DataSet Class

Namespace: System.Data

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

Represents an in-memory cache of data.

In this article

C# Copy

```
[System.Serializable]
public class DataSet : System.ComponentModel.MarshalByValueComponent,
System.ComponentModel.IListSource,
System.ComponentModel.ISupportInitializeNotification,
System.Runtime.Serialization.ISerializable,
System.Xml.Serialization.IXmlSerializable
```

Inheritance Object → MarshalByValueComponent → DataSet

Attributes SerializableAttribute

Implements IListSource , ISupportInitialize , ISupportInitializeNotification , ISerializable , IXmlSerializable

Examples

The following example consists of several methods that, combined, create and fill a DataSet from the **Northwind** database.

C# Copy

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

namespace Microsoft.AdoNet.DataSetDemo
{
    class NorthwindDataSet
    {
        static void Main()
        {
            string connectionString = GetConnectionString();
            ConnectToData(connectionString);
        }

        private static void ConnectToData(string connectionString)
        {
            //Create a SqlConnection to the Northwind database.
            using (SqlConnection connection =
                       new SqlConnection(connectionString))
            {
                //Create a SqlDataAdapter for the Suppliers table.
                SqlDataAdapter adapter = new SqlDataAdapter();

                // A table mapping names the DataTable.
                adapter.TableMappings.Add("Table", "Suppliers");

                // Open the connection.
                connection.Open();
                Console.WriteLine("The SqlConnection is open.");

                // Create a SqlCommand to retrieve Suppliers data.
                SqlCommand command = new SqlCommand(
                    "SELECT SupplierID, CompanyName FROM dbo.Suppliers;",
                    connection);
                command.CommandType = CommandType.Text;

                // Set the SqlDataAdapter's SelectCommand.
                adapter.SelectCommand = command;

                // Fill the DataSet.
                DataSet dataSet = new DataSet("Suppliers");
                adapter.Fill(dataSet);

                // Create a second Adapter and Command to get
                // the Products table, a child table of Suppliers.
                SqlDataAdapter productsAdapter = new SqlDataAdapter();
                productsAdapter.TableMappings.Add("Table", "Products");

                SqlCommand productsCommand = new SqlCommand(
                    "SELECT ProductID, SupplierID FROM dbo.Products;",
                    connection);
                productsAdapter.SelectCommand = productsCommand;

                // Fill the DataSet.
                productsAdapter.Fill(dataSet);

                // Close the connection.
                connection.Close();
                Console.WriteLine("The SqlConnection is closed.");

                // Create a DataRelation to link the two tables
                // based on the SupplierID.
                DataColumn parentColumn =
                    dataSet.Tables["Suppliers"].Columns["SupplierID"];
                DataColumn childColumn =
                    dataSet.Tables["Products"].Columns["SupplierID"];
                DataRelation relation =
                    new System.Data.DataRelation("SuppliersProducts",
                    parentColumn, childColumn);
                dataSet.Relations.Add(relation);
                Console.WriteLine(
                    "The {0} DataRelation has been created.",
                    relation.RelationName);
            }
        }

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

Remarks

The DataSet, which is an in-memory cache of data retrieved from a data source, is a major component of the ADO.NET architecture. The DataSet consists of a collection of DataTable objects that you can relate to each other with DataRelation objects. You can also enforce data integrity in the DataSet by using the UniqueConstraint and ForeignKeyConstraint objects. For further details about working with DataSet objects, see DataSets, DataTables, and DataViews.

Whereas DataTable objects contain the data, the DataRelationCollection allows you to navigate though the table hierarchy. The tables are contained in a DataTableCollection accessed through the Tables property. When accessing DataTable objects, note that they are conditionally case sensitive. For example, if one DataTable is named "mydatatable" and another is named "Mydatatable", a string used to search for one of the tables is regarded as case sensitive. However, if "mydatatable" exists and "Mydatatable" does not, the search string is regarded as case insensitive. For more information about working with DataTable objects, see Creating a DataTable.

A DataSet can read and write data and schema as XML documents. The data and schema can then be transported across HTTP and used by any application, on any platform that is XML-enabled. You can save the schema as an XML schema with the WriteXmlSchema method, and both schema and data can be saved using the WriteXml method. To read an XML document that includes both schema and data, use the ReadXml method.

In a typical multiple-tier implementation, the steps for creating and refreshing a DataSet, and in turn, updating the original data are to:

1. Build and fill each DataTable in a DataSet with data from a data source using a DataAdapter.
2. Change the data in individual DataTable objects by adding, updating, or deleting DataRow objects.
3. Invoke the GetChanges method to create a second DataSet that features only the changes to the data.
4. Call the Update method of the DataAdapter, passing the second DataSet as an argument.
5. Invoke the Merge method to merge the changes from the second DataSet into the first.
6. Invoke the AcceptChanges on the DataSet. Alternatively, invoke RejectChanges to cancel the changes.

ⓘ Note

The **DataSet** and **DataTable** objects inherit from **MarshalByValueComponent**, and support the **ISerializable** interface for remoting. These are the only ADO.NET objects that can be remoted.

ⓘ Note

Classes inherited from **DataSet** are not finalized by the garbage collector, because the finalizer has been suppressed in **DataSet**. The derived class can call the

ReRegisterForFinalize method in its constructor to allow the class to be finalized by the garbage collector.

Constructors

DataSet()	Initializes a new instance of the DataSet class.
DataSet(SerializationInfo, StreamingContext)	Initializes a new instance of a DataSet class that has the given serialization information and context.
DataSet(SerializationInfo, StreamingContext, Boolean)	Initializes a new instance of the DataSet class.
DataSet(String)	Initializes a new instance of a DataSet class with the given name.

Properties

CaseSensitive	Gets or sets a value indicating whether string comparisons within DataTable objects are case-sensitive.
Container	Gets the container for the component. (Inherited from MarshalByValueComponent)
DataSetName	Gets or sets the name of the current DataSet.
DefaultViewManager	Gets a custom view of the data contained in the DataSet to allow filtering, searching, and navigating using a custom DataViewManager.
DesignMode	Gets a value indicating whether the component is currently in design mode. (Inherited from MarshalByValueComponent)
EnforceConstraints	Gets or sets a value indicating whether constraint rules are followed when attempting any update operation.
Events	Gets the list of event handlers that are attached to this component. (Inherited from MarshalByValueComponent)
ExtendedProperties	Gets the collection of customized user information associated with the `DataSet`.

HasErrors	Gets a value indicating whether there are errors in any of the DataTable objects within this DataSet.
IsInitialized	Gets a value that indicates whether the DataSet is initialized.
Locale	Gets or sets the locale information used to compare strings within the table.
Namespace	Gets or sets the namespace of the DataSet.
Prefix	Gets or sets an XML prefix that aliases the namespace of the DataSet.
Relations	Gets the collection of relations that link tables and allow navigation from parent tables to child tables.
RemotingFormat	Gets or sets a SerializationFormat for the DataSet used during remoting.
SchemaSerialization Mode	Gets or sets a SchemaSerializationMode for a DataSet.
Site	Gets or sets an ISite for the DataSet.
Tables	Gets the collection of tables contained in the DataSet.

Methods

AcceptChanges()	Commits all the changes made to this DataSet since it was loaded or since the last time AcceptChanges() was called.
BeginInit()	Begins the initialization of a DataSet that is used on a form or used by another component. The initialization occurs at run time.
Clear()	Clears the DataSet of any data by removing all rows in all tables.
Clone()	Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.
Copy()	Copies both the structure and data for this DataSet.
CreateDataReader()	Returns a DataTableReader with one result set per DataTable, in the

	same sequence as the tables appear in the Tables collection.
CreateDataReader(DataTable[])	Returns a DataTableReader with one result set per DataTable.
DetermineSchemaSerializationMode(SerializationInfo, StreamingContext)	Determines the SchemaSerializationMode for a DataSet.
DetermineSchemaSerializationMode(XmlReader)	Determines the SchemaSerializationMode for a DataSet.
Dispose()	Releases all resources used by the MarshalByValueComponent. (Inherited from MarshalByValueComponent)
Dispose(Boolean)	Releases the unmanaged resources used by the MarshalByValueComponent and optionally releases the managed resources. (Inherited from MarshalByValueComponent)
EndInit()	Ends the initialization of a DataSet that is used on a form or used by another component. The initialization occurs at run time.
Equals(Object)	Determines whether the specified object is equal to the current object. (Inherited from Object)
GetChanges()	Gets a copy of the DataSet that contains all changes made to it since it was loaded or since AcceptChanges() was last called.
GetChanges(DataRowState)	Gets a copy of the DataSet containing all changes made to it since it was last loaded, or since AcceptChanges() was called, filtered by DataRowState.
GetDataSetSchema(XmlSchemaSet)	Gets a copy of XmlSchemaSet for the DataSet.
GetHashCode()	Serves as the default hash function. (Inherited from Object)
GetObjectData(SerializationInfo, StreamingContext)	Populates a serialization information object with the data needed to serialize the DataSet.

GetSchemaSerializable()	Returns a serializable XmlSchema instance.
GetSerialization Data(SerializationInfo, StreamingContext)	Deserializes the table data from the binary or XML stream.
GetService(Type)	Gets the implementer of the IServiceProvider. (Inherited from MarshalByValueComponent)
GetType()	Gets the Type of the current instance. (Inherited from Object)
GetXml()	Returns the XML representation of the data stored in the DataSet.
GetXmlSchema()	Returns the XML Schema for the XML representation of the data stored in the DataSet.
HasChanges()	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows.
HasChanges(DataRow State)	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows, filtered by DataRowState.
InferXmlSchema(Stream, String[])	Applies the XML schema from the specified Stream to the DataSet.
InferXmlSchema(String, String[])	Applies the XML schema from the specified file to the DataSet.
InferXmlSchema(Text Reader, String[])	Applies the XML schema from the specified TextReader to the DataSet.
InferXmlSchema(Xml Reader, String[])	Applies the XML schema from the specified XmlReader to the DataSet.
InitializeDerivedDataSet()	Deserialize all of the tables data of the DataSet from the binary or XML stream.
IsBinary Serialized(Serialization Info, StreamingContext)	Inspects the format of the serialized representation of the `DataSet`.
Load(IDataReader, Load	Fills a DataSet with values from a data source using the supplied

Option, DataTable[])	IDataReader, using an array of DataTable instances to supply the schema and namespace information.
Load(IDataReader, Load Option, FillErrorEvent Handler, DataTable[])	Fills a DataSet with values from a data source using the supplied IDataReader, using an array of DataTable instances to supply the schema and namespace information.
Load(IDataReader, Load Option, String[])	Fills a DataSet with values from a data source using the supplied IDataReader, using an array of strings to supply the names for the tables within the `DataSet`.
MemberwiseClone()	Creates a shallow copy of the current Object. (Inherited from Object)
Merge(DataRow[])	Merges an array of DataRow objects into the current DataSet.
Merge(DataRow[], Boolean, MissingSchema Action)	Merges an array of DataRow objects into the current DataSet, preserving or discarding changes in the `DataSet` and handling an incompatible schema according to the given arguments.
Merge(DataSet)	Merges a specified DataSet and its schema into the current `DataSet`.
Merge(DataSet, Boolean)	Merges a specified DataSet and its schema into the current `DataSet`, preserving or discarding any changes in this `DataSet` according to the given argument.
Merge(DataSet, Boolean, MissingSchemaAction)	Merges a specified DataSet and its schema with the current `DataSet`, preserving or discarding changes in the current `DataSet` and handling an incompatible schema according to the given arguments.
Merge(DataTable)	Merges a specified DataTable and its schema into the current DataSet.
Merge(DataTable, Boolean, MissingSchema Action)	Merges a specified DataTable and its schema into the current `DataSet`, preserving or discarding changes in the `DataSet` and handling an incompatible schema according to the given arguments.
OnProperty Changing(Property ChangedEventArgs)	Raises the OnPropertyChanging(PropertyChangedEventArgs) event.
OnRemoveRelation(Data Relation)	Occurs when a DataRelation object is removed from a DataTable.
OnRemoveTable(Data	Occurs when a DataTable is removed from a DataSet.

Table)	
RaiseProperty Changing(String)	Sends a notification that the specified DataSet property is about to change.
ReadXml(Stream)	Reads XML schema and data into the DataSet using the specified Stream.
ReadXml(Stream, Xml ReadMode)	Reads XML schema and data into the DataSet using the specified Stream and XmlReadMode.
ReadXml(String)	Reads XML schema and data into the DataSet using the specified file.
ReadXml(String, XmlRead Mode)	Reads XML schema and data into the DataSet using the specified file and XmlReadMode.
ReadXml(TextReader)	Reads XML schema and data into the DataSet using the specified TextReader.
ReadXml(TextReader, Xml ReadMode)	Reads XML schema and data into the DataSet using the specified TextReader and XmlReadMode.
ReadXml(XmlReader)	Reads XML schema and data into the DataSet using the specified XmlReader.
ReadXml(XmlReader, Xml ReadMode)	Reads XML schema and data into the DataSet using the specified XmlReader and XmlReadMode.
ReadXmlSchema(Stream)	Reads the XML schema from the specified Stream into the DataSet.
ReadXmlSchema(String)	Reads the XML schema from the specified file into the DataSet.
ReadXmlSchema(Text Reader)	Reads the XML schema from the specified TextReader into the DataSet.
ReadXmlSchema(Xml Reader)	Reads the XML schema from the specified XmlReader into the DataSet.
ReadXmlSerializable(Xml Reader)	Ignores attributes and returns an empty DataSet.
RejectChanges()	Rolls back all the changes made to the DataSet since it was created, or since the last time AcceptChanges() was called.

Reset()	Clears all tables and removes all relations, foreign constraints, and tables from the DataSet. Subclasses should override Reset() to restore a DataSet to its original state.
ShouldSerializeRelations()	Gets a value indicating whether Relations property should be persisted.
ShouldSerializeTables()	Gets a value indicating whether Tables property should be persisted.
ToString()	Returns a String containing the name of the Component, if any. This method should not be overridden. (Inherited from MarshalByValueComponent)
WriteXml(Stream)	Writes the current data for the DataSet using the specified Stream.
WriteXml(Stream, XmlWriteMode)	Writes the current data, and optionally the schema, for the DataSet using the specified Stream and XmlWriteMode. To write the schema, set the value for the `mode` parameter to `WriteSchema`.
WriteXml(String)	Writes the current data for the DataSet to the specified file.
WriteXml(String, XmlWriteMode)	Writes the current data, and optionally the schema, for the DataSet to the specified file using the specified XmlWriteMode. To write the schema, set the value for the `mode` parameter to `WriteSchema`.
WriteXml(TextWriter)	Writes the current data for the DataSet using the specified TextWriter.
WriteXml(TextWriter, XmlWriteMode)	Writes the current data, and optionally the schema, for the DataSet using the specified TextWriter and XmlWriteMode. To write the schema, set the value for the `mode` parameter to `WriteSchema`.
WriteXml(XmlWriter)	Writes the current data for the DataSet to the specified XmlWriter.
WriteXml(XmlWriter, XmlWriteMode)	Writes the current data, and optionally the schema, for the DataSet using the specified XmlWriter and XmlWriteMode. To write the schema, set the value for the `mode` parameter to `WriteSchema`.
WriteXmlSchema(Stream)	Writes the DataSet structure as an XML schema to the specified Stream object.
WriteXmlSchema(Stream, Converter<Type,String>)	Writes the DataSet structure as an XML schema to the specified Stream object.
WriteXmlSchema(String)	Writes the DataSet structure as an XML schema to a file.

WriteXmlSchema(String, Converter<Type,String>)	Writes the DataSet structure as an XML schema to a file.
WriteXmlSchema(TextWriter)	Writes the DataSet structure as an XML schema to the specified TextWriter object.
WriteXmlSchema(TextWriter, Converter<Type,String>)	Writes the DataSet structure as an XML schema to the specified TextWriter.
WriteXmlSchema(XmlWriter)	Writes the DataSet structure as an XML schema to an XmlWriter object.
WriteXmlSchema(XmlWriter, Converter<Type,String>)	Writes the DataSet structure as an XML schema to the specified XmlWriter.

Events

Disposed	Adds an event handler to listen to the Disposed event on the component. (Inherited from MarshalByValueComponent)
Initialized	Occurs after the DataSet is initialized.
MergeFailed	Occurs when a target and source DataRow have the same primary key value, and EnforceConstraints is set to true.

Explicit Interface Implementations

IListSource.ContainsListCollection	For a description of this member, see ContainsListCollection.
IListSource.GetList()	For a description of this member, see GetList().
IXmlSerializable.GetSchema()	For a description of this member, see GetSchema().
IXmlSerializable.ReadXml(XmlReader)	For a description of this member, see ReadXml(XmlReader).

IXmlSerializable.WriteXml(XmlWriter)	For a description of this member, see WriteXml(XmlWriter).

Applies to

.NET Core

3.1, 3.0, 2.2, 2.1, 2.0

.NET Framework

4.8, 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 Standard

2.1, 2.0

Xamarin.Android

7.1

Xamarin.iOS

10.8

Xamarin.Mac

3.0

Thread Safety

This type is safe for multithreaded read operations. You must synchronize any write operations.

See also

- Using DataSets in ADO.NET

Is this page helpful?

Yes No