.NET Framework 4

**DataSets, DataTables, and DataViews (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 regardless of the source of the data it contains. A [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) represents a complete set of data including the tables that contain, order, and constrain the data, as well as the relationships between the tables.

There are several ways of working with a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), which can be applied independently or in combination. You can:

* Programmatically create a [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx), [DataRelation](http://msdn.microsoft.com/en-us/library/system.data.datarelation.aspx), and [Constraint](http://msdn.microsoft.com/en-us/library/system.data.constraint.aspx) within a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) and populate the tables with data.
* Populate the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) with tables of data from an existing relational data source using a **DataAdapter**.
* Load and persist the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) contents using XML. For more information, see [Using XML in a DataSet (ADO.NET)](http://msdn.microsoft.com/en-us/library/84sxtbxh.aspx).

A strongly typed [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) can also be transported using an XML Web service. The design of the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) makes it ideal for transporting data using XML Web services. For an overview of XML Web services, see [XML Web Services Overview](http://msdn.microsoft.com/en-us/library/w9fdtx28.aspx). For an example of consuming a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) from an XML Web service, see [Consuming a DataSet from an XML Web Service (ADO.NET)](http://msdn.microsoft.com/en-us/library/s5xy331f.aspx).

.NET Framework 4

**Creating a DataSet (ADO.NET)**

You create an instance of a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) by calling the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) constructor. Optionally specify a name argument. If you do not specify a name for the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), the name is set to "NewDataSet".

You can also create a new [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) based on an existing [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx). The new [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) can be an exact copy of the existing [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx); a clone of the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) that copies the relational structure or schema but that does not contain any of the data from the existing [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx); or a subset of the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), containing only the modified rows from the existing [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) using the [GetChanges](http://msdn.microsoft.com/en-us/library/system.data.dataset.getchanges.aspx) method. For more information, see [Copying DataSet Contents (ADO.NET)](http://msdn.microsoft.com/en-us/library/kax0zhe7.aspx).

The following code example demonstrates how to construct an instance of a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx).

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl15_code');" \o "Copy Code)

Dim customerOrders As DataSet = New DataSet("CustomerOrders")

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl16_code');" \o "Copy Code)

DataSet customerOrders = new DataSet("CustomerOrders");

.NET Framework 4

**Adding a DataTable to a DataSet (ADO.NET)**

ADO.NET enables you to create [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) objects and add them to an existing [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx). You can set constraint information for a [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) by using the [PrimaryKey](http://msdn.microsoft.com/en-us/library/system.data.datatable.primarykey.aspx) and [Unique](http://msdn.microsoft.com/en-us/library/system.data.datacolumn.unique.aspx) properties.

 Example

The following example constructs a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), adds a new [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) object to the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx), and then adds three [DataColumn](http://msdn.microsoft.com/en-us/library/system.data.datacolumn.aspx) objects to the table. Finally, the code sets one column as the primary key column.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl04_code');" \o "Copy Code)

Dim customerOrders As DataSet = New DataSet("CustomerOrders")

Dim ordersTable As DataTable = customerOrders.Tables.Add("Orders")

Dim pkOrderID As DataColumn = ordersTable.Columns.Add( \_

"OrderID", Type.GetType("System.Int32"))

ordersTable.Columns.Add("OrderQuantity", Type.GetType("System.Int32"))

ordersTable.Columns.Add("CompanyName", Type.GetType("System.String"))

ordersTable.PrimaryKey = New DataColumn() {pkOrderID}

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl34_ctl00_ctl05_code');" \o "Copy Code)

DataSet customerOrders = new DataSet("CustomerOrders");

DataTable ordersTable = customerOrders.Tables.Add("Orders");

DataColumn pkOrderID =

ordersTable.Columns.Add("OrderID", typeof(Int32));

ordersTable.Columns.Add("OrderQuantity", typeof(Int32));

ordersTable.Columns.Add("CompanyName", typeof(string));

ordersTable.PrimaryKey = new DataColumn[] { pkOrderID };

 Case Sensitivity

Two or more tables or relations with the same name, but different casing, can exist in a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx). In such cases, references by name to tables and relations are case sensitive. For example, if the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) **dataSet** contains tables **Table1** and **table1**, you would reference **Table1** by name as **dataSet.Tables["Table1"]**, and **table1** as **dataSet.Tables["table1"]**. Attempting to reference either of the tables as **dataSet.Tables["TABLE1"]** would generate an exception.

The case-sensitivity behavior does not apply if only one table or relation has a particular name. For example, if the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) has only **Table1**, you can reference it using **dataSet.Tables["TABLE1"]**.

|  |
| --- |
| **NoteNote** |
| The [CaseSensitive](http://msdn.microsoft.com/en-us/library/system.data.dataset.casesensitive.aspx) property of the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) does not affect this behavior. The [CaseSensitive](http://msdn.microsoft.com/en-us/library/system.data.dataset.casesensitive.aspx) property applies to the data in the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) and affects sorting, searching, filtering, enforcing constraints, and so on. |

 Namespace Support

In versions of ADO.NET earlier than 2.0, two tables could not have the same name, even if they were in different namespaces. This limitation was removed in ADO.NET 2.0. A [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) can contain two tables that have the same [TableName](http://msdn.microsoft.com/en-us/library/system.data.datatable.tablename.aspx) property value but different [Namespace](http://msdn.microsoft.com/en-us/library/system.data.datatable.namespace.aspx) property values.

.NET Framework 4

**Adding DataRelations (ADO.NET)**

In a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) with multiple [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) objects, you can use [DataRelation](http://msdn.microsoft.com/en-us/library/system.data.datarelation.aspx) objects to relate one table to another, to navigate through the tables, and to return child or parent rows from a related table.

The arguments required to create a **DataRelation** are a name for the **DataRelation** being created, and an array of one or more [DataColumn](http://msdn.microsoft.com/en-us/library/system.data.datacolumn.aspx) references to the columns that serve as the parent and child columns in the relationship. After you have created a **DataRelation**, you can use it to navigate between tables and to retrieve values.

Adding a **DataRelation** to a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) adds, by default, a [UniqueConstraint](http://msdn.microsoft.com/en-us/library/system.data.uniqueconstraint.aspx) to the parent table and a [ForeignKeyConstraint](http://msdn.microsoft.com/en-us/library/system.data.foreignkeyconstraint.aspx) to the child table. For more information about these default constraints, see [DataTable Constraints (ADO.NET)](http://msdn.microsoft.com/en-us/library/st1t2c35.aspx).

The following code example creates a **DataRelation** using two [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) objects in a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx). Each [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) contains a column named **CustID**, which serves as a link between the two [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) objects. The example adds a single **DataRelation** to the **Relations** collection of the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx). The first argument in the example specifies the name of the **DataRelation** being created. The second argument sets the parent **DataColumn** and the third argument sets the child **DataColumn**.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl14_code');" \o "Copy Code)

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

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

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

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl15_code');" \o "Copy Code)

customerOrders.Relations.Add("CustOrders",

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

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

A **DataRelation** also has a **Nested** property which, when set to **true**, causes the rows from the child table to be nested within the associated row from the parent table when written as XML elements using [WriteXml](http://msdn.microsoft.com/en-us/library/system.data.dataset.writexml.aspx) . For more information, see [Using XML in a DataSet (ADO.NET)](http://msdn.microsoft.com/en-us/library/84sxtbxh.aspx).

.NET Framework 4

**Typed DataSets (ADO.NET)**

Along with late bound access to values through weakly typed variables, the [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) provides access to data through a strongly typed metaphor. Tables and columns that are part of the **DataSet** can be accessed using user-friendly names and strongly typed variables.

A typed **DataSet** is a class that derives from a **DataSet**. As such, it inherits all the methods, events, and properties of a **DataSet**. Additionally, a typed **DataSet** provides strongly typed methods, events, and properties. This means you can access tables and columns by name, instead of using collection-based methods. Aside from the improved readability of the code, a typed **DataSet** also allows the Visual Studio .NET code editor to automatically complete lines as you type.

Additionally, the strongly typed **DataSet** provides access to values as the correct type at compile time. With a strongly typed **DataSet**, type mismatch errors are caught when the code is compiled rather than at run time.

.NET Framework 4

**Generating Strongly Typed DataSets (ADO.NET)**

Given an XML Schema that complies with the XML Schema definition language (XSD) standard, you can generate a strongly typed [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) using the XSD.exe tool provided with the Windows Software Development Kit (SDK).

The following code shows the syntax for generating a **DataSet** using this tool.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl02_code');" \o "Copy Code)

xsd.exe /d /l:CS XSDSchemaFileName.xsd /eld /n:XSDSchema.Namespace

In this syntax, the /d directive tells the tool to generate a **DataSet**, and the /l: tells the tool what language to use (for example, C# or Visual Basic .NET). The optional /eld directive specifies that you can use LINQ to DataSet to query against the generated **DataSet.** This option is used when the /d option is also specified. For more information, see [Querying Typed DataSets](http://msdn.microsoft.com/en-us/library/bb399351.aspx). The optional /n: directive tells the tool to also generate a namespace for the **DataSet** called **XSDSchema.Namespace**. The output of the command is XSDSchemaFileName.cs, which can be compiled and used in an ADO.NET application. The generated code can be compiled as a library or a module.

The following code shows the syntax for compiling the generated code as a library using the C# compiler (csc.exe).

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl04_code');" \o "Copy Code)

csc.exe /t:library XSDSchemaFileName.cs /r:System.dll /r:System.Data.dll

The /t: directive tells the tool to compile to a library, and the /r: directives specify dependent libraries required to compile. The output of the command is XSDSchemaFileName.dll, which can be passed to the compiler when compiling an ADO.NET application with the /r: directive.

The following code shows the syntax for accessing the namespace passed to XSD.exe in an ADO.NET application.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl05_code');" \o "Copy Code)

Imports XSDSchema.Namespace

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl06_code');" \o "Copy Code)

using XSDSchema.Namespace;

The following code example uses a typed **DataSet** named **CustomerDataSet** to load a list of customers from the **Northwind** database. Once the data is loaded using the **Fill** method, the example loops through each customer in the **Customers** table using the typed **CustomersRow** (**DataRow**) object. This provides direct access to the **CustomerID** column, as opposed to through the **DataColumnCollection**.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl07_code');" \o "Copy Code)

Dim customers As CustomerDataSet= New CustomerDataSet()

Dim adapter As SqlDataAdapter New SqlDataAdapter( \_

"SELECT \* FROM dbo.Customers;", \_

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

"Security=SSPI;Initial Catalog=Northwind")

adapter.Fill(customers, "Customers")

Dim customerRow As CustomerDataSet.CustomersRow

For Each customerRow In customers.Customers

Console.WriteLine(customerRow.CustomerID)

Next

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl08_code');" \o "Copy Code)

CustomerDataSet customers = new CustomerDataSet();

SqlDataAdapter adapter = new SqlDataAdapter(

"SELECT \* FROM dbo.Customers;",

"Data Source=(local);Integrated " +

"Security=SSPI;Initial Catalog=Northwind");

adapter.Fill(customers, "Customers");

foreach(CustomerDataSet.CustomersRow customerRow in customers.Customers)

Console.WriteLine(customerRow.CustomerID);

Following is the XML Schema used for the example.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl09_code');" \o "Copy Code)

<?xml version="1.0" encoding="utf-8"?>

<xs:schema id="CustomerDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata">

<xs:element name="CustomerDataSet" msdata:IsDataSet="true">

<xs:complexType>

<xs:choice maxOccurs="unbounded">

<xs:element name="Customers">

<xs:complexType>

<xs:sequence>

<xs:element name="CustomerID" type="xs:string" minOccurs="0" />

</xs:sequence>

</xs:complexType>

</xs:element>

</xs:choice>

</xs:complexType>

</xs:element>

</xs:schema>

.NET Framework 4

**DataTables (ADO.NET)**

A [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) is made up of a collection of tables, relationships, and constraints. In ADO.NET, [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) objects are used to represent the tables in a **DataSet**. A **DataTable** represents one table of in-memory relational data; the data is local to the .NET-based application in which it resides, but can be populated from a data source such as Microsoft SQL Server using a **DataAdapter** For more information, see [Populating a DataSet from a DataAdapter (ADO.NET)](http://msdn.microsoft.com/en-us/library/bh8kx08z.aspx).

The **DataTable** class is a member of the **System.Data** namespace within the .NET Framework class library. You can create and use a **DataTable** independently or as a member of a **DataSet**, and **DataTable** objects can also be used in conjunction with other .NET Framework objects, including the [DataView](http://msdn.microsoft.com/en-us/library/system.data.dataview.aspx). You access the collection of tables in a **DataSet** through the **Tables** property of the **DataSet** object.

The schema, or structure of a table is represented by columns and constraints. You define the schema of a **DataTable** using [DataColumn](http://msdn.microsoft.com/en-us/library/system.data.datacolumn.aspx) objects as well as [ForeignKeyConstraint](http://msdn.microsoft.com/en-us/library/system.data.foreignkeyconstraint.aspx) and [UniqueConstraint](http://msdn.microsoft.com/en-us/library/system.data.uniqueconstraint.aspx) objects. The columns in a table can map to columns in a data source, contain calculated values from expressions, automatically increment their values, or contain primary key values.

In addition to a schema, a **DataTable** must also have rows to contain and order data. The [DataRow](http://msdn.microsoft.com/en-us/library/system.data.datarow.aspx) class represents the actual data contained in a table. You use the **DataRow** and its properties and methods to retrieve, evaluate, and manipulate the data in a table. As you access and change the data within a row, the **DataRow** object maintains both its current and original state.

You can create parent-child relationships between tables using one or more related columns in the tables. You create a relationship between **DataTable** objects using a [DataRelation](http://msdn.microsoft.com/en-us/library/system.data.datarelation.aspx). **DataRelation** objects can then be used to return the related child or parent rows of a particular row. For more information, see [Adding DataRelations (ADO.NET)](http://msdn.microsoft.com/en-us/library/ay82azad.aspx).

.NET Framework 4

**DataTableReaders (ADO.NET)**

The [DataTableReader](http://msdn.microsoft.com/en-us/library/system.data.datatablereader.aspx) presents the contents of a [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx) or a [DataSet](http://msdn.microsoft.com/en-us/library/system.data.dataset.aspx) in the form of one or more read-only, forward-only result sets.

When you create a **DataTableReader** from a **DataTable**, the resulting **DataTableReader** object contains one result set with the same data as the **DataTable** from which it was created, except for any rows that have been marked as deleted. The columns appear in the same order as in the original **DataTable**.

A **DataTableReader** may contain multiple result sets if it was created by calling [CreateDataReader](http://msdn.microsoft.com/en-us/library/system.data.dataset.createdatareader.aspx). The results are in the same order as the **DataTables** in the **DataSet** object's [Tables](http://msdn.microsoft.com/en-us/library/system.data.dataset.tables.aspx) collection.

.NET Framework 4

**DataViews (ADO.NET)**

A [DataView](http://msdn.microsoft.com/en-us/library/system.data.dataview.aspx) enables you to create different views of the data stored in a [DataTable](http://msdn.microsoft.com/en-us/library/system.data.datatable.aspx), a capability that is often used in data-binding applications. Using a **DataView**, you can expose the data in a table with different sort orders, and you can filter the data by row state or based on a filter expression.

A **DataView** provides a dynamic view of data in the underlying **DataTable**: the content, ordering, and membership reflect changes as they occur. This behavior differs from the **Select** method of the **DataTable**, which returns a [DataRow](http://msdn.microsoft.com/en-us/library/system.data.datarow.aspx) array from a table based on a particular filter and/or sort order: thiscontent reflects changes to the underlying table, but its membership and ordering remain static. The dynamic capabilities of the **DataView** make it ideal for data-binding applications.

A **DataView** provides you with a dynamic view of a single set of data, much like a database view, to which you can apply different sorting and filtering criteria. Unlike a database view, however, a **DataView** cannot be treated as a table and cannot provide a view of joined tables. You also cannot exclude columns that exist in the source table, nor can you append columns, such as computational columns, that do not exist in the source table.

You can use a [DataViewManager](http://msdn.microsoft.com/en-us/library/system.data.dataview.dataviewmanager.aspx) to manage view settings for all the tables in a **DataSet**. The **DataViewManager** provides you with a convenient way to manage default view settings for each table. When binding a control to more than one table of a **DataSet**, binding to a **DataViewManager** is the ideal choice.