# How to connect to a database and run a command by using ADO.NET 2005 and Visual C# 2005 or by using ADO.NET and Visual C# .NET

This article was previously published under Q306636

For a Microsoft Visual Basic .NET version of this article, see 301075.

For a Microsoft Visual J# .NET version of this article, see 322045.

This article refers to the following Microsoft .NET Framework Class Library namespaces:

- **System.Data**
- **System.Data.SqlClient**

## SUMMARY

This step-by-step article shows you how to use ADO.NET 2005 or ADO.NET to connect to a database and to run a command.

### Requirements

The following list outlines the recommended hardware, software, network infrastructure, and service packs that you need:

- Microsoft Windows Server 2003, Microsoft Windows 2000 Professional, Windows 2000 Server, Windows 2000 Advanced Server, or Microsoft Windows NT 4.0 Server
- Microsoft Visual Studio 2005 or Microsoft Visual Studio .NET

This article assumes that you are familiar with the following topics:

- Database terminology
- Structured Query Language (SQL)

## How to run a command

Commands are issued against databases to take actions against data stores and to include any statement that can be issued against a database. You can use the **OleDbCommand** or the **SqlCommand** classes to get a command to your data store, and **OleDbCommand** can be specific to the data store. This article demonstrates both the **SqlClient** class (to connect to a computer that is running Microsoft SQL Server) and the **OleDb** class (for any database that has an OLE DB or ODBC driver available) within ADO.NET. However, the code is generally the same for both.

With ADO, you can issue commands through the **Command**, the **Connection**, or the **Recordset** object. In ADO.NET, only the **Command** objects (**SqlCommand** or **OleDbCommand**) run commands.

To run a command, follow these steps:

1. Follow these steps to create a new console application in Microsoft Visual C# 2005 or in Microsoft Visual C# .NET:
   a. Start Microsoft Visual Studio 2005 or Microsoft Visual Studio .NET.

   b. On the **File** menu, point to **New**, and then click **Project**.

   c. In the **New Project** dialog box, click **Visual C# Projects** under **Project Types**, and then click **Console Application** under **Templates**.

      **Note** In Visual Studio 2005, click **Visual C#** under **Project Types** in the **New Project** dialog box, and then click **Console Application** under **Templates**.

2. Make sure that your project contains a reference to the **System.Data** namespace, and add a reference if it does not.

3. Use the **using** statement on the **System** and **System.Data** namespaces so that you do not have to qualify declarations in those namespaces later in your code. You can also include **System.Data.SqlClient** or **System.Data.OleDb**, depending on which one you are using.

```
using System;
using System.Data;
using System.Data.SqlClient;
```

4. Before you can create a connection to a database, you must have a connection string. Connection strings contain all of the information that you need to establish a database connection, including the server name, the database name, the user ID, and the password. For example, the following connection string points to a local computer that is running SQL Server:

   For OleDb connections:

   **Note** User ID <UID> must have appropriate permissions to perform these operations on the database.

   ```
   Provider=SQLOLEDB.1;User ID=<UID>;Initial Catalog=pubs;Data Sourc
   e=(local)
   ```

   For SqlClient connections:

   ```
   User ID=<UID>;Initial Catalog=pubs;Data Source=(local)
   ```

   **Note** If you need more assistance determining the connection string for your database, search for "ConnectionString" on the Microsoft Developer Network (MSDN) Library at:

   http://msdn.microsoft.com/en-us/default.aspx

5. Visual Studio creates a static class and an empty **Main()** procedure. Declare a string variable, and store the appropriate connection string for your database in this procedure.

   **Note** User ID <UID> must have appropriate permissions to perform these operations on the database.

```
class Class1
{
        static void Main(string[] args)
        {
                string sConnectionString =
"User ID=<UID>;Initial Catalog=pubs;Data Source=(local)";
        }
}
```

6. Using this connection string, create a new **OleDbConnection** or **SqlConnection** object, and call its **Open** method to establish a connection to your database:

```
SqlConnection objConn = new SqlConnection(sConnectionString);
objConn.Open();
```

7. Create a **SqlCommand** or **OleDbCommand** object, and pass in the command that you want to run and the connection object that you created in the previous step. The following sample code passes in the INSERT statement:

```
string sSQL = "INSERT INTO Employee " +
  "(emp_id, fname, minit, lname, job_id, job_lvl, pub_id, hire_da
te) " +
  "VALUES ('MSD12923F', 'Duncan', 'W', 'Mackenzie', 10, 82,'087
7','2001-01-01')";
SqlCommand objCmd = new SqlCommand(sSQL,objConn);
```

8. After you create the **SqlCommand** or **OleDbCommand** object, you can call the **ExecuteNonQuery** method to run the command that it represents. **ExecuteNonQuery** is designed for commands that do not return any results (such as the DELETE, UPDATE, and INSERT statements). If the statement runs without throwing an exception (see the following code), the command has been executed successfully against the database.

```
objCmd.ExecuteNonQuery();
```

9. Save your project. On the **Debug** menu, click **Start** to run your command against the database.

## How to use parameters

When you run commands against a database (such as the UPDATE, the INSERT, and the DELETE statements or calls to stored procedures), these commands are frequently parameterized. This allows the command to be created one time but executed multiple times with different values that are inserted instead of parameters. Consider the corresponding DELETE statement to the INSERT statement that is used in the previous section:

```
string sSQL = "DELETE FROM Employee WHERE emp_id = @emp_id"
```

The parameter name ("@emp_id") in this DELETE statement represents a parameter than you can replace with different values each time you run the command.

To use parameters with your command, follow these steps:

1. Create your **OleDbConnection** or **SqlConnection** object, as you did in the "How to run a command" section.

2. Replace values with placeholders (for example, "@emp_id" or "@fname") so that your command text uses parameters. See the DELETE statement before these steps for an example.

3. Create your **OleDbCommand** or **SqlCommand** object, and pass in the connection object that you created in the first step and the command text that contains the parameter placeholders.

4. For each parameter, add a parameter object to the command object's parameters collection. For each parameter, you must specify a name and data type.

```
objCmd.Parameters.Add("@emp_id", SqlDbType.Char, 9);
```

5. Stored procedures can have parameters that return values and output parameters. You must also set a value for each input parameter before you can run the query:

```
objCmd.Parameters["@emp_id"].Value = "MSD12923F";
```

6. Run the query as follows:

```
try
{
        objCmd.ExecuteNonQuery();
}
catch (System.Exception e)
{
        Console.WriteLine(e.Message);
}
Console.WriteLine("Record Deleted");
```

## Complete code listing

**Note** You must change User ID =<UID> to the correct values before you run this code. Make sure that <UID> has the appropriate permissions to perform this operation on the database.

```
using System;
using System.Data;
using System.Data.SqlClient;

    /// <summary>
    /// Summary description for Class1.
    /// </summary>
    class Class1
    {
        static void Main(string[] args)
        {
                AddRecord();
                RemoveRecord();
                Pause();
        }

        static void Pause()
        {
                Console.WriteLine("Press Enter To Continue....");
                Console.ReadLine();
        }

        static void AddRecord()
        {
                string sConnectionString = "User ID=<UID>;Initial Cata
log=pubs;Data Source=(local)";
                SqlConnection objConn = new SqlConnection(sConnectionS
tring);
                objConn.Open();
                string sSQL = "INSERT INTO Employee " +
                  "(emp_id, fname, minit, lname, job_id, job_lvl, pub_
id, hire_date) " +
                  "VALUES ('MSD12923F', 'Duncan', 'W', 'Mackenzie', 1
0, 82,'0877','2001-01-01')";
                SqlCommand objCmd = new SqlCommand(sSQL,objConn);
                try
                {
                        objCmd.ExecuteNonQuery();
                        }
                catch (System.Exception e)
                {
                        Console.WriteLine(e.Message);
                }
                Console.WriteLine("Record Added");
        }

        static void RemoveRecord()
        {
                string sConnectionString = "User ID=<UID>;Initial Cata
log=pubs;Data Source=(local)";
                SqlConnection objConn = new SqlConnection(sConnectionS
tring);
                objConn.Open();
                string sSQL = "DELETE FROM Employee WHERE emp_id = @em
p_id";
                SqlCommand objCmd = new SqlCommand(sSQL,objConn);
                objCmd.Parameters.Add("@emp_id", SqlDbType.Char, 9);
```

```
                    objCmd.Parameters["@emp_id"].Value = "MSD12923F";
                    try
                    {
                            objCmd.ExecuteNonQuery();
                    }
                    catch (System.Exception e)
                    {
                            Console.WriteLine(e.Message);
                    }
                    Console.WriteLine("Record Deleted");
            }
    }
```

## REFERENCES

For more information about how to use ADO.NET, database commands, and stored procedures, visit the following Microsoft Web sites:

SQL Server 2000 Stored Procedures
http://msdn2.microsoft.com/en-us/library/aa214299(SQL.80).aspx

"Diving into Data Access," MSDN Voices column
http://msdn2.microsoft.com/en-us/library/ms810295.aspx

ADO.NET for the ADO Programmer
http://msdn2.microsoft.com/en-us/library/ms973217.aspx

MSDN Online .NET Developer Center
http://msdn.microsoft.com/net

For more information, see the following book:

Sharp, John and Jon Jagger. Microsoft Visual C# .NET Step by Step. Microsoft Press, 2003.

For more information, see the following Microsoft Training & Certification course:

2389 Programming with ADO.NET
http://www.microsoft.com/learning/syllabi/en-us/2389Bfinal.mspx

## Properties

Article ID: 306636 - Last Review: 08/17/2007 02:22:07 - Revision: 4.9

Applies to
Microsoft Visual C# .NET 2002 Standard Edition

Microsoft Visual C# 2005

Keywords:
kbhowtomaster kbsqlclient kbsystemdata KB306636

Report a support scam

Disability Answer Desk

Locate Microsoft addresses worldwide

English (Canada)