Step 1. Configure a Connection String

For ASP.NET 2.0 applications, you should store connection strings in the <**connectionStrings**> section of the application's Web.config file. The connection string used with SQL authentication must include the user name and password of the database user, as shown here.

<configuration>

<connectionStrings>

<add name="MyDbConn"

connectionString="Server=MyServer; Database=pubs; User Id=MyUser; password= P@ssw0rd"

providerName="System.Data.SqlClient" />

</connectionStrings>

</configuration>

Step 2. Encrypt the Connection String

To help make sure that the SQL account credentials remain confidential, you should encrypt the connection string in the Web.config file. To do so, you use the Aspnet**\_**regiis utility with either the Windows Data Protection API (DPAPI) or RSA protected configuration providers.

The **DataProtectionConfigurationProvider** uses DPAPI and the **RSAProtectedConfigurationProvider** uses RSA public-key encryption. Because of the ease with which RSA keys can be exported, use the **RSAProtectedConfigurationProvider** if your application is deployed in a Web farm.

The following command shows how to encrypt the <**connectionStrings**> section of the Web.config file by using the **DataProtectionConfigurationProvider**:

**aspnet\_regiis -pe "connectionStrings" -app "/*MyWebsite*" -prov "DataProtectionConfigurationProvider"**

Where /*MyWebsite* is the virtual path to your ASP.NET application.

To use the encrypted connection string in your application, simply access the string value at run time as shown in the following example.

string dbConn = ConfigurationManager.ConnectionString["MyDbConn"].ToString();

ASP.NET automatically decrypts encrypted sections at run time.

**Note**   You can programmatically encrypt and decrypt connection strings and other sections of your configuration file by using the**System.Configuration.SectionInformation** class and the methods **ProtectSection** and **UnProtectSection**.

For more information about using the protected configuration providers to encrypt configuration file sections, see:

* [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using DPAPI](https://msdn.microsoft.com/en-us/library/ms998280.aspx)
* [How To: Encrypt Configuration Sections in ASP.NET 2.0 Using RSA](https://msdn.microsoft.com/en-us/library/ms998283.aspx)

Step 3. Configure SQL Server Security

To access a SQL Server database using SQL Server authentication, SQL Server must be configured for SQL authentication and you must create a login account.

**To verify the SQL Server authentication mode**

1. Start SQL Server Enterprise Manager.
2. Locate the SQL Server instance within the SQL Server group.
3. Right-click your SQL Server and click **Properties**.
4. Click the **Security** tab and make sure that authentication is set to **SQL Server and Windows**.

**To create a new SQL Server login**

1. Expand the **Security** folder in the SQL Server Enterprise Manager, right-click **Logins**, and click**New Login**.
2. Select the **SQL Server Authentication** option for **Authentication**.
3. Enter a login name and strong password. You should use a combination of letters, numbers, and non-alphanumeric characters for the password and use at least 8 characters total.

SQL Server can now authenticate the new login, but the login has no permission to access any of the user-defined databases.

**To grant database access**

1. Locate the **Users**folder for the database you want to grant access to, right-click, and click **New Database User**.
2. Select the **Login name** from the drop-down list and click **OK**. The database user name defaults to the same name.
3. Select **Database Roles**, right-click and then click **New Database Role**.

Avoid granting permissions to individual database users because the database user name may change and/or additional users may require the same permissions. Instead, grant permissions to database roles.

1. Enter a role name and select the **Database Role** type as **Standard Role**. Click the **Add** button, add the new user you created, and click **OK**.
2. Select the new role, right-click, and then click **Properties**.
3. Click **Permissions**. Locate the relevant stored procedures, and grant the EXEC (execute) permission to these objects.

By default, a new database role has no permission to read the base tables or views in the database and cannot execute stored procedures. It is best practice to grant permissions to stored procedures and views because then you can control what users can read and update.

As an alternative to using Enterprise Manager, you can run the following database script to perform the steps outlined above.

-- Create a new SQL Server Login

exec sp\_addlogin @loginame = 'MyUser', @passwd = 'P@ssw0rd'

-- Grant the SQL login access to your database.

-- Create a database user called WebAppUser to which the login is associated

Use YourDatabase

GO

exec sp\_grantdbaccess 'MyUser'

-- Create a user-defined database role.

exec sp\_addrole 'WebAppUserRole'

-- Add the database user to the new database role.

exec sp\_addrolemember 'WebAppUserRole', 'MyUser'

-- Grant the role execute permissions on the stored procedure called sprocname

grant execute on sprocname to WebAppUserRole

Step 4. Test Security Access

To test database access, create a test ASP.NET Web application and add the following .aspx page.

<%@ Page Language="C#" %>

<%@ Import Namespace="System.Data" %>

<%@ Import Namespace="System.Data.SqlClient" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<script runat="server">

protected void Page\_Load(object sender, EventArgs e)

{

using (SqlConnection cn = new SqlConnection(ConfigurationManager.ConnectionStrings["MyDbConn"].ToString()))

{

SqlCommand cmd = new SqlCommand("SELECT COUNT(\*) FROM authors", cn);

cn.Open();

SqlDataReader rdr = cmd.ExecuteReader(CommandBehavior.CloseConnection);

rdr.Read();

Response.Write(rdr[0].ToString()); //read a value

}

}

</script>

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>SQL Authentication</title>

</head>

<body/>

</html>

Add a Web.config file and add a connection string entry as described in Step 1. Build and run the application. If you have not specifically permitted SELECT access to the authors table, you will see an error message such as the following:

SELECT permission denied on object 'authors', database 'pubs', owner 'dbo'.

Step 5. Secure the Communications Channel to the Database

You might consider it necessary to encrypt the user name and data transmission, particularly if it contains sensitive application data between SQL Server and your ASP.NET application server. If your application is in a physically secured Internet data center, then this might not be necessary. In other situations, you might want to secure the communication channel to mitigate the risk posed by network eavesdropping.

There are two options for securing the data transmission:

* Internet Protocol Security (IPSec)
* Secure Sockets Layer (SSL) using a server certificate on SQL Server

IPSec

IPSec is a transport-level service that provides port filtering, authentication, encryption, and integrity for all traffic between two computers. IPSec is transparent to applications; therefore, it can be used to secure applications without requiring their modification. However, IPSec is not intended as a replacement for application-level security. Consider using IPSec to provide additional infrastructure security to secure the communication channel between your Web application and database server and to restrict the computers that can communicate with your database server. For example, you can help secure a database server by establishing a policy that permits requests only from a trusted client computer, such as an application or Web server. You can also restrict communication to specific IP protocols and TCP/UDP ports.

For detailed information about using IPSec, see:

* "How To: Use IPSec to Provide Secure Communications Between Two Servers," at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod32.asp](https://msdn.microsoft.com/en-us/library/cc402160.aspx)
* "How To: Use IPSec for Filtering Ports and Authentication," at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/HTUseIPSec.asp](https://msdn.microsoft.com/en-us/library/aa302366.aspx)

SSL

SSL uses cryptographic functions to provide an encrypted channel between client and server applications, and can be used to secure data transmission between a SQL Server and an ASP.NET application server. With SSL, each application can choose whether or not to create an encrypted communications channel; therefore, it offers more granularity than IPSec.

To configure SQL Server to use SSL, you must install a server certificate on the server, and the clients must have a root certificate authority that matches the server certificate.

For detailed information about how to install SSL, see:

* "How To: Use SSL to Secure Communication with SQL Server 2000," at [http://msdn.microsoft.com/library/default.asp?url=/library/en-us/secmod/html/secmod33.asp](https://msdn.microsoft.com/en-us/library/cc402161.aspx)
* "How To: Enable SSL Encryption for SQL Server 2000 with Microsoft Management Console," at <http://support.microsoft.com/default.aspx?scid=kb;en-us;316898>
* "How to Enable SSL Encryption for SQL Server 2000 with Certificate Server," at <http://support.microsoft.com/default.aspx?scid=276553>

You can choose to configure SQL Server to force all clients to use SSL by running the SQL Server Network Utility (Svrnetcn.exe**)** tool and checking the **Force protocol encryption** option for TCP/IP. However, you may prefer to choose when to use SSL at an application level on a per-connection basis. For example, your application can use an unencrypted connection when it is not handling sensitive data, and only use an encrypted connection when necessary.

If you are using the ADO.NET SQL Server managed data provider and a **SqlConnection** object to connect to SQL Server, then you can enable encryption by adding the **encrypt=true** parameter to the connection string as shown here.

<configuration>

<connectionStrings>

<add name="MyDbConn"

connectionString="Server=MyServer;Database=pubs;User Id=MyUser; password=P@ssw0rd;encrypt=true"

providerName="System.Data.SqlClient" />

</connectionStrings>

</configuration>

using System;

using System.Data;

using System.Data.SqlClient;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace SqlTest\_CSharp

{

class Program

{

static void Main(string[] args)

{

*// Create the connection to the resource!*

*// This is the connection, that is established and*

*// will be available throughout this block.*

using (SqlConnection conn = new SqlConnection())

{

*// Create the connectionString*

*// Trusted\_Connection is used to denote the connection uses Windows Authentication*

conn.ConnectionString = "Server=[server\_name];Database=[database\_name];Trusted\_Connection=true";

conn.Open();

*// Create the command*

SqlCommand command = new SqlCommand("SELECT \* FROM TableName WHERE FirstColumn = @0", conn);

*// Add the parameters.*

command.Parameters.Add(new SqlParameter("0", 1));

*/\* Get the rows and display on the screen!*

*\* This section of the code has the basic code*

*\* that will display the content from the Database Table*

*\* on the screen using an SqlDataReader. \*/*

using (SqlDataReader reader = command.ExecuteReader())

{

Console.WriteLine("FirstColumn\tSecond Column\t\tThird Column\t\tForth Column\t");

while (reader.Read())

{

Console.WriteLine(String.Format("{0} \t | {1} \t | {2} \t | {3}",

reader[0], reader[1], reader[2], reader[3]));

}

}

Console.WriteLine("Data displayed! Now press enter to move to the next section!");

Console.ReadLine();

Console.Clear();

*/\* Above code was used to display the data from the Database table!*

*\* This following section explains the key features to use*

*\* to add the data to the table. This is an example of another*

*\* SQL Command (INSERT INTO), this will teach the usage of parameters and connection.\*/*

Console.WriteLine("INSERT INTO command");

*// Create the command, to insert the data into the Table!*

*// this is a simple INSERT INTO command!*

SqlCommand insertCommand = new SqlCommand("INSERT INTO TableName (FirstColumn, SecondColumn, ThirdColumn, ForthColumn) VALUES (@0, @1, @2, @3)", conn);

*// In the command, there are some parameters denoted by @, you can*

*// change their value on a condition, in my code they're hardcoded.*

insertCommand.Parameters.Add(new SqlParameter("0", 10));

insertCommand.Parameters.Add(new SqlParameter("1", "Test Column"));

insertCommand.Parameters.Add(new SqlParameter("2", DateTime.Now));

insertCommand.Parameters.Add(new SqlParameter("3", false));

*// Execute the command, and print the values of the columns affected through*

*// the command executed.*

Console.WriteLine("Commands executed! Total rows affected are " + insertCommand.ExecuteNonQuery());

Console.WriteLine("Done! Press enter to move to the next step");

Console.ReadLine();

Console.Clear();

*/\* In this section there is an example of the Exception case*

*\* Thrown by the SQL Server, that is provided by SqlException*

*\* Using that class object, we can get the error thrown by SQL Server.*

*\* In my code, I am simply displaying the error! \*/*

Console.WriteLine("Now the error trial!");

*// try block*

try

{

*// Create the command to execute! With the wrong name of the table (Depends on your Database tables)*

SqlCommand errorCommand = new SqlCommand("SELECT \* FROM someErrorColumn", conn);

*// Execute the command, here the error will pop up!*

*// But since we're catching the code block's errors, it will be displayed inside the console.*

errorCommand.ExecuteNonQuery();

}

*// catch block*

catch (SqlException er)

{

*// Since there is no such column as someErrorColumn (Depends on your Database tables)*

*// SQL Server will throw an error.*

Console.WriteLine("There was an error reported by SQL Server, " + er.Message);

}

}

*// Final step, close the resources flush dispose them. ReadLine to prevent the console from closing.*

Console.ReadLine();

}

}

}

# Beginners guide to accessing SQL Server through C#

[**Matt Newman**](http://www.codeproject.com/script/Membership/View.aspx?mid=12023), 22 Aug 2004 [CPOL](http://www.codeproject.com/info/cpol10.aspx)http://www.codeproject.com/App_Themes/CodeProject/Img/read32.png 2.3Mhttp://www.codeproject.com/App_Themes/CodeProject/Img/bookmark32.png 368

|  |  |
| --- | --- |
| http://dj9okeyxktdvd.cloudfront.net/script/Ratings/Images/stars-fill-lg.png  http://dj9okeyxktdvd.cloudfront.net/script/Ratings/Images/stars-empty-lg.png | 4.56 (192 votes) |
|  | |  |  |  | | --- | --- | --- | | Rate this: | [vote 1vote 2vote 3vote 4vote 5](http://www.codeproject.com/Articles/4416/Beginners-guide-to-accessing-SQL-Server-through-C) |  | |

A beginners guide to accessing a SQL or MSDE Server with C#

## Introduction

In this article I plan to demonstrate how to insert and read data from a SQL Server or MSDE database. This code should work on both SQL Server , I am using 2000, and MSDE. I am using Visual Studio 2002, but this should work with Visual Studio 2003, Web Matrix, and the command line SDK. This code should work with both C# applications and C# web applications and webservices. This code does not compile on the FreeBSD with[Rotor](http://www.microsoft.com/downloads/details.aspx?FamilyId=3A1C93FA-7462-47D0-8E56-8DD34C6292F0&displaylang=en) [[^](http://www.microsoft.com/downloads/details.aspx?FamilyId=3A1C93FA-7462-47D0-8E56-8DD34C6292F0&displaylang=en)].

## Background

Part of my current project required me too store and retrieve information from a database. I decided to use C# as my target language since I am currently reading [Inside C# Second Edition](http://www.microsoft.com/mspress/books/5027.asp) [[^](http://www.microsoft.com/mspress/books/5027.asp)] by [Tom Archer](http://www.codeproject.com/script/profile/whos_who.asp?id=232) [[^](http://www.codeproject.com/script/profile/whos_who.asp?id=232)], which by the way is a must have book. However I could not find any examples that were clear and just generic accesing SQL Server with C#.

## Using the code

I did not include a sample application because the code provide within the article can really be dropped in and should work with no problem. Also through out the article I will refer to SQL Server, MSDE is a free version of SQL Server that does not have some of the GUI tools and has a few other limits such as database size. This code will work on both without problem.

## Making the Love Connection

There is no real voodoo magic to creating a connection to a SQL Server assuming it is properly setup, which I am not going to go into in this article, in fact .NET has made working with SQL quite easy. First step is add the SQL Client namespace:

Hide   Copy Code

using System.Data.SqlClient;

Then we create a SqlConnection and specifying the connection string.

Hide   Copy Code

SqlConnection myConnection = new SqlConnection("user id=username;" +

"password=password;server=serverurl;" +

"Trusted\_Connection=yes;" +

"database=database; " +

"connection timeout=30");

Note: line break in connection string is for formatting purposes only

### SqlConnection.ConnectionString

The connection string is simply a compilation of options and values to specify how and what to connect to. Upon investigating the Visual Studio .NET help files I discovered that several fields had multiple names that worked the same, like Password and Pwd work interchangeably. I have not included all of the options forSqlConnection.ConnectionString at this time. As I get a chance to test and use these other options I will include them in the article.

#### User ID

The User ID is used when you are using SQL Authentication. In my experience this is ignored when using a Trusted\_Connection, or Windows Authentication. If the username is associated with a password Password or Pwdwill be used.

"user id=userid;"

#### Password or Pwd

The password field is to be used with the User ID, it just wouldn't make sense to log in without a username, just a password. Both Password and Pwd are completely interchangeable.

"Password=validpassword;"**-or-**  
"Pwd=validpassword;"

#### Data Source or Server or Address or Addr or Network Address

Upon looking in the MSDN documentation I found that there are several ways to specify the network address. The documentation mentions no differences between them and they appear to be interchangeable. The address is an valid network address, for brevity I am only using the localhost address in the examples.

"Data Source=localhost;"  
**-or-**  
"Server=localhost;"  
**-or-**  
"Address=localhost;"**-or-**"Addr=localhost;"  
**-or-**"Network Address=localhost;"

#### Integrated Sercurity or Trusted\_Connection

Integrated Security and Trusted\_Connection are used to specify wheter the connnection is secure, such as Windows Authentication or SSPI. The recognized values are true, false, and sspi. According to the MSDN documentation sspi is equivalent to true. **Note:** I do not know how SSPI works, or affects the connection.

#### Connect Timeout or Connection Timeout

These specify the time, in seconds, to wait for the server to respond before generating an error. The default value is 15 (seconds).

"Connect Timeout=10;"**-or-**  
"Connection Timeout=10;"

#### Initial Catalog or Database

Initial Catalog and Database are simply two ways of selecting the database associated with the connection.

"Inital Catalog=main;"  
**-or-**  
"Database=main;"

#### Network Library or Net

The Network Library option is essential if your are communicating with the server on a protocl other than TCP/IP. The default value for Network Library is dbmssocn, or TCP/IP. The following options are available: dbnmpntw(Named Pipes), dbmsrpcn (Multiprotocol), dbmsadsn (Apple Talk), dbmsgnet (VIA), dbmsipcn (Shared Memory), and dbmsspxn (IPX/SPX), and dbmssocn (TCP/IP). And as before Network Library and Net can be user interchangably. **Note:** The corresponding network protocol **must** be installed on the system to which you connect.

## SqlConnection.Open()

This is the last part of getting connected and is simply executed by the following (remember to make sure your connection has a connection string first):

Hide   Copy Code

try

{

myConnection.Open();

}

catch(Exception e)

{

Console.WriteLine(e.ToString());

}

SqlConnection.Open() is a void function and does not return an error but throws an exception so remember to put it in a try/catch brace. rather than having the program explode in front of the user.

## Command thee

SQL commands are probably the most difficult part of using an SQL database, but the .NET framework has wrapped up everything up nicely and takes most of the guess work out.

### SqlCommand

Any guesses on what SqlCommand is used for? If you guessed for SQL commands then you are right on. AnSqlCommand needs at least two things to operate. A command string, and a connection. First we'll look at the connection requirement. There are two ways to specify the connection, both are illustrated below:

Hide   Copy Code

SqlCommand myCommand = new SqlCommand("Command String", myConnection);

*// - or -*

myCommand.Connection = myConnection;

The connection string can also be specified both ways using the SqlCommand.CommandText property. Now lets look at our first SqlCommand. To keep it simple it will be a simple INSERT command.

Hide   Copy Code

SqlCommand myCommand= new SqlCommand("INSERT INTO table (Column1, Column2) " +

"Values ('string', 1)", myConnection);

*// - or -*

myCommand.CommandText = "INSERT INTO table (Column1, Column2) " +

"Values ('string', 1)";

Now we will take a look at the values . table is simply the table within the database. Column1 and Column2 are merely the names of the columns. Within the values section I demonstrated how to insert a string type and anint type value. The string value is placed in single quotes and as you can see an integer is just passed as is. The final step is to execute the command with:

Hide   Copy Code

myCommand.ExecuteNonQuery();

### SqlDataReader

Inserting data is good, but getting the data out is just as important. Thats when the SqlDataReader comes to the rescue. Not only do you need a data reader but you need a SqlCommand. The following code demonstrates how to set up and execute a simple reader:

Hide   Copy Code

try

{

SqlDataReader myReader = null;

SqlCommand myCommand = new SqlCommand("select \* from table",

myConnection);

myReader = myCommand.ExecuteReader();

while(myReader.Read())

{

Console.WriteLine(myReader["Column1"].ToString());

Console.WriteLine(myReader["Column2"].ToString());

}

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

As you can see the SqlDataReader does not access the database, it merely holds the data and provides an easy interface to use the data. The SqlCommand is fairly simple, table is the table your are going to read from.Column1 and Column2 are just the columns as in the table. Since there is a very high probability your will be reading more than one line a while loop is required to retrieve all of the records. And like always you want totry it and catch it so you don't break it.

### SqlParameter

There is a small problem with using SqlCommand as I have demonstrated, it leaves a large security hole. For example, with the way previously demonstrated your command string would be constructed something like this if you were to get input from a user:

Hide   Copy Code

SqlCommand myCommand = new SqlCommand(

"SELECT \* FROM table WHERE Column = " + input.Text, myConnection);

Its all fine and dandy if the user puts in correct syntax, however, what happens if the user puts value1, DROP table. Best case scenario it will cause an exception (I haven't checked to see what this example will do but it demonstrates a point), worst case you can kiss your table goodbye. You could parse all user input and strip out anything that could cause problems OR you could use an SqlParameter. Now the SqlParameter class is pretty big, but I will just show you a basic parameter usage. Basically you need three things to create a parameter. A name, data type, and size. (note for some data types you will want to leave off the size, such as Text).

Hide   Copy Code

SqlParameter myParam = new SqlParameter("@Param1", SqlDbType.VarChar, 11);

myParam.Value = "Garden Hose";

SqlParameter myParam2 = new SqlParameter("@Param2", SqlDbType.Int, 4);

myParam2.Value = 42;

SqlParameter myParam3 = new SqlParameter("@Param3", SqlDbType.Text);

myParam.Value = "Note that I am not specifying size. " +

"If I did that it would trunicate the text.";

It is naming convention, it might be required I'm not sure, to name all parameters starting with the @ symbol. Now how do you use a parameter? Will its pretty easy as the following code shows.

Hide   Copy Code

SqlCommand myCommand = new SqlCommand(

"SELECT \* FROM table WHERE Column = @Param2", myConnection);

myCommand.Parameters.Add(myParam2);

Now this keeps a rogue user from high-jacking your command string. This isn't all there is to parameters if you want to learn more advanced topics a good place to start is [here](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemdatasqlclientsqlparameterclasstopic.asp)[[^](http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemdatasqlclientsqlparameterclasstopic.asp)].

## Don't forget to close up when your done!

Closing a connection is just as easy as opening it. Just callSqlConnection.Close() but remember to put it in try/catch because like SqlConnection.Open() it does not return errors but throws an exception instead.

Hide   Copy Code

try

{

myConnection.Close();

}

catch(Exception e)

{

Console.WriteLine(e.ToString());

}

## When good connections go bad

The trusted connection had always been a mystery to me, I had never figured why IIS and SQL server never seemed to get along. Fortunately Pete (moredip) pointed out a helpful section of the documentation. To make it more simple I have decided to add it to this article. I am going to split this into 2 different sections. IIS 6, and other versions of IIS. To get started your going to want to make sure osql.exe is in your system path, or find it. It should be located wherever your SQL Server 2000 server/client tools directory. On my system it is something like this: %Install Directory%\80\Tools\BINN\.For simplicity I will use psuedo-variables in the examples so as not to create confusion. For example a psuedo-variable will look like this: %VARIABLE%. The server will be referred to as %SERVER%\%INSTANCE%. If you aren't using any instance names it can be just %SERVER%, (local)if the server is the local machine. If you are instance names it would be something likeServerName\ServerInstance etc etc. I will also be using %DATABASE% to refer to the database name.

### IIS 6 on Windows 2003 Server

I know this will work on IIS 6 with Windows 2003 Server because I have done it and that is currently the only OS with IIS 6. On IIS 6 the ASP.NET process runs under the account 'NT AUTHORITY\NETWORK SERVICE'.

Hide   Copy Code

osql -E -S %SERVER%\%INSTANCE% -Q "sp\_grantlogin 'NT AUTHORITY\NETWORK SERVICE'"

Now our ASP.NET application will be able to log into the server. Now all thats left is to grant access to the databases.

Hide   Copy Code

osql -E -S %SERVER%\%INSTANCE% -d %DATABASE% -Q

"sp\_grantdbaccess 'NT AUTHORITY\NETWORK SERVICE'"

osql -E -S %SERVER%\%INSTANCE% -d %DATABASE% -Q

"sp\_addrolemember 'db\_owner', 'NT AUTHORITY\NETWORK SERVICE'"

These 2 lines will add access to one of the databases. So if you want to add access to another database just change %DATABASE% and run both lines.

### IIS 5.1

This should work on all other IIS 5.1 (possibly other versions) combinations. The only difference between IIS 5.1 and IIS 6 is the account the ASP.NET process runs under. IIS 5.1 runs under a %MACHINENAME%\ASPNET where%MACHINENAME% is the machine name.

Hide   Copy Code

osql -E -S %SERVER%\%INSTANCE% -Q "sp\_grantlogin '%MACHINENAME%\ASPNET'"

Now our ASP.NET application will be able to log into the server. Now all thats left is to grant access to the databases.

Hide   Copy Code

osql -E -S %SERVER%\%INSTANCE% -d %DATABASE%

-Q "sp\_grantdbaccess '%MACHINENAME%\ASPNET'"

osql -E -S %SERVER%\%INSTANCE% -d %DATABASE%

-Q "sp\_addrolemember 'db\_owner', '%MACHINENAME%\ASPNET'"

These 2 lines will add access to one of the databases. So if you want to add access to another database just change %DATABASE% and run both lines.

# C# SQL Server Connection

C# SQl Server ConnectionYou can connect your C# application to data in a SQL Server database using the .NET Framework Data Provider for SQL Server. The first step in a C# application is to create an instance of the Server object and to establish its connection to an instance of Microsoft SQL Server.

The SqlConnection Object is Handling the part of physical communication between the C# application and the SQL Server Database . An instance of the SqlConnection class in C# is supported the Data Provider for SQL Server Database. The SqlConnection instance takes Connection String as argument and pass the value to the Constructor statement.

**Sql Server connection string**

connetionString="Data Source=ServerName;

Initial Catalog=DatabaseName;User ID=UserName;Password=Password"

If you have a named instance of SQL Server, you'll need to add that as well.

"Server=localhost\sqlexpress"

When the connection is established , SQL Commands will execute with the help of the Connection Object and retrieve or manipulate the data in the database. Once the Database activities is over , Connection should be closed and release the Data Source resources .

cnn.Close();

The Close() method in SqlConnection Class is used to close the Database Connection. The Close method rolls back any pending transactions and releases the Connection from the SQL Server Database.

A Sample C# Program that connect SQL Server using connection string.

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString = null;

SqlConnection cnn ;

connetionString = "Data Source=ServerName;Initial Catalog=DatabaseName;User ID=UserName;Password=Password"

cnn = new SqlConnection(connetionString);

try

{

cnn.Open();

MessageBox.Show ("Connection Open ! ");

cnn.Close();

}

catch (Exception ex)

{

MessageBox.Show("Can not open connection ! ");

}

}

}

}

**Connect via an IP address**

connetionString="Data Source=IP\_ADDRESS,PORT;

Network Library=DBMSSOCN;Initial Catalog=DatabaseName;

User ID=UserName;Password=Password"

1433 is the default port for SQL Server.

**Trusted Connection from a CE device**

connetionString="Data Source=ServerName;

Initial Catalog=DatabaseName;Integrated Security=SSPI;

User ID=myDomain\UserName;Password=Password;

This will only work on a CE device

**Connecting to SQL Server using windows authentication**

"Server= localhost; Database= employeedetails;

Integrated Security=SSPI;"

A sample c# program that demonstrate how to execute sql statement and read data from SQL server.

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString = null;

SqlConnection connection ;

SqlCommand command ;

string sql = null;

SqlDataReader dataReader ;

connetionString = "Data Source=ServerName;Initial Catalog=DatabaseName;User ID=UserName;Password=Password";

sql = "Your SQL Statement Here , like Select \* from product";

connection = new SqlConnection(connetionString);

try

{

connection.Open();

command = new SqlCommand(sql, connection);

dataReader = command.ExecuteReader();

while (dataReader.Read())

{

MessageBox.Show(dataReader.GetValue(0) + " - " + dataReader.GetValue(1) + " - " + dataReader.GetValue(2));

}

dataReader.Close();

command.Dispose();

connection.Close();

}

catch (Exception ex)

{

MessageBox.Show("Can not open connection ! ");

}

}

}

}

A sample C# program that perform Data Manipulation tasks like Insert , Update , Delete etc. also perform by the ExecuteNonQuery() of SqlCommand Object.

[**Next :  C# OLEDB Connection**](http://csharp.net-informations.com/data-providers/csharp-oledb-connection.htm)

[**Download Source Code**](http://csharp.net-informations.com/data-providers/files/download/csharp-sql-server-connection_download.htm)

[**Print Source Code**](http://csharp.net-informations.com/data-providers/files/print/csharp-sql-server-connection_print.htm)

using System;

using System.Windows.Forms;

using System.Data.SqlClient;

namespace WindowsApplication1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string connetionString = null;

SqlConnection connection ;

SqlCommand command ;

string sql = null;

connetionString = "Data Source=ServerName;Initial Catalog=DatabaseName;User ID=UserName;Password=Password";

sql = "Your SQL Statemnt Here";

connection = new SqlConnection(connetionString);

try

{

connection.Open();

command = new SqlCommand(sql, connection);

command.ExecuteNonQuery();

command.Dispose();

connection.Close();

MessageBox.Show (" ExecuteNonQuery in SqlCommand executed !!");

}

catch (Exception ex)

{

MessageBox.Show("Can not open connection ! ");

}

}

}

}

|  |  |  |  |
| --- | --- | --- | --- |
| <asp:Textbox id="TxtBox1" runat="server" />  <asp:Textbox id="TxtBox2" runat="server" />  <asp:Button id="Button1" runat="server" Onclick="Button1\_Click" />  and the codebehind:  protected void Button1\_Click(Object sender,EventArgs e)  {  string Text1=TextBox1.Text;  string Text2=TextBox2.Text;  string connectionString="Data Source=servername;InitialCatalog=DataBaseNameUserID=sa; Password=YourPassword;"  SqlConnection sqlConnection=new SqlConnection(connectionString);  string insertStatement="INSERT INTO TableName(column1,column2) VALUES Txtb1+","+Txtb2";  SqlCommand sqlCommand=new SqlCommand(insertStatement,sqlConnection);  try  {  sqlConnection.Open();  sqlCommand.ExecuteNonQuery();  }  finally  {  sqlConnection.Close();  }  }  I have created the data.mdf file in it i have created table containing fields but i am getting Connection has not established error can any one help me to resolve my problem???  [c#](http://stackoverflow.com/questions/tagged/c%23) [asp.net](http://stackoverflow.com/questions/tagged/asp.net) [sql](http://stackoverflow.com/questions/tagged/sql) [database](http://stackoverflow.com/questions/tagged/database)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/q/13974308)[improve this question](http://stackoverflow.com/posts/13974308/edit) | [edited Dec 20 '12 at 14:36](http://stackoverflow.com/posts/13974308/revisions)  [[https://www.gravatar.com/avatar/da2ed5086124b0ddd01b9da494a508ce?s=32&d=identicon&r=PG](http://stackoverflow.com/users/124966/nicholas-murray)](http://stackoverflow.com/users/124966/nicholas-murray)  [Nicholas Murray](http://stackoverflow.com/users/124966/nicholas-murray) **6,297**73764 | asked Dec 20 '12 at 14:28  [[https://www.gravatar.com/avatar/7d6a38ae3f889fea1ee374754b8374c0?s=32&d=identicon&r=PG](http://stackoverflow.com/users/1918566/user1918566)](http://stackoverflow.com/users/1918566/user1918566)  [user1918566](http://stackoverflow.com/users/1918566/user1918566) **28**115 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | | 2 |  | | Don't use the sa account in applications! This code is incredibly vulnerable to a sql injection attack, not to mention the fact that the connection string is malformed, and the inserStatment line will not compile. – [asawyer](http://stackoverflow.com/users/426894/asawyer) [Dec 20 '12 at 14:31](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280432_13974308) | | |  |  | | --- | --- | |  |  | | Sorry..,I didn't get you could you please express me about sa??? –  [user1918566](http://stackoverflow.com/users/1918566/user1918566) [Dec 20 '12 at 14:34](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280509_13974308) | | |  |  | | --- | --- | | 1 |  | | You are using the administrator account. The one that has full access to the sql server. All you need is an account with write access to the one table. Always use the minimum permissions possible. –  [asawyer](http://stackoverflow.com/users/426894/asawyer) [Dec 20 '12 at 14:35](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280547_13974308) | | |  |  | | --- | --- | |  |  | | Proper research is a must: [connectionstrings.com](http://www.connectionstrings.com/) –  [Ramhound](http://stackoverflow.com/users/261581/ramhound) [Dec 20 '12 at 14:36](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280591_13974308) | | |  |  | | --- | --- | |  |  | | kk if you don't mind can you post the modified code by taking small example taking the two text fields and after clicking the submit button the data will be stored into the database(i'm using visual studio 2010 and sql server 2008 along with it based on my requirement can you please post the alternate code) –  [user1918566](http://stackoverflow.com/users/1918566/user1918566)[Dec 20 '12 at 14:38](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280640_13974308) |   add a comment |

4 Answers

[active](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c?answertab=active#tab-top)[oldest](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c?answertab=oldest#tab-top)[votes](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c?answertab=votes#tab-top)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote5down vote | This connection string is wrong.  string connectionString="Data Source=servername;InitialCatalog=DataBaseNameUserID=sa;" +  "Password=YourPassword;"  should be  string connectionString="Data Source=servername;InitialCatalog=DataBaseName; UserID=sa;" +  "Password=YourPassword;"  However, NEVER use the sa user in your applications. Create a new user with Management Studio or better use Integrated Security.  Also the remainder of your code should be written in this way  using(SqlConnection sqlConnection=new SqlConnection(connectionString))  {  string insertStatement="INSERT INTO TableName(column1,column2) VALUES (@col1, @col2)";  SqlCommand sqlCommand=new SqlCommand(insertStatement,sqlConnection);  sqlCommand.Parameters.AddWithValue("@col1", Text1);  sqlCommand.Parameters.AddWithValue("@col2", Text2);  sqlConnection.Open();  sqlCommand.ExecuteNonQuery();  }   * Use the [using statement](http://msdn.microsoft.com/en-us/library/yh598w02%28v=VS.80%29.aspx) to correctly close and dispose your SqlConnection * Use [parametrized query](http://www.c-sharpcorner.com/uploadfile/puranindia/parameterized-query-and-sql-injection-attacks/) in your sql commands to avoid parsing problems an [Sql Injection Attacks](http://www.unixwiz.net/techtips/sql-injection.html)  |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/a/13974424)[improve this answer](http://stackoverflow.com/posts/13974424/edit) | [edited Dec 20 '12 at 14:40](http://stackoverflow.com/posts/13974424/revisions) | answered Dec 20 '12 at 14:33  [[http://i.stack.imgur.com/cjKiP.png?s=32&g=1](http://stackoverflow.com/users/1197518/steve)](http://stackoverflow.com/users/1197518/steve)  [Steve](http://stackoverflow.com/users/1197518/steve) **110k**1054110 | |
|  | |  |  |  |  | | --- | --- | --- | --- | | |  |  | | --- | --- | |  |  | | Beat me to the re-write too. I always enclose the SqlCommand in a using block as well. –  [Slippery Pete](http://stackoverflow.com/users/1718413/slippery-pete) [Dec 20 '12 at 14:42](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280761_13974424) | | |  |  | | --- | --- | |  |  | | Yes, should be done also on SqlCommand, but it is on the SqlConnection that the problem is big –  [Steve](http://stackoverflow.com/users/1197518/steve)[Dec 20 '12 at 14:44](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19280834_13974424) | | |  |  | | --- | --- | |  |  | | after clicking the submit button based on the above given code there is no response for that code – [user1918566](http://stackoverflow.com/users/1918566/user1918566) [Dec 20 '12 at 15:24](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19282161_13974424) | | |  |  | | --- | --- | |  |  | | I don't understand. What do you mean 'there is no response'? The code is not executed?, the record is not written to the database? do you expect any visual feedback of code execution? Please explain –  [Steve](http://stackoverflow.com/users/1197518/steve) [Dec 20 '12 at 15:26](http://stackoverflow.com/questions/13974308/how-to-establish-databasesql-connection-using-c#comment19282236_13974424) |   add a comment |

[](http://careers.stackoverflow.com/)

|  |  |  |  |
| --- | --- | --- | --- |
| up vote0down vote | This code does not compile. You need to add a semicolon to this line:  string connectionString="Data Source=servername;InitialCatalog=DataBaseNameUserID=sa; Password=YourPassword;"  Like so:  string connectionString="Data Source=servername;InitialCatalog=DataBaseNameUserID=sa; Password=YourPassword;";  You also need to change this line:  string insertStatement="INSERT INTO TableName(column1,column2) VALUES Txtb1+","+Txtb2";  To this:  string insertStatement = "INSERT INTO TableName(column1,column2) VALUES " + Text1 + ", " + Text2;   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/13974466)[improve this answer](http://stackoverflow.com/posts/13974466/edit) | answered Dec 20 '12 at 14:36  [[https://www.gravatar.com/avatar/7962995b50a355cd34e46d990621b91d?s=32&d=identicon&r=PG&f=1](http://stackoverflow.com/users/1718413/slippery-pete)](http://stackoverflow.com/users/1718413/slippery-pete)  [Slippery Pete](http://stackoverflow.com/users/1718413/slippery-pete) **2,108**139 | |
|  | add a comment |
| up vote0down vote | The connection string itself may be the issue, windows has an easy way to create a valid connection string that you can test and then just copy it to your app. I wrote about it a couple of months ago [here](http://www.megustaulises.com/2012/04/how-to-create-connection-string.html):  Tips for your app:   * As pointed by asawyer, don't use the sa account, create a login with access to the database(s) you need. * Use the [using](http://msdn.microsoft.com/en-us/library/yh598w02%28v=vs.80%29.aspx) statement   example:  using (SqlConnection connection = new SqlConnection(connectionString))  {  //here you open and execute the command  }   * User parameterized queries to avoid sql injection. [Here is a good article](http://www.codinghorror.com/blog/2005/04/give-me-parameterized-sql-or-give-me-death.html).  |  |  | | --- | --- | | [share](http://stackoverflow.com/a/13974551)[improve this answer](http://stackoverflow.com/posts/13974551/edit) | answered Dec 20 '12 at 14:40  [[https://www.gravatar.com/avatar/3eb352bda735a857b5cf25d2cc755a20?s=32&d=identicon&r=PG](http://stackoverflow.com/users/620448/ulises)](http://stackoverflow.com/users/620448/ulises)  [Ulises](http://stackoverflow.com/users/620448/ulises) **6,237**31533 | |
|  | add a comment |

|  |  |  |  |
| --- | --- | --- | --- |
| up vote0down vote | use the connection string as follows:  connectionString="Data Source=ServerAddress;Initial Catalog=DataBaseName;Integrated Security=SSPI;User ID=Domain\Username;Password=Password;"   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/13975484)[improve this answer](http://stackoverflow.com/posts/13975484/edit)  What is ADO.NET?   * ADO.NET is a part of the .NET Framework * ADO.NET consists of a set of classes used to handle data access * ADO.NET is entirely based on XML * ADO.NET has, unlike ADO, no Recordset object   Create a Database Connection  We are going to use the Northwind database in our examples.  First, import the "System.Data.OleDb" namespace. We need this namespace to work with Microsoft Access and other OLE DB database providers. We will create the connection to the database in the Page\_Load subroutine. We create a dbconn variable as a new OleDbConnection class with a connection string which identifies the OLE DB provider and the location of the database. Then we open the database connection:  <%@ Import Namespace="System.Data.OleDb" %>  <script runat="server"> sub Page\_Load dim dbconn dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; data source=" & server.mappath("northwind.mdb")) dbconn.Open() end sub </script>  **Note:** The connection string must be a continuous string without a line break!  Create a Database Command  To specify the records to retrieve from the database, we will create a dbcomm variable as a new OleDbCommand class. The OleDbCommand class is for issuing SQL queries against database tables:  Ads by AllSaver[Ad Options](http://luu.lightquartrate.com/sd/apps/adinfo-1.1-p/index.html?bj1BbGxTYXZlciZoPWx1dS5saWdodHF1YXJ0cmF0ZS5jb20mYz1ncmVlbiZvPWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZkPSZ0PSZhPTkxMDAmcz0xMDA1Jnc9d3d3Lnczc2Nob29scy5jb20mb291PWh0dHA6Ly9kZmcuZGV2bWFuYWdlb3B0aW9ucy5jb20vb3B0X291dC8xMSZiPTEmcmQ9JnJpPQ==)  <%@ Import Namespace="System.Data.OleDb" %>  <script runat="server"> sub Page\_Load dim dbconn,sql,dbcomm dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; data source=" & server.mappath("northwind.mdb")) dbconn.Open() sql="SELECT \* FROM customers" dbcomm=New OleDbCommand(sql,dbconn) end sub </script>  Create a DataReader  The OleDbDataReader class is used to read a stream of records from a data source. A DataReader is created by calling the ExecuteReader method of the OleDbCommand object:  <%@ Import Namespace="System.Data.OleDb" %>  <script runat="server"> sub Page\_Load dim dbconn,sql,dbcomm,dbread dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; data source=" & server.mappath("northwind.mdb")) dbconn.Open() sql="SELECT \* FROM customers" dbcomm=New OleDbCommand(sql,dbconn) dbread=dbcomm.ExecuteReader() end sub </script>  Bind to a Repeater Control  Then we bind the DataReader to a Repeater control:  Example  <%@ Import Namespace="System.Data.OleDb" %>  <script runat="server"> sub Page\_Load dim dbconn,sql,dbcomm,dbread dbconn=New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0; data source=" & server.mappath("northwind.mdb")) dbconn.Open() sql="SELECT \* FROM customers" dbcomm=New OleDbCommand(sql,dbconn) dbread=dbcomm.ExecuteReader() customers.DataSource=dbread customers.DataBind() dbread.Close() dbconn.Close() end sub </script>  <html> <body>  <form runat="server"> <asp:Repeater id="customers" runat="server">  <HeaderTemplate> <table border="1" width="100%"> <tr> <th>Companyname</th> <th>Contactname</th> <th>Address</th> <th>City</th> </tr> </HeaderTemplate>  <ItemTemplate> <tr> <td><%#Container.DataItem("companyname")%></td> <td><%#Container.DataItem("contactname")%></td> <td><%#Container.DataItem("address")%></td> <td><%#Container.DataItem("city")%></td> </tr> </ItemTemplate>  <FooterTemplate> </table> </FooterTemplate>  </asp:Repeater> </form>  </body> </html>  [**Show example »**](http://www.w3schools.com/aspnet/showaspx.asp?filename=demo_dbconn_repeater)  Close the Database Connection  Always close both the DataReader and database connection after access to the database is no longer required:  dbread.Close() dbconn.Close() | answered Dec 20 '12 at 15:33  [[https://www.gravatar.com/avatar/72dbd1c1ee930a6170f08042ca42edfb?s=32&d=identicon&r=PG](http://stackoverflow.com/users/1632305/rashedul-rubel)](http://stackoverflow.com/users/1632305/rashedul-rubel)  [Rashedul.Rubel](http://stackoverflow.com/users/1632305/rashedul-rubel) **782**31 | |