SqlConnection Class

Namespace: System.Data.SqlClient

Assemblies: System.Data.SqlClient.dll, System.Data.dll, netstandard.dll

Represents a connection to a SQL Server database. This class cannot be inherited.

In this article

C# Copy

```
public sealed class SqlConnection : System.Data.Common.DbConnection, ICloneable, IDisposable
```

Inheritance Object → MarshalByRefObject → Component → SqlConnection

Implements IDbConnection , ICloneable , IDisposable

Examples

The following example creates a SqlCommand and a SqlConnection. The SqlConnection is opened and set as the Connection for the SqlCommand. The example then calls ExecuteNonQuery. To accomplish this, the ExecuteNonQuery is passed a connection string and a query string that is a Transact-SQL INSERT statement. The connection is closed automatically when the code exits the using block.

C# Copy

```
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();
    }
}
```

Remarks

A SqlConnection object represents a unique session to a SQL Server data source. With a client/server database system, it is equivalent to a network connection to the server. SqlConnection is used together with SqlDataAdapter and SqlCommand to increase performance when connecting to a Microsoft SQL Server database. For all third-party SQL Server products and other OLE DB-supported data sources, use OleDbConnection.

When you create an instance of SqlConnection, all properties are set to their initial values. For a list of these values, see the SqlConnection constructor.

See ConnectionString for a list of the keywords in a connection string.

If the SqlConnection goes out of scope, it won't be closed. Therefore, you must explicitly close the connection by calling `Close` or `Dispose`. `Close` and `Dispose` are functionally equivalent. If the connection pooling value `Pooling` is set to `true` or `yes`, the underlying connection is returned back to the connection pool. On the other hand, if `Pooling` is set to `false` or `no`, the underlying connection to the server is actually closed.

> ⓘ Note
>
> Login and logout events will not be raised on the server when a connection is fetched from or returned to the connection pool, because the connection is not actually closed when it is returned to the connection pool. For more information, see SQL Server Connection Pooling (ADO.NET).

To ensure that connections are always closed, open the connection inside of a `using` block, as shown in the following code fragment. Doing so ensures that the connection is automatically closed when the code exits the block.

C# Copy

```
using (SqlConnection connection = new SqlConnection(connectionString))
    {
        connection.Open();
        // Do work here; connection closed on following line.
    }
```

> ⓘ Note
>
> To deploy high-performance applications, you must use connection pooling. When you use the .NET Framework Data Provider for SQL Server, you do not have to enable connection pooling because the provider manages this automatically, although you can modify some settings. For more information, see SQL Server Connection Pooling (ADO.NET).

If a SqlException is generated by the method executing a SqlCommand, the SqlConnection remains open when the severity level is 19 or less. When the severity level is 20 or greater, the server ordinarily closes the SqlConnection. However, the user can reopen the connection and continue.

An application that creates an instance of the SqlConnection object can require all direct and indirect callers to have sufficient permission to the code by setting declarative or imperative security demands. SqlConnection makes security demands using the SqlClientPermission object. Users can verify that their code has sufficient permissions by using the SqlClientPermissionAttribute object. Users and administrators can also use the Caspol.exe (Code Access Security Policy Tool) to modify security policy at the machine, user, and enterprise levels. For more information, see Security in .NET. For an example demonstrating how to use security demands, see Code Access Security and ADO.NET.

For more information about handling warning and informational messages from the server, see Connection Events. For more information about SQL Server engine errors and error messages, see Database Engine Events and Errors.

> ⊗ Caution
>
> You can force TCP instead of shared memory. You can do that by prefixing tcp: to the server name in the connection string or you can use localhost.

Constructors

SqlConnection()	Initializes a new instance of the SqlConnection class.
SqlConnection(String)	Initializes a new instance of the SqlConnection class when given a string that contains the connection string.
SqlConnection(String, SqlCredential)	Initializes a new instance of the SqlConnection class given a connection string, that does not use `Integrated Security = true` and a SqlCredential object that contains the user ID and password.

Properties

AccessToken	Gets or sets the access token for the connection.
CanRaiseEvents	Gets a value indicating whether the component can raise an event. (Inherited from Component)
ClientConnectionId	The connection ID of the most recent connection attempt, regardless of whether the attempt succeeded or failed.
ColumnEncryptionKeyCacheTtl	Gets or sets the time-to-live for column encryption key entries in the column encryption key cache for the Always Encrypted feature. The default value is 2 hours. 0 means no caching at all.
ColumnEncryptionQueryMetadataCacheEnabled	Gets or sets a value that indicates whether query metadata caching is enabled (true) or not (false) for parameterized queries running against Always Encrypted enabled databases. The default value is true.
ColumnEncryptionTrustedMasterKeyPaths	Allows you to set a list of trusted key paths for a database server. If while processing an application query the driver receives a key path that is not on the list, the query will fail. This property provides additional protection against security attacks that involve a compromised SQL Server providing fake key paths, which may lead to leaking key store credentials.
ConnectionString	Gets or sets the string used to open a SQL Server database.
ConnectionTimeout	Gets the time to wait (in seconds) while trying to establish a connection before terminating the attempt and generating an error.
Container	Gets the IContainer that contains the Component. (Inherited from Component)
Credential	Gets or sets the SqlCredential object for this connection.
Database	Gets the name of the current database or the database to be used after a connection is opened.
DataSource	Gets the name of the instance of SQL Server to which to connect.
DesignMode	Gets a value that indicates whether the Component is currently in design mode. (Inherited from Component)
Events	Gets the list of event handlers that are attached to this Component. (Inherited from Component)
FireInfoMessageEventOnUserErrors	Gets or sets the FireInfoMessageEventOnUserErrors property.
PacketSize	Gets the size (in bytes) of network packets used to communicate with an instance of SQL Server.
ServerVersion	Gets a string that contains the version of the instance of SQL Server to which the client is connected.
Site	Gets or sets the ISite of the Component. (Inherited from Component)
State	Indicates the state of the SqlConnection during the most recent network operation performed on the connection.
StatisticsEnabled	When set to `true`, enables statistics gathering for the current connection.
WorkstationId	Gets a string that identifies the database client.

Methods

BeginTransaction()	Starts a database transaction.
BeginTransaction(IsolationLevel)	Starts a database transaction with the specified isolation level.
BeginTransaction(IsolationLevel, String)	Starts a database transaction with the specified isolation level and transaction name.
BeginTransaction(String)	Starts a database transaction with the specified transaction name.
ChangeDatabase(String)	Changes the current database for an open SqlConnection.
ChangePassword(String, SqlCredential, SecureString)	Changes the SQL Server password for the user indicated in the SqlCredential object.
ChangePassword(String, String)	Changes the SQL Server password for the user indicated in the connection string to the supplied new password.
ClearAllPools()	Empties the connection pool.
ClearPool(SqlConnection)	Empties the connection pool associated with the specified connection.
Close()	Closes the connection to the database. This is the preferred method of closing any open connection.
CreateCommand()	Creates and returns a SqlCommand object associated with the SqlConnection.
CreateObjRef(Type)	Creates an object that contains all the relevant information required to generate a proxy used to communicate with a remote object. (Inherited from MarshalByRefObject)
Dispose()	Releases all resources used by the Component. (Inherited from Component)
Dispose(Boolean)	Releases the unmanaged resources used by the Component and optionally releases the managed resources. (Inherited from Component)
EnlistDistributedTransaction(ITransaction)	Enlists in the specified transaction as a distributed transaction.
EnlistTransaction(Transaction)	Enlists in the specified transaction as a distributed transaction.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetLifetimeService()	Retrieves the current lifetime service object that controls the lifetime policy for this instance. (Inherited from MarshalByRefObject)
GetSchema()	Returns schema information for the data source of this SqlConnection. For more information about scheme, see SQL Server Schema Collections.
GetSchema(String)	Returns schema information for the data source of this SqlConnection using the specified string for the schema name.
GetSchema(String, String[])	Returns schema information for the data source of this SqlConnection using the specified string for the schema name and the specified string array for the restriction values.
GetService(Type)	Returns an object that represents a service provided by the Component or by its Container.

	(Inherited from Component)
GetType()	Gets the Type of the current instance. (Inherited from Object)
InitializeLifetimeService()	Obtains a lifetime service object to control the lifetime policy for this instance. (Inherited from MarshalByRefObject)
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
MemberwiseClone(Boolean)	Creates a shallow copy of the current MarshalByRefObject object. (Inherited from MarshalByRefObject)
Open()	Opens a database connection with the property settings specified by the ConnectionString.
OpenAsync(CancellationToken)	An asynchronous version of Open(), which opens a database connection with the property settings specified by the ConnectionString. The cancellation token can be used to request that the operation be abandoned before the connection timeout elapses. Exceptions will be propagated via the returned Task. If the connection timeout time elapses without successfully connecting, the returned Task will be marked as faulted with an Exception. The implementation returns a Task without blocking the calling thread for both pooled and non-pooled connections.
RegisterColumnEncryptionKeyStoreProviders(IDictionary<String,SqlColumnEncryptionKeyStoreProvider>)	Registers the column encryption key store providers.
ResetStatistics()	If statistics gathering is enabled, all values are reset to zero.
RetrieveStatistics()	Returns a name value pair collection of statistics at the point in time the method is called.
ToString()	Returns a String containing the name of the Component, if any. This method should not be overridden. (Inherited from Component)

Events

Disposed	Occurs when the component is disposed by a call to the Dispose() method. (Inherited from Component)
InfoMessage	Occurs when SQL Server returns a warning or informational message.

Explicit Interface Implementations

ICloneable.Clone()	Creates a new object that is a copy of the current instance.

Applies to

.NET Core

3.0, 2.2, 2.1, 2.0, 1.1, 1.0

.NET Framework

4.8, 4.7.2, 4.7.1, 4.7, 4.6.2, 4.6.1, 4.6, 4.5.2, 4.5.1, 4.5, 4.0, 3.5, 3.0, 2.0, 1.1

.NET Platform Extensions

3.0, 2.2, 2.1

Xamarin.Android

7.1

Xamarin.iOS

10.8

Xamarin.Mac

3.0

See also

- Connecting to a Data Source in ADO.NET
- SQL Server and ADO.NET
- ADO.NET Overview

Is this page helpful?

Yes No