



ADO.NET DataTable

DataTable represents relational data into tabular form. ADO.NET provides a DataTable class to create and use data table independently. It can also be used with DataSet also. Initially, when we create DataTable, it does not have table schema. We can create table schema by adding columns and constraints to the table. After defining table schema, we can add rows to the table.

We must include **System.Data** namespace before creating DataTable.

DataTable Class Signature

```
public class DataTable : System.ComponentModel.MarshalByValueComponent, System.ComponentModel.IListSource,
System.ComponentModel.ISupportInitializeNotification, System.Runtime.Serialization.ISerializable,
System.Xml.Serialization.IXmlSerializable
```

DataTable Constructors

The following table contains the DataTable class constructors.

Constructors	Description
DataTable()	It is used to initialize a new instance of the DataTable class with no arguments.
DataTable(String)	It is used to initialize a new instance of the DataTable class with the specified table name.
DataTable(SerializationInfo, StreamingContext)	It is used to initialize a new instance of the DataTable class with the SerializationInfo and the StreamingContext.
DataTable(String, String)	It is used to initialize a new instance of the DataTable class using the specified table name and namespace.

DataTable Properties

The following table contains the DataTable class properties.

Property	Description
Columns	It is used to get the collection of columns that belong to this table.
Constraints	It is used to get the collection of constraints maintained by this table.
DataSet	It is used to get the DataSet to which this table belongs.
DefaultView	It is used to get a customized view of the table that may include a filtered view.
HasErrors	It is used to get a value indicating whether there are errors in any of the rows in the table of the DataSet.
MinimumCapacity	It is used to get or set the initial starting size for this table. ↑

PrimaryKey	It is used to get or set an array of columns that function as primary keys for the data table.
Rows	It is used to get the collection of rows that belong to this table.
TableName	It is used to get or set the name of the DataTable.

DataTable Methods

The following table contains the DataTable class methods.

Method	Description
AcceptChanges()	It is used to commit all the changes made to this table.
Clear()	It is used to clear the DataTable of all data.
Clone()	It is used to clone the structure of the DataTable.
Copy()	It is used to copy both the structure and data of the DataTable.
CreateDataReader()	It is used to returns a DataTableReader corresponding to the data within this DataTable.
CreateInstance()	It is used to create a new instance of DataTable.
GetRowType()	It is used to get the row type.
GetSchema()	It is used to get schema of the table.
ImportRow(DataRow)	It is used to copy a DataRow into a DataTable.
Load(IDataReader)	It is used to fill a DataTable with values from a data source using the supplied IDataReader.
Merge(DataTable, Boolean)	It is used to merge the specified DataTable with the current DataTable.
NewRow()	It is used to create a new DataRow with the same schema as the table.
Select()	It is used to get an array of all DataRow objects.
WriteXml(String)	It is used to write the current contents of the DataTable as XML using the specified file.



DataTable Example

Here, in the following example, we are creating a data table that populates data to the browser. This example contains the following files.

// DataTableForm.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DataTableForm.aspx.cs"
Inherits="DataTableDemo.DataTableForm" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
```



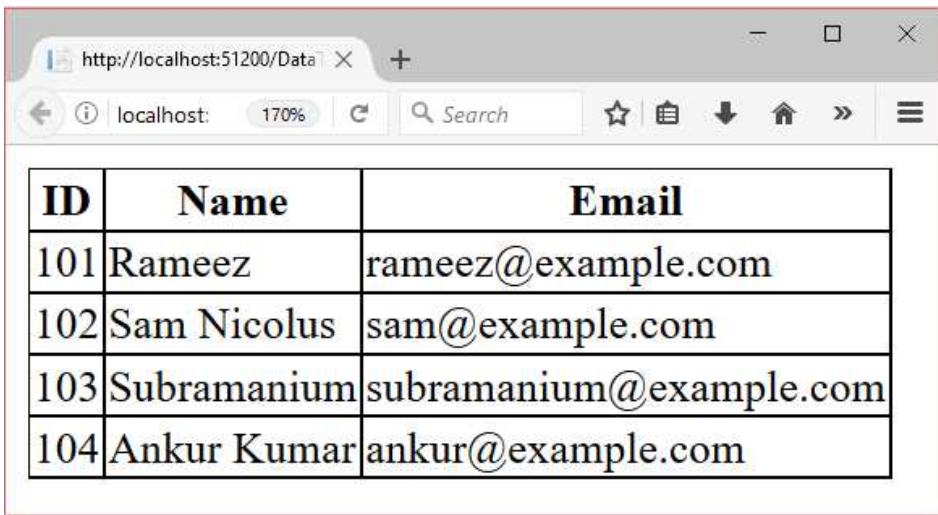
```
<div>
</div>
<asp:GridView ID="GridView1" runat="server">
</asp:GridView>
</form>
</body>
</html>
```

CodeBehind

// DataTableForm.aspx.cs

```
using System;
using System.Collections.Generic;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
namespace DataTableDemo
{
    public partial class DataTableForm : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            DataTable table = new DataTable();
            table.Columns.Add("ID");
            table.Columns.Add("Name");
            table.Columns.Add("Email");
            table.Rows.Add("101", "Rameez", "rameez@example.com");
            table.Rows.Add("102", "Sam Nicolus", "sam@example.com");
            table.Rows.Add("103", "Subramanium", "subramanium@example.com");
            table.Rows.Add("104", "Ankur Kumar", "ankur@example.com");
            GridView1.DataSource = table;
            GridView1.DataBind();
        }
    }
}
```

Output:



ID	Name	Email
101	Rameez	rameez@example.com
102	Sam Niculus	sam@example.com
103	Subramanium	subramanium@example.com
104	Ankur Kumar	ankur@example.com

C# Public Access Specifier Example

```
using System;

namespace AccessSpecifiers
{
    class PublicTest
    {
        public string name = "Santosh Singh";
        public void Msg(string msg)
        {
            Console.WriteLine("Hello "+ msg);
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            PublicTest publicTest = new PublicTest();
            // Accessing public variable
            Console.WriteLine("Hello "+publicTest.name);
            // Accessing public method
            publicTest.Msg("Peter Dicosta");
        }
    }
}
```

Output:

```
Hello Santosh Singh
Hello Peter Dicosta
```




--	--	--	--	--	--


Please Share




Learn Latest Tutorials




Selenium




Control System




Rust



IntelliJ



DBMS



Aptitude

