Module 05 Notes

Data Driven Applications

All applications use data, but most also store that data in a persistent format. This is because data used in a program always runs from the computer’s memory, but since that data disappears once the application closes it needs to be saved somewhere else if it is to be used again.

In this module we will demonstrate how data is stored in a database, one of the most common places to store application data.

# Databases

Database is a generic term that covers a wide verity of storage structures. However, there are common traits in most of them.

1. Data is stored in a one-dimensional set called a Row or Tuple. These rows identify their members by either an index or a key (column).
2. Rows are stored in a two-dimensional set called a Table or Entity.

One of the most popular types of databases is the Structured Query Language (SQL) database. These databases use SQL to create and manages data. Microsoft’s Enterprise version of this is MS SQL Server, which we will be using in our course.

## Creating a Database

To create a database with SQL you use this simple syntax:

CREATE DATABASE MYDEMODB;

To focus your connection on the new database you use this command:

USE MYDEMODB;

TIP: You should use the MS SQL Server key work GO to force SQL to process a command immediately after is issued (without a semicolon.) For Example:

CREATE DATABASE MYDEMODB;

Go

USE MYDEMODB;

Go

## Creating a Table

To create a table in the database you use this command:

Create Table MyDemoTable (Col1 int, Col2 float, Col3 nVarchar(50));

This table will now exist as an empty two-dimensional array. The Column names are used to identify elements of the first dimension (row).

Managing a Table

You manage the table’s data using the standard SQL statements; Select, Insert, Update, and Delete.

Insert Into MyDemoTable (Col1, Col2, Col3) Values (1, 2.5, 'test data');

Go

Select Col1, Col2, Col3 From MyDemoTable;

Tip: For more on this topic see: <http://www.w3schools.com/sql/>

# Microsoft’s SQL Server LocalDB

Microsoft provides several free options for using MS SQL Server when developing applications. Visual Studio comes with a the option known as the **LocalDB**.

**NOTE:** LocalDB is a light version of SQL Express that should have been installed with VS. However, it sometimes is not installed or somehow fails to registers correctly. If LocalDB does not work, try installing it from <https://msdn.microsoft.com/en-us/sqlserver2014express.aspx>

“MicrosoftSQL Server 2014 Express**LocalDB** is an execution mode of SQL Server Express targeted to program developers. **LocalDB** installation copies a minimal set of files necessary to start the SQL Server Database Engine.” (<https://msdn.microsoft.com/en-us/library/hh510202(v=sql.120).aspx)>

More Information on LocalDB can be found at this link:

<https://blogs.msdn.microsoft.com/sqlexpress/2011/07/12/introducing-localdb-an-improved-sql-express/>

Instruction on how to use the LocalDB database engine to create a database and tables can be found here: <https://msdn.microsoft.com/library/ms233763(VS.110).aspx>

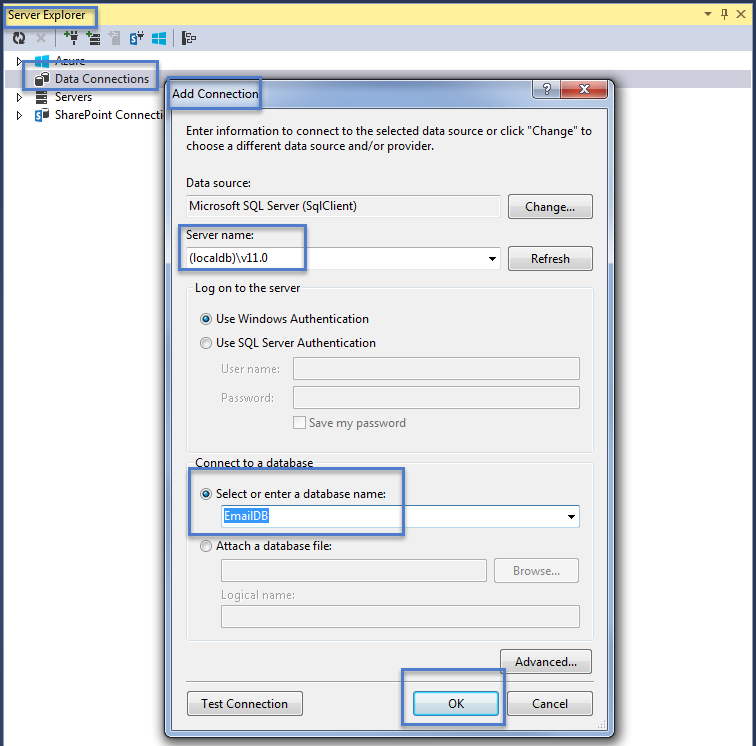
# **Demo**: Using to the LocalDB from Visual Studio

Your instructor will now demo how to connect to the LocalDB, create a database, create a table in that database, and add data to the table.

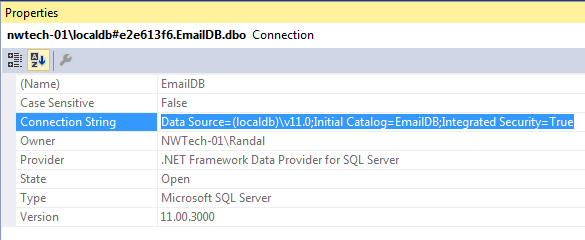
# Lab 05-1

In this lab, you will create a database and table using the LocalDB.

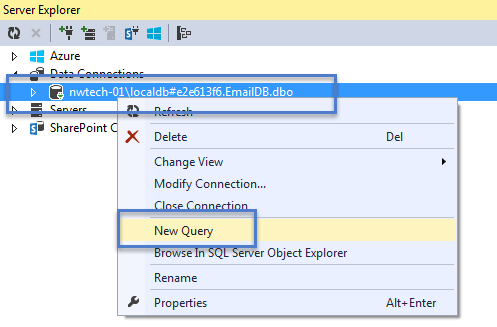
1. *Create a new Windows Form Application project in Visual Studio called Module05Labs.*
2. Create a new database called EmailsDB using the LocalDB option in Visual Studio.



1. Review the Properties of the new database and note the connection string.



3) Open a new Query window.



4) Run the following command in the query window and review the result.

SP\_HelpDB EmailsDB;

Go

5) Create an Emails table using this code:

CREATE TABLE Emails

(

[EmailID] INT NOT NULL PRIMARY KEY,

[EmailFullName] NVARCHAR(100) NOT NULL,

[EmailAddress] NVARCHAR(500) NOT NULL

)

Go

6) Add data using this SQL code:

Insert into Emails(EmailID,EmailFullName,EmailAddress)

Values

(1,'Bob Smith','BSmith@MyCo.com'),(2,'Sue Jones','SJones@MyCo.com');

Go

Select EmailID,EmailFullName,EmailAddress From Emails

Go

Estimated Time: 20 Minutes

# ADO

“ADO.NET is a set of classes that expose data access services to the .NET programmer. ADO.NET provides functionality to developers writing managed code similar to the functionality provided to native COM developers by ADO. ADO.NET provides consistent access to data sources such as Microsoft® SQL Server™, as well as data sources exposed through OLE DB and XML. Data-sharing consumer applications can use ADO.NET to connect to these data sources and retrieve, manipulate, and update data.” (<https://msdn.microsoft.com/en-us/library/aa286484.aspx>)

## Entity Framework

“The Entity Framework exists as a new part of the ADO.NET family of technologies. ADO.NET will LINQ-enable many data access components: LINQ to SQL, LINQ to DataSet and LINQ to Entities.” (<https://msdn.microsoft.com/en-us/library/aa697427(v=vs.80).aspx>)

## Connections

“A Connection object represents a unique session with a data source. In a client/server database system, it may be equivalent to an actual network connection to the server. Depending on the functionality supported by the provider, some collections, methods, or properties of a Connection object may not be available.” ( <https://msdn.microsoft.com/en-us/library/ms681519(v=vs.85).aspx> )

private void buttonConnections\_Click(object sender, EventArgs e)

{

SqlConnection objCon = new SqlConnection();

/\*\* option 1: The "HardCoded" way \*\*/

objCon.ConnectionString = @"Data Source=.\SQLEXPRESS;Initial Catalog = DB1; Integrated Security=True;";

/\*\* option 2: The "Before .Net v2.0" way\*\*/

objCon.ConnectionString = ConfigurationSettings.AppSettings["Conn2"];

/\*\* option 3: The "Common" way (Note: You must have a reference to the System.Configuration .dll) \*\*/

objCon.ConnectionString = ConfigurationManager.ConnectionStrings["Conn3"].ConnectionString;

objCon.StateChange += new StateChangeEventHandler(objCon\_StateChange);

try

{

objCon.Open();

objCon.ChangeDatabase("Northwind");

MessageBox.Show("Database is now: " + objCon.Database);

}

finally

{

objCon.Close();

}

MessageBox.Show("Database state after Close(): " + objCon.State.ToString());

}

public void objCon\_StateChange(object sender, System.Data.StateChangeEventArgs e)

{

MessageBox.Show("Connection state changed event: " + e.CurrentState.ToString());

}

## Commands

“Use a Command object to query a database and return records in a [Recordset](https://msdn.microsoft.com/en-us/library/ms681510(v=vs.85).aspx) object, to execute a bulk operation, or to manipulate the structure of a database. Depending on the functionality of the provider, some Command collections, methods, or properties may generate an error when they are referenced.” ( <https://msdn.microsoft.com/en-us/library/ms677502(v=vs.85).aspx> )

private void buttonCommands\_Click(object sender, EventArgs e)

{

string connectionString = @"Data Source=(Local);Initial Catalog = Master; Integrated Security=True;";

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandType = CommandType.Text;

command.CommandText = "Select name, database\_id from Sys.databases";

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

Console.WriteLine(reader[0]); // Gets the name Column

Console.WriteLine(reader["database\_id"]); //Get the Database\_id column

}

}//end while

}//end using

}//end using

}

# **Demo**: Using ADO.NET

Your instructor will now demo how to use a connection and command object to select and insert data into a table.

# Lab 05-2

In this lab, you will add and display data from the Emails table.

1. Create a three textboxes and a button on Form1 to accept user data.
2. Insert data into the table using Static Data and verify that it worked.
3. Insert data into the table using data from the textboxes and verified that it worked.

Estimated Time: 20 Minutes

# Stored Procedures

“A stored procedure is nothing more than prepared SQL code that you save so you can reuse the code over and over again. So if you think about a query that you write over and over again, instead of having to write that query each time you would save it as a stored procedure and then just call the stored procedure to execute the SQL code that you saved as part of the stored procedure.” ( [https://www.mssqltips.com/sqlservertutorial/160/sql-server-stored-procedure-tutorial](https://www.mssqltips.com/sqlservertutorial/160/sql-server-stored-procedure-tutorial/) )

private void buttonCmdWithSprocs\_Click(object sender, EventArgs e)

{

/\* SQL Code Needed ...

Create Proc GetCompanyContacts

as

Select CompanyName, ContactName

from Customers

GO

-- Runs with this command

Exec GetCompanyContacts

\*/

//Notice that this code is different then the last example, but it still works just fine.

//There are lots of ways to write this ADO.NET code!

System.Data.SqlClient.SqlConnection objCon;

objCon = new System.Data.SqlClient.SqlConnection();

objCon.ConnectionString = @"Data Source=(local);Initial Catalog = Northwind; Integrated Security=True;";

objCon.Open();

SqlCommand objCmd = new SqlCommand();

objCmd.Connection = objCon;

objCmd.CommandType = CommandType.StoredProcedure;

objCmd.CommandText = "GetCompanyContacts";

//objCmd.CommandType = CommandType.Text;

//objCmd.CommandText = "Select CompanyName, ContactName from Customers";

SqlDataReader objDR = objCmd.ExecuteReader();

//Process the Results

while (objDR.Read())

{

Console.WriteLine(objDR[0] + ": " + objDR[1]);

}

objCon.Close();

}

## Stored Procedure Parameters

“The real power of stored procedures is the ability to pass parameters and have the stored procedure handle the differing requests that are made. In this topic we will look at passing parameter values to a stored procedure.” ( [https://www.mssqltips.com/sqlservertutorial/162/how-to-create-a-sql-server-stored-procedure-with-parameters](https://www.mssqltips.com/sqlservertutorial/162/how-to-create-a-sql-server-stored-procedure-with-parameters/) )

private void buttonSprocWithParams\_Click(object sender, EventArgs e)

{

/\* SQL Code Needed ...

Create Proc GetCompanyContactsWithParams

( @SearchWord varchar(100) )

as

Select CompanyName, ContactName

from Customers

Where CompanyName Like '%' + @SearchWord + '%'

GO

-- Runs with this command

Exec GetCompanyContactsWithParams @SearchWord = 'z'

\*/

System.Data.SqlClient.SqlConnection objCon;

objCon = new System.Data.SqlClient.SqlConnection();

objCon.ConnectionString = @"Data Source=(local);Initial Catalog = Northwind; Integrated Security=True;";

objCon.Open();

System.Data.SqlClient.SqlCommand objCmd;

objCmd = new System.Data.SqlClient.SqlCommand();

objCmd.Connection = objCon;

objCmd.CommandType = CommandType.StoredProcedure;

objCmd.CommandText = "GetCompanyContactsWithParams";

System.Data.SqlClient.SqlParameter objP1 = new System.Data.SqlClient.SqlParameter();

objP1.Direction = ParameterDirection.Input;

objP1.ParameterName = "@SearchWord";

objP1.SqlDbType = SqlDbType.VarChar;

objP1.Size = 100;

objP1.Value = "zz";

objCmd.Parameters.Add(objP1);

System.Data.SqlClient.SqlDataReader objDR

= objCmd.ExecuteReader();

//Process the Results

while (objDR.Read())

{

Console.WriteLine(objDR[0] + ": " + objDR[1]);

}

objCon.Close();

}