.NET Framework 4.7.2

Produkt

.NET Core

Version

2.1

2.0

1.1

1.0

.NET Framework

Version

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

Version

2.1

.NET Standard

Version

2.0

1.6

1.5

1.4

1.3

1.2

1.1

1.0

Xamarin.Android

Version

7.1

Xamarin.iOS

Version
10.8
Xamarin.Mac
Version
3.0

Suche

SqlBulkCopy.DestinationTableName Property

Namespace: System.Data.SqlClient

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

In diesem Artikel

Der Name der Zieltabelle auf dem Server.

C# Kopieren

```
public string DestinationTableName { get; set; }
```

Eigenschaftswert

String

Der Zeichenfolgenwert der DestinationTableName-Eigenschaft oder NULL, wenn kein Wert angegeben wurde.

Beispiele

Die folgenden Konsolenanwendung wird veranschaulicht, wie Sie das Massenladen von Daten über eine Verbindung, die bereits geöffnet ist. Die Zieltabelle ist eine Tabelle in der **AdventureWorks** Datenbank.

In diesem Beispiel wird zuerst die Verbindung zum Lesen von Daten aus einer SQL Server-Tabelle, verwendet eine SqlDataReader Instanz. Die Quelldaten muss nicht auf SQL Server befinden; können Sie alle Datenquellen, die zu lesende ein IDataReader oder Laden in ein DataTable.

Wichtig

In diesem Beispiel wird nicht ausgeführt werden, es sei denn, Sie die Arbeitstabellen erstellt haben, wie in beschrieben Einrichtung der Massenkopierbeispiele. Dieser Code wird bereitgestellt, um zu veranschaulichen die Syntax für die Verwendung von "**SqlBulkCopy**" nur. Wenn die Quelle und Ziel-Tabellen in der gleichen SQL Server-Instanz ist, ist es einfacher und schneller mit einer Transact-SQL `INSERT … SELECT` Anweisung, um die Daten zu kopieren.

C# Kopieren

```
using System.Data.SqlClient;

class Program
{
    static void Main()
    {
        string connectionString = GetConnectionString();
        // Open a sourceConnection to the AdventureWorks database.
        using (SqlConnection sourceConnection =
                   new SqlConnection(connectionString))
        {
            sourceConnection.Open();

            // Perform an initial count on the destination table.
            SqlCommand commandRowCount = new SqlCommand(
                "SELECT COUNT(*) FROM " +
                "dbo.BulkCopyDemoMatchingColumns;",
                sourceConnection);
            long countStart = System.Convert.ToInt32(
                commandRowCount.ExecuteScalar());
            Console.WriteLine("Starting row count = {0}", countStart);

            // Get data from the source table as a SqlDataReader.
            SqlCommand commandSourceData = new SqlCommand(
                "SELECT ProductID, Name, " +
                "ProductNumber " +
                "FROM Production.Product;", sourceConnection);
            SqlDataReader reader =
                commandSourceData.ExecuteReader();

            // Open the destination connection. In the real world you would
            // not use SqlBulkCopy to move data from one table to the other
            // in the same database. This is for demonstration purposes only.
            using (SqlConnection destinationConnection =
                       new SqlConnection(connectionString))
            {
                destinationConnection.Open();

                // Set up the bulk copy object.
                // Note that the column positions in the source
                // data reader match the column positions in
                // the destination table so there is no need to
                // map columns.
                using (SqlBulkCopy bulkCopy =
                           new SqlBulkCopy(destinationConnection))
```

```
                {
                    bulkCopy.DestinationTableName =
                        "dbo.BulkCopyDemoMatchingColumns";

                    try
                    {
                        // Write from the source to the destination.
                        bulkCopy.WriteToServer(reader);
                    }
                    catch (Exception ex)
                    {
                        Console.WriteLine(ex.Message);
                    }
                    finally
                    {
                        // Close the SqlDataReader. The SqlBulkCopy
                        // object is automatically closed at the end
                        // of the using block.
                        reader.Close();
                    }
                }

                // Perform a final count on the destination
                // table to see how many rows were added.
                long countEnd = System.Convert.ToInt32(
                    commandRowCount.ExecuteScalar());
                Console.WriteLine("Ending row count = {0}", countEnd);
                Console.WriteLine("{0} rows were added.", countEnd - countStart);
                Console.WriteLine("Press Enter to finish.");
                Console.ReadLine();
            }
        }
    }

    private static string GetConnectionString()
        // To avoid storing the sourceConnection string in your code,
        // you can retrieve it from a configuration file.
    {
        return "Data Source=(local); " +
            " Integrated Security=true;" +
            "Initial Catalog=AdventureWorks;";
    }
}
```

Hinweise

Wenn DestinationTableName nicht festgelegt wurde beim WriteToServer aufgerufen wird, eine ArgumentNullException ausgelöst.

Wenn DestinationTableName wird geändert, während eine WriteToServer Vorgang ausgeführt wird, wird die Änderung wirkt sich nicht auf den aktuellen Vorgang. Die neue DestinationTableName Wert wird verwendet, das nächste Mal eine WriteToServer Methode wird aufgerufen.

DestinationTableName ist ein dreiteiliger Name (`<database>.<owningschema>.<name>`). Falls gewünscht, können Sie den Tabellennamen mit seiner Datenbank und dem besitzenden Schema qualifizieren. Jedoch wenn Sie den Namen der Tabelle einen Unterstrich ("") *oder ein anderes Sonderzeichen verwendet, Sie müssen mit Escapezeichen versehen die Namen, die in Klammern wie in (* `[<database>.<owningschema>.<name_01>]` *).* Weitere Informationen finden Sie unter Datenbankbezeichner.

Können Sie das Massenkopieren von Daten in eine temporäre Tabelle mit einem Wert wie z. B. `tempdb..#table` oder `tempdb.<owner>.#table` für die DestinationTableName Eigenschaft.

Gilt für:

.NET Core

2.1, 2.0, 1.1, 1.0

.NET Framework

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

.NET Platform Extensions

2.1

Xamarin.Android

7.1

Xamarin.iOS

10.8

Xamarin.Mac

3.0

Siehe auch

- Durchführen von Massenkopiervorgängen Performing Bulk Copy Operations
- Übersicht über ADO.NET ADO.NET Overview