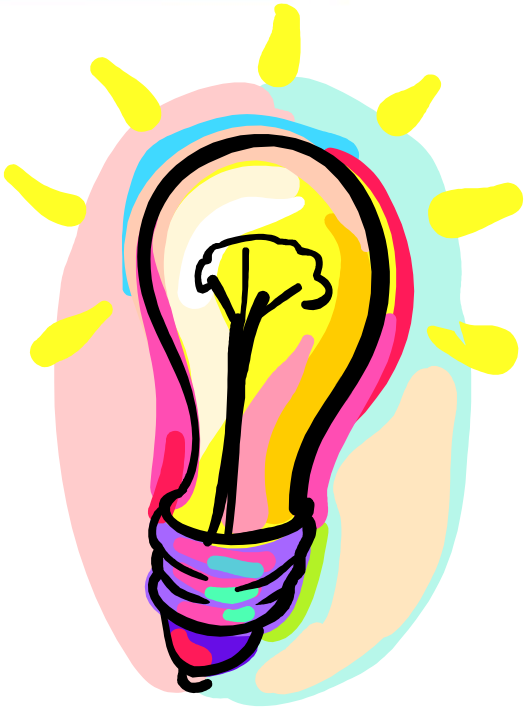
ADO.NET

Objectives

- Guide for research :
  - ✓ ADO.NET overview
  - ✓ Connection
  - ✓ Command
  - ✓ Command Type
  - ✓ Command Method
  - ✓ Parameter



Section 1

ADO.NET OVERVIEW

Before ADO.NET

- User CANNOT direct access to SQL server
- User CANNOT use SQL command to manipulate data
- Data comes from various data source: SQL, Oracle, Excel, …

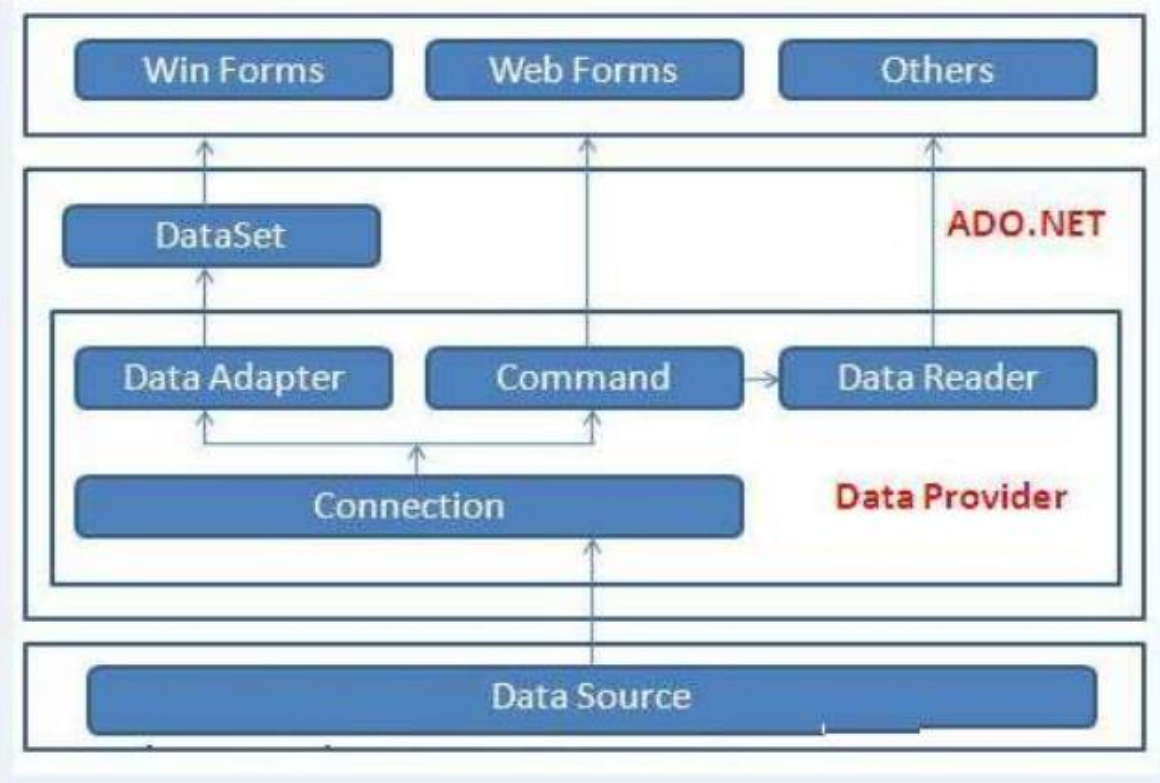
What is ADO.NET ?

- *ADO.net is the data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways.*
- *It is based on the .NET framework and it is highly integrated with the rest of the framework class library.*

What is ADO.NET ?

- *A collection of classes, interfaces, structures, enumerated type that mange data access from the relational data stores within the .NET framework.*

ADO.NET Architecture



Discussion

- Which layer of project that ADO.NET belong to?

- Which type of code should be included in?

Section 2

TYPE OF ARCHITECTURE

Type of Architecture

- Connection Oriented Architecture: *connects to a database, retrieving data, storing the data in a dataset, reading the retrieved data, and updating the database*
  - *Connection,*
  - *Command,*
  - *DataReader,*
  - *DataAdapter*

Type of Architecture

- Disconnected Oriented Architecture: a memory-based relational representation of data.
  - *DataTableCollection,*
  - *DataRelationCollection,*
  - *DataTable,*
  - *DataRowCollection,*
  - *DataColoumnCollection*

Section 3

CONNECTION OBJECT

Connection Object

- **Connection Object:** allows to establish a connection to a data source.
- The connection objects have the methods for opening and closing connections, for beginning a transaction of data.
- Two types of connection classes:
  - ✓ SqlConnection: to connect to Microsoft SQL Server
  - ✓ OleDbConnection: to provide connection to a wide range of databases, such as Microsoft Access and Oracle

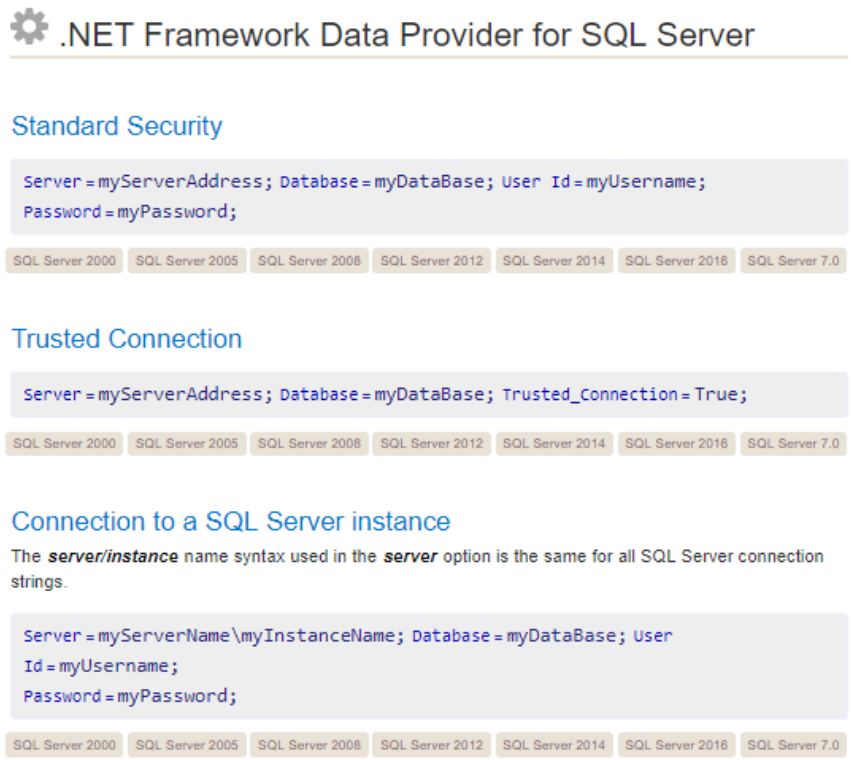
Steps to work

- Instantiate the SqlConnection class.
- Open connection.
- Pass the connection to ADO.NET objects.
- Perform the database operations with ADO.NET object.
- Close the connection.

ConnectionString

- All format:
  - ✓ https://www.connectionstrings.com/
  - ✓ https://www.connectionstrings.com/sql-server/
- Example:

Server=myServerAddress;Database=myDataBase;User Id=myUsername;
Password=myPassword;



ConnectionString

No.	Connection String Parameter Name	Description
1	Data Source	Identify the server. Could be local machine, machine domain name, or IP Address.
2	Initial Catalog	Data base name.
3	Integrated Security	Set to SSIP to make connection with user's window login.
4	User ID	Name of user configured in SQL Server.
5	Password	Password matching SQL Server User ID

```
<connectionStrings>
  <add name="MyConnection"
    connectionString="Server=myServerAddress;Database=myDataBase;User Id=myUsername;
Password=myPassword;"
    providerName="System.Data.SqlClient" />
</connectionStrings>
```

Provider

No	Provider	Description
1	System.Data.SqlClient	Provides data for Microsoft SQL Server
2	System.Data.OleDb	Provides data access for data sources exposed using OLE DB
3	System.Data.Odbc	Provides data access for data source exposed using ODBC.
4	System.Data.OracleClient	Provides data access for Oracle.

```
<connectionStrings>
  <add name="MyConnection"
      connectionString="Server=myServerAddress;Database=myDataBase;User Id=myUsername;
Password=myPassword;"
      providerName="System.Data.SqlClient" />
 </connectionStrings>
```

Best Practice

- **DONOT** hardcode connection string in your code. Put it in configurable file (App.config, Web.config, …)
- Step 1: Put connection string in App.config/Web.config
- Step 2: Get it in your code

```
<connectionStrings>
  <add name="MyConnection"
connectionString="Server=myServerAddress;Database=myDataBase;User Id=myUsername; Password=myPassword;"
       providerName="System.Data.SqlClient" />
</connectionStrings>
```

```
SqlConnection sqlConnection = new SqlConnection();
sqlConnection.ConnectionString =
ConfigurationManager.ConnectionStrings["MyConnection"].ConnectionString;
```

Section 4

COMMAND OBJECT

Command Object

- An object executes SQL statements on the database.
- A connection object specifies the type of interaction to perform with the database,
- Statements can be SELECT, INSERT, UPDATE, or DELETE.
- Statements are in string, and usually are combined by arguments

CommandType

- CommandType.Text: An SQL text command. (Default.)
- CommandType.StoredProcedure: The name of a stored procedure.
- CommandType.TableDirect: The name of a table, only supported by the .NET Framework Data Provider for **OLE DB**

```
SqlCommand command = new SqlCommand(sqlConnection);
command.CommandType = System.Data.CommandType.Text;
```

Examples

- CommandType.Text:

```
SqlCommand command = new SqlCommand();
command.Connection = sqlConnection;
command.CommandType = System.Data.CommandType.Text;
command.CommandText = "SELECT Name, Salary FROM Employee";
```

- CommandType.StoredProcedure:

```
SqlCommand command = new SqlCommand();
command.Connection = sqlConnection;
command.CommandType = System.Data.CommandType.StoredProcedure;
command.CommandText = "GetEmployee";
```

Execute Methods

- **ExecuteNonQuery**:
  - ✓ Executes a Transact-SQL statement against the connection and returns the number of rows affected.
  - ✓ Usually used in INSERT, UPDATE, DELETE statements

```
command.CommandType = System.Data.CommandType.StoredProcedure;
 command.CommandText = "UPDATE Employee SET Salary = 12345 WHERE Name = 'Peter'";
 command.ExecuteNonQuery()
```

Execute Methods

- **ExecuteReader:**
  - ✓ Sends the System.Data.SqlClient.SqlCommand.CommandText to the System.Data.SqlClient.SqlCommand.Connection and builds a System.Data.SqlClient.SqlDataReader.
  - ✓ Usually used in SELECT statement

```
command.CommandType = System.Data.CommandType.StoredProcedure;
command.CommandText = "GetEmployee";
var sqlDataReader = command.ExecuteReader();
```

Execute Methods

- **ExecuteScalar:**
  - ✓ Returns the first column of the first row in the result set returned by the query. Additional columns or rows are ignored.
  - ✓ Usually used to select a value such as: MIN, MAX, COUNT, AVG, …

```
command.CommandType = System.Data.CommandType.StoredProcedure;
command.CommandText = "SELECT MAX(Salary) FROM Employee";
var result = command.ExecuteScalar();
```

SQL Injection

- Try this code

```
var name = GetNameFromUser();
command.CommandText = string.Format("UPDATE Employee SET Salary = 12345
WHERE Name = '{0}'", name);
```

- The name that you expect

  name = “Peter”

- The name that user/hacker gives

  name = “Peter' OR 1 =1 --”

- Run and check your data
- How many record are updated?

SqlParameter

- Is used to pass parameter to SqlCommand.
- Makes SQL queries easier to build and read
- Is an idea to avoid SQL Injection.

SqlParameter

```
var name = GetValueFromUser();
SqlCommand command = new SqlCommand();
command.Connection = sqlConnection;
SqlParameter sqlParameterName = new SqlParameter("@parameterName", name);
command.CommandType = System.Data.CommandType.StoredProcedure;
command.CommandText = "UPDATE Employee SET Salary = 12345 WHERE Name =
'@parameterName'";
command.Parameters.Add(sqlParameterName);
command.ExecuteNonQuery();
```

Summary

Thank you