**SQLCommand Class**

Represents a Transact-SQL statement or stored procedure to execute against a SQL Server database. This class cannot be inherited.

Constructors

|  |  |  |
| --- | --- | --- |
|    | Name | Description |
|  | [SqlCommand()](http://msdn.microsoft.com/en-us/library/9s8ekk5c.aspx) | Initializes a new instance of the SqlCommand class. |
|  | [SqlCommand(String)](http://msdn.microsoft.com/en-us/library/sebfsz50.aspx) | Initializes a new instance of the SqlCommand class with the text of the query. |
|  | [SqlCommand(String, SqlConnection)](http://msdn.microsoft.com/en-us/library/877h0y3a.aspx) | Initializes a new instance of the SqlCommand class with the text of the query and a [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx). |
|  | [SqlCommand(String, SqlConnection, SqlTransaction)](http://msdn.microsoft.com/en-us/library/352y4sff.aspx) | Initializes a new instance of the SqlCommand class with the text of the query, a [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx), and the [SqlTransaction](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqltransaction.aspx). |

The Most Important Properties

|  |  |  |
| --- | --- | --- |
|    | Name | Description |
|  | [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) | Gets or sets the Transact-SQL statement, table name or stored procedure to execute at the data source. (Overrides [DbCommand.CommandText](http://msdn.microsoft.com/en-us/library/system.data.common.dbcommand.commandtext.aspx).) |
|  | [CommandTimeout](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtimeout.aspx) | Gets or sets the wait time before terminating the attempt to execute a command and generating an error. (Overrides [DbCommand.CommandTimeout](http://msdn.microsoft.com/en-us/library/system.data.common.dbcommand.commandtimeout.aspx).) |
|  | [CommandType](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtype.aspx) | Gets or sets a value indicating how the [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) property is to be interpreted. (Overrides [DbCommand.CommandType](http://msdn.microsoft.com/en-us/library/system.data.common.dbcommand.commandtype.aspx).) |
|  | [Connection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.connection.aspx) | Gets or sets the [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) used by this instance of the SqlCommand. |

**Transactional Methods is SQLCommand Class**

**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. 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, 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();

**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). 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). 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.

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.ExecuteXmlReader 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 an [XmlReader](http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx) object. The [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) property ordinarily specifies a Transact-SQL statement with a valid FOR XML clause. However, [CommandText](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlcommand.commandtext.aspx) can also specify a statement that returns ntext or nvarchar data that contains valid XML, or the contents of a column defined with the xml data type. A typical ExecuteXmlReader query can be formatted as in the following Microsoft Visual C# example:

SqlCommand command = new SqlCommand("SELECT \* FROM dbo.Customers FOR XML AUTO, XMLDATA", SqlConn);

This method can also be used to retrieve a single-row, single-column result set that contains XML data. In this case, if more than one row is returned, the ExecuteXml Reader method attaches the [XmlReader](http://msdn.microsoft.com/en-us/library/system.xml.xmlreader.aspx) to the value on the first row, and discards the rest of the result set.The multiple active result set (MARS) feature allows for multiple actions using the same connection.

**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](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.

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();

 }

}