ADO.NET SqlCommand Class

This class is used to store and execute SQL statement for SQL Server database. It is a sealed class so that cannot be inherited.

SqlCommand Signature

```
public sealed class SqlCommand : System.Data.Common.DbCommand, ICloneable, IDisposable
```

Constructors

This class provides the following constructors.

Constructor	Description
SqlCommand()	It is used to initialize a new instance of the SqlCommand class.
SqlCommand(String)	It is used to initialize a new instance of the SqlCommand class with a string parameter.
SqlCommand(String, SqlConnection)	It is used to initialize a new instance of the SqlCommand class. It takes two parameters, first is query string and second is connection string.
SqlCommand(String, SqlConnection, SqlTransaction)	It is used to initialize a new instance of the SqlCommand class. It takes three parameters query, connection and transaction string respectively.
SqlCommand(String, SqlConnection, SqlTransaction, SqlCommandColumnEncryptionSetting)	It Initializes a new instance of the SqlCommand class with specified command text, connection, transaction, and encryption setting.

Methods

Method	Description
BeginExecuteNonQuery()	It is used to Initiate the asynchronous execution of the SQL statement described by this SqlCommand.
Cancel()	It tries to cancel the execution of a SqlCommand.
Clone()	It creates a new SqlCommand object that is a copy of the current instance.
CreateParameter()	It creates a new instance of a SqlParameter object.
ExecuteReader()	It is used to send the CommandText to the Connection and builds a SqlDataReader.
ExecuteXmlReader()	It is used to send the CommandText to the Connection and builds an XmlReader object.
ExecuteScalar()	It executes the query and returns the first column of the first row in the result set. Additional columns or rows are ignored.

Prepare()	It is used to create a prepared version of the command by using the instance of SQL Server.
ResetCommandTimeout()	It is used to reset the CommandTimeout property to its default value.

Example

In this example, we are creating a SqlCommand instance and executing a SQL statement.

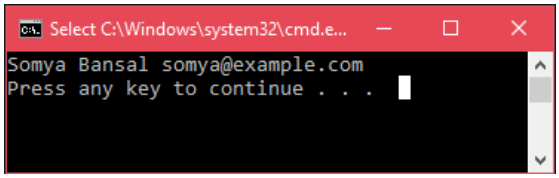
// Program.cs

```
using System;
using System.Data.SqlClient;
namespace AdoNetConsoleApplication
{
   class Program
   {
      static void Main(string[] args)
      {
         new Program().CreateTable();
      }
      public void CreateTable()
      {
         SqlConnection con = null;
         try
         {
            // Creating Connection
            con = new SqlConnection("data source=.; database=student; integrated security=SSPI");
            // writing sql query
            SqlCommand cm = new SqlCommand("select * from student", con);
            // Opening Connection
            con.Open();
            // Executing the SQL query
            SqlDataReader sdr = cm.ExecuteReader();
            while (sdr.Read())
            {
               Console.WriteLine(sdr["name"]+" "+ sdr["email"]);
            }
         }
         catch (Exception e)
         {
            Console.WriteLine("OOPs, something went wrong." + e);
         }
         // Closing the connection
         finally
```

```
        {
            con.Close();
        }
    }
  }
}
```

Output:

Execute this program by combination of **Ctrl+F5** and it will produce the following output.



It prints name and email of the student.

← prev next →

Help Others, Please Share

Learn Latest Tutorials