This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

**SqlCommand.ExecuteReader Method**

Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx) and builds a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx).

This member is overloaded. For complete information about this member, including syntax, usage, and examples, click a name in the overload list.

|  |  |
| --- | --- |
| **Name** | **Description** |
| [ExecuteReader(](http://msdn.microsoft.com/en-us/library/9kcbe65k.aspx)) | Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx) and builds a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx). |
| [ExecuteReader(CommandBehavior)](http://msdn.microsoft.com/en-us/library/y6wy5a0f.aspx) | Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx), and builds a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) using one of the [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) values. |

**SqlCommand.ExecuteReader Method**

Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx) and builds a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx).

#### Return Value

Type: [System.Data.SqlClient.SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx)

Remarks

When the [CommandType](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtype.aspx) property is set to **StoredProcedure**, the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) property should be set to the name of the stored procedure. The command executes this stored procedure when you call [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx).

|  |
| --- |
| **Note** |
| If a transaction is deadlocked, an exception may not be thrown until [Read](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.read.aspx) is called. |

When you use versions of SQL Server before SQL Server 2005, while the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) is being used, the associated [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) is busy serving the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx). While in this state, no other operations can be performed on the [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) other than closing it. This is the case until the [Close](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.close.aspx) method of the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) is called. Starting with SQL Server 2005, the multiple active result set (MARS) feature allows for multiple actions using the same connection.

If you use [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx) or [BeginExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutereader.aspx) to access XML data, SQL Server will return any XML results greater than 2,033 characters in length in multiple rows of 2,033 characters each. To avoid this behavior, use [ExecuteXmlReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executexmlreader.aspx) or [BeginExecuteXmlReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutexmlreader.aspx) to read FOR XML queries.

http://i.msdn.microsoft.com/Global/Images/clear.gifExamples

The following example creates a [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.aspx), and then executes it by passing a string that is a Transact-SQL SELECT statement, and a string to use to connect to the data source.

Visual Basic

Public Sub CreateCommand(ByVal queryString As String, \_

ByVal connectionString As String)

Using connection As New SqlConnection(connectionString)

connection.Open()

Dim command As New SqlCommand(queryString, connection)

Dim reader As SqlDataReader = command.ExecuteReader()

While reader.Read()

Console.WriteLine("{0}", reader(0))

End While

End Using

End Sub

C#

private static void CreateCommand(string queryString,

string connectionString)

{

using (SqlConnection connection = new SqlConnection(

connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand(queryString, connection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Console.WriteLine(String.Format("{0}", reader[0]));

}

}

}

**SqlCommand.ExecuteReader Method (CommandBehavior)**

Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx), and builds a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) using one of the [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) values.

#### Parameters

Type: [System.Data.CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx)

#### Return Value

Type: [System.Data.SqlClient.SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx)

Remarks

When the [CommandType](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtype.aspx) property is set to **StoredProcedure**, the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) property should be set to the name of the stored procedure. The command executes this stored procedure when you call [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx).

|  |
| --- |
| **Note** |
| Use [SequentialAccess](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) to retrieve large values and binary data. Otherwise, an [OutOfMemoryException](http://msdn.microsoft.com/en-us/library/system.outofmemoryexception.aspx) might occur and the connection will be closed. |

When you use versions of SQL Server earlier than SQL Server 2005, while the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) is being used, the associated [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) is busy serving the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx). While in this state, no other operations can be performed on the [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) other than closing it. This is the case until you call the [SqlDataReader.Close](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.close.aspx) method. If the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) is created with [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) set to **CloseConnection**, closing the [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) closes the connection automatically. Starting with SQL Server 2005, the multiple active result set (MARS) feature allows for multiple actions using the same connection.

If you use [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx) or [BeginExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutereader.aspx) to access XML data, SQL Server will return any XML results greater than 2,033 characters in length in multiple rows of 2,033 characters each. To avoid this behavior, use [ExecuteXmlReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executexmlreader.aspx) or [BeginExecuteXmlReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.beginexecutexmlreader.aspx) to read FOR XML queries.

http://i.msdn.microsoft.com/Global/Images/clear.gifExamples

The following example creates a [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.aspx), and then executes it by passing a string that is a Transact-SQL SELECT statement, and a string to use to connect to the data source. [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) is set to [CloseConnection](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx).

Visual Basic

Public Sub CreateCommand(ByVal queryString As String, \_

ByVal connectionString As String)

Using connection As New SqlConnection(connectionString)

Dim command As New SqlCommand(queryString, connection)

connection.Open()

Dim reader As SqlDataReader = \_

command.ExecuteReader(CommandBehavior.CloseConnection)

While reader.Read()

Console.WriteLine("{0}", reader(0))

End While

End Using

End Sub

C#

private static void CreateCommand(string queryString,

string connectionString)

{

using (SqlConnection connection = new SqlConnection(

connectionString))

{

SqlCommand command = new SqlCommand(queryString, connection);

connection.Open();

SqlDataReader reader =

command.ExecuteReader(CommandBehavior.CloseConnection);

while (reader.Read())

{

Console.WriteLine(String.Format("{0}", reader[0]));

}

}

}

**OleDbCommand.ExecuteReader Method**

Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.connection.aspx) and builds an [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx).

This member is overloaded. For complete information about this member, including syntax, usage, and examples, click a name in the overload list.

|  |  |
| --- | --- |
| **Name** | **Description** |
| [ExecuteReader()](http://msdn.microsoft.com/en-us/library/979byfca.aspx) | Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.connection.aspx) and builds an [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx). |
| [ExecuteReader(CommandBehavior)](http://msdn.microsoft.com/en-us/library/aet5yz0b.aspx) | Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.connection.aspx), and builds an [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) using one of the [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) values. |

**OleDbCommand.ExecuteReader Method**

Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.connection.aspx) and builds an [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx).

#### Return Value

Type: [System.Data.OleDb.OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx)

http://i.msdn.microsoft.com/Global/Images/clear.gifRemarks

When the [CommandType](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtype.aspx) property is set to StoredProcedure, the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) property should be set to the name of the stored procedure. The command executes this stored procedure when you call [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.executereader.aspx).

Before you close the [OleDbConnection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbconnection.aspx), first close the [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) object. You must also close the [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) object if you plan to reuse an [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx) object.

http://i.msdn.microsoft.com/Global/Images/clear.gifExamples

The following example creates an [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx), and then executes it by passing a string that is an SQL SELECT statement, and a string to use to connect to the data source.

Visual Basic

Public Sub CreateReader(ByVal connectionString As String, \_

ByVal queryString As String)

Using connection As New OleDbConnection(connectionString)

Dim command As New OleDbCommand(queryString, connection)

connection.Open()

Dim reader As OleDbDataReader = command.ExecuteReader()

While reader.Read()

Console.WriteLine(reader(0).ToString())

End While

reader.Close()

End Using

End Sub

C#

public void CreateReader(string connectionString, string queryString)

{

using (OleDbConnection connection = new OleDbConnection(connectionString))

{

OleDbCommand command = new OleDbCommand(queryString, connection);

connection.Open();

OleDbDataReader reader = command.ExecuteReader();

while (reader.Read())

{

Console.WriteLine(reader[0].ToString());

}

reader.Close();

}

}

**OleDbCommand.ExecuteReader Method (CommandBehavior)**

Sends the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) to the [Connection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.connection.aspx), and builds an [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) using one of the [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) values.

#### Parameter

Type: [System.Data.CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx)

#### Return Value

Type: [System.Data.OleDb.OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx)

Remarks

When you specify [SingleRow](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) with the [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.executereader.aspx) method of the [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx) object, the .NET Framework Data Provider for OLE DB performs binding using the OLE DB **IRow** interface if it is available. Otherwise, it uses the **IRowset** interface. If your SQL statement is expected to return only a single row, specifying [SingleRow](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) can also improve application performance.

When the [CommandType](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtype.aspx) property is set to StoredProcedure, the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.commandtext.aspx) property should be set to the name of the stored procedure. The command executes this stored procedure when you call [ExecuteReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.executereader.aspx).

The [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) supports a special mode that enables large binary values to be read efficiently. For more information, see the SequentialAccess setting for [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx).

Before you close the [OleDbConnection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbconnection.aspx), first close the [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) object. You must also close the [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) object if you plan to reuse an [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx) object. If the [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) is created with [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) set to CloseConnection, closing the [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx) closes the connection automatically.

http://i.msdn.microsoft.com/Global/Images/clear.gifExamples

The following example creates an [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx), and then executes it by passing a string that is a Transact-SQL SELECT statement, and a string to use to connect to the data source. [CommandBehavior](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx) is set to [CloseConnection](http://msdn.microsoft.com/en-us/library/system.data.commandbehavior.aspx).

Visual Basic

Public Sub CreateMyOleDbDataReader(queryString As String, \_

connectionString As String)

Dim connection As New OleDbConnection(connectionString)

Dim command As New OleDbCommand(queryString, connection)

connection.Open()

Dim reader As OleDbDataReader = command.ExecuteReader(CommandBehavior.CloseConnection)

While reader.Read()

Console.WriteLine(reader.GetString(0))

End While

reader.Close()

'Implicitly closes the connection because CommandBehavior.CloseConnection was specified.

End Sub

C#

public void CreateMyOleDbDataReader(string queryString,string connectionString)

{

OleDbConnection connection = new OleDbConnection(connectionString);

OleDbCommand command = new OleDbCommand(queryString, connection);

connection.Open();

OleDbDataReader reader = command.ExecuteReader(CommandBehavior.CloseConnection);

while(reader.Read())

{

Console.WriteLine(reader.GetString(0));

}

reader.Close();

//Implicitly closes the connection because CommandBehavior.CloseConnection was specified.

}

This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework Class Library

**SqlCommand.ExecuteNonQuery Method**

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

#### Return Value

Type: [System.Int32](http://msdn.microsoft.com/en-us/library/system.int32.aspx)  
The number of rows affected.

#### Implements

[IDbCommand.ExecuteNonQuery()](http://msdn.microsoft.com/en-us/library/system.data.idbcommand.executenonquery.aspx)

Remarks

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](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) 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.

Examples

The following example creates a [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.aspx) and then executes it using ExecuteNonQuery. The example is passed a string that is a Transact-SQL statement (such as UPDATE, INSERT, or DELETE) and a string to use to connect to the data source.

Visual Basic

Public Sub CreateCommand(ByVal queryString As String, \_

ByVal connectionString As String)

Using connection As New SqlConnection(connectionString)

Dim command As New SqlCommand(queryString, connection)

command.Connection.Open()

command.ExecuteNonQuery()

End Using

End Sub

C#

private static void CreateCommand(string queryString,

string connectionString)

{

using (SqlConnection connection = new SqlConnection(

connectionString))

{

SqlCommand command = new SqlCommand(queryString, connection);

command.Connection.Open();

command.ExecuteNonQuery();

}

}

**OleDbCommand.ExecuteNonQuery Method**

Executes an SQL statement against the [Connection](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.connection.aspx) and returns the number of rows affected.

#### Return Value

Type: [System.Int32](http://msdn.microsoft.com/en-us/library/system.int32.aspx)  
The number of rows affected.

#### Implements

[IDbCommand.ExecuteNonQuery()](http://msdn.microsoft.com/en-us/library/system.data.idbcommand.executenonquery.aspx)

Remarks

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](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) 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.

Examples

The following example creates an [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx) and then executes it using ExecuteNonQuery. The example is passed a string that is an SQL statement such as UPDATE, INSERT, or DELETE, and a string to use to connect to the data source.

Visual Basic

Private Sub CreateOleDbCommand( \_

ByVal queryString As String, ByVal connectionString As String)

Using connection As New OleDbConnection(connectionString)

connection.Open()

Dim command As New OleDbCommand(queryString, connection)

command.ExecuteNonQuery()

End Using

End Sub

C#

static private void CreateOleDbCommand(

string queryString, string connectionString)

{

using (OleDbConnection connection = new

OleDbConnection(connectionString))

{

connection.Open();

OleDbCommand command = new

OleDbCommand(queryString, connection);

command.ExecuteNonQuery();

}

}

This page is specific to

Microsoft Visual Studio 2010/.NET Framework 4

.NET Framework Class Library

**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](http://msdn.microsoft.com/en-us/library/system.object.aspx)  
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.

#### Implements

[IDbCommand.ExecuteScalar()](http://msdn.microsoft.com/en-us/library/system.data.idbcommand.executescalar.aspx)

Remarks

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](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.executereader.aspx) method, and then performing the operations that you need to generate the single value using the data returned by a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx).

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();

Examples

The following example creates a [SqlCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.aspx) and then executes it using ExecuteScalar. The example is passed a string representing a new value to be inserted into a table, and a string to use to connect to the data source. The function returns the new **Identity** column value if a new row was inserted, 0 on failure.

Visual Basic

Public Function AddProductCategory( \_

ByVal newName As String, ByVal connString As String) As Integer

Dim newProdID As Int32 = 0

Dim sql As String = \_

"INSERT INTO Production.ProductCategory (Name) VALUES (@Name); " \_

& "SELECT CAST(scope\_identity() AS int);"

Using conn As New SqlConnection(connString)

Dim cmd As New SqlCommand(sql, conn)

cmd.Parameters.Add("@Name", SqlDbType.VarChar)

cmd.Parameters("@Name").Value = newName

Try

conn.Open()

newProdID = Convert.ToInt32(cmd.ExecuteScalar())

Catch ex As Exception

Console.WriteLine(ex.Message)

End Try

End Using

Return newProdID

End Function

C#

static public int AddProductCategory(string newName, string connString)

{

Int32 newProdID = 0;

string sql =

"INSERT INTO Production.ProductCategory (Name) VALUES (@Name); "

+ "SELECT CAST(scope\_identity() AS int)";

using (SqlConnection conn = new SqlConnection(connString))

{

SqlCommand cmd = new SqlCommand(sql, conn);

cmd.Parameters.Add("@Name", SqlDbType.VarChar);

cmd.Parameters["@name"].Value = newName;

try

{

conn.Open();

newProdID = (Int32)cmd.ExecuteScalar();

}

catch (Exception ex)

{

Console.WriteLine(ex.Message);

}

}

return (int)newProdID;

}

**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](http://msdn.microsoft.com/en-us/library/system.object.aspx)  
The first column of the first row in the result set, or a null reference if the result set is empty.

#### Implements

[IDbCommand.ExecuteScalar()](http://msdn.microsoft.com/en-us/library/system.data.idbcommand.executescalar.aspx)

Remarks

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](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.executereader.aspx) method, and then performing the operations that are required to generate the single value using the data returned by an [OleDbDataReader](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdatareader.aspx).

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

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

Int32 count = (int32) ExecuteScalar();

Examples

The following example creates an [OleDbCommand](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbcommand.aspx) and then executes it using ExecuteScalar. The example is passed a string that is an SQL statement that returns an aggregate result, and a string to use to connect to the data source.

Visual Basic

Public Sub CreateMyOleDbCommand(queryString As String, \_

connection As OleDbConnection)

Dim command As New OleDbCommand(queryString, connection)

command.Connection.Open()

command.ExecuteScalar()

connection.Close()

End Sub

C#

public void CreateMyOleDbCommand(string queryString,

OleDbConnection connection)

{

OleDbCommand command = new OleDbCommand(queryString, connection);

command.Connection.Open();

command.ExecuteScalar();

connection.Close();

}