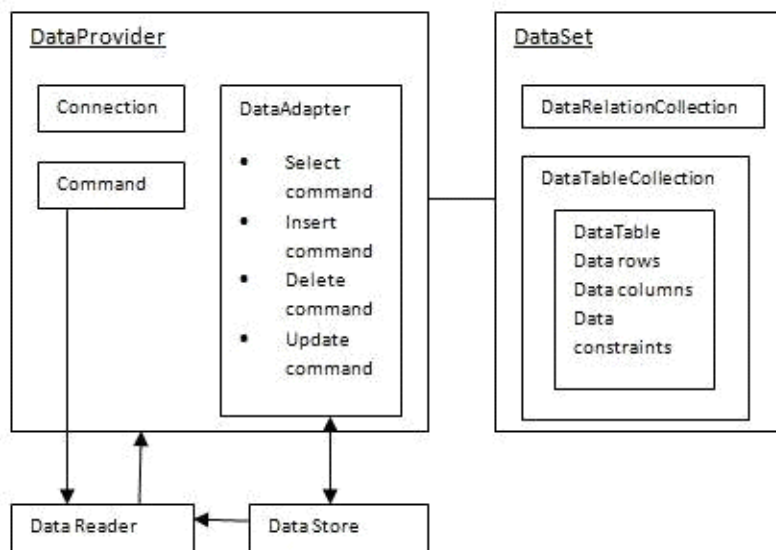


ADO.NET

ADO.NET provides a bridge between the front end controls and the back end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

The following figure shows the ADO.NET objects at a glance:



The DataSet Class

The dataset represents a subset of the database. It does not have a continuous connection to the database. To update the database a reconnection is required. The **DataSet** contains **DataTable** objects and **DataRelation** objects. The **DataRelation** objects represent the relationship between two tables.

Following table shows some important properties of the DataSet class:

Properties	Description
CaseSensitive	Indicates whether string comparisons within the data tables are case-sensitive.
Container	Gets the container for the component.
DataSetName	Gets or sets the name of the current data set.
DefaultViewManager	Returns a view of data in the data set.
DesignMode	Indicates whether the component is currently in design mode.
EnforceConstraints	Indicates whether constraint rules are followed when attempting any update operation.

Events	Gets the list of event handlers that are attached to this component.
ExtendedProperties	Gets the collection of customized user information associated with the DataSet.
HasErrors	Indicates if there are any errors.
IsInitialized	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	Returns the collection of DataRelation objects.

Tables	Returns the collection of DataTable objects.
--------	--

The following table shows some important methods of the DataSet class:

Methods	Description
AcceptChanges	Accepts all changes made since the DataSet was loaded or this method was called.
BeginInit	Begins the initialization of the DataSet. The initialization occurs at run time.
Clear	Clears data.
Clone	Copies the structure of the DataSet, including all DataTable schemas, relations, and constraints. Does not copy any data.

Copy	Copies both structure and data.
CreateDataReader()	Returns a DataReader with one result set per DataTable, in the same sequence as the tables appear in the Tables collection.
CreateDataReader(DataTable[])	Returns a DataReader with one result set per DataTable.
EndInit	Ends the initialization of the data set.
Equals(Object)	Determines whether the specified Object is equal to the current Object.
Finalize	Free resources and perform other

	cleanups.
GetChanges	Returns a copy of the DataSet with all changes made since it was loaded or the AcceptChanges method was called.
GetChanges(DataRowState)	Gets a copy of DataSet with all changes made since it was loaded or the AcceptChanges method was called, filtered by DataRowState.
GetDataSetSchema	Gets a copy of XmlSchemaSet for the DataSet.
GetObjectData	Populates a serialization information object with the data needed to serialize

	the DataSet.
GetType	Gets the type of the current instance.
GetXML	Returns the XML representation of the data.
GetXMLSchema	Returns the XSD schema for the XML representation of the data.
HasChanges()	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows.
HasChanges(DataRowState)	Gets a value indicating whether the DataSet has changes, including new, deleted, or modified rows, filtered by

	DataRowState.
IsBinarySerialized	Inspects the format of the serialized representation of the DataSet.
Load(IDataReader, LoadOption, 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, LoadOption, 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.

Merge()	Merges the data with data from another DataSet. This method has different overloaded forms.
ReadXML()	Reads an XML schema and data into the DataSet. This method has different overloaded forms.
ReadXMLSchema(0)	Reads an XML schema into the DataSet. This method has different overloaded forms.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.
WriteXML()	Writes an XML schema and data from the DataSet. This method

	has different overloaded forms.
WriteXMLSchema()	Writes the structure of the DataSet as an XML schema. This method has different overloaded forms.

The DataTable Class

The DataTable class represents the tables in the database. It has the following important properties; most of these properties are read only properties except the PrimaryKey property:

Properties	Description
ChildRelations	Returns the collection of child relationship.
Columns	Returns the Columns collection.
Constraints	Returns the Constraints collection.
DataSet	Returns the parent DataSet.
DefaultView	Returns a view of the table.

ParentRelations	Returns the ParentRelations collection.
PrimaryKey	Gets or sets an array of columns as the primary key for the table.
Rows	Returns the Rows collection.

The following table shows some important methods of the DataTable class:

Methods	Description
AcceptChanges	Commits all changes since the last AcceptChanges.
Clear	Clears all data from the table.
GetChanges	Returns a copy of the DataTable with all changes made since the AcceptChanges method was called.
GetErrors	Returns an array of rows with errors.
ImportRows	Copies a new row into the table.
LoadDataRow	Finds and updates a specific row, or creates a new one, if not found any.

Merge	Merges the table with another DataTable.
NewRow	Creates a new DataRow.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.
Reset	Resets the table to its original state.
Select	Returns an array of DataRow objects.

The DataRow Class

The DataRow object represents a row in a table. It has the following important properties:

Properties	Description
HasErrors	Indicates if there are any errors.
Items	Gets or sets the data stored in a specific column.
ItemArrays	Gets or sets all the values for the row.
Table	Returns the parent table.

The following table shows some important methods of the DataRow class:

Methods	Description
AcceptChanges	Accepts all changes made since this method was called.
BeginEdit	Begins edit operation.
CancelEdit	Cancels edit operation.
Delete	Deletes the DataRow.
EndEdit	Ends the edit operation.
GetChildRows	Gets the child rows of this row.
GetParentRow	Gets the parent row.
GetParentRows	Gets parent rows of DataRow object.
RejectChanges	Rolls back all changes made since the last call to AcceptChanges.

The DataAdapter Object

The DataAdapter object acts as a mediator between the DataSet object and the database. This helps the Dataset to contain data from multiple databases or other data source.

The DataReader Object

The DataReader object is an alternative to the DataSet and DataAdapter combination. This object provides a connection oriented access to the data records in the database. These objects are suitable for read-only access, such as populating a list and then breaking the connection.

DbCommand and DbConnection Objects

The DbConnection object represents a connection to the data source. The connection could be shared among different command objects.

The DbCommand object represents the command or a stored procedure sent to the database from retrieving or manipulating data.

Example

So far, we have used tables and databases already existing in our computer. In this example, we will create a table, add column, rows and data into it and display the table using a GridView object.

The source file code is as given:

```
<%@ Page Language="C#" AutoEventWireup="true"  
CodeBehind="Default.aspx.cs"  
Inherits="createdatabase._Default" %>
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0  
Transitional//EN"
```

"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

<head runat="server">

<title>

Untitled Page

</title>

</head>

<body>

<form id="form1" runat="server">

<div>

<asp:GridView ID="GridView1" runat="server">

</asp:GridView>

</div>

</form>

```
</body>
```

```
</html>
```

The code behind file is as given:

```
namespace createdatabase
```

```
{
```

```
    public partial class _Default : System.Web.UI.Page
```

```
    {
```

```
        protected void Page_Load(object sender, EventArgs  
e)
```

```
        {
```

```
            if (!IsPostBack)
```

```
            {
```

```
                DataSet ds = CreateDataSet();
```

```
                GridView1.DataSource = ds.Tables["Student"];
```

```
                GridView1.DataBind();
```

```
            }
```

```
        }
```

```
        private DataSet CreateDataSet()
```



```
{  
    //creating a DataSet object for tables  
    DataSet dataset = new DataSet();  
  
    // creating the student table  
    DataTable Students = CreateStudentTable();  
    dataset.Tables.Add(Students);  
    return dataset;  
}
```

```
private DataTable CreateStudentTable()  
{  
    DataTable Students = new DataTable("Student");  
  
    // adding columns  
    AddNewColumn(Students, "System.Int32",  
"StudentID");  
    AddNewColumn(Students, "System.String",  
"StudentName");  
    AddNewColumn(Students, "System.String",  
"StudentCity");  
}
```

```
// adding rows

AddNewRow(Students, 1, "M H Kabir", "Kolkata");

AddNewRow(Students, 1, "Shreya Sharma",
"Delhi");

AddNewRow(Students, 1, "Rini Mukherjee",
"Hyderabad");

AddNewRow(Students, 1, "Sunil Dubey",
"Bikaner");

AddNewRow(Students, 1, "Rajat Mishra", "Patna");

return Students;
}
```

```
private void AddNewColumn(DataTable table, string
columnType, string columnName)
{
    DataColumn column =
table.Columns.Add(columnName,
Type.GetType(columnType));
}
```

//adding data into the table

```
private void AddNewRow(DataTable table, int id,
string name, string city)
{
    DataRow newrow = table.NewRow();
    newrow["StudentID"] = id;
    newrow["StudentName"] = name;
    newrow["StudentCity"] = city;
    table.Rows.Add(newrow);
}
}
```

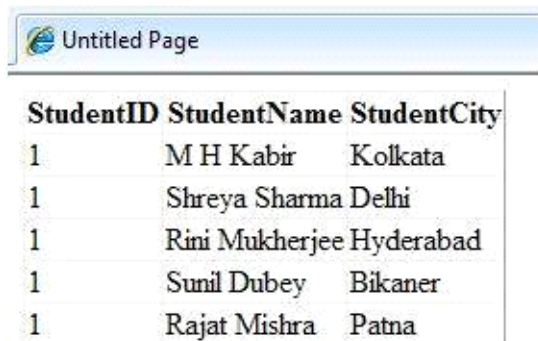
When you execute the program, observe the following:

The application first creates a data set and binds it with the grid view control using the DataBind() method of the GridView control.

The Createdataset() method is a user defined function, which creates a new DataSet object and then calls another user defined method CreateStudentTable() to create the table and add it to the Tables collection of the data set.

The CreateStudentTable() method calls the user defined methods AddNewColumn() and AddNewRow() to create the columns and rows of the table as well as to add data to the rows.

When the page is executed, it returns the rows of the table as shown:

A screenshot of a web browser window titled "Untitled Page". Inside the browser, there is a table with three columns: "StudentID", "StudentName", and "StudentCity". The table contains five rows of data.

StudentID	StudentName	StudentCity
1	M H Kabir	Kolkata
1	Shreya Sharma	Delhi
1	Rini Mukherjee	Hyderabad
1	Sunil Dubey	Bikaner
1	Rajat Mishra	Patna

ASP.NET OLEDB Connection

The ASP.NET OleDbConnection instance takes Connection String as argument and pass the value to the Constructor statement. An instance of the ASP.NET OleDbConnection class is supported the OLEDB Data

VB.Net

Dim connectionString As String

Dim connection As OleDbConnection

***connectionString =
ConfigurationManager.ConnectionStrings("OLEDbC
onnection").ToString***

***connection = New
OleDbConnection(connectionString)***

When the connection is established between ASP.NET application and the specified Data Source, SQL Commands will execute with the help of the Command Object and retrieve or manipulate data in the database. Once the Database activities is over Connection should be closed and release from the data source resources .

The Close() method in the OleDbConnection class is used to close the Database Connection. The Close method Rolls Back any pending transactions and releases the Connection from the Database connected by the OLEDB Data Provider.

Copy and paste the following content into your web.config file and provide the parameter values.

How to ADO.NET OleDbDataReader

OleDbDataReader Object provides a [connection](#) oriented data access to the [OLEDB](#) Data Sources. ExecuteReader() in the OleDbCommand Object send the SQL statements

to OleDbConnection Object and populate an OleDbDataReader Object based on the SQL statement.

```
Dim oledbReader As OleDbDataReader =  
oledbCmd.ExecuteReader()
```

When the ExecuteReader method in OleDbCommand Object execute , it instantiate an OleDb.OleDbDataReader Object. When started to read from an OleDbDataReader it should always be open and positioned prior to the first record. The Read() method in the OleDbDataReader is used to read the rows from OleDbDataReader and it always moves forward to a new valid row, if any row exist .

```
OleDbDataReader.Read()
```

What is OleDbDataAdapter

OleDbDataAdapter is a part of the ADO.NET [Data Provider](#) and it resides in the System.Data.OleDb namespace. OleDbDataAdapter provides the communication between the Dataset and the OleDb Data Sources. We can use OleDbDataAdapter Object in combination with Dataset Object.

The OleDbDataAdapter Object and DataSet objects are combine to perform both Data Access and Data Manipulation operations in the [OleDb](#) Data Sources.

When the user perform the SQL operations like Select , Insert etc. in the data containing in the DataSet Object , it won't directly affect the Database, until the user invoke the Update method in the OleDbDataAdapter.

VB.NET DataSet Examples

Use the DataSet type to store multiple DataTables together.

DataSet stores many DataTables in VB.NET programs. A DataSet is conceptually a set of DataTables and other information about those tables. It is an abstraction that makes programs simpler to develop.

Info: This is a container for multiple DataTables. You can use it to create XML. It is a useful abstraction for simplifying programs.

DataTable.

Data is often stored in tables. And these tables often reside in databases. In the .NET Framework, the DataTable type stores data in memory.

Type details. Used often in VB.NET programs, DataTable has columns and rows properties. It is an in-memory

representation of structured data (like data read from a database).

First example. We define GetTable—this returns a new DataTable. When the GetTable function is invoked, it creates a new DataTable and adds 4 columns to it.

Columns: These are named with a string argument and a Type argument. They have different types.

GetTable: In a DataTable, each column allows a specific type of data. The GetTable method adds 5 rows to the DataTable.

Tip: The arguments to the Rows.Add method are of the types specified in the columns.

VB.NET program that creates DataTable Module
Module1 Sub Main() ' Get a DataTable instance from
helper function. Dim table As **DataTable** = GetTable()
End Sub ''' <summary> ''' Helper function that creates
new DataTable. ''' </summary> Function GetTable() As
DataTable ' Create new DataTable instance. Dim table As
New **DataTable** ' Create four typed columns in the
DataTable. table.Columns.Add("Dosage",
GetType(Integer)) table.Columns.Add("Drug",


```
GetType(String)) table.Columns.Add("Patient",  
GetType(String)) table.Columns.Add("Date",  
GetType(DateTime)) ' Add five rows with those columns  
filled in the DataTable. table.Rows.Add(25, "Indocin",  
"David", DateTime.Now) table.Rows.Add(50, "Enebre",  
"Sam", DateTime.Now) table.Rows.Add(10,  
"Hydralazine", "Christoff", DateTime.Now)  
table.Rows.Add(21, "Combivent", "Janet",  
DateTime.Now) table.Rows.Add(100, "Dilantin",  
"Melanie", DateTime.Now) Return table End Function End  
Module
```

Rows example. Please paste in the GetTable method to run this program. In the For-Each loop, we loop over each Row. With Field(), we print the first Integer cell in each row.

Generic method: Field, part of DataRow, is a generic method. So we must specify the Of Integer part to indicate its parametric type.

DataRow.

A DataRow contains an individual row of data. It narrows the data abstraction to the level of the row. The DataRow

type provides ways to add, remove, or read cells from the enclosing data structure.

Example. As we begin, please notice that the examples that follow invoke a function called GetTable. This function returns a fully formed DataTable instance. It has four columns and five rows, for a total of 20 cells.**DataTable**

Tip: You can paste this function into the following examples to achieve compilation.

VB.NET program that creates DataTable instance

```
Module Module1
Function GetTable() As DataTable '
Generate a new DataTable. ' ... Add columns.
Dim table As DataTable = New DataTable
table.Columns.Add("Weight", GetType(Integer))
table.Columns.Add("Name", GetType(String))
table.Columns.Add("Breed", GetType(String))
table.Columns.Add("Date", GetType(DateTime)) ' ... Add
rows.
table.Rows.Add(57, "Koko", "Shar Pei",
DateTime.Now())
table.Rows.Add(130, "Fido",
"Bullmastiff", DateTime.Now())
table.Rows.Add(92, "Alex", "Anatolian Shepherd Dog", DateTime.Now())
table.Rows.Add(25, "Charles", "Cavalier Kind Charles
Spaniel", DateTime.Now())
table.Rows.Add(7, "Candy", "Yorkshire Terrier", DateTime.Now())
Return table
End
```

```
Function Sub Main() ' Acquire the DataTable instance.  
Dim table As DataTable = GetTable() End Sub End  
Module
```

DataGridView Class

Definition

Namespace:

[System.Windows.Forms](#)

Assembly:

System.Windows.Forms.dll

Displays data in a customizable grid.

C#Copy

```
[System.ComponentModel.ComplexBindingProperties("Da  
taSource", "DataMember")]
```

```
[System.Runtime.InteropServices.ComVisible(true)]
```

```
[System.Windows.Forms.Docking(System.Windows.Form  
s.DockingBehavior.Ask)]
```

```
[System.Runtime.InteropServices.ClassInterface(System.  
Runtime.InteropServices.ClassInterfaceType.AutoDispatc  
h)]
```

```
public class DataGridView :
```

```
System.Windows.Forms.Control,
```

```
System.ComponentModel.ISupportInitialize
```

Inheritance

[Object](#)

[MarshalByRefObject](#)

[Component](#)

[Control](#)

DataGridView

Attributes

[ComplexBindingPropertiesAttribute](#) [ComVisibleAttribute](#)
[DockingAttribute](#) [ClassInterfaceAttribute](#)

Implements

[ISupportInitialize](#)