**SqlCommand.ExecuteReader Method**

Sends the CommandText to the Connection and builds a SqlDataReader.

This member is overloaded.

ExecuteReader(): Sends the CommandText to the Connection and builds a SqlDataReader.

ExecuteReader(CommandBehavior): Sends the CommandText to the Connection, and builds a SqlDataReader using one of the CommandBehavior values.

**OleDbCommand.ExecuteReader Method**

Sends the CommandText to the Connection and builds an OleDbDataReader.

This member is overloaded.

ExecuteReader() : Sends the CommandText to the Connection and builds an OleDbDataReader.

ExecuteReader(CommandBehavior): Sends the CommandText to the Connection, and builds an OleDbDataReader using one of the CommandBehavior values.

**SqlCommand.ExecuteNonQuery Method**

Executes a Transact-SQL statement against the connection and returns the number of rows affected.

You can use the ExecuteNonQuery to perform catalog operations (for example, querying the structure of a database or creating database objects such as tables), or to change the data in a database without using a DataSet by executing UPDATE, INSERT, or DELETE statements.

Although the ExecuteNonQuery returns no rows, any output parameters or return values mapped to parameters are populated with data.

For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. When a trigger exists on a table being inserted or updated, the return value includes the number of rows affected by both the insert or update operation and the number of rows affected by the trigger or triggers. For all other types of statements, the return value is -1. If a rollback occurs, the return value is also -1.

**OleDbCommand.ExecuteNonQuery Method**

Executes an SQL statement against the Connection and returns the number of rows affected.

You can use the ExecuteNonQuery to perform catalog operations, for example, to query the structure of a database or to create database objects such as tables, or to change the data in a database without using a DataSet by executing UPDATE, INSERT, or DELETE statements.

Although the ExecuteNonQuery returns no rows, any output parameters or return values mapped to parameters are populated with data.

For UPDATE, INSERT, and DELETE statements, the return value is the number of rows affected by the command. For all other types of statements, the return value is -1. If a rollback occurs, the return value is also -1.

**SqlCommand.ExecuteScalar Method**

Executes the query, and returns the first column of the first row in the result set returned by the query. Additional columns or rows are ignored.

**Return Value**

**Type: System.Object**

The first column of the first row in the result set, or a null reference (Nothing in Visual Basic) if the result set is empty. Returns a maximum of 2033 characters.

Use the ExecuteScalar method to retrieve a single value (for example, an aggregate value) from a database. This requires less code than using the ExecuteReader method, and then performing the operations that you need to generate the single value using the data returned by a SqlDataReader.

A typical ExecuteScalar query can be formatted as in the following C# example:

cmd.CommandText = "SELECT COUNT(\*) FROM dbo.region";

Int32 count = (Int32) cmd.ExecuteScalar();

**OleDbCommand.ExecuteScalar Method**

Executes the query, and returns the first column of the first row in the result set returned by the query. Additional columns or rows are ignored.

**Return Value**

**Type: System.Object**

The first column of the first row in the result set, or a null reference if the result set is empty.

Use the ExecuteScalar method to retrieve a single value, for example, an aggregate value, from a data source. This requires less code than using the ExecuteReader method, and then performing the operations that are required to generate the single value using the data returned by an OleDbDataReader.

A typical ExecuteScalar query can be formatted as in the following C# example:

CommandText = "SELECT COUNT(\*) FROM region";

Int32 count = (int32) ExecuteScalar();

**ExecuteReader**

**Do not use:** when database query is going to provide for sure exactly 1 record. It may be getting record by its id (which is PK in the database) - GetOrderById and such. In this case use ExecuteNonQuery with output parameters.

**Use:** when database query is going to provide a set of records. It may be search or report.

**ExecuteNonQuery**

**Use:** when we are talking about a single database record - in Update, Insert, Delete and Get by Id. In all these cases we can use input/output/input-output parameters. Please note that from the application architecture point of view it is also good practices when your Insert and Update stored procedure returns changed record exactly like Get By Id method does.

**ExecuteScalar**

**Do not use:** when database query returns a single value and this value can be defined as parameter in T-SQL. ExecuteNonQuery with output parameter(s) is always preferred in this case since it is more flexible, tomorrow there will be 2 values therefore having ExecuteNonQuery we do not need to change method signatures.

**Use:** when database query returns a single value and this value cannot be defined as output parameter, because of T-SQL type limitation for variables. For example type *image* cannot be output parameter in MSSQL.

The most common example for ExecuteScalar is fetching a single image stored in the database and converting it to array of bytes. If you google it - most examples will demonstrate using of ExecuteReader to accomplish image handler, however ExecuteScalar will be more scalable and faster.

**Conclusion**

Always use ExecuteNonQuery except: when you have a set of records - use ExecuteReader and when you have a single output value that cannot be defined as a parameter - use ExecuteScalar.

|  |  |  |
| --- | --- | --- |
| **SqlCommand /**  **OleDbCommand** | **Return Type** | |
| **ExecuteReader** | System.Data.SqlClient.SqlDataReader | System.Data.OleDb.OleDbDataReader |
| **ExecuteNonQuery** | System.Int32 | System.Int32 |
| **ExecuteScalar** | System.Object | System.Object |

**Note: ExecuteReader** is overloaded with parameter **System.Data.CommandBehavior**

**ExecuteReader:** For accessing data. Returns forward-only, read-only connected record set.

**ExecuteNonQuery:** For data manipulation such as INSERT, UPDATE, DELETE. Returns number of rows affected.

**ExecuteScalar:** For retrieving first row first column value.