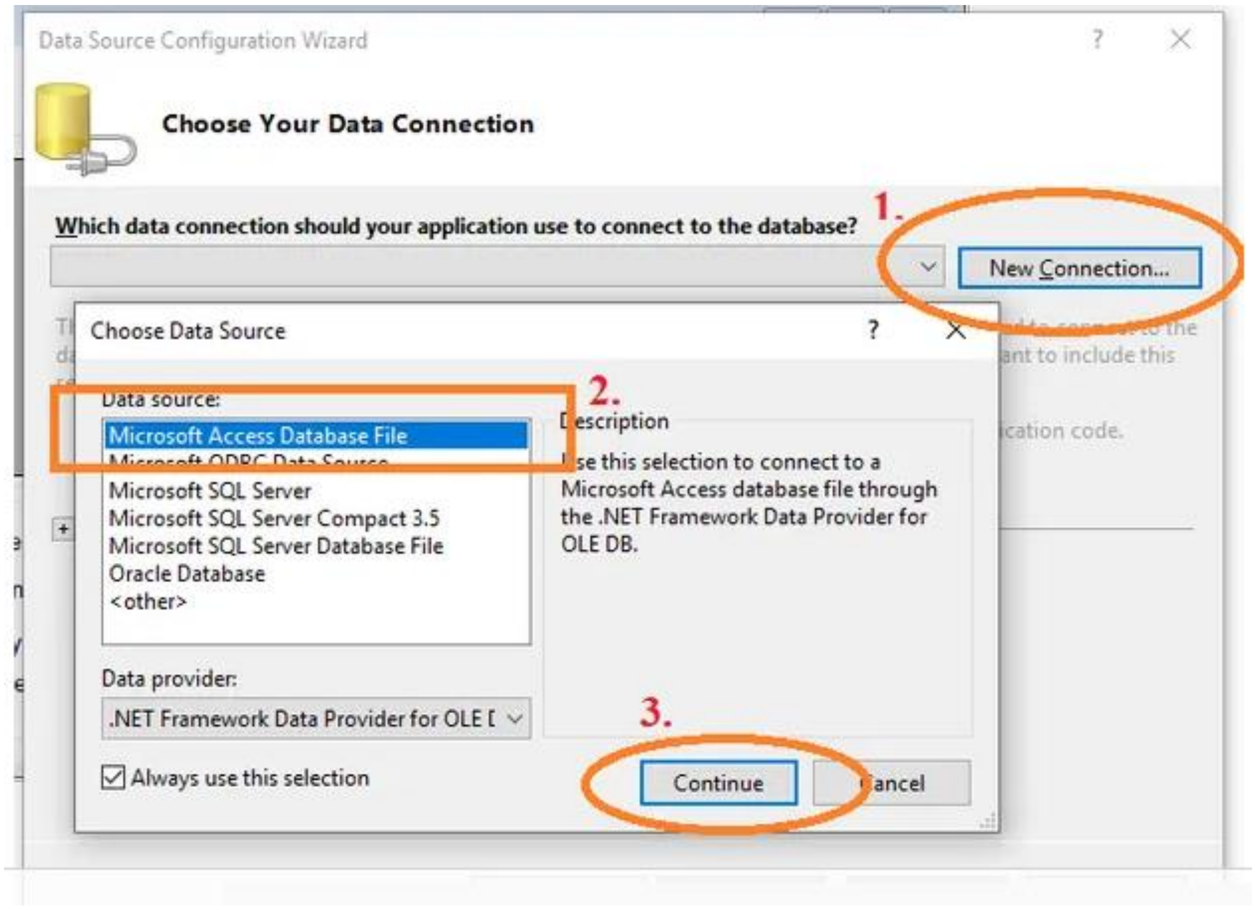
Where will the application get data from?

Lets you connect to a database and choose the database objects for your application. This option creates a dataset.

8. **Step 7: Choose a Data Source**

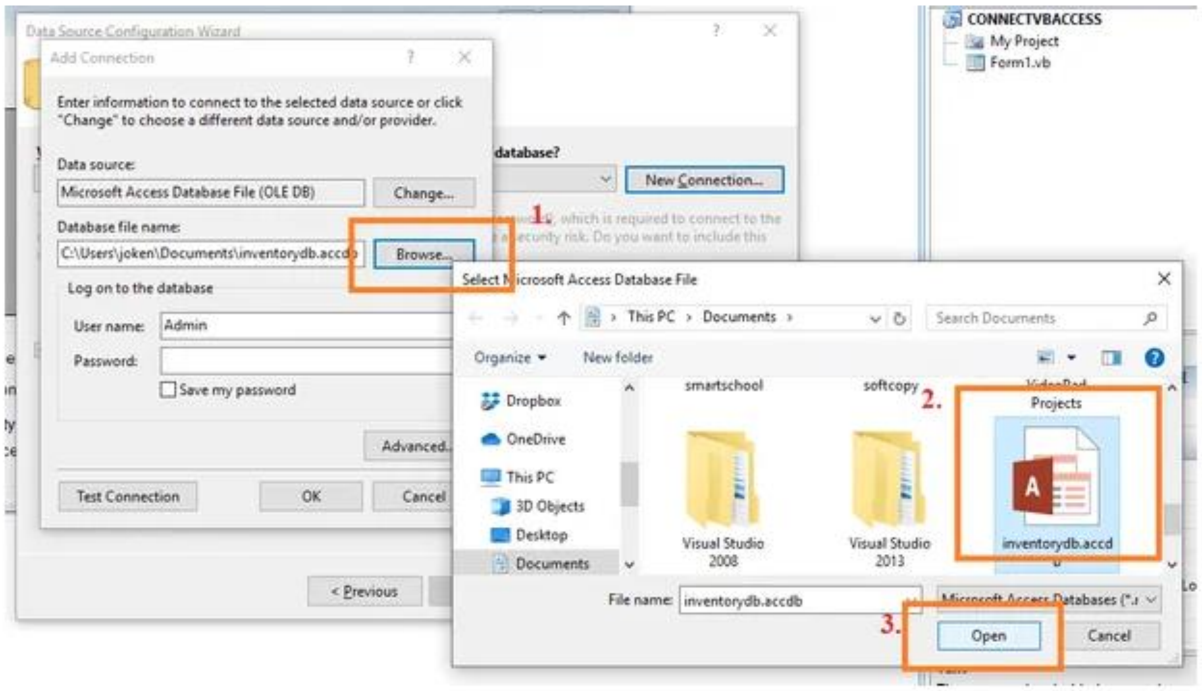
   Click **new Connection** then, Select **Microsoft AccessDatabase file** and,

   Click **Continue**. *Follow the image below.*


Data Source Configuration Wizard
Choose Your Data Connection
Which data connection should your application use to connect to the database?
1.
New Connection...
Choose Data Source
2.
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Compact 3.5
Microsoft SQL Server Database File
Oracle Database
<other>
Description
Use this selection to connect to a Microsoft Access database file through the .NET Framework Data Provider for OLE DB.
Data provider:
.NET Framework Data Provider for OLE D
Always use this selection
3.
Continue
Cancel

9. **Step 9: Add Connection**

First, click **Browse** Button then, Select “**inventorydb.accdb”**, Lastly, Click **Open**.



Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft Access Database File (OLE DB)

Database file name: C:\Users\joken\Documents\inventorydb.accdb

Log on to the database

User name: Admin

Password:

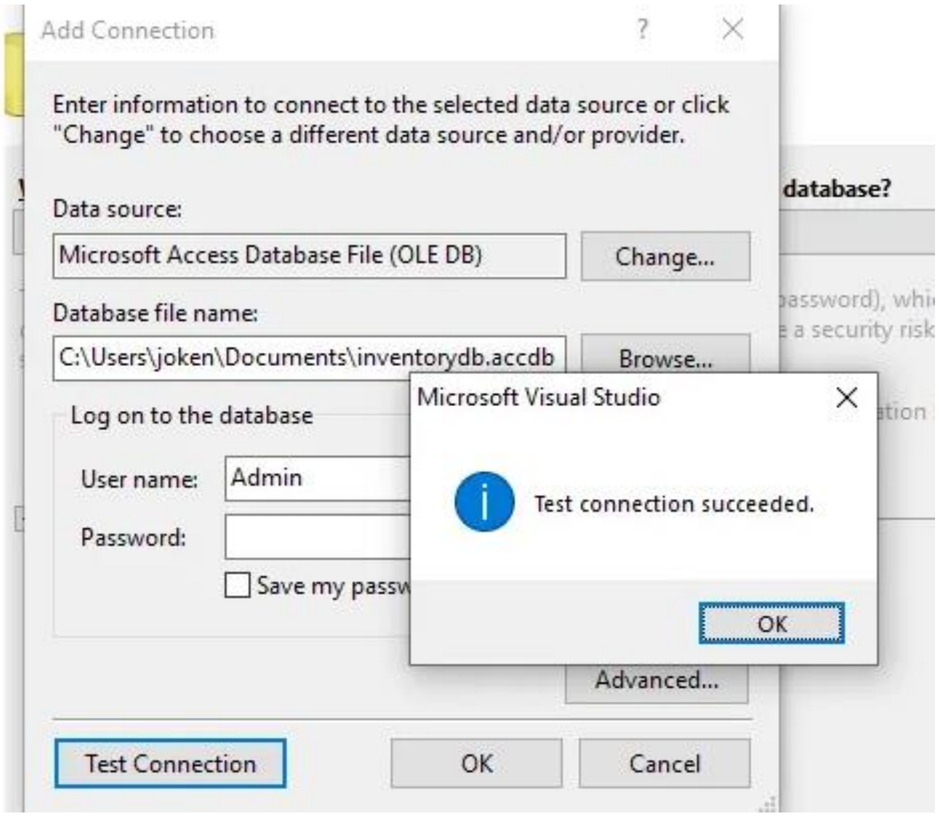
☐ Save my password

File name: inventorydb.accdb

Microsoft Access Databases (*.accdb)

10. **Step 10: Test Connection**

To test the connection, click the **“Test Connection”** button, finally click **“OK”** button at the side of the **“Cancel”** button.


Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft Access Database File (OLE DB)

Database file name: C:\Users\joken\Documents\inventorydb.accdb

Log on to the database

User name: Admin

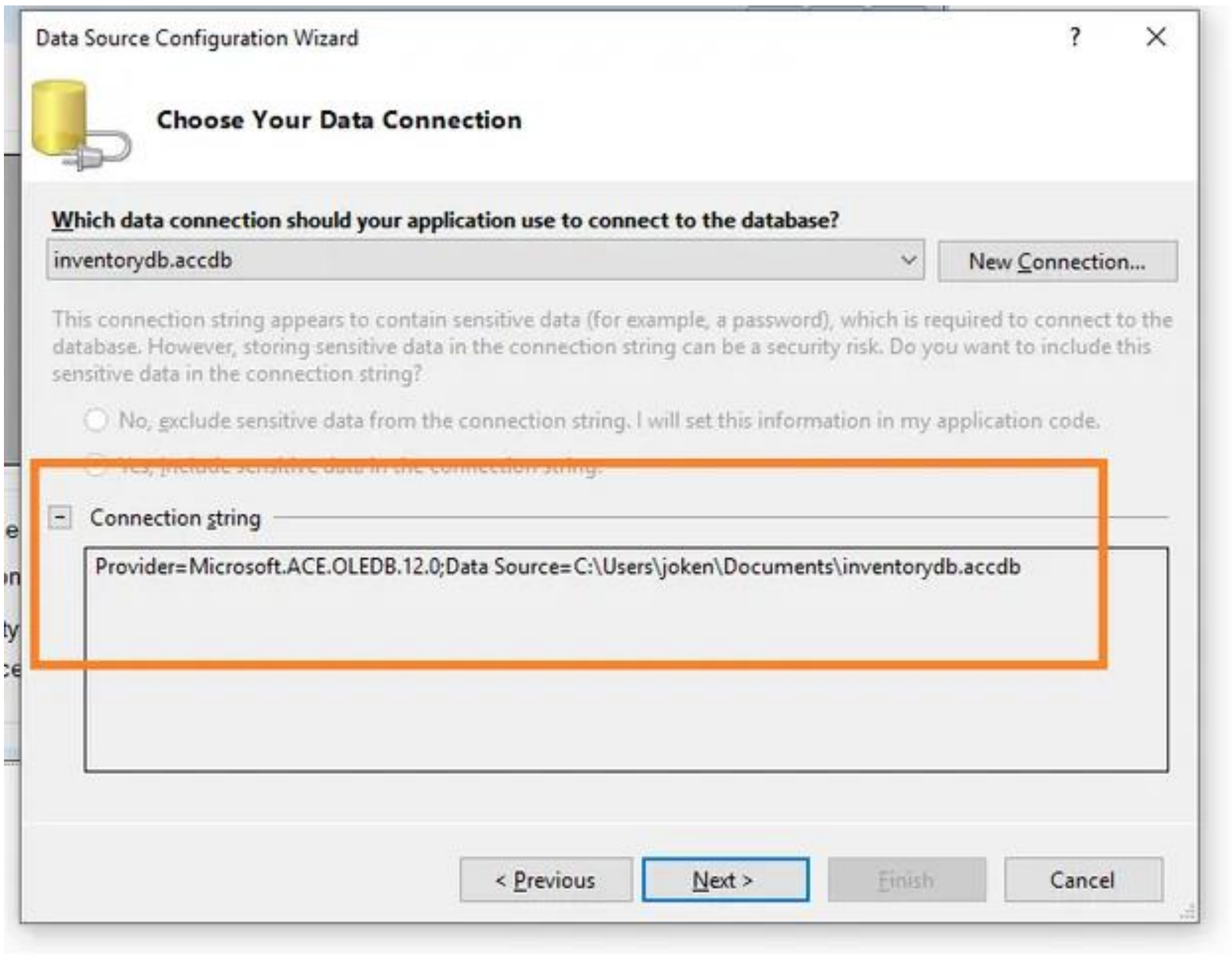
Password:

Microsoft Visual Studio

Test connection succeeded.

11. **Step 11: Copy the Connection String.**

Copy the connection string so that we can use this in our next step.



Choose Your Data Connection

Which data connection should your application use to connect to the database?

inventorydb.accdb

This connection string appears to contain sensitive data (for example, a password), which is required to connect to the database. However, storing sensitive data in the connection string can be a security risk. Do you want to include this sensitive data in the connection string?

No, exclude sensitive data from the connection string. I will set this information in my application code.

Yes, include sensitive data in the connection string.

Connection string

```
Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\Users\joken\Documents\inventorydb.accdb
```

12. **Step 12: Start Coding.**
    In this final step, we will now start adding functionality to our vb.net program by adding some functional codes.

❖ Code To Connect Access Database in VB.Net

Double the **"Form1"** and add the following code under **"Public Class Form1"**.

```
Dim con As New OleDb.OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\joken\Documents\inventorydb.accdb")
```

The Code above started with a **Declaration of Variable** name ***"con"*** with an **Ole Object Type OledbConnection**.

Inside **OledbConnection**, we pasted the connection string we copied from the **"Step 11"** instructions.

❖ Test the Connection of Access database in VB.Net

To test the **Connection between ms access database and VB.Net**, Double click the **form1** add the following code under "Form1_Load" events.

```
 Try
   con.Open()

   If con.State = ConnectionState.Open Then
      MsgBox("Connected")
   Else
      MsgBox("Not Connected!")

   End If
Catch ex As Exception
   MsgBox(ex.Message)
Finally
   con.Close()

End Try
```

- We use try-catch to the exceptions that may occur during runtime.
- open the connection
- check using if statement if the connection is open
- 'Display a message box if successfully connected or Not
- close the connection

```
Imports System.Data.OleDb
Public Class Form1
   Dim con As New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\Administrator\Documents\demo.accdb")
   Dim cmd As New OleDbCommand
   Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
      'TODO: This line of code loads data into the 'DemoDataSet.Table1' table. You can move, or remove it, as needed.
      con.Open()
      Me.Table1TableAdapter.Fill(Me.DemoDataSet.Table1)
      cmd.Connection = con
   End Sub

   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
      Table1BindingSource.AddNew()
      cmd.CommandText = "insert into table1 values( '" & TextBox1.Text & "','" & TextBox2.Text & "')"
      cmd.ExecuteNonQuery()

   End Sub

   Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
      Table1BindingSource.EndEdit()
      Table1TableAdapter.Update(DemoDataSet.Table1)
      MsgBox("hi")
   End Sub

   Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
      Table1BindingSource.RemoveCurrent()
      If Me.DataGridView1.Rows.Count > 0 Then
         cmd.CommandText = "delete from table1 where name='" & TextBox1.Text & "'"
         cmd.ExecuteNonQuery()
         ' Me.Table1TableAdapter.Fill(Me.DemoDataSet.Table1)
      End If

   End Sub

   Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
      Table1BindingSource.MoveNext()
   End Sub

   Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
      Table1BindingSource.MovePrevious()
   End Sub
End Class
```
………...

```
Imports System.Data.OleDb
Public Class Form1
   Dim con As New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\Administrator\Documents\demo.accdb")
   Dim cmd As New OleDbCommand
   Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
      'TODO: This line of code loads data into the 'DemoDataSet.Table1' table. You can move, or remove it, as needed.
      con.Open()
      Me.Table1TableAdapter.Fill(Me.DemoDataSet.Table1)
      cmd.Connection = con
   End Sub

   Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
      Table1BindingSource.AddNew()
   End Sub

   Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
      Table1BindingSource.EndEdit()
      Table1TableAdapter.Update(DemoDataSet.Table1)
      cmd.CommandText = "insert into table1 values( '" & TextBox1.Text & "','" & TextBox2.Text & "')"
      cmd.ExecuteNonQuery()
      MsgBox("hi")
   End Sub

   Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
      Table1BindingSource.RemoveCurrent()
      If Me.DataGridView1.Rows.Count > 0 Then
         cmd.CommandText = "delete from table1 where name='" & TextBox1.Text & "'"
         cmd.ExecuteNonQuery()
         ' Me.Table1TableAdapter.Fill(Me.DemoDataSet.Table1)
      End If

   End Sub

   Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
      Table1BindingSource.MoveNext()
   End Sub

   Private Sub Button5_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click
      Table1BindingSource.MovePrevious()
   End Sub
End Class
```