.NET Framework 4

**Populating a DataSet from a DataAdapter (ADO.NET)**

The ADO.NET [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) is a memory-resident representation of data that provides a consistent relational programming model independent of the data source. The **DataSet** represents a complete set of data that includes tables, constraints, and relationships among the tables. Because the **DataSet** is independent of the data source, a **DataSet** can include data local to the application, and data from multiple data sources. Interaction with existing data sources is controlled through the **DataAdapter**.

The **SelectCommand** property of the **DataAdapter** is a **Command** object that retrieves data from the data source. The **InsertCommand**, **UpdateCommand**, and **DeleteCommand** properties of the **DataAdapter** are **Command** objects that manage updates to the data in the data source according to modifications made to the data in the **DataSet**. These properties are covered in more detail in [Updating Data Sources with DataAdapters (ADO.NET)](http://msdn.microsoft.com/en-us/library/33y2221y.aspx).

The **Fill** method of the **DataAdapter** is used to populate a **DataSet** with the results of the **SelectCommand** of the **DataAdapter**. **Fill** takes as its arguments a **DataSet** to be populated, and a **DataTable** object, or the name of the **DataTable** to be filled with the rows returned from the **SelectCommand**.

|  |
| --- |
| **NoteNote** |
| Using the **DataAdapter** to retrieve all of a table takes time, especially if there are many rows in the table. This is because accessing the database, locating and processing the data, and then transferring the data to the client is time-consuming. Pulling all of the table to the client also locks all of the rows on the server. To improve performance, you can use the **WHERE** clause to greatly reduce the number of rows returned to the client. You can also reduce the amount of data returned to the client by only explicitly listing required columns in the **SELECT** statement. Another good workaround is to retrieve the rows in batches (such as several hundred rows at a time) and only retrieve the next batch when the client is finished with the current batch. |

The **Fill** method uses the **DataReader** object implicitly to return the column names and types that are used to create the tables in the **DataSet**, and the data to populate the rows of the tables in the **DataSet**. Tables and columns are only created if they do not already exist; otherwise **Fill** uses the existing **DataSet** schema. Column types are created as .NET Framework types according to the tables in [Data Type Mappings in ADO.NET](http://msdn.microsoft.com/en-us/library/4e5xt97a.aspx). Primary keys are not created unless they exist in the data source and **DataAdapter.MissingSchemaAction** is set to **MissingSchemaAction.AddWithKey**. If **Fill** finds that a primary key exists for a table, it will overwrite data in the **DataSet** with data from the data source for rows where the primary key column values match those of the row returned from the data source. If no primary key is found, the data is appended to the tables in the **DataSet**. **Fill** uses any mappings that may exist when you populate the **DataSet** (see [DataAdapter DataTable and DataColumn Mappings (ADO.NET)](http://msdn.microsoft.com/en-us/library/ks92fwwh.aspx)).

|  |
| --- |
| **NoteNote** |
| If the **SelectCommand** returns the results of an OUTER JOIN, the **DataAdapter** does not set a **PrimaryKey** value for the resulting **DataTable**. You must define the **PrimaryKey** yourself to make sure that duplicate rows are resolved correctly. For more information, see [Defining Primary Keys (ADO.NET)](http://msdn.microsoft.com/en-us/library/z24kefs8.aspx). |

The following code example creates an instance of a [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.aspx) that uses a [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) to the Microsoft SQL Server **Northwind** database and populates a [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) in a **DataSet** with the list of customers. The SQL statement and [SqlConnection](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqlconnection.aspx) arguments passed to the [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.aspx) constructor are used to create the [SelectCommand](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.selectcommand.aspx) property of the [SqlDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldataadapter.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Example

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl53_ctl00_ctl00_code');" \o "Copy Code)

' Assumes that connection is a valid SqlConnection object.

Dim queryString As String = \_

"SELECT CustomerID, CompanyName FROM dbo.Customers"

Dim adapter As SqlDataAdapter = New SqlDataAdapter( \_

queryString, connection)

Dim customers As DataSet = New DataSet

adapter.Fill(customers, "Customers")

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl53_ctl00_ctl01_code');" \o "Copy Code)

// Assumes that connection is a valid SqlConnection object.

string queryString =

"SELECT CustomerID, CompanyName FROM dbo.Customers";

SqlDataAdapter adapter = new SqlDataAdapter(queryString, connection);

DataSet customers = new DataSet();

adapter.Fill(customers, "Customers");

|  |
| --- |
| **NoteNote** |
| The code shown in this example does not explicitly open and close the **Connection**. The **Fill** method implicitly opens the **Connection** that the **DataAdapter** is using if it finds that the connection is not already open. If **Fill** opened the connection, it also closes the connection when **Fill** is finished. This can simplify your code when you deal with a single operation such as a **Fill** or an **Update**. However, if you are performing multiple operations that require an open connection, you can improve the performance of your application by explicitly calling the **Open** method of the **Connection**, performing the operations against the data source, and then calling the **Close** method of the **Connection**. You should try to keep connections to the data source open as briefly as possible to free resources for use by other client applications. |

http://i.msdn.microsoft.com/Global/Images/clear.gif Multiple Result Sets

If the **DataAdapter** encounters multiple result sets, it creates multiple tables in the **DataSet**. The tables are given an incremental default name of Table*N*, starting with "Table" for Table0. If a table name is passed as an argument to the **Fill** method, the tables are given an incremental default name of TableName*N*, starting with "TableName" for TableName0.

http://i.msdn.microsoft.com/Global/Images/clear.gif Populating a DataSet from Multiple DataAdapters

Any number of **DataAdapter**objects can be used with a **DataSet**. Each **DataAdapter** can be used to fill one or more **DataTable** objects and resolve updates back to the relevant data source. **DataRelation** and **Constraint** objects can be added to the **DataSet** locally, which enables you to relate data from dissimilar data sources. For example, a **DataSet** can contain data from a Microsoft SQL Server database, an IBM DB2 database exposed through OLE DB, and a data source that streams XML. One or more **DataAdapter** objects can handle communication to each data source.

**Example**

The following code example populates a list of customers from the **Northwind** database on Microsoft SQL Server 2000, and a list of orders from the **Northwind** database stored in Microsoft Access 2000. The filled tables are related with a **DataRelation**, and the list of customers is then displayed with the orders for that customer. For more information about **DataRelation** objects, see [Adding DataRelations (ADO.NET)](http://msdn.microsoft.com/en-us/library/ay82azad.aspx) and [Navigating DataRelations (ADO.NET)](http://msdn.microsoft.com/en-us/library/d6s958d6.aspx).

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl55_ctl00_ctl02_code');" \o "Copy Code)

' Assumes that customerConnection is a valid SqlConnection object.

' Assumes that orderConnection is a valid OleDbConnection object.

Dim custAdapter As SqlDataAdapter = New SqlDataAdapter( \_

"SELECT \* FROM dbo.Customers", customerConnection)

Dim ordAdapter As OleDbDataAdapter = New OleDbDataAdapter( \_

"SELECT \* FROM Orders", orderConnection)

Dim customerOrders As DataSet = New DataSet()

custAdapter.Fill(customerOrders, "Customers")

ordAdapter.Fill(customerOrders, "Orders")

Dim relation As DataRelation = \_

customerOrders.Relations.Add("CustOrders", \_

customerOrders.Tables("Customers").Columns("CustomerID"), \_

customerOrders.Tables("Orders").Columns("CustomerID"))

Dim pRow, cRow As DataRow

For Each pRow In customerOrders.Tables("Customers").Rows

Console.WriteLine(pRow("CustomerID").ToString())

For Each cRow In pRow.GetChildRows(relation)

Console.WriteLine(vbTab & cRow("OrderID").ToString())

Next

Next

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl55_ctl00_ctl03_code');" \o "Copy Code)

// Assumes that customerConnection is a valid SqlConnection object.

// Assumes that orderConnection is a valid OleDbConnection object.

SqlDataAdapter custAdapter = new SqlDataAdapter(

"SELECT \* FROM dbo.Customers", customerConnection);

OleDbDataAdapter ordAdapter = new OleDbDataAdapter(

"SELECT \* FROM Orders", orderConnection);

DataSet customerOrders = new DataSet();

custAdapter.Fill(customerOrders, "Customers");

ordAdapter.Fill(customerOrders, "Orders");

DataRelation relation = customerOrders.Relations.Add("CustOrders",

customerOrders.Tables["Customers"].Columns["CustomerID"],

customerOrders.Tables["Orders"].Columns["CustomerID"]);

foreach (DataRow pRow in customerOrders.Tables["Customers"].Rows)

{

Console.WriteLine(pRow["CustomerID"]);

foreach (DataRow cRow in pRow.GetChildRows(relation))

Console.WriteLine("\t" + cRow["OrderID"]);

}

http://i.msdn.microsoft.com/Global/Images/clear.gif SQL Server Decimal Type

By default, the **DataSet** stores data by using .NET Framework data types. For most applications, these provide a convenient representation of data source information. However, this representation may cause a problem when the data type in the data source is a SQL Server decimal or numeric data type. The .NET Framework **decimal** data type allows a maximum of 28 significant digits, whereas the SQL Server **decimal** data type allows 38 significant digits. If the **SqlDataAdapter** determines during a **Fill** operation that the precision of a SQL Server **decimal** field is larger than 28 characters, the current row is not added to the **DataTable**. Instead the **FillError** event occurs, which enables you to determine whether a loss of precision will occur, and respond appropriately. For more information about the **FillError** event, see [Handling DataAdapter Events (ADO.NET)](http://msdn.microsoft.com/en-us/library/6d1wk41s.aspx). To get the SQL Server **decimal** value, you can also use a [SqlDataReader](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.aspx) object and call the [GetSqlDecimal](http://msdn.microsoft.com/en-us/library/system.data.sqlclient.sqldatareader.getsqldecimal.aspx) method.

ADO.NET 2.0 introduced enhanced support for [System.Data.SqlTypes](http://msdn.microsoft.com/en-us/library/system.data.sqltypes.aspx) in the **DataSet**. For more information, see [SqlTypes and the DataSet (ADO.NET)](http://msdn.microsoft.com/en-us/library/ms172137.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif OLE DB Chapters

Hierarchical rowsets, or chapters (OLE DB type **DBTYPE\_HCHAPTER**, ADO type **adChapter**) can be used to fill the contents of a **DataSet**. When the [OleDbDataAdapter](http://msdn.microsoft.com/en-us/library/system.data.oledb.oledbdataadapter.aspx) encounters a chaptered column during a **Fill** operation, a **DataTable** is created for the chaptered column, and that table is filled with the columns and rows from the chapter. The table created for the chaptered column is named by using both the parent table name and the chaptered column name in the form "*ParentTableNameChapteredColumnName*". If a table already exists in the **DataSet** that matches the name of the chaptered column, the current table is filled with the chapter data. If there is no column in an existing table that matches a column found in the chapter, a new column is added.

Before the tables in the **DataSet** are filled with the data in the chaptered columns, a relation is created between the parent and child tables of the hierarchical rowset by adding an integer column to both the parent and child table, setting the parent column to auto-increment, and creating a **DataRelation** using the added columns from both tables. The added relation is named by using the parent table and chapter column names in the form "*ParentTableNameChapterColumnName*".

Note that the related column only exists in the **DataSet**. Subsequent fills from the data source can cause new rows to be added to the tables instead of changes being merged into existing rows.

Note also that, if you use the **DataAdapter.Fill** overload that takes a **DataTable**, only that table will be filled. An auto-incrementing integer column will still be added to the table, but no child table will be created or filled, and no relation will be created.

The following example uses the MSDataShape Provider to generate a chapter column of orders for each customer in a list of customers. A **DataSet** is then filled with the data.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl01_code');" \o "Copy Code)

Using connection As OleDbConnection = New OleDbConnection( \_

"Provider=MSDataShape;Data Provider=SQLOLEDB;" & \_

"Data Source=(local);Integrated " & \_

"Security=SSPI;Initial Catalog=northwind")

Dim adapter As OleDbDataAdapter = New OleDbDataAdapter( \_

"SHAPE {SELECT CustomerID, CompanyName FROM Customers} " & \_

"APPEND ({SELECT CustomerID, OrderID FROM Orders} AS Orders " & \_

"RELATE CustomerID TO CustomerID)", connection)

Dim customers As DataSet = New DataSet()

adapter.Fill(customers, "Customers")

End Using

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl57_ctl00_ctl02_code');" \o "Copy Code)

using (OleDbConnection connection = new OleDbConnection("Provider=MSDataShape;Data Provider=SQLOLEDB;" +

"Data Source=(local);Integrated Security=SSPI;Initial Catalog=northwind"))

{

OleDbDataAdapter adapter = new OleDbDataAdapter("SHAPE {SELECT CustomerID, CompanyName FROM Customers} " +

"APPEND ({SELECT CustomerID, OrderID FROM Orders} AS Orders " +

"RELATE CustomerID TO CustomerID)", connection);

DataSet customers = new DataSet();

adapter.Fill(customers, "Customers");

}

When the **Fill** operation is complete, the **DataSet** contains two tables: **Customers** and **CustomersOrders**, where **CustomersOrders** represents the chaptered column. An additional column named **Orders** is added to the **Customers** table, and an additional column named CustomersOrders is added to the **CustomersOrders** table. The **Orders** column in the **Customers** table is set to auto-increment. A **DataRelation**, **CustomersOrders**, is created by using the columns that were added to the tables with **Customers** as the parent table. The following tables show some sample results.

**TableName: Customers**

|  |  |  |
| --- | --- | --- |
| **CustomerID** | **CompanyName** | **Orders** |
| ALFKI | Alfreds Futterkiste | 0 |
| ANATR | Ana Trujillo Emparedados y helados | 1 |

**TableName: CustomersOrders**

|  |  |  |
| --- | --- | --- |
| **CustomerID** | **OrderID** | **CustomersOrders** |
| ALFKI | 10643 | 0 |
| ALFKI | 10692 | 0 |
| ANATR | 10308 | 1 |
| ANATR | 10625 | 1 |